I N A U G U R A L – D I S S E R T A T I O N

eingereicht bei der

Naturwissenschaftlich–Mathematischen Gesamtfakultät

der

R U P R E C H T – K A R L S – U N I V E R S I T Ä T

H E I D E L B E R G

zur Erlangung der

Doktorwürde der Naturwissenschaften

vorgelegt von

Dipl.–Math. Michael Jung

aus Heidelberg

Tag der mündlichen Prüfung

.....

Relaxations and Approximations for Mixed-Integer Optimal Control

Eingereicht am: 25.10.2013
1. Betreuer: Prof. Dr. Sebastian Sager, Universität Magdeburg
2. Betreuer: Prof. Dr. Gerhard Reinelt, Universität Heidelberg

Zusammenfassung

Diese Arbeit behandelt verschiedene Aspekte der Klasse gemischt-ganzzahliger nichtlinearer Optimalsteuerungsprobleme (MIOCPs). Dies sind Optimierungsprobleme, die die Problematik zugrundeliegender dynamischer Prozesse mit kombinatorischen Entscheidungen verbinden. Typischerweise sind diese Entscheidungen Schaltentscheidungen zwischen den verschiedenen Operationsmodi des dynamischen Systems.
In den letzten Jahrzehnten haben sich direkte Methoden als die Löser von MIOCPs durchgesetzt. Die Formulierung einer gültigen, engen und verlässlichen Relaxierung der Ganzzahligkeit, also die Formulierung eines Modells für gebrochene Werte, spielt eine wichtige Rolle bei diesen direkten Lösungsmethoden. Wir geben ausführlichen Einblick in verschiedene Vorgehen zur Relaxierung von MIOCPs und vergleichen sie im Hinblick auf ihre zugehörigen Strukturen. Insbesondere sind diese die Lösungsstrukturen und Eigenschaften wie Konvexität, Problemgröße und numerisches Verhalten. Aus diesen strukturellen Eigenschaften folgen direkt einige nötige Spezifikationen eines Lösers. Zusätzlich wird das für diese Klasse typische Problem von häufigem Springen zwischen verschiedenen Systemmodi durch die Modellierung und anschließende Beschränkung des Umschaltprozesses direkt angegangen.
Eine der Relaxierungen für MIOCPs ist die äußere Konvexifizierung bei der die Binärvariablen nur affin eingehen. Für diese Relaxierung greifen wir das von Sager als Teil eines Zerlegungsansatzes für MIOCPs mit affinen ganzzahligen Steuerungen entwickelte Steuerungsapproximationsproblem im integralen Sinne auf. Es beschreibt die optimale Approximation von fraktionellen Steuerungen durch ganzzahlige Steuerungen, so dass der zugehörige dynamische Prozess möglichst wenig verändert wird. Wir entwickeln eine neue Heuristik für das mehrdimensionale Problem, die zum ersten mal eine Schranke liefert, die nur von dem Steuerungsgitter und nicht mehr von der Anzahl der Steuerungen abhängt. Für eine Verallgemeinerung des Steuerungsapproximationsproblems durch zusätzliche Beschränkungen leiten wir einen maßgeschneiderten Branch-and-Bound-Algorithmus her, dem die Eigenschaften der Lagrange-Relaxierung des eindimensionalen Problems zugrunde liegen. Dieser Algorithmus schlägt moderne, kommerzielle Löser für gemischt-ganzzahlige lineare Programme (MILPs) für dieses spezielle Approximationsproblem um mehrere Größenordnungen.
Insgesamt stellen wir diverse, teilweise neue Modellierungsansätze für MIOCPs mit den sich ergebenden strukturellen Eigenschaften vor. Darauf aufbauend entwickeln wir neue Theorien zur Approximation von bestimmten relaxierten Lösungen. Wir gehen auch auf die effiziente Implementierung der sich daraus ergebenden strukturausnutzenden Algorithmen ein. Dadurch wird ein tieferes und besseres Verständnis von MIOCPs entwickelt. Die Praktikabilität der theoretischen Beobachtungen zeigen wir anhand von vier prototypischen Anwendungen. Die vorgestellten Modellierungen und Algorithmen ermöglichen auf ihrer Basis die direkte Entwicklung von Decision Support und Analyse Tools für die Praxis.

Abstract

This thesis treats different aspects of the class of Mixed-Integer Optimal Control Problems (MIOCPs). These are optimization problems that combine the difficulties of underlying dynamic processes with combinatorial decisions. Typically, these combinatorial decisions are realized as switching decisions between the system's different operations modes.
During the last decades, direct methods emerged as the state-of-the-art solvers for MIOCPs. The formulation of a valid, tight and dependable integral relaxation, i.e., the formulation of a model for fractional values, plays an important role for these direct solution methods. We give detailed insight into several relaxation approaches for MIOCPs and compare them with regard to their respective structures. In particular, these are the typical solution's structures and properties as convexity, problem size and numerical behavior. From these structural properties, we deduce some required specifications of a solver. Additionally, the modeling and subsequent limitation of the switching process directly tackle the class-specific typical issue of chattering solutions.
One of the relaxation methods for MIOCPs is the outer convexification, where the binary variables only enter affinely. For the approximation of this relaxation's solution, we took up on the control approximation problem in integral sense derived by Sager as part of a decomposition approach for MIOCPs with affine binary controls. This problem describes the optimal approximation of fractional controls with binary controls such that the corresponding dynamic process is changed as little as possible. For the multi-dimensional problem, we developed a new heuristic, which for the first time gives a bound that only depends on the control grid and not anymore on the number of the system's controls. For the generalization of the control approximation problem with additional constraints, we derived a tailored branch-and-bound algorithm, which is based on the properties of the Lagrangian relaxation of the one-dimensional problem. This algorithm beats state-of-the-art commercial solvers for Mixed-Integer Linear Programs (MILPs) for this special approximation problem by several orders of magnitude.
Overall, we present several, partially new modeling approaches for MIOCPs together with the accompanying structural properties. On this basis, we develop new theories for the approximation of certain relaxed solutions. We discuss the efficient implementation of the resulting structure exploiting algorithms. This leads to a deeper and better understanding of MIOCPs. We show the practicability of the theoretical observations with the help of four prototypical problems. The presented methods and algorithms allow on their basis the direct development of decision support and analysis tools in practice.

Danksagung

Mein aufrichtiger Dank gilt meinen Mentoren *Professor Dr. Sebastian Sager*, *Professor Dr. Gerhard Reinelt*, und *Dr. Christian Kirches* für die umfassende Unterstützung bei allen Teilen der Arbeit, und auch für das gegebene Vertrauen, das mich in meinem Tun immer bestärkt hat. Mit ihrem Wissen und ihrer Erfahrung haben sie den Grundstein meiner Arbeit gelegt und mich den gesamten Weg begleitet. Vor allem während der intensiven Phasen hat mir die gemeinsame Projektarbeit eine enorme Freude bereitet.

Das hervorragende akademische Umfeld der Ruprecht-Karls-Universität Heidelberg hat am Abschluss der Arbeit großen Anteil. Insbesondere war es spannend, bei der Entstehung der Nachwuchsforschungsgruppe *Mathematical and Computational Optimization* dabei zu sein und den Übergang in die Nachfolgegruppe *Mathematical Algorithmic Optimization* mitzuerleben. Die gemeinsamen Diskussionen in dieser Gruppe haben mir oft neue Perspektiven gegeben und maßgeblich zur Entstehung der Arbeit beigetragen. Es war schön, Mitglied der Großfamilie der Arbeitsgruppe *Simulation und Optimierung* mit allen verwandten Nachwuchsgruppen zu sein. Herausheben möchte ich aus diesen Gruppen *Holger Diedam*, *Jonas Rauch* und *Bernat Duran* für die fruchtbaren Gedankenaustausche und die gemeinsam verbrachte Zeit. Auch gilt mein Dank *Kathrin Hatz* – das gemeinsame Durchleiden der Schreibensphase hat diese deutlich ertäglicher gemacht.

Danken möchte ich explizit auch *Alberto Caprara*, *Moritz Diehl* und *Sebastian Sager*, die mich – jeder auf seine Weise – überhaupt erst dazu gebracht haben zu promovieren.

Für die finanzielle Unterstützung möchte ich mich bei der *Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences* und der *Otto-von-Guericke-Universität Magdeburg* (über das Projekt *EMBOCON*) bedanken.

Meinen Eltern *Hans* und *Dagmar* und meiner Schwester *Birgit* möchte ich für die stets bedingungslose Unterstützung und Liebe danken. Sie haben mich zu dem gemacht, der ich bin.

Mein letzter Dank geht an meine Tochter *Hanna*, die mein Leben so sehr verschönert hat, für ihre ansteckende Lebensfreude und Begeisterung und an meine Frau *Lena* für die tiefe Liebe und Unterstützung in jeglicher Hinsicht, aber auch für den einen oder anderen nötigen Schubs in die richtige Richtung.

„Je mehr du gedacht, je mehr du getan hast, desto länger hast du gelebt."

Immanuel Kant

Contents

1 Introduction

Optimization has always been part of the human nature and was originally dictated by Darwin's law of natural selection. Naturally, optimization is the process of finding the best solution to a certain problem. In mathematics, this is formalized such that mathematical optimization stands for the minimization or maximization of a certain objective function possibly under the presence of additional limitations. It has been a success story in the past decades in its application to various fields of our lives, as for example economics, biology, chemistry, physics and engineering. Mathematical methods to optimize different kinds of problems have been on the rise and more and more complex become solvable.

One class of problems whose solution process has been vastly studied with mathematical methods is the optimization of dynamic systems, i.e., *Optimal Control Problems (OCPs)*. A dynamic system is a system whose future states are determined by an evolution rule. This evolution rule prescribes the system's behavior in dependence on the current system states and potentially additional external controls. The optimization discipline that treats dynamic systems is called optimal control and the evolution rules are often systems of *Ordinary Differential Equations (ODEs)* or *Differential Algebraic Equations (DAEs)*.

Another widely studied class of problems are *Integer Programs (IPs)*. Here, the operator of a static or time discrete system is given choices between different, discrete options. These problems are also often called combinatorial problems due to their innate nature and involve mostly planning and strategic questions. The problems usually considered in this field have an integrality requirement for most or all variables of the problem and they have linear functions for both the objective and the constraints.

This thesis now treats the union of those two disciplines, which we call *Mixed-Integer Optimal Control (MIOC)* with regard to its mathematical origins. This class of problems consists of dynamic systems that can be run in different operation modes and contain logical decision-making. The system operator is now faced with the choice of the system's mode in addition to controlling each mode's own dynamic system. Mixed-Integer Optimal Control Problems (MIOCPs) have been gaining significantly increased attention over the last 15 years since the considered systems are usually highly complex due to their combinatorial, nonlinear and dynamic nature and thus have a high potential for optimization.

In the engineering world, these systems have also been attracting considerable interest. There, the class of problems is referred to as *hybrid optimal control*. This term was chosen to stress the hybrid nature of the problems in having both continuous-valued and discrete-valued variables [7, 39, 166]. Other terms for this class of problems, which are used less

often, include the description *dynamic optimization* instead of *optimal control* to get *mixed-logic dynamic optimization* and *mixed-integer dynamic optimization* [140]. Sometimes, since the discrete structure appears to be similar to a *bang-bang* structure, where the continuous controls take only values at the boundaries, the term *optimal bang-bang control* is used [132].
There is a wide range of different solution approaches regarding these MIOCPs. An overview of these approaches is given in [160]. One of the more prominent approaches is the indirect approach, which uses a *hybrid maximum principle* [4, 8, 19, 39, 166, 167, 174]. Here, the optimality conditions are applied on the infinite-dimensional optimization problem in the function space, and the solution is analytically derived. For this thesis, the focus is put on the *direct approach*, i.e., methods that *discretize first, then optimize*.
The modeling of a system having different system modes with binary variables is not unique but there exist several possibilities. Some of them are the *Inner Convexification (IC)* [78], the *Outer Convexification (OC)* [152], the formulation with *vanishing* and *complementarity constraints* [96, 121], or *logic-based programming* – also named *General Disjunctive Programming (GDP)* [57, 85, 137, 139, 180]. The GDP approach either uses *perspective functions* or a *Big-M* reformulation [84].
Several methods exist for the discretization of the resulting system dynamics. The most prominent examples are *collocation* [10, 23, 26, 27, 105] and *multiple shooting* [33, 117]. For direct methods, the state and control discretizations play an important role. In dependence on the problem and after the discretization is chosen the resulting problem may either become a Mixed-Integer Linear Program (MILP) through linearizations or a Mixed-Integer Nonlinear Program (MINLP). MILPs allow the usage of methods like *branch-and-bound* or *cutting plane* algorithms – for an example cf. [130]. MINLPs are solved with the generalized *nonlinear branch-and-bound* or with *outer approximation*, an example is given in [76]. The nonlinear formulation usually involves non-convexities due to the nature of the discretized dynamic system. Therefore, adaptations for *global optimization* might be needed to guarantee optimality [66, 123, 168]. However, in the context of *model-predictive control*, these means of an optimality certificate might take too long. The possible means to speed up the solution process of local methods for MIOCPs in this context have been studied in [111].
After an optimal solution has been determined for a certain discretization, it may be possible to enhance the solution in a post-processing step. The solution's switching structure between the system's modes is kept and assumed to be optimal, but the exact switching point times of the fixed system mode sequence can be re-optimized in order to improve the solution as proposed in [78, 107, 160, 175].
The applications of MIOC are manifold. Typical examples are the choice of gears in automotive control [76, 93, 111, 112, 159, 160, 177] or the choice of mode of driving for hybrid cars [181]. Other typical examples are the optimal operation of water or gas networks that involve pumps and valves [43, 44, 130] and the optimal control of processes in chemical engineering that also involve valves [106, 169]. The optimization of traffic flows on networks [68], the optimal control of continuous supply chain networks [80] or the operation of dis-

tributed autonomous robot systems [1] are other, less studied MIOCPs. An open benchmark library for MIOCPs is available at [155].

Thesis aims and contributions

The aim of this thesis is to help understanding MIOCPs and to augment their solution processes. The insights obtained in this thesis are split into two major parts as explicated in the following.

Firstly, insights concerning the problem formulation are bundled in Chapter 3: *'Mixed-Integer Optimal Control – Modeling and Relaxation'*. There, we give an elaborate overview of the different ways to formulate an MIOCP. The correct and problem specific choice of the formulation is an important part of the modeling, and it is vital for the optimization of the controlled dynamic process. We describe five distinct approaches with their advantages and disadvantages to facilitate a well-thought-out decision on the choice of the MIOCP's modeling formulation. The most important properties to consider are the resulting problem's dimensions (both of variables and constraints used), the tightness of the formulation for relaxation purposes and the formulation's behavior with regard to convexity and constraint qualifications. These properties strongly influence the choice of usable solution techniques and the behavior of the solution process. The corresponding solutions' features are shown for a sewage network example in Section 5.3 and a truck cruise control in Section 5.4. Secondly, a special rounding heuristic to obtain integer controls out of relaxed controls, which has an ϵ-optimality certification under certain circumstances, is presented in Chapter 4: *'The Control Approximation Problem for Control-Affine Systems'*. The OC formulation leads to control-affine systems with regard to the binary controls. In [157], a theoretical result, which derives a dependence of the difference of the two state trajectories from the difference of the corresponding control trajectories, has been presented. We translate this proof to a discrete setting, which is usually used in collocation. Already in [152], the basic problem of this chapter is presented – i.e., the *Control Approximation Problem* in the integral sense, which gives the best approximation of a continuous, relaxed control by an integral control. SAGER provided the Sum-up Rounding (SUR) heuristic for the corresponding one-dimensional or decoupled problem. This heuristic solves the problem for equidistant grids. We show that the adaptation of the basic idea to the multi-dimensional setting, where different controls are coupled with an Special Ordered Set type (SOS)1-constraint, results in a heuristic that can only give an optimality certificate in dependence on the number of controls. We derive an alternative Next-forced Rounding (NFR) heuristic, which overcomes this dependence on the problem dimensions and provides the provable (data-independent) best bound any heuristic can give for the problem. In contrast to the heuristics, we also study the true solution process of the problem. Therefore, we briefly examine the corresponding polytope describing the feasible set. This polytope is highly unstructured and data-dependent, which does not enable nice cutting plane algo-

rithms. Instead, we examine the properties of a LAGRANGIAN relaxation of the problem – especially in a branch-and-bound framework. From the one-dimensional LAGRANGIAN relaxation's solution, we derive a branching strategy that has advantageous properties for the bounds and lets them be computed in linear time. We show that these advantageous properties carry over to the multi-dimensional setting with SOS1-constraints. At last, we allow for additional constraints, as e.g. switching constraints, and show how to construct an algorithm that treats these changed circumstances. This algorithm and the derived LAGRANGIAN bounds are compared with the standard MILP solvers CPLEX© and SCIP and their Linear Programming (LP) bounds. This comparison is done for a LOTKA-VOLTERRA fishing problem in Section 5.2 and for a small, nonlinear, illustrative example in Section 5.1.

Contributions to publications

During the creation of this thesis, we contributed to four publications. Here, we describe the content of the papers and highlight the contributions of the author of this thesis:

[158] S. SAGER, M. JUNG AND C. KIRCHES, *Combinatorial Integral Approximation*, Mathematical Methods for Operations Research, 2011, Vol. 73(3):363–380.

We describe the Control Approximation Problem and give a short derivation as a decoupling method for control-affine MIOCPs. The new part of this paper is the addition of combinatorial constraints, e.g. switching constraints, to the problem. We formulate the resulting optimization problem and analyze the polytope describing the feasible set. Then, we give an alternative to standard MILP methods, i.e., a tailored branch-and-bound algorithm, and demonstrate its effectiveness in a LOTKA-VOLTERRA example.
For this paper, JUNG's main contribution is a preliminary version of the branch-and-bound algorithm to solve the Control Approximation Problem, cf. Algorithm 4.2, and the corresponding results. Additionally, he added the study of the facets of the problem's polytope, which is presented with more details in Section 4.5.2. SAGER derived the main mathematical models and KIRCHES implemented them with standard solvers, i.e., CPLEX©, IPOPT and BONMIN.

[104] M. JUNG, G. REINELT AND S. SAGER, *The Lagrangian Relaxation for the Combinatorial Integral Approximation Problem*, Optimization Methods and Software (Submitted).

In this work, we derived the theory of the LAGRANGIAN bounds of the Control Approximation Problem, which is explained in Section 4.3.2 and Section 4.4.3. We also compare the effectiveness of these LAGRANGIAN bounds with the canonical LP bounds, cf. Section 5.1.
As the first and corresponding author, JUNG wrote the publication and worked out the mathematical proofs and the implementations of the compared methods. The coauthors contributed in discussions and reviewed the paper before submission.

[103] M. JUNG, C. KIRCHES AND S. SAGER, *On Perspective Functions and Vanishing Constraints in Mixed-Integer Nonlinear Optimal Control*, in Facets of Combinatorial Optimization – Festschrift for MARTIN GRÖTSCHEL, M. JÜNGER AND G. REINELT, eds., Springer Berlin Heidelberg, 2013, pp. 387–417.

The paper includes the description of various modeling approaches for MIOCPs and a discussion of their advantages and disadvantages based on our experience. It demonstrates these properties for a cruise control problem for a heavy-duty truck.
As the first author, JUNG added the GDP point of view to the MIOC modeling and was responsible for the complete implementation. The presented formulations, derived from the GDP approach, are the Big-M formulation, cf. Section 3.7, the perspective formulation from Section 3.9, and a tightened version of the perspective formulation, cf. Section 3.9.2. KIRCHES provided the truck model, the insights into the complementarity formulations, and the visualization of the results. SAGER initiated the work on the paper and provided insights into the IC and OC formulations.

[56] B.J. DURAN, M. JUNG, C. OCAMPO-MARTINEZ, S. SAGER AND G. CEMBRANO, *Minimization of Sewage Network Overflow*, Water Resources Management (Accepted).

The paper describes the problem of optimal control in sewage networks. Due to limited capacity, overflow happens during periods of heavy rainfall, which is mathematically formulated with a maximum function. This results in a nonlinear, non-differentiable model. We provide the network modeling and give reformulations that overcome the problems of the maximum function. The different formulations are compared for a set of 22 rain scenarios for a part of the BARCELONA sewage network.
The work was created while DURAN stayed in Heidelberg with JUNG and SAGER. DURAN was responsible for the network modeling, worked out the Mixed Logical Dynamical (MLD) reformulation, and also provided the test scenarios. JUNG – mentored by SAGER – developed the other mathematical reformulations and their implementations, i.e., the smoothed nonlinear formulation, the constraint branching algorithm, and the perspective formulation. The other coauthors contributed from the engineering application point of view and by providing the challenging and interesting test problem. The paper has been written jointly by DURAN and JUNG.

Thesis overview

This thesis is structured in four major chapters, which are set up as follows.
Chapter 2 describes OCPs in an introductory manner. It first describes the problem class of deriving the optimal control for dynamic processes that are driven by ODE or DAE systems.

On this basis, we discuss three general solution approaches. The first one is the indirect approach, which is based on the *Maximum principle*. The second discussed approach is dynamic programming based on the solution of the HAMILTON-JACOBI-BELLMAN partial differential equations. The third approach covers direct methods and specifies the needed building blocks to implement a direct method for OCPs.
In Chapter 3, we thoroughly investigate MIOCPs. The chapter starts with the generalization of the previous chapter's problem description to the integral setting where the system can switch between different operation modes. Depending on the operation mode, the underlying ODE might change as well as the control and path constraints limiting the feasible region. Then, the modifications needed to adapt indirect methods, dynamic programming and direct methods to this new setting are outlined. A major part of the adaptation of direct methods is to properly formulate the integral relaxation of the different modes' ODE systems and constraints. Several ways to do so are presented in Sections 3.5–3.9, i.e., the IC technique, the OC technique, the Big-M approach, the formulation with vanishing constraints and the perspective formulation. The different approaches are also discussed in terms of their numerical stability and general solvability. They are illustrated and compared in Section 3.10. A natural property of MIOCPs is that a solution might need to switch infinitely many times, which is called ZENO's Phenomenon. However, controls of this type are impossible to implement in reality and hence the corresponding mathematical models should be modified to better reflect reality. Even for direct methods with finite control discretization, a discretized system might switch with high frequency between its different modes, which is not desirable. The prevention of this switching behavior is done by either penalizing switches or by directly limiting their occurrences. This is described in detail in Section 3.11. The chapter closes with an overview of solution methods for MINLPs, which are needed to solve the problems with direct methods.
Chapter 4 treats a special rounding problem to obtain integer control trajectories out of relaxed ones. The rounding strategy makes use of a property of MIOCPs where the supposedly integral controls enter affinely. For these problems, it can be shown that the state trajectory deviation of two controls depends linearly on the deviation of the two control trajectories in a special integral sense. This is shown in Section 4.2 and there this property is also carried over to a discrete setting. Then, the approximation in the integral sense of relaxed controls with integer controls is studied in different settings with increasing difficulty over the next sections. First, completely decoupled controls are examined and the SUR strategy is established as a heuristic in Section 4.3. Here, also the solution structure of a branch-and-bound algorithm with a LAGRANGIAN relaxation as subproblems is investigated. The insights are carried over to the more complex setting of SOS1-coupled controls in Section 4.4. This setting deserves special interest since it arises when the original MIOCP's ODE system is reformulated with the OC technique. The NFR heuristic is employed to give a better worst-case scenario than previously known. In addition, the results of the decoupled problem concerning the LAGRANGIAN relaxation are incorporated into this more general setting. At last in Section 4.5,

the modifications needed for additional constraints are discussed particularly with regard to switching constraints. The complexity of the resulting polytope is briefly examined, and a branch-and-bound algorithm is presented that utilizes the findings on the LAGRANGIAN relaxation.
The final Chapter 5 gives numerical results on the strategies discussed in the previous two chapters. The first application covers the OC relaxation followed by the control approximation in the integral sense for a nonlinear MIOCP. Here, the properties of the branch-and-bound algorithm derived in Chapter 4 are compared to standard solvers for MILPs – in an unconstrained and also in a limited switching context. The second application is the stabilization of a LOTKA-VOLTERRA type predator-prey model is examined with regard to the switching formulation. This is also executed in the framework of the OC formulation followed by the control approximation in the integral sense. The results are compared with the optimal solutions whenever easily attainable. Next, the optimal control of a sewage network is determined in order to prevent overflow during periods of extensive rain. Here, the Big-M formulation and the perspective formulation are applied and compared to another, more problem specific approach. These two approaches are singled out from the collection since they preserve linearity of the model. A heavy-duty truck's cruise control is the last application presented in Section 5.4. The five different formulation approaches are applied to the problem and the results are studied for two different scenarios.

General setup for computational experiments

All results are obtained on a machine with an Intel® Core™2 Duo CPU E7300 with 2.66GHz and 8GB RAM. The operating system is Ubuntu ® Linux™*v12.04.02*. The source code has been written in C++. It was compiled with the *GNU* C/C++ *compiler collection v4.6.3* compiler. Additionally, the *Standard Template Library* (STL) and the *Boost* C++ libraries were used for further functionalities. To enable a fair comparison of computational times, all programs were only run on a single core even if the code was parallelized. The following software packages were used in this thesis:

- MATLAB© *v.7.7.0* and *v.8.1.0* to generate graphics and to model some problems,
- AMPL *v.20130327* to model several problems,
- MUSCOD-II *v.6.0* to discretize and solve OCPs,
- IPOPT *v.3.11* with the HSL linear solvers as an Nonlinear Program (NLP) solver,
- BONMIN with CBC *v.2.7* and IPOPT *v.3.11* to solve MINLPs,
- SCIP *v.3.0.0* with SOPLEX *v.1.7.0* as an MILP solver,
- CPLEX©*v.12.1.0* and *v.12.5.0* as MILP solvers with the interfaces to AMPL, C++ and MATLAB©,

- the HYbrid System DEscription Language (HYSDEL) package *v1.2.8* for MATLAB©to generate MLD systems,
- PORTA *v.1.4.1* to generate polytopes for MILPs,
- BUCHNER's dynamic programming code [42] to generate a global solution for the truck cruise control of Section 5.4.

2 Optimal Control

This chapter covers solution approaches for a class of continuous OCPs. We briefly introduce the OCP and some variants in Section 2.1 and give the means to model a dynamic process correctly. There are several approaches of tackling these OCPs, we present three different concepts. The first two, i.e., the *indirect approach* in Section 2.2 and *dynamic programming* in Section 2.3, are presented in an introductory manner for the sake of completeness. The presentation of the *direct approach* in Section 2.4 is more detailed as it is used for the computational experiments in Section 5.
For the presentation of the widely known and used problem class of OCPs, we adhere closely to the overviews given in [55, 77, 111, 152].

2.1 Problem formulation

This section gives an overview of the problem class of continuous OCPs.

Definition 2.1 (Continuous optimal control problem)
A Optimal Control Problem (OCP) is an infinite-dimensional, constrained optimization problem of the following form:

$$
\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \quad & \Phi(\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)) && && (2.1)\\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t,\boldsymbol{x}(t),\boldsymbol{u}(t)), && t \in [t_0,t_f],\\
& \mathbf{0} = \boldsymbol{r}^{eq}((t_i,\boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}), && \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0,t_f],\\
& \mathbf{0} \leqslant \boldsymbol{r}^{in}((t_i,\boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}), && \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0,t_f],\\
& \mathbf{0} \leqslant \boldsymbol{c}(t,\boldsymbol{x}(t),\boldsymbol{u}(t)) && t \in [t_0,t_f],\\
& \boldsymbol{u}(t) \in \mathcal{U} && t \in [t_0,t_f],
\end{aligned}
$$

in which we minimize a cost functional

$$\Phi : \mathcal{X} \times \mathbb{R}^{n_u} \to \mathbb{R} \tag{2.2}$$

with the dynamic process

$$\boldsymbol{x} : [t_0,t_f] \to \mathbb{R}^{n_x} \tag{2.3}$$

determined through the Ordinary Differential Equations (ODEs) with right-hand side function

$$\boldsymbol{f} : [t_0, t_f] \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_x} \tag{2.4}$$

managed by the control function

$$\boldsymbol{u} : [t_0, t_f] \to \mathbb{R}^{n_u} \tag{2.5}$$

over the time interval $[t_0, t_f]$ *such that the point constraints*

$$(\boldsymbol{r}^{eq}, \boldsymbol{r}^{in}) : ([t_0, t_f] \times \mathbb{R}^{n_x})^{m+1} \to \mathbb{R}^{n_r} \tag{2.6}$$

on the grid points $\{t_i\}_{0 \leqslant i \leqslant m}$*, and the path constraints*

$$\boldsymbol{c} : [t_0, t_f] \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_c} \tag{2.7}$$

are satisfied. △

This problem is clearly infinite dimensional due to the unknowns being the control trajectory $\boldsymbol{u}(\cdot)$ and the controlled state trajectory $\boldsymbol{x}(\cdot)$. For the process to be well described and the controls to be applicable, we need the controls $\boldsymbol{u}(\cdot)$ to be measurable and essentially bounded and define the set of feasible controls $\mathcal{U} \subset \mathcal{L}^\infty([t_0, t_\mathrm{f}], \mathbb{R}^{n_\mathrm{u}})$. We also define the set of all state trajectories $\boldsymbol{x}(\cdot)$ as $\mathcal{X} \overset{\text{def}}{=} \{\boldsymbol{x} : [t_0, t_\mathrm{f}] \to \mathbb{R}^{n_\mathrm{x}}\} \subset \mathcal{C}^1([t_0, t_\mathrm{f}], \mathbb{R}^{n_\mathrm{x}})$. To ensure existence and uniqueness of the ODE system's solution through the PICARD-LINDELÖF *theorem*, we assume $\boldsymbol{f} : [t_0, t_\mathrm{f}] \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_\mathrm{x}}$ to be LIPSCHITZ continuous. Initial values, or more general boundary values, can be modeled through the point-wise constraints $\boldsymbol{r} : ([t_0, t_\mathrm{f}] \times \mathbb{R}^{n_\mathrm{x}})^{m+1} \to \mathbb{R}^{n_\mathrm{r}}$. These could also be of coupled nature as, e.g., periodicity constraints. The path constraints $\boldsymbol{c} : [t_0, t_\mathrm{f}] \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_\mathrm{c}}$ can contain pure state or control path constraints as well as those of mixed type.

Often, there is the possibility to move constraints to the objective by penalizing their violation, they become so-called *soft constraints*. However, this does not guarantee that they are satisfied in the solution but usually gives little violation. This is usually implemented for constraints that are not crucial for the process dynamics but exclude undesirable properties.

Definition 2.2 (Admissibility)
A trajectory $(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot))$ *is said to be admissible if* $\boldsymbol{x}(\cdot)$ *is absolutely continuous,* $\boldsymbol{u}(\cdot)$ *is measurable and essentially bounded and if they satisfy the constraints* (2.4)*,* (2.6) *and* (2.7) *of the OCP.*

A control trajectory $\boldsymbol{u}(\cdot)$ *is said to be admissible if there exists a trajectory* $\boldsymbol{x}(\cdot)$ *such that* $(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot))$ *is an admissible trajectory.* △

Definition 2.3 (Optimality)
A trajectory $(\boldsymbol{x}^*(\cdot), \boldsymbol{u}^*(\cdot))$ *is said to be globally optimal if it is admissible and it holds*

$$\Phi(\boldsymbol{x}^*(\cdot), \boldsymbol{u}^*(\cdot)) \leqslant \Phi(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) \tag{2.8}$$

for all admissible trajectories $(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot))$.
It is said to be locally optimal, if there exists $\delta > 0$ *such that* (2.8) *holds for all admissible trajectories* $(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot))$ *with*

$$\left|\left|\boldsymbol{u}(t) - \boldsymbol{u}^*(t)\right|\right| \leqslant \delta \qquad \forall t \in [t_0, t_f]. \qquad \triangle$$

Objective functionals

Prevalent objective functionals include the MAYER type functional

$$\Phi_{\mathrm{M}}(\boldsymbol{x}(\cdot)) = m(\boldsymbol{x}(t_{\mathrm{f}})), \tag{2.9}$$

which only considers the end-point values, the LAGRANGE type functional

$$\Phi_{\mathrm{L}}(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) = \int_{t_0}^{t_{\mathrm{f}}} l(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\,\mathrm{d}t, \tag{2.10}$$

which covers a continuous contribution of the states and controls and their combination, the BOLZA type functional

$$\Phi_{\mathrm{B}}(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) = \Phi_{\mathrm{L}}(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) + \Phi_{\mathrm{M}}(\boldsymbol{x}(\cdot)). \tag{2.11}$$

They can be transformed into each other through additional states, derivation and integration. All scenarios can be modeled with one of these objective functionals.

Parameters

We have to distinguish between controllable parameters and process-dependent parameters. The first class can be included into our framework. The second class is in our context fixed and part of the model equations. However, there is a whole branch of optimization addressing the issues of parameter estimation, cf. [31] for further information on this topic.
Here, we only describe the handling of the first class with the controllable parameters $\boldsymbol{p} \in \mathbb{R}^{n_{\mathrm{p}}}$:

$$\begin{aligned} \min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{p}} \quad & \Phi(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \boldsymbol{p}) && \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}) && t \in [t_0, t_{\mathrm{f}}], \\ & \boldsymbol{0} \leqslant \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}) && t \in [t_0, t_{\mathrm{f}}], \end{aligned} \tag{2.12}$$

$$\mathbf{0} \leqslant \boldsymbol{r}((t_i, \boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}) \qquad \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0, t_\mathrm{f}].$$

Most algorithms can directly handle this type of parameters. However, if a transformation to the original problem class (2.1) is needed, one can add n_p new differential states $\boldsymbol{x}_{\boldsymbol{p}}(\cdot)$ to the system with the corresponding part in the ODE being

$$\dot{\boldsymbol{x}}_{\boldsymbol{p}}(t) = \mathbf{0},$$

and the initial values being the degrees of freedom.

Variable time horizons

A variable time horizon $[t_0, t_\mathrm{f}]$ can be realized by the addition of a new differential state t for the unified time $\tau \in [0, 1]$, i.e.,

$$t(\tau) \stackrel{\text{def}}{=} t_0 + \left(t_\mathrm{f} - t_0\right) \tau.$$

Then, a time transformation has to be made for the ODE system:

$$\dot{\boldsymbol{x}}(\tau) = \left(t_\mathrm{f} - t_0\right) \boldsymbol{f}(t(\tau), \boldsymbol{x}(t(\tau)), \boldsymbol{u}(t(\tau))), \qquad \tau \in [0, 1].$$

Also for the objective and the constraints, the normal time t has to be replaced by $t(\tau)$. However, the chain-rule need not be considered there. This time transformation transforms the time interval $[t_0, t_\mathrm{f}]$ to $[0, 1]$, and all start and end times need to be adapted. The true start and end times are treated as controllable parameters as described in the section above.

Differential Algebraic Equations

The extension to DAE systems is possible, i.e., the addition of *algebraic states* $\boldsymbol{z} : [t_0, t_\mathrm{f}] \to \mathbb{R}^{n_\mathrm{y}}$ and *(differential) algebraic equations*:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t)), \qquad t \in [t_0, t_\mathrm{f}], \\ \mathbf{0} &= \boldsymbol{g}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t)), \qquad t \in [t_0, t_\mathrm{f}]. \end{aligned} \tag{2.13}$$

An important property of a DAE is the index. It is a measure for the regularity of the DAE and there are different index definitions, cf. e.g. [40, 46, 89]. One example is the *differential index*, which measures how often the algebraic equation has to be differentiated until we obtain an ODE. Another one is the *perturbation index*, which measures with what order a perturbation in the algebraic equation is propagated to the differential states. The index of the DAE determines how the methods need to be adapted to be able to solve the system.
We illustrate the differential index for a small example: Assume $\boldsymbol{g}$ and $\boldsymbol{u}$ to be differentiable and $\boldsymbol{g}_{\boldsymbol{z}}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t))$ to be invertible for all $(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t))$. Then, we can apply the

IMPLICIT FUNCTION THEOREM and obtain a function $\boldsymbol{z}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))$ that solves the algebraic equation (2.13). Furthermore, the system is of differential index 1 because we can obtain an explicit ODE system through one differentiation:

$$
\begin{aligned}
& \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{g}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t)) \\
\Rightarrow \quad & \mathbf{0} = \boldsymbol{g}_t(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t)) + \boldsymbol{g}_{\boldsymbol{x}}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t))\, \dot{\boldsymbol{x}}(t) \\
& \qquad + \boldsymbol{g}_{\boldsymbol{u}}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t))\, \dot{\boldsymbol{u}}(t) + \boldsymbol{g}_{\boldsymbol{z}}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t))\, \dot{\boldsymbol{z}}(t) \\
\Rightarrow \quad & \dot{\boldsymbol{z}}(t) = -\boldsymbol{g}_{\boldsymbol{z}}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t))^{-1} \Big(\boldsymbol{g}_t(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t)) \\
& \qquad\qquad + \boldsymbol{g}_{\boldsymbol{u}}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t))\, \dot{\boldsymbol{u}}(t) \\
& \qquad\qquad + \boldsymbol{g}_{\boldsymbol{x}}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t))\, \underbrace{\dot{\boldsymbol{x}}(t)}_{=\boldsymbol{f}(\cdot)} \Big).
\end{aligned}
$$

Multistage systems

Many systems cannot be described with one single set of model equations but they switch between n different sets depending on the time or the system states:

$$
\dot{\boldsymbol{x}}^j(t) = \boldsymbol{f}^j(t, \boldsymbol{x}^j(t), \boldsymbol{y}^j(t), \boldsymbol{u}^j(t)), \qquad\qquad t \in [t_j, t_{j+1}], 0 \leqslant j \leqslant n-1,
$$

These systems are called multistage systems. They have a different model (2.1) for every stage of the problem. If the sequence of stages is known and depends on the system states, switching functions are introduced

$$
\xi^j(t, \boldsymbol{x}^j(t)) = 0,
$$

which identify the exact switching times between the different model stages. The resulting system can be solved analogously to other OCPs. Often, the system could be modeled as a MIOCP, cf. Chapter 3 – however, the resulting problem may be more difficult than the adaptation of the OCP method to multiphase behavior. Yet, in the more interesting case, where the operator has the freedom to choose between different stages at every point in time and their sequence is free, the only way to model the system is to use integer controls as in MIOCPs.

An example multistage problem is the movement of a biped. In the different phases of the movement either none, one or both feet touch the ground and degrees of freedom disappear since new equations appear. The switching function is in this case described by the distance of the feet to the ground.

In [116], LEINEWEBER gives insight into the handling of the switching functions inside a multiple shooting framework as well as giving some chemical applications for multistage models. The behavior of an adsorption chillers is analyzed as a multistage process in [81, 82].

2.2 Indirect approach

The indirect approach is a long-known method, which is based on the works of PONTRYAGIN – who gives the main theorem his name – and his students BOLTYANSII and GAMKRELIDZE, cf. [144]. HESTENES also discovered the methodology independently around the same time, cf. [94]. The basic principle was already developed by CARATHÉODORY in 1935, cf. [141]. The idea is to *first optimize, then discretize*, i.e., the optimality conditions are applied in a function space. Further references on the topic can be found in [77].
We use an abbreviated formulation of the OCP without additional constraints and that uses an autonomous ODE system:

$$\begin{aligned} \min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \quad & \Phi(\boldsymbol{x}(t_\mathrm{f})) && && (2.14)\\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t)) && t \in [t_0, t_\mathrm{f}],\\ & \boldsymbol{u}(t) \in \mathcal{U} && t \in [t_0, t_\mathrm{f}],\\ & \boldsymbol{x}(t_0) = \boldsymbol{x_0}, \end{aligned}$$

with sufficiently smooth functions $\Phi(\cdot)$ and $\boldsymbol{f}(\cdot)$ and bounded set $\mathcal{U}$ of feasible controls. However, the approach is extendable to more general settings (extension to DAE, addition of constraints, etc.), which results in more intricate formulations of the theorem, cf. [79].
We need the following definition to formulate the main theorem of the indirect approach.

Definition 2.4 (HAMILTONIAN)
The HAMILTONIAN *of the optimal control problem* (2.14) *is defined as*

$$\mathcal{H}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\lambda}(t)) \overset{\text{def}}{=} \boldsymbol{\lambda}(t)^T \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t))$$

with the variables $\boldsymbol{\lambda} : [t_0, t_f] \to \mathbb{R}^{n_x}$*, called the adjoint variables.* △

Now, we can define the *maximum principle*, sometimes called PONTRYAGIN*'s maximum principle* or *minimum principle* for the simplified problem:

Theorem 2.1 (Maximum principle)
Let $\boldsymbol{u}^*(\cdot)$*,* $\boldsymbol{x}^*(\cdot)$ *be an optimal solution of problem* (2.14)*. There exist adjoint variables* $\boldsymbol{\lambda}^*(\cdot)$ *such that for almost all* $t \in [t_0, t_f]$ *it holds*

$$\begin{aligned} \dot{\boldsymbol{x}}^*(t) &= \mathcal{H}_{\boldsymbol{\lambda}}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),\boldsymbol{\lambda}^*(t)) = \boldsymbol{f}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t)),\\ \boldsymbol{x}(t_0) &= \boldsymbol{x_0},\\ \dot{\boldsymbol{\lambda}}^*(t)^T &= -\mathcal{H}_{\boldsymbol{x}}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),\boldsymbol{\lambda}^*(t)),\\ \boldsymbol{\lambda}^*(t_f)^T &= -\Phi_{\boldsymbol{x}}(\boldsymbol{x}^*(t_f)),\\ \boldsymbol{u}^*(t) &= \operatorname*{argmin}_{\boldsymbol{u}(t)\in\mathcal{U}} \mathcal{H}(\boldsymbol{x}^*(t),\boldsymbol{u}(t),\boldsymbol{\lambda}^*(t)). \end{aligned} \tag{2.15}$$

△

This approach gives only necessary conditions for an optimal solution of the problem. More insights on necessary conditions for variations of the problem can e.g. be found in [41, 91, 131, 134]. Usually, these conditions require a certain structure of $\mathcal{U}$ and involve derivatives of $\mathcal{H}(\cdot)$ with respect to $\boldsymbol{u}(\cdot)$. Sufficient conditions are also often used and usually need higher order derivatives, cf., e.g., [126].

Notice that the approach as stated above already covers MIOCPs as there are no further requirements on the set $\mathcal{U}$ than being bounded, it may very well be disjoint. This is possible since the constraint is transferred to the inner minimization problem (2.15), which needs to be solved globally to be able to use non-convex feasible sets. This is the global version of the *maximum principle*, and thus it does not state any additional requirements that the trajectories must fulfill in order to solve the inner minimization problem. These additional requirements would use derivative information to describe local minimizers of the HAMILTONIAN. However, this work focuses on the addition of integrality constraints for the controls, where discrete domains $\mathcal{U}$ arise. For these discrete sets derivative information cannot be provided and hence local versions of the theorem are not applicable.

The *first optimize, then discretize* approach works as follows: first the optimality conditions from the *maximum principle* 2.1 are applied to the OCP and the following Boundary Value Problem (BVP) is obtained:

$$
\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) && t \in [t_0, t_\mathrm{f}],\\
\dot{\boldsymbol{\lambda}}(t) &= -\mathcal{H}_{\boldsymbol{x}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)) && t \in [t_0, t_\mathrm{f}],\\
\boldsymbol{u}(t) &= \operatorname*{argmin}_{\boldsymbol{w}(t)\in\mathcal{U}} \mathcal{H}(\boldsymbol{x}(t), \boldsymbol{w}(t), \boldsymbol{\lambda}(t)) && t \in [t_0, t_\mathrm{f}],\\
\boldsymbol{x}(t_0) &= \boldsymbol{x_0},\\
\boldsymbol{\lambda}(t_\mathrm{f}) &= -\Phi_{\boldsymbol{x}}(t_\mathrm{f}).
\end{aligned}
$$

One main problem – but also a main advantage – of the approach is that the applicable *maximum principle* has to be determined and that the HAMILTONIAN and the solution of the minimization problem have to be computed analytically. This is quite hard for large scale models and one has to recompute large parts after small modifications of the system, e.g. the addition of a new constraint. However, the needed functional analysis may give insight into the solution's structure. Examples of successful applications of the indirect approach can be found in [32, 113, 141].

2.3 Dynamic programming

The *dynamic programming* approach was developed by BELLMAN to determine the optimal control of time-discrete OCPs with discrete state and control spaces. Its main insight is the *Principle of Optimality*, [16, Chapter III.3.]:

> "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

The principle leads to the BELLMAN equation to determine the *cost-to-go function*, which describes the minimal objective contribution from that point in time to the end of the time horizon. This formulation directly leads to a recursive solution algorithm. The method is generalizable to time-continuous OCPs, such as (2.1), with the HAMILTON–JACOBI–BELLMAN equation. In dynamic programming, the problem is decomposed into several smaller problems on the time horizon and thereby each subproblem only needs to consider the smaller time horizon $[t_i, t_{i+1}]$. The states and controls are discretized to be in finite-dimensional sets of possible state and control vectors. For all possible combinations of initial states at time step t_i, the optimal controls for the interval are computed, taking into account the previously computed and tabulated optimal continuation after time point t_{i+1}. This decomposition makes the increase in computational effort for finer time discretizations linear in the number of time intervals m.
The main disadvantage of the method is the so-called *curse of dimensionality*. It stems from the necessary discretization of continuous state and control spaces. Since for all possible combinations of states, the computations have to be made for each slice of the time horizon, the computational effort grows exponentially in the number of states n_{x} when the state discretization is made finer for all states. This is unfortunately a very strong drawback and limits the applicability of dynamic programming to problems with small numbers of states n_{x} and continuous controls n_{u}. Another disadvantage is that additional rounding errors are introduced at the grid points due to the computed trajectories being only in discretized form, whereas the simulated trajectory can also attain values in between. Now, the error is introduced as the optimal control was computed for the discretized values and not for the true values.
However, as an advantage, dynamic programming naturally respects integrality of controls since they would have to be discretized anyways, and does not introduce any additional effort. Another advantage of the method is that the whole feasible space is enumerated and hence the problem is automatically optimized globally – even if it is non-convex.
Further information on dynamic programming can e.g. be found in BERTSEKAS' textbook [20, 21].

2.4 Direct approach

In contrast to the *indirect approach*, the principle of the *direct approach* is to *first discretize, then optimize*. Following this motto, first, the whole problem – this includes the controls, the states, the constraints and the the objective function – is discretized, i.e., it is reformulated to obtain a finite-dimensional optimization problem. Then optimality conditions are applied to

the resulting finite-dimensional system.
For the approaches that we present here, we define the time grid

$$\mathbb{G}_m \stackrel{\text{def}}{=} \{t_0 < t_1 < \ldots < t_m = t_\text{f}\}$$

of $m+1$ points, with step sizes

$$\Delta t_i \stackrel{\text{def}}{=} t_{i+1} - t_i.$$

The time grid for the state discretization need not necessarily be the same as the grids for the control discretization and constraint discretization. However, usually the control grid and constraint grid coincide, whereas the state grid is a subset or superset thereof – depending on the chosen method. Obviously, for all methods it holds that a finer discretization grid leads to a better approximation of the function spaces.
A discretization method for optimal control problems is defined by these building blocks, which are described in the following sections:

- the control discretization with parameter vector $\boldsymbol{q}$ and control functions $\boldsymbol{u}(t;\boldsymbol{q})$,
- the state discretization with parameter vector $\boldsymbol{s}$ and state trajectories $\boldsymbol{x}(t;\boldsymbol{s},\boldsymbol{q})$,
- the constraint discretization,
- the computation of gradients for the resulting optimization problem,
- the choice of the algorithm to solve the optimization problem.

2.4.1 Control discretization

The space of feasible control functions $\mathcal{U} \subset \mathcal{L}^\infty([t_0,t_\text{f}],\mathbb{R}^{n_\text{u}})$ has to be replaced by a finite-dimensional subspace $\mathcal{L}^\infty_k([t_0,t_\text{f}],\mathbb{R}^{n_\text{u}})$, that is defined by k parameters $\boldsymbol{q} \in \mathbb{R}^{n_\text{k}}$. Then, every discretized control $\boldsymbol{u_k} \in \mathcal{L}^\infty_k([t_0,t_\text{f}],\mathbb{R}^{n_\text{u}})$ can be expressed with this parameter vector to be

$$\boldsymbol{u_k}(\cdot) \stackrel{\text{def}}{=} \sum_{i=1}^{k} q_i\, \boldsymbol{b^i}(\cdot), \tag{2.16}$$

where $\{\boldsymbol{b^1}(\cdot),\ldots,\boldsymbol{b^k}(\cdot)\}$ is a basis of $\mathcal{L}^\infty_k([t_0,t_\text{f}],\mathbb{R}^{n_\text{u}})$. In the following, this dependence on the vector $\boldsymbol{q}$ is indicated as $\boldsymbol{u_k}(t) = \boldsymbol{u_k}(t;\boldsymbol{q})$. The choice of discretization may be chosen for each control individually. We also use this situation to replace any dependence on $\boldsymbol{u_k}(\cdot)$ by a dependence on $\boldsymbol{q}$.
The most simple approach is to approximate the control function with piecewise constant functions on the time grid, i.e., the basis is defined as

$$\boldsymbol{b}^{i,j}(t) \stackrel{\text{def}}{=} \begin{cases} \boldsymbol{e}_j & \text{if } t \in [t_i, t_{i+1}), \\ 0 & \text{else,} \end{cases} \qquad 0 \leqslant i \leqslant m-1, \tag{2.17}$$

where $\boldsymbol{e_j}$ is the j-th unit vector. This leads to

$$\boldsymbol{u}(t) \stackrel{\text{def}}{=} \boldsymbol{q_i} \qquad t \in [t_i, t_{i+1}), 0 \leqslant i \leqslant m-1, \tag{2.18}$$

with m vectors $\boldsymbol{q_i} \in \mathbb{R}^{n_\text{u}}$ that together form $\boldsymbol{q}$. For completeness, the control at the final time step is set as

$$\boldsymbol{u}(t_\text{f}) \stackrel{\text{def}}{=} \boldsymbol{q}_m \stackrel{\text{def}}{=} \boldsymbol{q}_{m-1}.$$

However, this value is only needed for DAE models where the final algebraic state $\boldsymbol{y}(t_\text{f})$ may influence the final differential state $\boldsymbol{x}(t_\text{f})$.

Another standard way would be to use higher order B-splines as a representation of the controls. Choose an order $l \in \mathbb{N}$ for the control and use the corresponding B-splines as the basis for each control. The B-splines of order i are recursively defined as $b_j^i(\cdot)$:

$$b_j^0(t) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } t \in [t_j, t_{j+1}), \\ 0, & \text{else,} \end{cases} \qquad 0 \leqslant j \leqslant m-1,$$

$$b_j^i(t) \stackrel{\text{def}}{=} \frac{t - t_j}{t_{j+i} - t_j} b_j^{i-1}(t) + \frac{t_{j+i+1} - t}{t_{j+i+1} - t_{j+1}} b_{j+1}^{i-1}(t),$$
$$0 \leqslant j \leqslant m-i-1, 1 \leqslant i \leqslant m-1,$$

with the enlarged auxiliary grid

$$\mathbb{G}_m^l \stackrel{\text{def}}{=} \{\tau_i\}_{1 \leqslant i \leqslant m+2l-1}$$

defined through

$$\tau_i \stackrel{\text{def}}{=} \begin{cases} t_0 & \text{if } 1 \leqslant i \leqslant l, \\ t_{i-l} & \text{if } l+1 \leqslant i \leqslant m+l-1, \\ t_\text{f} = t_m & \text{if } m+l \leqslant i \leqslant m+2l-1. \end{cases}$$

The control is then parameterized by $m+l$ vectors $\boldsymbol{q_i} \in \mathbb{R}^{n_\text{u}}$ (also called DE BOOR points in this context) and is computed through

$$\boldsymbol{u_{m+l-1}}(t; \boldsymbol{q}) \stackrel{\text{def}}{=} \sum_{i=0}^{m+l-1} b_i^l(t)\, \boldsymbol{q_i} \qquad \in \mathcal{L}_{m+l-1}^\infty([t_0, t_\text{f}], \mathbb{R}^{n_\text{u}}).$$

This discretization is visualized in Figure 2.1. There are two advantageous properties of this control discretization. First, a required smoothness of the control can easily be satisfied by

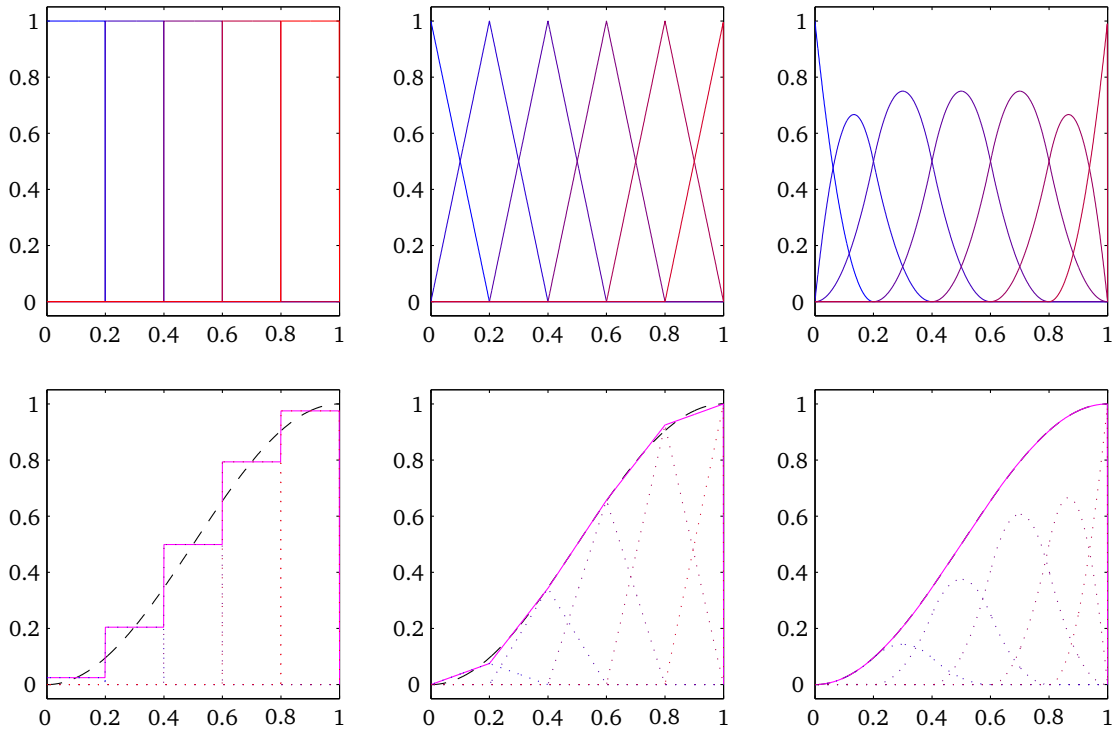


Figure 2.1: B-splines of different order ($i = 0$ on the left, $i = 1$ in the middle, $i = 2$ on the right) for an equidistant grid in the upper graphics.
Best approximation in $||\cdot||_{\mathcal{L}^1}$ of a sin function in the respective finite-dimensional spaces.

choosing the corresponding order l. Additionally, each function $\boldsymbol{b}_j^i(\cdot)$ has only local support on $[\tau_j, \tau_{j+l+1}]$. This leads to sparsity patterns in the derivatives. The exploitation of these patterns can strongly reduce the computational effort needed, cf. [74, 75].

If the multiple shooting discretization is used for the states as described in the next section, it is crucial for the efficient exploitation of the resulting structure that the control parameters have only local influence on the corresponding multiple shooting interval $[t_i, t_{i+1}]$, i.e., the basis functions need to have local support on one shooting interval only. Therefore, the B-spline approach is not applicable here for order greater 0, but one has to choose the control parameterization differently. For each multiple shooting interval $[t_i, t_{i+1}]$, we parameterize the control depending on the desired properties of the control trajectory. Examples for possible parametrizations are piecewise linear controls, which allow a continuous trajectory, and piecewise cubic splines, which allow continuous differentiability with given derivatives at the grid points.

For piecewise linear controls, two parameters $\boldsymbol{q}_i^k (k = 1, 2)$ are needed for each interval and

are combined with the basis functions $b_{ki} : [t_i, t_{i+1}] \to [0,1]$ defined as

$$b_{1i}(t) \stackrel{\text{def}}{=} \frac{t_{i+1} - t}{t_{i+1} - t_i},$$
$$b_{2i}(t) \stackrel{\text{def}}{=} \frac{t - t_i}{t_{i+1} - t_i}.$$

To enable continuity of the controls, additional constraints must be enforced though. These are constraints that couple two adjacent intervals through

$$0 = \boldsymbol{q}_i^{\mathbf{2}} - \boldsymbol{q}_{i+1}^{\mathbf{1}}, \qquad 0 \leqslant i \leqslant m-1.$$

Now, the difference between a B-spline formulation of the situation and this setting is that the B-spline formulation uses the equation to directly eliminate $\boldsymbol{q}_i^{\mathbf{2}}$ from the problem. This formulation instead lifts the controls to a higher dimension and passes both controls and the coupling constraint to the optimization problem.
Naturally, the flexibility gained through higher order parametrizations gives better results regarding the OCP at the cost of higher computational effort. This effect can be seen e.g. in [112]. However, piecewise constant and continuous, piecewise linear functions are the preferred control parametrizations, cf. e.g. [45, 172].

2.4.2 State discretization

The discretization of the states can be done through a one-step or a multi-step method, or a more complex integrator may be used for the steps. The extension to the DAE case is also possible here, and the algebraic states are discretized in the same manner as the differential states. If an integrator shall be used, it must naturally be chosen appropriately.
In the following, the three well-known methods *single shooting*, *multiple shooting* and *collocation* are presented. They could also be described as minimal discretization, reduced discretization, and full discretization with regard to the number of variables introduced in the optimization problem. These introduced variables, which describe the discretized states, are called $\boldsymbol{s_i}$ in the following.

Single shooting

As the dynamic process is completely described if the controls and initial values are known, the single shooting discretization uses the initial states as the sole degrees of freedom for the system states $\boldsymbol{s_0} = \boldsymbol{x_0}$. The trajectory and its derivatives are given by the solution of the corresponding Initial Value Problem (IVP) through an ODE solver also called integrator. Path constraints $\boldsymbol{c}(\cdot)$ are often enforced on the control grid only. The integrator has to be stopped at the points where the constraints are applied and derivatives of the trajectory have to be determined. Figure 2.2 visualizes the single shooting discretization approach.

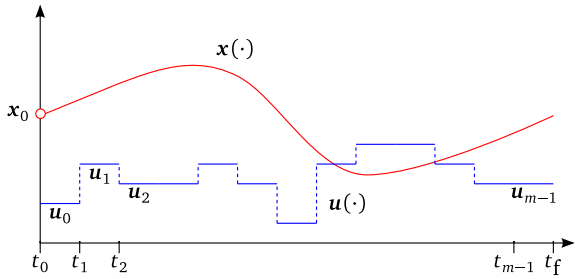


Figure 2.2: Visualization of the single shooting state discretization while marking the degrees of freedom, which are in this case $\boldsymbol{u}$, $\boldsymbol{x_0}$.

The advantage of the approach is that it is straightforwardly implemented, cf. [161]. Additionally, the number of variables in the resulting optimization problem is minimal.
However, there are two big disadvantages as well. First, the state trajectory cannot be initialized with a priori known information about the process except its initial values. Second, the error propagation is difficult to control for some systems, depending on the nonlinearity. For stiff problems, a very good initial guess for the initial state $\boldsymbol{x_0}$ and the controls must be known to prevent blowup effects from happening. If the initial guess is too far away, a solution may not even exist due to singularity. Due to the error propagation depending strongly on the nonlinearity of the process, the process may still run into numerical problems, e.g. a singularity – even if the initial values were good enough.

Multiple shooting

The direct multiple shooting method originates in [33, 143] for the solution of BVPs. The implementation MUSCOD-II was used for some models described in Chapter 5; it is described in [117].
In contrast to single shooting, direct multiple shooting employs additional variables such that the problem is lifted into a higher dimensional space as follows. The time horizon is divided with a time grid $\mathbb{G}_m$ – called *shooting grid* – and the new variable vectors $\boldsymbol{s_i} \in \mathbb{R}^{n_x}$ – called *shooting variables* – are used to describe the state at the grid points. These shooting variables are the initial values for the IVPs

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \qquad t \in [t_i, t_{i+1}), \\ \boldsymbol{x}(t_i) &= \boldsymbol{s_i}, \end{aligned} \tag{2.19}$$

with $0 \leq i \leq m-1$. To clarify which arc of the trajectory is meant, we use the notation

$$\boldsymbol{x}(t; \boldsymbol{u}(\cdot), \boldsymbol{s_i}) = \boldsymbol{s_i} + \int_{t_i}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau))\, \mathrm{d}\tau \qquad t \in [t_i, t_{i+1}).$$

This system (2.19) differs from the original one as continuity is not required in the grid

points. Therefore, the resulting problem would produce solutions as on the left-hand side of Figure 2.3.

The original problem is obtained if the discontinuity is prevented, hence *matching constraints* are employed to enforce continuity in the shooting grid nodes:

$$\boldsymbol{x}(t_{i+1}; \boldsymbol{u}(\cdot), \boldsymbol{s_i}) = \boldsymbol{s_{i+1}}, \qquad 0 \leq i \leq m-1.$$

The evaluation of these constraints requires the solution of an IVP to get $\boldsymbol{x}(t_{i+1}; \boldsymbol{u}(\cdot), \boldsymbol{s_i})$. Therefore, the numerical method of choice for the solution of the nonlinear optimization problem must be coupled with an integrator. This has to be chosen adequately for the type of the problem and the optimizer. If, e.g., algebraic states are present they should be resolved simultaneously, cf. [5, 13].

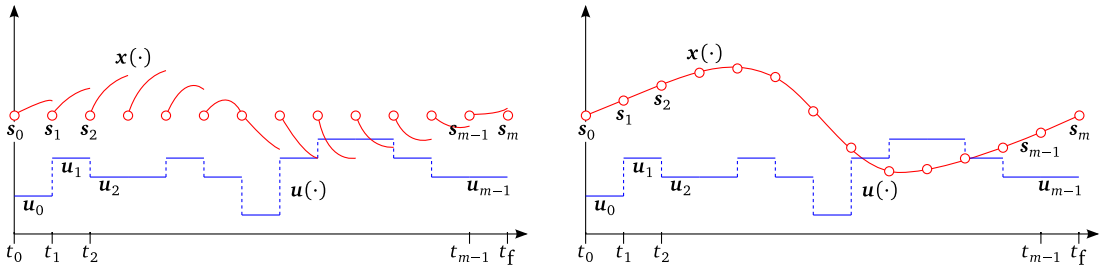


Figure 2.3: Visualization of multiple shooting state discretization. On the left-hand side a solution of the relaxed problem without matching constraints is displayed, whereas on the right-hand side they are added and the solution is again a solution of the original problem.

One advantage of this discretization and parameterization is that a priori knowledge about the optimal trajectory can be brought in through the initialization of the shooting variables $\boldsymbol{s_i}$. If this can be done, the system often converges very fast. An interpretation is that the fast, linear parts of the problem are made more dominant through the shortening of the horizon, or the nonlinearity diminishes, cf. [6]. Additionally, the stability of the process is improved, because blowup effects can only happen on the much shorter time horizons $[t_i, t_{i+1})$ and hence are exponentially less likely. Furthermore, the IVPs for the short time horizons can be simultaneously solved in parallel if the system architecture allows for parallelism. Another advantage compared to collocation is that state-of-the-art solvers for the IVP can be used to improve efficiency and precision, which can use adaptive step refinements and error control methods, cf. [5].

One might argue that there are strong negative effects due to the larger problem size that stems from the addition of the $\boldsymbol{s_i}$ variables in comparison to single shooting. However, through the so-called *condensing* technique, these variables can be eliminated from the actual computations. Still, there is some overhead needed to accomplish this.

Notice that we are only able to enforce the point-constraints on the shooting nodes with this

discretization, i.e., all points, where they are required, must be included in the shooting grid:

$$\begin{aligned}\mathbf{0} &= \boldsymbol{r}^{\mathrm{eq}}((t_i, \boldsymbol{s_i})_{0\leqslant i\leqslant m}),\\ \mathbf{0} &\leqslant \boldsymbol{r}^{\mathrm{in}}((t_i, \boldsymbol{s_i})_{0\leqslant i\leqslant m}).\end{aligned}$$

Collocation

In this case, the states are discretized on the time grid $\mathbb{G}_m$ as $\boldsymbol{x}(t_i) = \boldsymbol{s_i}$ and the values are connected with a one-step or multi-step integration method. All the corresponding computations are included in the optimization problem. If we have e.g. an explicit one-step method like the HEUN method, we replace the ODE through the equations given by the scheme

$$\begin{aligned}\boldsymbol{s_{i+1}} = \boldsymbol{s_i} &+ \frac{\Delta t_i}{2}\boldsymbol{f}(t_i, \boldsymbol{s_i}, \boldsymbol{q})\\ &+ \frac{\Delta t_i}{2}\boldsymbol{f}\left(t_i + \frac{\Delta t_i}{2}, \boldsymbol{s_i} + \frac{\Delta t_i}{2}\boldsymbol{f}(t_i, \boldsymbol{s_i}, \boldsymbol{q}), \boldsymbol{q}\right), \qquad 0 \leq i \leq m-1,\end{aligned}$$

where $\boldsymbol{q}$ are variables used to parameterize the controls and parameters in $[t_i, t_{i+1})$. A thorough overview on collocation is given in [27, Chapter 10].

These equations are now included into the optimization problem instead of the ODE. Depending on the solver, the derivatives of these equations with respect to the discretized states and controls may also be required. Since the ODE is not evaluated separately, the grid has to be quite fine to sufficiently resolve the trajectories. Therefore, a lot of variables have to be included in the optimization problem, yet the structure strongly depends on the implemented collocation scheme and the resulting JACOBIANS are sparse. It is essential for the performance that these structural properties are exploited, cf. [22, 25]. Instead of using a very fine grid right from the start, it is also possible to use a coarser grid and to adaptively refine it during run time, cf. [24]. However, convergence problems exist for singular control problems, where the optimal control is not at its bounds, cf. [27].

2.4.3 Constraint discretization

The point constraints $\boldsymbol{r}(\cdot)$ are already in a discretized formulation whereas the path constraints $\boldsymbol{c}(\cdot)$ still need to be discretized. Depending on the type of constraint, they are usually discretized on the same grid as the entering variables – i.e., pure state constraints are discretized on the state grid, pure control constraints are discretized on the control grid and mixed constraints are discretized on the intersection of both grids. If both were the same grids, we would obtain

$$\mathbf{0} \leqslant \boldsymbol{c}(t_i, \boldsymbol{x}(t_i), \boldsymbol{u}(t_i)), \qquad 0 \leqslant i \leqslant m.$$

If the desired accuracy for the constraints is not achieved, the constraints may also be required on additional points, i.e., a finer sub-grid of broad grids, which are used for the control and state discretizations, has to be employed. A problem is often that either the state trajectory is not known on these intermediate points or that the JACOBIAN cannot be provided. Therefore, sometimes the discretization of the states and/or controls need to be adapted as well to obtain the desired resolution of the constraints. One approach to introduce the continuous path constraints in the direct multiple shooting state discretization is the *minima tracking approach* described in [145], which adds new tracking points for the constraints without refining the resolution of the states.

2.4.4 Gradient computation

Most optimizers need at least gradients of the discretized objective function $\Phi(\cdot)$ and the JACOBIAN of the discretized constraints $\boldsymbol{c}(\cdot)$ and $\boldsymbol{r}(\cdot)$. The HESSIAN might be also needed, depending on the optimizer used. For the fully discretized system, i.e., if collocation was used to discretize the states, the derivatives of the functions can usually be directly computed. There are three ways to do so.

- *Analytic differentiation* is just computing the derivatives by hand and implementing them as functions. This includes *symbolic differentiation*, where tools are used to generate the exact derivative formulation. For larger problems, the effort to compute all function derivatives can be quite high and is often error-prone.
- *Finite differences* use different function evaluations to approximate the derivatives. However, it suffers from limited accuracy and high computational burden.
- *Automatic differentiation* uses the chain and product rules of differentiation to compute the derivatives out of C or FORTRAN code. It uses either operator overloading or source code transformation via a compiler. The approach is fully presented in the GRIEWANK's textbook [83]. Details on an implementation can e.g. be found in [183] for ADOL-C, an operator overloading package.

If the ODEs are solved with an integrator as in single shooting and multiple shooting, the computation of the derivatives of the trajectories with regard to the discretized states and controls is more complicated and needs access to the integrator. There are essentially two versions, the *sensitivity equation approach* and the *adjoint equation approach*. We only present them for single shooting and the dependence of the initial values, the extension to multiple shooting is straightforward.

Sensitivity equation approach

For the *sensitivity equation approach* or the solution of the *variational IVP*, the ODE is differentiated with regard to $\boldsymbol{x_0}$ and the discretized controls $\boldsymbol{q}$. We want to obtain the sensitivities

of the states with regard to the other states and controls, the so-called WRONSKI-matrices:

$$\boldsymbol{G}(t_i, t_0) \stackrel{\text{def}}{=} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x_0}} \boldsymbol{x}(t_i; \boldsymbol{q}),$$
$$\boldsymbol{G_q}(t_i, t_0) \stackrel{\text{def}}{=} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{q}} \boldsymbol{x}(t_i; \boldsymbol{q}).$$

Through differentiation, we obtain them as the solutions of the following IVPs:

$$\dot{\boldsymbol{G}}(t\ , t_0) = \boldsymbol{f_x}(t, \boldsymbol{x}(t), \boldsymbol{q})\, \boldsymbol{G}(t, t_0),$$
$$\boldsymbol{G}(t_0, t_0) = \boldsymbol{I},$$

and

$$\dot{\boldsymbol{G}}_{\boldsymbol{q}}(t\ , t_0) = \boldsymbol{f_x}(t, \boldsymbol{x}(t), \boldsymbol{q})\, \boldsymbol{G_q}(t, t_j) + \boldsymbol{f_q}(t, \boldsymbol{x}(t), \boldsymbol{q}),$$
$$\boldsymbol{G_q}(t_0, t_0) = \boldsymbol{0}.$$

However, one has to be careful to use the same integration scheme for the *variational ODE* as for the original ODE. Otherwise, the derivative is only a disturbed derivative. Notice that the right-hand sides of the sensitivity system can be efficiently computed with the forward mode of automatic differentiation.
The analogous *internal numeric differentiation* differentiates the discretized ODE with respect to the discretized states $\boldsymbol{s}$. The approach stems from BOCK's work in [28]. Notice that there is a difference between the presented *differentiate-then-discretize* scheme and the *discretize-then-differentiate* scheme, where instead of $\boldsymbol{x}$ its discretized version coming out of the integrator is differentiated. This difference vanishes if the ODE is not approximated but solved exactly. Discussions of additional cases in which the two also commutate for the approximated solution are e.g. given in [15, 29].
Usually, the *internal numeric differentiation* approach is implemented, since it guarantees to get the correct derivatives of the discretized system. Modifications of this method are e.g. presented in [40, 61].

Adjoint equation approach

The idea of the *adjoint equation* stems from the elimination of the sensitivities in the LAGRANGIAN relaxation of the ODE. It is used to calculate directional derivatives of the states with respect to the initial states and with respect to the discretized controls

$$\boldsymbol{x}_{\boldsymbol{d}}(t) \stackrel{\text{def}}{=} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x_0}} \boldsymbol{x}(t; \boldsymbol{q})\, \boldsymbol{d},$$
$$\boldsymbol{q}_{\boldsymbol{d}}(t) \stackrel{\text{def}}{=} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{q}} \boldsymbol{x}(t; \boldsymbol{q})\, \boldsymbol{d}$$

in direction $\boldsymbol{d} \in \mathbb{R}^{n_x}$. The adjoint system is in this case:

$$\begin{aligned}
\dot{\boldsymbol{x}}_{\boldsymbol{d}}(t) &= -\boldsymbol{f}_{\boldsymbol{x}}(t, \boldsymbol{x}(t), \boldsymbol{q})^T \boldsymbol{x}_{\boldsymbol{d}}(t), && t \in [t_0, t_f], \\
\boldsymbol{x}_{\boldsymbol{d}}(t_f) &= \boldsymbol{d}, \\
\dot{\boldsymbol{q}}_{\boldsymbol{d}}(t) &= -\boldsymbol{f}_{\boldsymbol{q}}(t, \boldsymbol{x}(t), \boldsymbol{q})^T \boldsymbol{d}, && t \in [t_0, t_f], \\
\boldsymbol{q}_{\boldsymbol{d}}(t_f) &= \boldsymbol{0}.
\end{aligned}$$

The adjoint approach is better suited if the problem has more degrees of freedom than states. Instead of computing the whole JACOBIAN and then a matrix-vector-product, the directional derivative is directly computed. Notice that the reverse mode of automatic differentiation can be used to efficiently calculate the right-hand side of the adjoint system.
As with the sensitivity equation approach, one has to be careful in the usage of this adjoint formalism with discretized systems such that the principle of internal numeric differentiation is satisfied, cf. [30]. The application of the formalism to the discretized system instead of the continuous system, as done above, directly satisfies this principle and ensures that the derivatives calculated are the true derivatives of the evaluated system. This is the case if the integration scheme for the ODE system is the adjoint of the integration scheme of the adjoint system.

2.4.5 Optimizer

The optimizer is the algorithm to solve the resulting finite-dimensional optimization problem. The applicable options naturally depend on the structure of the problem. For the continuous OCP, the resulting problem usually is a nonlinear problem. There are two prominent strategies to solve those: *interior point methods* and *active set methods* (e.g. the Sequential Quadratic Programming (SQP) method).
In a general setting, interior point methods are the methods of choice since they can be implemented very efficiently, cf. [182]. However, if similar problems have to be solved more than once, the *warm starting* possibilities of active-set methods become advantageous, cf. [120]. This is the case if the resulting problem is a MINLP, cf. [65, 84], or in a predictive control perspective, cf. [11]. SQP methods can also take advantage of separable structures through block updates for the approximation of the HESSIAN. These separable structures occur automatically in OCPs through the separation into different time intervals.
Historically, the SQP method has been prevalent in the OCP context, for both collocation methods [26, 90, 171] and multiple-shooting methods [33].

3 Mixed–Integer Optimal Control – Modeling and Relaxation

This chapter treats MIOCPs from the point of view of problem formulation and modeling for direct methods. It also briefly treats different solution approaches. The first Section 3.1 covers two general formulations of the problem class and establishes commonly used expressions. The following two Sections 3.2 and 3.3 briefly present the adjustments needed for two widely known approaches to solve MIOCPs, namely *indirect methods* and *dynamic programming*. However, they are only presented for the sake of completeness since they are not used to solve problems in the numerical results Chapter 5. Section 3.4 treats *direct methods* to solve MIOCPs. Then in the following Sections 3.5–3.9, we describe how different relaxed formulations can be obtained for both the ODE and constraints of a disjunction. These approaches are compared in Section 3.10. Then, we give insight into different ways to model switching constraints. Last, in Section 3.12, we cover the most important ways to solve MINLPs, which emerge from the discretization of the MIOCPs.

3.1 Problem formulation

We give an introduction to the MIOCP class that we are interested in. This class is model-based nonlinear optimal control that includes switching decisions, which are to be optimized together with piecewise continuous controls. We present two different ways of formulating a MIOCP and define some expressions. We start with a specific case of a MIOCP in ODEs of the following form:

Definition 3.1
A Mixed-Integer Optimal Control Problem is a continuous optimal control problem with integer feasibility requirement on a subset of the control trajectories of the form:

$$
\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{v}(\cdot)} \quad & \Phi(\boldsymbol{x}(t_f)) && && (3.1)\\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t,\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{v}(t)), && t \in [t_0,t_f],\\
& \boldsymbol{0} \leqslant \boldsymbol{c}(t,\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{v}(t)), && t \in [t_0,t_f],\\
& \boldsymbol{0} = \boldsymbol{r}^{eq}((t_i,\boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}), && \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0,t_f],\\
& \boldsymbol{0} \leqslant \boldsymbol{r}^{in}((t_i,\boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}), && \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0,t_f],\\
& \boldsymbol{u}(t) \in \mathcal{U}, && t \in [t_0,t_f],
\end{aligned}
$$

$$\boldsymbol{v}(t) \in \Omega \subset \mathbb{R}^{n_v}, \qquad t \in [t_0, t_f], |\Omega| = n_\omega < \infty,$$

where we determine a dynamic process $\boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^{n_x}$ *on the time horizon* $[t_0, t_f] \subset \mathbb{R}$ *described by a system of ODEs with the right-hand side function* $\boldsymbol{f} : [t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$. *This system is affected by a continuous, vector-valued control function* $\boldsymbol{u} : [t_0, t_f] \to \mathbb{R}^{n_u}$, *and another vector-valued control function, which attains only values from a finite discrete set* $\boldsymbol{v} : [t_0, t_f] \to \Omega \stackrel{\text{def}}{=} \{\boldsymbol{v^1}, \boldsymbol{v^2}, \ldots, \boldsymbol{v^{n_\omega}}\} \subseteq \mathbb{R}^{n_v}$ *with cardinality* $|\Omega| = n_\omega < \infty$. *The controls are to be determined such that we minimize a performance index* $\Phi : \mathbb{R}^{n_x} \to \mathbb{R}$ *and satisfy path constraints* $\boldsymbol{c} : [t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_c}$ *and coupled point constraints* $\boldsymbol{r} = (\boldsymbol{r}^{eq}, \boldsymbol{r}^{in}) : \left([t_0, t_f] \times \mathbb{R}^{n_x}\right)^{m+1} \to \mathbb{R}^{n_r}$ *on a finite number of* $m+1$ *grid points* $\{t_i\}_{0 \leqslant i \leqslant m} \subset [t_0, t_f]$. △

The addition of integrality usually means that the system can run in different operation modes. The following disjunctive formulation captures this interpretation of different modes in a clearer way:

Definition 3.2
A Mixed-Integer Optimal Control Problem in disjunctive form is a continuous optimal control problem with n_ω *different operation modes of the system. Their status is represented by Boolean control functions* $\boldsymbol{\omega} : [t_0, t_f] \to \{0, 1\}^{n_\omega}$.

$$\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), \boldsymbol{\omega}(\cdot)} \quad \Phi(\boldsymbol{x}(t_f)) \tag{3.2}$$

$$\text{s.t.} \quad \bigvee_{1 \leqslant i \leqslant n_\omega} \begin{bmatrix} \omega_i(t) = 1 \\ \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v^i}) \\ \mathbf{0} \leqslant \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v^i}) \end{bmatrix}, \qquad t \in [t_0, t_f], \tag{3.3}$$

$$\begin{aligned} \mathbf{0} &= \boldsymbol{r}^{eq}((\boldsymbol{x}(t_i))_{0 \leqslant i \leqslant m}), && \{t_i\}_{0 \leqslant i \leqslant m} \subset [t_0, t_f], \\ \mathbf{0} &\leqslant \boldsymbol{r}^{in}((\boldsymbol{x}(t_i))_{0 \leqslant i \leqslant m}), && \{t_i\}_{0 \leqslant i \leqslant m} \subset [t_0, t_f], \\ \boldsymbol{u}(t) &\in \mathcal{U}, && t \in [t_0, t_f], \\ \boldsymbol{\omega}(t) &\in \{0, 1\}^{n_\omega}, && t \in [t_0, t_f], \end{aligned}$$

where the different terms are used as in the Definition 3.1 above, except the addition of said $\boldsymbol{\omega}$, *which models the choice of the mode.* △

The disjunction (3.3) was given such that the close connection between the two problems (3.1) and (3.2) is visible. A more common approach would be to formulate the disjunction as

$$\bigvee_{1 \leqslant i \leqslant n_\omega} \begin{bmatrix} \omega_i(t) = 1 \\ \dot{\boldsymbol{x}}(t) = \boldsymbol{f^i}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \mathbf{0} \leqslant \boldsymbol{c^i}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \end{bmatrix}, \qquad t \in [t_0, t_f], \tag{3.4}$$

where a different function $\boldsymbol{f}^{\boldsymbol{i}} : [t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and a different function $\boldsymbol{c}^{\boldsymbol{i}} : [t_0, t_f] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ are evaluated for each mode i. This makes the OC approach more natural in comparison to the IC approach, which is further discussed in Sections 3.5 and 3.6.

Definition 3.3 (Binary control)
We use the term integer control $\boldsymbol{v}(\cdot)$*, when its image space is a discrete set, i.e.,*

$$\boldsymbol{v}(t) \in \Omega \stackrel{\text{def}}{=} \{\boldsymbol{v}^{\boldsymbol{1}}, \boldsymbol{v}^{\boldsymbol{2}}, \ldots, \boldsymbol{v}^{\boldsymbol{n_\omega}}\}, \tag{3.5}$$

with $\exists \varepsilon > 0, \forall i \neq j : \left|\left|\boldsymbol{v}^{\boldsymbol{i}} - \boldsymbol{v}^{\boldsymbol{j}}\right|\right| > \varepsilon$*. And we use the term* binary control *for the special case of*

$$\boldsymbol{v}(t) \in \{0, 1\}^{n_\omega}. \tag{3.6}$$

△

We use the expression *relaxed* whenever the restriction $\boldsymbol{v}(\cdot) \in \Omega$ is relaxed to a superset of Ω, in particular to its convex hull. Similarly, $\boldsymbol{\omega}(\cdot) \in \{0, 1\}^{n_\omega}$ is relaxed to the superset $[0, 1]^{n_\omega}$.
Analogously to the OCP case, different objective formulations can be incorporated and the general BOLZA functional can be used.
The reformulations and studies presented in this chapter can also be applied to more general algebraic systems with the algebraic states $\boldsymbol{z}(t)$ and an explicit algebraic system of index 1 that replaces the ODE:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)), && t \in [t_0, t_f], \\ \boldsymbol{0} &= \boldsymbol{g}(t, \boldsymbol{x}(t), \boldsymbol{z}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)), && t \in [t_0, t_f]. \end{aligned}$$

SAGER's theoretical basis can be translated to this setting, cf. [79]. As a direct consequence, our results can be adapted for index 1 DAE systems. However, for higher index models, already the solution of the relaxed OCPs is hard and further research in the combination of the two might be needed.
More general hybrid systems with state dependent switches between the system's modes can be modeled with implicit switches as described in [38]. Together with these implicit switches, the system mode is tracked implicitly and a multistage OCP as described in Section 2.1 is obtained. Alternatively, they can be modeled through the coupled path and control constraints:

$$\boldsymbol{0} \leqslant \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)), \qquad t \in [t_0, t_f],$$

where some settings of $\boldsymbol{v}^{\boldsymbol{i}}$ are only allowed in the state space that corresponds to this mode. These systems include the network overflow problem from Section 5.3.5 and might, e.g., also include ground contact models in robotics.
In addition to the constraints given, there may be constraints that change the domain of the integer control functions. Instead of only limiting the choices for each point in time, there may be additional constraints that consider the whole trajectory as, e.g., switching

constraints, which limit the amount of total mode switches a control is allowed to take. We can abbreviate this by requiring for the integer control:

$$\begin{aligned} \boldsymbol{\omega}(\cdot) \in \Omega_\omega \quad &\subseteq \mathcal{L}^1([t_0, t_\mathrm{f}]), \\ \boldsymbol{\nu}(\cdot) \in \Omega_\nu \quad &\subseteq \mathcal{L}^1([t_0, t_\mathrm{f}]). \end{aligned} \tag{3.7}$$

There are different approaches to solve these MIOCPs. In the following, we describe the adjustments needed for the *indirect approach*, the *dynamic programming approach* and the *direct approach*, which are explained for MIOCPs in Sections 3.2–3.4. Afterwards, we cover some issues concerning the integrality relaxation in Sections 3.5–3.9 needed for the direct approach and we describe different ways to model system switches as additional combinatorial constraints in Section 3.11.

3.2 Indirect approach

One approach to solve MIOCPs is the *indirect approach*. The indirect approach for OCPs is explained in Section 2.2. There, the *Maximum Principle* was formulated in a *global* way such that it is easily generalized to the integral case, cf. [78, 176]. The integrality constraint is just added to the inner minimization problem (2.15). This is possible since no assumptions were made on the set $\mathcal{U}$, which is the set of feasible controls. It may be non-convex and even disjoint. In case of disjoint feasible regions, the behavior of the inner minimization problem can be captured through the use of switching functions, i.e., the process is tracked in each region separately and it is switched depending on which region realizes the minimum, cf. [32]. This approach is called *competing* HAMILTONIANS.
Furthermore, for the extension to control functions that have additional global properties, e.g., a limited number of switches, typically additional states are needed that record the progress of these properties. Nonetheless, these additions can be made.

3.3 Dynamic programming

A second approach to solve MIOCPs is *dynamic programming*. In Section 2.3, the general dynamic programming approach is explained for OCPs. It can very easily be adapted to integer controls and the integrality constraint even is an advantage for this approach since all other controls have to be discretized and the natural discretization due to integrality is much coarser than the discretization typically employed for continuous controls. Therefore, the computational effort is less than for the relaxed problem. However, the *curse of dimensionality* still applies to the integral case and the approach is only viable for low numbers of differential states and controls.
The application of dynamic programming to a truck problem similar to the one presented in Section 5.4 is carried out in [42, 93].

3.4 Direct approach

Large parts of the contents of the following Sections 3.4–3.10 are based on our paper

[103] M. JUNG, C. KIRCHES AND S. SAGER, *On Perspective Functions and Vanishing Constraints in Mixed-Integer Nonlinear Optimal Control,* in Facets of Combinatorial Optimization – Festschrift for MARTIN GRÖTSCHEL, M. JÜNGER AND G. REINELT, eds., Springer Berlin Heidelberg, 2013, pp. 387–417.

In Section 2.4, the *direct approach* and its components are described for OCPs. As in the continuous control setting, also for the direct approach in the MIOCP setting, the states, controls, constraints and objective function have to be discretized. The obtained finite dimensional problem is an MINLP, the major difference to the continuous NLP is the addition of integrality constraints on some controls. Some of the possibilities to solve the resulting MINLP are described in Section 3.12. For the most part, the discretization can be done in the same manner as described in Section 2.4, but there are certain differences.
As the discretization principle is the same, also the *direct approach* for MIOCPs requires different grids to be specified. Assume for notational simplicity that the discretization of the ODE, the controls, and the constraints all happen on the same grid

$$\mathbb{G}_m \overset{\text{def}}{=} \{t_0 < t_1 < \ldots < t_m = t_\text{f}\}$$

of $m+1$ points with step sizes

$$\Delta t_i \overset{\text{def}}{=} t_{i+1} - t_i$$

and maximum step size

$$\Delta t \overset{\text{def}}{=} \max_{0 \leqslant i \leqslant m-1} \{\Delta t_i\}.$$

Most solvers for MINLPs need a differentiable function of the ODE $\boldsymbol{f}(\cdot)$ and the constraints $\boldsymbol{c}(\cdot)$: the problem is usually relaxed by dropping the integrality constraint and the resulting NLPs are solved with an NLP solver that requires second order differentiability. However, both functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{c}(\cdot)$ are not necessarily defined on intermediate values between the different $\boldsymbol{v}^i$ as in Definition 3.1. Therefore, one major difference is that reformulations of the functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{c}(\cdot)$ are needed such that a differentiable formulation is obtained. The possible reformulations differ between equations and inequalities and we present different approaches and study their characteristics, i.e., tightness, numerical stability etc., in the following sections.
There are lots of options to employ binary or integer variables such that for integral points exactly one mode of the system is chosen but the reformulations result in different relaxations.

Since the structure of the ODE system is distinct from the general structure of the constraints $\boldsymbol{c}(\cdot)$, we study them separately. Possible ways to reformulate the ODEs

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)), \qquad \boldsymbol{v}(t) \in \{\boldsymbol{v}^1, \boldsymbol{v}^2, \dots, \boldsymbol{v}^{n_\omega}\}, \tag{3.8}$$

with integer variables are presented in the following as the IC approach, the OC approach and the perspective approach, which includes variable lifting. Notice that this constraint is almost always non-convex as it is a nonlinear equation. Only if $\boldsymbol{f}(\cdot)$ is linear, the system may become convex, cf. [139].

As with the ODEs, there are also different, valid options to employ binary or integer formulations for more general constraints

$$\boldsymbol{0} \leqslant \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)), \qquad \boldsymbol{v}(t) \in \{\boldsymbol{v}^1, \boldsymbol{v}^2, \dots, \boldsymbol{v}^{n_\omega}\}. \tag{3.9}$$

The first variants are the same as for the ODEs, i.e., IC, OC and the formulation with perspective functions. However, we present the additional option of vanishing constraints. The main trait is that in the integer points, the corresponding active mode's formulation holds. Depending on the reformulation, there might be different kinds of constraint qualifications that are violated in a subset of feasible points; we address this issue where it arises.

The integrality relaxation used in our descriptions usually drops the integrality constraint while still using the variable's bounds, e.g., a binary variable $y \in \{0, 1\}$ becomes a continuous variable in $y \in [0, 1]$. However, there exist ways to directly reformulate the integrality constraint with usually non-convex constraints and obtain a correct representation of the MINLP as an NLP as explained in [170].

In the following, we only present the reformulations for inequality constraints. However, most reformulations are possible for equations as well; we explain the differences as they emerge.

3.5 Inner Convexification

One of the most natural approaches to unify the ODEs (3.8) as well as path constraints (3.9) for the different modes is to replace all occurrences of $\boldsymbol{v}(t)$ with a function $\boldsymbol{g}(\cdot)$ that realizes the different values $\boldsymbol{v}^i$ for integral values of additionally introduced integral variables. The unified ODE is then obtained as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{g}(\cdot)). \tag{3.10}$$

One such approach would be to use binary variables $\boldsymbol{\omega}$ connected with a SOS2-constraint,

which are used to model piecewise linear interpolations. They use the specifications

$$\boldsymbol{g}(t, \boldsymbol{\omega}(t), \alpha(t)) = \alpha(t)\, \boldsymbol{v}^{\boldsymbol{i}} + (1 - \alpha(t))\, \boldsymbol{v}^{\boldsymbol{i+1}}, \qquad \text{if } \omega_i(t) = 1,$$

i.e., if $\omega_i(t) = 1$ then $\boldsymbol{g}(\cdot)$ linearly interpolates $\boldsymbol{v}^{\boldsymbol{i}}$ and $\boldsymbol{v}^{\boldsymbol{i+1}}$. Additional variables $\boldsymbol{\mu}$ are needed to model this behavior correctly:

$$\begin{aligned}
0 = \mu_0 = \mu_{n_\omega} &= \omega_{n_\omega}, \\
0 \leqslant \mu_i(t) &\leqslant \omega_i(t) \leqslant 1, && 1 \leqslant i \leqslant n_\omega - 1, \\
1 &= \sum_{i=1}^{n_\omega - 1} \omega_i(t), \\
\boldsymbol{g}(t, \boldsymbol{\omega}(t), \boldsymbol{\mu}(t)) &= \sum_{j=1}^{n_\omega} \left(\omega_i(t) - \mu_i(t) + \mu_{i-1}(t)\right)\, \boldsymbol{v}^{\boldsymbol{i}}.
\end{aligned}$$

A second approach is to use a convex combination of the values $\boldsymbol{v}^{\boldsymbol{i}}$ with binary variables $\boldsymbol{\omega}$:

$$\begin{aligned}
1 &= \sum_{i=1}^{n_\omega} \omega_i(t), \\
\boldsymbol{g}(t, \boldsymbol{\omega(t)}) &= \sum_{i=1}^{n_\omega} \omega_i(t)\, \boldsymbol{v}^{\boldsymbol{i}}.
\end{aligned}$$

There are other possible reformulations that use fitted smooth convex functions $\boldsymbol{g}(\cdot)$ as e.g. suggested in [76].

The method is named Inner Convexification (IC) since the function $\boldsymbol{f}(\cdot)$ itself is not changed but the input control is replaced by a convex term. Naturally, if $\boldsymbol{f}(\cdot)$ was convex, this property would still hold for its reformulated version. However, for this property to hold, $\boldsymbol{f}(\cdot)$ has to be convex also with regard to the controls $\boldsymbol{v}(t)$. The reformulation's strongest property is the small computational effort, the reformulated right-hand side still only requires one evaluation of the function $\boldsymbol{f}(\cdot)$ and is hence as fast as possible. Unfortunately, this reformulation has some negative properties, which are described in the following. To be able to execute this reformulation one has to have access to and be able to change the formulation of $\boldsymbol{f}(\cdot)$ and it may not be treated as a *black box*. This reformulation may also yield a new function with singularities in between the discrete values $\boldsymbol{v}^{\boldsymbol{1}}, \ldots, \boldsymbol{v}^{\boldsymbol{n_\omega}}$ since their combination could produce division by 0 or something similar. For algebraic systems, the DAE index could change through this reformulation as described in [12]. Another unfavorable property is that the integrality gap between the optimal solution of the relaxed problem and the optimal integral solution can become arbitrarily large since the reachable set may become much larger through the use of intermediate values, cf. [152, Section 4.2].

For the constraints for the different modes, the method remains the same. They are unified

in one single constraint. Again, each occurrence of the discrete controls $\boldsymbol{v}(t)$ is replaced by a convex function $\boldsymbol{g}(t, \boldsymbol{\omega}(t))$ of the newly introduced integer controls $\boldsymbol{\omega}(t)$, which attains the values $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^{n_\omega}$ for integral choices of $\boldsymbol{\omega}(t)$. Usually, the same functions $\boldsymbol{g}(\cdot)$ is chosen for the constraints and the ODE. The IC constraint formulation is

$$\mathbf{0} \leqslant \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{g}(t, \boldsymbol{\omega}(t))), \tag{3.11}$$

such that for $\boldsymbol{\omega}(t) \in \{0,1\}^{n_\omega}$ there exists $0 \leqslant i \leqslant n_\omega$ with $\boldsymbol{g}(t, \boldsymbol{\omega}(t)) = \boldsymbol{v}^i$.

3.6 Outer Convexification

The OC technique was proposed in [152] and further investigated in [157, 160] to relax the MIOCP to an ordinary OCP. To reformulate the ODE (3.8) of the different modes, for every mode's element $\boldsymbol{v}^i$ of Ω, a binary control function $\omega_i(\cdot)$ is introduced, which represents the choice of the mode. Then, with these variables, the relaxed version of the ODE is a convex combination of the different modes' ODEs:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \sum_{i=1}^{n_\omega} \omega_i(t)\, \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i), \\ 1 &= \sum_{i=1}^{n_\omega} \omega_i(t), \\ \omega_i(t) &\in \{0,1\}. \end{aligned} \tag{3.12}$$

The resulting problem is convexified with respect to the binary controls. This is a convex combination for the different right-hand sides $\boldsymbol{f}^i(\cdot)$ of the disjunctive formulation. For the reformulated problem, we use the terms *partial Outer Convexification* to stress that convexification only occurs on the binary controls and *control-affine systems* when we neglect the effects of the continuous controls and focus on the effects of the linearly entering binary controls. The relaxation replaces the binary constraint by

$$\omega_i(t) \in [0,1].$$

This reformulation is only straightforward if the different system modes have the same corresponding differential states as in our definitions.

The main effort needed for one right-hand side evaluation of (3.12) is n_ω evaluations of $\boldsymbol{f}(\cdot)$ for the different modes. This may seem quite a lot since any feasible combination of integer variables may require its own mode $\boldsymbol{v}^i$. However, integer controls often decouple depending on the separability properties of $\boldsymbol{f}(\cdot)$. This can greatly reduce the number n_ω of feasible combinations, i.e., different system modes, cf. e.g. [82, 153]. An example for this

separability is linear coupling. Assume it holds

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}_1(\cdot, \boldsymbol{v}_1(t)) + \boldsymbol{f}_2(\cdot, \boldsymbol{v}_2(t)), \qquad \boldsymbol{v}_1(t) \in \Omega_1, \boldsymbol{v}_2(t) \in \Omega_2.$$

Then, instead of enumerating all possible modes in $\Omega_1 \times \Omega_2$ for all $t \in [t_0, t_\mathrm{f}]$:

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \sum_{i=1}^{n_{\omega 1}} \sum_{j=1}^{n_{\omega 2}} \left(\boldsymbol{f}_1(\cdot, \boldsymbol{v}_1^i) + \boldsymbol{f}_2(\cdot, \boldsymbol{v}_2^j) \right) \omega_{i,j}(t), && \boldsymbol{v}_1^i \in \Omega_1, \boldsymbol{v}_2^j \in \Omega_2, \\
1 &= \sum_{i=1}^{n_{\omega 1}} \sum_{j=1}^{n_{\omega 2}} \omega_{i,j}(t), \\
\omega_{i,j}(t) &\in \{0, 1\},
\end{aligned}$$

which would lead to $n_\omega = n_{\omega 1} n_{\omega 2}$ modes, it is possible to just combine them independently through

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \sum_{i=1}^{n_{\omega 1}} \boldsymbol{f}_1(\cdot, \boldsymbol{v}_1^i)\, \omega_{1,i}(t) + \sum_{j=1}^{n_{\omega 2}} \boldsymbol{f}_2(\cdot, \boldsymbol{v}_2^j)\, \omega_{2,j}(t), && \boldsymbol{v}_1^i \in \Omega_1, \boldsymbol{v}_2^j \in \Omega_2, \\
1 &= \sum_{i=1}^{n_{\omega 1}} \omega_{1,i}(t), \\
1 &= \sum_{j=1}^{n_{\omega 2}} \omega_{2,j}(t), \\
\omega_{i,j}(t) &\in \{0, 1\}.
\end{aligned}$$

Both formulations essentially result in the same type of problem, i.e., a control-affine system with SOS1-constraints, but the second formulation needs a lot less variables to obtain the formulation. The two formulations have the exact same solutions with regard to the states and continuous controls. This situation is found for most practical applications where binary control functions usually enter linearly, e.g., to indicate whether a term is present or not. Additionally, large parts of $\boldsymbol{f}(\cdot)$ are usually the same for all system modes and can be evaluated only once to save computational overhead.

In contrast to the *IC's relaxation*, the integrality gap between an integer solution to the OC (3.12) and a solution of its relaxed formulation – where $\omega_i(t) \in [0, 1]$ – is bounded linearly by the size Δt of the discretized time grid. This is proved in [157] and also presented in Section 4.2 since it establishes the basis for the approximation results of Chapter 4.

For the constraints $\boldsymbol{c}(\cdot)$, the same procedure is used. The residuals of the constraints are evaluated separately for each possible mode and the resulting constraint is posed as the convex

combination of these residuals, i.e.,

$$\begin{aligned} \mathbf{0} &\leqslant \sum_{i=1}^{n_\omega} \omega_i(t)\, \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i), \\ 1 &= \sum_{i=1}^{n_\omega} \omega_i(t), \\ \omega_i(t) &\in \{0, 1\}. \end{aligned} \tag{3.13}$$

Notice that the convex combinations should be chosen such that constraints of the same type are combined. For constraint types that exist only on a subset of modes, special measures must be taken that ensure that the constraints are inactive while the corresponding modes are inactive.
The relaxed formulation replaces $\omega_i(t) \in \{0, 1\}$ through $\omega_i(t) \in [0, 1]$. The separability reformulations should also be applied to the constraints in order to introduce as little variables as necessary.
As the constraints are aggregated into one single constraint, *compensatory effects* may lead to feasible residuals for fractional values of the convex multipliers as observed in [111].

Proposition 3.1 (Feasible region)
There exists $\boldsymbol{\omega}(t)$ *such that* $(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\omega}(t))$ *is feasible for the relaxed version of the constraints* (3.13) *if and only if there exists an index* $1 \leqslant i \leqslant n_\omega$ *such that*

$$0 \leqslant \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i). \qquad \triangle$$

The proposition states that the projection onto the $(\boldsymbol{x}, \boldsymbol{u})$-space of the feasible set of the problem after OC is exactly the union of all feasible sets of the disjunction.

Proof The backward direction is trivial:
If it exists $1 \leqslant i \leqslant n_\omega$ such that

$$0 \leqslant \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i),$$

then the following choice of $\boldsymbol{\omega}(t)$ satisfies (3.13):

$$\omega_j(t) = \begin{cases} 1, & \text{if } j = i, \\ 0, & \text{else.} \end{cases}$$

On the other hand, assume $\forall 1 \leqslant i \leqslant n_\omega$:

$$\boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i) < 0,$$

then for any choice $\boldsymbol{\omega}(t) \in [0,1]^{n_\omega}$ with

$$\sum_{i=1}^{n_\omega} \omega_i(t) = 1,$$

it holds:

$$\begin{aligned}\sum_{i=1}^{n_\omega} \omega_i(t)\, \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i) &\leqslant \sum_{i=1}^{n_\omega} \omega_i(t) \max_{j=1,\dots,n_\omega} \left\{\boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^j)\right\} \\ &= \max_{j=1,\dots,n_\omega} \left\{\boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^j)\right\} \sum_{i=1}^{n_\omega} \omega_i(t) < 0.\end{aligned}$$

This means, there exists no $\boldsymbol{\omega}(t)$ such that $(\boldsymbol{\omega}(t), \boldsymbol{x}(t), \boldsymbol{u}(t))$ is feasible for the relaxed constraints (3.13). □

3.7 Big-M formulation

This technique is only used to reformulate inequality constraints and not the ODE. The Big-M formulation stems from the integer programming community and is often used to reformulate nonlinear constraints into linear ones. The constraints are linearly relaxed through the binary variables corresponding to each mode. The bounds $\boldsymbol{M}^i \in \mathbb{R}^{n_c}$ have to be lower bounds that hold for all possible settings, i.e., $\forall t \in [t_0, t_f]$, $\boldsymbol{x}(\cdot)$ reachable and feasible $\boldsymbol{u}(\cdot)$:

$$\boldsymbol{M}^i \leqslant \boldsymbol{c}^i(t, \boldsymbol{x}(t), \boldsymbol{u}(t)).$$

The relaxed constraints become

$$\boldsymbol{M}^i \left(1 - \omega_i(t)\right) \leqslant \boldsymbol{c}^i(t, \boldsymbol{x}(t), \boldsymbol{u}(t)). \tag{3.14}$$

Remark 3.1
It is well known that the tightness of the constraints depends strongly on the choice of $\boldsymbol{M}^i$. However, even with the strongest possible choice the constraints can remain quite weak. At best one reaches the tightness of the convex hull formulation described in the perspective formulation section, cf. [99]. However, an advantage of the formulation is that it is easy to formulate, does not introduce additional non-convexities, and does not increase the problem's size.

3.7.1 Mixed Logical Dynamical systems

The MLD systems – as used by Bemporad and Morari and realized in the Multi-Parametric Toolbox (MPT) – use an approach, which results in a Big-M formulation. They automatically

combine the inherent logic of dynamic control problem with logical decisions involved in the linear setting, where the state transition, possibly present algebraic constraints, and path and control constraints of the discretized system are all linear. They employ new binary variables ω to capture the logics of the system and provide a set of inequalities to combine these binary variables with the dynamic structure of the continuous states. In [18], the formalism is described and we give a brief example in the following.

Let us consider the following equivalence condition of a binary variable ω and a linear function $\boldsymbol{g}(\cdot)$ as a constraint with the connection of the two being:

$$[\omega = 1] \longleftrightarrow [\boldsymbol{g}(\boldsymbol{x}) \geqslant \boldsymbol{0}].$$

This condition is equivalent to the following linear inequalities:

$$\begin{aligned} &\boldsymbol{g}(\boldsymbol{x}) \geqslant \boldsymbol{m}\,(1-\omega), \\ &\boldsymbol{g}(\boldsymbol{x}) \leqslant (\boldsymbol{M}+\boldsymbol{\epsilon})\,\omega - \boldsymbol{\epsilon}, \end{aligned}$$

where

$$\begin{aligned} m_i &\stackrel{\text{def}}{=} \min_{\boldsymbol{x}\in\mathcal{D}} g_i(\boldsymbol{x}), \\ M_i &\stackrel{\text{def}}{=} \max_{\boldsymbol{x}\in\mathcal{D}} g_i(\boldsymbol{x}), \end{aligned}$$

and $\boldsymbol{\epsilon}$ is a vector of small tolerance parameters beyond which the constraint is considered to be violated. Since $\boldsymbol{g}(\cdot)$ is a linear function and if we assume that the variables $\boldsymbol{x}$ belong to a bounded domain $\mathcal{D}$, then $\boldsymbol{m}$ and $\boldsymbol{M}$ can be computed or at least under- and overestimated, respectively. This is sufficient to guarantee the equivalence between the logic statement and the set of inequalities. As usual with Big-M-formulations, the reformulation computationally works better, the smaller the entries of $\boldsymbol{M}$ and the larger the entries of $\boldsymbol{m}$, as long as they remain valid bounds, cf. [185].

Under some assumptions, MLD systems have been shown to be equivalent to other system modeling formats including *linear complementarity* systems, *extended linear complementarity systems*, *piecewise affine systems*, and *max-min-plus-scaling* systems, cf. [92].

3.8 Vanishing and complementarity constraints

Instead of aggregating the different mode's constraints as done in the IC and OC approaches, this formulation accounts for the different modes' constraints independently. This is realized

through the addition of one constraint set per mode, the constraints (3.9) are replaced with

$$\begin{aligned} &0 \leqslant \omega_i(t)\, \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i), \qquad 1 \leqslant i \leqslant n_\omega, \\ &1 = \sum_{i=1}^{n_\omega} \omega_i(t). \end{aligned} \tag{3.15}$$

Notice that each mode's constraints are enforced if the corresponding multiplier is nonzero. The obtained NLPs or MINLPs lose constraint qualification and become Mathematical Programs with Vanishing Constraints (MPVCs), if only inequality constraints are present – or Mathematical Programs with Complementarity Constraints (MPCCs), if also equations are present that can be relaxed to inequalities if the corresponding mode is inactive. We mainly cover MPVC but try to also give some comments on MPCC.

Definition 3.4 (MPCC, MPVC)
An NLP

$$\begin{aligned} \min_{\boldsymbol{y} \in \mathbb{R}^{n_y}} \quad & \Phi(\boldsymbol{y}) && && (3.16) \\ \text{s.t.} \quad & 0 = r_i(\boldsymbol{y}), && i \in \mathcal{E}, \\ & 0 \leqslant r_i(\boldsymbol{y}), && i \in \mathcal{I}, \\ & \boldsymbol{0} \leqslant \boldsymbol{g}(\boldsymbol{y}), \\ & \boldsymbol{0} \leqslant \boldsymbol{h}(\boldsymbol{y}), \\ & \boldsymbol{g}(\boldsymbol{y}) \perp \boldsymbol{h}(\boldsymbol{y}), && && (3.17) \end{aligned}$$

with twice continuously differentiable functions Φ, $\boldsymbol{r}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ is called a Mathematical Program with Complementarity Constraints (MPCC) (3.17)*, cf. [14]. The $\perp$ operator represents complementarity and means that for each component i it holds $g_i(\boldsymbol{y}) = 0$ or $h_i(\boldsymbol{y}) = 0$.*
An NLP

$$\begin{aligned} \min_{\boldsymbol{y} \in \mathbb{R}^{n_y}} \quad & \Phi(\boldsymbol{y}) && && (3.18) \\ \text{s.t.} \quad & 0 = r_i(\boldsymbol{y}), && i \in \mathcal{E}, \\ & 0 \leqslant r_i(\boldsymbol{y}), && i \in \mathcal{I}, \\ & \boldsymbol{0} \leqslant \boldsymbol{h}(\boldsymbol{y}), \\ & \boldsymbol{0} \leqslant g_i(\boldsymbol{y})\, h_i(\boldsymbol{y}), && 1 \leqslant i \leqslant n_{gh}, && (3.19) \end{aligned}$$

with twice continuously differentiable functions Φ, $\boldsymbol{r}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ is called a Mathematical Program with Vanishing Constraints (MPVC) (3.19)*, cf. [3].* △

Remark 3.2 (Transformation MPVC into MPCC)
MPVCs can easily be transformed into MPCCs. A vector of slack variables $\boldsymbol{z}$ is used to relax

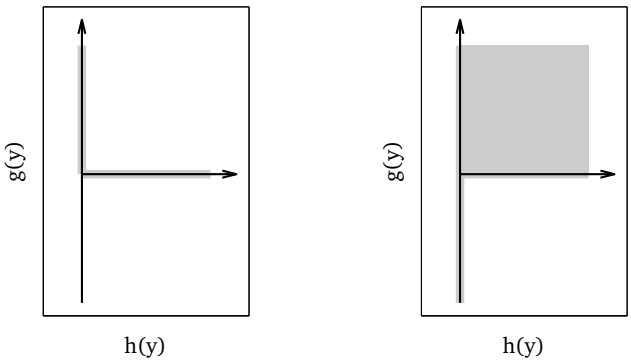


Figure 3.1: Feasible regions for complementarity constraints (3.17, left) and vanishing constraints (3.19, right).

the vanishing constraints $\boldsymbol{g}$ and complementarily combined with the indicator function $\boldsymbol{h}$:

$$\begin{aligned}
\min_{\boldsymbol{y}\in\mathbb{R}^{n_\mathrm{y}},\boldsymbol{z}\in\mathbb{R}^{n_\mathrm{gh}}} \quad & \Phi(\boldsymbol{y}) \\
\text{s.t.} \quad & 0 = r_i(\boldsymbol{y}), \qquad i \in \mathcal{E}, \\
& 0 \leqslant r_i(\boldsymbol{y}), \qquad i \in \mathcal{I}, \\
& \boldsymbol{0} \leqslant \boldsymbol{h}(\boldsymbol{y}), \\
& \boldsymbol{0} \leqslant \boldsymbol{z}, \\
& \boldsymbol{0} \leqslant \boldsymbol{g}(\boldsymbol{y}) + \boldsymbol{z}, \\
& \boldsymbol{z} \perp \boldsymbol{h}(\boldsymbol{y}).
\end{aligned}$$

The resulting problem can be treated with a Mathematical Program with Equilibrium Constraints (MPEC) algorithm as, e.g., [122, 148], which use interior point methods, or [102], which use an active set method. However, the resulting problem is degenerated as solutions are not unique in the slack variables and it is thus difficult for interior point methods to handle. Another disadvantage of this reformulation is that in a sense MPCC is a more difficult class of problems than MPVC – standard constraint qualifications are violated in each feasible point instead of a discrete set of points, cf. [50, 96, 164].

The following index sets for feasible points have special properties that are described later.

Definition 3.5 (Index sets)
Let $\bar{\boldsymbol{y}}$ be a feasible point of (3.16) *or* (3.18)*, the index set of active inequalities is defined as*

$$\mathcal{I}_r \stackrel{\text{def}}{=} \{i \mid r_i(\bar{\boldsymbol{y}}) = 0,\ i \in \mathcal{I}\}.$$

Let $\bar{\boldsymbol{y}}$ be a feasible point of the MPCC (3.16)*, the following index sets are defined and partition*

the set of complementarity constraint indices:

$$\begin{aligned}
\mathcal{I}_{0+} &\overset{\text{def}}{=} \{i \mid h_i(\bar{\boldsymbol{y}}) = 0,\ g_i(\bar{\boldsymbol{y}}) > 0\},\\
\mathcal{I}_{+0} &\overset{\text{def}}{=} \{i \mid h_i(\bar{\boldsymbol{y}}) > 0,\ g_i(\bar{\boldsymbol{y}}) = 0\},\\
\mathcal{I}_{00} &\overset{\text{def}}{=} \{i \mid h_i(\bar{\boldsymbol{y}}) = 0,\ g_i(\bar{\boldsymbol{y}}) = 0\}.
\end{aligned}$$

Let $\bar{\boldsymbol{y}}$ be a feasible point of the MPVC (3.18)*, the following index sets are defined and partition the set of vanishing constraint indices:*

$$\begin{aligned}
\mathcal{I}_{+0} &\overset{\text{def}}{=} \{i \mid h_i(\bar{\boldsymbol{y}}) > 0,\ g_i(\bar{\boldsymbol{y}}) = 0\},\\
\mathcal{I}_{++} &\overset{\text{def}}{=} \{i \mid h_i(\bar{\boldsymbol{y}}) > 0,\ g_i(\bar{\boldsymbol{y}}) > 0\},\\
\mathcal{I}_{00} &\overset{\text{def}}{=} \{i \mid h_i(\bar{\boldsymbol{y}}) = 0,\ g_i(\bar{\boldsymbol{y}}) = 0\},\\
\mathcal{I}_{0+} &\overset{\text{def}}{=} \{i \mid h_i(\bar{\boldsymbol{y}}) = 0,\ g_i(\bar{\boldsymbol{y}}) > 0\},\\
\mathcal{I}_{0-} &\overset{\text{def}}{=} \{i \mid h_i(\bar{\boldsymbol{y}}) = 0,\ g_i(\bar{\boldsymbol{y}}) < 0\}.
\end{aligned}$$

△

For the problem formulation (3.1) of the MIOCP, the functions $\boldsymbol{g}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are chosen to be:

$$\begin{aligned}
h_i(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\omega}(t)) &= \omega_i(t),\\
\boldsymbol{g_i}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\omega}(t)) &= \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^{\boldsymbol{i}}),
\end{aligned}$$

and the following MPVC is obtained

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{\omega}(\cdot)} \quad \Phi(\boldsymbol{x}(t_\mathrm{f})) \tag{3.20a}$$

$$\text{s.t.} \quad \bigvee_{1\leqslant i\leqslant n_\omega} \begin{bmatrix} \omega_i(t) = 1 \\ \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i) \end{bmatrix}, \qquad t \in [t_0, t_\mathrm{f}], \tag{3.20b}$$

$$\boldsymbol{0} \leqslant \omega_i(t)\, \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i), \qquad 1 \leqslant i \leqslant n_\omega, t \in [t_0, t_\mathrm{f}], \tag{3.20c}$$

$$\boldsymbol{0} = \boldsymbol{r}^{\mathrm{eq}}((t_i, \boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}), \qquad \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0, t_\mathrm{f}], \tag{3.20d}$$

$$\boldsymbol{0} \leqslant \boldsymbol{r}^{\mathrm{in}}((t_i, \boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}), \qquad \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0, t_\mathrm{f}], \tag{3.20e}$$

$$\boldsymbol{u}(t) \in \mathcal{U}, \qquad t \in [t_0, t_\mathrm{f}], \tag{3.20f}$$

$$\boldsymbol{\omega}(t) \in \{0, 1\}^{n_\omega}, \qquad t \in [t_0, t_\mathrm{f}]. \tag{3.20g}$$

Remark 3.3 (System of differential equations and complementarity constraints)
The quality of a reformulation of the dynamic process, i.e., the ODE system (3.20b), with complementarity constraints depends on the actual system at hand. In [149], the authors report promising results for a batch distillation column and a cryogenic distillation column.

Both formulations are DAE systems with two independent disjunctions of size 2 per tray. They discretize the system with collocation, and reformulate the disjunctions with the complementarity constraint formalism. They can solve realistic scenarios with IPOPT as the NLP solver, which includes an adaptation for complementarity constraints. In [14], 3 small case studies are also solved with this approach.
However, for large disjunctions, also large number of complementarity constraints need to be added to the model, which are generally difficult to solve. Also each equation that appears in a disjunction needs to be reformulated with 2 slack variables to be able to formulate it in the MPCC framework. We tried the formalism for the truck model from Section 5.4, but the resulting NLP could not be solved with IPOPT.

Remark 3.4 (Tightness)
A value of $\omega_i(t) > 0$ guarantees that the corresponding vanishing constraints are satisfied, i.e.,

$$\boldsymbol{c}(\cdot, \boldsymbol{v}^{\boldsymbol{i}}) \geqslant 0.$$

Therefore, the relaxation with $\omega_i(t) \in [0,1]$ is much tighter than the previously described relaxations as it is either fully active ($\omega_i(t) > 0$) or inactive ($\omega_i(t) = 0$) and nothing in between.

3.8.1 Constraint qualifications for MPVCs and MPCCs

MPCCs and MPVCs are highly challenging non-convex problems, which are known to possess critical points that violate the Linear Independence Constraint Qualification (LICQ). Their computational solution with standard NLP software is prone to numerical difficulties and often terminates in suboptimal points that possess trivial descent directions and are not local minimizers of (3.16) or (3.18).

Proposition 3.2 (No LICQ for vanishing constraints)
Let $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{\omega}})$ *be a feasible solution of the relaxation of a problem including the vanishing constraint formulation* (3.15) *with* $\mathcal{I}_{00} \neq \emptyset$*. Then LICQ is not satisfied.* △

Proof We look at the constraint matrix of all constraints in $(\cdot, \bar{\boldsymbol{\omega}})$, given by

$$\frac{\partial}{\partial(\omega_1, \ldots, \omega_{n_\omega}, (\boldsymbol{x}, \boldsymbol{u}))} \begin{bmatrix} \omega_1(t)\, \boldsymbol{c}(\cdot, \boldsymbol{v}^{\boldsymbol{1}}) \\ \vdots \\ \omega_{n_\omega}(t)\, \boldsymbol{c}(\cdot, \boldsymbol{v}^{\boldsymbol{n_\omega}}) \\ \sum_{i=1}^{n_\omega} \omega_i(t) - 1 \\ \omega_1(t) \\ \vdots \\ \omega_{n_\omega}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}(\cdot, \boldsymbol{v}^{\boldsymbol{1}}) & \cdots & \boldsymbol{0} & \bar{\omega}_1(t) \nabla \boldsymbol{c}(\cdot, \boldsymbol{v}^{\boldsymbol{1}}) \\ & \ddots & & \vdots \\ 0 & \cdots & \boldsymbol{c}(\cdot, \boldsymbol{v}^{\boldsymbol{n_\omega}}) & \bar{\omega}_{n_\omega}(t) \nabla \boldsymbol{c}(\cdot, \boldsymbol{v}^{\boldsymbol{n_\omega}}) \\ 1 & \cdots & 1 & 0 \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{bmatrix},$$

and notice that the rows that correspond to $\bar{\omega}_i(t)\, \boldsymbol{c}(\cdot, \boldsymbol{v}^i) \geqslant 0$ contain only zeros, leading to a trivially linear dependent constraint system. □

In [96], it is shown that not only LICQ is violated but that also the weaker MANGASARIAN-FROMOVITZ Constraint Qualification (MFCQ) and the ABADIE Constraint Qualification (ACQ) are violated. However, the cone-based GUIGNARD Constraint Qualification (GCQ) holds even for these points and hence local minimizers are still KARUSH-KUHN-TUCKER (KKT) points. Also, a specially tailored MPVC-LICQ can be derived, which considers only the row rank of a subset of constraints, where $\boldsymbol{h}$ and $\boldsymbol{g}$ are treated separately instead of treating their product (3.19).

Definition 3.6 (MPVC-LICQ)
Let $\bar{\mathbf{y}}$ be a feasible point of problem (3.18). *MPVC-LICQ is said to hold in $\bar{\mathbf{y}}$, if the gradients*

$$\begin{aligned}
&\nabla r_i(\bar{\mathbf{y}}), && i \in \mathcal{I}_r,\\
&\nabla r_i(\bar{\mathbf{y}}), && i \in \mathcal{E},\\
&\nabla h_i(\bar{\mathbf{y}}), && i \in \mathcal{I}_{00} \cup \mathcal{I}_{0-} \cup \mathcal{I}_{0+},\\
&\nabla g_i(\bar{\mathbf{y}}), && i \in \mathcal{I}_{00} \cup \mathcal{I}_{+0},
\end{aligned}$$

are linearly independent. △

Similarly, it can be shown that *all* feasible points for an MPCC violate MFCQ and as a direct consequence also LICQ, cf. [50, 164]. Therefore, in [64, 164], the analogous MPCC-LICQ is introduced:

Definition 3.7 (MPCC-LICQ)
Let $\bar{\mathbf{y}}$ be a feasible point of problem (3.16). *MPCC-LICQ is said to hold in $\bar{\mathbf{y}}$, if the gradients*

$$\begin{aligned}
&\nabla r_i(\bar{\mathbf{y}}), && i \in \mathcal{I}_r,\\
&\nabla r_i(\bar{\mathbf{y}}), && i \in \mathcal{E},\\
&\nabla h_i(\bar{\mathbf{y}}), && i \in \mathcal{I}_{0+} \cup \mathcal{I}_{00},\\
&\nabla g_i(\bar{\mathbf{y}}), && i \in \mathcal{I}_{+0} \cup \mathcal{I}_{00}
\end{aligned}$$

are linearly independent. △

These MPVC-LICQ and MPCC-LICQ can be used to derive applicable stationarity conditions and to construct applicable algorithms.

3.8.2 Regularization

Regularization formulations for MPVC relax the feasible set using $\varepsilon > 0$ to prevent the algorithms from reaching the points with violated LICQ. Instead of requiring the vanishing

constraints (3.15), their relaxed version

$$\begin{aligned} -\varepsilon \leqslant \omega_i(t)\, \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i), \qquad & 1 \leqslant i \leqslant n_\omega, \\ 1 = \sum_{i=1}^{n_\omega} \omega_i(t) & \end{aligned} \tag{3.21}$$

is required. Now, a sequence of perturbed problems is solved with $\varepsilon \searrow 0$, whose solutions converge. The solution should satisfy BOULIGAND *stationarity* to verify that it is not a *spurious stationary point*, which satisfies weaker stationarity conditions, but possesses descent directions. In [100], theoretical properties of solutions of the perturbed system are studied and under certain stationarity assumptions, the speed of convergence toward the true solution is given as $\mathcal{O}(\sqrt{\varepsilon})$.

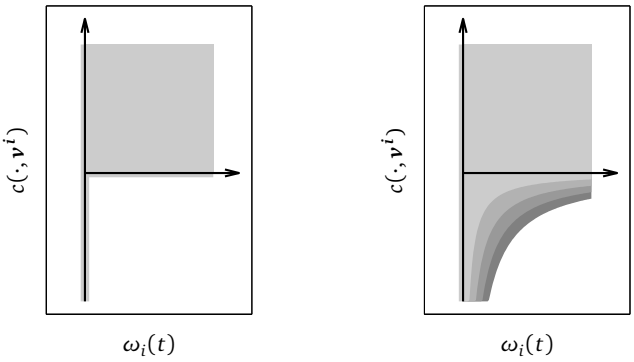


Figure 3.2: Feasible region for vanishing constraints (3.17, left) in comparison to the regularized version (3.21, right) with $\varepsilon \in \{0.05, 0.1, 0.15, 0.2\}$.

MPCCs can be regularized similarly and an overview of some possibilities and their properties can be gained in [97].

3.8.3 NCP functions

A function $\varphi : \mathbb{R}^2 \to \mathbb{R}$ is called Nonlinear Complementarity Problem (NCP) function if

$$\begin{aligned} \varphi(a,b) = 0 \quad &\Leftrightarrow \quad a \geqslant 0,\ b \geqslant 0,\ ab = 0, \qquad &\text{(MPCC)}, \\ \varphi(a,b) = 0 \quad &\Leftrightarrow \quad\ \ b \geqslant 0,\ ab \geqslant 0, \qquad &\text{(MPVC)}. \end{aligned}$$

For surveys on NCP functions in the MPCC context, cf. [63, 121]. Instead of the complementarity constraints or vanishing constraints, the reformulated problem includes the equation

$$\varphi\left(h(\boldsymbol{y}), g(\boldsymbol{y})\right) = 0,$$

or – for merit functions that satisfy $\forall(a,b)\in\mathbb{R}^2 : \varphi(a,b)\geqslant 0$ – the inequality

$$\varphi\left(h(\boldsymbol{y}),g(\boldsymbol{y})\right)\leqslant 0.$$

The most prominent NCP functions for MPCC are the FISCHER-BURMEISTER function

$$\varphi^{\mathrm{FB}}(a,b)\overset{\text{def}}{=}a+b-\sqrt{a^2+b^2},$$

and the min-function and its equivalent the natural residual function

$$\varphi^{\mathrm{NR}}(a,b)\overset{\text{def}}{=}\frac{1}{2}\left(a+b-\sqrt{(a-b)^2}\right)=\min\{a,b\}.$$

Observe that both are non-smooth functions and this is a desired property for the reformulation to be correct in $a=b=0$.

For MPVC, HOHEISEL gives reasoning to also use non-smooth functions since – with smooth functions – MPVC and hence also LICQ cannot hold at any feasible point in $\mathcal{I}_{0-}\cup\mathcal{I}_{+0}\cup\mathcal{I}_{00}$, cf. [96]. He proposes to use the following NCP function for MPVCs, which has also the advantage of being a merit function:

$$\varphi^{\mathrm{VC}}(a,b)\overset{\text{def}}{=}\frac{1}{2}\left(\sqrt{a^2b^2}-ab+\sqrt{b^2}-b\right)=-\min\{ab,0\}-\min\{b,0\}.$$

Non-smooth functions are desired for the reformulated problems to behave correctly, yet algorithms usually require a smooth setting to converge properly. Therefore, the functions are approximated with smooth functions with a smoothing parameter ε, cf. [63, 121]. Those functions are designed such that the limit case $\varepsilon=0$ is the non-smooth function, which shall be approximated. For MPCC, smoothing under the MPCC-LICQ assumption and different second-order conditions has been shown to yield B-stationarity of the limit point.

One such smoothing approach for MPCC, which was presented in [49, 60, 121] and showed good results, is the smoothed natural residual function with $\varepsilon<2$:

$$\varphi^{\mathrm{NR}}_{\varepsilon}(a,b)\overset{\text{def}}{=}\frac{1}{2}\left(a+b-\sqrt{(a-b)^2+\varepsilon\,ab}\right).$$

An advantage of this formulation is that the smoothed function remains an exact NCP function.

It is shown in [96] that these convergence results cannot be transfered directly to the MPVC setting. For example, the smoothing approach

$$\varphi^{\mathrm{VC}}_{\varepsilon}(a,b)\overset{\text{def}}{=}\frac{1}{2}\left(\sqrt{a^2b^2+\varepsilon^2}-ab+\sqrt{b^2+\varepsilon^2}-b\right)$$

can yield an empty feasible set for $\varepsilon>0$. HOHEISEL proposes to couple the smoothing ap-

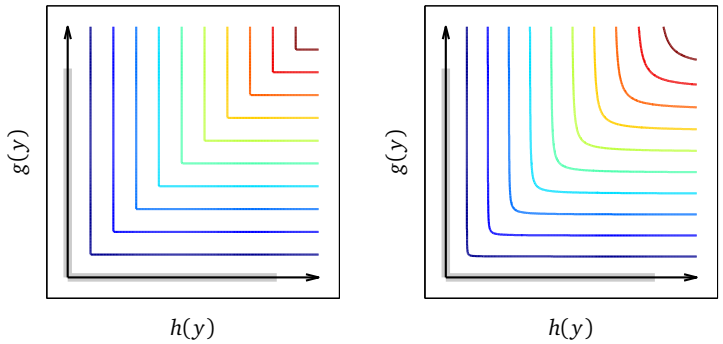


Figure 3.3: Contour lines for the non-smooth natural residual function $\varphi^{\mathrm{NR}}(\cdot)$ on the left and for the smoothed version $\varphi^{\mathrm{NR}}_{\varepsilon}(\cdot)$ on the right with $\varepsilon = \frac{1}{32}$.

proach with the already described regularization approach from Section 3.8.2, which produces the constraint

$$\varphi^{\mathrm{VC}}_{\varepsilon}\left(c(\cdot, \boldsymbol{v}^i), \omega_i(t)\right) \leqslant \varepsilon. \tag{3.22}$$

This formulation has the desired property of convergence for $\varepsilon \searrow 0$ to a strongly stationary limit point under MPVC-LICQ assuming existence of a sequence of feasible points for the sequence of smoothed, regularized problems and asymptotic non-degeneracy.

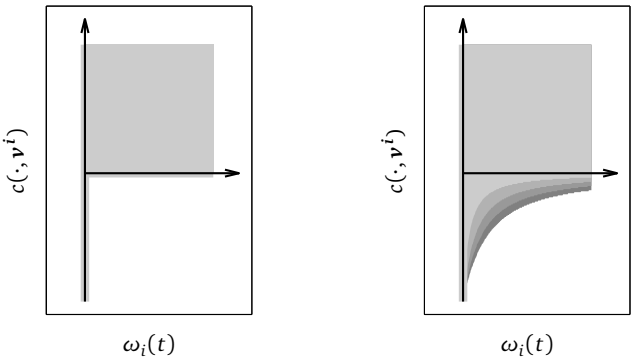


Figure 3.4: Feasible region for vanishing constraints (3.17, left) in comparison to the smoothed and regularized version (3.22, right) with $\varepsilon \in \{0.05, 0.1, 0.15, 0.2\}$.

3.9 Perspective formulation

The perspective formulation is originated in the MINLP community and stems from the GDP approach, cf. [9, 48, 84, 86, 115]. It is motivated by disjunctions as described in Definition 3.2 and it was originally developed for chemical applications in process synthesis. It was designed

for and applied to convex functions but it is now also studied for non-convex ones, cf. [150]. The authors propose to introduce convex underestimators for the non-convex functions, solve the resulting convex problem with the presented scheme and then tighten them in a spatial branch-and-bound framework.
The perspective formulation makes use of the perspective function, which is defined as

Definition 3.8 (Perspective function)
The perspective *of a function* $g : \mathbb{R}^n \to \mathbb{R}$ *is the function* $\hat{g} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ *defined by*

$$\hat{g}(\omega, \boldsymbol{y}) \stackrel{\text{def}}{=} \begin{cases} \omega\, g\left(\frac{\boldsymbol{y}}{\omega}\right), & \text{if } \omega > 0, \\ 0, & \text{if } \omega = 0, \boldsymbol{y} = \boldsymbol{0}, \\ \infty, & \text{otherwise.} \end{cases} \qquad \triangle$$

Perspective functions have been used for strong formulations in MINLPs, cf. [48]. They have also recently been used to implement the *perspective cuts*, cf. [88].

Remark 3.5
If a function $g(\cdot)$ is convex, then also its perspective $\hat{g}(\cdot)$ is convex, cf. [173].

The perspective formulation of the simple disjunction

$$\left[0 \geqslant g^1(\boldsymbol{y})\right] \vee \left[0 \geqslant g^2(\boldsymbol{y})\right], \qquad \boldsymbol{y} \in [\boldsymbol{m}^{\boldsymbol{y}}, \boldsymbol{M}^{\boldsymbol{y}}]$$

is described through

$$\begin{aligned} 0 &\geqslant \omega_1 g^1\left(\frac{\boldsymbol{y^1}}{\omega_1}\right), \\ 0 &\geqslant \omega_2 g^2\left(\frac{\boldsymbol{y^2}}{\omega_2}\right), \\ 1 &= \omega_1 + \omega_2, \\ \boldsymbol{y} &= \boldsymbol{y^1} + \boldsymbol{y^2}, \\ \boldsymbol{y^1} &\in \omega_1[\boldsymbol{m}^{\boldsymbol{y}}, \boldsymbol{M}^{\boldsymbol{y}}], \\ \boldsymbol{y^2} &\in \omega_2[\boldsymbol{m}^{\boldsymbol{y}}, \boldsymbol{M}^{\boldsymbol{y}}], \\ \omega_1, \omega_2 &\in \{0, 1\}. \end{aligned}$$

Its relaxed projection to the $(\boldsymbol{y}, \omega_1, \omega_2)$-space is the convex hull of the disjunction, if the involved functions $g^1(\cdot)$ and $g^2(\cdot)$ are convex.
For a more general description, the following theorem holds:

Theorem 3.1 (Convex hull of disjunction)
The convex hull in the $(\boldsymbol{\omega}, \boldsymbol{y})$-space of each single disjunction of the problem

$$\begin{aligned} \min_{\boldsymbol{y},\boldsymbol{\omega}} \quad & e(\boldsymbol{y}) + \sum_{k\in\mathcal{K}} e_k \\ \text{s.t.} \quad & \bigvee_{i\in\mathcal{I}_k} \begin{bmatrix} \omega_{ik} = 1 \\ \mathbf{0} \geqslant \boldsymbol{g_{ik}}(\boldsymbol{y}) \\ e_k = \gamma_{ik} \end{bmatrix}, \qquad k \in \mathcal{K}, \\ & \boldsymbol{y} \in [\boldsymbol{m}^{\boldsymbol{y}}, \boldsymbol{M}^{\boldsymbol{y}}], \\ & \boldsymbol{\omega} \in \{0,1\}^{n_\omega}, \end{aligned} \tag{3.23}$$

where $\boldsymbol{g_{ik}}(\boldsymbol{y})$ are convex inequalities, is the convex set given by the projection onto the $(\boldsymbol{\omega}, \boldsymbol{y})$-space of the feasible set of

$$\begin{aligned} \boldsymbol{y} &= \sum_{i\in\mathcal{I}_k} \boldsymbol{y^{ik}}, \\ 0 &\geqslant \omega_{ik}\, \boldsymbol{g_{ik}}\left(\frac{\boldsymbol{y^{ik}}}{\omega_{ik}}\right), \quad i \in \mathcal{I}_k, \\ 1 &= \sum_{i\in\mathcal{I}_k} \omega_{ik}, \\ e_k &= \sum_{i\in\mathcal{I}_k} \omega_{ik}\, \gamma_{ik}, \\ \boldsymbol{y^{ik}} &\in \omega_{ik}\, [\boldsymbol{m}^{\boldsymbol{y}}, \boldsymbol{M}^{\boldsymbol{y}}], \qquad i \in \mathcal{I}_k. \end{aligned} \tag{3.24}$$

△

Proof A proof can be found in [84]. □

Remark 3.6 (Numerical difficulties)
LICQ usually holds for perspective functions. However, due to the input structure and division by small numbers, numerical difficulties arise for ω close to 0. For regularization, we introduce a small $\varepsilon > 0$ and obtain the ε-dependent constraint

$$0 \geqslant (\omega + \varepsilon)\, g\left(\frac{\boldsymbol{y}}{\omega + \varepsilon}\right), \tag{3.25}$$

which was proposed in [84]. It avoids the problematic operation of division by zero. However, this formulation does not correctly model the feasibility for ω close to 0 as the point $(\omega, \boldsymbol{y}) = \mathbf{0}$ might be cut off, cf. Figure 3.5. This point is especially important since it marks the complete inactivity of a disjunct and it must not be cut off for good numerical behavior. Therefore, it was improved in [139] to better capture the original feasibility behavior as

$$0 \geqslant \omega\, g\left(\frac{\boldsymbol{y}}{\omega + \varepsilon}\right). \tag{3.26}$$

Yet, it was also shown that this reformulation sometimes transforms nonlinear convex constraints into nonlinear non-convex ones, cf. [138].

In [162, 163], convex reformulations are studied and compared, e.g. the reformulation

$$0 \geqslant (\omega + \varepsilon)\, g\left(\frac{\boldsymbol{y}}{\omega + \varepsilon}\right) - \varepsilon \max_{\mu,\boldsymbol{v}} g\left(\frac{\boldsymbol{v}}{\mu + \varepsilon}\right), \tag{3.27}$$

retains convexity but has the problem of finding the inner maximum. A second proposed reformulation is

$$0 \geqslant ((1-\varepsilon)\,\omega + \varepsilon)\; g\left(\frac{\boldsymbol{y}}{(1-\varepsilon)\,\omega + \varepsilon}\right) + \varepsilon\,(\omega - 1)\, g(\mathbf{0}). \tag{3.28}$$

Sawaya and Grossmann declare the second reformulation to be closer to the original perspective cut for almost all realistic scenarios and give conditions when exactly this is the case, cf. [163]. Also notice that this an exact problem representation for $\omega \in \{0,1\}$ and any choice of ε. However, for this reformulation to be well-posed, the function $g(\cdot)$ needs to be defined in $\mathbf{0}$. However, this can always be accomplished through an affine transformation of $\boldsymbol{y}$.

For MINLPs, the entering variables are duplicated for each side of a disjunction and instead of adding the constraints, their perspective formulations are added. Analogously, in [139], the authors proposed to lift all variables, i.e., differential states, controls, and horizon lengths, into a higher space. Therefore, for each system mode, we introduce disaggregated states $\boldsymbol{x}^i(t)$, and control decision variables $\boldsymbol{u}^i(t)$. The newly introduced variables have to be aggregated to give a meaningful interpretation:

$$\begin{aligned} \boldsymbol{x}(t) &= \sum_{i=1}^{n_\omega} \boldsymbol{x}^i(t), \\ \boldsymbol{u}(t) &= \sum_{i=1}^{n_\omega} \boldsymbol{u}^i(t). \end{aligned} \tag{3.29}$$

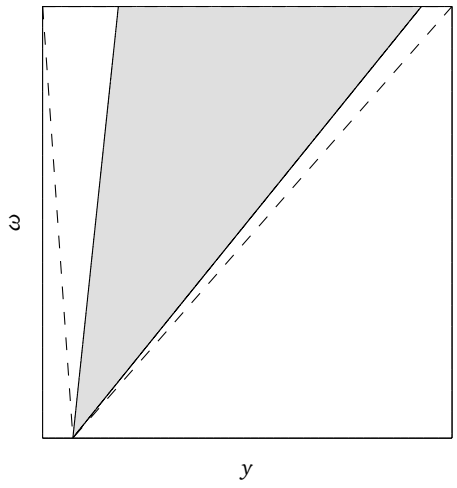
Using these variables, we are able to reformulate the disjunction for $t \in [t_0, t_f]$

$$\bigvee_{1 \leqslant i \leqslant n_\omega} \begin{bmatrix} \omega_i(t) = 1 \\ \dot{\boldsymbol{x}}(t) = \boldsymbol{f}^i(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \mathbf{0} \leqslant \boldsymbol{c}^i(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \end{bmatrix}$$

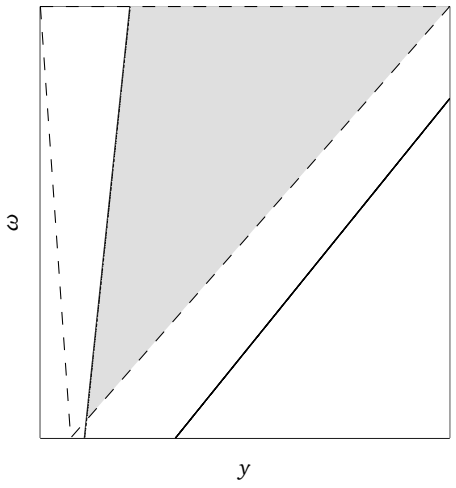
with perspective functions to be

$$\dot{\boldsymbol{x}}^i(t) = \omega_i(t)\, \boldsymbol{f}^i\left(t, \frac{\boldsymbol{x}^i(t)}{\omega_i(t)}, \frac{\boldsymbol{u}^i(t)}{\omega_i(t)}\right), \qquad 1 \leqslant i \leqslant n_\omega, \tag{3.30}$$

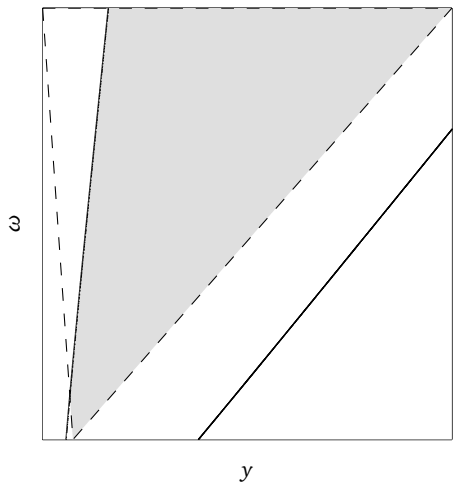
(a) Perspective formulation as in Definition (3.8)



(b) Regularized formulation (3.25)



(c) $(0,0)$-correction with maximum term (3.27)



(d) $(0,0)$-correction with constant (3.28)

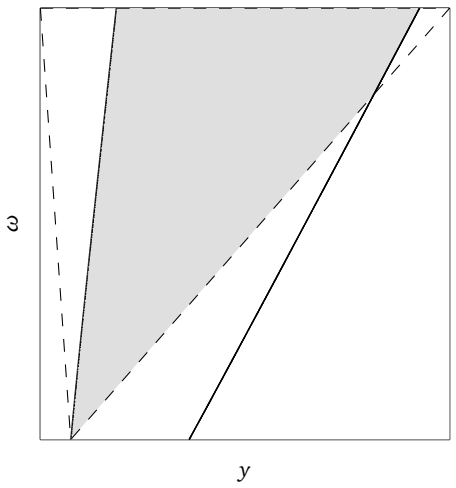


Figure 3.5: Feasible regions for the different perspective formulations of $g(y) = (y - 1.3)^2 - 1 \geqslant 0$ with $\varepsilon = 0.3$. The dashed lines display the bounds $y \in \omega[m^y, M^y]$.
Formulation (3.28) unifies all positive properties, i.e., correct formulation for $\omega \in \{0,1\}$, convexity, and numerical stability (if $g(\mathbf{0})$ defined), while all other formulations lack in at least one property.

$$
\begin{aligned}
\mathbf{0} \leqslant \omega_i(t)\, \boldsymbol{c}^i \left(t, \frac{\boldsymbol{x}^i(t)}{\omega_i(t)}, \frac{\boldsymbol{u}^i(t)}{\omega_i(t)} \right), \qquad & 1 \leqslant i \leqslant n_\omega, \\
\omega_i(t)\, \boldsymbol{m}^{\boldsymbol{x}} \leqslant \boldsymbol{x}^i(t) \leqslant \omega_i(t)\, \boldsymbol{M}^{\boldsymbol{x}}, \qquad & 1 \leqslant i \leqslant n_\omega, \\
\omega_i(t)\, \boldsymbol{m}^{\boldsymbol{u}} \leqslant \boldsymbol{u}^i(t) \leqslant \omega_i(t)\, \boldsymbol{M}^{\boldsymbol{u}}, \qquad & 1 \leqslant i \leqslant n_\omega, \\
1 = \sum_{i=1}^{n_\omega} \omega_i(t), & \\
\boldsymbol{\omega}(t) \in \{0,1\}^{n_\omega}, &
\end{aligned}
$$

where $\boldsymbol{m}^{\boldsymbol{x}}$, $\boldsymbol{M}^{\boldsymbol{x}}$, $\boldsymbol{m}^{\boldsymbol{u}}$ and $\boldsymbol{M}^{\boldsymbol{u}}$ are valid lower and upper bounds for the states and controls.

Remark 3.7 (Problematic dimension-increase)
The main problem with the perspective approach is the increase in the problem's dimensions. This is proportional to the size of the disjunctions, i.e., $\left|\mathcal{I}_k\right|$, and especially for larger disjunctions – as can be found in the truck problem of Section 5.4 – even local algorithms may have problems finding solutions in the large space due to the non-convexity of the ODE.

3.9.1 Projection of perspective's convex hull for order preserving functions

An order preserving function is defined through:

Definition 3.9
Let $g : E \to \mathbb{R}$, $E \subseteq \mathbb{R}^{n_x}$. g is independently increasing *(resp.* decreasing*) on the ith coordinate if $\forall \boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_n) \in \mathrm{dom}(g)$, $\boldsymbol{x'} = (x_1', \ldots, x_i', \ldots, x_n') \in \mathrm{dom}\, g$, with $x_i' \geqslant x_i$: $g(\boldsymbol{x'}) \geqslant$ (resp. $\leqslant$)$g(\boldsymbol{x})$.*
We say that g is independently monotone *on the i-th coordinate if it is independently increasing or decreasing on the given coordinate.*
g is order preserving *if it is independently monotone on each coordinate.* △

For order preserving functions $\boldsymbol{g} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_c}$ and "on/off" disjunctions, i.e., disjunctions of the form

$$
\left[\omega = 0, \boldsymbol{x} \in [\boldsymbol{l^0}, \boldsymbol{u^0}]\right] \vee \left[\omega = 1, \boldsymbol{x} \in [\boldsymbol{l^1}, \boldsymbol{u^1}], g(\boldsymbol{x}) \leqslant 0\right],
$$

a reformulation of the convex hull of the disjunction is introduced in [95], which adds 2^{n_x} constraints for each disjunction but does not need to lift the variables as described above. Since the addition of 2^{n_x} constraints is impractical, the authors propose to take only a subset of those constraints that already gives a relatively tight convex approximation of the convex hull of the disjunction. It is described through

$$
\begin{aligned}
& 0 \geqslant \omega\, g(h_\emptyset(\boldsymbol{x}, \omega)), \\
& \boldsymbol{x} \in (1-\omega)[\boldsymbol{l^0}, \boldsymbol{u^0}] + \omega[\boldsymbol{l^1}, \boldsymbol{u^1}], \\
& \omega \in [0,1],
\end{aligned}
\tag{3.31}
$$

with

$$(h_\emptyset(\boldsymbol{x},\omega))_i \stackrel{\text{def}}{=} \begin{cases} \frac{x_i}{\omega} - \frac{(1-\omega)u_i^0}{\omega}, & \text{if } g \text{ is independently increasing on index } i, \\ \frac{x_i}{\omega} - \frac{(1-\omega)l_i^0}{\omega}, & \text{if } g \text{ is independently decreasing on index } i. \end{cases} \tag{3.32}$$

Since the function is order preserving, the variable value's effect is limited by the boundary values. This can be seen best for linear functions, as the procedure computes the best Big-M formulation obtainable by using exclusively variable bounds. However, the restriction of the functions to be order preserving is quite strict and hence the reformulation cannot be applied very often.

3.9.2 Tightening the perspective formulation

The presented formulation describes the convex hull of the feasible integral points. However, it requires the lifting of the state and control variables into a higher dimensional space and this may be impractical with large disjunctions with many modes. Here, it may be advantageous to project the lifted variables back into the original space to get a tighter relaxation. Notice that this projected relaxation's feasible set must be non-convex since it realizes a true subset and it must contain the original disjunctions' constraints for integral ω whose convex hull is given by the perspective formulation.

One example projection of this type not only assigns each disjunct its own share of the space, but requires the controls and states to be partitioned in the same way as the spaces are. The following constraints are added to project the lifted states and controls into their respective original spaces:

$$\begin{aligned} \boldsymbol{x}^{\boldsymbol{i}}(t) &= \omega_i(t)\, x(t), \\ \boldsymbol{u}^{\boldsymbol{i}}(t) &= \omega_i(t)\, u(t). \end{aligned} \tag{3.33}$$

This requirement is consistent with the perspective formulation as the following constraints of the perspective's lifted variables are automatically fulfilled:

$$\begin{aligned} \boldsymbol{x}(t) &= \sum_{i=1}^{n_\omega} \boldsymbol{x}^{\boldsymbol{i}}(t) = \sum_{i=1}^{n_\omega} \omega_i(t)\, \boldsymbol{x}(t) = \boldsymbol{x}(t), \\ \boldsymbol{u}(t) &= \sum_{i=1}^{n_\omega} \boldsymbol{u}^{\boldsymbol{i}}(t) = \sum_{i=1}^{n_\omega} \omega_i(t)\, \boldsymbol{u}(t) = \boldsymbol{u}(t), \\ \boldsymbol{x}^{\boldsymbol{i}}(t) &= \omega_i(t)\, x(t) \in \omega_i(t)\, [\boldsymbol{m}^{\boldsymbol{x}}, \boldsymbol{M}^{\boldsymbol{x}}], \\ \boldsymbol{u}^{\boldsymbol{i}}(t) &= \omega_i(t)\, u(t) \in \omega_i(t)\, [\boldsymbol{m}^{\boldsymbol{u}}, \boldsymbol{M}^{\boldsymbol{u}}]. \end{aligned} \tag{3.34}$$

The other constraints can be reformulated in the original state to be

$$\omega_i(t)\,\dot{\boldsymbol{x}}(t) = \dot{\boldsymbol{x}}^{\boldsymbol{i}}(t) = \omega_i(t)\,\boldsymbol{f^i}\left(t, \frac{\boldsymbol{x^i}(t)}{\omega_i(t)}, \frac{\boldsymbol{u^i}(t)}{\omega_i(t)}\right) = \omega_i(t)\,\boldsymbol{f^i}\,(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{3.35}$$

$$\boldsymbol{0} \leqslant \omega_i(t)\,\boldsymbol{c^i}\left(t, \frac{\boldsymbol{x^i}(t)}{\omega_i(t)}, \frac{\boldsymbol{u^i}(t)}{\omega_i(t)}\right) = \omega_i(t)\,\boldsymbol{c^i}\,(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{3.36}$$

which is exactly the formulation from Section 3.8 with vanishing constraints (3.15) and lifted complementarity constraints:

$$0 = \omega_i(t)\left(\dot{\boldsymbol{x}}(t) - \boldsymbol{f^i}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\right) \tag{3.37}$$

$$\Leftrightarrow \begin{pmatrix} 0 \leqslant \omega_i(t) \perp \dot{\boldsymbol{x}}(t) - \boldsymbol{f^i}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{z_{1,i}}(t) \geqslant 0, \\ 0 \leqslant \omega_i(t) \perp \boldsymbol{f^i}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) - \dot{\boldsymbol{x}}(t) + \boldsymbol{z_{2,i}}(t) \geqslant 0, \\ 0 \leqslant \omega_i(t) \perp \boldsymbol{z_{1,i}}(t) \geqslant 0, \\ 0 \leqslant \omega_i(t) \perp \boldsymbol{z_{2,i}}(t) \geqslant 0. \end{pmatrix} \tag{3.38}$$

Therefore, we can state for the corresponding feasible sets:

Lemma 3.1
Let $\mathcal{F}^{GDP}$ the feasible set of the original integral problem, $\mathcal{F}^{VC/CC}$ the projection of the feasible set of the combination of vanishing constraints (3.15) *and complementarity constraints* (3.38) *onto the $(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\omega}(t))$-space, and $\mathcal{F}^{CH}$ the feasible set of the convex hull formulation* (3.30). *Then, it holds:*

$$\mathcal{F}^{GDP} \subset \mathcal{F}^{VC/CC} \subset \mathcal{F}^{CH}. \qquad \triangle$$

Proof The first inclusion is trivial since it is a relaxation. The second inclusion comes from the derivation with the additionally tightening constraints (3.33). □

As pointed out in Remark 3.3, this tightening may be too severe with regard to the usually nonlinear discretization of the ODE system. A more lenient formulation – as e.g. the OC of the ODE – might be in order, which aggregates the complementarity constraints (3.35) into a single constraint:

$$\begin{aligned} & \omega_i(t)\,\dot{\boldsymbol{x}}(t) = \omega_i(t)\,\boldsymbol{f^i}\,(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \Rightarrow \qquad & \dot{\boldsymbol{x}}(t) = \sum_{i=1}^{n_\omega} \omega_i(t)\,\boldsymbol{f^i}\,(t, \boldsymbol{x}(t), \boldsymbol{u}(t)). \end{aligned}$$

Another variant could be to project only the states into the original space, apply the aggregation onto the ODE and leave the lifted controls as additional degrees of freedom to be able to

satisfy the vanishing constraints more easily:

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \sum_{i=1}^{n_\omega} \omega_i(t)\, \boldsymbol{f^i}\left(t, \boldsymbol{x}(t), \frac{\boldsymbol{u^i}(t)}{\omega_i(t)}\right), \\ \boldsymbol{0} &\leqslant \omega_i(t)\, \boldsymbol{c^i}\left(t, \boldsymbol{x}(t), \frac{\boldsymbol{u^i}(t)}{\omega_i(t)}\right).\end{aligned} \tag{3.39}$$

3.10 Illustrative comparison of different formulations

In this section, we summarize some of the properties of the different formulations presented in Sections 3.5–3.8.
To demonstrate the different formulations for constraints and their effects a simple example disjunction is used in the following. It is a disjunction of two quadratic, convex constraints ($a_1(\boldsymbol{v}) \leqslant 0$):

$$\begin{aligned}&\left[c(y, \boldsymbol{v^1}) \stackrel{\text{def}}{=} a_1(\boldsymbol{v^1})\, y^2 + a_2(\boldsymbol{v^1})\, y + a_3(\boldsymbol{v^1}) \geqslant 0\right] \\ &\quad \vee \left[c(y, \boldsymbol{v^2}) \stackrel{\text{def}}{=} a_1(\boldsymbol{v^2})\, y^2 + a_2(\boldsymbol{v^2})\, y + a_3(\boldsymbol{v^2}) \geqslant 0\right], \\ &y \in [l_y, u_y].\end{aligned}$$

The functions stem from the truck model of Section 5.4 and are the functions $M_{\text{ind,max}}(\cdot)$, cf. (5.25). The parameters $\boldsymbol{v^1}$ and $\boldsymbol{v^2}$ correspond to the special settings of gears $\mu(s) = 10$ and $\mu(s) = 13$ and the variable y corresponds to the truck's speed.
The feasible sets must coincide to the original MIOCP formulation in $\omega \in \{0, 1\}$ for the formulations to be valid relaxations.

Inner Convexification

The IC formulation with two constraints is

$$c^{\text{IC}}(y, \omega) \stackrel{\text{def}}{=} c(y, \omega\, \boldsymbol{v^1} + (1 - \omega)\, \boldsymbol{v^2}) \geqslant 0. \tag{3.40}$$

As the feasible set shows, the reformulation does not retain convexity. This depends on how the parameters $\boldsymbol{v^i}$ enter the functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{c}(\cdot)$ and cannot necessarily be controlled.

Outer Convexification

The OC formulation with two constraints is

$$c^{\text{OC}}(y, \omega) \stackrel{\text{def}}{=} \omega\, c(y, \boldsymbol{v^1}) + (1 - \omega)\, c(y, \boldsymbol{v^2}) \geqslant 0. \tag{3.41}$$

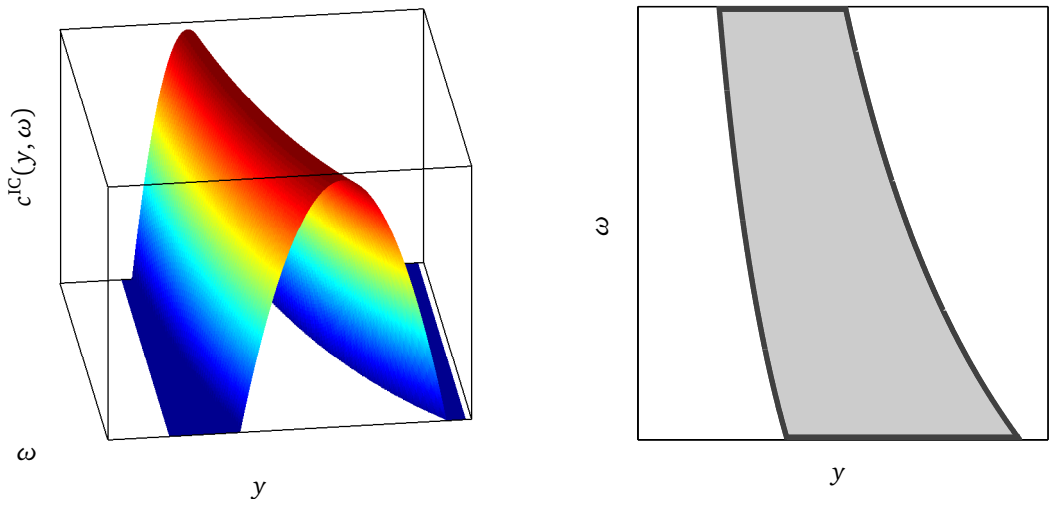


Figure 3.6: IC applied to a two sided disjunction – left: $c^{\mathrm{IC}}(\cdot)$ from (3.40) – right: feasible set $\{(y,\omega) \mid c^{\mathrm{IC}}(y,\omega) \geqslant 0\}$.

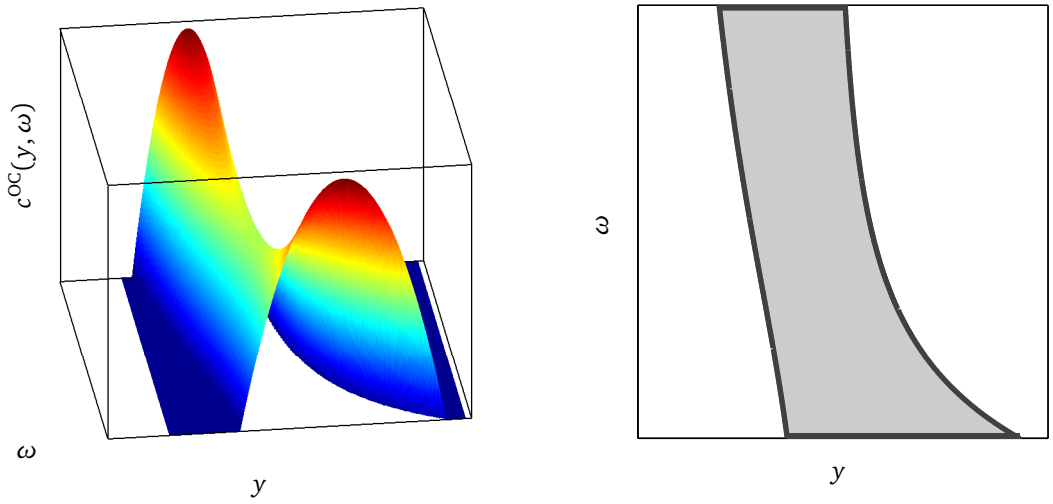


Figure 3.7: OC applied to a two sided disjunction – left: $c^{\mathrm{OC}}(\cdot)$ from (3.41) – right: feasible set $\{(y,\omega) \mid c^{\mathrm{OC}}(y,\omega) \geqslant 0\}$.

As the feasible set shows, the reformulation does not retain convexity. It can even be guaranteed that, if the two feasible sets are disjoint in the y-space, the feasible set after OC in the (y, ω)-space is also not connected. This is due to the feasible set in the y-space being the union of the feasible sets of the different modes. However, it remains convex in the y-space for fixed ω and vice versa. This property usually leads to good convergence of the relaxed OCP with standard NLP methods. One has to investigate the problem structure before the application of the OC technique to a problem since local methods are definitely not suited for problems with disconnected feasible sets.

Big-M formulation

The Big-M formulation with two constraints is

$$0 \leqslant c(y, \boldsymbol{\nu^1}) + M_1\,(1-\omega), \tag{3.42}$$
$$0 \leqslant c(y, \boldsymbol{\nu^2}) + M_2\,\omega. \tag{3.43}$$

The Big-M relaxation is quite weak since it provides a large feasible region, which can be seen in the comparison of the figures. Usually, due to this behavior, the Big-M relaxation gives the weakest relaxation of the ones presented.

Perspective formulation

The perspective formulation with two constraints introduces auxiliary states y^1 and y^2 and is

$$\begin{aligned}
&0 \leqslant \omega\, c\left(\frac{y^1}{\omega}, \boldsymbol{\nu^1}\right),\\
&0 \leqslant (1-\omega)\, c\left(\frac{y^2}{1-\omega}, \boldsymbol{\nu^2}\right),\\
&y \;\; = y^1 + y^2,\\
&y^1 \in \omega\,[l_y, u_y],\\
&y^2 \in (1-\omega)\,[l_y, u_y].
\end{aligned} \tag{3.44}$$

In contrast to the other formulations provided, except the Big-M formulation, the perspective relaxation retains convexity of the functions provided, which is a strong advantage. However, to guarantee this behavior, the variables involved in the disjunctions have to be duplicated for each disjunct of the disjunction. This process severely increases the problem's dimensions and usually comes at the cost of strongly increased computational effort. The special formulation for order preserving functions circumvents this variable duplication while almost giving the same strength of relaxation. Sometimes some variables may only be present in one disjunct as, e.g., in the sewage example from Section 5.3. Then, they only need to be introduced for

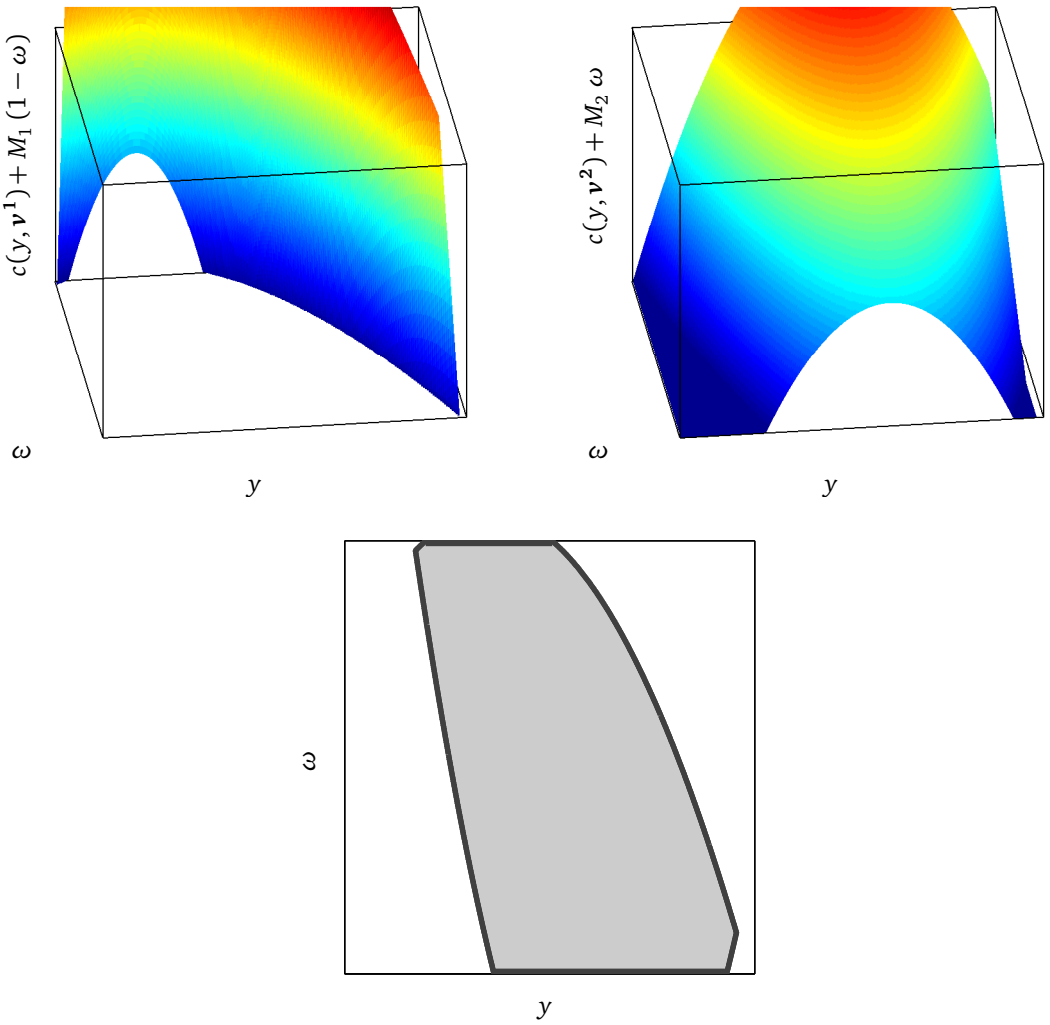


Figure 3.8: Big-M formulation applied to a two sided disjunction (The variable's bounds used to compute M_1 and M_2 are at the box's boundaries. Notice that the graphs have a skewed z-axis in comparison to the other graphs) – top: $c^{\text{Big-M}}(\cdot)$ from (3.42) and (3.43) – bottom: feasible set $\{(y, \omega) \mid c^{\text{Big-M}}(y, \omega) \geqslant 0\}$.

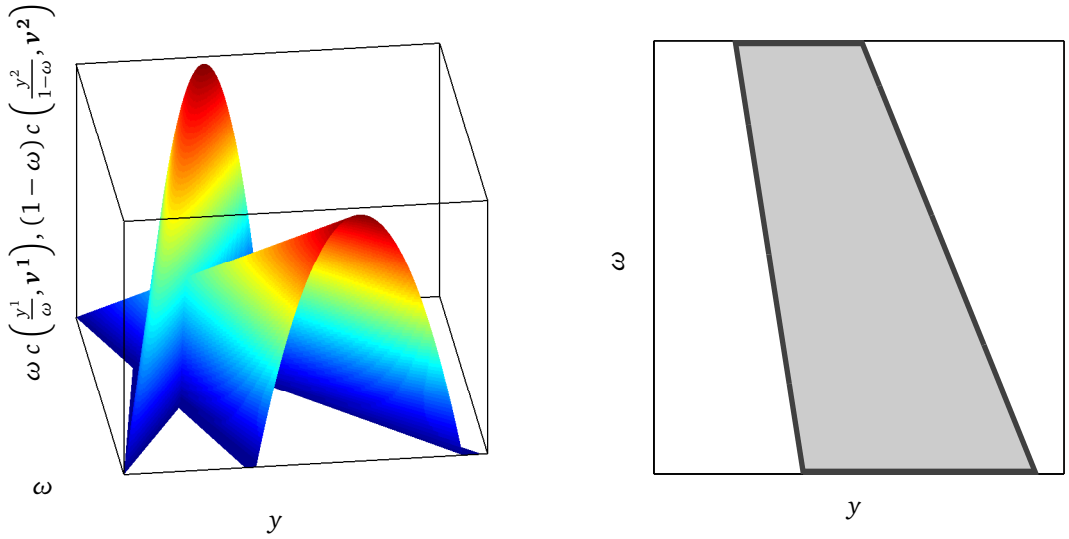


Figure 3.9: Perspective formulation applied to a two sided disjunction – left: perspective functions, where the y^1 and y^2 space are drawn onto the same axis – right: projection of feasible set $\{(y, \omega) \mid \exists y^1, y^2 : (y, y^1, y^2, \omega)$ satisfy (3.44)$\}$.

the disjuncts in which they are present and a lot of effort may be saved. This especially holds true for design problems, where the system mode may only be chosen once for each technical component.

Vanishing constraints

The vanishing constraint formulation is

$$0 \leqslant \omega\, c(y, \boldsymbol{\nu}^1), \tag{3.45}$$

$$0 \leqslant (1-\omega)\, c(y, \boldsymbol{\nu}^2). \tag{3.46}$$

As the OC formulation, this formulation may produce a disconnected feasible set, if the original feasible sets for the different disjuncts of the disjunctions are disjoint. In this case, local algorithms typically only minimize over the connected component in which the first feasible point is located. However, this situation is not found very often: typically the feasible sets overlap. Another disadvantage is that the feasible set is always non-convex and due to its structure, either special algorithms are needed that can handle weak stationarity or a smoothing, regularization reformulation must be used together with a homotopy to drive the smoothing and regularization parameter to zero. However, even those methods often cannot guarantee convergence to the optimal solution due to the non-convexities introduced.
An advantage of the vanishing constraint formulation for mode specific path constraints is that any rounded solution $\boldsymbol{\omega}(t)$ that is obtained from a fractional, relaxed solution $\boldsymbol{\alpha}(t)$, where no 0-value is rounded up, remains feasible for the vanishing constraints if the states

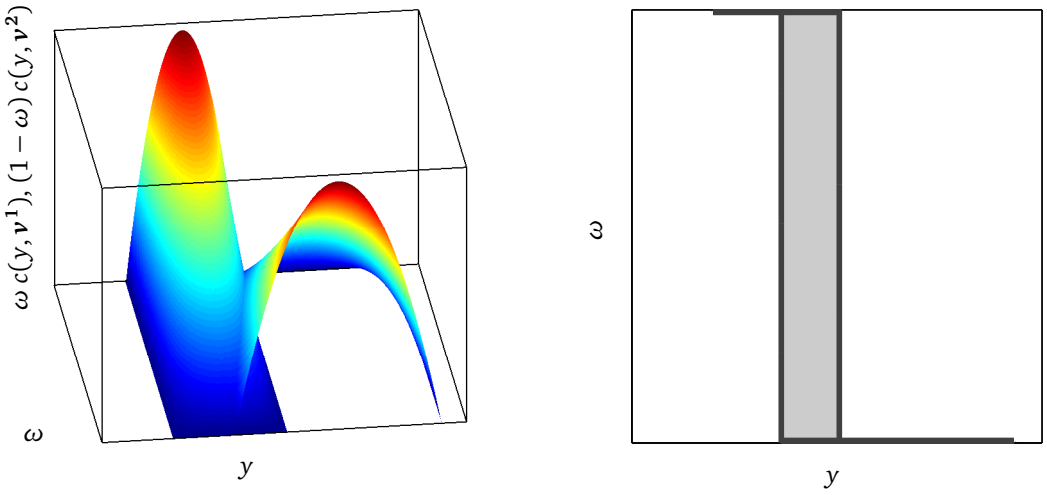


Figure 3.10: Vanishing constraint formulation applied to a two sided disjunction – left: both disjunctions' vanishing constraints (3.45) and (3.46) – right: feasible set $\{(y, \omega) \mid c^{\mathrm{VC}}(y, \omega) \geqslant 0\}$.

are approximated closely enough. This behavior allows easily accessible primal rounding heuristics to be applied in a branch-and-bound framework, which strongly enhances pruning. One of these rounding strategies that guarantees a close state approximation for control-affine systems is given in Chapter 4.

3.11 Controlling the switching behavior

Due to several reasons, it may happen that the optimal solution switches infinitely often between the system's modes. This typically happens when the relaxed problem's solution has fractional controls, which are called singular arcs. They are named due to their cause:

1. path-constrained arcs, cf. [160] for an example of a subway train with the velocity limit as path constraint,
2. sensitivity-seeking arcs, cf. Section 5.2 for an example of a LOTKA-VOLTERRA fishing problem with such an arc,
3. chattering arcs, cf. Fuller's problem [69].

The behavior of infinitely often switching optimal control functions is called ZENO*'s Phenomenon*, and it is not desired. These infinite switches might be necessary to perfectly track a constraint. However, if this solution occurs, the process is not well modeled since very fast switching can rarely be implemented to reality. Limiting the system to finitely many switches seems natural and the switching behavior can be excluded by assuming the integer control

functions to be of bounded variation, i.e., to use piecewise constant functions as a basis, which is $b_j^0(\cdot)$ as described in Section 2.4.1. Using piecewise constant functions might not be enough for a fine discretization as many approximations on a discretized time grid still incur chattering behavior as shown in Figure 3.11.

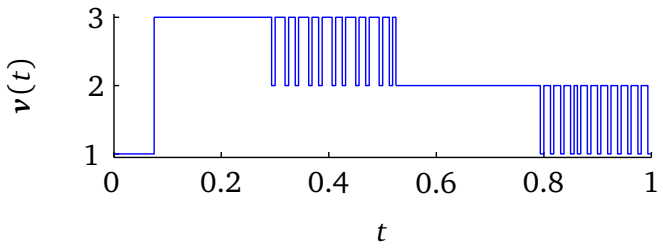


Figure 3.11: The singular arcs of the relaxed solution are typically approximated with a fast switching between system modes. This solution is taken from the example of Section 5.1.

There are several possible circumventions to prevent the system from highly frequent mode switching. The first is by limiting the number of system switches directly with an upper bound σ – a hard constraint. For relatively small σ, this makes chattering behavior infeasible since a large number of switches would be required to realize it. The second way is to penalize switching within the objective, i.e., to impose the switching constraint as a soft constraint. This might realize some operational cost associated with the switching operation or it might be employed to discourage excessive wear of the involved system parts through the modeling of the maintenance costs. The optimizer has to trade off the profit of highly frequent switching with the associated costs. A third way to prevent the system from high frequent switching is to employ a min-up time, i.e., the system has to stay in a given mode for a certain time after it switched into it. This may model mechanical switches in systems with small time scales, where the switching time cannot be neglected.

3.11.1 Switch modeling

In this section, we model switching only for binary control functions since they are mostly used to model system modes as in Definition 3.2. We limit ourselves to the discretized case, where the controls can only switch in the discretization points t_i. As already described, the basis functions to model the binary controls should be piecewise constant functions on an appropriate time grid

$$\mathbb{G}_m \stackrel{\text{def}}{=} \{t_0 < t_1 < \ldots < t_m = t_f\},$$

that may contain variable time points but is limited in its size m. The basis control functions on this grid would be

$$b_j^0(t) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } t \in [t_j, t_{j+1}), \\ 0, & \text{else,} \end{cases} \qquad 0 \leqslant j \leqslant m-1,$$

with the binary controls then being

$$\omega_i(t; \boldsymbol{q}) \stackrel{\text{def}}{=} \sum_{j=0}^{m-1} b_j^0(t)\, q_{i,j}, \qquad 1 \leqslant i \leqslant n_\omega.$$

As suggested in our work [158], a switch in control ω_i at time step t_j is captured by the term

$$\sigma_{i,j} \stackrel{\text{def}}{=} \left| q_{i,j} - q_{i,j-1} \right|, \qquad 1 \leqslant j \leqslant m. \tag{3.47}$$

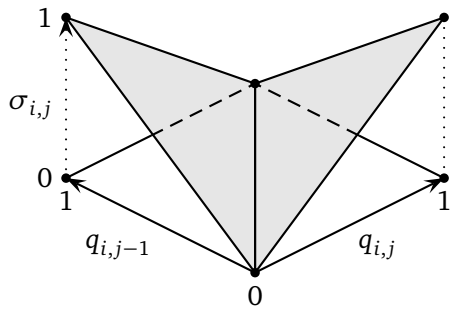


Figure 3.12: Switch formulation (3.47).

These switching variables $\sigma_{i,j}$ can now be used for the soft constraints by adding

$$\sum_{i=1}^{n_\omega} \sum_{j=1}^{m} \lambda_{i,j}\, \sigma_{i,j}$$

to the objective function. As a hard constraint, e.g. the total number of switches could be limited by $\sigma_{\max}$:

$$\sum_{i=1}^{n_\omega} \sum_{j=1}^{m} \sigma_{i,j} \leqslant 2\, \sigma_{\max}.$$

A known technique in linear programming to evade the non-differentiable absolute value

function $|\cdot|$ in formulation (3.47) is to use the following two constraints instead, which are the tightest underestimating hyperplanes for the absolute value function:

$$\sigma_{i,j} \geqslant q_{i,j} - q_{i,j-1},$$
$$\sigma_{i,j} \geqslant q_{i,j-1} - q_{i,j}.$$

Notice that this reformulation can only be done, if a higher value of $\sigma_{i,j}$ can never have a positive impact due to it being either punished in the objective or it being limited from above. If it is penalized in the objective and does not occur anywhere else, the solution always fulfills one of the two constraints with equality and hence is equal to the original formulation (3.47). The reformulation also is linear, which can be added to most constrained problems without pushing them into a more difficult problem class.
However, this formulation has one strong disadvantage, i.e., it favors solutions with $q_{i,j} \approx q_{i,j-1}$. Numerical experiments show that relaxed solutions often have fractional values for $q_{i,j}$ to evade switching. E.g., the all-$\frac{1}{2}$ solution produces no switch values, yet any binary solution that approximates the all-$\frac{1}{2}$ solution yields chattering behavior with lots of switches.

Convex combination

To account for this behavior, KIRCHES proposed in [111] to use a reformulation that penalizes this type of solutions. Instead of using the underestimating hyperplanes through the 4 feasible points, i.e., constraints (3.11.1), he uses the tightest overestimating hyperplanes

$$\sigma_{i,j} = \quad q_{i,j} + q_{i,j-1},$$
$$\sigma_{i,j} = 2 - q_{i,j} - q_{i,j-1},$$

and requires the switching variable to be equal to a convex combination of those:

$$\sigma_{i,j} = \alpha_{i,j}\,(q_{i,j} + q_{i,j-1}) + (1 - \alpha_{i,j})\,(2 - q_{i,j} - q_{i,j-1}). \tag{3.48}$$

Naturally, for the binary values, the attainable minimum values are exactly the desired values since they lie on one of the overestimating planes and fulfill the corresponding equation. Figure 3.13 shows the attainable minimal values for the switching variables $\sigma_{i,j}$. They are attained choosing the free convex multiplier $\alpha_{i,j}$ according to the choice of $q_{i,j}$ and $q_{i,j-1}$:

$$\alpha_{i,j} = \begin{cases} 1, & \text{if } q_{i,j} + q_{i,j-1} \leqslant 1, \\ 0, & \text{else.} \end{cases}$$

For fractional values of $\boldsymbol{q}$, the values of the switch variables are as high as possible while

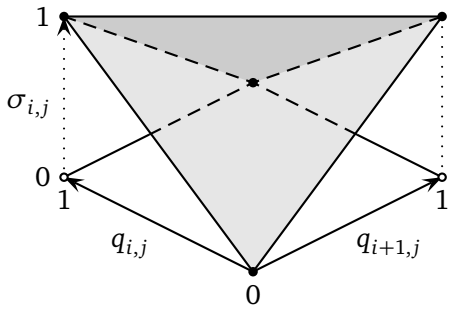


Figure 3.13: Minimal values of the convex combination (3.48) of the tightest overestimating hyperplanes for the switch values.

still being on at least one hyperplane through 3 of the 4 binary points, i.e., fractional values are punished as strongly as possible with a combination of the hyperplanes through the points. He describes the numerical results to be promising with this reformulation even though the problem's size increases through the addition of the new convex multipliers $\boldsymbol{\alpha}$ and the switching reformulation is non-convex since any nonlinear equation, e.g. equation (3.48), is non-convex. However, it can be relaxed to the constraint

$$\sigma_{i,j} \geqslant \alpha_{i,j}\,(q_{i,j} + q_{i,j-1}) + (1 - \alpha_{i,j})\,(2 - q_{i,j} - q_{i,j-1}), \tag{3.49}$$

which remains a valid formulation as long as switches only have a negative influence and is easier to handle. This formulation allows additional "fake" switches to appear in the switching variables $\boldsymbol{\sigma}$, which are not reflected in the controls $\boldsymbol{q}$, and hence it is only a valid reformulation if the optimizer tries to minimize each switch. However, the arising system has advantageous numerical properties.

3.11.2 Min-up time

There are several ways to model min-up times for the system's modes. The most general modeling approach forces the system to stay in mode j for $\zeta_{i,j}$ time units after it switched from mode i to mode j, i.e.,

$$q_{i,k} = 1, q_{j,k+1} = 1 \qquad \Rightarrow \qquad q_{j,k+l} = 1, \text{ for } 1 \leqslant l \leqslant \zeta_{i,j}.$$

The resulting formulation introduces new variables $\sigma_{i,j,k}$, which measure switches from mode i to j in time step k:

$$\sigma_{i,j,k} = \begin{cases} 1, & \text{if } q_{i,k} = 1, q_{j,k+1} = 1, \\ 0, & \text{else.} \end{cases}$$

These switching variables are then used to enforce the mode for the following time steps:

$$q_{j,l} \geqslant \sigma_{i,j,k}, \qquad k+1 \leqslant l \leqslant k+\zeta_{i,j}.$$

The definition of the switching variables could either be done with linear constraints to be

$$\sigma_{i,j,k} \geqslant -1 + q_{i,k} + q_{j,k+1},$$

or as for the switches with the tightest overestimating hyperplanes, which gives the constraints

$$\sigma_{i,j,k} \geqslant \alpha_{i,j,k}\, q_{i,k} + (1-\alpha_{i,j,k})\, q_{j,k+1}.$$

The linear formulation produces a quite weak relaxation, since it does not provide any tightening for $q_{i,k} + q_{j,k+1} \leqslant 1$. The second formulation considers any setting where $q_{i,k} > 0$ and $q_{j,k+1} > 0$ at least partly as a switching from mode i to mode j. However, as in the switching variable case from the previous section, this formulation needs quite some additional variables. The constraint to enforce a certain mode can also be tightened to be

$$q_{j,l+1} \geqslant \sum_{k=l+1-\zeta_{i,j}}^{l} \sigma_{i,j,k},$$

since a switch from mode i to mode j can only happen once in $[t_l, t_{l+\zeta_{i,j}}]$. This finally also does not favor fractional solutions in the $\boldsymbol{q}$-variables.

3.12 Mixed-Integer Nonlinear Programming

The discretization of an MIOCP is a MINLP. MINLPs meld the combinatorial difficulties of discrete variables with the difficulties created by nonlinear functions. They represent a very difficult class of problems and have been getting more and more interest in the last two decades. They are already quite difficult for small, static problems and become even more so for the discretized MIOCPs due to the usually larger problems' dimensions. Some recent surveys on this topic are given in [17, 36, 84].

This section gives a broad overview on the algorithmic building blocks to solve these problems. First, we briefly present the simple enumeration idea and its obstacles in Section 3.12.1. Then in Section 3.12.2, we elaborate on relaxation techniques – some are similar to the ones presented in Sections 3.5–3.9 to relax the integrality constraint, while other techniques relax the nonlinearity with simple, convex formulations. Also a small overview on current cutting plane techniques is given in Section 3.12.3 and at last in Section 3.12.4, we present the main ingredients needed to construct a branch-and-bound or branch-and-cut framework for MINLPs.

3.12.1 Enumeration

The simplest idea to solve a MINLP is to enumerate all possible choices for the integer controls and then solve the resulting NLPs. However, the complete enumeration of all possible choices might give a huge number of NLPs to solve. The exponential growth of the number of problems to solve is often referred to as the *combinatorial explosion* and especially holds for discretized MIOCPs. This is the case since the number of binary control variables depends on the number of control discretization intervals m and is $m\, n_\omega$ – the number of problems to be solved is hence $2^{m\, n_\omega}$.

To emphasize the meaning of this, we consider a special case. Assume we have a problem with 3 controls and a time discretization of 10 control intervals – a pretty small problem with a coarse time discretization – and assume each nonlinear subproblem is solved in one second. Then, the algorithm would need to solve 2^{30} problems, which would take more than 34 years. Therefore, the computational effort quickly prevents this technique from being successfully applied.

The dynamic programming approach from Section 3.3 is an augmentation of this complete enumeration approach. It dissects the time horizon into several parts – naturally into the same intervals as the control discretization – and enumerates all solutions for each slice. Then, instead of searching for all valid combinations of solutions over these time slices, the *cost-to-go function* is used to deduce sub-optimality of some combinations. However, an additional drawback of this method is that the continuous state and control spaces have to be discretized, which leads to a certain inexactness.

3.12.2 Relaxations

All methods that do not rely on enumeration for the integral variables need some kind of *integral relaxation*. Therefore, they need to be expressed with integral or binary variables, where the integrality constraint can be dropped. These formulations with binary variables were covered in Sections 3.5–3.9 for MIOCPs. The integral relaxations just drop the integrality constraint in the problem formulation and thereby obtain an NLP.

Polyhedral relaxations

A second form of relaxations is used for convex constraints and convex NLPs. Instead of including the convex constraints $c(\boldsymbol{x}) \leqslant 0$ into the problem, each constraint is relaxed with a set of n_j supporting hyperplanes that are obtained through first-order TAYLOR series approximations of the constraint in the points $\boldsymbol{x}_j^i$:

$$0 \geqslant c_j\left(\boldsymbol{x}_j^i\right) + \nabla c_j\left(\boldsymbol{x}_j^i\right)^T \left(\boldsymbol{x} - \boldsymbol{x}_j^i\right), \qquad 0 \leqslant i \leqslant n_\mathrm{j}, 0 \leqslant j \leqslant n_\mathrm{c}.$$

This kind of *polyhedral relaxation* is used in the *outer approximation* method – going back to [109]. The outer approximation method alternatively solves MILPs, which are created with those polyhedral relaxations, and NLPs, which check the integral solution's feasibility with regard to the original constraints with fixed integral variables. The NLP's solution is used to cut off the point if it was infeasible in the original setting. Otherwise, it is used to enforce a decrease in the upper bound on the objective as it provides a feasible point of the MINLP.

These polyhedral relaxations are also used in *Generalized* BENDER*'s Decomposition*, cf. [72]. This approach is similar to the outer approximation approach but creates a different MILP – here, a single constraint is used that combines all linearizations into one aggregated constraint. Generalized BENDER's Decomposition uses duality theory to derive this cut and in contrast to the outer approximation methods needs dual variables in the solution of the NLP. It can be shown that the approach is just a further relaxation of the outer approximation approach.

Another variant of the outer approximation idea is the *extended cutting-plane method* introduced by WESTERLUND and PETTERSSON in [184]. This method completely passes on NLPs and solely relies on linear problems. The linearizations are obtained through gradient evaluations instead of solving the NLPs.

Relaxations of non-convexity

The most common approach when trying to solve non-convex problems to optimality is to derive *convex underestimators* of the non-convex functions and thereby obtain a convex relaxation. The emerging convex relaxation can then be treated with the other methods from this section. A recent survey on global nonlinear optimization, which also covers MINLP formulations as well as convex DAE relaxations, is given in [66]. As the convex relaxation's solution might not sufficiently satisfy the original constraint, it might be necessary to refine the convexification to adaptively obtain an optimal solution of the original, non-convex problem. This refinement of the convexification is usually achieved by means of a *spatial branch-and-bound* algorithm. This algorithm creates subproblems by subdividing the feasible space of the continuous variables and adapts the convex relaxations of the subproblems using the variables' tighter bounds. The whole process may be strongly accelerated if the initial bounds on the variables are tightened using a *bound tightening* procedure (*feasibility-based bound tightening* or *optimality-based bound tightening*).

For nonlinear equations, convex envelopes (convex under- and concave overestimators) are needed. Since the discretized ODE system is usually such a function, special emphasis must be put on the convex envelope calculation in this case. The integrator, which solves the ODE, must be capable to handle interval arithmetics to provide a tight relaxation. Otherwise, an exponential growth in the reachable set cannot be prevented. Further information on integration with interval arithmetics can be found in [123, 168].

3.12.3 Cutting planes

To be able to implement cutting plane algorithms for convex MINLPs, the relaxed solutions need to lie on the boundary of the feasible region. This can be guaranteed by replacing any convex objective $\Phi(\boldsymbol{x})$ by the objective η with the additional convex constraint $\eta \geqslant \Phi(\boldsymbol{x})$. Now, cutting plane algorithms can be applied, which tighten the feasible set through additional constraints. The constraints generated in the following example procedures are always half-spaces generated by hyperplanes since the procedures usually have been derived from previously known versions that are used to solve MILPs.

Mixed-Integer Rounding cuts are a generalization of the purely integral *Gomory cuts*. They use the branching dichotomy $x \leqslant \lfloor \bar{x} \rfloor$ or $x \geqslant \lceil \bar{x} \rceil$ for integral x and the fractional solution $\bar{x}$ of the integral relaxation together with single linear inequality constraints. Depending on the linear inequality, a tighter linear inequality may be generated, which cuts off the fractional solution. This approach only works on linear cuts, but those are often created and used in different MINLP algorithms.

Perspective cuts are outer approximation cuts, i.e., supporting hyperplanes that are first-order TAYLOR series approximations of convex constraints, of the convex hull obtained through the perspective formulation. The addition of these cuts might be advantageous in comparison to the complete perspective formulation, and in the limit of generating all separating hyperplanes, the two formulations coincide.

Disjunctive cuts use any disjunction of the feasible set to split the feasible set into two disjoint subsets. One such disjunction could be the branching dichotomy on an integral variable (the split disjunction). The convex hull of the two subsets can be described with a convex formulation using the perspective formulation presented in Section 3.9. This requires lifting and then the separation procedure can calculate a cut in the higher dimensional space obtained from the perspective formulation. However, the perspective formulation comes with numerical problems and the lifted problem has twice the size of the relaxed problem, hence cutting procedures with disjunctive cuts are usually slow. However, BONAMI [34] overcomes these difficulties for disjunctions created from splits. KILINÇ et al. [110] solve a series of Linear Programs (LPs) instead of a separation NLP to speed up and stabilize the process.

3.12.4 Nonlinear branch-and-bound

A *nonlinear branch-and-bound* algorithm is created with the same principles in mind as its linear version. The problem's integral relaxation is solved. If the solution is fractional, branching on the integral variables is executed, which excludes the fractional optimal point and the two subproblems are solved and so on – it is an implementation of the classical divide-and-conquer principle. The problems can be displayed in a tree structure where each node corresponds to a problem with a set of fixed integral variables, cf. Figure 3.14. Using only branching, the algorithm would not perform much better than any enumeration method described in Section 3.12.1. However, nodes can be pruned due to several observations:

1. Lower bounds are obtained for each subproblem automatically as the integer relaxation NLP is solved – non-convex problems need to be solved globally for this to hold. This bound describes the best achievable value for this problem and all subproblems. If an integral possibly suboptimal solution to the original MINLP is known, it can be used to prune parts of the tree, which cannot generate a better solution.

2. Infeasible nodes can be pruned and their corresponding subtree can be disregarded since subproblems cannot become feasible again.

3. Nodes with integral solutions also do not need to be examined any further since they already realize the best solution obtainable for all subproblems that could be created from these nodes.

These pruning rules prevent the whole tree from being searched. In practice, most MINLP solvers contain a branch-and-bound framework. There are several choices when implementing these algorithms, which shall just briefly be mentioned here. The algorithms may combine all those choices and change between different settings as they progress.

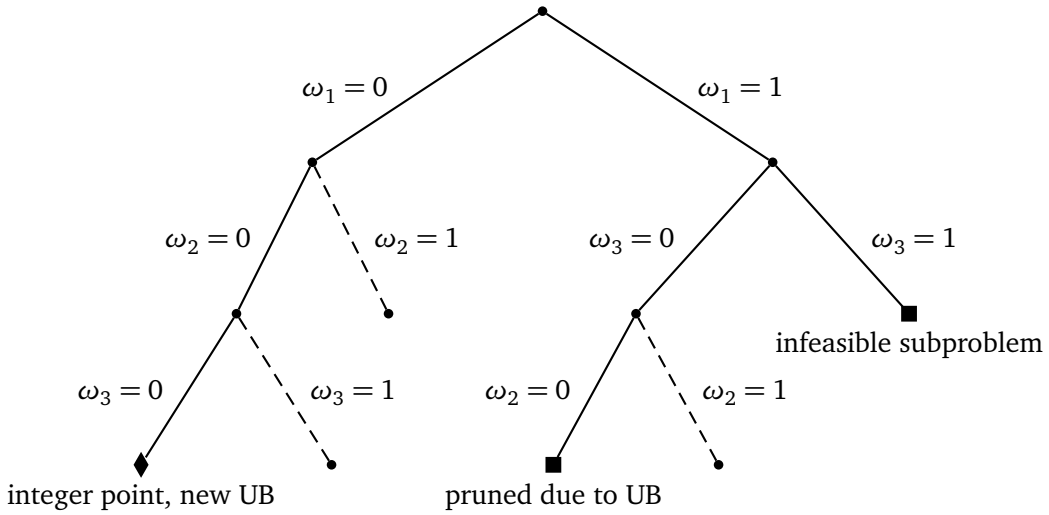


Figure 3.14: Branch-and-bound tree

The branching rules, i.e., how child nodes are generated from a node with a fractional solution, are freely selectable for the implementation as long as the fractional solution is infeasible for both child problems. There are several different branching strategies presented in [37] for nonlinear branch-and-bound and in [2] for linear branch-and-bound, which can directly be translated for MINLP algorithms. One example is *maximum fractional branching*, where dichotomy branching is done on the variable with the maximum distance from an integral value. A second strategy is *strong branching* where the child nodes are solved for all integer variables with fractional value in the relaxation. Then, the variable to branch on is chosen

with the evaluation of the *degradation*, i.e., the increase in the lower bound, of both subproblems, if both subproblems are feasible. Infeasible subproblems can be used to tighten a node or even to prune it. *Pseudocost branching* does not compute the correct degradations by solving all subproblems, but keeps a history of achieved degradations for all variables and solved subproblems as pseudocosts, which measure a degradation per unit of fractionality. These pseudocosts are used with the current fractionalities to estimate the effect that branching on a certain variable would have and the variable is chosen based on a certain criterion. A combination of strong branching and pseudocost branching is *reliability branching*, which uses strong branching early until reliable pseudocosts are derived and then uses pseudocosts. Another, less often used branching strategy is branching on general disjunctions, which is for example used for *SOS*1-constraints.

A second choice for the implementation are the node selection rules, i.e., which of all open subproblems is solved next? The general methodologies are *depth-first search* and *best-first search*, which choose the deepest or shallowest nodes, respectively. Depth-first search focuses on finding good integral solutions fast and has to manage only a small-sized tree, while best-first search focuses on finding the best solution as fast as possible but often has very large sets of open subproblems. An often used strategy combines the two methods: the *diving* strategy implements depth-first search until an integral solution is found, then backtracks to the best open subproblem and starts a new depth-first search from there on. For this part of the algorithm, methods from linear branch-and-bound frameworks, as presented in [2, 124], may be adopted to the nonlinear setting. However, their performance can differ from the linear setting. An example is that there are no good warm-start techniques for MINLP with interior point methods and hence the application of depth-first search in this context loses its advantage coming from the usage of the *dual simplex* in the linear case, cf. [37].

An important part of the branch-and-bound algorithm is the pruning step, which is stronger when good integral solutions are known. A possibility of getting these solutions are heuristics. Heuristics are naturally also very important when algorithms need to be stopped before completion due to time constraints, since then at least a good feasible solution can be returned after the algorithm is stopped. There are certain heuristics that are based on the fractional solution of a subproblem and try to find a close integral solution. One such strategy is *MILP-based rounding*, cf. [133], where a close integral solution in the ℓ_1-norm is computed for a polyhedral approximation of the original problem. This integral solution is put into the MINLP, all integral variables are fixed and a solution of the original MINLP is derived. The *feasibility pump* heuristic is a close relative, cf. [35]. There are lots of other heuristics and also the diving strategy for node selection implicitly is a heuristic, cf. [17] for detailed descriptions of several heuristics.

Cutting planes, cf. Section 3.12.3, can easily be included into a branch-and-bound algorithm to obtain a branch-and-cut algorithm. After solving each subproblem, the solution can be cut off via the cutting plane approach. This is repeated until a certain threshold is reached, then branching on the remaining problem is done. Usually, more cuts are generated close to the

root node since they are applicable for large parts of the branching tree. Deeper in the tree, less effort is spent on local cuts since those would only be valid for a smaller subtree.
An alternative to the full nonlinear branch-and-bound algorithm is the LP/NLP-based branch-and-bound algorithm introduced in [147]. It is also a close relative to the outer approximation technique described in Section 3.12.2. As explained there, it uses an MILP master problem with first-order TAYLOR series approximations of the nonlinear functions. Instead of resolving the master problem, only one master problem is solved. Once an integer point is reached, the NLP is solved that corresponds to this setting of integer variables. The solution is used to update all open nodes of the master problem with a new outer approximation and it is additionally used to enhance the global upper bound if feasible. An implementation of this technique can be found in LEYFFER's thesis [119]. The general idea can also be extended to Generalized BENDER's Decomposition and all other methods that only focus on the addition of cutting planes.

4 The Control Approximation Problem for Control-Affine Systems

In this chapter we derive the *Control Approximation Problem* for control-affine systems, i.e., problems where, for fixed states, the controls only enter linearly. These control-affine systems are, e.g., obtained through an Outer Convexification of the ODE system. The theoretical behavior of the Control Approximation Problem is studied from simple settings to more and more complex settings. Different heuristics are given as well as a solution approach that finds the optimal solution.

We give the theoretical justification of the Control Approximation Problem in Section 4.2 since it is a central building stone for the main contributions of this thesis. Then, we carry the observations over to discrete systems, i.e., systems that we obtain after discretization. Some other problems, e.g. turn-based games as in [59, 156], fit into this setting as well.

In Section 4.3, we derive the Control Approximation Problem in its most simple setting, i.e., one control and no additional constraints. Then, we give the Sum-up Rounding (SUR) heuristic from [152] and analyze its behavior. We also present conditions for optimality of its solution. Some of the analysis was previously done by SAGER, BOCK and DIEHL in [157], but most are new contributions. Furthermore, we examine the LAGRANGIAN relaxation of the problem and give its analytical solutions inside a branching framework as described in our work [104].

Section 4.4 covers the extension to the multi-dimensional setting of the Control Approximation Problem obtained through the OC technique. Therein, SOS1-constraints couple the different controls, which is the main difference to the one-dimensional or decoupled case. We present the SOS1-SUR heuristic from [152] and outline its weakness as already done in [157]. In contrast, we give a new heuristic, the so-called Next-forced Rounding (NFR) heuristic, which overcomes this weakness. Additionally, some of the insights gained for the LAGRANGIAN relaxation of the one-dimensional problem are carried over to this setting.

In Section 4.3, the Control Approximation Problem with added combinatorial constraints is considered. Additionally, we explain why additional combinatorial constraints may be necessary even if they are not inherently included in the model. The derived problem becomes an MILP and the polyhedral structure of the Control Approximation Problem is examined. It is found to be very data dependent and unstructured. Therefore, the common MILP technique of using cutting planes is hardly applicable in this setting. As an alternative, the branch-and-bound algorithm from our work [104] is presented. It uses the LAGRANGIAN relaxation for

bound calculation.

4.1 Problem formulation

SAGER presented a novel decomposition approach to solve MIOCPs in his dissertation [152]. In this approach, firstly, an OCP is solved in which the integer variables are relaxed to be continuous, and secondly, the obtained relaxed controls are approximated by integer controls in the *integral sense*. The quality of the second step is specified in Theorem 4.1. In [157], SAGER et al. refined the theoretical foundations of the idea. The solution process of the relaxed problem – the OCP – is described in Chapter 2.
Consider an MIOCP whose relaxation is formulated such that the integer controls enter the ODEs affinely – this may be obtained with the OC technique from Section 3.6:

$$
\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{\omega}(\cdot)} \quad & \Phi(\boldsymbol{x}(t_\mathrm{f})) \\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \sum_{i=1}^{n_\omega} \omega_i(t)\, \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i), \\
& \mathbf{0} \leqslant \boldsymbol{g}(\boldsymbol{\omega}(t), \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t))), \\
& 1 = \sum_{i=1}^{n_\omega} \omega_i(t), && t \in [t_0, t_\mathrm{f}], \\
& \mathbf{0} = \boldsymbol{r}^{\mathrm{eq}}((t_i, \boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}), && \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0, t_\mathrm{f}], \\
& \mathbf{0} \leqslant \boldsymbol{r}^{\mathrm{in}}((t_i, \boldsymbol{x}(t_i))_{0\leqslant i\leqslant m}), && \{t_i\}_{0\leqslant i\leqslant m} \subset [t_0, t_\mathrm{f}], \\
& \boldsymbol{u}(t) \in \mathcal{U}(t), && t \in [t_0, t_\mathrm{f}], \\
& \boldsymbol{v}(t) \in \Omega \subset \mathbb{R}^{n_\mathrm{v}}, |\Omega| < \infty, && t \in [t_0, t_\mathrm{f}], \\
& \boldsymbol{\omega}(t) \in \{0,1\}^{n_\omega}, && t \in [t_0, t_\mathrm{f}], \\
& \boldsymbol{\omega}(\cdot) \in \Omega_\omega \subseteq \mathcal{L}^1([t_0, t_\mathrm{f}]),
\end{aligned}
$$

where $\boldsymbol{g}(\boldsymbol{\omega}(t), \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)))$ realizes the relaxed formulation of the mode-dependent constraints, e.g. according to one of the choices given in Sections 3.5–3.9.
The decomposition approach now proposes to decompose this problem into several different problems:

1. Solve the relaxed OCP, where the integrality constraint is relaxed to be $\boldsymbol{\omega}(t) \in [0,1]^{n_\omega}$ and the additional combinatorial constraints $\boldsymbol{\omega}(\cdot) \in \Omega_\omega$ may be relaxed or even dropped. This gives a lower bound on the optimal solution and a state trajectory $\boldsymbol{x}(\cdot)$ obtained through relaxed controls $\boldsymbol{\alpha}(\cdot)$.

2. Approximate the relaxed controls $\boldsymbol{\alpha}(\cdot)$ of the solution with integer controls $\boldsymbol{\omega}(\cdot)$ through

the solution of the Control Approximation Problem with an appropriate norm

$$\begin{aligned}
\min_{\boldsymbol{\omega}} \quad & \max_{t\in[t_0,t_\mathrm{f}]} \left\| \int_{t_0}^{t} \alpha(\tau) - \omega(\tau)\,\mathrm{d}\tau \right\| \\
\text{s.t.} \quad & 1 = \sum_{i=1}^{n_\omega} \omega_i(t), && t \in [t_0, t_\mathrm{f}], \\
& \boldsymbol{\omega}(t) \in \{0,1\}^{n_\omega}, && t \in [t_0, t_\mathrm{f}], \\
& \boldsymbol{\omega}(\cdot) \in \Omega_\omega.
\end{aligned}$$

Theorem 4.1 guarantees that the deviation of states obtained through the relaxed controls from states obtained through the integer controls, which solve the corresponding Control Approximation Problem, is bounded linearly by the objective of the Control Approximation Problem. This means that a close approximation of the controls entails a close approximation of the states. As long as this approximation can be made quite close, the states obtained from the relaxed controls are almost matched and the mixed path and control constraints can be fulfilled up to a certain threshold. The choice of an appropriate norm is discussed in Section 4.5.1.

3. Another OCP with the integral controls fixed through the solution $\boldsymbol{\omega}(\cdot)$ of the Control Approximation Problem is solved to obtain consistent states and corresponding continuous controls. If the constraint violation is below a given threshold, the scheme obtained a feasible point of the original MIOCP. If the approximation's solution is good, the relaxed solution should be approximated closely and also the upper bound obtained by this new integer solution should approximate the lower bound obtained from the relaxation.

There are means to reiterate the process if the constraints are violated beyond the threshold. Between iterations, the control grid usually has to be refined to improve the approximation. If there are no additional combinatorial constraints, the SOS1-SUR heuristic and the NFR heuristic can approximate the relaxed controls arbitrarily close as long as the grid can be made sufficiently fine. We can thereby provide an arbitrarily good approximative solution of the original problem in comparison to the relaxed solution and obtain a solution scheme for the original problem up to a given precision. In presence of combinatorial constraints, this property is unfortunately lost and the scheme becomes a heuristic. It can be used, though, to either get a good solution much faster than the optimal solution of the MIOCP or to initialize an MINLP algorithm with.

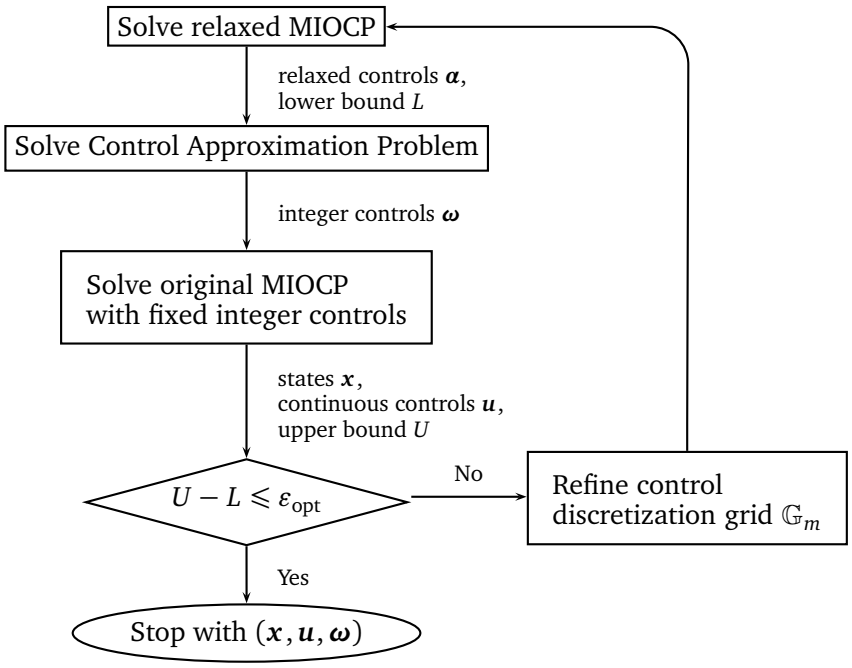


Figure 4.1: Scheme to obtain a solution with certain maximal deviation ε_{opt} from an optimal solution when no additional combinatorial constraints are present.

4.2 Approximation of differential states

We want to compare the evolution of two differential states $\boldsymbol{x}(\cdot)$ and $\boldsymbol{y}(\cdot)$ belonging to the same ODE system but driven by two different controls $\boldsymbol{\alpha}(\cdot)$ and $\boldsymbol{\omega}(\cdot)$. Of particular interest is the distance between the differential states dependending on the distance of the controls. The following variant of GRÖNWALL's Lemma is taken from [78].

Lemma 4.1 (Variant of GRÖNWALL's Lemma)
Let $[t_0, t_f]$ be an interval and $w, z : [t_0, t_f] \to \mathbb{R}$ integrable functions. If for a constant $L \geqslant 0$

$$w(t) \leqslant z(t) + L \int_{t_0}^{t} w(\tau)\,\mathrm{d}\tau \tag{4.1}$$

holds for $t \in [t_0, t_f]$ almost everywhere, then also

$$w(t) \leqslant z(t) + L \int_{t_0}^{t} e^{L(t-\tau)} z(\tau)\,\mathrm{d}\tau \tag{4.2}$$

holds for $t \in [t_0, t_f]$ almost everywhere.

If $z(\cdot)$ is additionally essentially bounded*, i.e., $z(\cdot) \in L^\infty([t_0, t_f], \mathbb{R})$ then*

$$w(t) \leqslant ||z(\cdot)||_\infty \, e^{L(t-t_0)} \tag{4.3}$$

holds for $t \in [t_0, t_f]$ almost everywhere. △

Proof Define $a : [t_0, t_\mathrm{f}] \to \mathbb{R}$ through

$$a(t) \stackrel{\text{def}}{=} L \int_{t_0}^{t} w(\tau) \, \mathrm{d}\tau, \tag{4.4}$$

it is absolutely continuous since w is integrable.
Due to the assumption, we have

$$w(t) = z(t) + a(t) + \delta(t) \tag{4.5}$$

with δ non-positive and integrable, i.e., $\delta \in L^1([t_0, t_\mathrm{f}], \mathbb{R}_0^-)$. Plugging (4.5) into (4.4) yields

$$a(t) = L \int_{t_0}^{t} z(\tau) + a(\tau) + \delta(\tau) \, \mathrm{d}\tau. \tag{4.6}$$

Differentiating both sides yields the following inhomogeneous, linear differential equation

$$\dot{a}(t) = La(t) + L(z(t) + \delta(t))$$

with initial value $a(t_0) = 0$. This equation can be solved analytically and we get

$$a(t) = L \int_{t_0}^{t} e^{L(t-\tau)}(z(\tau) + \delta(\tau)) \, \mathrm{d}\tau,$$

which can now be used to eliminate a from (4.5) to obtain

$$w(t) = z(t) + \delta(t) + L \int_{t_0}^{t} e^{L(t-\tau)}(z(\tau) + \delta(\tau)) \, \mathrm{d}\tau.$$

The first claim can directly be derived off of this equation and the non-positivity of δ.
The second claim can now be derived with the first claim and the essential bound as follows

$$\begin{aligned} w(t) &\leqslant z(t) + L \int_{t_0}^{t} e^{L(t-\tau)} z(\tau) \, \mathrm{d}\tau \\ &\leqslant ||z(\cdot)||_\infty \left(1 + L \int_{t_0}^{t} e^{L(t-\tau)} \, \mathrm{d}\tau \right) \\ &= ||z(\cdot)||_\infty \, e^{L(t-t_0)}. \end{aligned}$$

□

Now, consider the following setting. We have an initial value problem, which is linear in the control, i.e.,

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t, \boldsymbol{x}(t))\, \boldsymbol{\alpha}(t), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x_0}. \tag{4.7}$$

$\boldsymbol{A} \in \mathbb{R}^{n_x \times n_\omega}$ may depend nonlinearly on t and $\boldsymbol{x}(t)$. In addition, a term independent of $\boldsymbol{\alpha}$ could be included through an extra component of $\boldsymbol{\alpha}$, which is then fixed to 1. Under additional assumptions, i.e., $\boldsymbol{A}$ is differentiable, LIPSCHITZ continuous with respect to $\boldsymbol{x}(\cdot)$ and $\boldsymbol{y}(\cdot)$ and essentially bounded on the time horizon $[t_0, t_f]$, the following theorem gives the dependence of the difference of two solutions of the same IVP with respect to the difference between the control inputs. The theorem stems from [157].

Theorem 4.1
Let $\boldsymbol{x}(\cdot)$ and $\boldsymbol{y}(\cdot)$ be solutions of the initial value problems

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t, \boldsymbol{x}(t))\, \boldsymbol{\alpha}(t), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x_0}, \tag{4.8a}$$

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{A}(t, \boldsymbol{y}(t))\, \boldsymbol{\omega}(t), \qquad \boldsymbol{y}(t_0) = \boldsymbol{y_0}, \tag{4.8b}$$

with $t \in [t_0, t_f]$, given measurable functions $\boldsymbol{\alpha}, \boldsymbol{\omega} : [t_0, t_f] \to [0,1]^{n_\omega}$ and let $\boldsymbol{A} : \mathbb{R}^{n_x+1} \to \mathbb{R}^{n_x \times n_\omega}$ be differentiable. If a vector norm $||\cdot||$ and a consistent matrix norm $|||\cdot|||$ exist together with positive numbers $\theta, C, L, M \in \mathbb{R}^+$ such that for all $t \in [t_0, t_f]$:

$$\left|\left|\left| \frac{\mathrm{d}}{\mathrm{d}t} \boldsymbol{A}(t, \boldsymbol{x}(t)) \right|\right|\right| \leqslant C, \tag{4.8c}$$

$$\left|\left|\left| \boldsymbol{A}(t, \boldsymbol{y}(t)) - \boldsymbol{A}(t, \boldsymbol{x}(t)) \right|\right|\right| \leqslant L \left|\left| \boldsymbol{y}(t) - \boldsymbol{x}(t) \right|\right|, \tag{4.8d}$$

and $\boldsymbol{A}(\cdot, \boldsymbol{x}(\cdot))$ is essentially bounded by M on $[t_0, t_f]$, and it holds for all $t \in [t_0, t_f]$

$$\left|\left| \int_{t_0}^{t} \alpha(\tau) - \omega(\tau)\, \mathrm{d}\tau \right|\right| \leqslant \theta, \tag{4.8e}$$

then it also holds for all $t \in [t_0, t_f]$

$$\left|\left| \boldsymbol{y}(t) - \boldsymbol{x}(t) \right|\right| \leqslant \left(\left|\left| \boldsymbol{y_0} - \boldsymbol{x_0} \right|\right| + \left(M + C(t - t_0) \right) \theta \right) e^{L(t-t_0)}. \tag{4.8f}$$

△

Proof We define some auxiliary terms to shorten the notations:

$$\boldsymbol{\Delta\omega}(t) \overset{\text{def}}{=} \boldsymbol{\alpha}(t) - \boldsymbol{\omega}(t),$$

$$\boldsymbol{\Delta a}(t) \overset{\text{def}}{=} \int_{t_0}^{t} \boldsymbol{\Delta\omega}(\tau)\, \mathrm{d}\tau.$$

Notice that $\Delta a(t_0) = 0$ and due to (4.8e) it holds $||\Delta \boldsymbol{a}(t)|| \leqslant \theta$. Since the range space of $\boldsymbol{\alpha}$ and $\boldsymbol{\omega}$ is $[0,1]^{n_\omega}$, we have

$$||\boldsymbol{\alpha}(t)|| \leqslant 1,$$
$$||\boldsymbol{\omega}(t)|| \leqslant 1.$$

The FUNDAMENTAL THEOREM OF CALCULUS yields for $t \in [t_0, t_\mathrm{f}]$

$$\boldsymbol{x}(t) = \boldsymbol{x_0} + \int_{t_0}^{t} \dot{\boldsymbol{x}}(\tau)\,\mathrm{d}\tau = \boldsymbol{x_0} + \int_{t_0}^{t_\mathrm{f}} \boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\,\alpha(\tau)\,\mathrm{d}\tau,$$
$$\boldsymbol{y}(t) = \boldsymbol{y_0} + \int_{t_0}^{t} \dot{\boldsymbol{y}}(\tau)\,\mathrm{d}\tau = \boldsymbol{y_0} + \int_{t_0}^{t_\mathrm{f}} \boldsymbol{A}(\tau, \boldsymbol{y}(\tau))\,\omega(\tau)\,\mathrm{d}\tau,$$

With these observations, we now obtain for all $t \in [t_0, t_\mathrm{f}]$

$$\begin{aligned}
\big|\big|\boldsymbol{y}(t) - \boldsymbol{x}(t)\big|\big| &\leqslant \big|\big|\boldsymbol{y_0} - \boldsymbol{x_0}\big|\big| + \left|\left|\int_{t_0}^{t} \boldsymbol{A}(\tau, \boldsymbol{y}(\tau))\,\omega(\tau) - \boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\,\alpha(\tau)\,\mathrm{d}\tau\right|\right| \\
&\leqslant \big|\big|\boldsymbol{y_0} - \boldsymbol{x_0}\big|\big| + \left|\left|\int_{t_0}^{t} \boldsymbol{A}(\tau, \boldsymbol{y}(\tau))\,\omega(\tau) - \boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\,\omega(\tau)\,\mathrm{d}\tau\right|\right| \\
&\quad + \left|\left|\int_{t_0}^{t} \boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\,\omega(\tau) - \boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\,\alpha(\tau)\,\mathrm{d}\tau\right|\right| \\
&= \big|\big|\boldsymbol{y_0} - \boldsymbol{x_0}\big|\big| + \left|\left|\int_{t_0}^{t} \big(\boldsymbol{A}(\tau, \boldsymbol{y}(\tau)) - \boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\big)\ \omega(\tau)\,\mathrm{d}\tau\right|\right| \\
&\quad + \left|\left|\int_{t_0}^{t} \boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\,\Delta\boldsymbol{\omega}(\tau)\,\mathrm{d}\tau\right|\right| \\
&= \big|\big|\boldsymbol{y_0} - \boldsymbol{x_0}\big|\big| + \left|\left|\int_{t_0}^{t} \big(\boldsymbol{A}(\tau, \boldsymbol{y}(\tau)) - \boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\big)\ \omega(\tau)\,\mathrm{d}\tau\right|\right| \\
&\quad + \left|\left|\boldsymbol{A}(t, \boldsymbol{x}(t))\,\Delta\boldsymbol{a}(\boldsymbol{t}) - \int_{t_0}^{t} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\,\Delta\boldsymbol{a}(\tau)\,\mathrm{d}\tau\right|\right| \\
&\leqslant \big|\big|\boldsymbol{y_0} - \boldsymbol{x_0}\big|\big| + L\int_{t_0}^{t} \big|\big|\boldsymbol{y}(\tau) - \boldsymbol{x}(\tau)\big|\big|\,||\omega(\tau)||\ \mathrm{d}\tau \\
&\quad + |||\boldsymbol{A}(t, \boldsymbol{x}(t))|||\,\theta + \int_{t_0}^{t} \left|\left|\left|\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{A}(\tau, \boldsymbol{x}(\tau))\right|\right|\right| ||\Delta\boldsymbol{a}(\tau)||\ \mathrm{d}\tau \\
&\leqslant \big|\big|\boldsymbol{y_0} - \boldsymbol{x_0}\big|\big| + L\int_{t_0}^{t} \big|\big|\boldsymbol{y}(\tau) - \boldsymbol{x}(\tau)\big|\big|\ \mathrm{d}\tau + \big(M + C(t - t_0)\big)\,\theta,
\end{aligned}$$

with the only non-trivial step being a partial integration.
The functions

$$w(t) \stackrel{\text{def}}{=} \left|\left|\boldsymbol{y}(t) - \boldsymbol{x}(t)\right|\right|,$$
$$z(t) \stackrel{\text{def}}{=} \left|\left|\boldsymbol{y_0} - \boldsymbol{x_0}\right|\right| + \left(M + C(t - t_0)\right)\theta$$

satisfy the assumptions of Lemma 4.1, namely both are integrable and

$$z \in L^\infty([t_0, t_\mathrm{f}], \mathbb{R}).$$

The application of said lemma yields the claim

$$\left|\left|\boldsymbol{y}(t) - \boldsymbol{x}(t)\right|\right| \leqslant \left(\left|\left|\boldsymbol{y_0} - \boldsymbol{x_0}\right|\right| + \left(M + C(t - t_0)\right)\theta\right) e^{L(t-t_0)}.$$ □

Remark 4.1 (Quality of the approximation)
The theorem gives a linear dependence of the state error on the control deviation θ. In combination with the SUR strategy from Section 4.3.1, and if the controls enter independently of each other and are not coupled, this yields a linear dependence of the state error on the grid size Δt.

Remark 4.2 (Dependence of approximation on grid)
The direct consequence of Theorem 4.1 is that for the special IVP, which is linear in the integral controls, the state approximation is driven by the approximation of the controls in the integral sense and the initial error – which is usually 0. A numerical example for this theoretical behavior is given in Section 5.2 and in [157]. Section 4.3 treats the Control Approximation Problem in the integral sense for this setting.

Remark 4.3 (Convex right-hand side)
This theorem can easily be extended for right-hand side functions that are not affine in the controls but instead convex in the integer controls, when all other components are fixed. First, one rewrites the integer controls as a convex combination of the integer values. Then, one uses the convexity property to obtain an affine system and then carries on as above. However, using the OC technique makes the application of this observation unnecessary.

4.2.1 Translation to a discrete setting

In the following, we transfer Theorem 4.1 to a discrete setting. This is done such that the work carries over to the actually used implementations, cf. Section 2.4. All direct algorithms do not treat the ODE as is but instead transform them onto a discrete setting. The propositions are given as broadly applicable as possible but with the discretization of an ODE system in mind.

As a discrete variant of GRÖNWALL*'s Lemma* we use the sharpest version from HOLTE [98, Theorem 4]:

Lemma 4.2 (Sharp Discrete Variant of GRÖNWALL*'s Lemma*)
Assume (y_n), (f_n), *and* (g_n) *are non-negative sequences and*

$$y_n \leqslant f_n + \sum_{k=0}^{n-1} g_k \, y_k, \qquad \text{for } n \geqslant 0. \tag{4.9}$$

Then

$$y_n \leqslant f_n + \sum_{k=0}^{n-1} f_k \, g_k \prod_{j=k+1}^{n-1} \left(1 + g_j\right). \tag{4.10}$$

△

Proof For abbreviation, let

$$G_i^j \stackrel{\text{def}}{=} \prod_{k=i}^{j} \left(1 + g_k\right).$$

The proof is by induction. For $n = 0$, inequalities (4.9) and (4.10) are the same:

$$y_0 \leqslant f_0.$$

Assume that inequality (4.10) holds for $n \geqslant 0$. By assumption and induction hypothesis, we get

$$\begin{aligned}
y_{n+1} &\leqslant f_{n+1} + \sum_{i=0}^{n} g_i \, y_i \\
&\leqslant f_{n+1} + \sum_{i=0}^{n} g_i \left(f_i + \sum_{k=0}^{i-1} f_k \, g_k \, G_{k+1}^{i-1} \right) \\
&= f_{n+1} + \sum_{i=0}^{n} g_i \, f_i + \sum_{i=0}^{n} g_i \left(\sum_{k=0}^{i-1} f_k \, g_k \, G_{k+1}^{i-1} \right) \\
&= f_{n+1} + \sum_{i=0}^{n} g_i \, f_i + \sum_{k=0}^{n-1} f_k \, g_k \left(\sum_{i=k+1}^{n} g_i \, G_{k+1}^{i-1} \right) \\
&= f_{n+1} + \sum_{k=0}^{n} f_k \, g_k \left(1 + \sum_{i=k+1}^{n} g_i \, G_{k+1}^{i-1} \right).
\end{aligned}$$

We use the recursive definition of G, i.e.,

$$G_i^{j+1} = \left(1 + g_{j+1}\right) G_i^j, \qquad G_i^{i-1} = 1,$$

which yields

$$g_{j+1}\, G_i^j = G_i^{j+1} - G_i^j.$$

Now, we can get the proposed inequality through a telescope sum

$$\begin{aligned} y_{n+1} &\leqslant f_{n+1} + \sum_{k=0}^{n} f_k\, g_k \left(1 + \sum_{i=k+1}^{n} G_{k+1}^i - G_{k+1}^{i-1}\right) \\ &= f_{n+1} + \sum_{k=0}^{n} f_k\, g_k \left(1 + G_{k+1}^n - G_{k+1}^k\right) \\ &= f_{n+1} + \sum_{k=0}^{n} f_k\, g_k\, G_{k+1}^n. \end{aligned}$$

□

Consider the setting of a discrete process with the following state transfer equation:

$$\boldsymbol{x}^{\boldsymbol{n}} = \boldsymbol{x}^{\boldsymbol{n-1}} + \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{n-1}})\, \boldsymbol{\alpha}^{\boldsymbol{n-1}},$$

where $\boldsymbol{x}^{\boldsymbol{i}} \in \mathbb{R}^{n_x}$ are the states of the discrete process with initial states $\boldsymbol{x}^{\boldsymbol{0}}$, $\boldsymbol{A} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_\omega}$ is the state transfer and $\boldsymbol{\alpha}^{\boldsymbol{i}} \in \mathbb{R}^{n_\omega}$ are the controls. As in the continuous case, a term independent of $\boldsymbol{\alpha}$ could be included through an additional component of $\boldsymbol{\alpha}$, which would then be fixed to 1.

We also need another lemma as the discrete analogon to partial integration to be able to carry the theorem's proof over:

Lemma 4.3 (Discrete analogon to partial integration)
Assume that $(\boldsymbol{x}^{\boldsymbol{i}})_{0 \leqslant i \leqslant n}$ is a sequence in $\mathbb{R}^{n_x}$, $(\boldsymbol{\alpha}^{\boldsymbol{i}})_{0 \leqslant i \leqslant n}$ and $(\boldsymbol{\omega}^{\boldsymbol{i}})_{0 \leqslant i \leqslant n}$ are sequences in $\mathbb{R}^{n_\omega}$ and $\boldsymbol{A} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_\omega}$, then it holds for all $n \in \mathbb{N}$

$$\begin{aligned} \sum_{j=0}^{n} \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{j}}) \left(\boldsymbol{\alpha}^{\boldsymbol{j}} - \boldsymbol{\omega}^{\boldsymbol{j}}\right) = \quad & \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{n}}) \left(\sum_{i=0}^{n} \boldsymbol{\alpha}^{\boldsymbol{i}} - \boldsymbol{\omega}^{\boldsymbol{i}}\right) \\ & + \sum_{j=0}^{n-1} \left(\boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{j}}) - \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{j+1}})\right) \left(\sum_{i=0}^{j} (\boldsymbol{\alpha}^{\boldsymbol{i}} - \boldsymbol{\omega}^{\boldsymbol{i}})\right). \end{aligned}$$

△

Proof The claim can be proven through a sequence of reformulations:

$$\begin{aligned} \sum_{j=0}^{n} \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{j}}) \left(\boldsymbol{\alpha}^{\boldsymbol{j}} - \boldsymbol{\omega}^{\boldsymbol{j}}\right) &= \sum_{j=0}^{n} \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{j}}) \left(\sum_{i=0}^{j} \left(\boldsymbol{\alpha}^{\boldsymbol{i}} - \boldsymbol{\omega}^{\boldsymbol{i}}\right) - \sum_{i=0}^{j-1} \left(\boldsymbol{\alpha}^{\boldsymbol{i}} - \boldsymbol{\omega}^{\boldsymbol{i}}\right)\right) \\ &= \quad \sum_{j=0}^{n} \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{j}}) \left(\sum_{i=0}^{j} \boldsymbol{\alpha}^{\boldsymbol{i}} - \boldsymbol{\omega}^{\boldsymbol{i}}\right) - \sum_{j=1}^{n} \boldsymbol{A}(\boldsymbol{x}^{\boldsymbol{j}}) \left(\sum_{i=0}^{j-1} \boldsymbol{\alpha}^{\boldsymbol{i}} - \boldsymbol{\omega}^{\boldsymbol{i}}\right) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{j=0}^{n} A(x^j)\left(\sum_{i=0}^{j}\alpha^i-\omega^i\right)-\sum_{j=0}^{n-1}A(x^{j+1})\left(\sum_{i=0}^{j}\alpha^i-\omega^i\right)\\
&= A(x^n)\left(\sum_{i=0}^{n}\alpha^i-\omega^i\right)\\
&\quad+\sum_{j=0}^{n}\left(A(x^j)-A(x^{j+1})\right)\left(\sum_{i=0}^{j}\alpha^i-\omega^i\right).
\end{aligned}
$$

□

Under certain assumptions, we can now give the maximum deviation between two states x and y that belong to the same system but different controls α and ω.

Theorem 4.2

Let $(x^n)_{0\leqslant n\leqslant m}$ and $(y^n)_{0\leqslant n\leqslant m}$ be sequences in $\mathbb{R}^{n_x}$, which are given solutions of the discrete system

$$
\begin{aligned}
x^{n+1} &= x^n + A(x^n)\,\alpha^n,\\
y^{n+1} &= y^n + A(y^n)\,\omega^n,
\end{aligned}
$$

with $0\leqslant n<m$ for given control sequences $(\alpha^n)_{0\leqslant n<m},(\omega^n)_{0\leqslant n<m}\in[0,1]^{n_\omega m}$, initial values x^0 and y^0. If for given vector norm $||\cdot||$ and consistent matrix norm $|||\cdot|||$ there exist positive numbers L,M,C such that A is Lipschitz *continuous with constant L, bounded by M and its discretized derivative is bounded by C, i.e., for all $0\leqslant n\leqslant m$*

$$
\begin{aligned}
\left|\left|\left|A(x^n)-A(y^n)\right|\right|\right| &\leqslant L\left||x^n-y^n\right||,\\
|||A(x^n)||| &\leqslant M,\\
\left|\left|\left|A(x^n)-A(x^{n+1})\right|\right|\right| &\leqslant C,
\end{aligned}
$$

and if it holds for all $0\leqslant n<m$

$$
\left|\left|\sum_{i=0}^{n}\alpha^i-\omega^i\right|\right|\leqslant\theta,
$$

then it also holds for all $0\leqslant n<m$

$$
\left||x^n-y^n\right||\leqslant(1+L)^n\left||x^0-y^0\right||+\theta\,(M+(n-1)C)\left(1+L\sum_{k=0}^{n-2}(1+L)^k\right). \tag{4.11}
$$

△

Proof Due to the control space being $[0,1]^{n_\omega m}$, we have for $0\leqslant i<m$

$$
\begin{aligned}
\left||\alpha^i\right||&\leqslant 1,\\
\left||\omega^i\right||&\leqslant 1.
\end{aligned}
$$

We obtain for $1 \leqslant n \leqslant m$

$$
\begin{aligned}
\left|\left|\boldsymbol{x}^{n}-\boldsymbol{y}^{n}\right|\right| &\leqslant \left|\left|\boldsymbol{x}^{0}+\sum_{k=0}^{n-1} A(\boldsymbol{x}^{k})\,\boldsymbol{\alpha}^{k}-\boldsymbol{y}^{0}-\sum_{k=0}^{n-1} A(\boldsymbol{y}^{k})\,\boldsymbol{\omega}^{k}\right|\right| \\
&\leqslant \left|\left|\boldsymbol{x}^{0}-\boldsymbol{y}^{0}\right|\right|+\left|\left|\sum_{k=0}^{n-1} A(\boldsymbol{x}^{k})\,\boldsymbol{\alpha}^{k}-A(\boldsymbol{y}^{k})\,\boldsymbol{\omega}^{k}\right|\right| \\
&\leqslant \left|\left|\boldsymbol{x}^{0}-\boldsymbol{y}^{0}\right|\right|+\left|\left|\sum_{k=0}^{n-1} A(\boldsymbol{x}^{k})\,\boldsymbol{\alpha}^{k}-A(\boldsymbol{x}^{k})\,\boldsymbol{\omega}^{k}\right|\right| \\
&\quad+\left|\left|\sum_{k=0}^{n-1} A(\boldsymbol{x}^{k})\,\boldsymbol{\omega}^{k}-A(\boldsymbol{y}^{k})\,\boldsymbol{\omega}^{k}\right|\right| \\
&\leqslant \left|\left|\boldsymbol{x}^{0}-\boldsymbol{y}^{0}\right|\right|+\underbrace{\left|\left|\sum_{k=0}^{n-1} A(\boldsymbol{x}^{k})\left(\boldsymbol{\alpha}^{k}-\boldsymbol{\omega}^{k}\right)\right|\right|}_{\text{Lemma 4.3}} \\
&\quad+\sum_{k=0}^{n-1}\underbrace{\left|\left|\left|A(\boldsymbol{x}^{k})-A(\boldsymbol{y}^{k})\right|\right|\right|}_{\leqslant L\left|\left|\boldsymbol{x}^{k}-\boldsymbol{y}^{k}\right|\right|}\underbrace{\left|\left|\boldsymbol{\omega}^{k}\right|\right|}_{\leqslant 1} \\
&\leqslant \left|\left|\boldsymbol{x}^{0}-\boldsymbol{y}^{0}\right|\right|+L\sum_{k=0}^{n-1}\left|\left|\boldsymbol{x}^{k}-\boldsymbol{y}^{k}\right|\right| \\
&\quad+\left|\left|A(\boldsymbol{x}^{n-1})\left(\sum_{i=0}^{n-1}\boldsymbol{\alpha}^{i}-\boldsymbol{\omega}^{i}\right)\right.\right. \\
&\qquad\left.\left.+\sum_{j=0}^{n-2}\left(A(\boldsymbol{x}^{j})-A(\boldsymbol{x}^{j+1})\right)\left(\sum_{i=0}^{j}\boldsymbol{\alpha}^{i}-\boldsymbol{\omega}^{i}\right)\right|\right| \\
&\leqslant \left|\left|\boldsymbol{x}^{0}-\boldsymbol{y}^{0}\right|\right|+L\sum_{k=0}^{n-1}\left|\left|\boldsymbol{x}^{k}-\boldsymbol{y}^{k}\right|\right|+\left|\left|\left|A(\boldsymbol{x}^{n-1})\right|\right|\right|\left|\left|\sum_{i=0}^{n-1}\boldsymbol{\alpha}^{i}-\boldsymbol{\omega}^{i}\right|\right| \\
&\quad+\sum_{j=0}^{n-2}\left|\left|\left|A(\boldsymbol{x}^{j})-A(\boldsymbol{x}^{j+1})\right|\right|\right|\left|\left|\sum_{i=0}^{j}\boldsymbol{\alpha}^{i}-\boldsymbol{\omega}^{i}\right|\right| \\
&\leqslant \left|\left|\boldsymbol{x}^{0}-\boldsymbol{y}^{0}\right|\right|+\sum_{k=0}^{n-1}L\left|\left|\boldsymbol{x}^{k}-\boldsymbol{y}^{k}\right|\right|+M\theta+C(n-1)\theta \\
&= (M+C(n-1))\,\theta+\left|\left|\boldsymbol{x}^{0}-\boldsymbol{y}^{0}\right|\right|+\sum_{k=0}^{n-1}L\left|\left|\boldsymbol{x}^{k}-\boldsymbol{y}^{k}\right|\right|.
\end{aligned}
$$

At this point, we use Lemma 4.2 with the specifications

$$y_n \stackrel{\text{def}}{=} \left|\left|\boldsymbol{x}^{n}-\boldsymbol{y}^{n}\right|\right|,$$

$$f_n \overset{\text{def}}{=} \begin{cases} (M+(n-1)C)\,\theta, & \text{for } n>0, \\ \left|\left|\boldsymbol{x^0}-\boldsymbol{y^0}\right|\right|, & \text{for } n=0, \end{cases}$$

$$g_n \overset{\text{def}}{=} \begin{cases} L, & \text{for } n>0, \\ 1+L, & \text{for } n=0, \end{cases}$$

to obtain the claim

$$\begin{aligned} \left|\left|\boldsymbol{x^n}-\boldsymbol{y^n}\right|\right| &\leqslant (1+L)^n \left|\left|\boldsymbol{x^0}-\boldsymbol{y^0}\right|\right| \\ &\quad + \theta\left(M+(n-1)C+\sum_{k=1}^{n-1}(M+(n-1)C)\,L\,(1+L)^{n-k-1}\right) \\ &\leqslant (1+L)^n \left|\left|\boldsymbol{x^0}-\boldsymbol{y^0}\right|\right| + \theta\,(M+(n-1)C)\left(1+L\sum_{k=0}^{n-2}(1+L)^k\right). \end{aligned}$$

□

Analogously to the continuous case, this description can be applied to more general settings through some simple reformulations.

Remark 4.4 (Linearity of the controls)
The assumption that the discrete systems equations are linear in the controls $\boldsymbol{\alpha}$ is not a strong assumption for integer controls. We can use the OC technique described in Section 3.6 to obtain a system that is affine linear in the controls. Afterwards, we multiply the affine part with an auxiliary control, which is fixed to 1 to obtain the setting of the theorem.

Remark 4.5 (Direct dependence on the step number)
If the system equations depend directly on the step number, the system can be made autonomous by adding an additional state ν for the step number and the corresponding affine transition equation $\nu_{k+1} = \nu_k + 1$. The resulting system fits into the framework.

Remark 4.6 (Convex state transfer functions)
Theorem 4.2's statement also holds for state transfer functions that are convex in the controls when the states are fixed with the same reasoning as in the continuous case. However, using the OC technique makes the application of this additional observation unnecessary.

Discretization of an ODE system with the explicit EULER method

We verify Theorem 4.2 in a dynamic setting and show that it leads to the same qualitative conclusion as Theorem 4.1 in this setting. Consider the ODE system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(\boldsymbol{x}(t))\,\boldsymbol{\alpha}(t), \qquad t \in [t_0, t_\mathrm{f}],$$

discretized with equal time steps of size Δt and the explicit EULER method

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \Delta t \boldsymbol{F}(\boldsymbol{x}^k)\,\boldsymbol{\alpha}^k.$$

Since it is a convergent method, we know that, as we take finer grids, the discretized system's solution approximates the ODE system's solution increasingly well. Now, we quantify this behavior.

If we discretize the interval $[t_0, t_f]$ into m equal parts of size $\Delta t = \frac{1}{m}$, the state transfer function becomes

$$\boldsymbol{A}(\boldsymbol{x}^k) \stackrel{\text{def}}{=} \Delta t \boldsymbol{F}(\boldsymbol{x}^k).$$

Let the assumptions for the continuous case in Theorem 4.1 hold: $\boldsymbol{F}$ is essentially bounded by M, it is LIPSCHITZ continuous with constant L and its derivative is bounded by C. The deviation of the two controls in the integral sense is bounded by θ. This translates to the discretized setting as

$$\left|\left|\left|\boldsymbol{A}(\boldsymbol{x}^k)\right|\right|\right| = \Delta t \left|\left|\left|\boldsymbol{F}(\boldsymbol{x}^k)\right|\right|\right| \leqslant \frac{M}{m},$$

$$\left|\left|\left|\boldsymbol{A}(\boldsymbol{x}^k) - \boldsymbol{A}(\boldsymbol{y}^k)\right|\right|\right| = \Delta t \left|\left|\left|\boldsymbol{F}(\boldsymbol{x}^k) - \boldsymbol{F}(\boldsymbol{y}^k)\right|\right|\right| \leqslant \frac{L}{m} \left|\left|\boldsymbol{x}^k - \boldsymbol{y}^k\right|\right|,$$

$$\begin{aligned}\left|\left|\left|\boldsymbol{A}(\boldsymbol{x}^k) - \boldsymbol{A}(\boldsymbol{x}^{k+1})\right|\right|\right| &= \Delta t \left|\left|\left|\boldsymbol{F}(\boldsymbol{x}^k) - \boldsymbol{F}(\boldsymbol{x}^{k+1})\right|\right|\right| \leqslant \Delta t\, C \left|\left|\boldsymbol{x}^k - \boldsymbol{x}^{k+1}\right|\right| \\ &\leqslant \Delta t^2\, C \left|\left|\left|\boldsymbol{F}(\boldsymbol{x}^k)\right|\right|\right| \left|\left|\boldsymbol{\alpha}^k\right|\right| \leqslant \frac{CM}{m^2},\end{aligned}$$

$$\theta \geqslant \left|\left| \int_{t_0}^{t} \boldsymbol{\alpha}(\tau) - \boldsymbol{\omega}(\tau)\,\mathrm{d}\tau \right|\right| = \Delta t \left|\left| \sum_{k=0}^{n} \boldsymbol{\alpha}^k - \boldsymbol{\omega}^k \right|\right|.$$

Plugging these bounds into Theorem 4.2, we obtain for $0 \leqslant n \leqslant m$

$$\begin{aligned}\left|\left|\boldsymbol{x}^n - \boldsymbol{y}^n\right|\right| &\leqslant \left(1 + \frac{L}{m}\right)^n \left|\left|\boldsymbol{x}^0 - \boldsymbol{y}^0\right|\right| \\ &\quad + \frac{\theta}{\Delta t}\left(\frac{M}{m} + (n-1)\frac{C\,M}{m^2}\right)\left(1 + \frac{L}{m}\sum_{k=0}^{n-2}\left(1 + \frac{L}{m}\right)^k\right) \\ &\leqslant \left(1 + \frac{L}{m}\right)^{m\frac{n}{m}} \left|\left|\boldsymbol{x}^0 - \boldsymbol{y}^0\right|\right| \\ &\quad + \frac{\theta}{\Delta t}\frac{M}{m}\left(1 + \frac{n-1}{m}C\right)\left(1 + \frac{L}{m}\sum_{k=0}^{n-2}\left(1 + \frac{L}{m}\right)^{m\frac{k}{m}}\right).\end{aligned}$$

If we now focus on a single point t in the time interval $[t_0, t_f]$, the ratio $\tau \stackrel{\text{def}}{=} \frac{n}{m} = \frac{t-t_0}{t_f-t_0}$ stays constant while we can arbitrarily increase m. This leads to

$$\begin{aligned}
\left\|\boldsymbol{x}(t)-\boldsymbol{y}(t)\right\| &\leqslant \left(1+\frac{L}{m}\right)^{m\tau} \left\|\boldsymbol{x}^{\mathbf{0}}-\boldsymbol{y}^{\mathbf{0}}\right\| \\
&\quad + \frac{\theta}{\Delta t} M \Delta t \left(1+\left(\tau-\frac{1}{m}\right)C\right) + \left(1+\frac{L}{m}\sum_{k=0}^{n-2}\underbrace{\left(1+\frac{L}{m}\right)^{m\frac{k}{m}}}_{\leqslant\left(1+\frac{L}{m}\right)^{m\tau}}\right) \\
&\leqslant \underbrace{\left(1+\frac{L}{m}\right)^{m\tau}}_{\nearrow e^{L\tau}} \left\|\boldsymbol{x}^{\mathbf{0}}-\boldsymbol{y}^{\mathbf{0}}\right\| \\
&\quad + \theta\, M \underbrace{\left(1+\left(\tau-\frac{1}{m}\right)C\right)}_{\nearrow 1+\tau\, C} \left(1+\underbrace{L\frac{n-1}{m}}_{\nearrow L\,\tau} \underbrace{\left(1+\frac{L}{m}\right)^{m\tau}}_{\nearrow e^{L\tau}}\right),
\end{aligned}$$

which becomes in the asymptotic case

$$\left\|\boldsymbol{x}(t)-\boldsymbol{y}(t)\right\| \leqslant e^{L\tau}\left\|\boldsymbol{x}^{\mathbf{0}}-\boldsymbol{y}^{\mathbf{0}}\right\| + \theta\, M(1+C\tau)\left(1+L\,\tau\, e^{L\tau}\right).$$

This shows that the qualitative behavior is the same for the asymptotic case of the discretized theorem and the continuous Theorem 4.1. Both formulations amplify the linearly entering initial error exponentially over time and the linearly entering control error (in the integral sense) also exponentially with $\tau e^{L\tau}$. The determining factor for the exponential amplification is in both cases the LIPSCHITZ constant L of the right-hand side of the process.

Remark 4.7 (Truely discrete systems)
The theorem gives a theoretical bound on the state approximation in dependence on the control difference also for truly discrete systems. However, if there is no refinement possible, the bound cannot be driven to 0 and an arbitrarily close approximation is not possible in most cases.

4.3 Approximation of decoupled controls in the integral sense

We start with decoupled systems, in which each relaxed control $\alpha_i(\cdot)$ can be approximated independently. Therefore, we can reduce the problem in this section to a one-dimensional one, i.e., $n_\omega = 1$ and the measurable control to be approximated is $\alpha : [t_0, t_f] \to [0,1]$. The norm naturally becomes the absolute value.

Let $\mathbb{G}_m \stackrel{\text{def}}{=} \{t_0 < t_1 < \ldots < t_m = t_f\}$ be the grid on which we want to approximate α with a

piecewise constant function $\omega : [t_0, t_f] \to \{0, 1\}$. We define the time increments

$$\Delta t_i \stackrel{\text{def}}{=} t_{i+1} - t_i, \qquad 0 \leqslant i \leqslant m-1, \tag{4.12}$$

and the maximum time increment

$$\Delta t \stackrel{\text{def}}{=} \max_{0 \leqslant i \leqslant m-1} \Delta t_i. \tag{4.13}$$

We can now parameterize the function $\omega(\cdot)$ with a vector $\boldsymbol{p} \in \{0, 1\}^{m+1}$ as

$$\omega(t) = p_i, \qquad i \in [t_i, t_{i+1}),$$

with the end point $\omega(t_f) = p_m$.

Theorem 4.1 gives a bound on the deviation of the differential states of a system in dependence of the deviation of the controls in the integral sense. Naturally, it makes sense to compute an approximation such that this deviation becomes minimal, i.e.,

$$\min_{\omega(\cdot)} \quad \max_{t \in [t_0, t_f]} \quad \left| \int_{t_0}^{t} \alpha(\tau) - \omega(\tau) \, \mathrm{d}\tau \right|. \tag{4.14}$$

To get rid of the inner maximization problem, we use a well-known trick and add an additional variable θ to obtain the *Control Approximation Problem in integral sense* for given measurable $\alpha(\cdot)$:

$$\begin{aligned} \min_{\theta, \omega(\cdot)} \quad & \theta \\ \text{s.t.} \quad & \theta \geqslant \left| \int_{t_0}^{t} \alpha(\tau) - \omega(\tau) \, \mathrm{d}\tau \right|, \qquad \forall t \in [t_0, t_f], \end{aligned} \tag{4.15}$$

which is now discretized on the time grid $\mathbb{G}_m$ as

$$\begin{aligned} \min_{\theta, \boldsymbol{p} \in \{0,1\}^{m+1}} \quad & \theta \\ \text{s.t.} \quad & \theta \geqslant \left| \int_{t_0}^{t} \alpha(\tau) \, \mathrm{d}\tau - \left(\sum_{k=0}^{i-1} p_k \Delta t_k - p_i (t - t_i) \right) \right|, \qquad \forall t \in [t_i, t_{i+1}), \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 0 \leqslant i \leqslant m-1. \end{aligned} \tag{4.16}$$

The control at the last grid point $t_m = t_f$ does not have any effect on the problem and is hence omitted from here on, i.e., $\boldsymbol{p}$ is treated to be in $\{0, 1\}^m$.

Notice that the minimum or maximum values for the integral

$$\int_{t_0}^{t} \alpha(\tau) - \omega(\tau)\,\mathrm{d}\tau \tag{4.17}$$

on the interval $[t_n, t_{n+1}]$ are taken at one of the grid's points t_n or t_{n+1}. This is the case, since for $t \in [t_n, t_{n+1}]$, either $0 = p_n \leqslant \alpha(t)$ and the integral is monotonically increasing, or $1 = p_n \geqslant \alpha(t)$ and the integral is monotonically decreasing. Therefore, the maximum value of the norm of integral (4.17) is taken at a grid point t_r and we only need to add constraints for the grid points. Problem (4.16) is equivalent to

$$\begin{aligned} \min_{\theta, \boldsymbol{p} \in \{0,1\}^m} \quad & \theta \\ \text{s.t.} \quad & \theta \geqslant \left| \int_{t_0}^{t_i} \alpha(\tau)\,\mathrm{d}\tau - \left(\sum_{k=0}^{i-1} p_k \Delta t_k \right) \right|, \qquad \forall 0 \leqslant i \leqslant m. \end{aligned} \tag{4.18}$$

Since the individual values of the relaxed control only appear in the integral, we can simplify the problem description by using the mean values $\boldsymbol{q} \in \mathbb{R}^m$ over the intervals, i.e.,

$$q_i \stackrel{\text{def}}{=} \frac{1}{\Delta t_i} \int_{t_i}^{t_{i+1}} \alpha(\tau)\,\mathrm{d}\tau, \qquad 0 \leqslant i \leqslant m-1, \tag{4.19}$$

such that it holds

$$\int_{t_0}^{t_{i+1}} \alpha(\tau)\,\mathrm{d}\tau = \sum_{j=0}^{i} q_j \Delta t_j, \qquad 0 \leqslant i \leqslant m-1.$$

With this reformulation and dissolving the norm, we obtain the *Integral Approximation Problem* for control-affine systems with decoupled controls, which is the MILP

$$\begin{aligned} \min_{\theta \in \mathbb{R}, \boldsymbol{p} \in \mathbb{R}^m} \quad & \theta \\ \text{s.t.} \quad & \theta \geqslant +\sum_{j=0}^{i} \left(q_j - p_j \right) \Delta t_j, \qquad 0 \leqslant i \leqslant m-1, \\ & \theta \geqslant -\sum_{j=0}^{i} \left(q_j - p_j \right) \Delta t_j, \qquad 0 \leqslant i \leqslant m-1. \end{aligned} \tag{4.20}$$

Remark 4.8 (Two-dimensional problem with SOS1-constraint)
This setting also covers the case where exactly two modes of the system are present and they are coupled through an SOS1-constraint. One can directly use this constraint to eliminate one control without changing the feasible set of the other control. After that, the same setting

as for decoupled controls is obtained.

4.3.1 Sum-up Rounding control scheme

The Sum-up Rounding (SUR) control scheme from SAGER [152] gives a very good solution of the problem over the described time grid. The following proposition from [158] describes its quality.

Proposition 4.1 (Deviation of Control Integrals by SUR)
Let $\alpha : [t_0, t_f] \to [0,1]$ be a measurable function. Let $\mathbb{G}_m$ be a time grid as described above. Then, it holds that the feasible point $\boldsymbol{p}^{SUR}$ of the Control Approximation Problem in integral sense (4.18) *provided by the iterative SUR control scheme has a solution quality of*

$$\theta \leqslant \frac{1}{2}\Delta t. \tag{4.21}$$

It is defined for $0 \leqslant i \leqslant m-1$ as

$$p_i^{SUR} \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } \displaystyle\int_{t_0}^{t_{i+1}} \alpha(\tau)\,\mathrm{d}\tau - \sum_{k=0}^{i-1} p_k^{SUR}\Delta t_k \geqslant \frac{1}{2}\Delta t_i, \\ 0, & \text{else.} \end{cases} \tag{4.22}$$

△

Proof This is shown by complete induction for the grid points. The claim is trivial for the first grid point t_0 since no control decision had to be made until then.
Let the assumption (4.17) hold for all grid points t_i with $0 \leqslant i \leqslant r$. Let us now distinguish two different cases for the value of the next combined integral

$$\int_{t_0}^{t_{r+1}} \alpha(\tau)\,\mathrm{d}\tau - \int_{t_0}^{t_r} \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau, \tag{4.23}$$

which lead to the different controls of the scheme.
Assume the integral (4.23) $< \frac{1}{2}\Delta t_r$, then the SUR control scheme provides $p_r^{\mathrm{SUR}} = 0$ and we obtain

$$\int_{t_0}^{t_{r+1}} \alpha(\tau) - \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau = \int_{t_0}^{t_{r+1}} \alpha(\tau)\,\mathrm{d}\tau - \int_{t_0}^{t_r} \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau < \frac{1}{2}\Delta t_r \leqslant \frac{1}{2}\Delta t,$$

and due to the assumption, it holds

$$\int_{t_0}^{t_{r+1}} \alpha(\tau) - \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau = \underbrace{\int_{t_0}^{t_r} \alpha(\tau) - \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau}_{\geqslant -\frac{1}{2}\Delta t} + \underbrace{\int_{t_r}^{t_{r+1}} \alpha(\tau)\,\mathrm{d}\tau}_{\geqslant 0} \geqslant -\frac{1}{2}\Delta t.$$

Assume the integral (4.23) $\geqslant \frac{1}{2}\Delta t_r$, then the control scheme provides $p_r^{\mathrm{SUR}} = 1$ and we obtain

$$\begin{aligned}\int_{t_0}^{t_{r+1}} \alpha(\tau) - \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau &= \int_{t_0}^{t_r} \alpha(\tau) - \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau + \int_{t_r}^{t_{r+1}} \alpha(\tau) - 1\,\mathrm{d}\tau \\ &= \underbrace{\int_{t_0}^{t_{r+1}} \alpha(\tau)\,\mathrm{d}\tau - \int_{t_0}^{t_r} \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau}_{\geqslant \frac{1}{2}\Delta t_r} \underbrace{- \int_{t_r}^{t_{r+1}} 1\,\mathrm{d}\tau}_{=\Delta t_r} \\ &\geqslant -\frac{1}{2}\Delta t_r,\end{aligned}$$

and due to the assumption, it holds

$$\begin{aligned}\int_{t_0}^{t_{r+1}} \alpha(\tau) - \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau &= \underbrace{\int_{t_0}^{t_r} \alpha(\tau) - \omega^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau}_{\leqslant \frac{1}{2}\Delta t} + \underbrace{\int_{t_r}^{t_{r+1}} \alpha(\tau) - 1\,\mathrm{d}\tau}_{\leqslant 0} \\ &\leqslant \frac{1}{2}\Delta t.\end{aligned}$$

This completes the proof for all grid points. □

Proposition 4.2 (Best possible bound)
The best bound that any algorithm for arbitrary measurable $\alpha(\cdot)$ and arbitrary time grid

$$\mathbb{G}_m \stackrel{\mathrm{def}}{=} \{t_0 < t_1 < \ldots < t_m = t_f\}$$

can give is $\frac{1}{2}\Delta t$, as given by the SUR strategy. △

Proof For any time interval $[t_i, t_{i+1}]$ with $\Delta t_i = \Delta t$, consider any relaxed control $\alpha(\cdot)$ and let $\boldsymbol{\omega}(\cdot)$ be its integer approximation until time point t_i, which may use the known data for $t > t_i$. However, for the algorithm to approximate better than the SUR algorithm, it must hold

$$\delta \stackrel{\mathrm{def}}{=} \int_{t_0}^{t_i} \alpha(\tau) - \omega(\tau)\,\mathrm{d}\tau \in \left(-\frac{1}{2}\Delta t_i, \frac{1}{2}\Delta t_i\right),$$

since the next interval could be the biggest.
Depending on this situation, let the control $\alpha(\cdot)$ to be approximated have the following constant value on the next grid interval:

$$\alpha(t) = \frac{1}{2} - \frac{\delta}{\Delta t_i}, \qquad \forall t \in [t_i, t_{i+1}].$$

The two possible approximation choices on the next interval are either 0 or 1:

$$\begin{aligned}
\int_{t_0}^{t_{i+1}} \alpha(\tau) - \omega(\tau)\,\mathrm{d}\tau &= \delta + \int_{t_i}^{t_{i+1}} \left(\frac{1}{2} - \frac{\delta}{\Delta t_i}\right) - 0\,\mathrm{d}\tau \\
&= \delta + \left(\frac{1}{2} - \frac{\delta}{\Delta t_i}\right)\Delta t_i \\
&= \frac{1}{2}\Delta t_i, \\
\int_{t_0}^{t_{i+1}} \alpha(\tau) - \omega(\tau)\,\mathrm{d}\tau &= \delta + \int_{t_i}^{t_{i+1}} \left(\frac{1}{2} - \frac{\delta}{\Delta t_i}\right) - 1\,\mathrm{d}\tau \\
&= \delta + \left(\frac{1}{2} - \frac{\delta}{\Delta t_i} - 1\right)\Delta t_i \\
&= -\frac{1}{2}\Delta t_i.
\end{aligned}$$

For both choices, the best reachable approximation is $\theta = \frac{1}{2}\Delta t_i$. □

Remark 4.9 (Suboptimal solution)
The SUR scheme does not necessarily give the optimal solution. This is illustrated in the following example, see Figure 4.2. Consider the problem given by the time grid $\mathbb{G}_m$ and relaxed controls

$$\begin{aligned}
\mathbb{G}_3 &= \left(0, \frac{3}{4}, \frac{5}{4}, \frac{11}{4}\right)^T, \\
\boldsymbol{q} &= \left(\frac{1}{3}, 0, \frac{2}{3}\right)^T.
\end{aligned}$$

Then, the SUR solution is

$$\boldsymbol{p}^{\mathrm{SUR}} = (0,1,1)^T,$$

with an objective value of $\theta^{\mathrm{SUR}} = \frac{3}{4} \leqslant \frac{1}{2}\Delta t = \frac{3}{4}$ – whereas the optimal solution is

$$\boldsymbol{p}^{\mathrm{OPT}} = (0,0,1)^T,$$

with an objective value of $\theta^{\mathrm{OPT}} = \frac{1}{4}$.

However, for equidistant grids it solves the problem to optimality.

Proposition 4.3 (Optimal solution for equidistant grids)
The SUR scheme solves problem (4.20) *to optimality for equidistant grids, i.e.,*

$$t_i \overset{\mathrm{def}}{=} t_0 + i\Delta t.$$ △

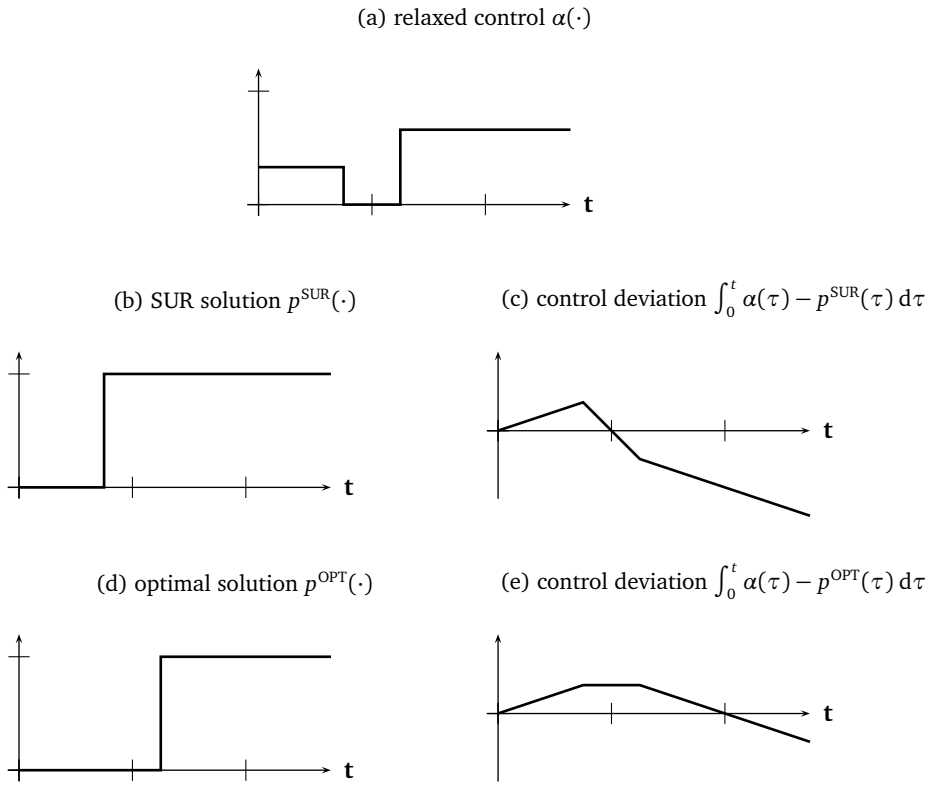


Figure 4.2: Comparison of the SUR solution (b) and the optimal solution (d) for given relaxed controls (a). The maximum integral deviations are $\frac{3}{4}$ and $\frac{1}{4}$, respectively. The issue is that the SUR scheme locally tries to bring the deviation below Δt_i and thereby prevents the whole deviation from reaching Δt. However, locally exceeding this limit of Δt_i might be optimal if Δt_i is small enough.

Proof Consider another solution $\bar{\boldsymbol{p}} \neq \boldsymbol{p}^{\mathrm{SUR}}$. Let k be the first time step where they differ, i.e., $\bar{p}_k \neq p_k^{\mathrm{SUR}}$ and $\bar{p}_i = p_i^{\mathrm{SUR}}$ for $i < k$. Due to the SUR properties, it holds

$$\left| \int_{t_0}^{t_{k-1}} \alpha(\tau) - p^{\mathrm{SUR}}(\tau)\,\mathrm{d}\tau \right| = \left| \int_{t_0}^{t_{k-1}} \alpha(\tau) - \bar{p}(\tau)\,\mathrm{d}\tau \right| \leqslant \frac{1}{2}\Delta t.$$

Consider the next time step and let the control determining value be $> \frac{1}{2}\Delta t$, i.e.,

$$\int_{t_0}^{t_k} \alpha(\tau)\,\mathrm{d}\tau - \sum_{i=0}^{k-1} p_i^{\mathrm{SUR}} \Delta t > \frac{1}{2}\Delta t, \tag{4.24}$$

and hence $1 = p_k^{\mathrm{SUR}} \neq \bar{p}_k = 0$. Then, it holds

$$\int_{t_0}^{t_k} \alpha(\tau)\,\mathrm{d}\tau - \sum_{i=0}^{k} \bar{p}_i \Delta t = \int_{t_0}^{t_k} \alpha(\tau)\,\mathrm{d}\tau - \sum_{i=0}^{k-1} p_i^{\mathrm{SUR}}(\tau)\Delta t - 0\Delta t > \frac{1}{2}\Delta t,$$

which leads to $\theta > \frac{1}{2}\Delta t$. Therefore, this solution is worse than the guaranteed worst case for the SUR solution. It is easy to see that also for the other possible control determining values on the left-hand side of equation (4.24), i.e., $= \frac{1}{2}\Delta t$ and $<= \frac{1}{2}\Delta t$, we get the result that the solution can either only be of the same value as the worst case SUR solution or is even worse. □

This scheme gives an algorithm that can approximate a measurable control trajectory $\alpha(\cdot)$ arbitrarily close in a weak sense as long as the grid is chosen fine enough. In combination with Theorem 4.1, this allows for arbitrarily close approximations of the differential states. Together, this allows us to come arbitrarily close to a solution of the control-affine purely path constrained IVP (4.7) through first solving the relaxed problem and then approximating its solution.

4.3.2 Analysis of the LAGRANGIAN

The contents of this section are based on the paper

[104] M. JUNG, G. REINELT AND S. SAGER, *The Lagrangian Relaxation for the Combinatorial Integral Approximation Problem*, Optimization Methods and Software (Submitted).

In this section, we analyze the behavior of the LAGRANGIAN relaxation of the *Control Approximation Problem in integral sense* in a branching framework. First, we give a brief introduction to the general LAGRANGIAN in linear systems, then we analyze the LAGRANGIAN in this special setting.

The LAGRANGIAN relaxation is a popular relaxation for Mixed-Integer Linear Programs (MILPs), e.g. [73, 118]. However, the approach is also generally used in all kinds of constrained optimization problems. The principle is to relax a problem by dropping constraints and penalizing their violation in the objective function. We brief introduce the LAGRANGIAN *relaxation*, the LAGRANGIAN *function* and the LAGRANGIAN *problem* for an Integer Program (IP):

Definition 4.1 (LAGRANGIAN relaxation)
Consider the IP

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}, \\ & \boldsymbol{D}\boldsymbol{x} \leqslant \boldsymbol{d}, \\ & \boldsymbol{x} \text{ integer.} \end{aligned} \tag{4.25}$$

A LAGRANGIAN relaxation *of* (4.25) *is given by the IP*

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \\ \text{s.t.} \quad & \boldsymbol{D}\boldsymbol{x} \leqslant \boldsymbol{d}, \\ & \boldsymbol{x} \text{ integer} \end{aligned} \tag{4.26}$$

for given non-negative LAGRANGIAN *multipliers* $\boldsymbol{\lambda} \in \mathbb{R}_0^+$. △

This can also be done for equality constraints, then the multipliers can also be negative.

Definition 4.2 (LAGRANGIAN function)
The LAGRANGIAN function *corresponding to the* LAGRANGIAN *relaxation of* (4.26) *is defined as*

$$z_{\mathrm{LR}}(\boldsymbol{\lambda}) \stackrel{\text{def}}{=} \min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \;\middle|\; \boldsymbol{D}\boldsymbol{x} \leqslant \boldsymbol{d}, \boldsymbol{x} \text{ integer} \right\}. \qquad \triangle$$

Definition 4.3 (LAGRANGIAN problem)
The LAGRANGIAN problem *of* (4.26) *is defined as the problem of finding the most constricting choice of* LAGRANGIAN *multipliers, i.e., to maximize the* LAGRANGIAN *function with respect to the feasible multipliers:*

$$Z_{\mathrm{LR}} \stackrel{\text{def}}{=} \max_{\boldsymbol{\lambda}} \left\{ z_{\mathrm{LR}}(\boldsymbol{\lambda}) \;\middle|\; \boldsymbol{\lambda} \geqslant \boldsymbol{0} \right\}. \qquad \triangle$$

Proposition 4.4 (LAGRANGIAN relaxation vs. LP relaxation)
Let there exist a feasible point of the canonical LP-relaxation. Let Z_{LR} *be the value of the* LAGRANGIAN *problem's solution and let* Z_{LP} *the value of the canonical LP-relaxation of* (4.25) *that drops the integrality constraint. Then, it holds*

$$Z_{\mathrm{LR}} \geqslant Z_{\mathrm{LP}}. \qquad \triangle$$

The proposition states that the LAGRANGIAN relaxation is tighter than the LP-relaxation if the best multipliers are chosen. This well known property of the LAGRANGIAN relaxation is proven according to [73]:

Proof Let $Z_{\text{LP-dual}}$ the objective value of the dual LP of the canonical LP-relaxation with respect to $\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$:

$$z_{\text{LP-dual}}(\boldsymbol{\lambda}) \stackrel{\text{def}}{=} \min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \;\middle|\; \boldsymbol{D}\boldsymbol{x} \leqslant \boldsymbol{d} \right\}, \tag{4.27}$$

$$Z_{\text{LP-dual}} = \max_{\boldsymbol{\lambda}} \left\{ z_{\text{LP-dual}}(\boldsymbol{\lambda}) \;\middle|\; \boldsymbol{\lambda} \geqslant \boldsymbol{0} \right\}. \tag{4.28}$$

Due to the *duality theorem for linear programming* for a subset of constraints, cf. [71, Section 6.1], and assuming that a feasible point exists, the optimal value of the canonical LP-

relaxation is equal to its dual (4.28):

$$Z_{\text{LP-dual}} = Z_{\text{LP}}. \tag{4.29}$$

There exists a multiplier $\bar{\boldsymbol{\lambda}}$ realizing the outer maximization of the dual LP (4.28) with:

$$z_{\text{LP-dual}}(\bar{\boldsymbol{\lambda}}) = Z_{\text{LP-dual}}. \tag{4.30}$$

The feasible set of the inner minimization in $z_{\text{LR}}(\boldsymbol{\lambda})$ is a subset of the feasible set of $z_{\text{LP-dual}}(\boldsymbol{\lambda})$, hence the inner minimization cannot be solved better:

$$z_{\text{LR}}(\bar{\boldsymbol{\lambda}}) \geqslant z_{\text{LP-dual}}(\bar{\boldsymbol{\lambda}}). \tag{4.31}$$

The dual variable $\bar{\boldsymbol{\lambda}}$ does not necessarily provide the best solution of the LAGRANGIAN relaxation, but could provide a suboptimal point:

$$Z_{\text{LR}} \geqslant z_{\text{LR}}(\bar{\boldsymbol{\lambda}}). \tag{4.32}$$

The chain of inequalities (4.29)–(4.32) proofs the proposition. □

Formulation of the LAGRANGIAN in the Control Approximation Problem

For the discretized *Control Approximation Problem in integral sense* (4.20), we choose to penalize all constraints in (4.20) to obtain the following LAGRANGIAN function

$$z_{\text{LR}}(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \min_{\theta \in \mathbb{R}, \boldsymbol{p} \in \mathbb{R}^m} L(\boldsymbol{\lambda}, \boldsymbol{\mu}, \theta, \boldsymbol{p})$$

with

$$\begin{aligned} L(\boldsymbol{\lambda}, \boldsymbol{\mu}, \theta, \boldsymbol{p}) &\stackrel{\text{def}}{=} \theta + \sum_{i=0}^{m-1} \lambda_i \left(-\theta + \sum_{j=0}^{i} (p_j - q_j) \Delta t_j \right) \\ &\quad + \sum_{i=0}^{m-1} \mu_i \left(-\theta - \sum_{j=0}^{i} (p_j - q_j) \Delta t_j \right) \\ &= \theta \left(1 - \sum_{i=0}^{m-1} \lambda_i + \mu_i \right) + \sum_{i=0}^{m-1} (\lambda_i - \mu_i) \left(\sum_{j=0}^{i} (p_j - q_j) \Delta t_j \right). \end{aligned}$$

However, if $\sum_{i=0}^{m-1} (\lambda_i - \mu_i) \neq 1$, the inner function $L(\cdot)$ of the LAGRANGIAN function is unbounded due to the free variable θ. Therefore, any solution of the LAGRANGIAN problem satisfies this condition – and thereby θ can be eliminated from the problem to get the modified

LAGRANGIAN function

$$\bar{z}_{\mathrm{LR}}(\boldsymbol{\lambda},\boldsymbol{\mu}) = \min_{\boldsymbol{p}\in\mathbb{R}^m} \sum_{i=0}^{m-1}(\lambda_i-\mu_i)\left(\sum_{j=0}^{i}(p_j-q_j)\Delta t_j\right),$$

and the corresponding LAGRANGIAN problem

$$\begin{aligned} \max_{\boldsymbol{\lambda},\boldsymbol{\mu}\in\mathbb{R}^m_+} \quad & \min_{\boldsymbol{p}\in\{0,1\}^m} \quad \sum_{i=0}^{m-1}(\lambda_i-\mu_i)\left(\sum_{j=0}^{i}(p_j-q_j)\Delta t_j\right) \\ \text{s.t.} \quad & \sum_{i=0}^{m-1}(\lambda_i-\mu_i)=1. \end{aligned} \tag{4.33}$$

Proposition 4.5 (Unconstrained problem)
The value of the LAGRANGIAN *problem* (4.33) *is*

$$Z_{\mathrm{LR}} = 0. \qquad \triangle$$

Proof Reordering the terms in the LAGRANGIAN function results in

$$\bar{z}_{\mathrm{LR}}(\boldsymbol{\lambda},\boldsymbol{\mu}) = \min_{p\in\{0,1\}^m} \sum_{i=0}^{m-1}(p_i-q_i)\Delta t_i\left(\sum_{j=i}^{m-1}\lambda_j-\mu_j\right).$$

If $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are fixed to any feasible values, then, the optimal values for $\boldsymbol{p}$ directly emerge, since there is no coupling between the different points in time:

$$p_i^* = \begin{cases} 0, & \text{if } \sum_{j=i}^{m-1}\lambda_j-\mu_j \geqslant 0, \\ 1, & \text{else.} \end{cases} \tag{4.34}$$

Partitioning the index set $\mathcal{I}=\{0,\dots,m-1\}$ into two subsets according to those two cases, i.e.,

$$\mathcal{G}(\boldsymbol{\lambda},\boldsymbol{\mu}) = \left\{ i\in\mathcal{I} \;\middle|\; \sum_{j=i}^{m-1}\lambda_j-\mu_j \geqslant 0 \right\},$$

and its complement $\mathcal{I}\backslash\mathcal{G}$, and using the optimal solution $\boldsymbol{p}^*$, the LAGRANGIAN function can be

rewritten as

$$\bar{z}_{\mathrm{LR}}(\boldsymbol{\lambda},\boldsymbol{\mu}) = \sum_{i\in\mathcal{I}\setminus\mathcal{G}} \underbrace{(1-q_i)\Delta t_i \left(\sum_{j=i}^{m-1}\lambda_j - \mu_j\right)}_{\leqslant 0} - \sum_{i\in\mathcal{G}} \underbrace{q_i\Delta t_i \left(\sum_{j=i}^{m-1}\lambda_j - \mu_j\right)}_{\geqslant 0} \leqslant 0.$$

The maximum value $Z_{\mathrm{LR}} = 0$ is taken, e.g. with the following feasible multipliers

$$\bar{\lambda}_0 = \bar{\mu}_0 = 0.5 \text{ and } \bar{\lambda}_i = \bar{\mu}_i = 0 \text{ for } i \neq 0. \qquad \square$$

Behavior of the LAGRANGIAN relaxation during branching

We could use the LAGRANGIAN problem's solution in a branch-and-bound algorithm for the bounding procedure. As already seen in Proposition 4.5, the bound of the root node, i.e., the unconstrained case, would be 0. However, we can use this bound after some branching. Here, we consider the standard branching procedure for binary variables, i.e., for each node we choose one variable and construct one child where this variable is set to 0 and another one where it is set to 1.
Consider a deeper node in the branching tree. Some controls p_i are fixed to the values p_i^{fixed}, $i \in \mathcal{F} \subseteq \mathcal{I} \stackrel{\mathrm{def}}{=} \{0,\dots,m-1\}$ due to branching.
To identify the optimal solution of the LAGRANGIAN relaxation of this node, some abbreviating notations are defined:

Definition 4.4
For two indices $0 \leqslant i_1 \leqslant i_2 \leqslant m-1$, we define the following terms $v_{i_1,i_2}, \bar{v}_{i_1,i_2}$ as

$$v_{i_1,i_2} \stackrel{\mathrm{def}}{=} \sum_{i\in[i_1,i_2]\cap\mathcal{F}} (p_i^{\mathrm{fixed}} - q_i)\Delta t_i - \sum_{i\in[i_1,i_2]\setminus\mathcal{F}} q_i\Delta t_i, \tag{4.35}$$

$$\bar{v}_{i_1,i_2} \stackrel{\mathrm{def}}{=} \sum_{i\in[i_1,i_2]\cap\mathcal{F}} (p_i^{\mathrm{fixed}} - q_i)\Delta t_i + \sum_{i\in[i_1,i_2]\setminus\mathcal{F}} (1-q_i)\Delta t_i. \tag{4.36}$$

These terms give the value of the control approximation during a time interval $[t_{i_1}, t_{i_2}]$ where all free controls are either 0 (in the case of v) or 1 (in the case of $\bar{v}$).
We also define some indices that are needed to properly state the results:

- *$i^* \in \mathcal{I}$ such that v_{0,i^*} is maximal,*
- *$i_1^*, i_2^* \in I$ such that $v_{i_1^*,i_2^*}$ is maximal,*
- *$j^* \in I$ such that $\bar{v}_{0,j^*}$ is minimal,*
- *$j_1^*, j_2^* \in I$ such that $\bar{v}_{j_1^*,j_2^*}$ is minimal.* $\triangle$

With these terms, we can directly give the value of the LAGRANGIAN relaxation.

Theorem 4.3 (LAGRANGIAN relaxation after branching)
Assume we are in the described setting for the LAGRANGIAN *relaxation of the Control Approximation Problem with fixed variables p_i for $i \in \mathcal{F}$.*

Then, the value of the LAGRANGIAN *relaxation is*

$$Z_{\text{LR}} = \max\left\{ v_{0,i^*},\ -\bar{v}_{0,j^*},\ \frac{1}{2} v_{i_1^*,i_2^*},\ -\frac{1}{2}\bar{v}_{j_1^*,j_2^*} \right\}.$$

In the optimal solution, depending on which of the terms is the maximum, either $\lambda_{i^*} = 1$ or $\mu_{j^*} = 1$ or $\mu_{i_1^*} = \lambda_{i_2^*} = \frac{1}{2}$ or $\lambda_{j_1^*} = \mu_{j_2^*} = \frac{1}{2}$. △

Proof The outline of the proof is as follows. We interpret the problem as being similar to a knapsack problem and we introduce variables to enhance readability of the procedure. Next, we explain the concept of phases that helps to understand the objective function of interest. As a result, four dominant phases naturally emerge as the phases that maximize this special objective in different circumstances.

For the LAGRANGIAN relaxation we have the following min-max problem:

$$\begin{aligned} \max_{\boldsymbol{\lambda},\boldsymbol{\mu}\in\mathbb{R}_+^m} \quad & \min_{\boldsymbol{p}\in\{0,1\}^m} \quad \sum_{i=0}^{m-1}(\lambda_i - \mu_i)\left(\sum_{j=0}^{i}(p_j - q_j)\Delta t_j\right) \\ & \text{s.t.} \quad p_i \text{ fixed to } p_i^{\text{fixed}} \text{ for } i \in \mathcal{F}, \\ \text{s.t.} \quad & \sum_{i=0}^{m-1}(\lambda_i - \mu_i) = 1. \end{aligned} \tag{4.36}$$

We reformulate the LAGRANGIAN relaxation using the solution of the inner minimization problem. This solution remains of the same structure as observed in the unrestricted case (4.34). The only difference is that we have to distinguish three cases for each variable p_j instead of only two. The additional case to be considered emerges when the variable p_j is fixed and not free. With the set

$$\mathcal{G}(\boldsymbol{\lambda},\boldsymbol{\mu}) = \left\{ i \in \mathcal{I} \;\middle|\; \sum_{j=i}^{m-1} \lambda_j - \mu_j \geqslant 0 \right\}$$

as above, the problem takes the form

$$\max_{\boldsymbol{\lambda},\boldsymbol{\mu}\in\mathbb{R}_+^m} \quad \sum_{i=0}^{m-1}\left(\sum_{j=i}^{m-1}\lambda_i-\mu_i\right)\Delta t_i \begin{cases}(p_i^{\text{fixed}}-q_i), & \text{if } i\in\mathcal{F},\\ (-q_i), & \text{if } i\in\mathcal{G}(\boldsymbol{\lambda},\boldsymbol{\mu}),\\ (1-q_i), & \text{if } i\notin\mathcal{F}\cup\mathcal{G}(\boldsymbol{\lambda},\boldsymbol{\mu}),\end{cases} \tag{4.37}$$
$$\text{s.t.} \quad \sum_{i=0}^{m-1}(\lambda_i-\mu_i)=1.$$

Here, we can see the clear structure of a budget of 1 for $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ as the constraint and a special objective function that already incorporates the solution of the inner minimization problem.

Definition 4.5
We introduce variables $\boldsymbol{a}$ and $\boldsymbol{b}$ to enhance readability:

$$a_i \overset{\text{def}}{=} \sum_{j=i}^{m-1}\lambda_j, \quad a_m \overset{\text{def}}{=} 0,$$
$$b_i \overset{\text{def}}{=} \sum_{j=i}^{m-1}\mu_j, \quad b_m \overset{\text{def}}{=} 0. \qquad \triangle$$

The old variables $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ mean a change in the new variables $\boldsymbol{a}$ and $\boldsymbol{b}$. The restrictions that were posed on the multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are easily translated into

$$0\leqslant\lambda_i=a_i-a_{i+1}, \tag{4.38a}$$
$$0\leqslant\mu_i=b_i-b_{i+1}, \tag{4.38b}$$
$$1=\sum_{i=0}^{m-1}\lambda_i+\mu_i=a_0+b_0. \tag{4.38c}$$

These restrictions can be described as a restriction on the initial sum of the factors $\boldsymbol{a}$ and $\boldsymbol{b}$ and both are non-increasing. As a direct result, the total change in $\boldsymbol{a}$ and $\boldsymbol{b}$ is limited by 1. With these variables, the objective of the LAGRANGIAN function becomes

$$\tilde{z}_{\text{LR}}(\boldsymbol{a},\boldsymbol{b})=\sum_{i=0}^{m-1}(a_i-b_i)\Delta t_i \begin{cases}(p_i^{\text{fixed}}-q_i), & \text{if } i\in\mathcal{F},\\ (-q_i), & \text{if } i\notin\mathcal{F} \text{ and } a_i\geqslant b_i,\\ (1-q_i), & \text{if } i\notin\mathcal{F} \text{ and } a_i<b_i.\end{cases} \tag{4.39}$$

Taking a closer look at these terms, the only possible positive summands are those with $i\in\mathcal{F}$ and $\operatorname{sgn}(a_i-b_i)=\operatorname{sgn}(p_i^{\text{fixed}}-q_i)$. The LAGRANGIAN relaxation maximizes this function under

the above stated restrictions on $\boldsymbol{a}$ and $\boldsymbol{b}$. The aim is to optimally use these changes from the initial factors whose sum is 1 to avoid negative parts and to optimally use positive summands in the objective.

Since only the change in $\boldsymbol{a}$ and $\boldsymbol{b}$ is limited, we introduce a phase $\pi = (i_\pi, j_\pi, \delta_\pi)$ as an interval in time at whose initial border, i_π, $\boldsymbol{a} - \boldsymbol{b}$ increases by $\delta \in [-1, 1]$ and at whose ending border, j_π, this change is reversed by adding $-\delta$. The value regarding the objective contribution of a phase $\pi = (i_\pi, j_\pi, \delta_\pi)$ is

$$v((i_\pi, j_\pi, \delta_\pi), \boldsymbol{a}, \boldsymbol{b}) = \delta_\pi \sum_{k=i_\pi}^{j_\pi} \Delta t_k \begin{cases} (p_k^{\text{fixed}} - q_k), & \text{if } k \in \mathcal{F}, \\ (-q_k), & \text{if } k \notin \mathcal{F} \text{ and } a_k \geqslant b_k, \\ (1 - q_k), & \text{if } k \notin \mathcal{F} \text{ and } a_k < b_k. \end{cases}$$

Each solution $(\boldsymbol{a}, \boldsymbol{b})$ to the problem is composed of an overlapping of such phases. For starting phases, i.e., phases that begin at time step 0, the cost is only contributed by the change at the end, because for the beginning factors $a_0 + b_0 = 1$ must hold, i.e., all of the budget is given there. Therefore, the cost $c(\pi, \boldsymbol{a}, \boldsymbol{b})$ of the phase π in budget terms is

$$c((0, j_\pi, \delta_\pi), \boldsymbol{a}, \boldsymbol{b}) = \left|\delta_\pi\right|, \qquad 0 \leqslant j_\pi \leqslant m - 1.$$

For intermediate phases, the costs of the phase are contributed at both its start and its end and we get

$$c((i_\pi, j_\pi, \delta_\pi), \boldsymbol{a}, \boldsymbol{b}) = 2\left|\delta_\pi\right|, \qquad 1 \leqslant i_\pi \leqslant j_\pi \leqslant m - 1.$$

We can now consider any solution $(\boldsymbol{a}, \boldsymbol{b})$ as an overlapping of phases. In the following, for each solution we choose a unique decomposition into a set of phases $P(\boldsymbol{a}, \boldsymbol{b})$, which allows a decomposition of the costs into the costs of the single phases and a decomposition of the objective value into the values of the phases. The idea is to decompose the graph into maximally sized vertical slices as done on the left-hand side of Figure 4.3. The following three properties have to be considered to get this set of phases $P(\boldsymbol{a}, \boldsymbol{b})$:

1. The most important part is that no phases with different signs of δ may overlap. This implies that by knowing the sign of a phase, we directly also know the sign of $\boldsymbol{a} - \boldsymbol{b}$ during that phase, it is the same. We can hence determine the value of $v(\pi, \boldsymbol{a}, \boldsymbol{b})$ of a phase π without knowing $\boldsymbol{a}$ and $\boldsymbol{b}$ and always the same choice is made in the inner brackets. This also forbids "invisible" additional phases that cancel each other's effects, i.e., all phases can be visualized in the graph as done in Figure 4.3.
2. The second choice is such that we can directly get the budget consumption as the sum over phase heights. For this, a phase has to be as wide as possible regarding time without changing its sign (e.g. δ_1 on the left-hand side of Figure 4.3 is correct, whereas δ_4

should be chosen wider on the right-hand side). This ensures that the true change in the multiplier structure is captured instead of additional artificial changes as done on the right-hand side of Figure 4.3 between phases 4 and 5.

3. To get a unique set of phases we have to add the constraint that each phase is chosen as high as possible regarding $|\delta|$, else each horizontal slice could be decomposed into any number of slices, which are as wide as the original but combine to the same height.

In Figure 4.3 these choices are made correctly on the left-hand side, whereas there are some mistakes on the right-hand side.

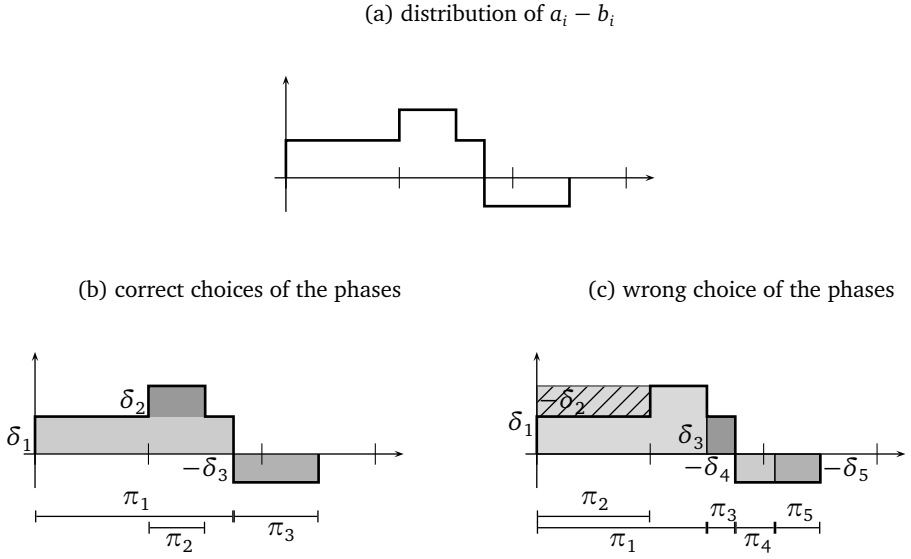


Figure 4.3: On the left-hand side (b) every phase is chosen according to the rules above. On the right-hand side (c), e.g. the phase π_1 cannot overlap with the phase π_2 since the corresponding δ_1 and δ_2 have different signs (property (1)), π_1 has to be made wider such that it includes the π_3-piece, since width is chosen first (property (2)), and it has to be made lower (property (3)).

A set of multipliers thus produces a unique set of phases $\mathcal{P}(\boldsymbol{a},\boldsymbol{b})$. Let each phase π be specified through $\pi = (i_\pi, j_\pi, \delta_\pi)$ as above. Let $\mathcal{P}_0(\boldsymbol{a},\boldsymbol{b})$ be the set of phases starting at time step 0 and let $\mathcal{P}_+(\boldsymbol{a},\boldsymbol{b})$ be the set of phases starting later.

The total changes of the multipliers can be decomposed into the old changes and the changes that happen from step i to step $i+1$:

$$a_{i+1} - b_{i+1} = a_i - b_i + \underbrace{\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ i+1=i_\pi}} \delta_\pi}_{\text{phases starting at } i+1} - \underbrace{\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ i=j_\pi}} \delta_\pi}_{\text{phases ending at } i}, \quad 1 \leqslant i \leqslant m-1. \tag{4.40}$$

Due to the properties 1 and 2 of the phases in $\mathcal{P}(\boldsymbol{a},\boldsymbol{b})$, all the δ-terms on the right-hand side

of this equation (4.40) have the same sign, i.e., the starting phases have a different sign of δ than the ending phases. Therefore, we can move the norm into all individual terms:

$$\Big|\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ i_\pi=i+1}}\delta_\pi-\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ j_\pi=i}}\delta_\pi\Big|=\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ i_\pi=i+1}}\big|\delta_\pi\big|+\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ j_\pi=i}}\big|\delta_\pi\big|$$

and obtain the following property for all changes from time step i to $i+1$

$$\begin{aligned}\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ i_\pi=i+1}}\big|\delta_\pi\big|+\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ j_\pi=i}}\big|\delta_\pi\big|&=\Big|\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ i_\pi=i+1}}\delta_\pi-\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ j_\pi=i}}\delta_\pi\Big|\\ &\underset{(4.40)}{=}\big|(a_{i+1}-b_{i+1})-(a_i-b_i)\big|\\ &\leqslant\big|a_{i+1}-a_i\big|+\big|b_{i+1}-b_i\big|\\ &\underset{(4.38a),(4.38b)}{=}a_i-a_{i+1}+b_i-b_{i+1}.\end{aligned}$$

Summing these up for $0\leqslant i\leqslant m-1$, we get only the end contribution for the starting phases and both contributions for the phases starting later and a telescope sum on the right-hand side:

$$\begin{aligned}&\sum_{i=0}^{m-1}\Big(\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ b_\pi=i+1}}\big|\delta_\pi\big|+\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ e_\pi=i}}\big|\delta_\pi\big|\Big)\leqslant a_0+b_0-a_m-b_m\\ \Leftrightarrow\quad&\sum_{\pi\in\mathcal{P}_0(\boldsymbol{a},\boldsymbol{b})}\big|\delta_\pi\big|+\sum_{\pi\in\mathcal{P}_+(\boldsymbol{a},\boldsymbol{b})}2\big|\delta_\pi\big|\leqslant a_0+b_0=1\end{aligned}\tag{4.41}$$

Due to the properties of $\mathcal{P}(\boldsymbol{a},\boldsymbol{b})$ and with the definitions of ν and $\bar{\nu}$, see (4.35) and (4.36), the objective value of a solution $(\boldsymbol{a},\boldsymbol{b})$ can directly be split into the values of the single phases, i.e.,

$$\tilde{z}_{\mathrm{LR}}(\boldsymbol{a},\boldsymbol{b})=\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi>0}}\delta_\pi\nu_{i_\pi,j_\pi}+\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi<0}}\delta_\pi\bar{\nu}_{i_\pi,j_\pi}.$$

With the indices i^*, j^*, etc. defined as above, we can now show the claim:

$$\begin{aligned}\tilde{z}_{\mathrm{LR}}(\boldsymbol{a},\boldsymbol{b})&=\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi>0}}\delta_\pi\nu_{i_\pi,j_\pi}+\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi<0}}\delta_\pi\bar{\nu}_{i_\pi,j_\pi}\\ &=\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi>0}}\big|\delta_\pi\big|\,\nu_{i_\pi,j_\pi}-\sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi<0}}\big|\delta_\pi\big|\,\bar{\nu}_{i_\pi,j_\pi}\end{aligned}$$

$$
\begin{aligned}
&\underset{\substack{\text{def } i^*,\\ \text{etc.}}}{\leqslant} \sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi>0, i_\pi=0}} \left|\delta_\pi\right| \nu_{0,i^*} - \sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi<0, i_\pi=0}} \left|\delta_\pi\right| \bar{\nu}_{0,j^*} + \sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi>0, i_\pi\neq 0}} \left|\delta_\pi\right| \nu_{i_1^*,i_2^*} - \sum_{\substack{\pi\in\mathcal{P}(\boldsymbol{a},\boldsymbol{b})\\ \delta_\pi<0, i_\pi\neq 0}} \left|\delta_\pi\right| \bar{\nu}_{j_1^*,j_2^*} \\
&\leqslant \sum_{\pi\in\mathcal{P}_0(\boldsymbol{a},\boldsymbol{b})} \left|\delta_\pi\right| \max\left\{\nu_{0,i^*}, -\bar{\nu}_{0,j^*}\right\} + \sum_{\pi\in\mathcal{P}_+(\boldsymbol{a},\boldsymbol{b})} 2\left|\delta_\pi\right| \max\left\{\frac{1}{2}\nu_{i_1^*,i_2^*}, -\frac{1}{2}\bar{\nu}_{j_1^*,j_2^*}\right\} \\
&\leqslant \left(\sum_{\pi\in\mathcal{P}_0(\boldsymbol{a},\boldsymbol{b})} \left|\delta_\pi\right| + \sum_{\pi\in\mathcal{P}_+(\boldsymbol{a},\boldsymbol{b})} 2\left|\delta_\pi\right|\right) \max\left\{\nu_{0,i^*},\ -\bar{\nu}_{0,j^*},\ \frac{1}{2}\nu_{i_1^*,i_2^*},\ -\frac{1}{2}\bar{\nu}_{j_1^*,j_2^*}\right\} \\
&\underset{(4.41)}{\leqslant} \max\left\{\nu_{0,i^*},\ -\bar{\nu}_{0,j^*},\ \frac{1}{2}\nu_{i_1^*,i_2^*},\ -\frac{1}{2}\bar{\nu}_{j_1^*,j_2^*}\right\}.
\end{aligned}
$$

All four values inside the maximum can be attained with the following multipliers:

$$
\begin{aligned}
\nu_{0,i^*}: &\quad \lambda_{i^*} = 1, & \frac{1}{2}\nu_{i_1^*,i_2^*}: &\quad \lambda_{i_1^*} = \mu_{i_2^*+1} = \frac{1}{2},\\
-\bar{\nu}_{0,j^*}: &\quad \mu_{j^*} = 1, & -\frac{1}{2}\bar{\nu}_{j_1^*,j_2^*}: &\quad \mu_{j_1^*} = \lambda_{j_2^*+1} = \frac{1}{2}.
\end{aligned}
$$

Since the maximum's value is attainable with the multipliers as described above, this directly gives the optimal solution's value. □

Remark 4.10 (Solution interpretation)
The inner problem for the solutions with $\lambda_{i^*} = 1$ or $\mu_{j^*} = 1$ contains just one constraint of the type

$$\theta \geqslant \pm \sum_{i=0}^{i^*/j^*} (p_i - q_i)\,\Delta t_i,$$

which provides the tightest setting for θ. The free controls would be set considering the worst-case scenario regarding the bound (either all 0 in the "+"-case or all 1 in the "−"-case). Hence, the bound takes a stronger value when there are less degrees of freedom left in $[t_0, t_{i^*/j^*}]$.
The solutions with two multipliers set to $\frac{1}{2}$ only consider a time interval $[t_{i_1}, t_{i_2}]$ and the largest term

$$\pm \sum_{i=i_1}^{i_2} (p_i - q_i)\Delta t_i.$$

Since there is no valid inequality containing this term and θ, we have to split a valid inequality to explain the behavior:

$$\theta \geqslant \pm \sum_{i=0}^{i_2} (p_i - q_i)\Delta t_i$$

$$= \pm \sum_{i=i_1}^{i_2} (p_i - q_i)\Delta t_i \underbrace{\pm \sum_{i=0}^{i_1-1} (p_i - q_i)\Delta t_i}_{\geqslant -\theta}$$

$$\Rightarrow \quad \theta \geqslant \pm \frac{1}{2} \sum_{i=i_1}^{i_2} (p_i - q_i)\Delta t_i .$$

We observe that the factor of $\frac{1}{2}$ is needed to cancel possibly bad control decisions during the interval $[t_0, t_{i_1})$.
Thus, Theorem 4.3 gives two different types of bounds, which also vastly differ in their quality. The second type calculates the value in the same way, but has a pre-factor of $\frac{1}{2}$, which weakens the bound considerably. Hence, branching strategies should be preferred, which lead to the first type of bound.

The choice that puts most emphasis on this way of bound derivation fixes the controls forward in time, i.e., $\mathcal{F} = \{0, \ldots, |\mathcal{F}| - 1\}$. We expand the analysis for this set $\mathcal{F}$ of fixed controls.

Proposition 4.6
If the first $|\mathcal{F}|$ controls are the fixed ones, i.e., $\mathcal{F} = \{0, \ldots, |\mathcal{F}| - 1\}$, then it holds for all time indices in $\mathcal{F}$:

$$\max\left\{v_{0,i^*}, -\bar{v}_{0,j^*}\right\} \geqslant \quad \frac{1}{2} v_{i_1,i_2}, \qquad \forall\, i_1 < i_2 < |\mathcal{F}|,$$
$$\max\left\{v_{0,i^*}, -\bar{v}_{0,j^*}\right\} \geqslant -\frac{1}{2} \bar{v}_{j_1,j_2}, \qquad \forall\, j_1 < j_2 < |\mathcal{F}|,$$

and hence, the value of the LAGRANGIAN *relaxation becomes*

$$Z_{\mathrm{LR}} = \max\left\{v_{0,i^*}, -\bar{v}_{0,j^*}\right\}. \qquad \triangle$$

Proof Both inequalities can be proven directly and we only show the first case, since the second one can be shown analogously. It holds for $i, j < |\mathcal{F}|$: $v_{i,j} = \bar{v}_{i,j}$.

$$\begin{aligned}
\frac{1}{2} v_{i_1,i_2} &\underset{\text{def } v}{=} \frac{1}{2}\left(v_{0,i_2} - v_{0,i_1-1}\right) \\
&\underset{\substack{\text{def } i^*,j^*,\\ v=\bar{v}}}{\leqslant} \frac{1}{2}\left(v_{0,i^*} - \bar{v}_{0,j^*}\right) \\
&\leqslant \frac{1}{2}\left(\max\left\{v_{0,i^*}, -\bar{v}_{0,j^*}\right\} + \max\left\{v_{0,i^*}, -\bar{v}_{0,j^*}\right\}\right) \\
&= \max\left\{v_{0,i^*}, -\bar{v}_{0,j^*}\right\}.
\end{aligned}$$

□

Remark 4.11 (Possible application of the proposition)
Theorem 4.3 and Proposition 4.6 do not have a lot of value considering the presented setting, because here the SUR scheme already provides very good solutions in linear time. However, in the presence of additional constraints, which limit the feasible set to be a subset of the corners of the n-dimensional binary cube, the SUR scheme might produce an infeasible solution. In this case, Theorem 4.3 provides a way to calculate bounds in a branch-and-bound framework in linear/quadratic time, which correspond to the solution of the LAGRANGIAN relaxation of the relaxed problem that ignores the additional constraints.

4.4 Approximation of SOS1-coupled controls in the integral sense

As already specified in Section 3.6, general nonlinear ODE systems can be reformulated into control-affine ODE systems as (4.7) through the OC approach. However, as stated there, one gets additional SOS1-constraints on the binary controls. Therefore, it is of high interest to also cover the Control Approximation Problem in these systems.

After applying the OC technique, Theorem 4.1 is applicable and we can approximate the relaxed controls $\boldsymbol{\alpha} : [t_0, t_f] \to \mathbb{R}^{n_\omega}$ with binary ones $\boldsymbol{\omega} : [t_0, t_f] \to \{0,1\}^{n_\omega}$. This takes place on a time grid $\mathbb{G}_m \stackrel{\text{def}}{=} \{t_0 < t_1 < \ldots < t_m = t_f\}$ with time increments

$$\Delta t_i = t_{i+1} - t_i,$$

and maximum time increment

$$\Delta t \stackrel{\text{def}}{=} \max_{0 \leqslant i \leqslant m-1} \Delta t_i.$$

Let $\omega : [t_0, t_f] \to \{0,1\}^{n_\omega}$ be piecewise constant with possible jumps in the grid points. Then, ω can be parameterized with a vector $\boldsymbol{p} \in \{0,1\}^{n_\omega m}$ as

$$\omega_i(t) = p_{i,j}, \qquad 1 \leqslant i \leqslant n_\omega, t \in [t_j, t_{j+1}).$$

The values in the end points $\omega_i(t_f)$ do not contribute and can be omitted in the parametrization. The SOS1-constraints is also required for the relaxed controls $\boldsymbol{\alpha}(\cdot)$, i.e.,

$$\sum_{i=1}^{n_\omega} \alpha_i(t) = 1, \qquad t \in [t_0, t_f], \tag{4.42}$$

as the relaxation should not drop this important constraint.

As described in Section 4.3, it makes sense to compute an approximation such that the inte-

grated control deviation becomes minimal, i.e.,

$$\begin{aligned} \min_{\boldsymbol{\omega}(\cdot)} \quad & \max_{t\in[t_0,t_f]} \quad \left|\left| \int_{t_0}^{t} \alpha_i(\tau) - \omega_i(\tau)\,\mathrm{d}\tau \right|\right| \\ \text{s.t.} \quad & \sum_{i=1}^{n_\omega} \omega_i(t) = 1, \qquad \forall t \in [t_0, t_f]. \end{aligned} \tag{4.43}$$

Analogously to Section 4.3, we can state that the norm takes its maximum values at grid points and thereby consider the maximization problem only on grid points. Since we only need to consider grid points, we can also consider mean values $\boldsymbol{q}$ for the relaxed controls, i.e.,

$$q_{i,j} \stackrel{\text{def}}{=} \frac{1}{\Delta t_j} \int_{t_j}^{t_{j+1}} \alpha_i(\tau)\,\mathrm{d}\tau.$$

The SOS1-constraint of $\boldsymbol{\alpha}(\cdot)$ directly carries over to $\boldsymbol{q}$

$$\sum_{i=1}^{n_\omega} q_{i,j} = 1, \qquad 0 \leqslant j \leqslant m-1. \tag{4.44}$$

The algorithms presented in this section use the *maximum norm* in (4.43) since it allows for an MILP reformulation. Other common norms would be the *Manhattan norm* and the *Euclidean norm*; the latter is briefly considered in Section 4.5. With the maximum norm and a commonly used reformulation, the problem becomes

$$\min_{\theta,\boldsymbol{p}} \quad \theta \tag{4.45a}$$

$$\text{s.t.} \quad \theta \geqslant \sum_{k=0}^{j} \left(q_{i,k} - p_{i,k}\right) \Delta t_k, \qquad 0 \leqslant j \leqslant m-1, 1 \leqslant i \leqslant n_\omega, \tag{4.45b}$$

$$\theta \geqslant -\sum_{k=0}^{j} \left(q_{i,k} - p_{i,k}\right) \Delta t_k, \qquad 0 \leqslant j \leqslant m-1, 1 \leqslant i \leqslant n_\omega, \tag{4.45c}$$

$$1 = \sum_{i=1}^{n_\omega} p_{i,j}, \qquad 0 \leqslant j \leqslant m-1, \tag{4.45d}$$

$$\boldsymbol{p} \in \{0,1\}^{n_\omega m}. \tag{4.45e}$$

Due to the SOS1-constraint (4.45d), we have for each time interval $[t_j, t_{j+1})$ exactly one control i that takes the value 1 and the others take the value 0. In this context, we use the terms *active* control i or *activation* of control i in time step j.

4.4.1 Sum-up Rounding with SOS1-coupled controls

The first algorithmic approximation was given in [152] as the SOS1-SUR scheme. Its idea is to activate the control that would deviate most from 0 without an activation. In [154], the authors give a proof for its guaranteed quality but also proof that the quality of this heuristic does depend on the number of controls n_ω. We prove these results in this section.

Proposition 4.7 (Deviation of Control Integrals by SOS1-SUR)
Let $\mathbb{G}_m$ be a time grid as described above. Let $\boldsymbol{\alpha} : [t_0, t_f] \to [0,1]^{n_\omega}$ be a measurable function, which satisfies the SOS1-constraint (4.44)*, with mean values $\boldsymbol{q}$ on the grid intervals:*

$$q_{ij} = \frac{1}{\Delta t_j} \int_{t_j}^{t_{j+1}} \alpha_i(\tau)\, \mathrm{d}\tau.$$

Let the auxiliary expressions $\tilde{p}_{i,j}$ be defined as

$$\tilde{p}_{i,j} \overset{\text{def}}{=} \sum_{k=0}^{j} q_{i,k} \Delta t_k - \sum_{k=0}^{j-1} p_{i,k}^{SUR} \Delta t_k.$$

Then, it holds that the feasible point of the SOS1-coupled Control Approximation Problem in integral sense (4.45) *provided by the iterative SOS1-SUR control scheme, i.e.,*

$$p_{i,j}^{SUR} \overset{\text{def}}{=} \begin{cases} 1, & \text{if } \tilde{p}_{i,j} \geqslant \tilde{p}_{k,j} \forall k \neq i, \text{ and } \forall k \text{ with } \tilde{p}_{i,j} = \tilde{p}_{k,j} : i \leqslant k, \\ 0, & \text{else,} \end{cases}$$

has a limited objective value with

$$\theta \leqslant (n_\omega - 1)\, \Delta t. \qquad \triangle$$

Proof We prove the proposition by contradiction.
Assume there exists a grid point t_r such that

$$\left| \sum_{j=0}^{r} \left(q_{i,j} - p_{i,j}^{\mathrm{SUR}} \right) \Delta t_j \right| > (n_\omega - 1)\, \Delta t.$$

Let the maximally violating control have the index

$$k \overset{\text{def}}{=} \underset{1 \leqslant i \leqslant n_\omega}{\operatorname{argmax}} \left| \sum_{j=0}^{r} \left(q_{i,j} - p_{i,j}^{\mathrm{SUR}} \right) \Delta t_j \right|.$$

First assume that $\theta > (n_\omega - 1)\,\Delta t$ because

$$\sum_{j=0}^{r} \left(q_{k,j} - p_{k,j}^{\mathrm{SUR}}\right) \Delta t_j < -\left(n_\omega - 1\right) \Delta t. \tag{4.46}$$

Let $l(k)$ be the time step of the last activation of control k before time step r, i.e.,

$$p_{k,l(k)} = 1, \qquad p_{k,j} = 0, \qquad l(k) < j < r.$$

Notice that there must have been at least one activation of k because inequality (4.46) holds.

Due to (4.46) and $l(k)$ being k's last activation, it holds

$$\begin{aligned}
\sum_{j=0}^{l(k)} \left(q_{k,j} - p_{k,j}^{\mathrm{SUR}}\right) \Delta t_j &= \sum_{j=0}^{l(k)} q_{k,j} \Delta t_j - \sum_{j=0}^{r-1} p_{k,j}^{\mathrm{SUR}} \Delta t_j \\
&\leqslant \sum_{j=0}^{r-1} \left(q_{k,j} - p_{k,j}^{\mathrm{SUR}}\right) \Delta t_j \\
&< -\left(n_\omega - 1\right) \Delta t,
\end{aligned}$$

and since $p_{k,l(k)}^{\mathrm{SUR}} = 1$, it holds

$$\sum_{j=0}^{l(k)} q_{k,j} \Delta t_j - \sum_{j=0}^{l(k)-1} p_{k,j}^{\mathrm{SUR}} \Delta t_j < -(n_\omega - 1)\Delta t + \Delta t_{l(k)} \leqslant -n_\omega \Delta t.$$

Due to the scheme's choice of the activated control k in time step $l(k)$, it must hold for all $1 \leqslant i \leqslant n_\omega$:

$$\begin{aligned}
&\tilde{p}_{k,l(k)} \geqslant \tilde{p}_{i,l(k)} \\
\Leftrightarrow \quad -n_\omega \Delta t > \sum_{j=0}^{l(k)} q_{k,j} \Delta t_j - \sum_{j=0}^{l(k)-1} p_{k,j}^{\mathrm{SUR}} \Delta t_j &\geqslant \sum_{j=0}^{l(k)} q_{i,j} \Delta t_j - \sum_{j=0}^{l(k)-1} p_{i,j}^{\mathrm{SUR}} \Delta t_j.
\end{aligned}$$

Summing this up for all controls and using the SOS1-constraint on both $\boldsymbol{p}$ and $\boldsymbol{q}$, we obtain

$$\begin{aligned}
-n_\omega^2 \Delta t = -\sum_{i=1}^{n_\omega} n_\omega \Delta t &> \sum_{i=1}^{n_\omega} \left(\sum_{j=0}^{l(k)} q_{i,j} \Delta t_j - \sum_{j=0}^{l(k)-1} p_{i,j}^{\mathrm{SUR}} \Delta t_j \right) \\
&= \sum_{j=0}^{l(k)} \underbrace{\left(\sum_{i=1}^{n_\omega} q_{i,j} \right)}_{=1} \Delta t_j - \sum_{j=0}^{l(k)-1} \underbrace{\left(\sum_{i=1}^{n_\omega} p_{i,j}^{\mathrm{SUR}} \right)}_{=1} \Delta t_j \\
&= \Delta t_{l(k)}. \qquad \lightning
\end{aligned}$$

Therefore, $\theta > (n_\omega - 1)\,\Delta t$ can only be because

$$\sum_{j=0}^{r} \left(q_{k,j} - p_{k,j}^{\mathrm{SUR}}\right) \Delta t_j > (n_\omega - 1)\,\Delta t. \tag{4.47}$$

Assume this is the case. Due to SOS1-constraint on both $\boldsymbol{p}^{\mathrm{SUR}}$ and $\boldsymbol{q}$, we get

$$\sum_{i=1}^{n_\omega} \left(\sum_{j=0}^{r-1} \left(q_{i,j} - p_{i,j}^{\mathrm{SUR}}\right) \Delta t_j \right) = 0.$$

With assumption (4.47), we obtain

$$\sum_{\substack{i=1 \\ i \neq k}}^{n_\omega} \left(\sum_{j=0}^{r-1} \left(q_{i,j} - p_{i,j}^{\mathrm{SUR}}\right) \Delta t_j \right) + (n_\omega - 1)\,\Delta t < 0.$$

The left-hand side can be split up into the following $n_\omega - 1$ terms

$$\Delta t + \sum_{j=0}^{r-1} \left(q_{i,j} - p_{i,j}^{\mathrm{SUR}}\right) \Delta t_j, \qquad 1 \leqslant i \leqslant n_\omega, i \neq k.$$

For the whole sum to be negative, at least one of the terms has to be negative. Let c be its index:

$$\sum_{j=0}^{r-1} \left(q_{c,j} - p_{c,j}^{\mathrm{SUR}}\right) \Delta t_j < -\Delta t. \tag{4.48}$$

Let again $l(c)$ be the last time step before r, where c was activated. Control c must have been activated due to inequality (4.48). This leads to

$$\begin{aligned} -\Delta t &> \sum_{j=0}^{r-1} \left(q_{c,j} - p_{c,j}^{\mathrm{SUR}}\right) \Delta t_j \\ &\geqslant \sum_{j=0}^{l(c)} \left(q_{c,j} - p_{c,j}^{\mathrm{SUR}}\right) \Delta t_j \\ &= \sum_{j=0}^{l(c)} q_{c,j} \Delta t_j - \sum_{j=0}^{l(c)-1} p_{c,j}^{\mathrm{SUR}} \Delta t_j - \Delta t_{l(c)} \\ \Rightarrow \qquad 0 &> \sum_{j=0}^{l(c)} q_{c,j} \Delta t_j - \sum_{j=0}^{l(c)-1} p_{c,j}^{\mathrm{SUR}} \Delta t_j. \end{aligned} \tag{4.49}$$

As the scheme activated control c in time step $l(c)$, holds for $1 \leqslant i \leqslant n_\omega$

$$\begin{aligned} & \tilde{p}_{c,l(c)} \geqslant \tilde{p}_{i,l(c)} \\ \Leftrightarrow \quad 0 > \sum_{j=0}^{l(c)} q_{c,j}\Delta t_j - \sum_{j=0}^{l(c)-1} p_{c,j}^{\mathrm{SUR}}\Delta t_j \geqslant & \sum_{j=0}^{l(c)} q_{i,j}\Delta t_j - \sum_{j=0}^{l(c)-1} p_{i,j}^{\mathrm{SUR}}\Delta t_j. \end{aligned}$$

Summing them up for $1 \leqslant i \leqslant n_\omega$ and using again the SOS1-constraints, we obtain the contradiction:

$$0 > \sum_{j=0}^{l(c)} \underbrace{\left(\sum_{i=1}^{n_\omega} q_{i,j}\right)}_{=1} \Delta t_j - \sum_{j=0}^{l(c)-1} \underbrace{\left(\sum_{i=1}^{n_\omega} p_{i,j}^{\mathrm{SUR}}\right)}_{=1} \Delta t_j = \Delta t_{l(c)}. \qquad \lightning$$

The solution quality of SOS1-SUR can hence not be worse than $\left(n_\omega - 1\right)\Delta t$. □

The problem with this heuristic is its actual dependence on the number of controls, as can be seen in the next example.

Remark 4.12 (Best *SOS1-SUR* objective value must depend on n_ω)
Consider the following setting, there are n_ω controls and n_ω equidistant time steps of length Δt. The relaxed controls are constant on the time intervals and are hence identical to their mean values $\boldsymbol{q}$ as follows:

$$q_{i,j} = \begin{cases} \frac{1}{n_\omega - j}, & \text{if } i \geqslant j, \\ 0, & \text{if } i < j, \end{cases} \qquad 1 \leqslant i \leqslant n_\omega, 0 \leqslant j \leqslant n_\omega - 1.$$

Here, the SOS1-SUR scheme activates the control that was increased the last time and remains at 0 thereafter. At each point in time, the error made is locally maximal. We get the controls

$$p_{i,j}^{\mathrm{SUR}} = \begin{cases} 1, & \text{if } i = j+1, \\ 0, & \text{else}, \end{cases} \qquad 1 \leqslant i \leqslant n_\omega, 0 \leqslant j \leqslant n_\omega - 1,$$

with the maximal deviation at time $n_\omega - 1$ for control n_ω of

$$\theta = \sum_{j=0}^{n_\omega - 1} \left(q_{n_\omega,j}\Delta t\right) - \Delta t = \Delta t \sum_{j=0}^{n_\omega - 2} \frac{1}{n_\omega - j} = \Delta t \sum_{j=2}^{n_\omega} \frac{1}{j},$$

which is Δt times the harmonic number, which is approximately $\Delta t \log(n_\omega)$.
SAGER conjectured in [158] that this is the worst case behavior of the algorithm, i.e., instead of $\theta \leqslant \left(n_\omega - 1\right)\Delta t$, we have $\theta \leqslant \mathcal{O}(\log(n_\omega)\Delta t)$.

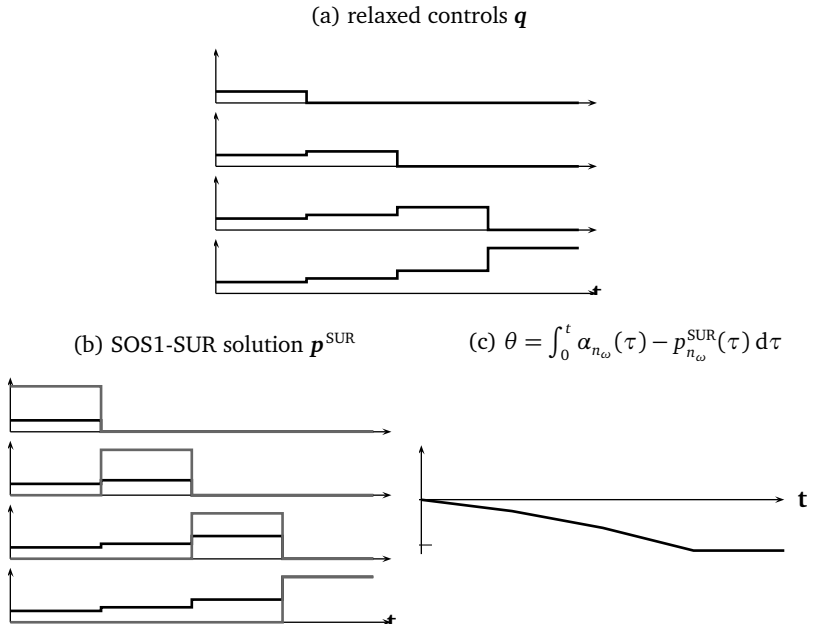


Figure 4.4: The behavior of the SOS1-SUR solution in the described example with $n_\omega = 4$. As shown, the approximation value θ in this example is Δt times the harmonic number of n_ω and hence the solution quality can never be independent of n_ω.

4.4.2 Next-forced Rounding for SOS1-coupled controls

In Section 4.4.1, it was shown that the deviation value of the SOS1-SUR scheme depends on the number of controls n_ω. Here, we present a new algorithm that does not. Instead, it provides a feasible point of problem (4.45) with an objective value of $\theta \leqslant \Delta t$. In this context, we distinguish different types of activations. An *admissible activation* is one where the activation would not violate $\theta \leqslant \Delta t$ and a *forced activation* is an activation that is needed to prevent $\theta > \Delta t$:

Definition 4.6 (Admissible activation)
An activation of control c_a is admissible in time step i if it holds

$$\sum_{j=0}^{i} q_{c_a,j}\Delta t_j - \sum_{j=0}^{i-1} p_{c_a,j}\Delta t_j \geqslant -\Delta t + \Delta t_i. \tag{4.50}$$

△

Definition 4.7 (Forced activation)
An activation of control c_f is forced in time step i if it holds

$$\sum_{j=0}^{i} q_{c_f,j}\Delta t_j - \sum_{j=0}^{i-1} p_{c_f,j}\Delta t_j > \Delta t. \tag{4.51}$$

△

With these terms, we can now state the Next-forced Rounding (NFR) heuristic 4.1 in pseudo-code.

Algorithm 4.1: Next-forced Rounding algorithm for SOS1-coupled controls.

Data: Time increments Δt_i, mean values of relaxed controls $q_{j,i}$ satisfying SOS1-constraint (4.44) for $i = 0 : m-1$, $j = 1 : n_\omega$.

Result: Binary controls $p_{i,j}^{\mathrm{NFR}}$ that give a feasible point of the Control Approximation Problem (4.45) with SOS1-coupled constraints with $\theta \leqslant \Delta t$.

for *all time steps* $i = 0 : m-1$ **do**
  for *all controls* $j = 1 : n_\omega$ **do**
    $p_{j,i}^{\mathrm{NFR}} = 0.$
  end
  $\mathcal{A}_i \overset{\text{def}}{=}$ the set of admissible controls in this time step i.
  if *there exists control* c_f *with forced activation in time step* i **then**
    $p_{c_f,i}^{\mathrm{NFR}} = 1.$
  else if *it exists an admissible control that becomes forced without activation, i.e.,*
  $\exists j \in \mathcal{A}_i : \sum_{k=0}^{m-1} q_{j,k}\Delta t_k - \sum_{k=0}^{i-1} p_{j,k}\Delta t_k > \Delta t$ **then**
    For all $j \in \mathcal{A}_i$, calculate the next point $f(j)$ of forced activation (m if none exists):
    $$f(j) \overset{\text{def}}{=} \min\left\{ l \in \mathbb{N} \,\middle|\, i+1 \leqslant l \leqslant m-1, \sum_{k=0}^{l} q_{j,k}\Delta t_k - \sum_{k=0}^{i-1} p_{j,k}\Delta t_k > \Delta t \right\} \cup \{m\}.$$
    Find the earliest point of forced activation:
    $f_a = \min\{f(j) \mid j \in \mathcal{A}_i\}.$
    $c_a = \min\{j \in \mathcal{A}_i \mid f(j) = f_a\}.$
    $p_{c_a,i}^{\mathrm{NFR}} = 1.$
  else
    $c_a = \min\{j \in \mathcal{A}_i\}.$
    $p_{c_a,i}^{\mathrm{NFR}} = 1.$
  end
end

Proposition 4.8 (Solution quality of NFR)
Algorithm 4.1 provides a feasible point of problem (4.45) *with*

$$\theta \leqslant \Delta t. \qquad \triangle$$

Proof First, we note that the algorithm only uses admissible and forced activations. If the algorithm provided a solution it would automatically satisfy $\theta \leqslant \Delta t$. Any solution would

automatically satisfy the SOS1-constraints (4.45d) since exactly one control is activated for each time step during the corresponding step of the loop.

Now, we only have to prove that the algorithm always produces a complete solution. We split this into two parts: First, we show that during each time step in the for-loop there always exists at least one admissible control. Then, we show that there exists at most one forced activation during each time step.

1. We prove that for each time step there is at least one admissible activation that could be chosen. This implies that the algorithm runs through the whole loop. We show this by contradiction:
   Consider the situation after $k-1$ time steps, there has been an activation for each time step due to the mechanics of the loop. Let no controls in time step k be admissible, i.e.,

   $$\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j < -\Delta t + \Delta t_k, \qquad 1 \leqslant i \leqslant n_\omega.$$

   Then, we sum these up and use the SOS1-constraint for both $\boldsymbol{p}$ and $\boldsymbol{q}$ to obtain

   $$\begin{aligned} & \sum_{i=1}^{n_\omega} \left(\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j \right) < n_\omega \left(-\Delta t + \Delta t_k\right) \\ \Leftrightarrow \quad & \sum_{j=0}^{k} \Delta t_j - \sum_{j=0}^{k-1} \Delta t_j < n_\omega \left(-\Delta t + \Delta t_k\right) \\ \Leftrightarrow \quad & \left(1 - n_\omega\right) \Delta t_k < -n_\omega \Delta t. \qquad \lightning \end{aligned}$$

   Therefore, there has to be an admissible control during time step k.
2. We prove that there may be at most one forced activation for each time step k. If this were not the case, the algorithm would not be well defined. We also show this by contradiction: Let the algorithm be run through time steps $0, \ldots, k-1$ and let k be the first time step at which it encounters at least two forced activations of controls i_1 and i_2. Then, it holds

   $$\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j > \Delta t, \qquad i \in \{i_1, i_2\}. \tag{4.52}$$

   For the combination of all controls, and using the fact that each previous time step had exactly one activation, it holds

   $$\sum_{i=1}^{n_\omega} \left(\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j \right) = \sum_{j=0}^{k} \Delta t_j - \sum_{j=0}^{k-1} \Delta t_j = \Delta t_k \leqslant \Delta t. \tag{4.53}$$

We can decompose the left-hand side with i_1 and i_2 into

$$\begin{aligned}
\Delta t \geqslant & \sum_{i=1}^{n_\omega} \left(\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j \right) \\
= & \sum_{\substack{i=1 \\ i\notin\{i_1,i_2\}}}^{n_\omega} \left(\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j \right) \\
& + \underbrace{\sum_{i\in\{i_1,i_2\}} \left(\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j \right)}_{>2\Delta t}.
\end{aligned}$$

Therefore, we get for the remaining part

$$\sum_{\substack{i=1 \\ i\notin\{i_1,i_2\}}}^{n_\omega} \left(\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j \right) < -\Delta t.$$

Since the whole sum is negative, there is at least one negative summand. Let c be the corresponding control index such that

$$\sum_{j=0}^{k} q_{c,j}\Delta t_j - \sum_{j=0}^{k-1} p_{c,j}\Delta t_j < 0.$$

Let $l(c)$ be the time step of the last activation of control c, then it holds

$$\begin{aligned}
0 > & \sum_{j=0}^{k} q_{c,j}\Delta t_j - \sum_{j=0}^{k-1} p_{c,j}\Delta t_j \\
= & \sum_{j=0}^{k} q_{c,j}\Delta t_j - \sum_{j=0}^{l(c)} p_{c,j}\Delta t_j \\
= & \sum_{j=0}^{k} q_{c,j}\Delta t_j - \sum_{j=0}^{l(c)-1} p_{c,j}\Delta t_j - \Delta t_{l(c)}. \\
\Rightarrow \quad & \sum_{j=0}^{k} q_{c,j}\Delta t_j - \sum_{j=0}^{l(c)-1} p_{c,j}\Delta t_j < \Delta t_{l(c)} \leqslant \Delta t.
\end{aligned}$$

This shows that the activation of control c would not have been forced at time step k. Now, we have established that there exists at least one control that – without activation – would not have become forced at time step k, and yet still was activated before that time step. Let c_l be the control of this type that was activated last. We now show that there

cannot be a control i_1 that becomes forced at time step k in this scenario.
Due to the choice of c_l, all controls that were activated after $l(c_l)$ would have become forced until time step k, we collect them in $\mathcal{A}$ together with i_1, which becomes forced at time step k. For each of those controls $i \in \mathcal{A}\backslash\{i_1\}$, the last activation would have become forced at or before time step k, i.e.,

$$\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{l(i)-1} p_{i,j}\Delta t_j > \Delta t, \qquad i \in \mathcal{A}\backslash\{i_1\}.$$

They were last activated at time step $l(i)$ and with this activation we get

$$\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j > \Delta t - \Delta t_{l(i)} \geqslant 0, \qquad i \in \mathcal{A}\backslash\{i_1\}. \tag{4.54}$$

For i_1 this is even stricter since there was no activation yet. We obtain

$$\sum_{j=0}^{k} q_{i_1,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i_1,j}\Delta t_j > \Delta t. \tag{4.55}$$

However, since c_l was chosen at time step $l(c_l)$ and due to the else-part in Algorithm 4.1, none of the controls in $\mathcal{A}$ has been admissible at time step $l(c_l)$ and none of them could hence be activated:

$$\begin{aligned} &\sum_{j=0}^{l(c_l)} q_{i,j}\Delta t_j - \sum_{j=0}^{l(c_l)-1} p_{i,j}\Delta t_j < -\Delta t + \Delta t_{l(c_l)} \leqslant 0, \qquad i \in \mathcal{A}, \\ \Rightarrow \quad &\sum_{j=0}^{l(c_l)} q_{i,j}\Delta t_j - \sum_{j=0}^{l(c_l)} p_{i,j}\Delta t_j < -\Delta t + \Delta t_{l(c_l)} \leqslant 0, \qquad i \in \mathcal{A}. \end{aligned} \tag{4.56}$$

Taking the sum of inequalities (4.54), (4.55) and (4.56), we get

$$\sum_{i\in\mathcal{A}} \left(\sum_{j=0}^{k} q_{i,j}\Delta t_j - \sum_{j=0}^{k-1} p_{i,j}\Delta t_j \right) > \Delta t, \tag{4.57}$$

$$\sum_{i\in\mathcal{A}} \left(\sum_{j=0}^{l(c_l)} q_{i,j}\Delta t_j - \sum_{j=0}^{l(c_l)} p_{i,j}\Delta t_j \right) < 0. \tag{4.58}$$

Subtracting (4.58) from (4.57), we obtain

$$\sum_{i\in\mathcal{A}} \left(\sum_{j=l(c_l)+1}^{k} q_{i,j}\Delta t_j - \sum_{j=l(c_l)+1}^{k-1} p_{i,j}\Delta t_j \right) > \Delta t.$$

Since by definition of $\mathcal{A}$ it contains all control activations between time steps $l(c_l)+1$ and $k-1$ and due to the SOS1-constraint on $\boldsymbol{q}$, we can rearrange the terms to become

$$\begin{aligned}\Delta t &< \sum_{j=l(c_l)+1}^{k} \Delta t_j \underbrace{\sum_{i\in\mathcal{A}} q_{i,j}}_{\leqslant 1} - \sum_{j=l(c_l)+1}^{k-1} \Delta t_j \underbrace{\sum_{i\in\mathcal{A}} p_{i,j}}_{=1} \\ &\leqslant \sum_{j=l(c_l)+1}^{k} \Delta t_j - \sum_{j=l(c_l)+1}^{k-1} \Delta t_j = \Delta t_k \leqslant \Delta t. \qquad \lightning\end{aligned}$$

Therefore, we have shown that there cannot be two forced control activations at time step k.

This concludes the proof. We have shown that the algorithm is well-defined because there is always exactly one choice that can be made. □

Remark 4.13 (Computational effort)
The computational effort of the NFR algorithm is $\mathcal{O}(n_\omega m^2)$, whereas the effort for the SOS1-SUR algorithm is only $\mathcal{O}(n_\omega m)$. Both provide worst-case bounds for the objective of the Control Approximation Problem that are linear in Δt and hence both suffice to also drive the state approximation arbitrarily close according to Theorem 4.1, if the grid is just chosen fine enough. Notice that the SUR heuristic only needs past data to derive the decision for each step while the NFR heuristic needs the whole time horizon (or at least until the next forced activation) to make its decisions.

4.4.3 The LAGRANGIAN relaxation for SOS1-coupled controls

In this section, we investigate the LAGRANGIAN relaxation with more than two binary controls ($n_\omega > 2$) but no additional combinatorial constraints, cf. problem (4.45). The case of two SOS1-coupled controls can be transformed to the one-dimensional case through the elimination of one control using the SOS1-constraint.

In Section 4.3.2, we determined the solution of the LAGRANGIAN relaxation of the one-dimensional problem. Due to the SOS1-constraint, a valid branching strategy is to create at each node n_ω child nodes, which each fix one different control with the same time index to 1 and all the other controls with this time index to 0. For this section, we focus on a branching strategy that applies this branching forward in time, i.e., the set of fixed controls for a node of depth d is $\{p_{i,j} \mid j < d\}$. Using $\mathcal{F}$ analogously to Section 4.3.2, we obtain $\mathcal{F} = \{0, \dots, |\mathcal{F}|-1\}$.

We use the same LAGRANGIAN relaxation as in Section 4.3.2, which relaxes the constraints (4.45b) and (4.45c) that are used to reformulate the maximum of the norm and eliminates

θ:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\mu}\in\mathbb{R}_+^{n_\omega m}} \min_{\boldsymbol{p}\in\{0,1\}^{n_\omega m}} \quad \sum_{i=0}^{n_\omega}\sum_{j=0}^{m-1}(\lambda_{i,j}-\mu_{i,j})\left(\sum_{k=0}^{j}(p_{i,k}-q_{i,k})\Delta t_k\right) \tag{4.59}$$

$$\text{s.t.} \quad p_{i,j} \text{ fixed to } p_{i,j}^{\text{fixed}} \qquad j\in\mathcal{F}, 1\leqslant i\leqslant n_\omega,$$

$$\sum_{i=1}^{n_\omega} p_{i,j}=1, \qquad j\notin\mathcal{F},$$

$$\text{s.t.} \quad \sum_{i=1}^{n_\omega}\sum_{j=0}^{m-1}(\lambda_{i,j}-\mu_{i,j})=1.$$

Proposition 4.9
The objective value of the solution of the LAGRANGIAN *relaxation* (4.59) *with time ordered branching is*

$$Z_{\text{LR}} = \max_{\substack{1\leqslant i\leqslant n_\omega,\\ j\in\mathcal{F}}} \left|\left|\sum_{k=0}^{j}(p_{i,k}^{\text{fixed}}-q_{i,k})\Delta t_k\right|\right|_\infty \qquad \triangle$$

Proof For fixed λ, μ, after reordering of terms, the inner minimization problem can be rewritten as

$$\min_{\boldsymbol{p}\in\{0,1\}^{n_\omega m}} \quad \sum_{i=0}^{n_\omega}\sum_{j=0}^{m-1}(p_{i,j}-q_{i,j})\Delta t_j\left(\sum_{k=j}^{m-1}(\lambda_{i,k}-\mu_{i,k})\right) \tag{4.60a}$$

$$\text{s.t.} \quad p_{i,j} \text{ fixed to } p_{i,j}^{\text{fixed}} \qquad j\in\mathcal{F}, 1\leqslant i\leqslant n_\omega, \tag{4.60b}$$

$$\sum_{i=1}^{n_\omega} p_{i,j}=1, \qquad j\notin\mathcal{F}. \tag{4.60c}$$

With the abbreviation

$$c_{i,j} \stackrel{\text{def}}{=} \Delta t_j\left(\sum_{k=j}^{m-1}(\lambda_{i,k}-\mu_{i,k})\right),$$

we can directly solve the inner problem for fixed $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ and obtain the solution

$$p_{i,j}^* = \begin{cases} p_{i,j}^{\text{fixed}}, & \text{if } j\in\mathcal{F}, \\ 1, & \text{if } j\notin\mathcal{F}, \forall k\neq i: c_{i,j}\leqslant c_{k,j}, \forall k \text{ with } c_{i,j}=c_{k,j}: k>i, \\ 0, & \text{else.} \end{cases}$$

This is obtained since, for each time interval, exactly one control has to be activated. Since there is no interconnection between the different time intervals, the problem can be solved on each interval $[t_j, t_{j+1}]$ individually. There, the control with the smallest coefficient is activated:

$$c_j^* = \min_{1 \leqslant i \leqslant n_\omega} c_{i,j}.$$

Assume the relaxed controls $\boldsymbol{q}$ fulfill the SOS1-constraint (consistency of the relaxation), the value of the not fixed intervals is non-positive:

$$\begin{aligned}\sum_{i=1}^{n_\omega} \sum_{j \notin \mathcal{F}} c_{i,j}(p_{i,j} - q_{i,j}) &\underset{\text{def } \boldsymbol{p}^*}{=} \sum_{j \notin \mathcal{F}} \left(c_j^* - \sum_{i=1}^{n_\omega} c_{i,j}\, q_{i,j} \right) \\ &\underset{\text{def } c_j^*}{\leqslant} \sum_{j \notin \mathcal{F}} \left(c_j^* - \sum_{i=1}^{n_\omega} c_j^* q_{i,j} \right) \underset{\text{SOS1 on } \boldsymbol{q}}{=} \sum_{j \notin \mathcal{F}} (c_j^* - c_j^*) = 0.\end{aligned}$$

Therefore, the outer maximization problem sets $\lambda_{i,j} = \mu_{i,j}$ for all $j \notin \mathcal{F}$ and hence the objective contribution of the corresponding terms is 0. However, for the fixed part, the LAGRANGIAN relaxation just describes a convex combination of the terms

$$\pm \sum_{k=0}^{j} (p_{i,k}^{\text{fixed}} - q_{i,k}) \Delta t_k$$

as can be seen in the original problem formulation (4.59). Since there is no degree of freedom left for the $p_{i,j}$ with $j < |\mathcal{F}|$ regardless of the values of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, the optimal solution is the maximum of these terms. □

Remark 4.14 (Comparison: LAGRANGIAN and LP)
If the developed branching scheme is combined with a standard LP relaxation of the problem, the bounds of the LP relaxation and the LAGRANGIAN relaxation coincide. Due to the fixed parts, the *Control Approximation Problem* (4.45) contains the constraints

$$\theta \geqslant \pm \sum_{k=0}^{j} (p_{i,k}^{\text{fixed}} - q_{i,k}) \Delta t_k, \qquad j \in \mathcal{F}, 1 \leqslant i \leqslant n_\omega,$$

and hence also

$$\theta \geqslant Z_{\text{LR}} = \max_{\substack{1 \leqslant i \leqslant n_\omega, \\ j \in \mathcal{F}}} \left\| \sum_{k=0}^{j} (p_{i,k}^{\text{fixed}} - q_{i,k}) \Delta t_k \right\|_\infty .$$

Since $\theta = Z_{\mathrm{LP}} \leqslant Z_{\mathrm{LR}}$, it holds

$$Z_{\mathrm{LP}} = Z_{\mathrm{LR}}.$$

4.5 Approximation of generally coupled controls

In this section, we allow more general connections between different controls. We still want to approximate relaxed controls with binary ones, but now they have to fulfill additional constraints. However, we assume that the problem formulation uses the OC technique to model the different modes of the system and hence we assume the SOS1-constraint to be present. We give some examples for interesting classes of such constraints and examine the consequences of their presence.

As already seen in the SOS1-constrained multi-dimensional problem, the optimization need only consider the grid points. We can also replace the controls $\boldsymbol{\alpha}(\cdot)$ by their mean values $\boldsymbol{q}$ over the time intervals.

Let $\mathcal{C}$ be the set of feasible binary controls, which may e.g. be described by constraints

$$\boldsymbol{c}_{\mathcal{C}}(\boldsymbol{p}) \geqslant \mathbf{0}.$$

With the OC technique from Section 3.6, we can always reformulate $\boldsymbol{c}_{\mathcal{C}}$ to be affine in $\boldsymbol{p}$, i.e.,

$$\boldsymbol{c}_{\mathcal{C}}^T \boldsymbol{p} \geqslant \boldsymbol{c}_{\mathbf{0}}.$$

Therefore, the *Control Approximation Problem* in general form is

$$\begin{aligned} \min_{\boldsymbol{p}} \quad & \max_{0 \leq j \leq m-1} \left\| \sum_{k=0}^{j} \left(q_{i,k} - p_{i,k} \right) \Delta t_k \right\| \\ \text{s.t.} \quad & 1 = \sum_{i=1}^{n_\omega} p_{i,j}, \qquad 0 \leqslant j \leqslant m-1, \\ & \boldsymbol{c}_{\mathbf{0}} \leqslant \boldsymbol{c}_{\mathcal{C}}^T \boldsymbol{p}, \\ & \boldsymbol{p} \in \{0,1\}^{n_\omega m}. \end{aligned} \tag{4.61}$$

This problem is a bilevel optimization problem. Recently, in [101] an approach is made to give the structure of the corresponding feasible set. However, JONGEN and SHIKHMAN restrict themselves to problems where the inner problem has only one variable. Further information about bilevel programming can be found in [53, 54].

It is possible to reformulate the inner level with the already used standard trick, i.e., through the addition of a new variable θ. We replace the objective function by θ and add the con-

straints

$$\theta \geqslant \left|\left| \sum_{k=0}^{j} \left(q_{i,k} - p_{i,k} \right) \Delta t_k \right|\right|, \qquad 0 \leq j \leq m-1. \tag{4.62}$$

This reformulation generally does not remove the difficulty of the bilevel problem, it just hides it in the extra dimension of θ, where the feasible set becomes quite complicated as can be seen in Section 4.5.2.

4.5.1 Choice of the norm

It is possible to take different norms since Theorem 4.1 does not specify which norm to use. Depending on the choice of the norm, the reformulation may become an MILP or an MINLP. In the following, we consider the common cases of the *maximum norm*, the *Euclidean norm* and the *Manhattan norm*. Preliminary tests have shown that the quality of the results does not strongly depend on the norm used.

Maximum norm

This norm was already used in the pure SOS1-case. With the maximum norm and a common MILP reformulation for absolute values, the constraints become

$$\theta \geqslant \pm \sum_{j=0}^{k} \left(q_{i,j} - p_{i,j} \right) \Delta t_j, \qquad 1 \leqslant i \leqslant n_\omega, 0 \leqslant k \leqslant m-1, \tag{4.63}$$

and the whole problem becomes an MILP. This is an advantageous formulation since there exist very good solvers for MILPs.

Manhattan norm

With the Manhattan norm $||\cdot||_1$ and a common MILP reformulation for absolute values, problem (4.61) becomes

$$\begin{aligned} \theta &\geqslant \sum_{i=1}^{n_\omega} \mu_{ij}, && 0 \leqslant j \leqslant m-1, \\ \mu_{ij} &\geqslant \pm \sum_{k=0}^{j} \left(q_{i,k} - p_{i,k} \right) \Delta t_k, && 1 \leqslant i \leqslant n_\omega, 0 \leqslant j \leqslant m-1. \end{aligned}$$

In comparison with the maximum norm it introduces $n_\omega m$ extra variables μ are introduced in the MILP.

Euclidean norm

With the Euclidean norm the constraints become

$$\theta \geqslant \sqrt{\sum_{i=1}^{n_\omega} \left(\sum_{j=0}^{k} \left(q_{i,j} - p_{i,j} \right) \Delta t_j \right)^2}, \qquad 0 \leqslant k \leqslant m-1.$$

Since the inner part can be both positive and negative, these are nonlinear constraints, which cannot be replaced with a linear term. This makes the problem a MINLP, which makes this much harder to solve than the Manhattan norm formulation or the maximum norm formulation.
Due to these observations, we use the maximum norm in the following. It provides an MILP problem with the addition of only one continuous variable:

$$\begin{aligned}
\min_{\theta, \boldsymbol{p}} \quad & \theta && && (4.64)\\
\text{s.t.} \quad & \theta \geqslant \pm \sum_{j=0}^{k} \left(q_{i,j} - p_{i,j} \right) \Delta t_j && 1 \leqslant i \leqslant n_\omega, 0 \leqslant k \leqslant m-1,\\
& 1 = \sum_{i=1}^{n_\omega} p_{i,j} && 0 \leqslant j \leqslant m-1,\\
& \boldsymbol{c_0} \leqslant \boldsymbol{c}_C^T \boldsymbol{p},\\
& \boldsymbol{p} \in \{0,1\}^{n_\omega m}.
\end{aligned}$$

4.5.2 The Control Approximation Problem's polytope

Usually, for MILPs, it is a good idea to take a closer look at the polyhedral structure of the core problem. Insights obtained through this can often be applied to the problem through separation procedures.
However, it is well-known that the reformulation of the maximum as in (4.62) does not provide a good structure. In this context, this usually means that the facets defining the surface of the feasible set are dense and almost parallel. Often, this also coincides with rather large amounts of facets. We analyze the facets of the feasible space with regard to the Control Approximation Problem's reformulation (4.63) without additional constraints, i.e., $C = \emptyset$. To identify the structure of the convex hull of all feasible points of MILP (4.64), we use the software-package PORTA *v.1.4.1* [51, 52]. It uses the FOURIER-MOTZKIN *elimination* to obtain the outer description, i.e., all facet-defining inequalities, from the inner description, i.e., all feasible points. However, this method is limited to rather small problems with reasonable computational times.
Obviously the feasible points depend on the data as all the inequalities (4.63) do. This carries over to the facets and hence the structure of the polytope is strongly dependent on the data.

m	relaxed control $\alpha(\cdot)$ set to constant					
	0.0	0.1	0.2	0.3	0.4	0.5
6	13	23	30	96	99	38
7	15	31	50	204	251	58
8	17	40	97	472	516	80
9	19	50	182	1432	1088	120
10	21	61	319	4062	2457	162
11	23	73	502	7993	4296	242
12	25	119	855	20421	8440	334

Table 4.1: Number of facets of the polytope of feasible points. Due to the problem symmetry if the roles of 0 and 1 were swapped, only the values between 0 and 0.5 are taken. Exactly the same observations are made for two controls that are coupled with SOS1-constraints. The SOS1-equations are added but the rest of the facets remain the same.

m	constant relaxed controls $(\alpha_1(t), \alpha_2(t)) =$					
	(0.0,0.0)	(0.0,0.1)	(0.0,0.2)	(0.0,0.3)	(0.0,0.4)	(0.0,0.5)
5	16	35	47	293	208	59
6	19	50	96	906	695	84
7	22	70	235	2970	1808	267
m	constant relaxed controls $(\alpha_1(t), \alpha_2(t)) =$					
		(0.1,0.1)	(0.1,0.2)	(0.1,0.3)	(0.1,0.4)	
5		102	407	1013	1573	
6		296	1574	4767	8625	
7		1521	5410	26043	56870	
m	constant relaxed controls $(\alpha_1(t), \alpha_2(t)) =$					
		(0.2,0.2)	(0.2,0.3)	(0.2,0.4)	(0.3,0.3)	
5		353	2330	424	2305	
6		975	14226	1963	12138	
7		4723	106501	6656	*	

Table 4.2: Number of facets of the polytope of feasible points for the 3-dimensional problem with SOS1-constraints. Due to the problem symmetry we only looked at ordered controls where $\alpha_1 \leqslant \alpha_2 \leqslant \alpha_3 = 1 - \alpha_1 - \alpha_2$. The m SOS1-equations are not included in the above numbers. The $(0.2, 0.3)$-polytope took a little more than 70 days of computational time to compute and the $(0.3, 0.3)$-polytope should have around the same number of facets and hence its solution was not included in the thesis.

To examine a simple set of problems, the control to be approximated is fixed to a constant value over all time steps and an equidistant grid is used. Here, the classes of facets should be identifiable by hand if any obvious classes are present. Table 4.1 gives the numbers of facets for varying data for the one-dimensional case as in Section 4.3 and table 4.2 gives the numbers for SOS1-constrained three-dimensional cases as in Section 4.4. If all control values were 0 or 1, we could write down the exact description for any number of time steps m. However, the most complicated polytopes are those where the controls are all fractional and different from each other. These are also clearly the most difficult scenarios for the algorithm since there is no inherent problem symmetry to be exploited and the approximation is non-trivial. Here, we can see an exponential growth in the number of facets as the number m of time steps increases.

The facets of the Control Approximation Problem's polytope are relatively dense, i.e., there are lots of non-zero coefficients. There are also a lot of almost parallel facets where each facet on its own just cuts away a very small part of the infeasible set. These results can examplarily be seen in figure 4.5.

```
(1073) −2x1−2x2−2x3− x4− x5− x6− x7    −2x9− x10     − 5x12 <=  −8
(1074) −2x1−2x2− x3−2x4−2x5− x6−2x7− x8              − 5x12 <=  −8
(1075) −2x1−2x2−2x3− x4− x5    − x7− x8−2x9−2x10     − 5x12 <=  −8
(1076) −2x1−2x2−2x3− x4    −2x6−2x7− x8− x9− x10     − 5x12 <=  −8
(1077) −2x1−2x2−2x3− x4    − x6− x7− x8−2x9−2x10     − 5x12 <=  −8
(1078)     − x2−2x3−2x4−2x5−2x6−2x7− x8− x9− x10     − 5x12 <=  −8
(1079)     − x2− x3−2x4−2x5−2x6−2x7−2x8− x9− x10     − 5x12 <=  −8
(1080)         − x3−2x4− x5−2x6−2x7− x8−2x9−2x10− x11− 5x12 <=  −8
   ⋮
(2922) +2x1+2x2+ x3+ x4+ x5+ x6+ x7+2x8+ x9+2x10     − 5x12 <=   3
(2923) +2x1+2x2+ x3+ x4+ x5+2x6+ x7+2x8+ x9+ x10     − 5x12 <=   3
(2924) +2x1+2x2+ x3+ x4+2x5+ x6+ x7+2x8+ x9+ x10     − 5x12 <=   3
(2925) +3x1+3x2+ x3+ x4+ x5        + x8+ x9+ x10+ x11− 5x12 <=   3
(2926) +2x1+2x2+ x3+ x4+ x5+ x6+ x7+3x8+ x9+ x10     − 5x12 <=   3
(2927) +3x1+2x2+ x3+ x4+ x5+ x6+ x7+2x8+ x9+ x10     − 5x12 <=   3
(2928)     − x2               +3x7+4x8+3x9           − 5x12 <=   3
(2929)          − x4          +3x7+4x8+3x9           − 5x12 <=   3
```

Figure 4.5: Two excerpts of the output file of PORTA for the one-dimensional problem with $m = 11$ time steps and the relaxed control being constant $\alpha(\cdot) = 0.4$.
x1, ..., x11 represent the controls p_i and x12 represents θ.

This structure does make an analysis of different classes of facets difficult and we could not find a single valid generally applicable class of facets except the simple bounds and SOS1-constraints. Due to this difficult structure or absence of structure, we assume that there is no good cutting plane approach for this problem class.

4.5.3 Typical combinatorial constraints

Theorem 4.1 gives an upper bound for the possible state approximation by the control approximation in dependence of the grid size Δt. In the different settings described in Sections 4.3 and 4.4, we gave algorithms that realize this control approximation in direct dependence of the grid size Δt. Therefore, by refining the grid, both approximations can be made closer and the optimal solution of the relaxed problem can be approximated arbitrarily closely. However, this solution is often not desired since it usually incorporates chattering solutions, i.e., solutions with very fast switching between the different system modes as explained in Section 3.11.
To prevent this effect from happening, two possible steps can be taken:

- One could employ switching variables $\boldsymbol{\sigma}$ that measure the switching from one mode to another as described in Section 3.11. These switches are either penalized or limited to be below a given threshold $\sigma_{\max}$. The penalization is not considered if the problem is decoupled into the relaxed OCP and the Control Approximation Problem. It is very hard to connect the original objective with the control deviation of the integral controls, but at least the corresponding penalized value can be computed after a solution is obtained to provide an upper bound for following computations.

- One could employ min-up times, i.e., after the activation of a certain mode j in time step k coming from mode i, the system has to stay in this mode for a given time number of time intervals $\zeta_{i,j}$ as described in Section 3.11.2.

4.5.4 Branch-and-bound algorithm

The ideas of this section were first introduced in our work

[158] S. SAGER, M. JUNG AND C. KIRCHES, *Combinatorial Integral Approximation*, Mathematical Methods for Operations Research, 2011, Vol. 73(3):363–380.

In this section, we introduce a method to solve the Control Approximation Problem (4.64) with additional constraints to optimality. As already explained in Section 4.5.2, a branch-and-cut method seems not suited to the problem. Therefore, we present a tailored branch-and-bound method. Its special SOS1-branching strategy branches forward in time and for each time step all controls become fixed at once: one is activated whereas all others are set to 0. This is due to the insights obtained in Sections 4.3.2 and 4.4.3 from the derivation of the the one-dimensional LAGRANGIAN and the LAGRANGIAN coupled with SOS1-constraints.
The branch-and-bound algorithm's nodes are specified by their depth d in the tree, the fixed control variables $p_{i,j}$ for all $j < d$, and the corresponding lower bound θ of the objective function. Therefore, we abbreviate any node by $(d, \boldsymbol{p}, \theta)$. The algorithm in pseudo code is given as Algorithm 4.2.

Algorithm 4.2: Combinatorial branch-and-bound algorithm with special SOS1-branching.

Data: Time grid $\mathbb{G}_m$, relaxed mean values of controls $\boldsymbol{q}$, SOS1-constraints and additional coupling constraints ${\boldsymbol{c}_C}^T \boldsymbol{p} \geqslant \boldsymbol{c_0}$.
Result: Optimal solution $(\theta^*, \boldsymbol{p}^*)$ of problem (4.64).

Initialize the B&B tree $\mathcal{Q}$ with an empty node (0, {}, 0.0).
Set best solution: $p^*_{1,j} \leftarrow 1, p^*_{k,j} \leftarrow 0, 2 \leqslant k \leqslant n_\omega, 0 \leqslant j \leqslant m-1$.

$$\theta^* \leftarrow \max\left\{ \max_{2 \leqslant i \leqslant n_\omega} \left\{ \sum_{j=0}^{m-1} q_{i,j} \Delta t_j \right\}, \sum_{j=0}^{m-1} (1 - q_{1,j}) \Delta t_j \right\}.$$

while $\mathcal{Q} \neq \emptyset$ **do**
  $a \leftarrow$ next node of $\mathcal{Q}$ given by search strategy.
  if $a.\theta > \theta^*$ **then** Pruning step.
    Prune node a.
  else if $a.d = m$ **then**
    Mark a as currently optimal node.
    Set new best solution: $\theta^* \leftarrow a.\theta$, $\boldsymbol{p}^* \leftarrow a.\boldsymbol{p}$.
  else Create child nodes, use special SOS1-branching.
    forall the *elements of* $\left\{ \varphi \in \{0,1\}^{n_\omega} \mid \sum_{j=1}^{n_\omega} \varphi_j = 1 \right\}$ **do**
      Create new node n with:
      $n.d \leftarrow a.d + 1$,
      $n.\boldsymbol{p} \leftarrow a.\boldsymbol{p}$,
      $n.p_{i,n.d} \leftarrow \varphi_i, \qquad 1 \leq i \leq n_\omega$.
      if *$n.p$ fulfills the combinatorial constraints until time step $d+1$* **then**
        $n.\theta \leftarrow \max\left\{ a.\theta, \max_{i=1}^{n_\omega} \left\{ \pm \sum_{j=0}^{d} (p_{i,j} - q_{i,j}) \Delta t_j \right\} \right\}$
        Add n into $\mathcal{Q}$.
      end
    end
  end
end
return *optimal solution* $(\theta^*, \boldsymbol{p}^*)$.

The algorithm should be adapted in the presence of additional combinatorial constraints. The easiest adaptation modifies the node creation rules according to the constraints. For limited switching, it is best to fix all controls after the last allowed switching happened as done in Algorithm 4.3. If the maximum number of switches is reached, the resulting controls can be extended to the whole time horizon and hence give a solution. This solution can directly be compared with the best known solution.

Algorithm 4.3: Node creation algorithm in a limited switching environment.

Data: Node $a = (a.d, a.\boldsymbol{p}, a.\theta)$, limit of allowed switches $\sigma_{\max}$, branching tree $\mathcal{Q}$.
Result: Adds feasible child nodes to the branching tree.

$\sigma \leftarrow$ number of switches already used in $\boldsymbol{a.p}$.
forall the *elements of* $\left\{\varphi \in \{0,1\}^{n_\omega} \mid \sum_{j=1}^{n_\omega} \varphi_j = 1\right\}$ **do**
  Create new empty node n.
  $n.\boldsymbol{p} \leftarrow a.\boldsymbol{p}$.
  Let i be the index of φ such that $\varphi_i = 1$.
  if *this activation uses last allowed switch:* $a.p_{i,a.d-1} \neq 1$ *and* $\sigma + 1 = \sigma_{\max}$ **then**

$$
\begin{aligned}
n.d &\leftarrow m,\\
n.p_{i,j} &\leftarrow 1, \qquad a.d \leqslant j \leqslant m-1,\\
n.p_{k,j} &\leftarrow 0, \qquad a.d \leqslant j \leqslant m-1, 1 \leqslant k \leqslant n_\omega, k \neq i,\\
n.\theta &\leftarrow \max\left\{a.\theta, \max_{\substack{1\leqslant i\leqslant n_\omega,\\ a.d\leqslant j\leqslant m-1}} \left|\sum_{k=0}^{j}(n.p_{i,k} - q_{i,k})\Delta t_k\right|\right\}.
\end{aligned}
$$

  else

$$
\begin{aligned}
n.d &\leftarrow a.d + 1,\\
n.p_{i,a.d} &\leftarrow 1,\\
n.p_{k,a.d} &\leftarrow 0, \qquad 1 \leqslant k \leqslant n_\omega, k \neq i,\\
n.\theta &\leftarrow \max\left\{a.\theta, \max_{1\leqslant i\leqslant n_\omega} \left|\sum_{k=0}^{a.d}(n.p_{i,k} - q_{i,k})\Delta t_k\right|\right\}.
\end{aligned}
$$

  end
  Add n into $\mathcal{Q}$.
end

The adaptation for the combinatorial min-up times can be done analogously. To prevent infeasible nodes from being checked upon, all child nodes are created respecting the min-up times already, i.e., instead of only fixing the controls for the next time step, the controls for the next $\zeta_{i,j}$ time intervals are fixed when control i is activated coming from mode j.
The bound computation does not change at all in the presence of additional constraints, hence the algorithm still behaves good in the two following two scenarios: First, if the additional constraints are very tight and hence the feasible set is much smaller, the branching strategy

can prune big parts of the branching tree much faster and maintain a good performance. The second scenario is when the additional combinatorial constraints are very loose and the LAGRANGIAN relaxation, which drops all additional combinatorial constraints, is quite close to the true LAGRANGIAN relaxation, hence the algorithm uses good bounds and explores only the necessary parts of the branching tree.

Technical realization

The algorithm has been implemented in C++. There are several points that need further explanation in the pseudo-code Algorithm 4.2: As the node selection strategy, we implemented a *best-first search* to compute the optimal solution as fast as optimal. The only problem that may arise due to this choice is that very little nodes can be pruned since there is no good solution known. However, each node requires only the array of binary controls to be stored, which is less than most other algorithms require. The implementation uses BOOST's dynamic_bitset to store the control arrays and an STL priority_queue as the branching tree of open nodes. Since each child node's bound is computed during node creation, the strategy can also be described as full strong branching, where all true bounds are computed as part of the node selection routine.

5 Numerical Results

This chapter covers four different MIOCPs:

1. A prototype MIOCP from the literature; it is a small path-constrained, nonlinear MIOCP.

2. A LOTKA-VOLTERRA type predator-prey model for fish species with a fishing control to stabilize the system.

3. A sewage network where the overflow onto the streets and into the environment has to be controlled during periods of high rainfall.

4. A truck cruise control problem where a truck is driven along a street in an energy-optimal way.

These problems are handled with the methods that have been described in the previous chapters. The different formulations presented in Sections 3.5–3.9 are applied and compared for the truck model and the sewage network model. The behavior of the solution provided through the Control Approximation Problem is studied for the LOTKA-VOLTERRA problem and the nonlinear MIOCP.
An open online benchmark library for the problem class of MIOCPs is available at [151].

5.1 Mixed-Integer Optimal Control with nonlinear ODE

We are interested in the quality of the LAGRANGIAN bound in comparison to the LP bounds obtained from integral relaxation and we want to compare the computational times for the corresponding algorithms. We examine these effects for an MIOCP with a nonlinear ODE system, no constraints inside the disjunction and a limited number of switchings between the different controls. The model is described in Section 5.1.1 and the computational experiments are detailed in Section 5.1.2.
The contents of this section are based on the paper

[104] M. JUNG, G. REINELT AND S. SAGER, *The Lagrangian Relaxation for the Combinatorial Integral Approximation Problem*, Optimization Methods and Software (Submitted).

5.1.1 Problem formulation

We present numerical results for a benchmark MIOCP from a previous study [157] with the addition of switching constraints. In its original form, the problem was:

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{v}} \quad & x_2(t_\mathrm{f}) && (5.1)\\
\text{s.t.} \quad & \dot{x}_0(t) = -\frac{x_0(t)}{\sin(1)}\,\sin(v_1(t)) + \big(x_0(t)+x_1(t)\big)\,v_2^2(t)\\
& \qquad + \big(x_0(t)-x_1(t)\big)\,v_3^3(t),\\
& \dot{x}_1(t) = \big(x_0(t)+2x_1(t)\big)\,v_1(t) + \big(x_0(t)-2x_1(t)\big)\,v_2(t)\\
& \qquad + \big(x_0(t)+x_1(t)\big)\,v_3(t) + \big(x_0(t)\,x_1(t)-x_2(t)\big)\,v_2^2(t)\\
& \qquad - \big(x_0(t)\,x_1(t)-x_2(t)\big)\,v_2^3(t),\\
& \dot{x}_2(t) = x_0^2(t)+x_1^2(t),\\
& x_1(t) \geqslant 0.4,\\
& \boldsymbol{x}(0) = (0.5, 0.5, 0)^T,\\
& \boldsymbol{v}(t) \in \{(1,0,0),(0,1,0),(0,0,1)\},
\end{aligned}
$$

with $t \in [t_0, t_\mathrm{f}] = [0,1]$. In disjunctive formulation, it becomes much more simple and reads with $n_\omega = 3$:

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{v}} \quad & x_2(t_\mathrm{f}) && (5.2)\\
\text{s.t.} \quad & \begin{bmatrix} \omega_1(t)=1 \\ \dot{x}_0(t) = -x_0(t) \\ \dot{x}_1(t) = x_0(t)+2x_1(t) \end{bmatrix} \vee \begin{bmatrix} \omega_2(t)=1 \\ \dot{x}_0(t) = x_0(t)+x_1(t) \\ \dot{x}_1(t) = x_0(t)-2x_1(t) \end{bmatrix}\\
& \qquad \vee \begin{bmatrix} \omega_3(t)=1 \\ \dot{x}_0(t) = x_0(t)-x_1(t) \\ \dot{x}_1(t) = x_0(t)+x_1(t) \end{bmatrix},\\
& \dot{x}_2(t) = x_0^2(t)+x_1^2(t),\\
& \boldsymbol{x}(0) = (0.5, 0.5, 0)^T,\\
& x_1(t) \geqslant 0.4,\\
& \qquad 1 = \sum_{i=1}^{n_\omega} \omega_i(t),\\
& \boldsymbol{\omega}(t) \in \{0,1\}^{n_\omega}.
\end{aligned}
$$

The problem was constructed in [157] such that its disjunctive formulation resembled a problem originally described in [58] and later taken up in [175]. As can be seen in this case, the

original interpolating formulations may be very complicated whereas the formulation only in integral variables is much cleaner. In contrast to the original formulation, we add the additional path constraint $x_1(t) \geq 0.4$, $t \in [t_0, t_f]$ to generate a path-constrained arc in the optimal solution of the relaxed problem, which makes the Control Approximation Problem non-trivial.

The problem's disjunction is reformulated with the OC technique. Then, a control discretization $\boldsymbol{q}$ on an equidistant control grid with size $\Delta t = \frac{1}{m}$ is applied. The states are discretized with the vector $\boldsymbol{s}$ on the same grid with with the *direct multiple shooting* method and an explicit integration scheme $\boldsymbol{\varphi}(t_i, t_{i+1}, \boldsymbol{s_i}, \boldsymbol{q_i})$, which is a discrete representation of the IVP

$$
\begin{aligned}
\dot{x}_0(t) &= -x_0(t)\, q_{1,i} + \big(x_0(t) + x_1(t)\big)\, q_{2,i} + \big(x_0(t) - x_1(t)\big)\, q_{3,i},\\
\dot{x}_1(t) &= \big(x_0(t) + 2x_1(t)\big)\, q_{1,i} + \big(x_0(t) - 2x_i(t)\big)\, q_{2,i} + \big(x_0(t) + x_1(t)\big)\, q_{3,i},\\
\dot{x}_2(t) &= x_0^2(t) + x_1^2(t),\\
\boldsymbol{x}(t_i) &= \boldsymbol{s_i}.
\end{aligned}
$$

After the relaxation of the integrality constraint, we obtain the OCP

$$
\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{q}} \quad & s_{2,m} && && (5.3)\\
\text{s.t.} \quad & \boldsymbol{s_{i+1}} = \boldsymbol{\varphi}\left(t_i, t_{i+1}, \boldsymbol{s_i}, \boldsymbol{q_i}\right), && 0 \leqslant i \leqslant m-1\\
& s_{1,i} \geq 0.4, && 0 \leqslant i \leqslant m,\\
& \boldsymbol{s_0} = (0.5, 0.5, 0)^T,\\
& 1 = \sum_{j=1}^{n_\omega} q_{j,i}, && 0 \leqslant i \leqslant m,\\
& q_{j,i} \in [0,1], && 0 \leqslant i \leqslant m-1, 1 \leqslant j \leqslant n_\omega.
\end{aligned}
$$

5.1.2 Computational experiments

We examine the problem (5.3) with different equidistant control discretizations. The corresponding relaxed OCPs are solved a priori with MUSCOD-II, cf. [117]. It implements BOCK's *direct multiple shooting* method, cf. Section 2.4.2. We use a RUNGE-KUTTA-FEHLBERG 4/5 method, cf. [62], as the explicit integration scheme $\boldsymbol{\varphi}(\cdot)$. For the resulting relaxed controls $\boldsymbol{q}$, we use three different methods to solve the *Control Approximation Problem*

$$
\begin{aligned}
\min_{\theta,\boldsymbol{p}} \quad & \theta && && (5.4)\\
\text{s.t.} \quad & \theta \geqslant \pm \sum_{j=0}^{k} \left(q_{i,j} - p_{i,j}\right) \Delta t_j && 1 \leqslant i \leqslant n_\omega, 0 \leqslant k \leqslant m-1,
\end{aligned}
$$

$$
\begin{aligned}
1 &= \sum_{i=1}^{n_\omega} p_{i,j} && 0 \leqslant j \leqslant m-1, \\
\boldsymbol{p} &\in \{0,1\}^{n_\omega m}.
\end{aligned}
$$

Additionally, for four fixed values of $m \in \{50, 60, 70, 75\}$, the Control Approximation Problem is solved with the addition of switching constraints as described in Section 3.11:

$$
\begin{aligned}
\sigma_j &\geqslant \quad q_{i,j} - q_{i,j-1}, && 1 \leqslant j \leqslant m, 1 \leqslant i \leqslant n_\omega, \\
\sigma_j &\geqslant -q_{i,j} + q_{i,j-1}, && 1 \leqslant j \leqslant m, 1 \leqslant i \leqslant n_\omega, \\
\sigma_{\max} &\geqslant \sum_{j=1}^{m} \sigma_j.
\end{aligned}
\tag{5.5}
$$

The three methods to solve problem (5.4) (with and without switching constraints (5.5)) are Algorithm 4.2, CPLEX© and SCIP. We compare the computational times and nodes visited for solving and additionally the bounds computed during the solution process. Algorithm 4.2 was described in Section 4.5.

CPLEX© *v.12.1.0* is called through AMPL with default options except the following: The search strategy is set to be *traditional branch-and-cut*. The *backtrack* parameter is set to 0.1 to emphasize the best-bound node selection strategy. The *feasibility pump heuristic* and the *relaxation induced neighborhood search heuristic* are disabled. The length of the list of candidates for *strong branching* is increased to 15. CPLEX© is only run on one thread to easily compare times with the other algorithms and to enforce deterministic behavior.

SCIP *v.3.0.0* is also called with default options. In contrast to CPLEX©, this algorithm's parameters are not tuned for better behavior. The implementation is realized with SCIP's C++ interface and SOPLEX *v.1.7.0* is used as its LP solver.

As in [158] for a MIOCP without SOS1-constraint, the computational times of our implementation are orders of magnitude lower than for CPLEX© and SCIP for the Control Approximation Problem related to problem (5.3). To understand how the computational speedup is achieved, we plot a) the number of processed nodes within the branch-and-bound tree and b) the average computation time per node in Figure 5.1 and Figure 5.2 on a logarithmic scale. The average computation time per node in CPLEX© and SCIP are biased by having cutting planes and heuristics, whose times are included in the node computation. However, the given options provide very good overall computational times in comparison to other options. Therefore, they give a reasonable trade-off between the number of nodes and the time spent in those procedures. More specifically, the procedures are usually quite fast in comparison to the LP solving. The reduced number of nodes (due to pruning and better bounds) in most cases outweighs the effort. Additionally, we examine in Figure 5.3 how much worse the bounds of Algorithm 4.2 are in the presence of switching constraints. There, we also show how much the tighter bounds reduce the branching tree's size.

Starting the algorithms in the optimal solution and just proving its optimality would be good to compute no unnecessary problems. However, knowing a very good initial θ allows the cutting procedures of CPLEX© and SCIP to considerably tighten up the problem since θ is present in all the deviation defining constraints – often the algorithms are able to prove optimality in the root node if the optimal solution is provided in advance. Therefore, we decided to not give any initial solution to the algorithms.

Computational results

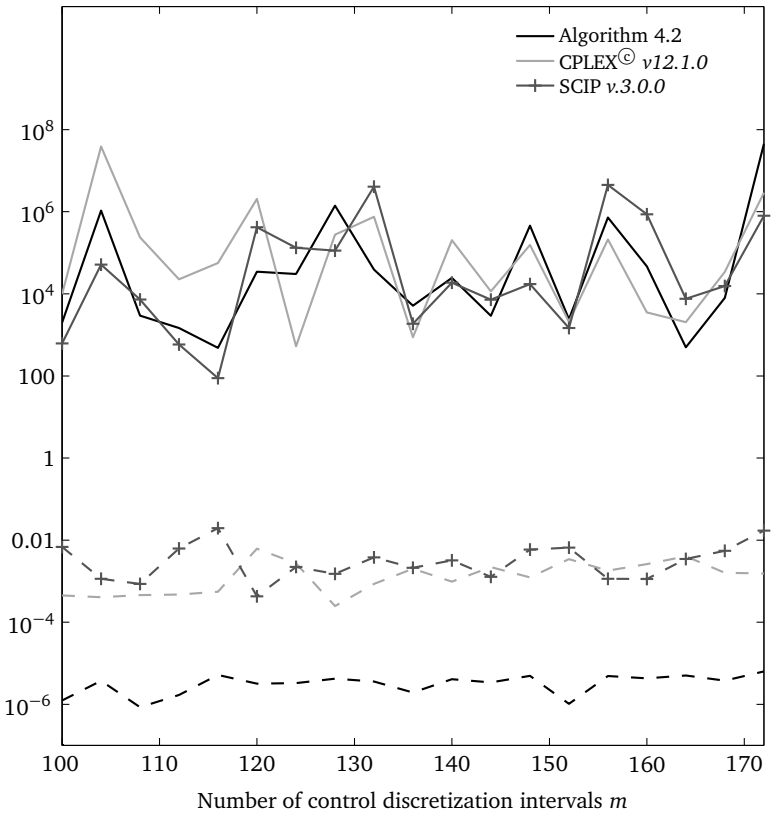


Figure 5.1: Results for different control discretizations of problem (5.4) without switching limits. The number of nodes solved (solid) and the average time per node in seconds (dashed) are displayed.

In Figure 5.1, the number of control discretization intervals is varied in the range between 100 and 172 in steps of 4. The number of switches is not limited in this case. Here, the bound of Algorithm 4.2 is the true LAGRANGIAN relaxation's bound. However, it is easy to show that the LP bounds are the same and not dominated by Algorithm 4.2's bounds, cf.

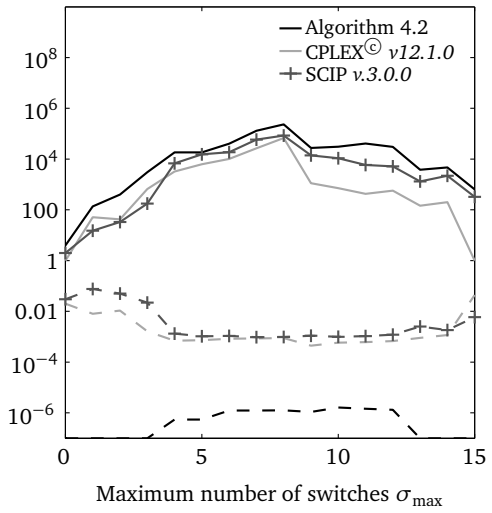
Remark 4.14. The SUR strategy would solve these problems in $\mathcal{O}(mn_\omega)$ but cannot take into account additional constraints as proposed in the next computations. The numerical results allow a couple of interpretations:

- The numbers of branch-and-bound nodes qualitatively behave similar for the three algorithms. Problems that are difficult for one algorithm to solve are generally also difficult for the other algorithms to solve.
- The average computational times per node are almost constant and the effort needed for the LAGRANGIAN relaxation's bounds is two to three orders of magnitudes lower than for solving an LP.
- The effort is not monotone in the problem size. This behavior is connected to the objective values of the approximation. They do not necessarily get better for finer grids if the new grid is not a subdivision of the old grid, which it is if and only if the new grid size is a multiple of the old one. As a rule of thumb, the lower the optimal objective, the smaller the tree that has to be searched. We give a possible explanation for this behavior in the next observations.

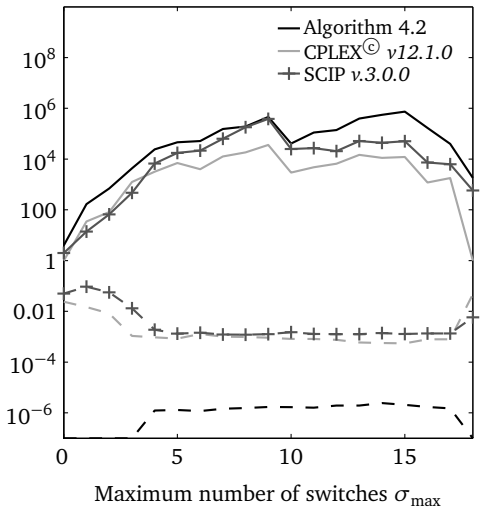
In Figure 5.2, four different control discretization grid sizes are kept constant, while the maximum number of switches $\sigma_{\max}$ is varied. Again, we make some observations:

- CPLEX© solves the problems with $\sigma_{\max} = 0$ and some of the bigger values for $\sigma_{\max}$ in a preprocessing step or the root node.
- Again, qualitatively the curves of the number of branch and bound nodes, which need to be solved, are similar, and the average computational costs are almost constant. The higher computational effort spent in the root node visibly increases the average effort per node for instances that only need to solve small numbers of nodes.
- Since the bounding procedure of Algorithm 4.2 ignores the additional constraints, the LP bounds are better in this case, which can be seen looking at the higher number of nodes needed for Algorithm 4.2 in comparison to the LP-based algorithms. There are between one and two orders of magnitude of additional nodes needed. CPLEX©'s computational times come within the same order of magnitude for the easy instances, which can be solved faster than one or two seconds. However, for the more difficult instances, the computational times of Algorithm 4.2 are still two to three orders of magnitude lower.
- As a function of $\sigma_{\max}$, the number of processed B&B nodes has its maximum in the middle between 0 and the number of switches the SUR strategy would take. As already briefly discussed in Section 4.5.4, combinatorial constraints are most difficult if they are neither too severe nor too light. Too severe restrictions allow the branch and bound

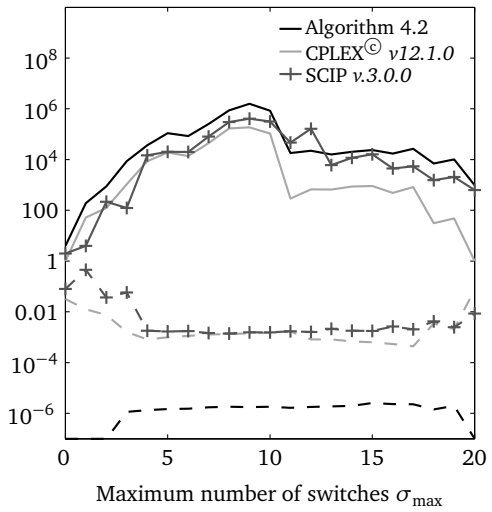
(a) B&B nodes and time per node for $m = 50$:



(b) B&B nodes and time per node for $m = 60$:



(c) B&B nodes and time per node for $m = 70$:



(d) B&B nodes and time per node for $m = 75$:

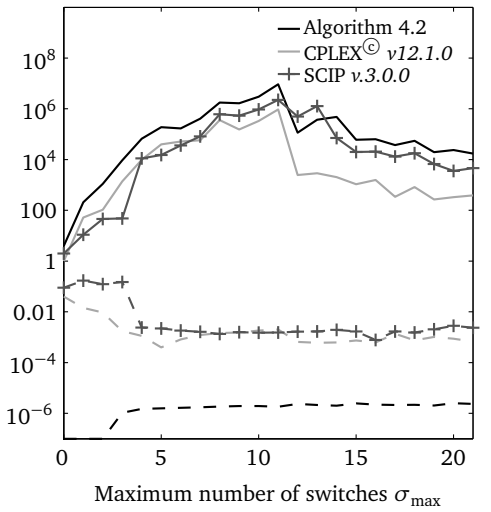


Figure 5.2: Results for different control discretizations of problem (5.4) with varying switching constraints (5.5). The number of solved nodes (solid) and the average time per node in seconds (dashed) are displayed.

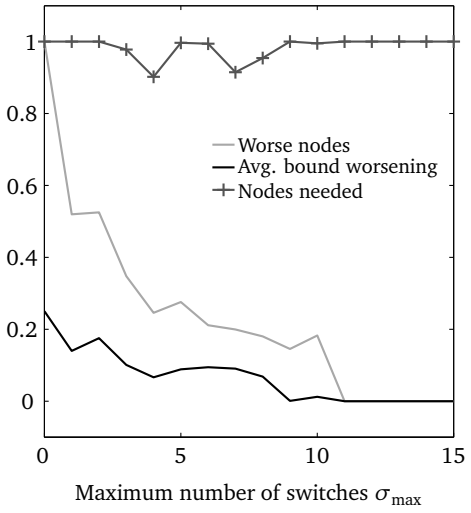
algorithm to cut off a major part of its tree. And for very light restrictions, the bound of the relaxations are very good and close to the optimum.

- The size of the B&B tree varies strongly in some cases, when $\sigma_{\max}$ is increased. It seems to be the case that this is related to changes of the optimal objective value θ of (5.4). If θ has a value slightly above a multiple of Δt and is reduced below that value with more switches allowed, the number of nodes, which have to be processed, significantly decreases. This is because there are many nodes with a bound of multiples of Δt. They are created when one or more bad decision are made higher up in the tree, and afterwards all decisions for the controls are made such that the min-max term is dominated by the early deviation. Apparently, there is a large number of nodes that does not violate this bound and cannot be pruned. For the initial parts, until the hardest problem is reached, the two algorithms behave quite similar. After the hardest problem, there is usually one sharp decrease as explained above and then a steady decrease until the solution stays the same for all values of $\sigma_{\max}$. At this point the switching constraint is no true constraint anymore, because the SUR solution already fulfills the switching constraint.

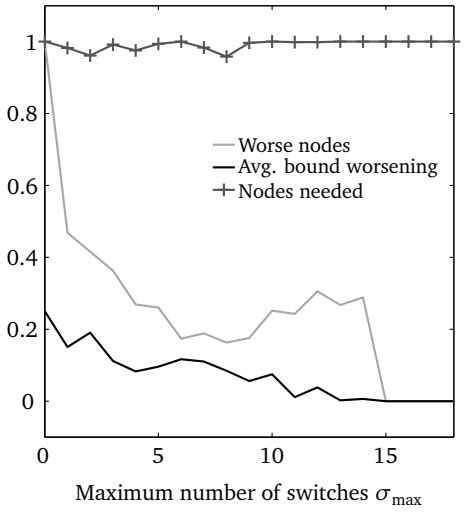
To further distinguish the effect of the different bounding procedures from the effect of the different branching schemes and node selection strategies, we implemented in Algorithm 4.2 also a procedure to compute the standard LP relaxation's bound for each node, but no additional cutting plane techniques to tighten this bound. Nodes, which could be cut off due to the LAGRANGIAN bound, are cut off before their LP relaxation is computed. For this setting, we illustrate the ratio of visited nodes whose bound of Algorithm 4.2 is worse than the bound given by the LP relaxation. We also give the average worsening of the LAGRANGIAN bound in comparison to the LP bound as a ratio over the worse nodes. In addition, we display the ratio of nodes that still need to be solve when using the LP relaxation instead of the worse LAGRANGIAN relaxation in this framework. Naturally, the computational times strongly increase in comparison to Algorithm 4.2: the worst instance without LP bounds takes 19.1 seconds, while the worst instance with those bounds takes approximately 5000 seconds. The results are displayed in Figure 5.3. Some observations can be made from these results:

- As the number of switches increases, the two bounds become closer since the constraints, which are only considered for the LP bounds, become less tight.
- The average worsening takes the value 1 as long as the LP bound is non-trivial while the LAGRANGIAN bound still is, which only happens at the highest nodes. In the case of $\sigma_{\max} = 0$, the number of nodes is very small and only the root nodes' bounds differ.
- Even if the bounds are stronger for the LP relaxation, this does not necessarily lead to a reduced number of processed nodes, as the bounds might not be strong enough for pruning.

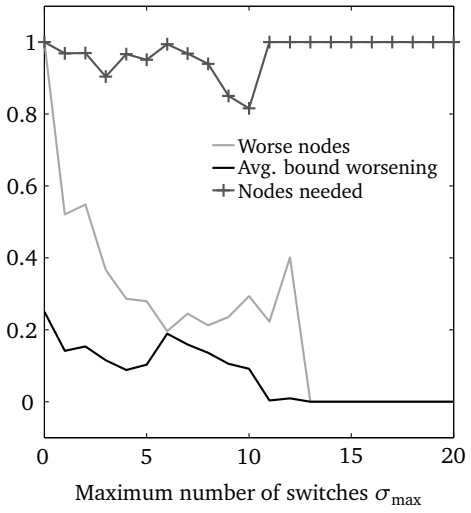
(a) Bound comparison for $m = 50$:



(b) Bound comparison for $m = 60$:



(c) Bound comparison for $m = 70$:



(d) Bound comparison for $m = 75$:

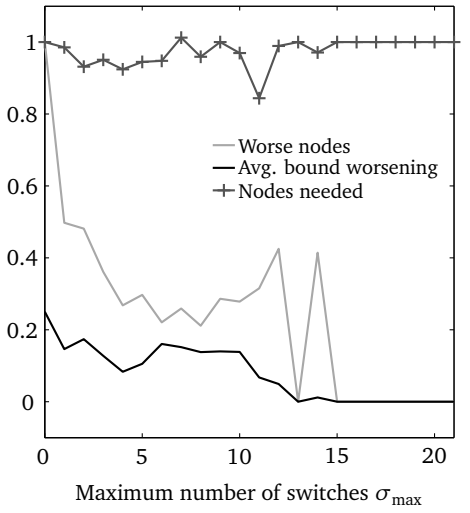


Figure 5.3: Bound comparison for different control discretizations of problem (5.4) with varying switching constraints (5.5). The LAGRANGIAN relaxation's bounds Z_{LR} from Algorithm 4.2 are compared with the LP bounds Z_{LP} in the same branching framework.
"Nodes needed" displays the percentage of nodes still needed when using the worse LAGRANGIAN bound instead of the LP bound.
"Worse nodes" displays the percentage of nodes visited with $Z_{\mathrm{LR}} < Z_{\mathrm{LP}}$.
"Avg. bound worsening" displays the average of the relative bound worsening $\frac{Z_{\mathrm{LP}} - Z_{\mathrm{LR}}}{Z_{\mathrm{LP}}}$ taken over all worse nodes.

- The size of the branching tree can only be reduced by maximally 20%, when the standard LP relaxation is used instead of the LAGRANGIAN relaxation, with a mean reduction of 2.13%. This does not provide a significant decrease to warrant the additional effort to compute the bound. The worse nodes are usually located in the higher parts of the tree, close to the root node, and the values of the bounds become closer the deeper the nodes are in the tree. Nodes high up in the tree rarely have bounds tight enough for pruning and hence the effect of those better bounds close to the root is weak.
- The additional reduction of tree size in the CPLEX© and SCIP results must be originated in the the different branching strategies and in the additionally implemented procedures that tighten the bounds, e.g. the addition of cutting planes.

Summing up, the improvement of the combinatorial branch-and-bound approach is due to an almost constant factor of two to three orders of magnitude going back to a faster evaluation of the relaxations. Although the LAGRANGIAN relaxation's bound can be proven to be the same as the LP relaxation's in the unconstrained problem, this does not hold for the constrained case. Therefore, any algorithm that uses only the LAGRANGIAN bound might need to visit more nodes to solve a problem. However, this additional effort is easily accounted for by the relaxations' speedup.

5.1.3 Conclusions

We analyzed the behaviors of the LAGRANGIAN relaxation of the unconstrained Control Approximation Problem and of the LP relaxation of the constrained Control Approximation Problem. We showed that the LAGRANGIAN bound is helpful for the Control Approximation Problem, although it is unconventional in the sense that it is used inside a branch-and-bound scheme, and not used as a reformulation of the whole problem.
We showed that the special SOS1 branching strategy yields good results in practice. The corresponding theoretical analysis shows that this is the case because it leads to LAGRANGIAN relaxation bounds without a pre-factor $\frac{1}{2}$, cf. Theorem 4.3 and the special case of Lemma 4.6. The bounds used within Algorithm 4.2 use the exact values of the LAGRANGIAN relaxation when no additional constraints are present.
It is shown that even though the LAGRANGIAN bounds may be worse than the bounds obtained through an LP relaxation, the faster computation for each node more than outweighs the worsening in the bound quality for a reasonably sized example. It can be observed that the behavior of Algorithm 4.2 is very competitive even after adding further combinatorial constraints.

5.2 LOTKA-VOLTERRA fishing problem

The contents of this section are based on the paper

[158] S. SAGER, M. JUNG AND C. KIRCHES, *Combinatorial Integral Approximation*, Mathematical Methods for Operations Research, 2011, Vol. 73(3):363–380.

In this section, we present a predator-prey model of the LOTKA-VOLTERRA type with a control specifying hunting or fishing activities. The model can be found in the MIOCP benchmark library [151]. For this scenario, the controls are required to have a maximum number of switches. The resulting problem is treated with the methods described in Chapter 3 to obtain a valid integral model and its relaxation. The resulting problem is particularly suited for our study, because the optimal relaxed solution contains a singular arc, cf. [152]. Then, the newly developed combinatorial branch-and-bound Algorithm 4.2 is applied to the results of the relaxation and compared to a solution obtained by a MINLP solver, which tries to solve the discretized MIOCP directly.

5.2.1 The predator-prey model and the balancing task

The simple predator-prey model of the LOTKA-VOLTERRA type describes a model with two species: the prey with its biomass x_0 and the predator with its biomass x_1. The LOTKA-VOLTERRA model is based on three assumptions:

1. The prey population has sufficient amounts of nutrition and can thus reproduce at a constant rate g_0.

2. The predator species only eats the prey species and thus its survival is entirely based on the prey population, its mortality rate in biomass without finding food is constant g_1.

3. There is no evolution and the environment stays the same.

If predator and prey meet, the prey species' biomass declines by α_0 units and the predator species' biomass increases by $\alpha_1 \leqslant \alpha_0$. The population dynamics are then governed by the ODEs:

$$\begin{aligned} \dot{x}_0(t) &= \quad g_0\, x_0(t) - \alpha_0\, x_0(t)\, x_1(t), \\ \dot{x}_1(t) &= -g_1\, x_1(t) + \alpha_1\, x_0(t)\, x_1(t). \end{aligned}$$

This model typically produces a periodically oscillating behavior for both species.
For this example, both species are assumed to be fish and men now interferes with the dynamic system through fishing. Generally, this could be modeled through the following dis-

Name	Description	Domain
x_0	Biomass of prey species	$\mathbb{R}_0^+$
x_1	Biomass of predator species	$\mathbb{R}_0^+$
x_2	Squared $\mathcal{L}^2$-deviation from reference states	$\mathbb{R}_0^+$
ω	Fishing activity	$\{0,1\}$

Table 5.1: States and controls of the LOTKA-VOLTERRA model.

junction

$$\begin{bmatrix} \omega(t) = 0 \\ \dot{x}_0(t) = \quad g_0\, x_0(t) - \alpha_0\, x_0(t)\, x_1(t) \\ \dot{x}_1(t) = -g_1\, x_1(t) + \alpha_1\, x_0(t)\, x_1(t) \end{bmatrix} \vee \begin{bmatrix} \omega(t) = 1 \\ \dot{x}_0(t) = \quad g_0\, x_0(t) - \alpha_0\, x_0(t)\, x_1(t) - \beta_0\, x_0(t) \\ \dot{x}_1(t) = -g_1\, x_1(t) + \alpha_1\, x_0(t)\, x_1(t) - \beta_1\, x_1(t) \end{bmatrix},$$

where $\omega(t) \in \{0,1\}$ represents whether fishing is done or not and β_0 and β_1 are the rates with which the corresponding species are caught depending on the nets used, the fish sizes etc.

Since there are no additional path constraints in the disjunctions, only the ODEs, we directly choose the OC technique from Section 3.6 to represent the disjunction. This gives us a tighter relaxation than with the perspective formulation and creates a well-behaving IVP. The resulting ODEs are:

$$\begin{aligned} \dot{x}_0(t) &= \quad g_0\, x_0(t) - \alpha_0\, x_0(t)\, x_1(t) - \beta_0\, x_0(t)\, \omega(t), \\ \dot{x}_1(t) &= -g_1\, x_1(t) + \alpha_1\, x_0(t)\, x_1(t) - \beta_1\, x_1(t)\, \omega(t), \\ \omega(t) &\in \{0,1\}. \end{aligned}$$

Additionally, this formulation is nice, since it can be interpreted as the fisher interacting the same way with both species as they interact among themselves. Fishing is done in this case to drive the system towards a reference state $(x_0^{\text{ref}}, x_1^{\text{ref}})$. This is e.g. achieved through the following LAGRANGE type functional that measures the square of the $\mathcal{L}^2$-norm of the deviation:

$$\Phi_{\text{L}}(\boldsymbol{x}(\cdot)) = \int_{t_0}^{t_{\text{f}}} \left(x_0(t) - x_0^{\text{ref}}\right)^2 + \left(x_1(t) - x_1^{\text{ref}}\right)^2 \, \mathrm{d}t.$$

The objective is transformed through the addition of an artificial state x_2 that accumulates the deviation from the reference state:

$$\dot{x}_2(t) = \left(x_0(t) - x_0^{\text{ref}}\right)^2 + \left(x_1(t) - x_1^{\text{ref}}\right)^2$$

and the corresponding MAYER objective

$$\min x_2(t_\text{f}).$$

Name	Description
g_0	Reproduction rate of prey species
g_1	Mortality rate of predator species
α_0	Predation rate when predator meets prey
α_1	Predator reproduction rate when predator meets prey
β_0	Fishing rate of prey species
β_1	Fishing rate of predator species
$x_{0,0}$	Initial prey population
$x_{1,0}$	Initial predator population
x_0^{ref}	Desired prey population
x_1^{ref}	Desired predator population

Table 5.2: Parameters of the LOTKA-VOLTERRA model.

Discretization and limited switching model

We introduce discretized states $\boldsymbol{s}$ on the time grid

$$\mathbb{G}_{k\,m} \overset{\text{def}}{=} \left\{t_0 < t_1 < \ldots < t_{k\,m} = t_\text{f}\right\},$$

and a discretized control $\boldsymbol{q}$ on the sub-grid $\tilde{\mathbb{G}}_m$ with $\tilde{t}_i = t_{k\,i}$ such that it holds with a certain integration scheme $\boldsymbol{\varphi}$ for the right-hand side:

$$\begin{aligned}
\omega(t) &= q_i, && t \in [\tilde{t}_i, \tilde{t}_{i+1}),\ 0 \leqslant i \leqslant m-1,\\
\boldsymbol{x}(t_i) &= \boldsymbol{s_i}, && 0 \leqslant i \leqslant k\,m,\\
\boldsymbol{s_{i+1}} &= \boldsymbol{s_i} + \boldsymbol{\varphi}(t_i, t_{i+1}, \boldsymbol{s_i}, \boldsymbol{s_{i+1}}, q_j), && k\,j \leqslant i \leqslant k\,(j+1)-1,\ 0 \leqslant j \leqslant m-1.
\end{aligned}$$

Since the system is very stable, we use the explicit EULER method as the integration scheme φ.

Due to economic reasons, we limit the number of fishing operations and we limit the number of system switches, which are directly connected with each other. The system switches are modeled with the different means described in Section 3.11.

First, we use unlimited switches for the relaxation:

$$
\begin{aligned}
\min_{\boldsymbol{s},\boldsymbol{q},\boldsymbol{\sigma}} \quad & s_{2,m} && && (5.6)\\
\text{s.t.} \quad & s_{0,i+1} = s_{0,i} + (t_{i+1} - t_i) \left(\quad g_0\, s_{0,i} - \alpha_0\, s_{0,i}\, s_{1,i} - \beta_0\, s_{0,i}\, q_j \right), \\
& s_{1,i+1} = s_{0,i} + (t_{i+1} - t_i) \left(-g_1\, s_{1,i} + \alpha_1\, s_{0,i}\, s_{1,i} - \beta_1\, s_{1,i}\, q_j \right), \\
& s_{2,i+1} = s_{2,i} + (t_{i+1} - t_i) \left(\left(s_{0,i} - x_0^{\text{ref}} \right)^2 + \left(s_{1,i} - x_1^{\text{ref}} \right)^2 \right), \\
& \qquad\qquad k\,j \leqslant i \leqslant k\,(j+1) - 1,\ 0 \leqslant j \leqslant m-1, \\
& \quad q_j \in [0,1], \qquad\qquad 0 \leqslant j \leqslant m-1, \\
& \quad s_{0,0} = x_{0,0}, \\
& \quad s_{1,0} = x_{1,0}, \\
& \quad s_{2,0} = 0.
\end{aligned}
$$

Second, we apply the linear switching reformulation through the addition of the variables $\boldsymbol{\sigma} \in \{0,1\}^{m-1}$ defined through:

$$\sigma_j \geq \pm \left(q_j - q_{j-1} \right), \qquad 1 \leqslant j \leqslant m-1. \tag{5.7}$$

The switching limit is given by

$$\sigma_{\max} \geqslant \sum_{j=1}^{m-1} \sigma_j. \tag{5.8}$$

The third formulation additionally introduces the auxiliary variables $\alpha \in \{0,1\}^{m-1}$ to use the convex combination of the switch defining hyperplanes as described in Section 3.11 instead of (5.7):

$$\sigma_j = \alpha_j \left(q_j + q_{j-1} \right) + \left(1 - \alpha_j \right) \left(2 - q_j - q_{j-1} \right), \qquad 1 \leqslant j \leqslant m-1. \tag{5.9}$$

5.2.2 Computational experiments

The scenario, the population dynamics and parameters are used as described in [151], which is:

$$g_0 = 1, \quad \alpha_0 = 1, \quad \beta_0 = 0.4, \quad x_{0,0} = 0.5, \quad x_0^{\text{ref}} = 1, \quad t_0 = 0,$$
$$g_1 = 1, \quad \alpha_1 = 1, \quad \beta_1 = 0.2, \quad x_{1,0} = 0.7, \quad x_1^{\text{ref}} = 1, \quad t_{\text{f}} = 12.$$

The control ω has been discretized with varying values of $m \in \{10, 25, 50, 80, 100, 200\}$ with piecewise constant basis functions. The differential equations have been discretized with an explicit EULER method and 10000 equidistant time steps, independent of the control discretization.
We first solve the discretized, relaxed OCPs (5.6) with IPOPT *v.3.11* using the HSL linear algebra solvers. Then, for the resulting relaxed controls, the *Control Approximation Problem* with switching constraints (5.7) and (5.8) is solved, once with CPLEX© *v.12.5.0* and once with Algorithm 4.2. These results are compared with the true optimal solutions obtained by directly solving the discretized MIOCPs with BONMIN *v.1.7* (internally working with CBC *v.2.7* and the aforementioned IPOPT) denoted by $\cdot^{\text{opt}}$.

Computational results

In the following, Φ_{L} is the objective function value of the OCP, i.e., the $\mathcal{L}^2$-deviation of the reference states, τ represents computational times, θ is the objective value of the *Control Approximation Problem* and σ gives the number of control switches used in the corresponding solutions. The calculations of the reference solutions were limited to half an hour of computational time and if they were not finished, the placeholder $*$ is used.

m	10	25	50	100	125	200
τ^{rel}	2.452	3.520	2.900	3.248	2.976	2.520
$\Phi_{\text{L}}^{\text{rel}}$	1.34915	1.34718	1.34683	1.34649	1.34640	1.34626
$\Phi_{\text{L}}^{\text{SUR}}$	1.60251	1.37175	1.38366	1.35561	1.35396	1.35328
Φ_{gap}	18.8%	1.8%	2.7%	0.7%	0.6%	0.5%
θ^{SUR}	0.37984	0.23650	0.11993	0.05973	0.04792	0.02997
$\frac{1}{\Delta t}\theta^{\text{SUR}}$	0.31653	0.49270	0.49971	0.49777	0.49913	0.49956
σ^{SUR}	2	4	8	14	18	24

Table 5.3: SUR results for LOTKA-VOLTERRA fishing problem.

The realization of some theoretical results in practice are shown in Table 5.3. We compare the solution of the integrally relaxed OCP ($\boldsymbol{q} \in [0,1]^m$) with the solution obtained through SUR,

cf. Section 4.3.1. One can observe that the approximation value is limited by $\theta^{\mathrm{SUR}} \leqslant \frac{1}{2}\Delta t$, cf. Proposition 4.1, and that it hence decreases linearly with the grid size. It is also interesting to observe that these values approach $\frac{1}{2}\Delta t$ as m increases due to the increased probability to find a maximum close to the upper bound $\frac{1}{2}$. It is also possible to observe that – with grid refining – the integrality gap

$$\Phi_{\mathrm{gap}} \stackrel{\mathrm{def}}{=} \frac{\Phi_{\mathrm{L}}^{\mathrm{SUR}} - \Phi_{\mathrm{L}}^{\mathrm{rel}}}{\Phi_{\mathrm{L}}^{\mathrm{rel}}}$$

between the relaxed solution and the SUR solution becomes smaller and smaller. It can also be observed that the SUR solution becomes more and more chattering, i.e., it employs highly frequent switching, the finer the grid.

Since the chattering behavior is often undesirable, additional switching constraints are added to the problem. They can be added at several different points and we will discuss some of these settings in the following. First, the switching constraints are added to the Control Approximation Problem and the solution is compared with the solution of the true MINLP discretization with the same switching limit (obtained with BONMIN *v.1.7.0* and either one of the two switching formulations). In Tables 5.4 and 5.5, we compare two algorithms for solving the Control Approximation Problem with respect to computational times, i.e., the tailored branch-and-bound Algorithm 4.2 with switching constraint handling Algorithm 4.3 is compared with CPLEX© *v.12.5.0* (run in deterministic mode on one single core). It can immediately be seen that the tailored branch-and-bound is about two orders of magnitude faster for the difficult instances while finishing immediately for the easy instances. For switching-constrained problems, the control deviation cannot be driven to 0 for finer discretizations, but is bounded strictly away from 0. The less tight the switching restriction – with more switches allowed – the closer the controls can be approximated and the better the solutions of the procedure become. The objective function values Φ^{CAP} will converge against the value of Φ^{rel}, as $m \to \infty$ and $\sigma_{\max}$ large enough. It can also be seen that the solution of the Control Approximation Problem is generally close to the optimum solution for this state tracking or stabilization problem. Also notice that the complete problem is really difficult to solve, with BONMIN being unable to solve a lot of the bigger problems ($m \geqslant 100$), while the decomposed problem can usually be solved within a couple of seconds.

Second, the switching limit could already be employed inside the relaxation, whose solution is then to be approximated with the Control Approximation Problem. For this problem, the linear switching limit formulation (5.7) and (5.8) is loose enough for the original relaxation to already satisfy it. Therefore, in Tables 5.6 and 5.7, only the effects of adding the nonlinear switching constraints (5.8) and (5.9) to the relaxed problem are examined. Due to the non-convex nature of the switching description, using IPOPT to solve the relaxation sometimes leads to problems with local infeasibility, i.e., a local minimizer of the constraint violation is reached and the algorithm cannot continue. Instances where the infeasibility problems

occurred are crossed out with †. The non-convexity also leads to problems of only locally optimal solutions with IPOPT, which would cut off the global optimum when taken as lower bound, the instances where this happens are marked with ⋄ at the bound. The relaxation produces results that are much easier approximated with the Control Approximation Problem. For most instances, the normalized control deviation is lower than $\frac{1}{2}\Delta t$ or at least close to this value. This means that the SUR is either feasible or produces an almost feasible solution. However, another strong disadvantage of this formulation is the computational effort needed to at least locally solve the relaxation. It is one to two orders of magnitude higher than solving the relaxation without the nonlinear switching formulation.

5.2.3 Conclusions

The effects of the theoretical results of Chapter 4 could be verified in practice. These take into account the SUR strategy as well as the quality of the Control Approximation Problem's solution. Although this solution is not necessarily optimal for the MIOCP, a feasible solution is provided. This solution converges to the solution of the nonlinear MIOCP if the switching constraint does not become active as the time discretization is refined. If the switching constraint is active, knowledge of system properties – such as the LIPSCHITZ constant of the right-hand side function of the ODE system – allows to formulate an upper bound on the state deviation between the states induced by the Control Approximation Problem's solution and states of the solution of the relaxed OCP. This upper bound depends linearly on the objective function value of the Control Approximation Problem as specified in Theorem 4.1.
The nonlinear switching formulation is shown to provide favorable properties with regard to provided solutions since it attracts integrality. However, it is very difficult to reliably get the relaxation's solution in comparable computational time. The approach of using soft-constraints for the switches, as in [111], instead of hard-constraints, as done here, seems advantageous with regard to the computational effort. However, a problem-dependent reasonable setting for the penalty parameter is needed.

m	10	25	50	100	125	200
Φ_L^{rel}	1.34915	1.34718	1.34683	1.34649	1.34640	1.34626
Maximum of $\sigma_{max} = 3$ switches:						
τ^{bb}	0.000	0.000	0.000	0.000	0.000	0.020
τ^{cpx}	0.008	0.036	0.116	0.596	1.096	10.825
Φ_L^{CAP}	1.60251	1.52323	1.67052	1.55515	1.54225	1.60619
θ^{CAP}	0.37984	0.38772	0.23298	0.22267	0.23064	0.20979
σ^{CAP}	2	3	2	3	2	2
τ^{opt}	54.659	245.655	620.663	863.730	788.045	*
Φ_L^{opt}	1.60251	1.52323	1.38746	1.38741	1.38676	*
σ^{opt}	2	3	2	3	3	*
Maximum of $\sigma_{max} = 4$ switches:						
τ^{bb}	0.000	0.000	0.000	0.000	0.000	0.030
τ^{cpx}	0.012	0.020	0.068	0.520	1.088	4.840
Φ_L^{CAP}	1.60251	1.37175	1.36718	1.42269	1.40501	1.40632
θ^{CAP}	0.37984	0.23650	0.16121	0.14089	0.14744	0.11924
σ^{CAP}	2	4	4	4	4	4
τ^{opt}	54.975	156.870	813.967	62.836	64.108	68.208
Φ_L^{opt}	1.60251	1.37175	1.35883	1.35661	1.35798	1.36555
σ^{opt}	2	4	4	4	4	4
Maximum of $\sigma_{max} = 5$ switches:						
τ^{bb}	0.000	0.000	0.000	0.000	0.020	0.140
τ^{cpx}	0.012	0.020	0.052	2.080	5.340	17.609
Φ_L^{CAP}	1.60251	1.37175	1.36718	1.39684	1.38364	1.40632
θ^{CAP}	0.37984	0.23650	0.16121	0.14089	0.14663	0.11924
σ^{CAP}	2	4	4	4	5	4
τ^{opt}	52.535	150.005	519.820	1789.660	*	*
Φ_L^{opt}	1.60251	1.37175	1.35883	1.35661	*	*
σ^{opt}	2	4	4	4	*	*

Table 5.4: Control approximation results for LOTKA-VOLTERRA fishing problem – part 1.

m	10	25	50	100	125	200
Φ_L^{rel}	1.34915	1.34718	1.34683	1.34649	1.34640	1.34626
Maximum of $\sigma_{max} = 6$ switches:						
τ^{bb}	0.000	0.000	0.000	0.000	0.020	0.360
τ^{cpx}	0.012	0.016	0.036	0.660	3.404	17.029
Φ_L^{CAP}	1.60251	1.37175	1.36540	1.35484	1.38803	1.40313
θ^{CAP}	0.37984	0.23650	0.12127	0.10344	0.10183	0.09021
σ^{CAP}	2	4	6	6	6	6
τ^{opt}	50.767	150.005	519.820	*	1317.890	*
Φ_L^{opt}	1.60251	1.37175	1.35233	*	1.35175	*
σ^{opt}	2	4	6	*	6	*
Maximum of $\sigma_{max} = 7$ switches:						
τ^{bb}	0.000	0.000	0.000	0.000	0.040	1.060
τ^{cpx}	0.008	0.020	0.036	3.536	4.680	90.698
Φ_L^{CAP}	1.60251	1.37175	1.36533	1.39789	1.38547	1.39883
θ^{CAP}	0.37984	0.23650	0.12086	0.10300	0.10183	0.09021
σ^{CAP}	2	4	7	7	6	6
τ^{opt}	50.699	151.821	531.173	*	*	*
Φ_L^{opt}	1.60251	1.37175	1.35233	*	*	*
σ^{opt}	2	4	6	*	*	*
Maximum of $\sigma_{max} = 8$ switches:						
τ^{bb}	0.000	0.000	0.000	0.000	0.000	0.500
τ^{cpx}	0.012	0.020	0.036	0.788	6.096	61.331
Φ_L^{CAP}	1.60251	1.37175	1.38366	1.37767	1.38275	1.35198
θ^{CAP}	0.37984	0.23650	0.11993	0.08709	0.07574	0.07436
σ^{CAP}	2	4	8	8	8	8
τ^{opt}	51.091	152.638	540.806	*	*	*
Φ_L^{opt}	1.60251	1.37175	1.35233	*	*	*
σ^{opt}	2	4	6	*	*	*

Table 5.5: Control approximation results for LOTKA-VOLTERRA fishing problem – part 2.

m	10	25	50	100	125	200
Maximum of $\sigma_{\max} = 3$ switches:						
τ^{rel}	6.748	16.297	100.702	150.533	234.491	†
$\Phi_{\text{L}}^{\text{rel}}$	1.34915	1.35413	1.38715	⋄1.47415	⋄1.41862	†
$\Phi_{\text{L}}^{\text{CAP}}$	1.60251	1.53360	1.39174	1.47415	1.41882	†
θ^{CAP}	0.37984	0.12476	0.07485	0.00000	0.00942	†
$\frac{1}{\Delta t}\theta^{\text{CAP}}$	0.31653	0.25991	0.31189	0.00000	0.09813	†
σ^{CAP}	2	3	3	3	2	†
τ^{opt}	54.659	245.655	620.663	863.730	788.045	*
$\Phi_{\text{L}}^{\text{opt}}$	1.60251	1.52323	1.38746	1.38741	1.38676	*
σ^{opt}	2	3	2	3	3	*
Maximum of $\sigma_{\max} = 4$ switches:						
τ^{rel}	6.696	41.779	42.715	†	†	†
$\Phi_{\text{L}}^{\text{rel}}$	1.34915	1.34724	⋄1.36447	†	†	†
$\Phi_{\text{L}}^{\text{CAP}}$	1.60251	1.42241	1.52981	†	†	†
θ^{CAP}	0.37984	0.23358	0.15021	†	†	†
$\frac{1}{\Delta t}\theta^{\text{CAP}}$	0.31653	0.48662	0.62589	†	†	†
σ^{CAP}	2	4	3	†	†	†
τ^{opt}	54.975	156.870	813.967	62.836	64.108	68.208
$\Phi_{\text{L}}^{\text{opt}}$	1.60251	1.37175	1.35883	1.35661	1.35798	1.36555
σ^{opt}	2	4	4	4	4	4
Maximum of $\sigma_{\max} = 5$ switches:						
τ^{rel}	6.612	11.181	93.910	†	†	226.038
$\Phi_{\text{L}}^{\text{rel}}$	1.34915	1.34718	1.34716	†	†	1.35771
$\Phi_{\text{L}}^{\text{CAP}}$	1.60251	1.37175	1.35428	†	†	1.36520
θ^{CAP}	0.37984	0.23650	0.16121	†	†	0.11924
$\frac{1}{\Delta t}\theta^{\text{CAP}}$	0.31653	0.49270	0.51914	†	†	0.28959
σ^{CAP}	2	4	4	†	†	5
τ^{opt}	52.535	150.005	519.820	1789.660	*	*
$\Phi_{\text{L}}^{\text{opt}}$	1.60251	1.37175	1.35883	1.35661	*	*
σ^{opt}	2	4	4	4	*	*

Table 5.6: Results for nonlinear switch description of LOTKA-VOLTERRA fishing problem – part 1.

m	10	25	50	200
Maximum of $\sigma_{\max} = 6$ switches:				
τ^{rel}	6.524	10.553	45.655	143.841
$\Phi_{\text{L}}^{\text{rel}}$	1.34915	1.34718	1.34716	1.35771
$\Phi_{\text{L}}^{\text{CAP}}$	1.60251	1.37175	1.36540	1.35484
θ^{CAP}	0.37984	0.23650	0.12127	0.10344
$\frac{1}{\Delta t}\theta^{\text{CAP}}$	0.31653	0.49270	0.48570	0.56295
σ^{CAP}	2	4	6	6
τ^{opt}	50.767	150.005	519.820	*
$\Phi_{\text{L}}^{\text{opt}}$	1.60251	1.37175	1.35233	*
σ^{opt}	2	4	6	*
Maximum of $\sigma_{\max} = 7$ switches:				
τ^{rel}	6.672	9.529	33.266	†
$\Phi_{\text{L}}^{\text{rel}}$	1.34915	1.34718	1.34683	†
$\Phi_{\text{L}}^{\text{CAP}}$	1.60251	1.37175	1.36533	†
θ^{CAP}	0.37984	0.23650	0.12086	†
$\frac{1}{\Delta t}\theta^{\text{CAP}}$	0.31653	0.49270	0.50029	†
σ^{CAP}	2	4	7	†
τ^{opt}	50.699	151.821	531.173	*
$\Phi_{\text{L}}^{\text{opt}}$	1.60251	1.37175	1.35233	*
σ^{opt}	2	4	6	*
Maximum of $\sigma_{\max} = 8$ switches:				
τ^{rel}	6.660	9.769	40.423	†
$\Phi_{\text{L}}^{\text{rel}}$	1.34915	1.34718	1.34683	†
$\Phi_{\text{L}}^{\text{CAP}}$	1.60251	1.37175	1.38366	†
θ^{CAP}	0.37984	0.23650	0.11993	†
$\frac{1}{\Delta t}\theta^{\text{CAP}}$	0.31653	0.49270	0.49978	†
σ^{CAP}	2	4	8	†
τ^{opt}	51.091	152.638	540.806	*
$\Phi_{\text{L}}^{\text{opt}}$	1.60251	1.37175	1.35233	*
σ^{opt}	2	4	6	*

Table 5.7: Results for nonlinear switch description of LOTKA-VOLTERRA fishing problem – part 2.

5.3 Sewage network overflow

The contents of this section are based on the paper

[56] B.J. DURAN, M. JUNG, C. OCAMPO-MARTINEZ, S. SAGER AND G. CEMBRANO, *Minimization of Sewage Network Overflow*, Water Resources Management (Accepted).

In this section, we present the optimal control of sewage networks. In certain locations, it is of high public interest to minimize the overflow of sewage onto the streets and to the natural environment that may occur during periods of heavy rain. The assumption of linear flow in a discrete time setting was proven to be adequate for the practical control of larger systems. However, the possibility of overflow introduces a nonlinear and non-differentiable element to the formulation as it can be described by means of a maximum of linear terms. This particular challenge can be addressed by smoothing methods that result in a continuous nonlinear OCP or by logical constraints that result in a linear MIOCP. We discuss one approach to handle the nonlinear OCP and two approaches to handle the linear MIOCP – a constraint branching algorithm and a disjunctive programming approach that leads to an MILP. We compare the approaches for a set of realistic rain scenarios.

5.3.1 Problem formulation

Combined sewer networks are present in many large cities all over the world. These networks carry both wastewater and storm water together. During low to moderate rainfall, this water is carried to wastewater treatment plants where it is treated before being released to the receiving environment (usually a river or the sea). However, during heavy-rain events the network capacity can be easily overloaded, causing urban surface flooding as well as untreated water discharges to the environment, known as Combined Sewer Overflow (CSO). To avoid these unwanted discharges, retention tanks are usually built along the network to store the water during the peak rain intensity periods and later release it at lower flow rates. Since these tanks are clearly expensive and difficult to locate in urban areas, their efficient operation has become a topic of major interest.

Due to the network structure as well as the uncertain distribution of the rain inflows, global real-time control through network monitoring and rain intensity forecasts is regarded as the best control option, cf. [165]. This approach has been studied by [47, 70, 127, 128, 129, 135, 136, 142, 146] among others.

In this section, a network model is presented to be used in an optimal control strategy to minimize unwanted sewage discharges. Having in mind that a real-time control approach would require the operator to solve the OCPat every time step (in a model predictive control strategy), the main contribution of this section is to explore some of the applicable formulations, which were discussed in Section 3.4, and the corresponding solving procedures in order to determine the best algorithmic option in terms of solving time.

Outline

First, in Section 5.3.2, we present a generic modeling approach for the overflow in sewage networks and also specify our choices for the flow system, which result in a linear system for the network flows and a piecewise linear equations system for the overflows. Then, in Section 5.3.3, we present four ways to handle the piecewise linear equation system: Firstly, we use a nonlinear formulation, which smooths the non-differentiable kinks. Secondly, we reformulate the system using the MLD formulation, which results in an MILP. Thirdly, we present the GDP approach for this system, which is a formal generalization of the MLD approach, and apply it in two different ways, namely a *constraint branching algorithm* and a lifted formulation. The implementations of all but the nonlinear approaches are described in Section 5.3.4 and their computational results are compared. Finally, we we draw the main conclusions in Section 5.3.5.

5.3.2 Modeling of overflow in sewage networks

The main difference among the aforementioned studies is the mathematical model that is used to describe the network dynamics. Sewer networks are usually non-pressurized, i.e., water flows only due to the effect of gravity. The classical physical model for open-channel flow is based on a nonlinear partial differential equation model (the SAINT-VENANT equations). Such a model is not adequate for control purposes, especially in an optimal/predictive control framework, where the model equations appear directly as constraints of an optimization problem, thus transmitting its features to it (e.g. number of variables, convexity, linearity, presence of integer/boolean variables). The discretization of the SAINT-VENANT equations would in turn lead to a huge system of nonlinear equations, which would lead to an optimization problem not solvable in a real-time control framework.
Therefore, simplified models for the network dynamics should be developed. These models must be able to capture the most notable features of the dynamics at computationally acceptable efforts, compatible with the real-time computation of the control strategies. The model presented in the following is based on the so-called *virtual tank* model, first developed by GELORMINO and RICKER in [70] and later used in [47, 135, 136, 146]. However, novel ways to deal with CSO as well as surface flooding are presented here.
In this approach, sewage networks consist of several elements, which are described in the following. For water storage, there are water retention *tanks* built by the network operator and so-called *virtual tanks*, each of which represents a set of sewage collectors for a specific zone of the city. These tanks are taken together in the set $\mathcal{N}$ – their main difference is that for tanks the in- and outflow can usually be controlled, whereas for virtual tanks, there are not necessarily controllable flows. There is a treatment plant to clean the sewage where, optimally, all the sewage should be processed. To connect these different tanks and virtual tanks, there are *sewers*, which can be partly controlled with pumps and valves. In some sewers, there are *redirection gates* to manipulate and redirect the flow. Other sewers are connected by

simple junctions. Both these structures are treated as tanks with a maximum volume of 0 and where all inflow is directly forwarded as outflow. These networks can be displayed as directed graphs. A conceptual example of a small network of this type is displayed in Figure 5.4.

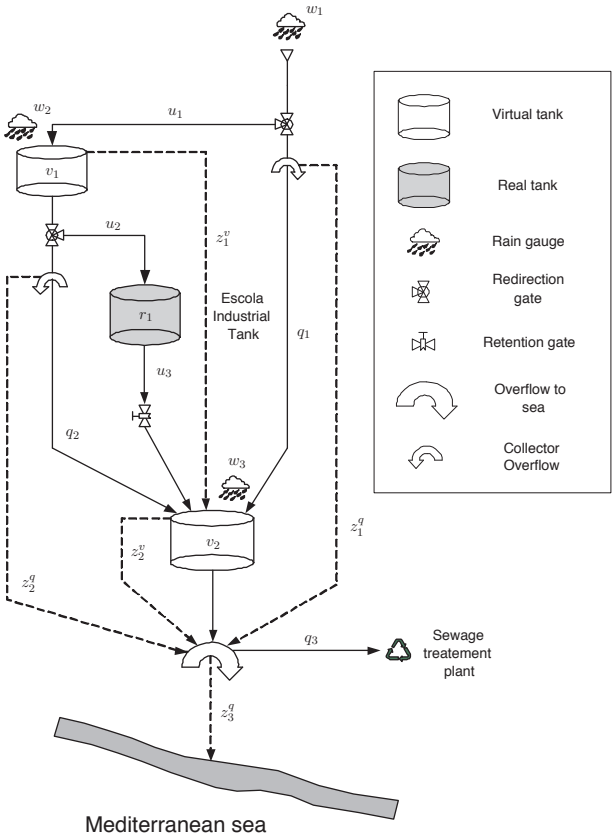


Figure 5.4: A small example network with two virtual tanks v_1, v_2 and one retention tank r_1. Flow paths that appear due to overflow are represented as dashed connections.

Network modeling

During normal operation, the network can easily transport the sewage to the treatment plant. However, in the presence of heavy rain, it may happen that there exists no viable flow path for the network to process all the incoming water. In these scenarios, overflow happens – flow paths appear that are not always present and depend on the system state and inputs. It is assumed that a short term prediction of the rain intensity as well as a prediction of sewage inflow at all points of the network is available in order to run the model. The system states are the volumes $\boldsymbol{v}$ in each tank and virtual tank. We also need to take into account the flows $\boldsymbol{q}$ through the sewers as dependent variables. However, some of the flows may be directly controlled with valves or pumps.

For notational simplicity, we introduce the accumulated inflow $q_i^{\text{in}}(t_k)$ for each node $i \in \mathcal{N}$ at each time step t_k. This also takes into account the inputs into the system – rain as well as normal sewage – given by $\boldsymbol{w}(t_k)$:

$$q_i^{\text{in}}(t_k) \stackrel{\text{def}}{=} w_i(t_k) + \sum_{\substack{j \in \mathcal{N}: \\ (ji) \in \mathcal{U} \cup \mathcal{C}}} q_{ji}(t_k).$$

For each tank i of the system, there is only one regular outflow q_i^{out} – controllable or not. For the uncontrolled outflows of the tanks due to gravity, we use a rather simplifying assumption according to [146]: we assume them to be linear in the amount of water, i.e.,

$$q_i^{\text{out}}(t_k) = \beta_i \, v_i(t_k), \tag{5.10}$$

where the parameter β_i is obtained from historical sensor data or, in a real-time control approach, is to be calibrated online.

A more precise formulation would consider the water pressure in both nodes adjacent to the sewer as well as the friction along it. However, this formulation employs nonlinearities in the model and its contribution is neglectable in our setting. Thus, it shall not be considered in this work. The outflows that completely leave the system are the desired flow into the treatment plant and undesired overflows into the surrounding area polluting the environment.

The flow system is governed by the law of conservation of mass for each node i:

$$\dot{v}_i(t) = q_i^{\text{in}}(t) - q_i^{\text{out}}(t), \qquad t \in [t_0, t_{\text{f}}], i \in \mathcal{N}, \tag{5.11}$$

with the cumulative inflow $q_i^{in}(\cdot)$ and the outflow $q_i^{\text{out}}(\cdot)$.

This system is discretized with an explicit EULER method on an equidistant time grid with step length Δt and m time intervals to obtain a discrete-time linear flow system

$$v_i(t_{j+1}) = v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j) \right), \qquad j = 0, \ldots, m-1, i \in \mathcal{N}, \tag{5.12}$$

where $t_j = j\Delta t$ represents the j-th time step and m is the total number of time steps that the model simulates. For abbreviation purposes, we introduce the set of discretization indices

$$\mathcal{D} \stackrel{\text{def}}{=} \{0, \ldots, m-1\}.$$

All the controlled flows are limited due to physical constraints such as sewer sizes and pump capacities. We assume that the volumes of the real retention tanks are limited as well and that they cannot overflow since they are often placed underground and not connected to the surface. Their inflows are hence always controlled to prevent overflow that could physically happen. In real applications, an overflow emergency mechanism is always present in case there is a malfunction in the controlled devices, but we will not deal with such special

situations in this thesis.

The network structure of the sewage network is given by the following sets in Table 5.8, the states and controls of the system are summarized in Table 5.9, and the scenario specific parameters are summarized in Table 5.10.

Name	Description
$\mathcal{N}$	Set of nodes, these include retention tanks, virtual tanks, redirection gates, the environment and treatment plants
$\mathcal{U}$	Set of uncontrolled flow connections
$\mathcal{C}$	Set of controlled flow connections
$\mathcal{Z}$	Set of connections that appear due to overflow

Table 5.8: Sets which determine the sewage network structure.

Name	Description	Unit	Domain
v_i	Volume of node $i \in \mathcal{N}$	m^3	$[0, v_i^{\max}]$
q_{ij}	Controlled network flows from node i to node j, $(ij) \in \mathcal{C}$	m^3/s	$[0, q_{ij}^{\max}]$
q_{ij}	Uncontrolled network flows from node i to node j, $(ij) \in \mathcal{U}$	m^3/s	$[0, q_{ij}^{\max}]$
q_i^{in}	Accumulated inflow into node i, $i \in \mathcal{N}$	m^3/s	$\mathbb{R}_0^+$
q_i^{out}	Accumulated outflow from node i, $i \in \mathcal{N}$	m^3/s	$\mathbb{R}_0^+$
z_i	Network overflows from node i, $i \in \mathcal{N}$	m^3/s	$\mathbb{R}_0^+$

Table 5.9: States and controls of the sewage model.

Name	Description	Unit	Domain
$\boldsymbol{v}^{\max}$	Capacity of the nodes	m^3/s	$\left(\mathbb{R}_0^+\right)^{\lvert\mathcal{N}\rvert}$
$\boldsymbol{q}^{\max}$	Capacity of the pipes	m^3/s	$\left(\mathbb{R}_0^+\right)^{\lvert\mathcal{C}\cup\mathcal{U}\rvert}$
$\boldsymbol{w}$	Forecasts of rain and sewage inputs for each node	m^3/s	$\left(\mathbb{R}_0^+\right)^{\lvert\mathcal{N}\rvert}$
$\boldsymbol{\beta}$	Slope parameter that determines the flows by gravity	–	$\left(\mathbb{R}_0^+\right)^{\lvert\mathcal{N}\rvert}$

Table 5.10: Parameters of the sewage model.

Overflow modeling

Without overflows, the system would be fully described with the equations from above and could be implemented as a discretized OCP. As mentioned before, overflow happens if the volume obtained by the flow system will exceed the maximum capacity $v_i^{\max}$ of the node i. Joints of different sewers are treated as tanks with maximum capacity $v_i^{\max} = 0$. In case of the flows being at the physical limits, no overflow happens, but if they exceed the physical limits, the excess amount is considered to be overflow. Overflows in each virtual tank and sewer joint can be redirected to other virtual tanks, i.e., to another catchment area, or to the environment (usually a river or the sea). Only in the latter case, the overflow volume leaves the network permanently.
As soon as an overflow $z_i(t_j)$ appears, the corresponding node's mass conservation equation changes to

$$\dot{v}_i(t) = q_i^{\text{in}}(t) - q_i^{\text{out}}(t) - z_i(t), \qquad t \in [t_0, t_f], i \in \mathcal{N}.$$

All the overflow from one node i flows to the same target node j with $(ij) \in \mathcal{Z}$. The accumulated inflow is adapted to include inflows due to overflow:

$$q_i^{\text{in}}(t_k) \overset{\text{def}}{=} w_i(t_k) + \sum_{\substack{j \in \mathcal{N}: \\ (ji) \in \mathcal{U} \cup \mathcal{C}}} q_{ji}(t_k) + \sum_{\substack{j \in \mathcal{N}: \\ (ji) \in \mathcal{Z}}} z_j(t_k).$$

The overflow $z_i(t_j)$ can e.g. be modeled with logical decisions

$$\begin{aligned} \text{IF} \quad & v_i^{\max} \leqslant v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j) \right), \\ \text{THEN} \quad & z_i(t_j) = \frac{1}{\Delta t} \left(v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j) \right) - v_i^{\max} \right), \\ \text{ELSE} \quad & z_i(t_j) = 0, \end{aligned} \tag{5.13}$$

where the THEN expression together with (5.12) adjusted for overflow, i.e.,

$$v_i(t_{j+1}) = v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j) - z_i(t_j) \right), \qquad j \in \mathcal{D}, i \in \mathcal{N},$$

also sets $v_i(t_{j+1}) = v_i^{\max}$. This approach is further explained in Section 3.7.1.
Another modeling approach uses a disjunction as described in Section 3.1:

$$\begin{bmatrix} \omega_i(t_j) = 0 \\ z_i(t_j) = 0 \end{bmatrix} \vee \begin{bmatrix} \omega_i(t_j) = 1 \\ v_i^{\max} \leqslant v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j) \right) \\ z_i(t_j) = \frac{1}{\Delta t} \left(v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j) \right) - v_i^{\max} \right) \end{bmatrix}, \qquad j \in \mathcal{D}, i \in \mathcal{N}. \tag{5.14}$$

Equivalently, the overflow can be modeled directly with the maximum function

$$z_i(t_j) \stackrel{\text{def}}{=} \frac{1}{\Delta t} \max\left\{0,\ v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right\}, \quad i \in \mathcal{N}, j \in \mathcal{D}. \quad (5.15)$$

Then, the constraints are piecewise affine equations and thus nonlinear and non-differentiable. Therefore, each system directly containing these constraints becomes not only nonlinear but also non-convex and non-differentiable. One such constraint is displayed in Figure 5.5.

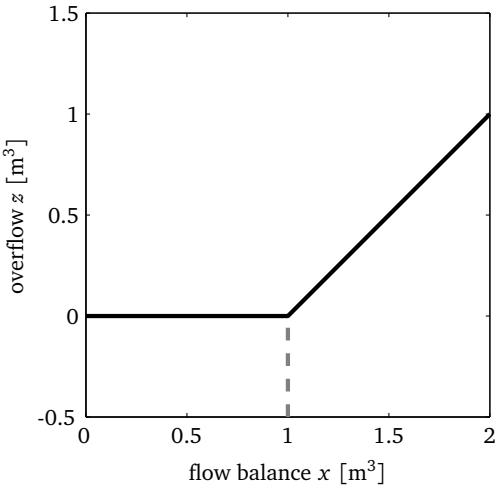


Figure 5.5: The overflow function $z(x) = \max\{0, x-1\}$ with maximum capacity $v^{\max} = 1$, $\Delta t = 1$ and water volume change x. It is nonlinear, non-convex and non-differentiable in the kink at $x = 1$.

Overflows in joints (junctions and redirection gates) are modeled in an analogous way by considering a joint as a tank with zero capacity, and hence constant volume of 0 (also $\dot{v}_i(\cdot) = 0$). In this case, overflows occur when the inflow to the joint exceeds the maximum outflow. Due to notational simplicity, we omit this part of the formulation for the remainder of this section since there are no additional insights gained. All formulations directly carry over to these slightly changed conditions. Here, we briefly show how to adapt the logical formulation:

$$\begin{aligned} \text{IF} \quad & q_i^{\text{in}}(t_j) \geqslant q_i^{\max} \\ \text{THEN} \quad & z_i(t_j) = q_i^{\text{in}}(t_j) - q_i^{\max} \\ \text{ELSE} \quad & z_i(t_j) = 0, \end{aligned}$$

where $q_i^{\max}$ is the maximum capacity of the sewers leaving joint i.

The objective of the network operators is to minimize all overflows, because they have two negative impacts: inside the city they disturb the residents and outside the city they pollute the environment. However, different weights $\lambda_i > 0$ can be associated with different overflow points depending on their geographic location. Therefore, the management objective can be formulated as follows:

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{D}} \lambda_i \, z_i(t_j).$$

5.3.3 Reformulations for optimization

Going a step further from the optimal control approach, a predictive control strategy solves an OCP at each time step taking advantage of new sensor information of the system and new predictions of the perturbations. Thus, it is of great interest to develop fast solving algorithms that guarantee that the solution is obtained within the available time. This would be the goal of the application to the real world.
This section covers four approaches to handle the given overflow formulation and obtain optimization models that can either be solved by standard algorithms or the algorithm needed is directly specified. The first approach is based on replacing the non-differentiable overflow equations by differentiable approximations. We obtain an NLP, which can be locally solved by a derivative-based algorithm. The other three approaches are based on the fact that the nonlinearities and non-differentiability are induced by the piecewise linear overflow equation. Therefore, they can be covered with a disjunctive formulation. These models can be formulated as described in Sections 3.5–3.8 to produce optimization problems including binary variables. Since our problem is mostly linear, we apply only those approaches that preserve linearity and can hence be solved using MILP algorithms. One major distinction of the last three approaches is that the MLD and the GDP approach make explicit use of the binary variables to obtain a problem to be solved by standard MILP solvers while the constraint branching approach makes use of the problem properties to come up with a branching strategy that implicitly realizes the logic decisions. A different kind of reformulation for piecewise linear functions that also avoids the usage of binary variables is the introduction of SOS2-constraints. However, it is known that the resulting problems are equivalent to the formulation with binary variables as presented in the GDP approach, cf. [108].

Nonlinear smoothing

Constraints (5.15) can be reformulated using smoothing for the non-differentiabilities. We use hyperbolic smoothing to connect the two arms of the maximum function with a portion of a circle of radius r. The circle center is at (m_1, r) with

$$m_1 \stackrel{\text{def}}{=} (1 - \sqrt{2})r + v_i^{\max}.$$

The resulting overflow formulation is

$$z_i^r(x_i(t_j)) = \begin{cases} 0, & \text{if } x_i(t_j) \leqslant m_1, \\ r - \sqrt{r^2 - (x_i(t_j) - m_1)^2}, & \text{if } x_i(t_j) \in \left(m_1, m_1 + \frac{r}{\sqrt{2}}\right), \\ x_i(t_j) - v_i^{\max}, & \text{if } x_i(t_j) \geqslant m_1 + \frac{r}{\sqrt{2}}, \end{cases}$$

with

$$x_i(t_j) = \frac{1}{\Delta t} v_i(t_j) + q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j).$$

We obtain the continuously differentiable function $z_i^r(\cdot) \in C^1(\mathbb{R})$, which is sufficiently differentiable for the usual nonlinear solving methods, i.e., active set and interior point based on accumulated first derivative approximations for the HESSIAN. The change from the previous formulation is displayed in Figure 5.6.

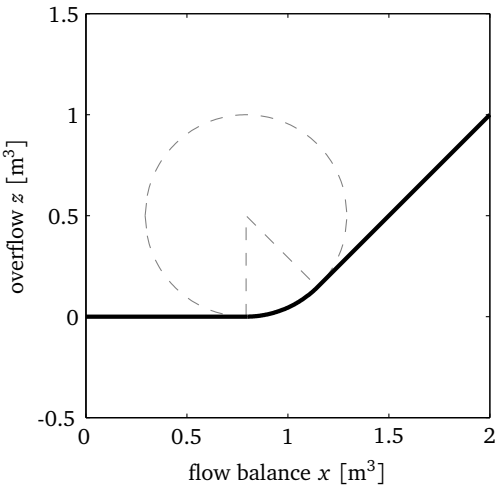


Figure 5.6: The smoothened function with maximum capacity $v_i^{\max} = 1$, $\Delta t = 1$ and radius $r = 0.5$. Due to smoothening, the non-differentiable kink is replaced by a differentiable approximation.

Mixed Logical Dynamical system

The MLD system formulation can be used to combine the inherent logic of the overflows with the dynamical control problem. The general approach has been described in Section 3.7.1. In the case of overflow modeling, we use binary variables $\omega_i(t_j)$, which model whether there is

overflow of node i at time step t_j. This results in the following conditional IFF-THEN-system:

$$\left[\omega_i(t_j)=1\right] \leftrightarrow \left[v_i(t_j)+\Delta t\left(q_i^{\text{in}}(t_j)-q_i^{\text{out}}(t_j)\right)-v_i^{\max}\geqslant 0\right], \tag{5.16}$$

and

$$\begin{aligned} &\text{IF} && \omega_i(t_j)=1 \\ &\text{THEN} && z_i(t_j)=v_i(t_j)+\Delta t\left(q_i^{\text{in}}(t_j)-q_i^{\text{out}}(t_j)\right)-v_i^{\max} \\ &\text{ELSE} && z_i(t_j)=0. \end{aligned} \tag{5.17}$$

together with the flow equation

$$v_i(t_{j+1})=v_i(t_j)+\Delta t\left(q_i^{\text{in}}(t_j)-q_i^{\text{out}}(t_j)-z_i(t_j)\right).$$

The IFF-THEN-formulations (5.16) are put into inequalities with a tolerance parameter ε and bounds m_i and M_i, which can be computed using the bounds of the involved variables:

$$\begin{aligned} &v_i(t_j)+\Delta t\left(q_i^{\text{in}}(t_j)-q_i^{\text{out}}(t_j)\right)-v_i^{\max}\geqslant m_i\left(1-\omega_i(t_j)\right), \\ &v_i(t_j)+\Delta t\left(q_i^{\text{in}}(t_j)-q_i^{\text{out}}(t_j)\right)-v_i^{\max}\leqslant\left(M_i+\varepsilon\right)\omega_i(t_j)-\varepsilon. \end{aligned}$$

The IF-THEN-ELSE-formulations (5.17) are reformulated with

$$z_i(t_j)=\omega_i(t_j)\left(v_i(t_j)+\Delta t\left(q_i^{\text{in}}(t_j)-q_i^{\text{out}}(t_j)\right)-v_i^{\max}\right),$$

which is, for binary $\omega_i(t_j)$, equivalent to the linear system

$$\begin{aligned} &z_i(t_j)\leqslant M_i\omega_i(t_j), \\ &z_i(t_j)\geqslant m_i\omega_i(t_j), \\ &z_i(t_j)\leqslant v_i(t_j)+\Delta t\left(q_i^{\text{in}}(t_j)-q_i^{\text{out}}(t_j)\right)-v_i^{\max}-m_i\left(1-\omega_i(t_j)\right), \\ &z_i(t_j)\geqslant v_i(t_j)+\Delta t\left(q_i^{\text{in}}(t_j)-q_i^{\text{out}}(t_j)\right)-v_i^{\max}-M_i\left(1-\omega_i(t_j)\right), \end{aligned}$$

with the same m_i and M_i as used above. As these bounds are needed, this is a realization of the Big-M formulation of Section 3.7.

The reformulation can be automatically done in MATLAB©with HYSDEL, cf. [179], which is now part of MPT, cf. [114]. However, HYSDEL does not deal with systems including time dependent disturbances, which appear in our case as the rain inflow to the network. To solve this problem, rain inflow variables are initially regarded as controlled variables and later the resulting matrices of the MLD format produced by HYSDEL are split to separate the actual controlled flows from the rain inflow disturbances. A drawback of this workaround is that artificial bounds w_i^{max} on the rain flow variables have to be added.

If we used the nonlinear pressure formulation for the outflow instead of the linearized version,

this approach would produce a MINLP. These problems are usually very difficult to solve – especially considering the problem's size, which arises from the time discretization together with the network size.

Constraint branching algorithm

We use the specific problem structure induced by the formulation of overflow to create a branch-and-bound algorithm. It branches on decisions without adding the corresponding variables explicitly to the problem. In contrast to the previously described MLD method, we do not have to add artificial variable bounds and tolerance parameters, which have an impact on the behavior of the algorithm since they slightly disturb the model.
The maximum equation formulation (5.15) from Section 5.3.2

$$z_i(t_j) \overset{\text{def}}{=} \frac{1}{\Delta t} \max\left\{0,\ v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right\}$$

can be relaxed to the linear inequalities

$$z_i(t_j) \geqslant 0, \tag{5.18}$$

$$z_i(t_j) \geqslant \frac{1}{\Delta t} \left(v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right). \tag{5.19}$$

Thereby, we obtain an LP as the relaxed problem. The meaning of the relaxation is that the controller can choose, during each time step for each node, how much overflow there is – even if in this situation no overflow would actually occur in reality. It can have a positive impact to artificially generate an overflow because it may prevent overflow at another part of the system, which may have worse weights λ_i or may propagate further overflow. However, since overflows $z_i(t_j)$ are to be minimized, for optimal solutions often one of the inequality constraints (5.18) and (5.19) holds with equality.
If the solution of the relaxed problem fulfills for all pairs of inequalities (5.18) and (5.19) one of both with equality, then it is already a solution of the original problem. If this is not the case for all overflows, the algorithm has to enforce this behavior. Branching is done on the decision which one of the two inequalities shall be fulfilled with equality for all children in the branching tree. Instead of dichotomy branching on variable values and adding the constraints $x \leqslant \lfloor \bar{x} \rfloor$ or $x \geqslant \lceil \bar{x} \rceil$ for a fractional variable $\bar{x}$, we add for one branch the constraint

$$z_i(t_j) \leqslant 0, \tag{5.20}$$

and for the second branch the constraint

$$z_i(t_j) \leqslant \frac{1}{\Delta t} \left(v_i(t_j) + \Delta t \left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right), \tag{5.21}$$

if the relaxed solution $\bar{z}$ fulfilled both with ">". This is done until for each overflow (5.20) or (5.21) holds, which is then a solution of the original problem with the maximum equation (5.15). This property is enforced through the branching procedure.
We use the following problem formulation OF($\mathcal{A},\mathcal{B}$) for subproblems at the nodes. We use the sets $\mathcal{A}$ and $\mathcal{B}$ of index pairs to describe where the overflow has already been fixed for this node:

$$\min_{\boldsymbol{z},\boldsymbol{q},\boldsymbol{v}} \quad \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{D}} \lambda_i\, z_i(t_j) \qquad \text{(OF}(\mathcal{A},\mathcal{B})\text{)}$$

s. t. flow equations and network limits,

$$\begin{aligned}
z_i(t_j) &\geq \frac{1}{\Delta t}\left(v_i(t_j) + \Delta t\left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right), && j\in\mathcal{D}, i\in\mathcal{N},\\
z_i(t_j) &\geq 0, && j\in\mathcal{D}, i\in\mathcal{N},\\
z_i(t_j) &= \frac{1}{\Delta t}\left(v_i(t_j) + \Delta t\left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right), && (i,j)\in\mathcal{A},\\
z_i(t_j) &= 0, && (i,j)\in\mathcal{B}.
\end{aligned}$$

Each node of the branching tree contains its own version of those two sets to describe the fixed constraints and the general algorithm is outlined in Algorithm 5.1. This is a simple branching framework, which can easily be upgraded to contain a bounding procedure to obtain a branch-and-bound algorithm. We also experimented with the strategy to get the constraint on which to branch next. The best results were obtained with the intuitive strategy of choosing the earliest overflows.
The presented approach is similar to standard branching, with the difference that we branch on the disjunctive constraints instead of on the variables. It can be applied to other problems as well. One has to have piecewise continuously differentiable equations and that the relaxation of the equations to inequalities provides a convex (preferably linear) problem. It helps if the violation of the equations is penalized in the relaxation – even more so if this is an innate property of the problem. This approach is very dependent on the exact problem structure and to our knowledge has not yet been explored.
Notice that there exists a different kind of constraint branching, as e.g. RYAN-FOSTER branching for SOS1-constraints, e.g. presented in [67], which partitions the feasible set based on special constraint structures.

General Disjunctive Programming

Instead of the two preceding approaches, overflows can also be reformulated with the GDP framework from [87], which is the foundation of the perspective formulation from Section 3.9. The authors use general disjunctions in a first step to model reality and then propose different ways to handle these disjunctions. With the disjunctive formulation (5.14) and with

Algorithm 5.1: Constraint branching algorithm.

Data: Network structure, time discretization, rain inputs.
Result: Optimal solution, i.e., best solution in the solution pool.

Create queue of active tree nodes $\mathcal{Q}$ and add the empty root node ({}, {}) to $\mathcal{Q}$.
while $\mathcal{Q}$ is not empty **do**
  Choose node n with corresponding fixed constraints $(\mathcal{A}_n, \mathcal{B}_n)$ by search strategy and remove n from $\mathcal{Q}$.
  Solve problem OF$(\mathcal{A}_n, \mathcal{B}_n)$ and obtain solution variables $(\boldsymbol{z}, \boldsymbol{v}, \boldsymbol{q})$.
  if problem is feasible **then**
    if $\forall (i,j) \notin \mathcal{A}_n \cup \mathcal{B}_n$:
    $z_i(t_j) = 0$ or
    $z_i(t_j) = \frac{1}{\Delta t}\left(v_i(t_j) + \Delta t\left(q_i^{in}(t_j) - q_i^{out}(t_j)\right) - v_i^{max}\right)$
    then
      Add n's solution $(\boldsymbol{z}, \boldsymbol{v}, \boldsymbol{q})$ to the pool of solutions.
    else
      Find a pair (i,j) such that
      $z_i(t_j) > 0$ and
      $z_i(t_j) > \frac{1}{\Delta t}\left(v_i(t_j) + \Delta t\left(q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j)\right) - v_i^{\text{max}}\right)$
      Add two new child nodes to $\mathcal{Q}$, which have the following fixed constraints:
      - $(\mathcal{A}_n \cup \{(i,j)\}, \mathcal{B}_n)$: There is overflow at node i at time step j.
      - $(\mathcal{A}_n, \mathcal{B}_n \cup \{(i,j)\})$: There is no overflow at node i at time step j.
    end
  end
end

the auxiliary formulation

$$x_i(t_j) \stackrel{\text{def}}{=} \frac{1}{\Delta t} v_i(t_j) + q_i^{\text{in}}(t_j) - q_i^{\text{out}}(t_j),$$

with upper bounds $\boldsymbol{M}$ for the new states $\boldsymbol{x}$, we obtain the problem

$$\begin{aligned}
\min_{\boldsymbol{z},\boldsymbol{q},\boldsymbol{v},\boldsymbol{\omega}} \quad & \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{D}} \lambda_i\, z_i(t_j) \\
\text{s.t.} \quad & \text{flow equations and network limits,} \\
& \begin{bmatrix} \omega_i(t_j) = 1 \\ z_i(t_j) = x_i(t_j) - \dfrac{1}{\Delta t} v_i^{\max} \\ x_i(t_j) \geqslant \dfrac{1}{\Delta t} v_i^{\max} \end{bmatrix} \vee \begin{bmatrix} \omega_i(t_j) = 0 \\ z_i(t_j) = 0 \\ x_i(t_j) \leqslant \dfrac{1}{\Delta t} v_i^{\max} \end{bmatrix}, \qquad j \in \mathcal{D}, i \in \mathcal{N}, \\
& 0 \leqslant x_i(t_j) \leqslant M_i, \\
& \omega_i(t_j) \in \{0,1\}.
\end{aligned}$$

To obtain an MILP in standard form from this formulation, we can either use a Big-M reformulation or a *convex hull* reformulation. It is well known that the convex hull formulation is the same as the perspective formulation presented in Section 3.9. As discussed in Section 3.7, the Big-M formulation's relaxation usually produces weak bounds, whereas the relaxation of the convex hull formulation provides tighter bounds. However, the enhancement effect of the tighter bounds can diminish due to the higher effort needed because of the state duplication. The given Big-M formulation is equivalent to the previously presented MLD approach. Thus, only the convex hull approach is examined in the following.

For the convex hull reformulation, one needs to duplicate the $\boldsymbol{x}$-variables for each branch of the disjunction and to introduce the binary variables $\boldsymbol{\omega}$ as convex multipliers to combine these different branches. The $\boldsymbol{\omega}$-variables model the decision which branch is taken: the one with overflow ($\omega_i = 1$) or the one without ($\omega_i = 0$).

$$\begin{aligned}
\min_{\boldsymbol{z},\boldsymbol{q},\boldsymbol{v},\boldsymbol{\omega},\boldsymbol{x}^1,\boldsymbol{x}^2} \quad & \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{D}} \lambda_i\, z_i(t_j) \qquad (5.22) \\
\text{s.t.} \quad & \text{flow equations and network limits,} \\
& x_i(t_j) = x_i^1(t_j) + x_i^2(t_j), \\
& z_i(t_j) = x_i^1(t_j) - \frac{1}{\Delta t} v_i^{\max} \omega_i(t_j), \\
& x_i^1(t_j) \in \omega_i(t_j) \left[\frac{1}{\Delta t} v_i^{\max}, M_i \right], \qquad j \in \mathcal{D}, i \in \mathcal{N}, \\
& x_i^2(t_j) \in \left(1 - \omega_i(t_j)\right) \left[0, \frac{1}{\Delta t} v_i^{\max} \right],
\end{aligned}$$

$$\omega_i(t_j) \in \{0,1\}.$$

The difference between the convex hull reformulation, which is a general constraint branching framework, and the *constraint branching Algorithm* 5.1 described in the previous section lies in the different relaxations. The relaxation of the hull reformulation by letting $\omega_i(t_j) \in [0,1]$ is tighter than the LP relaxation proposed in Section 5.3.3, which can be seen in Figure 5.7. However, the tighter part is only in the upper part of the feasible region, which leads to higher overflow values and is not desired since z is to be minimized.

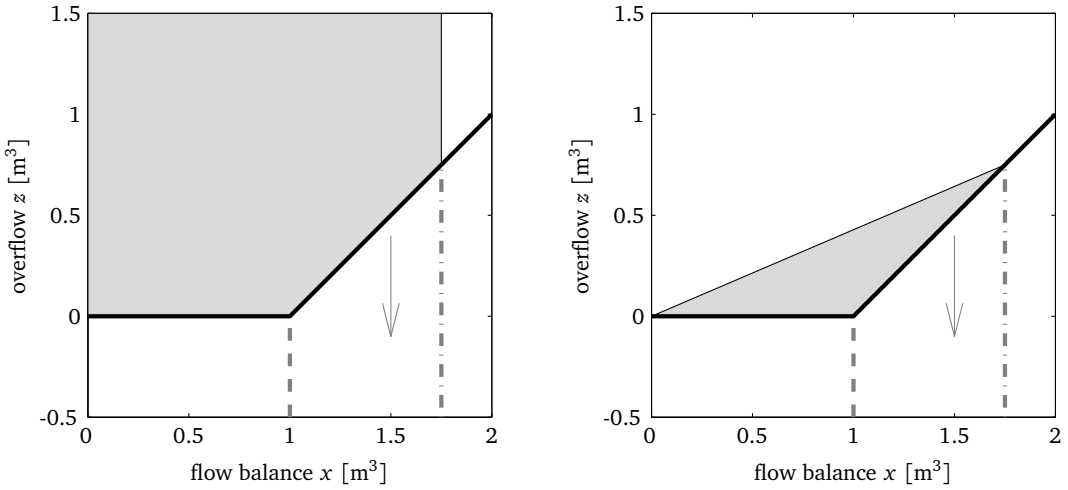


Figure 5.7: On the left-hand side, the LP relaxation from the constraint branching algorithm is drawn, whereas on the right-hand side, the projection of the hull formulation's relaxation from (5.22) is drawn. The left dashed line represents the scaled maximum capacity $\frac{1}{\Delta t} v_i^{\max}$ and the right dashed line represents the bound M on the maximum inflow. The arrow represents the objective.

To assess the properties of this formulation, two observations are made: Firstly, as in the MLD formulation, tight bounds M_i are needed to provide a good formulation. For the overflow problem, these bounds are dependent on the rain scenario and hence need to be recalculated for each instance individually to be accurate. Secondly, the relaxation is only tighter in the non-desired part of the feasible region.

5.3.4 Computational results

We used different versions of the approaches presented in the previous sections to compare their performance for the minimization of the overflow of a particular sewage network.

Test network description

We used a small scale sewage network that is a part of the Barcelona sewer network and for this network data has been provided to build the virtual tank model by the company responsible for the management of the network, CLAVEGUERAM DE BARCELONA S.A. (CLABSA). This network consists of one real tank, 11 virtual tanks and 4 redirection gates, as shown in Figure 5.8.
For the rain data we used 22 different real rain scenarios to compare the algorithms. This data has also been provided by CLABSA and consists of a selection of rain events that occurred in the period between 1996 and 1999, ranging from 4 to 12 hours each. These rain events are representative of the different distribution of the rain intensity both in space and time. Since heavy rain events occur only sporadically, to obtain a larger set of test scenarios, some have been artificially increased by a factor of 2.
The flow $w_i \left[\frac{\mathrm{m}^3}{\mathrm{s}}\right]$ entering the network at virtual tank i is computed from pluviometer data using a conceptual rainfall-runoff model, which is also based on the virtual tank concept, together with a sewage forecast for the catchment area. The rain inflow is obtained by multiplying the rain intensity $I \left[\frac{\mathrm{mm}}{\mathrm{s}}\right]$ with the catchment area $A_i[\mathrm{m}^2]$ and scaling with a dimensionless ground absorption coefficient φ (calibrated online) to account for infiltration losses, cf. [146]:

$$w_i(t) = \varphi_i \, A_i \, I_i(t).$$

The resulting problem dimensions for the different formulations are displayed in Figure 5.9. The difference between the instances is the number of time steps of the discretization, which varies between 40 and 186. The MLD model is always bigger than the model used by the *constraint branching algorithm* as it needs to add integer variables and additional constraints to model the logical decisions. The GDP model requires even more variables than the MLD model, but a little less constraints to formulate the behavior. The number of constraints needed for the constraint branching algorithm is the number of constraints in the root node relaxation. The constraints, which are added in the process of branching, transform inequalities into equations corresponding to their activation. The smoothed, nonlinear formulation has the same number of inequalities as the constraint branching formulation since the constraint branching formulation realizes one part of the maximum term with simple bounds.

Implementation details

The hyperbolic smoothing approach from Section 5.3.3 with a nonlinear local algorithm gives results which are worse than the results from the other approaches. Since the formulation is nonconvex, the results are not necessarily optimal in two senses: firstly, the reformulation at the kink of the overflow function is not exact and secondly, one would need to use a nonlinear

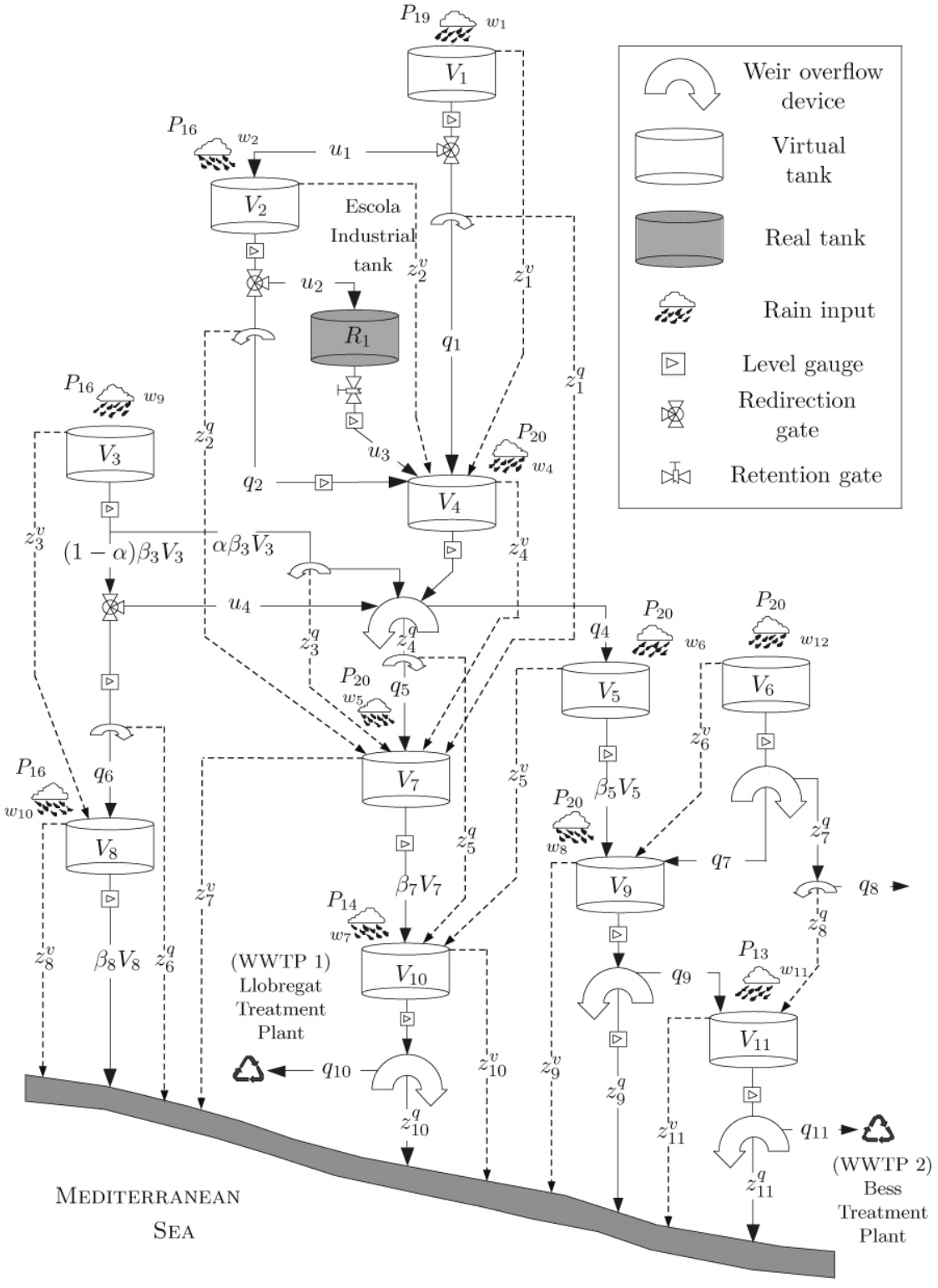


Figure 5.8: Partial Barcelona sewer network with 1 retention tank and 11 virtual tanks.

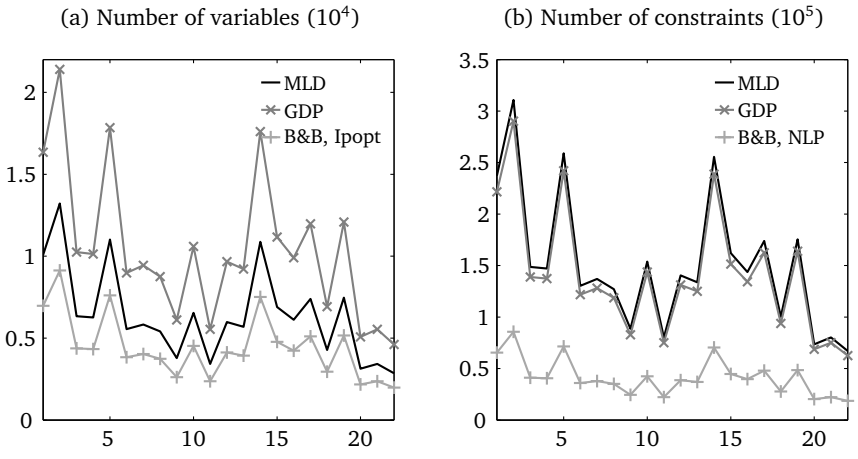


Figure 5.9: Problem dimensions for the different algorithms on 22 rain scenarios for the sewage network displayed in Figure 5.8.

global algorithm to guarantee that the found optimal solution is the true optimal solution, since it is not a convex reformulation. Therefore, purely local algorithms are generally not sufficient to correctly solve the problem.

Nonetheless, we tried to obtain some results for local algorithms since they usually are the subprocedures called in global algorithms. We tested the interior point method IPOPT *v.3.11* with the HSL linear solver MA57. The starting point was the totally uncontrolled network, i.e., all controls and states were set to 0. The solutions differ slightly from the optimal solution obtained by the other approaches, as shown in Figure 5.12. However, the solutions were quite close to the global solutions. The deviation is due to the smoothing with $r \sim 10^{-6} \max_{i \in \mathcal{N}} \left\{ v_i^{\max} \right\}$.

For the MLD approach, we used the HYSDEL package *v.1.2.8*, cf. [179], in MATLAB©*v.7.7.0* to describe the problem and obtain an MILP. Then, the resulting MILP is solved with CPLEX© *v.12.1.0*. The tolerance parameter ε, used to transform strict inequalities into non-strict ones, has been set to $\varepsilon = 10^{-3}$, which is enough taking into account the variables' scale. Additionally, we had to provide artificial upper bounds $w_i^{\max}$ for the rain inputs to allow the transformation of the problem, since they need to be modeled as controlled variables in the MLD framework and then later fixed to the correct perturbation values.

For the *constraint branching algorithm*, we used two different implementations. The first one – a proof of concept – is a very basic implementation of a branch-and-bound algorithm based on the C++ Standard Template Library (STL) priority queue for the active nodes of the branching tree with a *best-first search* as the node selection strategy. The resulting LPs are solved with CPLEX©. However, the algorithm does neither use parallelization for the treatment of nodes nor does it use warm-starting as the solution remains feasible in the dual sense. The second implementation uses CPLEX©'s branch-and-bound framework via

callback-routines to use those features. This framework also provides a more sophisticated search strategy for the branching tree, which leads to more LP iterations but also provides a good solution faster.
For the GDP approach, we reduced the model (5.22) by eliminating the $\boldsymbol{x}^2$-variables from the model. The needed bounds for the additional $\boldsymbol{x}^1$-variables are set very coarsely and safely by setting $M_i = v_i^{\max} + W$, where W is the total rain input. The resulting MILPs were solved with CPLEX©.
All the problems were solved with CPLEX©'s standard settings.

Computational results

The computational results were obtained on a machine with an Intel dual core CPU with 2.66GHz and 8GB RAM. They are displayed in Figures 5.10–5.12.

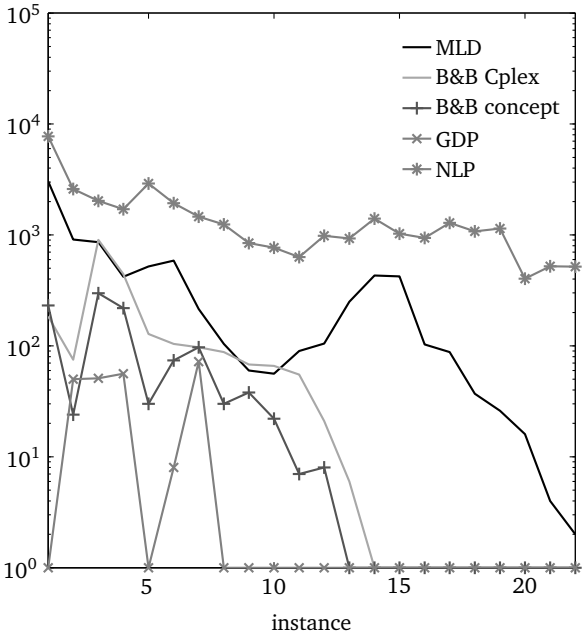


Figure 5.10: Iteration numbers (log) for the different algorithms.

In the figures, it can generally be seen that the problem difficulty does not necessarily coincide with the problems' sizes, which are proportional to the length of the time horizon. The difficulty is more dependent on the amount of overflow happening in the optimal solution.
IPOPT solves all instances but one, although the nonlinear model is nonconvex. Only instance 1 could not be solved to an acceptable level and got stuck in a point of local infeasibility. The objective values are always a little worse (but less than 1% worse) than the objective values of the other algorithms even though the solution is almost identical. This is originated in the smoothing technique used.

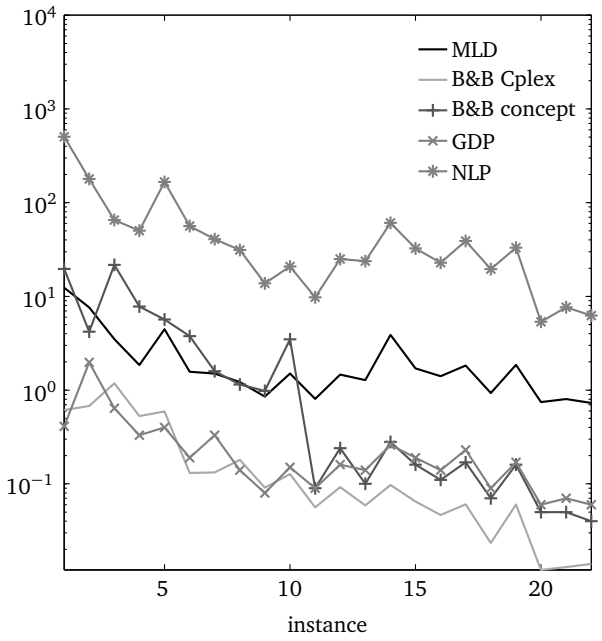


Figure 5.11: Computational times (log) for the different algorithms.

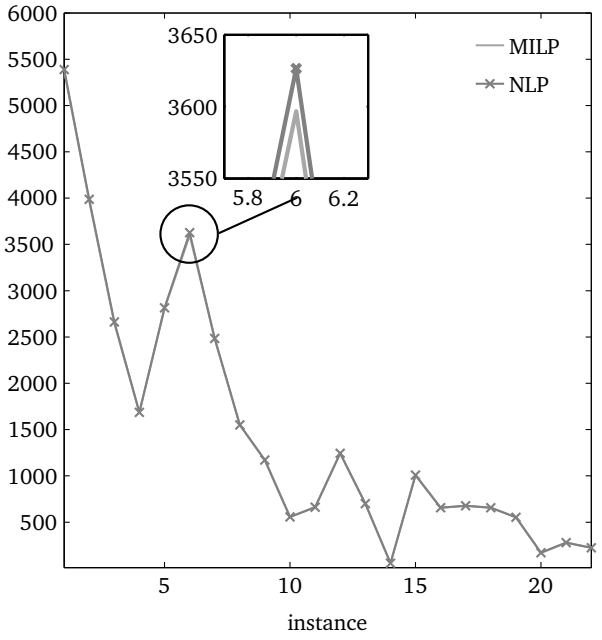


Figure 5.12: Objective values, i.e., total overflow [m^3], for the linear formulations and the nonlinear formulation.

Comparing the iteration numbers, cf. Figure 5.10, the GDP approach is clearly superior to the other approaches solving most problems in the root node, i.e., 17 of the 22 scenarios. Both constraint branching algorithms compare favorably with the MLD approach with regard to the iteration numbers. They also solve about half the problems in the root node, whereas the MLD approach does not solve any problem in the root. This indicates that the GDP algorithm and the constraint branching algorithm provide much better scalability to larger networks than the MLD approach. Notice that the MLD and GDP approaches use binary variables in the models and can hence use cutting planes, e.g. Mixed-Integer Rounding (MIR)-cuts, to tighten the feasible region, whereas the constraint branching formulation only indirectly introduces those binary variables. For these models, the iteration numbers are the numbers of nodes solved in the branch-and-bound framework. The nonlinear model's iterations are NEWTON-type iterations of the interior point method IPOPT. They can hardly be compared with the other formulations' iteration numbers.

With regard to the computational times, the iteration results do not directly carry over, but additionally the problems' dimensions as well as the implementations have to be taken into account. First, we compare both constraint branching schemes. The more sophisticated approach is about one order of magnitude faster than the simple scheme, which is due to hot-starting and parallelization on two cores. Yet, it needs more iterations, which also comes from the parallelization and from the different search strategy, which incorporates diving to provide a fast near-optimal solution. The simple realization is quite slow and sometimes needs even more computational time than the MLD realization, which has to solve larger problems. The more sophisticated implementation overcomes this and is almost always the fastest procedure. The GDP approach is slower than the *constraint branching algorithm* since each iteration (especially the root node) is more costly. One part of this higher effort needed is due to the higher problem dimensions, but the time spent in heuristics and cutting plane procedures to tighten the bounds also increases the effort per node. The constraint branching algorithm cannot apply these procedures due to no incorporated integral variables. For IPOPT, the overall computational times are 2–3 orders of magnitude worse than the computational times of the other approaches. Even for these small test instances, the algorithm comes much closer to the overall time limit of 5 minutes – and for instance 1, it takes more than 5 minutes to arrive at the local minimizer of feasibility.

Overall, we can state that the GDP approach takes a little longer than the *constraint branching algorithm* but the bounds produced by the procedure are tighter. Additionally, the obtained MILP formulation allows the usage of standard algorithms, which can tighten the formulation even further with cutting planes. Therefore, 17 of the 22 test instances were already solved in the root node, in comparison to 10 for Algorithm 5.1, and the number of iterations were in all but one instance the smallest for the GDP approach.

5.3.5 Conclusions

We introduced a modeling approach for sewage networks and presented the corresponding overflow problem. The overflow process cannot be modeled by straightforward means. We showed several ways to model it with binary variables resulting in different MIOCPs. We classified those modeling approaches, if possible, through the classes that have been presented in Chapter 3. Resulting from these different models, we provided several algorithms that solve this kind of problems in the desired 5 minute time window. The considered problems optimize the controls for the network with rain and water consumption forecasts of almost two days taken into account.

The hyperbolic smoothing approach from Section 5.3.3 did not provide the desired results. It converged in all but one instance to the global optimum but it needed much more time to find the optimum than the other algorithms.

The GDP approach has some advantages in comparison to the other two approaches: It is an easily applied standard modeling approach with no additional thought process involved. It provides tighter bounds than the constraint branching algorithm and the MLD approach, which probably makes it scale better towards bigger problems even though the computational times are worse than those of the constraint branching algorithm.

Overall, the GDP's perspective formulation seems to be very well suited for disjunctions of small size – in this case, each disjunction has only two clauses. It preserves linearity of the constraints in the disjunctions, which make the approach especially suited for linear models. The MLD's Big-M approach compares unfavorably. Although it preserves linearity and is as easily implemented, the bounds provided by the relaxation are much worse and hence the resulting algorithm is worse as well. The tailored branching algorithms provide an alternative but need to exploit the special problem structure to be able to compete with the generally applicable perspective formulation.

There remains the open question of the application into a moving horizon framework. For the test instances, the complete problem could be solved within the time limit but for larger instances, this might not be the case anymore. When the horizon shifts, the control of the past problem's first time step is applied and shifted out of the horizon. The old time steps $1, \ldots, m-1$ are shifted to become the new problem's time steps $0, \ldots, m-2$, and a new time step $m-1$ is created with new forecast data. It is certainly possible to start in the old solution for the remaining time steps and to enumerate all possible branches for the newly created time step. This gives a good initial solution if the new forecast data does not deviate too much from the old forecast data and if the model correctly represents the real system and not only an approximation. It remains open whether it is possible to re-use some of the branching information from the previous runs. The application of *model predictive control techniques* to OCPs in [55, 111] can be taken as a vantage point for future research.

Even though the computational times of IPOPT are much worse than those of the other algorithms, the starting point plays an important role. While we chose an all-0 starting point, in

a moving horizon framework, the algorithm could start much closer to the optimum. This would probably speed-up its convergence by a large amount. This question should be studied when the whole process is transferred to a MPC framework where the solutions of the past time steps are used to initialize the new starting point.
All of the presented approaches generally remain applicable, if the linear gravity driven flow is replaced with a pressure dependent equation, which also considers friction, to augment the model. However, the MLD and GDP approaches would produce MINLPs. These are often very hard to solve and the state-of-the-art software is not comparably reliable and fast as, e.g., CPLEX©in the MILP case. On the other hand, the constraint branching approach would give a branching framework with nonlinear subproblems that is already a tailored solution to a MINLP and that seems at first glance at least as solvable.

5.4 Dynamic truck model: Cruise control

The contents of this section are based on the paper

[103] M. JUNG, C. KIRCHES AND S. SAGER, *On Perspective Functions and Vanishing Constraints in Mixed-Integer Nonlinear Optimal Control*, in Facets of Combinatorial Optimization – Festschrift for MARTIN GRÖTSCHEL, M. JÜNGER AND G. REINELT, eds., Springer Berlin Heidelberg, 2013, pp. 387–417.

In this section we describe a cruise control problem for a heavy-duty truck and model it as a MIOCP. Then, we compare the different relaxation approaches for direct methods presented in Section 3.4.
The heavy-duty trucks usually require large amounts of fuel and hence energy-optimal driving is strongly desired. The typically available controls to reach this goal are the gear shifting decisions (typically between 8 and 24 gears) and the acceleration processes. However, this is a non-trivial task and requires extensive training and/or long work-experience from the drivers. Therefore, it is subject of intense scientific research, e.g. [93, 111, 178]. Hybrid engines also fit into the modeling framework, this could be of further interest for future projects.

5.4.1 Problem formulation

Our truck model is slightly simplified by the following assumptions. The track is assumed to be straight and hence we can reduce the position to be one-dimensional, the degree of freedom being the progress toward the destination. The track's height is directly dependent on the position. Additionally, we neglect traffic, which would pose a lot of constraints (mostly speed constraints), but usually one cannot predict the exact behavior of traffic and hence it is omitted.

Dynamics and controls

We describe a basic ODE truck model as introduced in [178]. The controls are the gear decision μ and directly the choice of the indicated engine torque M_{ind} as well as the engine brake torque M_{EB}. Table 5.11 summarizes the controls. Notice that we do not model a retarder or service brake and assume that an a-posteriori breakdown of the engine brake into the three braking devices may be applied.
The vehicle model involves two differential states, velocity v (in m/s) and accumulated fuel consumption Q (in l), which is used to reformulate a LAGRANGE objective functional as a MAYER functional. The traveled distance s (in m) is chosen as the independent variable and we consider the interval $[0, s_f]$. The velocity v is given through the ODE derived from NEWTON's

Name	Description	Unit	Domain
μ	Gear choice	–	$\{1,\dots,n_\mu\}$
M_{ind}	Indicated engine torque	Nm	$[0, M_{\text{ind,max}}]$
M_{EB}	Engine brake torque	Nm	$[0, M_{\text{EB,max}}]$

Table 5.11: Controls of the truck model.

second law

$$\dot{v}(s) = \frac{1}{m\, v(s)} \left(\left(M_{\text{acc}}(s) - M_{\text{brk}}(s) \right) \frac{i_{\text{A}}}{r_{\text{stat}}} - M_{\text{air}}(s) - M_{\text{road}}(s) \right), \tag{5.23}$$

which is the sum of the directed torques, i.e., the effective accelerating torque M_{acc}, the effective braking torque M_{brk}, the turbulent friction torque M_{air} and the kinetic friction torque M_{road} (all in Nm). The factor $\frac{1}{v(s)}$ stems from transformation of the time derivative, in which NEWTON's laws are formulated, to the traveled distance derivative.
The change of the aggregated fuel consumption $Q(s)$ is naturally the current fuel consumption

$$\dot{Q}(s) = \frac{1}{v(s)}\, Q_{\text{fuel}} \left(n_{\text{eng}}(s, \mu(s)), M_{\text{ind}}(s) \right), \tag{5.24}$$

which is a nonlinear function depending on the engine speed n_{eng} (in 1/min) and the indicated engine torque M_{ind} (in Nm). In Table 5.12 a brief overview of the differential states of this truck model is given.

Name	Description	Unit	Domain
v	Velocity	m/s	$(0, v_{\text{max}}]$
Q	Accumulated fuel consumption	l	$[0, \infty)$

Table 5.12: Differential states of the truck model.

Several missing terms are computed from algebraic formulas. The transmitted engine speed n_{eng} depends on the selected gear's transmission ratio i_{T}, the rear axle transmission ratio i_{A} and the static rear tire radius r_{stat}. It is obtained for a given velocity v as

$$n_{\text{eng}}(s, \mu(s)) \overset{\text{def}}{=} \frac{i_{\text{A}}\, i_{\text{T}}(\mu(s))}{2\, \pi\, r_{\text{stat}}}\, 60[s]\, v(s).$$

The accelerating torque M_{acc} is computed from the gearbox transmission ratio i_{T} and gearbox

efficiency η_T as

$$M_{acc}(s,\mu(s)) \stackrel{def}{=} i_T(\mu(s))\,\eta_T(\mu(s))\,M_{ind}(s).$$

Braking torques M_{brk} are combined from the controlled engine brake torque M_{EB} and the internal engine friction torque M_{fric}, which nonlinearly depends on the engine speed:

$$M_{brk}(s,\mu(s)) \stackrel{def}{=} M_{EB}(s) + i_T(\mu(s))\,M_{fric}\left(n_{eng}(s,\mu(s))\right).$$

The additional external braking torques M_{air} and M_{road} are computed as

$$M_{air}(s) \stackrel{def}{=} \frac{1}{2} c_w\,A\,\varrho_{air}\,v^2(s),$$
$$M_{road}(s) \stackrel{def}{=} m\,g\,\left(\sin\gamma(s) + f_r\,\cos\gamma(s)\right),$$

with the shape coefficient c_w, the flow surface A and the air density ϱ_{air} for the turbulent friction. For the kinetic road friction, we have the vehicle's mass m, gravity g, the road's slope $\gamma(s)$ and the rolling friction coefficient f_r. These algebraic states are summarized in Table 5.13.

Name	Description	Unit
n_{eng}	Transmitted engine speed	1/min
M_{acc}	Accelerating torque	Nm
M_{brk}	Total internal braking torque	Nm
M_{air}	Turbulent friction due to air	Nm
M_{road}	Kinetic rolling friction	Nm

Table 5.13: Algebraic states of the truck model.

All the described parameters are summarized in Table 5.14 and Table 5.15.

Constraints

Operational constraints are path and control constraints and both are present in this model. The first operational constraints are due to the mechanics of the truck. There is a maximum state-dependent indicated acceleration torque

$$M_{ind}(s,\mu(s)) \leqslant M_{ind}^{max}\left(n_{eng}(s,\mu(s))\right) \tag{5.25}$$

Name	Description	Unit
A	Flow surface	m^2
c_w	Aerodynamic shape coefficient	–
f_r	Rolling friction coefficient	–
g	Gravity constant	m/s^2
i_A	Rear axle transmission ratio	–
i_T	Gearbox transmission ratio	–
m	Vehicle mass	kg
r_{stat}	Static rear tire radius	m
η_T	Gearbox efficiency	–
ϱ_{air}	Air density	kg/m^3

Table 5.14: Car specific parameters of the truck model.

Name	Description	Unit
Q_{start}	Initially consumed fuel	l
s_f	Track length	m
v_{start}	Initial velocity	m/s
$v_{des}(\cdot)$	Desired velocity profile	m/s
$v_{track}(\cdot)$	Maximum velocity profile (law and curvature)	m/s
$\gamma(\cdot)$	Road's slope	rad

Table 5.15: Scenario specific parameters of the truck model.

with a truck-dependent function of the engine speed on the right-hand side. The same holds for the engine braking torque

$$M_{EB}(s,\mu(s)) \leqslant M_{brk}^{max}\left(n_{eng}(s,\mu(s))\right),$$

which is also limited depending on the engine speed.
To protect the engine, we also give constant limitations on the minimum and maximum engine speeds

$$n_{eng}^{min} \leqslant n_{eng}(s,\mu(s)) \leqslant n_{eng}^{max}.$$

The aforementioned constraints are dependent on the current gear choice and hence will

have to be modeled in a disjunction.

Additional path constraints are due to track properties, e.g. the legal speed limits can be taken into account or sharp bends of the road, which the truck can only pass at a certain velocity:

$$v(s) \leqslant v_{\text{track}}(s).$$

Objective criteria

There are different objective criteria that shall be optimized. We have two important criteria and one lesser important ones. The first criterion is the tracking problem, i.e., the deviation from a given velocity profile v_{des} is to be minimized

$$\Phi_{\text{dev}} \overset{\text{def}}{=} \int_0^{s_f} \left(v(s) - v_{\text{des}}(s)\right)^2 \, \mathrm{d}s. \tag{5.26}$$

This is a LAGRANGE functional but could be reformulated into a MAYER functional as explained in Section 2.1. This criterion also models a desired arrival time at the destination point.

The second important criterion is the aggregated fuel consumption after the complete trip

$$\Phi_{\text{fuel}} \overset{\text{def}}{=} Q(s_f) = \int_0^{s_f} \frac{1}{v(s)} Q_{\text{fuel}} \left(n_{\text{eng}}(s, \mu(s)), M_{\text{ind}}(s)\right) \, \mathrm{d}s. \tag{5.27}$$

The lesser important one is the driving comfort, which is modeled through the change in acceleration and braking torques, i.e.,

$$\Phi_{\text{comf}} \overset{\text{def}}{=} \int_0^{s_f} \dot{M}_{\text{ind}}^2(s) + \dot{M}_{\text{brk}}^2(s) \, \mathrm{d}s. \tag{5.28}$$

There are different ways to handle this situation of multiple objective functions. The most natural one is to weight the different criteria against each other. The objective becomes

$$\min \Phi \overset{\text{def}}{=} \lambda_{\text{dev}} \Phi_{\text{dev}} + \lambda_{\text{fuel}} \Phi_{\text{fuel}} + \lambda_{\text{comf}} \Phi_{\text{comf}}. \tag{5.29}$$

However, this involves a choice for the weights λ. Due to the higher importance of the first two criteria, their weights would be set comparably larger than λ_{comf}. The other two could be adjusted by needs, e.g. if the arrival time is the critical part, then λ_{dev} would be set higher. Whereas, if the arrival time is not that important, minimizing the fuel consumption might be more interesting.

A much more time consuming variant would be to compute the PARETO front, i.e., the surface that contains the optimal solutions for all possible combinations of the weights respecting an

additional normalizing constraint, e.g.

$$\lambda_{\text{dev}} + \lambda_{\text{fuel}} + \lambda_{\text{comf}} = 1.$$

The PARETO front has been studied for a driving car example in an OC setting of the disjunction in [125]. Our main goal here is to study the effects of the chosen relaxation and not to correctly calibrate a real-world application. Therefore, we study only one setting of the weights for each scenario.

Complete formulation

We arrive at a complete model for the operation of a heavy-duty truck. The model in disjunctive form is

$$\min \quad \lambda_{\text{dev}}\Phi_{\text{dev}} + \lambda_{\text{fuel}}\Phi_{\text{fuel}} + \lambda_{\text{comf}}\Phi_{\text{comf}} \tag{5.30}$$

$$\text{s.t.} \quad \bigvee_{1\le i\le n_\mu} \begin{bmatrix} \mu(s) = i \\ \dot{v}(s) = \dfrac{1}{m\, v(s)} \left(M_{\text{int}}(s,\mu(s)) + M_{\text{ext}}(s)\right) \\ \dot{Q}(s) = \dfrac{1}{v(s)} Q_{\text{fuel}}\left(n_{\text{eng}}(s,\mu(s)), M_{\text{ind}}(s)\right) \\ n_{\text{eng}}(s,\mu(s)) = \dfrac{i_{\text{A}}\, i_{\text{T}}(\mu(s))}{2\,\pi\, r_{\text{stat}}}\, 60\, v(s) \\ M_{\text{int}}(s,\mu(s)) = \dfrac{i_{\text{A}}}{r_{\text{stat}}} \left(M_{\text{acc}}(s,\mu(s)) - M_{\text{brk}}(s,\mu(s))\right) \\ M_{\text{brk}}(s,\mu(s)) = M_{\text{EB}}(s) + i_{\text{T}}(\mu(s))\, M_{\text{fric}}\left(n_{\text{eng}}(s,\mu(s))\right) \\ M_{\text{acc}}(s,\mu(s)) = i_{\text{T}}(\mu(s))\, \eta_{\text{T}}(\mu(s))\, M_{\text{ind}}(s) \\ M_{\text{ind}}(s,\mu(s)) \in \left[0, M_{\text{ind}}^{\max}\left(n_{\text{eng}}(s,\mu(s))\right)\right] \\ M_{\text{EB}}(s,\mu(s)) \in \left[0, M_{\text{brk}}^{\max}\left(n_{\text{eng}}(s,\mu(s))\right)\right] \\ n_{\text{eng}}(s,\mu(s)) \in \left[n_{\text{eng}}^{\min}, n_{\text{eng}}^{\max}\right] \end{bmatrix}, \tag{5.31}$$

$$\begin{aligned} M_{\text{ext}}(s) &= -M_{\text{air}}(s) - M_{\text{road}}(s), \\ M_{\text{air}}(s) &= \frac{1}{2} c_{\text{w}}\, A\, \varrho_{\text{air}}\, v^2(s), \\ M_{\text{road}}(s) &= m\, g\, \left(\sin\gamma(s) + f_{\text{r}}\, \cos\gamma(s)\right), \\ v(s) &\in \left(0, v_{\text{track}}(s)\right], \qquad\qquad s \in [0, s_f], \\ \Phi_{\text{dev}} &= \int_0^{s_f} \left(v(s) - v_{\text{des}}(s)\right)^2 \, \mathrm{d}s, \\ \Phi_{\text{fuel}} &= Q(s_f), \end{aligned}$$

$$\Phi_{\text{comf}} = \int_0^{s_f} \dot{M}_{\text{ind}}^2 + \dot{M}_{\text{brk}}^2 \, \mathrm{d}s,$$
$$v(0) = v_{\text{start}},$$
$$Q(0) = Q_{\text{start}}.$$

Example truck realization

The formulations were given as a general description, which would fit most cars and trucks. Here, we specify the parameters and functions for one special truck with $n_\mu = 16$ gears, which we then examine with the relaxation techniques described in Section 3.4. We compare the results for different scenarios in Section 5.4.3.

The truck-specific functions that define the engine characteristics – the internal engine friction torque $M_{\text{fric}}(n_{\text{eng}}(s,\mu(s)))$, the maximally indicated engine torque $M_{\text{ind}}^{\max}(n_{\text{eng}}(s,\mu(s)))$ and the fuel consumption rate $Q_{\text{fuel}}(n_{\text{eng}}(s,\mu(s)), M_{\text{ind}}(s))$ – are displayed in Figure 5.13, whereas $M_{\text{brk}}^{\max}$ is set to be constant. They are specified as

$$\begin{aligned}
M_{\text{fric}}(n_{\text{eng}}(s,\mu(s))) &\overset{\text{def}}{=} c_{\text{fric},2} n_{\text{eng}}^2(s,\mu(s)) + c_{\text{fric},1} n_{\text{eng}}(s,\mu(s)) + c_{\text{fric},0},\\
M_{\text{ind}}^{\max}(n_{\text{eng}}(s,\mu(s))) &\overset{\text{def}}{=} M_{\text{ind}}^{\max} - c_{\text{ind}} \left(n_{\text{eng}}(s,\mu(s)) - n_{\text{eng}}^{\text{best}} \right)^2,\\
M_{\text{brk}}^{\max}(n_{\text{eng}}(s,\mu(s))) &\overset{\text{def}}{=} M_{\text{brk}}^{\max},\\
Q_{\text{fuel}}(n_{\text{eng}}(s,\mu(s)), M_{\text{ind}}(s)) &\overset{\text{def}}{=} c_{\text{fuel},0} + c_{\text{fuel},1}\, n_{\text{eng}}^2(s,\mu(s))\\
&\qquad + c_{\text{fuel},2}\, n_{\text{eng}}(s,\mu(s))\, M_{\text{ind}}(s) + c_{\text{fuel},3}\, M_{\text{ind}}^2(s),
\end{aligned}$$

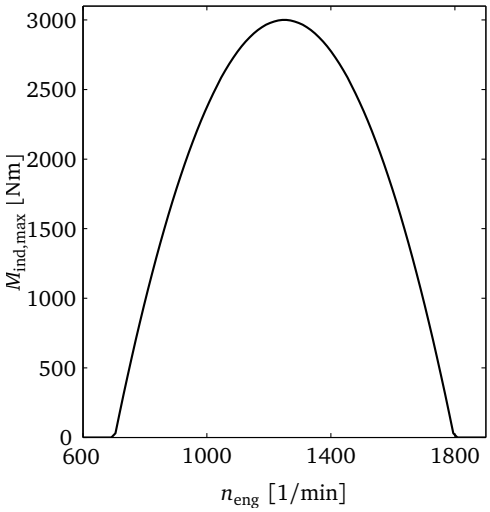
where the engine speed $n_{\text{eng}}^{\text{best}}$ allows the maximum torque and different truck-specific parameters c are added.

The concrete truck-specific parameter values are given at the MIOCP benchmark library MINTOC.DE, described in [151, 155].

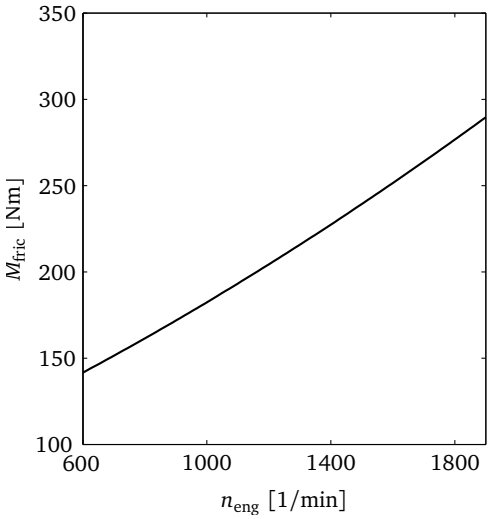
5.4.2 Formulation of problem relaxations

We want to apply a direct method to the truck problem and hence discretize the controls and states of the system (5.30). The controls are discretized with n_c piecewise constant functions, such that on each discretization interval, only one variable is needed per control. The differential states are discretized with a collocation approach in n_s intervals, where the different points are connected via an implicit EULER integration scheme. We choose the state discretization to be a refinement of the control discretization, i.e., $n_s = l\, n_c$ with $l \in \mathbb{N}$. The grids are chosen to be equidistant with step lengths h_c and $h_s = \frac{h_c}{l}$, respectively. The tracking integral Φ_{fuel} and the comfort integral Φ_{comf} in the objective are discretized with the composite trapezoidal rule. The derivatives $\dot{M}_{\text{ind}}$ and $\dot{M}_{\text{ind}}$ are only needed for the objective contribution

(a) Maximum indicated engine torque $M_{\text{ind,max}}$ as a function of the engine speed n_{eng}.



(b) Engine friction M_{fric} reduces torque depending on engine speed n_{eng}.



(c) Fuel consumption Q_{fuel} depending on the current engine speed n_{eng} and the indicated engine torque M_{ind}.

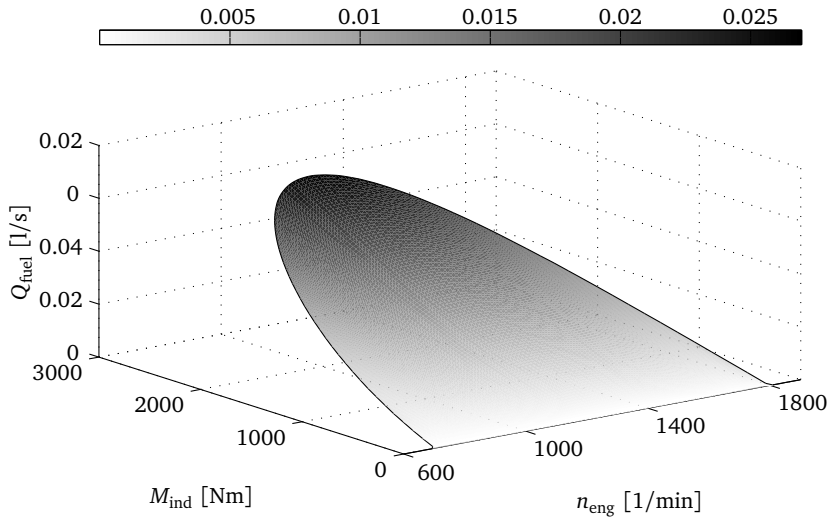


Figure 5.13: Truck engine characteristics for computational example.

and are hence approximated with finite differences. The path and control constraints are enforced only on the control grid's points.
We obtain the discretized formulation

$$\min \qquad \lambda_{\text{dev}}\Phi_{\text{dev}} + \lambda_{\text{fuel}}\Phi_{\text{fuel}} + \lambda_{\text{comf}}\Phi_{\text{comf}} \tag{5.32}$$

$$
\text{s.t.} \quad \bigvee_{1 \le i \le n_\mu} \left[\begin{array}{ll}
\mu_k = i & \\
v_{k,j+1} = v_{k,j} + h_s \, \dfrac{M_{\text{int},k,j+1} + M_{\text{ext},k,j+1}}{m \, v_{k,j+1}}, & 0 \leqslant j \leqslant l-1, \\
Q_{k,j+1} = Q_{k,j} + h_s \, \dfrac{Q_{\text{fuel}}\left(n_{\text{eng},k,j+1}, M_{\text{ind},k}\right)}{v_{k,j+1}}, & 0 \leqslant j \leqslant l-1, \\
n_{\text{eng},k,j} = \dfrac{i_\text{A} \, i_\text{T}(\mu_k)}{2 \, \pi \, r_\text{stat}} \, 60 \, v_{k,j}, & 0 \leqslant j \leqslant l, \\
M_{\text{int},k,j} = \dfrac{i_\text{A}}{r_\text{stat}} \left(M_{\text{acc},k} - M_{\text{brk},k,j} \right), & 0 \leqslant j \leqslant l, \\
M_{\text{brk},k,j} = M_{\text{EB},k} + i_\text{T}(\mu_k) \, M_\text{fric}\left(n_{\text{eng},k,j+1}\right), & 0 \leqslant j \leqslant l, \\
M_{\text{acc},k} = i_\text{T}(\mu_k) \, \eta_\text{T}(\mu_k) \, M_{\text{ind},k}, & \\
M_{\text{ind},k} \in \left[0, M_\text{ind}^\text{max}\left(n_{\text{eng},k,0}\right)\right], & \\
n_{\text{eng},k,0} \in \left[n_\text{eng}^\text{min}, n_\text{eng}^\text{max}\right], & \\
& 0 \leqslant k \leqslant n_c,
\end{array} \right],
$$

$$
\begin{aligned}
v_{k,l} &= v_{k+1,0} && 0 \leqslant k \leqslant n_c - 1, \\
Q_{k,l} &= Q_{k+1,0} && 0 \leqslant k \leqslant n_c - 1, \\
M_{\text{ext},k,j} &= -M_{\text{air},k,j} - M_{\text{road},k}, && 0 \leqslant j \leqslant n, 0 \leqslant k \leqslant n_c, \\
M_{\text{air},k,j} &= \frac{1}{2} c_\text{w} \, A \, \varrho_\text{air} \, v_{k,j}^2, && 0 \leqslant j \leqslant l, 0 \leqslant k \leqslant n_c, \\
M_{\text{road},k} &= m \, g \, \left(\sin\gamma_k + f_\text{r} \, \cos\gamma_k\right), && 0 \leqslant k \leqslant n_c, \\
M_{\text{EB},k} &\in \left[0, M_\text{brk}^\text{max}\right], && 0 \leqslant k \leqslant n_c, \\
v_{k,0} &\in \left(0, v_{\text{track},k}\right], && 0 \leqslant k \leqslant n_c, \\
v_{0,0} &= v_\text{start}, \\
Q_{0,0} &= Q_\text{start}.
\end{aligned}
$$

We apply various of the approaches presented in Sections 3.5–3.9. In all approaches, new binary variables $\omega_{k,i}$ are introduced with the meaning

$$
\omega_{k,i} = \begin{cases} 1, & \text{if } \mu_k = i, \\ 0, & \text{else.} \end{cases}
$$

For all formulations, the relaxations of these binary variables to $[0,1]$ are examined and compared. The discretized formulation already uses a constant M_brk^max.

Inner Convexification

The IC approach was described in Section 3.5. Here, it replaces any occurrence of a gear-dependent parameter (the ratios $i_\mathrm{T}(\mu)$ and $\eta_\mathrm{T}(\mu)$) with the functions $\tilde{i}_\mathrm{T}(\boldsymbol{\omega_k})$ and $\tilde{\eta}_\mathrm{T}(\boldsymbol{\omega_k})$, which are a convex combination of the possible choices in combination with the newly introduced gear choice variables ($0 \leqslant k \leqslant n_c$):

$$\begin{aligned} \tilde{i}_\mathrm{T}(\boldsymbol{\omega_k}) &\overset{\text{def}}{=} \sum_{l=1}^{n_\mu} i_\mathrm{T}(l)\ \omega_{k,l}, \\ \tilde{\eta}_\mathrm{T}(\boldsymbol{\omega_k}) &\overset{\text{def}}{=} \sum_{l=1}^{n_\mu} \eta_\mathrm{T}(l)\ \omega_{k,l}. \end{aligned} \tag{5.33}$$

Thereby, the disjuncts are aggregated into one set of constraints.

Outer Convexification

The OC approach was described in Section 3.6. Here, it replaces any discretization of an ODE with the convex combination of right-hand sides, where the gear choice is fixed. This results in the ODE system with $0 \leqslant k \leqslant n_c$ and $0 \leqslant j \leqslant l-1$:

$$\begin{aligned} v_{k,j+1} &= v_{k,j} + \frac{h_s}{m\ v_{k,j+1}} \left(\sum_{i=1}^{n_\mu} M_{\mathrm{int},k,j+1,i} + M_{\mathrm{ext},k,j+1} \right)\ \omega_{k,i}, \\ Q_{k,j+1} &= Q_{k,j} + \frac{h_s}{v_{k,j+1}} \sum_{i=1}^{n_\mu} Q_\mathrm{fuel}\left(n_{\mathrm{eng},k,j+1,i}, M_{\mathrm{ind},k}\right)\ \omega_{k,i}, \end{aligned} \tag{5.34}$$

where the auxiliary terms are defined as

$$\begin{aligned} n_{\mathrm{eng},k,j,i} &= \frac{i_\mathrm{A}\ i_\mathrm{T}(i)}{2\ \pi\ r_\mathrm{stat}}\ 60\ v_{k,j}, \\ M_{\mathrm{int},k,j,i} &= \frac{i_\mathrm{A}}{r_\mathrm{stat}} \left(M_{\mathrm{acc},k,i} - M_{\mathrm{brk},k,j,i} \right), \\ M_{\mathrm{brk},k,j,i} &= M_{\mathrm{EB},k} + i_\mathrm{T}(i)\ M_\mathrm{fric}\left(n_{\mathrm{eng},k,j,i}\right), \\ M_{\mathrm{acc},k,i} &= i_\mathrm{T}(i)\ \eta_\mathrm{T}(i)\ M_{\mathrm{ind},k}. \end{aligned} \tag{5.35}$$

The path and control constraints are also aggregated by usage of the convex combination:

$$\begin{aligned} M_{\mathrm{ind},k} &\in \left[0, \sum_{i=1}^{n_\mu} M_\mathrm{ind}^\mathrm{max}\left(n_{\mathrm{eng},k,0,i}\right)\ \omega_{k,i} \right], \\ \sum_{i=1}^{n_\mu} n_{\mathrm{eng},k,0,i}\ \omega_{k,i} &\in \left[n_\mathrm{eng}^\mathrm{min}, n_\mathrm{eng}^\mathrm{max} \right]. \end{aligned} \tag{5.36}$$

Big-M formulation

The Big-M constraint approach was described in Section 3.7. We can only relax the constraint formulation with this approach since we do not want to reformulate the ODEs to two inequalities, which are then relaxed, but we want to remain tight in the ODE sense. The ODE is reformulated with the OC approach (5.34), (5.35). The bounds on the engine speeds are derived from the linear dependence between engine speed and the gear's transmission ratio $i_T(i)$ and the maximum transmission ratio $i_T(1)$ and minimum transmission ratio $i_T(n_\mu)$. The bound on the indicated engine torque is simply the constant offset of the corresponding nonlinear function, since the quadratic part has the upper bound 0. The derived bounds that hold for all gears and that represent the Big-Ms in this case are

$$\begin{aligned} n_{\text{eng},k,0,i} &\geqslant n_{\text{eng}}^{\min} \frac{i_T(i)}{i_T(1)}, \\ n_{\text{eng},k,0,i} &\leqslant n_{\text{eng}}^{\max} \frac{i_T(i)}{i_T(n_\mu)}, \\ M_{\text{ind},k} &\leqslant M_{\text{ind}}^{\max}. \end{aligned}$$

With these constants, the constraints are reformulated for $0 \leqslant k \leqslant n_c$ and $1 \leq i \leq n_\mu$ to be:

$$\begin{aligned} M_{\text{ind},k} &\geqslant 0, \\ M_{\text{ind},k} &\leqslant M_{\text{ind}}^{\max} - \omega_{k,i}\, c_{\text{ind}} \left(n_{\text{eng},k,0,i} - n_{\text{eng}}^{\text{best}} \right)^2, \\ n_{\text{eng},k,0,i} &\geqslant n_{\text{eng}}^{\min} \frac{i_T(i)}{i_T(1)} + \omega_{k,i} \left(n_{\text{eng}}^{\min} - n_{\text{eng}}^{\min} \frac{i_T(i)}{i_T(1)} \right), \\ n_{\text{eng},k,0,i} &\leqslant n_{\text{eng}}^{\max} \frac{i_T(i)}{i_T(n_\mu)} + \omega_{k,i} \left(n_{\text{eng}}^{\max} - n_{\text{eng}}^{\max} \frac{i_T(i)}{i_T(n_\mu)} \right). \end{aligned} \tag{5.37}$$

Vanishing constraints

The vanishing constraint approach was described in Section 3.8. As already described there, using a complementarity formulation for the ODE system produces nonlinear problems, which are very difficult to solve, and we had no solver at hand that could handle the arising systems. Therefore, the ODEs of the disjunctions are handled with the OC technique, the corresponding equations are (5.34), (5.35). The disjunctions' path and control constraints are reformulated with the vanishing constraint technique to be

$$\begin{aligned} \left(M_{\text{ind}}^{\max} - M_{\text{ind},k} \left(n_{\text{eng},k,0,i} \right) \right)\ \omega_{k,i} &\geqslant 0, \\ \left(n_{\text{eng}}^{\max} - n_{\text{eng},k,0,i} \right)\ \omega_{k,i} &\geqslant 0, \\ \left(n_{\text{eng},k,0,i} - n_{\text{eng}}^{\min} \right)\ \omega_{k,i} &\geqslant 0. \end{aligned} \tag{5.38}$$

As this formulation has special numerical disadvantages, standard solvers cannot be expected to solve the problem, which can also be seen for this formulation. IPOPT's barrier method directly cuts off the lower arc of the vanishing constraints and hence cannot find a feasible solution. Therefore, a relaxed formulation and an additionally smoothed formulation are also tested. The relaxed formulation is given with a relaxation parameter $\varepsilon > 0$ to be

$$\begin{aligned} \left(M_{\text{ind}}^{\max} - M_{\text{ind},k}\left(n_{\text{eng},k,0,i}\right)\right)\ \omega_{k,i} &\geqslant -\varepsilon, \\ \left(n_{\text{eng}}^{\max} - n_{\text{eng},k,0,i}\right)\ \omega_{k,i} &\geqslant -\varepsilon, \\ \left(n_{\text{eng},k,0,i} - n_{\text{eng}}^{\min}\right)\ \omega_{k,i} &\geqslant -\varepsilon. \end{aligned} \tag{5.39}$$

The smoothed variant is with the smoothing and relaxing parameter $\varepsilon > 0$

$$\begin{aligned} \varphi_{\varepsilon}^{\text{VC}}\left(M_{\text{ind}}^{\max}\left(n_{\text{eng},k,0,i}\right) - M_{\text{ind},k}, \omega_{k,i}\right) &\geqslant -\varepsilon, \\ \varphi_{\varepsilon}^{\text{VC}}\left(n_{\text{eng}}^{\max} - n_{\text{eng},k,0,i}, \omega_{k,i}\right) &\geqslant -\varepsilon, \\ \varphi_{\varepsilon}^{\text{VC}}\left(n_{\text{eng},k,0,i} - n_{\text{eng}}^{\min}, \omega_{k,i}\right) &\geqslant -\varepsilon, \end{aligned} \tag{5.40}$$

with the NCP function

$$\varphi_{\varepsilon}^{\text{VC}}(a,b) \overset{\text{def}}{=} \frac{1}{2}\left(\sqrt{a^2b^2+\varepsilon^2} - ab + \sqrt{b^2+\varepsilon^2} - b\right),$$

which also gives favorable theoretical results.

Both methods need to drive $\varepsilon \searrow 0$ and hence are employed in the homotopy Algorithm 5.2.

Algorithm 5.2: Homotopy method for problems with constraints (5.39) or (5.40).

Data: problem formulation and initial point
Result: solution σ^* of problem that satisfies constraints (5.39) or (5.40) with $\varepsilon \leqslant 10^{-4}$

$\delta = \frac{3}{5}$, $\varepsilon^* = 10^5$, set σ^* to initial point,
while $\varepsilon^* > 10^{-4}$ **do**
  $\varepsilon = \delta\varepsilon^*$
  solve the problem corresponding to ε starting from last solution σ^*
  if terminal point infeasible, or cannot restore feasibility of the initial point **then**
    $\delta = \frac{8}{5}\,\delta$
  else
    store solution as σ^*, $\varepsilon^* = \varepsilon$, $\delta = \frac{5}{6}\,\delta$.
  end
end

Perspective formulation

The perspective function approach is described in Section 3.9. It can be applied to both the ODE system and the constraints. All states and controls that are present in the disjunction are duplicated for each disjunct. The original variables are replaced by their sums and the formulation has to ensure that all variables belonging to one disjunct are driven to zero in linear dependence on the corresponding binary variable $\omega_{k,i}$. We introduce the lifted versions of the states v, Q with three indices and their aggregated formulation on the control grid with one index only, which are needed to properly describe the objective. The lifted controls M_{ind}, M_{brk} have two indices whereas their aggregated versions have only one. The other variables are auxiliary variables and are adjusted according to the needs. Notice that all constraints of the disjunction were reformulated with perspective functions and the integrality condition is already relaxed for $\boldsymbol{\omega}$.

$$\min \quad \lambda_{\text{dev}}\Phi_{\text{dev}} + \lambda_{\text{fuel}}\Phi_{\text{fuel}} + \lambda_{\text{comf}}\Phi_{\text{comf}} \tag{5.41}$$

$$\begin{aligned}
\text{s.t.} \quad & v_{k,j+1,i} = v_{k,j,i} + \frac{h_s}{m\, v_{k,j+1,i}} \left(M_{\text{int},k,j+1,i} + M_{\text{ext},k,j+1,i}\right)\, \omega_{k,i}, \\
& Q_{k,j+1,i} = Q_{k,j,i} + \frac{h_s}{v_{k,j+1,i}}\, Q_{\text{fuel}}\left(\frac{n_{\text{eng},k,j+1,i}}{\omega_{k,i}}, \frac{M_{\text{ind},k,i}}{\omega_{k,i}}\right)\, \omega_{k,i}^2, \\
& \qquad\qquad 0 \leqslant j \leqslant l-1, 0 \leqslant k \leqslant n_c, 1 \leqslant i \leqslant n_\mu, \\
& n_{\text{eng},k,j,i} = \frac{i_{\text{A}}\, i_{\text{T}}(i)}{2\,\pi\, r_{\text{stat}}}\, 60\, v_{k,j,i}, \\
& M_{\text{int},k,j,i} = \frac{i_{\text{A}}}{r_{\text{stat}}}\left(M_{\text{acc},k,i} - M_{\text{brk},k,j,i}\right), \\
& M_{\text{brk},k,j,i} = M_{\text{EB},k,i} + i_{\text{T}}(i)\, M_{\text{fric}}\left(\frac{n_{\text{eng},k,j+1,i}}{\omega_{k,i}}\right)\, \omega_{k,i}, \\
& M_{\text{ext},k,j,i} = -M_{\text{air},k,j,i} - M_{\text{road},k,i}, \\
& M_{\text{air},k,j,i} = \frac{1}{2} c_{\text{w}}\, A\, \varrho_{\text{air}}\, v_{k,j,i}^2, \\
& n_{\text{eng},k,j,i} \in \left[n_{\text{eng}}^{\min}, n_{\text{eng}}^{\max}\right]\, \omega_{k,i}, \\
& v_{k,j,i} \in \left[0, v_{\text{track},k}\right]\, \omega_{k,i}, \\
& \qquad\qquad 0 \leqslant j \leqslant l, 0 \leqslant k \leqslant n_c, 1 \leqslant i \leqslant n_\mu, \\
& M_{\text{acc},k,i} = i_{\text{T}}(i)\, \eta_{\text{T}}(i)\, M_{\text{ind},k,i}, \\
& M_{\text{road},k,i} = m\, g\, \left(\sin\gamma_k + f_{\text{r}}\, \cos\gamma_k\right)\, \omega_{k,i}, \\
& M_{\text{ind},k,i} \in \left[0, M_{\text{ind}}^{\max}\left(\frac{n_{\text{eng},k,0,i}}{\omega_{k,i}}\right)\right]\, \omega_{k,i},
\end{aligned}$$

$$
\begin{aligned}
M_{\mathrm{EB},k,i} &\in \left[0, M_{\mathrm{brk}}^{\max}\right] \ \omega_{k,i}, \\
\omega_{k,i} &\in [0,1], && 0 \leqslant k \leqslant n_c, 1 \leqslant i \leqslant n_\mu, \\
M_{\mathrm{ind},k} &= \sum_{i=1}^{n_\mu} M_{\mathrm{ind},k,i}, \\
M_{\mathrm{brk},k} &= \sum_{i=1}^{n_\mu} M_{\mathrm{brk},k,i}, \\
v_k &= \sum_{i=1}^{n_\mu} v_{k,0,i}, \\
Q_k &= \sum_{i=1}^{n_\mu} Q_{k,0,i}, \\
1 &= \sum_{i=1}^{n_\mu} \omega_{k,i}, && 0 \leqslant k \leqslant n_c, \\
v_{k+1} &= \sum_{i=1}^{n_\mu} v_{k,l,i}, && 0 \leqslant k \leqslant n_c - 1, \\
Q_{k+1} &= \sum_{i=1}^{n_\mu} Q_{k,l,i}, && 0 \leqslant k \leqslant n_c - 1, \\
v_0 &= v_{\mathrm{start}}, \\
Q_0 &= Q_{\mathrm{start}}, .
\end{aligned}
$$

The formulation is numerically instable if $v_{k,j+1,i} = 0$ since there is a division by this term in both ODEs. For the other models this does not happen, since a velocity of 0 will usually not be desired. However, for the non-active disjuncts, the lifted variables are all set to 0 and hence the division $\frac{0}{0}$ will happen in this model. The divisor $v_{k,j+1,i}$ is replaced in the ODE discretization with $(1-\varepsilon)\ v_{k,j,i} + \varepsilon$ to circumvent this behavior.

This formulation is still numerically instable due to the division by 0 if $\omega_{k,i} = 0$. To circumvent this behavior, the 0-corrected formulation (3.28) has been applied in the computational model.

The results are also not very promising since the relaxation seems to give too much freedom to the optimizer, we hence also formulated two tightened versions of this formulation:

The first tightened version reformulates the problem such that only the truly necessary parts of the problem are lifted. With the notation as in Chapter 3, it is formulated to be in abstract

form

$$
\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{\omega}(\cdot)} \quad & \Phi(\boldsymbol{x}(t_\mathrm{f})) \\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}_\mathrm{disj}(t) + \boldsymbol{f}_\mathrm{indep}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \\
& \bigvee_{1 \leqslant i \leqslant n_\omega} \begin{bmatrix} \omega_i(t) = 1 \\ \boldsymbol{f}_\mathrm{disj}(t) = \boldsymbol{f}_\mathrm{dep}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i) \\ \mathbf{0} \leqslant \boldsymbol{c}(t, \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}^i) \end{bmatrix}, \\
& \boldsymbol{x}(t) \in [\boldsymbol{m}^x, \boldsymbol{M}^x], \\
& \boldsymbol{u}(t) \in [\boldsymbol{m}^u, \boldsymbol{M}^u], \\
& \boldsymbol{\omega}(t) \in \{0,1\}^{n_\omega}, \qquad\qquad t \in [t_0, t_\mathrm{f}],
\end{aligned}
$$

where the right-hand side is decomposed into a mode-dependent part and a mode-independent part $\boldsymbol{f}(\cdot) = \boldsymbol{f}_\mathrm{dep}(\cdot) + \boldsymbol{f}_\mathrm{indep}(\cdot)$. The dependent part is temporarily treated as if it was neither dependent on the states nor on the controls. Now, the perspective relaxation is applied to this problem and we obtain

$$
\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{\omega}(\cdot)} \quad & \Phi(\boldsymbol{x}(t_\mathrm{f})) && (5.42) \\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}_\mathrm{disj}(t) + \boldsymbol{f}_\mathrm{indep}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \\
& \boldsymbol{f}^i_\mathrm{disj}(t) = \boldsymbol{f}_\mathrm{dep}\left(t, \frac{\boldsymbol{x}^i(t)}{\omega_i(t)}, \frac{\boldsymbol{u}^i(t)}{\omega_i(t)}, \boldsymbol{v}^i\right) \omega_i(t), \\
& \mathbf{0} \leqslant \boldsymbol{c}\left(t, \frac{\boldsymbol{x}^i(t)}{\omega_i(t)}, \frac{\boldsymbol{u}^i(t)}{\omega_i(t)}, \boldsymbol{v}^i\right) \omega_i(t), \\
& 1 = \sum_{i=1}^{n_\omega} \omega_i(t), \\
& \boldsymbol{f}_\mathrm{disj}(t) = \sum_{i=1}^{n_\omega} \boldsymbol{f}^i_\mathrm{disj}(t), && (5.43) \\
& \boldsymbol{x}(t) = \sum_{i=1}^{n_\omega} \boldsymbol{x}^i(t), \\
& \boldsymbol{x}^i(t) \in [\boldsymbol{m}^x, \boldsymbol{M}^x]\, \omega_i(t), \\
& \boldsymbol{u}(t) = \sum_{i=1}^{n_\omega} \boldsymbol{u}^i(t), \\
& \boldsymbol{u}^i(t) \in [\boldsymbol{m}^u, \boldsymbol{M}^u]\, \omega_i(t), \\
& \boldsymbol{\omega}(t) \in [0,1]^{n_\omega}, \qquad\qquad t \in [t_0, t_\mathrm{f}],
\end{aligned}
$$

The states and controls that need to be lifted are only those that occur in either $\boldsymbol{c}(\cdot)$ or $\boldsymbol{f}_\mathrm{dep}(\cdot)$.

Eliminating the auxiliary term $\boldsymbol{f}_{\text{disj}}(t)$ through equation (5.43) directly aggregates the ODE formulation to be similar to the OC technique. One difference lies in the dependent part, here, the lifted variables occur. The main disparity to the OC formulation is in the constraints, which are formulated with the lifted variables only instead of being aggregated. This version of the problem is much easier to handle than the original perspective formulation but still provides a quite weak relaxation as can be seen in the computational results.
The second reformulation is the tightened perspective formulation as described in Section 3.9. It eliminates the lifted states with the constraint

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(t)\,\omega_i(t), \qquad t \in [t_0, t_\text{f}].$$

The result is similar to the vanishing constraint formulation of the constraints (5.38) with an OC of the ODE (5.34). The single difference is that the controls are still lifted and their lifted version appears in both the vanishing constraints as well as the right-hand side of the ODE:

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \sum_{i=1}^{n_\omega} \omega_i(t)\,\boldsymbol{f}\left(t, \boldsymbol{x}(t), \frac{\boldsymbol{u}^i(t)}{\omega_i(t)}, \boldsymbol{v}^i\right),\\ \boldsymbol{0} &\leqslant \omega_i(t)\,\boldsymbol{c}\left(t, \boldsymbol{x}(t), \frac{\boldsymbol{u}^i(t)}{\omega_i(t)}, \boldsymbol{v}^i\right).\end{aligned}$$

The vanishing constraints are smoothed as above, cf. system (5.40), and the same homotopy method 5.2 is called to drive $\varepsilon \searrow 0$.

5.4.3 Comparison of relaxations

All problems were solved with IPOPT *v.3.11* invoked from AMPL *v.20130327* using the HSL linear algebra solvers. Most problems were run with the standard solver options, only the perspective formulation (5.41) needed a larger amount of iterations in combination with a slight modification of the initial parameters to converge for scenario 1. For the homotopy methods, IPOPT warm start options were added.
The reference solutions were provided with a dynamic programming approach with a state discretization of 0.001 [m/s] for the velocity and 20 [Nm] for the engine torques M_ind and M_EB. The fuel consumption is solely dependent on the velocity and gear choices and hence is not needed to be considered for the discretization. These solutions can be guaranteed to be sufficiently close to the global, optimal solution of the problem.
The following nine problems described in the previous section were solved for both scenarios:

- IC of both the constraints and the ODE system,
- OC of both the constraints and the ODE system,
- Big-M formulation of the constraints with OC of the ODEs,

- relaxed vanishing constraints with OC of the ODE system,
- smoothed, relaxed vanishing constraints with OC of the ODEs,
- 0-corrected full perspective formulation,
- 0-corrected decomposed perspective formulation,
- tightened perspective formulation, i.e., smoothed, relaxed vanishing constraints with OC of ODE and lifted controls,
- dynamic programming to provide global, optimal solutions.

We present numerical results for two selected scenarios in Figures 5.14–5.17. The track's height profile is shown on top, followed by nine plots of the relaxed (local) optimal solutions identified by IPOPT for the described formulations and the global solution identified by BUCHNER's dynamic programming code, cf. [42].
We use four properties to analyze the formulations' behavior, which are the objective function value, the fractionality, the infeasibility and the computational time. The objective can be used to describe the tightness of the relaxation. The fractionality property is defined to be the average (over the different time steps) Manhattan norm difference from an integral point:

$$\text{fractionality} \overset{\text{def}}{=} \sum_{i=0}^{n_c-1} \frac{1}{n_c} \sum_{j=1}^{n_\mu} \left(0.5 - \left|y_{i,j} - 0.5\right|\right).$$

The infeasibility property describes the average infeasibility with regard to the constraints $\boldsymbol{c}(\cdot)$ of the disjunction and is calculated using the vanishing constraint formulation (5.38). It provides the weighted violation of the constraints for each gear, the ones for inactive modes are set to 0 since their multipliers are 0. Here, aggregation effects – also called compensatory effects – can be seen best. The vanishing constraint interpretation of the constraints also enables feasible results after the usage of rounding strategies, which only round up nonzeros. For each control interval, the summed infinity norm violations are averaged:

$$\begin{aligned}\text{infeasibility} \overset{\text{def}}{=} \sum_{k=0}^{n_c-1} \frac{1}{3\,n_c} \Bigg(& \max_{1\leqslant i\leqslant n_\mu} \left\{ y_{k,i} \left(M_{\text{ind}}^{\max}\left(n_{\text{eng},k,0,i}\right) - M_{\text{ind},k} \right) \right\} \\ & + \max_{1\leqslant i\leqslant n_\mu} \left\{ y_{k,i} \left(n_{\text{eng},k,0,i} - n_{\text{eng}}^{\min} \right) \right\} \\ & + \max_{1\leqslant i\leqslant n_\mu} \left\{ y_{k,i} \left(n_{\text{eng}}^{\max} - n_{\text{eng},k,0,i} \right) \right\} \Bigg).\end{aligned}$$

The scenarios are chosen such that the deviation of the velocity profile is prioritized during optimization. The desired velocity is set to be constant 22 m/s. From both scenarios, a clear picture emerges.

- IC favors fractional solutions but they are computed very fast. Yet, they are quite far from satisfying the vanishing constraints. The solutions combine two gears to get a maximum acceleration maintaining feasibility with regard to the aggregated constraints – these aggregations allow compensatory effects to happen.
- OC is quite fast as well and and yields reasonable approximations, which also suffer from compensatory effects but to a much lesser degree than IC.
- The Big-M formulation of constraints with OC of the ODE lies somewhere in between the IC and OC with regard to the feasibility. However, it is farther away from integral solutions and also provides a weaker relaxation than both previous formulations. Since the computation times are comparable with OC, it is dominated by the OC formulation.
- The relaxed vanishing constraint formulation succeeds in yielding feasible solutions for both scenarios. The fractionality is also very low. However, the effects of the non-convexity of the problem can be seen here very well as the provided solution is only a local solution. It is worse than the global solution found by dynamic programming. The neglect of global solutions by the usage of local solvers – as e.g. IPOPT – removes the relaxation property since only the global solution would provide a lower bound to the problem. For the computational time, around half of it is spent to solve the initial problem of the homotopy and the rest is spent driving $\varepsilon \searrow 0$ and re-solving the warm-started problems. In comparison to the previously presented formulations, the computational time is much higher.
- The relaxed and smoothed vanishing constraint formulation is quite comparable to the relaxed one but suffers even more from being stuck in a local solution while $\varepsilon \searrow 0$ and hence needs longer to compute.
- The full perspective formulation seems to be a very weak relaxation for this problem. The engine constraints can be satisfied while the highest and lowest gear are combined to allow maximum acceleration while the lowest gear is used as much as needed to hit the desired velocity profile.
- The slightly tighter decomposed variant of the perspective formulation provides a solution that looks much better, but is still inferior in all aspects to the OC formulation.
- The tightened perspective formulation needs much more computational time than the other problems described so far. However, it provides the solution that is – with regard to the integral variables – the closest one to the optimal solution. As the other formulation with vanishing constraints, also this formulation succeeds in finding a locally optimal feasible point in one case. In the other case, it is a proper relaxation. The infeasibility is due to the additional freedom of having lifted the controls. Their aggregation does not necessarily fulfill the constraints anymore.

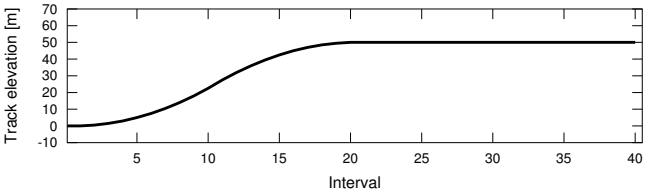
- Dynamic programming provides a global, optimal solution. However, the computational effort is several orders of magnitude higher than the effort spent to calculate the relaxations' solutions, locally.

5.4.4 Conclusions

We implemented the different relaxed formulations from Chapter 3 for MIOCPs and compared them for a truck cruise control problem in two different scenarios. The different sizes and complexities of the resulting problems let the computational times for the relaxations differ by a wide margin. The longer solution times do not necessarily lead to stronger solutions but may be a result of the large problem dimensions as for the full perspective approach. The high number of disjuncts (16) makes the full perspective formulation too lenient as well as too demanding.
The approach that gives the closest solutions to the optimal one is the tightened perspective formulation, which essentially unifies the best options in using the OC technique for the ODE while using vanishing constraints for the disjunctions' constraints and lifting the controls to be able to correctly use each single system mode. However, it takes a lot of time to compute due to the usage of smoothed vanishing constraints inside a homotopy algorithm while lifting the controls into a higher dimension and thereby enlarging the formulation.
The solution approaches for the other vanishing constraint formulations have problems overcoming the non-convexity of the formulation. Therefore, they are no true relaxations in this case. However, they provide integral or almost integral solutions, which are quite close to the optimal solution. Like the tightened perspective, these solutions take quite long to compute.
The remaining formulations have some strong disadvantages as to attracting fractional and infeasible solutions with regard to the disjunctions' constraints. However, their computational times are much smaller and the OC approach seems to do the best with regard to those two disadvantageous properties.

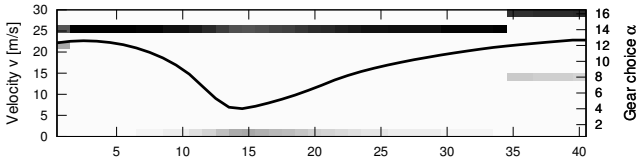
Scenario 1

$\lambda_{\text{fuel}} = 25$
$\lambda_{\text{dev}} = 1$
$\lambda_{\text{comf}} = 1$



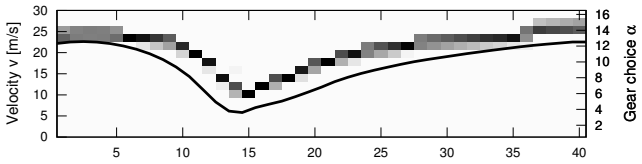
Inner Convexification

Constraints	(5.33)
Objective	54389
Fractionality	0.18
Infeasibility	$5.8e^{+02}$
Runtime	1.68 sec



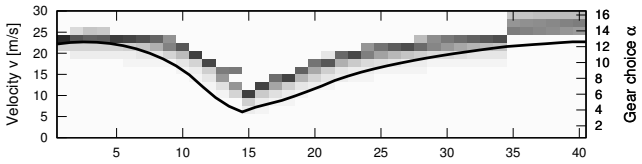
Outer Convexification

Constraints	(5.34),(5.35),(5.36)
Objective	58749.4
Fractionality	0.4
Infeasibility	1.8
Runtime	8.68 sec



Big-M

Constraints	(5.34),(5.35),(5.37)
Objective	53235.4
Fractionality	0.68
Infeasibility	46
Runtime	9.04 sec



Relaxed Vanishing Constraints

Constraints	(5.34),(5.35),(5.39)
Objective	63807.6
Fractionality	0.0095
Infeasibility	$1.6e^{-05}$
Runtime	35.4 sec

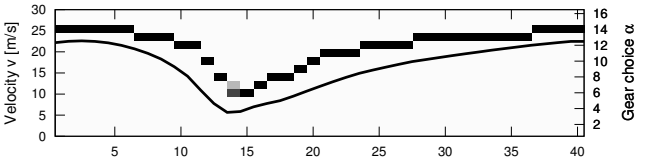
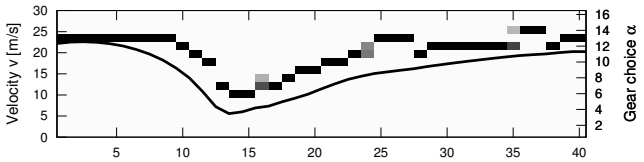


Figure 5.14: Relaxed gear choices $\boldsymbol{y}$ (reflected by the intensity of gray) and corresponding velocities v for scenario 1.

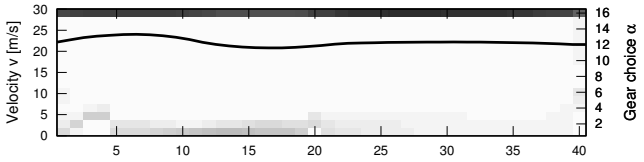
Smoothed Vanishing Constraints

Constraints	(5.34),(5.35),(5.40)
Objective	68268.3
Fractionality	0.044
Infeasibility	0
Runtime	83.3 sec



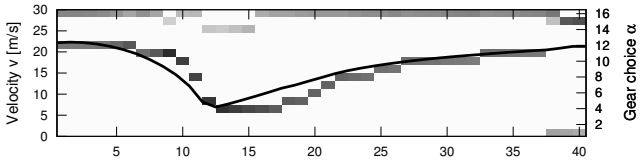
Full perspective

Constraints	(5.41)
Objective	1384.16
Fractionality	0.35
Infeasibility	$3.4e^{+03}$
Runtime	50.4 sec



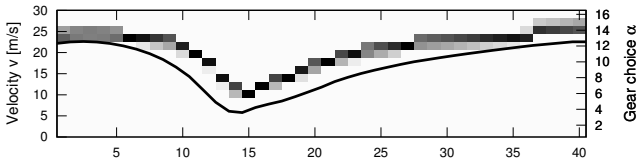
Decomposed perspective

Constraints	(5.42)
Objective	50794.7
Fractionality	0.679503
Infeasibility	$9.5e^{+03}$
Runtime	12.4 sec



Tightened perspective

Constraints	(5.34),(5.35),(5.40) with lifted controls
Objective	59102
Fractionality	0.35
Infeasibility	1.6
Runtime	318 sec



Dynamic programming

Constraints	integer formulation
Objective	58703.7
Fractionality	0
Infeasibility	0
Runtime	$\sim$ 4 days

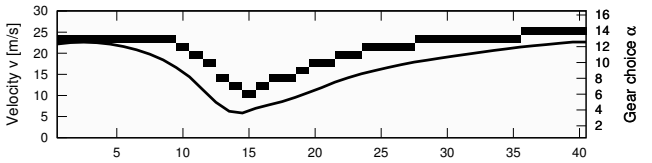
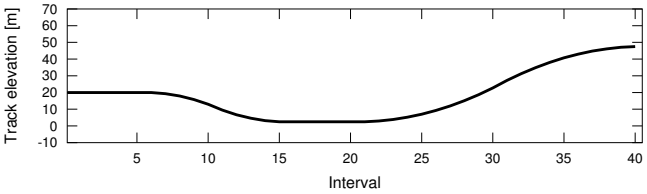


Figure 5.15: Relaxed gear choices $\boldsymbol{y}$ (reflected by the intensity of gray) and corresponding velocities v for scenario 1.

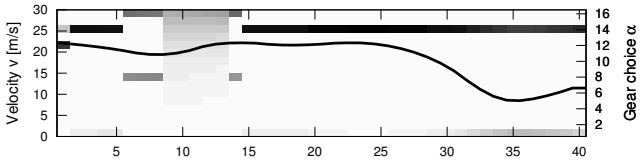
Scenario 2

$\lambda_{\text{fuel}} = 10$
$\lambda_{\text{dev}} = 100$
$\lambda_{\text{comf}} = 1$



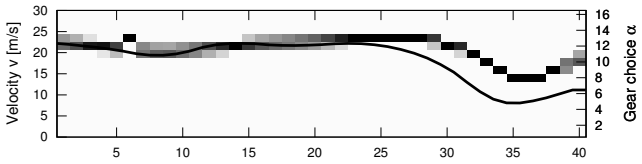
Inner Convexification

Constraints	(5.33)
Objective	3821850
Fractionality	0.31
Infeasibility	$8.2e^{+02}$
Runtime	2.46 sec



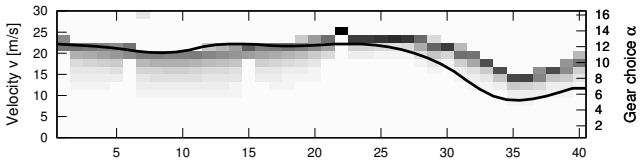
Outer Convexification

Constraints	(5.34),(5.35),(5.36)
Objective	4054180
Fractionality	0.31
Infeasibility	6.7
Runtime	10.6 sec



Big-M

Constraints	(5.34),(5.35),(5.37)
Objective	3608030
Fractionality	0.73
Infeasibility	$1.9e^{+02}$
Runtime	9.91 sec



Relaxed Vanishing Constraints

Constraints	(5.34),(5.35),(5.39)
Objective	4111570
Fractionality	0
Infeasibility	$1.5e^{-05}$
Runtime	53.7 sec

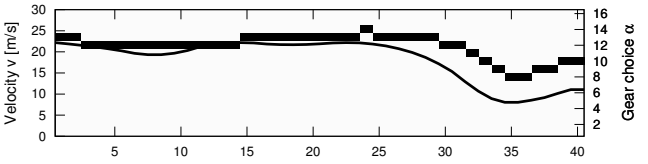
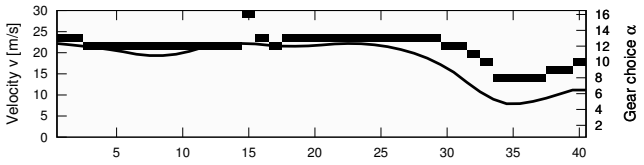


Figure 5.16: Relaxed gear choices $\boldsymbol{y}$ (reflected by the intensity of gray) and corresponding velocities v for scenario 2.

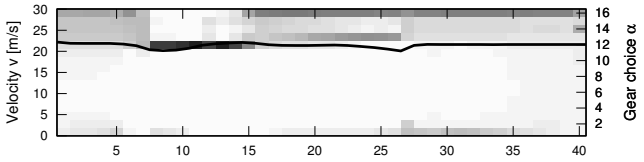
Smoothed Vanishing Constraints

Constraints	(5.34),(5.35),(5.40)
Objective	4135250
Fractionality	0
Infeasibility	$1e^{-06}$
Runtime	124 sec



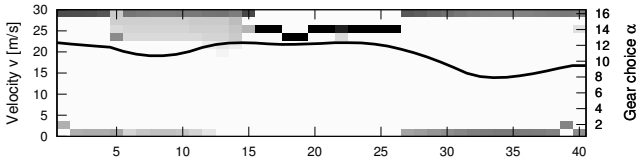
Full perspective

Constraints	(5.41)
Objective	96291.4
Fractionality	0.87
Infeasibility	$2.4e^{+03}$
Runtime	3358 sec



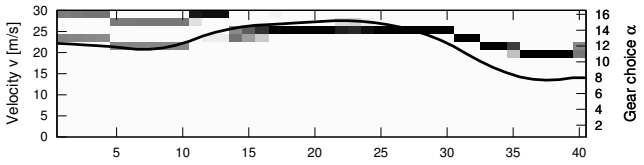
Decomposed perspective

Constraints	(5.42)
Objective	1592100
Fractionality	0.6
Infeasibility	$4e^{+03}$
Runtime	124 sec



Tightened perspective

Constraints	(5.34),(5.35),(5.40) with lifted controls
Objective	2100620
Fractionality	0.32
Infeasibility	1
Runtime	345 sec



Dynamic programming

Constraints	integer formulation
Objective	4008200
Fractionality	0
Infeasibility	0
Runtime	$\sim$ 3 days

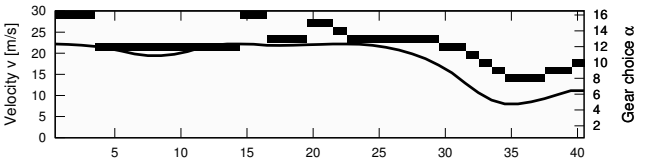


Figure 5.17: Relaxed gear choices $\boldsymbol{y}$ (reflected by the intensity of gray) and corresponding velocities v for scenario 2.

Bibliography

[1] P. Abichandani, H. Benson, and M. Kam, *Multi-vehicle path coordination under communication constraints*, in American Control Conference, 2008, pp. 650–656.

[2] T. Achterberg, T. Koch, and A. Martin, *Branching rules revisited*, Operations Research Letters, 33 (2005), pp. 42–54.

[3] W. Achtziger and C. Kanzow, *Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications*, Mathematical Programming Series A, 114 (2008), pp. 69–99.

[4] M. Alamir and S. Attia, *On solving optimal control problems for switched hybrid nonlinear systems by strong variations algorithms*, in 6th IFAC Symposium, NOLCOS, Stuttgart, Germany, 2004.

[5] 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.

[6] J. Albersmeyer and M. Diehl, *The Lifted Newton method and its application in optimization*, SIAM Journal on Optimization, 20 (2010), pp. 1655–1684.

[7] P. Antsaklis and X. Koutsoukos, *On hybrid control of complex systems: A survey*. In 3rd International Conference ADMP'98, Automation of Mixed Processes: Dynamic Hybrid Systems, March 1998.

[8] S. Attia, M. Alamir, and C. Canudas de Wit, *Sub optimal control of switched nonlinear systems under location and switching constraints*, in IFAC World Congress, 2005.

[9] E. Balas and M. Perregaard, *Lift and project for mixed 0-1 programming: Recent progress*, tech. rep., MSRR No. 627, Graduate School of Industrial Administration, Carnegie Mellon University, 1999.

[10] 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.

[11] R. BARTLETT, A. WÄCHTER, AND L. BIEGLER, *Active set vs. interior point strategies for model predicitve control*, in Proceedings of the American Control Conference, Chicago, IL, 2000, pp. 4229–4233.

[12] P. BARTON, *The modelling and simulation of combined discrete/continuous processes*, PhD thesis, Department of Chemical Engineering, Imperial College of Science, Technology and Medicine, London, 1992.

[13] I. BAUER, *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik*, PhD thesis, Universität Heidelberg, 1999.

[14] B. BAUMRUCKER AND L. BIEGLER, *MPEC strategies for optimization of a class of hybrid dynamic systems*, Journal of Process Control, 19 (2009), pp. 1248–1256. Special Section on Hybrid Systems: Modeling, Simulation and Optimization.

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

[16] R. BELLMAN, *Dynamic Programming*, University Press, Princeton, N.J., 6th ed., 1957. ISBN 0-486-42809-5.

[17] P. BELOTTI, C. KIRCHES, S. LEYFFER, J. LINDEROTH, J. LUEDTKE, AND A. MAHAJAN, *Mixed-integer nonlinear optimization*, in Acta Numerica, A. Iserles, ed., vol. 22, Cambridge University Press, 2013, pp. 1–131.

[18] A. BEMPORAD AND M. MORARI, *Control of systems integrating logic, dynamics, and constraints*, Automatica, 35 (1999), pp. 407–427.

[19] S. BENGEA AND R. DECARLO, *Optimal control of switching systems*, Automatica, 41 (2005), pp. 11–27.

[20] D. BERTSEKAS, *Dynamic programming and optimal control, Volume 1*, Athena Scientific, Belmont, Mass., 3. ed., 2005.

[21] D. BERTSEKAS, *Dynamic programming and optimal control, Volume 2*, Athena Scientific, Belmont, Mass., 4. ed., 2012.

[22] J. BETTS, *Trajectory Optimization using Sparse Sequential Quadratic Programming*, vol. 111 of International Series of Numerical Mathematics, Birkhäuser Verlag, 1993, pp. 115–128.

[23] ——, *Practical Methods for Optimal Control Using Nonlinear Programming*, SIAM, Philadelphia, 2001.

[24] J. BETTS AND W. HUFFMAN, *Mesh refinement in direct transcription methods for optimal control*, Optimal Control Applications and Methods, 19 (1998), pp. 1–21.

[25] ——, *Exploiting sparsity in the direct transcription method for optimal control*, Computational Optimization and Applications, 14 (1999), pp. 179–201.

[26] L. BIEGLER, *Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation*, Computers & Chemical Engineering, 8 (1984), pp. 243–248.

[27] ——, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*, Series on Optimization, SIAM, 2010.

[28] H. BOCK, *Numerical treatment of inverse problems in chemical reaction kinetics*, in Modelling of Chemical Reaction Systems, K. Ebert, P. Deuflhard, and W. Jäger, eds., vol. 18 of Springer Series in Chemical Physics, Springer, Heidelberg, 1981, pp. 102–125.

[29] ——, *Recent advances in parameter identification techniques for ODE*, in Numerical Treatment of Inverse Problems in Differential and Integral Equations, P. Deuflhard and E. Hairer, eds., Birkhäuser, Boston, 1983, pp. 95–121.

[30] ——, *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, vol. 183 of Bonner Mathematische Schriften, Universität Bonn, 1987.

[31] H. BOCK, T. CARRARO, W. JÄGER, S. KÖRKEL, R. RANNACHER, AND J. SCHLÖDER, eds., *Model Based Parameter Estimation: Theory and Applications*, vol. 4 of Contributions in Mathematical and Computational Sciences, Springer, 2013.

[32] H. BOCK AND R. LONGMAN, *Computation of optimal controls on disjoint control sets for minimum energy subway operation*, in Proceedings of the American Astronomical Society. Symposium on Engineering Science and Mechanics, Taiwan, 1982.

[33] H. BOCK AND K. PLITT, *A Multiple Shooting algorithm for direct solution of optimal control problems*, in Proceedings of the 9th IFAC World Congress, Budapest, 1984, Pergamon Press, pp. 242–247.

[34] P. BONAMI, *Lift-and-project cuts for mixed integer convex programs*, in Proceedings of IPCO 2011, O. Günlük and G. Woeginger, eds., vol. 6655 of Lecture Notes in Computer Science, Berlin Heidelberg, 2011, Springer-Verlag, pp. 52–64.

[35] P. BONAMI, G. CORNUEJOLS, A. LODI, AND F. MARGOT, *Feasibility pump for mixed integer nonlinear programs*, Mathematical Programming, 199 (2009), pp. 331–352.

[36] P. BONAMI, M. KILINÇ, AND J. LINDEROTH, *Algorithms and software for convex mixed integer nonlinear programs*, in Mixed Integer Nonlinear Programming, J. Lee and S. Leyffer, eds., vol. 154 of The IMA Volumes in Mathematics and its Applications, Springer New York, 2012, pp. 1–39.

[37] P. BONAMI, J. LEE, S. LEYFFER, AND A. WÄCHTER, *More branch-and-bound experiments in convex nonlinear integer programming*, Preprint ANL/MCS-P1949-0911, Mathematics and Computer Science Division, Argonne National Laboratory, September 2011.

[38] U. BRANDT-POLLMANN, *Numerical solution of optimal control problems with implicitly defined discontinuities with applications in engineering*, PhD thesis, Universität Heidelberg, 2004.

[39] M. BRANICKY, V. BORKAR, AND S. MITTER, *A Unified Framework for Hybrid Control: Model and Optimal Control*, IEEE Transactions on Automatic Control, 43 (1999), pp. 31–45.

[40] K. BRENAN, S. CAMPBELL, AND L. PETZOLD, *Numerical solution of initial-value problems in differential-algebraic equations*, SIAM, Philadelphia, 1996. Classics in Applied Mathematics 14.

[41] A. BRYSON AND Y.-C. HO, *Applied Optimal Control*, Wiley, New York, 1975.

[42] A. BUCHNER, *Auf Dynamischer Programmierung basierende nichtlineare modellprädiktive Regelung für LKW*, Diploma thesis, Ruprecht-Karls-Universität Heidelberg, January 2010.

[43] J. BURGSCHWEIGER, B. GNÄDIG, AND M. STEINBACH, *Optimization models for operative planning in drinking water networks*, Optimization and Engineering, 10 (2008), pp. 43–73.

[44] ——, *Nonlinear programming techniques for operative planning in large drinking water networks*, The Open Applied Mathematics Journal, 3 (2009), pp. 1–16.

[45] C. BÜSKENS, *Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen*, PhD thesis, Westfälische Wilhelms-Universität Münster, 1998.

[46] S. CAMPBELL AND C. GEAR, *The index of general nonlinear DAEs*, Numerische Mathematik, 72 (1995), pp. 173–196.

[47] G. CEMBRANO, J. QUEVEDO, M. SALAMERO, V. PUIG, J. FIGUERAS, AND J. MARTÍ, *Optimal control of urban drainage systems: a case study*, Control Engineering Practice, 12 (2004), pp. 1–9.

[48] S. Ceria and J. Soares, *Convex programming for disjunctive convex optimization*, Mathematical Programming A, 86 (1999), pp. 595–614.

[49] B. Chen and P. Harker, *Smooth approximations to nonlinear complementarity problems*, SIAM Journal on Optimization, 7 (1997), pp. 403–420.

[50] Y. Chen and M. Florian, *The nonlinear bilevel programming problem: Formulations, regularity, and optimality conditions*, Optimization, 32 (1995), pp. 193–209.

[51] T. Christof and A. Löbel, *PORTA – POlyhedron Representation Transformation Algorithm*. http://www.zib.de/Optimization/Software/Porta/. PORTA Homepage.

[52] T. Christof and G. Reinelt, *Combinatorial optimization and small polytopes*, TOP, 4 (1996), pp. 1 – 53.

[53] B. Colson, P. Marcotte, and G. Savard, *Bilevel programming: A survey*, A Quarterly Journal of Operations Research, 3 (2005), pp. 87–107.

[54] S. Dempe, *Foundations of Bilevel Programming*, Kluwer Academic Publishers, 2002.

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

[56] B. Duran, M. Jung, C. Ocampo-Martinez, S. Sager, and G. Cembranoa, *Minimization of sewage network overflow*. Accepted in Water Resources Management, 2013.

[57] M. Duran and I. Grossmann, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Mathematical Programming, 36 (1986), pp. 307–339.

[58] M. Egerstedt, Y. Wardi, and H. Axelsson, *Transition-time optimization for switched-mode dynamical systems*, IEEE Transactions on Automatic Control, 51 (2006), pp. 110–115.

[59] M. Engelhart, J. Funke, and S. Sager, *A decomposition approach for a new test-scenario in complex problem solving*, Journal of Computational Science, 4 (2013), pp. 245–254.

[60] F. Facchinei, H. Jiang, and L. Qi, *A smoothing method for mathematical programs with equilibrium constraints*, Mathematical Programming, 85 (1999), pp. 107–134.

[61] W. Feehery, J. Tolsma, and P. Barton, *Efficient sensitivity analysis of large-scale differential-algebraic systems*, Applied Numerical Mathematics, 25 (1997), pp. 41–54.

[62] E. Fehlberg, *Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme*, Computing, 6 (1970), pp. 61–71.

[63] M. FERRIS AND C. KANZOW, *Complementarity and related problems: A survey*, Mathematical Programming Technical Report 98–17, Department of Computer Science, University of Wisconsin–Madison, November 1998.

[64] R. FLETCHER, S. LEYFFER, D. RALPH, AND S. SCHOLTES, *Local convergence of SQP methods for mathematical programs with equilibrium constraints*, SIAM Journal on Optimization, 17 (2006), pp. 259 – 286.

[65] C. FLOUDAS, *Nonlinear and Mixed-Integer Optimization - Fundamentals and Applications*, Topics in Chemical Engineering, University Press, Oxford, 1995.

[66] C. FLOUDAS AND C. GOUNARIS, *A review of recent advances in global optimization*, Journal of Global Optimum, 45 (2009), pp. 3–38.

[67] J. FORREST AND J. TOMLIN, *Branch and bound, integer, and non-integer programming*, Annals of Operations Research, 149 (2007), pp. 81–87.

[68] A. FÜGENSCHUH, M. HERTY, A. KLAR, AND A. MARTIN, *Combinatorial and continuous models for the optimization of traffic flows on networks*, SIAM Journal on Optimization, 16 (2006), pp. 1155–1176.

[69] A. FULLER, *Study of an optimum nonlinear control system*, Journal of Electronics and Control, 15 (1963), pp. 63–71.

[70] M. GELORMINO AND N. RICKER, *Model-predictive control of a combined sewer system*, International Journal of Control, 59 (1994), pp. 793–816.

[71] A. GEOFFRION, *Duality in nonlinear programming: A simplified applications-oriented development*, SIAM Review, 13 (1971), pp. 1–37.

[72] ——, *Generalized Benders decomposition*, Journal of Optimization Theory and Applications, 10 (1972), pp. 237–260.

[73] ——, *Approaches to integer programming*, North-Holland Pub Co, 1974, ch. Lagrangean Relaxation for Integer Programming, pp. 82–114.

[74] M. GERDTS, *Numerische Methoden optimaler Steuerprozesse mit differential-algebraischen Gleichungssystemen höheren Indexes und ihre Anwendungen in der Kraftfahrzeugsimulation und Mechanik*, PhD thesis, Universität Bayreuth, 2001.

[75] ——, *Direct shooting method for the numerical solution of higher index dae optimal control problems*, Journal of Optimization Theory and Applications, 117 (2003), pp. 267–294.

[76] ——, *Solving mixed-integer optimal control problems by Branch&Bound: A case study from automobile test-driving with gear shift*, Optimal Control Applications and Methods, 26 (2005), pp. 1–18.

[77] ——, *Optimal Control of Ordinary Differential Equations and Differential-Algebraic Equations*, Habilitation, University of Bayreuth, 2006.

[78] ——, *A variable time transformation method for mixed-integer optimal control problems*, Optimal Control Applications and Methods, 27 (2006), pp. 169–182.

[79] M. GERDTS AND S. SAGER, *Mixed-integer DAE optimal control problems: Necessary conditions and bounds*, in Control and Optimization with Differential-Algebraic Constraints, L. Biegler, S. Campbell, and V. Mehrmann, eds., SIAM, 2012, pp. 189–212.

[80] S. GÖTTLICH, M. HERTY, C. KIRCHNER, AND A. KLAR, *Optimal control for continuous supply network models*, Networks and Heterogenous Media, 1 (2007), pp. 675–688.

[81] M. GRÄBER, *Energieoptimale Regelung von Kälteprozessen*, PhD thesis, Technische Universität Carolor-Wilhelmina zu Braunschweig, 2013.

[82] M. GRÄBER, C. KIRCHES, H. BOCK, J. SCHLÖDER, W. TEGETHOFF, AND J. KÖHLER, *Determining the optimum cyclic operation of adsorption chillers by a direct method for periodic optimal control*, International Journal of Refrigeration, 34 (2011), pp. 902–913.

[83] A. GRIEWANK, *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*, no. 19 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 2000.

[84] I. GROSSMANN, *Review of nonlinear mixed-integer and disjunctive programming techniques*, Optimization and Engineering, 3 (2002), pp. 227–252.

[85] I. GROSSMANN, P. AGUIRRE, AND M. BARTTFELD, *Optimal synthesis of complex distillation columns using rigorous models*, Computers & Chemical Engineering, 29 (2005), pp. 1203–1215.

[86] I. GROSSMANN AND S. LEE, *Generalized disjunctive programming: Nonlinear convex hull relaxation and algorithms*, Computational Optimization and Applications, 26 (2003), pp. 83–100.

[87] I. GROSSMANN AND J. RUIZ, *Generalized disjunctive programming: A framework for formulation and alternative algorithms for MINLP optimization*, in Mixed Integer Nonlinear Programming, J. Lee and S. Leyffer, eds., vol. 154 of The IMA Volumes in Mathematics and its Applications, Springer, 2012, ch. 4, pp. 93–116.

[88] O. GÜNLÜK AND J. LINDEROTH, *Perspective reformulations of mixed integer nonlinear programs with indicator variables*, Mathematical Programming B, 124 (2010), pp. 183–205.

[89] E. HAIRER, C. LUBICH, AND M. ROCHE, *The numerical solution of differential-algebraic systems by Runge-Kutta methods*, no. 1409 in Lecture Notes in Mathematics, Springer, Heidelberg, 1989.

[90] C. HARGRAVES AND S. PARIS, *Direct trajectory optimization using nonlinear programming and collocation*, AIAA J. Guidance, 10 (1987), pp. 338–342.

[91] R. HARTL, S. SETHI, AND R. VICKSON, *A survey of the maximum principles for optimal control problems with state constraints*, SIAM Review, 37 (1995), pp. 181–218.

[92] W. HEEMELS, B. DE SCHUTTER, AND A. BEMPORAD, *Equivalence of hybrid dynamical models*, Automatica, 37 (2001), pp. 1085–1091.

[93] E. HELLSTRÖM, M. IVARSSON, J. ASLUND, AND L. NIELSEN, *Look-ahead control for heavy trucks to minimize trip time and fuel consumption*, Control Engineering Practice, 17 (2009), pp. 245–254.

[94] M. HESTENES, *Calculus of variations and optimal control theory*, Wiley, New York, 1966.

[95] H. HIJAZI, P. BONAMI, G. CORNUEJOLS, AND A. OUOROU, *Mixed-integer nonlinear programs featuring "on/off" constraints*, Computational Optimization and Applications, 52 (2009), pp. 537–558.

[96] T. HOHEISEL, *Mathematical Programs with Vanishing Constraints*, PhD thesis, Julius–Maximilians–Universität Würzburg, July 2009.

[97] T. HOHEISEL, C. KANZOW, AND A. SCHWARTZ, *Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints*, Mathematical Programming, 137 (2013), pp. 257–288.

[98] J. HOLTE, *Discrete Gronwall Lemma*, tech. rep., Gustavus Adolphus College, 2009.

[99] J. HOOKER, *A principled approach to mixed integer/linear problem formulation*, in Operations Research and Cyber-Infrastructure, J. Chinneck, B. Kristjansson, and M. Saltzman, eds., Springer, 2009, pp. 79–100.

[100] A. IZMAILOV AND M. SOLODOV, *Mathematical programs with vanishing constraints: Optimality conditions, sensitivity, and a relaxation method*, Journal of Optimization Theory and Applications, 142 (2009), pp. 501–532.

[101] H. JONGEN AND V. SHIKHMAN, *Bilevel optimization: on the structure of the feasible set*, Mathematical Programming, 136 (2012), pp. 65–89.

[102] J. JÚDICE, H. SHERALI, I. RIBEIRO, AND A. FAUSTINO, *Complementarity active-set algorithm for mathematical programming problems with equilibrium constraints*, Journal of Optimization Theory and Applications, 134 (2007), pp. 467–481.

[103] M. JUNG, C. KIRCHES, AND S. SAGER, *On perspective functions and vanishing constraints in mixed-integer nonlinear optimal control*, in Facets of Combinatorial Optimization – Festschrift for Martin Grötschel, M. Jünger and G. Reinelt, eds., Springer Berlin Heidelberg, 2013, pp. 387–417.

[104] M. JUNG, G. REINELT, AND S. SAGER, *The Lagrangian relaxation for the combinatorial integral approximation problem*, Optimization Online, 2 (2012). Submitted to Optimization Methods and Software 2013.

[105] S. KAMESWARAN AND L. BIEGLER, *Simultaneous dynamic optimization strategies: Recent advances and challenges*, Computers & Chemical Engineering, 30 (2006), pp. 1560–1575.

[106] Y. KAWAJIRI AND L. BIEGLER, *A nonlinear programming superstructure for optimal dynamic operations of simulated moving bed processes*, I&EC Research, 45 (2006), pp. 8503–8513.

[107] C. KAYA AND J. NOAKES, *A computational method for time-optimal control*, Journal of Optimization Theory and Applications, 117 (2003), pp. 69–92.

[108] A. KEHA, I. FARIAS JR., AND G. NEMHAUSER, *Models for representing piecewise linear cost functions*, Operations Research Letters, 32 (2004), pp. 44–48.

[109] J. KELLEY, *The cutting-plane method for solving convex programs*, Journal of SIAM, 8 (1960), pp. 703–712.

[110] M. KILINÇ, J. LINDEROTH, AND J. LUEDTKE, *Effective separation of disjunctive cuts for convex mixed integer nonlinear programs*, tech. rep., Computer Sciences Department, University of Wisoncsin–Madison, 2010.

[111] C. KIRCHES, *Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control*, Advances in Numerical Mathematics, Springer Vieweg, Wiesbaden, July 2011.

[112] C. KIRCHES, S. SAGER, H. BOCK, AND J. SCHLÖDER, *Time-optimal control of automobile test drives with gear shifts*, Optimal Control Applications and Methods, 31 (2010), pp. 137–153.

[113] P. KRÄMER-EIS, H. BOCK, R. LONGMAN, AND J. SCHLÖDER, *Numerical determination of optimal feedback control in nonlinear problems with state/control constraints*, Advances in the Astronautical Sciences, 105 (2000), pp. 53–71.

[114] M. KVASNICA, P. GRIEDER, AND M. BAOTIĆ, *Multi-Parametric Toolbox (MPT)*. http://control.ee.ethz.ch/~mpt/, 2004.

[115] S. LEE AND I. GROSSMANN, *New algorithms for nonlinear generalized disjunctive programming*, Computers and Chemical Engineering Journal, 24 (2000), pp. 2125–2141.

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

[117] D. LEINEWEBER, I. BAUER, H. BOCK, AND J. SCHLÖDER, *An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: Theoretical aspects*, Computers & Chemical Engineering, 27 (2003), pp. 157–166.

[118] C. LEMARECHAL, *Lagrangian relaxation*, in Computational Combinatorial Optimization, M. Jünger and D. Naddef, eds., vol. 2241 of Lecture Notes in Computer Science, Springer, 2001, ch. 4, pp. 112–156.

[119] S. LEYFFER, *Deterministic methods for mixed-integer nonlinear programming*, PhD thesis, University of Dundee, 1993.

[120] ——, *The return of the active–set method*, preprint ANL/MCS-P1277-0805, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA, February 2005.

[121] ——, *Optimization with Multivalued Mappings*, vol. 2, Springer, 2006, ch. Complementarity Constraints as Nonlinear Equations: Theory and Numerical Experiences, pp. 169–208.

[122] S. LEYFFER, G. LÓPEZ-CALVA, AND J. NOCEDAL, *Interior methods for mathematical programs with complementarity constraints*, SIAM Journal on Optimization, 17 (2006), pp. 52–77.

[123] Y. LIN AND M. STADTHERR, *Deterministic global optimization of nonlinear dynamic systems*, AIChE Journal, 53 (2007), pp. 866–875.

[124] J. LINDEROTH AND M. SAVELSBERGH, *A computational study of branch and bound search strategies for mixed integer programming*, INFORMS Journal on Computing, 11 (1999), pp. 173–187.

[125] F. LOGIST, S. SAGER, C. KIRCHES, AND J. VAN IMPE, *Efficient multiple objective optimal control of dynamic systems with integer controls*, Journal of Process Control, 20 (2010), pp. 810–822.

[126] K. MALANOWSKI, H. MAURER, AND S. PICKENHAIN, *Second-order sufficient conditions for state-constrained optimal control problems*, Journal of Optimization Theory and Applications, 123 (2004), pp. 595–617.

[127] M. MARINAKI AND M. PAPAGEORGIOU, *Nonlinear optimal flow control for sewer networks*, Proceedings of the American Control Conference, 2 (1998), pp. 1289–1293.

[128] ——, *Rolling-horizon optimal control of sewer networks*, Proceedings of the IEEE International Conference on Control Applications, 1 (2001), pp. 594–599.

[129] ——, *Optimal Real-time Control of Sewer Networks*, Springer, London, 2005.

[130] A. MARTIN, M. MÖLLER, AND S. MORITZ, *Mixed integer models for the stationary case of gas network optimization*, Mathematical Programming, 105 (2006), pp. 563–582.

[131] H. MAURER, *On optimal control problems with boundary state variables and control appearing linearly*, SIAM Journal on Control and Optimization, 15 (1977), pp. 345–362.

[132] H. MAURER AND N. OSMOLOVSKII, *Second order sufficient conditions for time-optimal bang-bang control*, SIAM Journal on Control and Optimization, 42 (2004), pp. 2239–2263.

[133] G. NANNICINI AND P. BELOTTI, *Rounding-based heuristics for nonconvex minlps*, Mathematical Programming Computation, 4 (2012), pp. 1–31.

[134] L. NEUSTADT, *Optimization: A Theory of Necessary Conditions*, Princeton University Press, 1976.

[135] C. OCAMPO-MARTINEZ, *Model predictive control of wastewater systems*, Springer, 2011.

[136] C. OCAMPO-MARTINEZ AND V. PUIG, *Piece-wise linear functions-based model predictive control of large-scale sewage systems*, IET Control Theory and Applications, 4 (2010), pp. 1581–1593.

[137] J. OLDENBURG, *Logic–based modeling and optimization of discrete–continuous dynamic systems*, vol. 830 of Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik, VDI Verlag, Düsseldorf, 2005.

[138] J. OLDENBURG AND W. MARQUARDT, *Optimization of discrete-continuous dynamic systems based on disjunctive programming*, in GAMM Annual Meeting 2005, Luxembourg, Luxembourg, vol. 5, 2005, pp. 51–54.

[139] ——, *Disjunctive modeling for optimal control of hybrid systems*, Computers & Chemical Engineering, 32 (2008), pp. 2346–2364.

[140] J. OLDENBURG, W. MARQUARDT, D. HEINZ, AND D. LEINEWEBER, *Mixed logic dynamic optimization applied to batch distillation process design*, AIChE Journal, 49 (2003), pp. 2900–2917.

[141] H. PESCH AND R. BULIRSCH, *The maximum principle, Bellman's equation and Caratheodory's work*, Journal of Optimization Theory and Applications, 80 (1994), pp. 203–229.

[142] M. PLEAU, H. COLAS, P. LAVALLÉE, G. PELLETIER, AND R. BONIN, *Global optimal real-time control of the Quebec urban drainage system*, Environmental Modelling and Software, 20 (2005), pp. 401–413.

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

[144] L. PONTRYAGIN, V. BOLTYANSKI, R. GAMKRELIDZE, AND E. MISCENKO, *The Mathematical Theory of Optimal Processes*, Wiley, Chichester, 1962.

[145] A. POTSCHKA, H. BOCK, AND J. SCHLÖDER, *A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems*, Optimization Methods and Software, 24 (2009), pp. 237–252.

[146] V. PUIG, G. CEMBRANO, J. ROMERA, J. QUEVEDO, B. AZNAR, G. RAMÓN, AND J. CABOT, *Predictive optimal control of sewer networks using CORAL tool: application to Riera Blanca catchment in Barcelona*, Water Science and Technology, 60 (2009), pp. 869–878.

[147] I. QUESADA AND I. GROSSMANN, *An LP/NLP based branch and bound algorithm for convex MINLP optimization problems*, Computers & Chemical Engineering, 16 (1992), pp. 937–947.

[148] A. RAGHUNATHAN AND L. BIEGLER, *Mathematical programs with equilibrium constraints (MPECs) in process engineering*, Computers & Chemical Engineering, 27 (2003), pp. 1381–1392.

[149] A. RAGHUNATHAN, M. DIAZ, AND L. BIEGLER, *An MPEC formulation for dynamic optimization of distillation operations*, Computers & Chemical Engineering, 28 (2004), pp. 2037–2052.

[150] J. RUIZ AND I. GROSSMANN, *Using convex nonlinear relaxations in the global optimization of nonconvex generalized disjunctive programs*, Computers & Chemical Engineering, 49 (2013), pp. 70–84.

[151] S. SAGER, *MIOCP benchmark site*. http://mintoc.de.

[152] ——, *Numerical methods for mixed–integer optimal control problems*, Der andere Verlag, Tönning, Lübeck, Marburg, 2005.

[153] ——, *Reformulations and algorithms for the optimization of switching decisions in nonlinear optimal control*, Journal of Process Control, 19 (2009), pp. 1238–1247.

[154] ——, *On the integration of optimization approaches for mixed-integer nonlinear optimal control*. Habilitation. University of Heidelberg, August 2011.

[155] ——, *A benchmark library of mixed-integer optimal control problems*, in Mixed Integer Nonlinear Programming, J. Lee and S. Leyffer, eds., Springer, 2012, pp. 631–670.

[156] 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 (2011), pp. 936–959.

[157] S. SAGER, H. BOCK, AND M. DIEHL, *The integer approximation error in mixed-integer optimal control*, Mathematical Programming A, 133 (2012), pp. 1–23.

[158] S. SAGER, M. JUNG, AND C. KIRCHES, *Combinatorial integral approximation*, Mathematical Methods of Operations Research, 73 (2011), pp. 363–380.

[159] S. SAGER, C. KIRCHES, AND H. BOCK, *Fast solution of periodic optimal control problems in automobile test-driving with gear shifts*, in Proceedings of the 47th IEEE Conference on Decision and Control (CDC 2008), Cancun, Mexico, 2008, pp. 1563–1568.

[160] S. SAGER, G. REINELT, AND H. BOCK, *Direct methods with maximal lower bound for mixed-integer optimal control problems*, Mathematical Programming, 118 (2009), pp. 109–149.

[161] R. SARGENT AND G. SULLIVAN, *The development of an efficient optimal control package*, in Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2, J. Stoer, ed., Heidelberg, 1978, Springer.

[162] N. SAWAYA, *Reformulations, Relaxations and Cutting Planes for Generalized Disjunctive Programming*, PhD thesis, Carnegie Mellon University, 2006.

[163] N. SAWAYA AND I. GROSSMANN, *Computational implementation of non-linear convex hull reformulation*, Computers & Chemical Engineering, 31 (2007), pp. 856–866.

[164] H. SCHEEL AND S. SCHOLTES, *Mathematical programs with complementarity constraints: Stationarity, optimality and sensitivity*, Mathematics of Operations Research, 25 (2000), pp. 1 – 22.

[165] M. SCHÜTZE, A. CAMPISANO, H. COLAS, W. SCHILLING, AND P. VANROLLEGHEM, *Real time control of urban wastewater systems – where do we stand today?*, Journal of Hydrology, 299 (2004), pp. 335–348.

[166] M. SHAIKH, *Optimal Control of Hybrid Systems: Theory and Algorithms*, PhD thesis, Department of Electrical and Computer Engineering, McGill University, Montreal, Canada, 2004.

[167] M. SHAIKH AND P. CAINES, *On the hybrid optimal control problem: Theory and algorithms.*, IEEE Transactions on Automatic Control, 52 (2007), pp. 1587–1603.

[168] A. SINGER AND P. BARTON, *Global optimization with nonlinear ordinary differential equations*, Journal of Global Optimization, 34 (2006), pp. 159–190.

[169] C. SONNTAG, O. STURSBERG, AND S. ENGELL, *Dynamic optimization of an industrial evaporator using graph search with embedded nonlinear programming*, in Proc. 2nd IFAC Conf. on Analysis and Design of Hybrid Systems (ADHS), 2006, pp. 211–216.

[170] O. STEIN, J. OLDENBURG, AND W. MARQUARDT, *Continuous reformulations of discrete-continuous optimization problems*, Computers & Chemical Engineering, 28 (2004), pp. 3672–3684.

[171] O. STRYK, *Numerical solution of optimal control problems by direct collocation*, in Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods, vol. 111, Bulirsch et al., 1993, pp. 129–143.

[172] ——, *Numerische Lösung optimaler Steuerungsprobleme: Diskretisierung, Parameteroptimierung und Berechnung der adjungierten Variablen*, PhD thesis, TU Munich, 1995.

[173] R. STUBBS AND S. MEHROTRA, *A branch-and-cut method for 0-1 mixed convex programming*, Mathematical Programming, 86 (1999), pp. 515–532.

[174] H. SUSSMANN, *A maximum principle for hybrid optimal control problems*, in Conference proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, 1999.

[175] M. SZYMKAT AND A. KORYTOWSKI, *The method of monotone structural evolution for dynamic optimization of switched systems*, in IEEE CDC08 Proceedings, 2008.

[176] N. TAUCHNITZ, *Das Pontrjaginsche Maximumprinzip für eine Klasse hybrider Steuerungsprobleme mit Zustandsbeschränkungen und seine Anwendung*, PhD thesis, BTU Cottbus, 2010.

[177] S. TERWEN, *Vorausschauende Längsregelung schwerer Lastkraftwagen*, PhD thesis, Universität Karlsruhe (TH), März 2010.

[178] S. TERWEN, M. BACK, AND V. KREBS, *Predictive powertrain control for heavy duty trucks*, in Proceedings of IFAC Symposium in Advances in Automotive Control, Salerno, Italy, 2004, pp. 451–457.

[179] F. Torrisi and A. Bemporad, *HYSDEL – A tool for generating computational hybrid models for analysis and synthesis problems*, IEEE Transactions on Control Systems Technology, 12 (2004), pp. 235–249.

[180] M. Turkay and I. Grossmann, *Logic-based MINLP algorithms for the optimal synthesis of process networks*, Computers & Chemical Engineering, 20 (1996), pp. 959–978.

[181] K. Uthaichana, S. Bengea, R. DeCarlo, S. Pekarek, and M. Zefran, *Hybrid model predictive control tracking of a sawtooth driving profile for an HEV*, in American Control Conference, 2008, 2008, pp. 967–974.

[182] A. Wächter and L. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Mathematical Programming, 106 (2006), pp. 25–57.

[183] A. Walther and A. Griewank, *Getting started with ADOL-C*, in Combinatorial Scientific Computing, U. Naumann and O. Schenk, eds., Chapman-Hall CRC Computational Science, 2012, ch. 7, pp. 181–202.

[184] T. Westerlund and F. Pettersson, *An extended cutting plane method for solving convex MINLP problems*, Computers & Chemical Engineering, 19 (1995), pp. 131–136.

[185] H. Williams, *Model Building in Mathematical Programming, 4th Edition*, Wiley, 4 ed., October 1999.

Nomenclature

Throughout this thesis, lowercase roman and greek letter in boldface ($\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$) are used for vectors. Matrices use uppercase roman letters in boldface ($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$). Scalars are denoted by lowercase roman and greek letters (f, g, λ, μ), while sets use uppercase calligraphic style ($\mathcal{A}$, $\mathcal{F}$, $\mathcal{X}$). Finally, number spaces are denoted in uppercase blackboard style ($\mathbb{N}$, $\mathbb{R}$, $\mathbb{Z}$).

Several notational conventions mandate a brief explanation. Vector values are printed in boldface and are assumed to be column vectors. Transposition of a vector $\boldsymbol{v}$ is indicated by $\boldsymbol{v}^T$ and is omitted in the concatenation of column vectors,

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{v}_1^T & \boldsymbol{v}_2^T \end{bmatrix}^T.$$

Sets are denoted by calligraphic letters in upper case and may at times be used as index sets selecting a subset of elements of a vector or matrix. To this end, we use the convention

$$\boldsymbol{v}_{\mathcal{I}} \stackrel{\text{def}}{=} (v_i)_{i \in \mathcal{I}}$$

for a vector $\boldsymbol{v} \in \mathbb{R}^n$ and for a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ we write

$$\boldsymbol{A}_{\mathcal{I}} \stackrel{\text{def}}{=} \left(A_{ij}\right)_{i \in \mathcal{I}, j \in \{1,\ldots,n\}}, \qquad \boldsymbol{A}_{\star\mathcal{J}} \stackrel{\text{def}}{=} \left(A_{ij}\right)_{i \in \{1,\ldots,m\}, j \in \mathcal{J}}, \qquad \boldsymbol{A}_{\mathcal{I}\mathcal{J}} \stackrel{\text{def}}{=} \left(A_{ij}\right)_{i \in \mathcal{I}, j \in \mathcal{J}}.$$

The gradient of a scalar valued function $f : \mathbb{R}^n \to \mathbb{R}$ with respect to a vector valued unknown $\boldsymbol{x}$ is denoted by $f_{\boldsymbol{x}}$ and is understood as a *row vector*,

$$f_{\boldsymbol{x}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{\mathrm{d}f(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix},$$

while the derivative of a vector valued function $\boldsymbol{f} : \mathbb{R} \to \mathbb{R}^m$ with respect to the scalar valued unknown x is denoted by $\boldsymbol{f}_x$ and is understood as a *column vector*,

$$\boldsymbol{f}_x(x) \stackrel{\text{def}}{=} \frac{\mathrm{d}\boldsymbol{f}(x)}{\mathrm{d}x} = \begin{bmatrix} \frac{\mathrm{d}f_1(x)}{\mathrm{d}x} \\ \vdots \\ \frac{\mathrm{d}f_m(x)}{\mathrm{d}x} \end{bmatrix},$$

Consequentially, the Jacobian of a vector valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$, denoted by $\boldsymbol{f}_{\boldsymbol{x}}$, is an

$m \times n$ matrix composed from the m gradients of the component functions $f_i : \mathbb{R}^n \to \mathbb{R}$,

$$\boldsymbol{f}_{\boldsymbol{x}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{\mathrm{d}\boldsymbol{f}(\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}} = \begin{bmatrix} f_{1,\boldsymbol{x}}(\boldsymbol{x}) \\ \vdots \\ f_{m,\boldsymbol{x}}(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\boldsymbol{x})}{\partial x_n} \end{bmatrix}.$$

List of Symbols

$\triangle$	End of a definition, lemma, proposition, theorem, or corollary
$\square$	End of a proof
$\lightning$	Contradiction symbol
$\stackrel{\text{def}}{=}$	Defined to be equal
$(\cdot)$	Wildcard notation for the omitted list of function arguments
$\lceil\cdot\rceil$	Component-wise mapping of a real number to the next largest integer value
$\lfloor\cdot\rfloor$	Component-wise mapping of a real number to the next smallest integer value
$\lvert\cdot\rvert$	Component-wise mapping of a real number to the absolute value
$\lVert\cdot\rVert$	A norm of a vector
$\lvert\lvert\lvert\cdot\rvert\rvert\rvert$	A norm of a matrix
$\{\ \}$	Set delimiters
$\cup$	Set–theoretic union ("unified with")
$\cap$	Set–theoretic intersection ("intersected with")
$\subseteq, \subset$	Subset of a set ("is a (proper) subset of")
$\in, \notin$	Set membership ("is (not) an element of")
$\times$	Cartesian product of sets
$\emptyset$	The empty set
$\forall$	Universal quantifier ("for all")
$\exists$	Existential quantifiers ("there exist")
$\vee$	Logical inclusive disjunction ("or")
$\perp$	Complementarity operator
$\boldsymbol{A}^T$, $\boldsymbol{x}^T$	Transpose of matrix or vector
$f_{\boldsymbol{x}}$	Gradient of the scalar function $f(\cdot)$ with respect to unknown $\boldsymbol{x}$
$\boldsymbol{f}_{\boldsymbol{x}}$	Jacobian of the vector valued function $\boldsymbol{f}(\cdot)$ with respect to unknown $\boldsymbol{x}$
x_i	i–th entry of row or column vector $\boldsymbol{x}$, a scalar value
$x_{\mathcal{I}}$	Subvector of elements of $\boldsymbol{x}$ whose indices are contained in $\mathcal{I} \subset \mathbb{N}$

Roman Symbols

$\boldsymbol{A}$	Coefficient matrix for affine systems
$\boldsymbol{G}$	WRONSKI-matrices
L	LIPSCHITZ constant

M	Big-M bounds for functions and variables
Z_{LP}	Value of the LP relaxation
Z_{LR}	Value of the LAGRANGIAN relaxation
$\boldsymbol{b}_i$	Discretization base functions
$\boldsymbol{c}(\cdot)$	Path constraint function
$\boldsymbol{e}_i$	The i–th unit column vector
e	EULER's number
$\boldsymbol{f}$	ODE system right hand side
m	Number of intervals in a discretization
n	Dimension of a vector, row or column dimension of a matrix
$\boldsymbol{p}$	Vector of binary control parametrization
$\boldsymbol{q}$	Vector of control parametrization
$\boldsymbol{r}$	Point constraint
$\boldsymbol{s}$	Vector of discretized states
t	Model or process time
t_0	Initial model or process time, start of time horizon
t_{f}	Final model or process time, end of time horizon
$\boldsymbol{u}(\cdot)$	Trajectory of continuous process controls
$\boldsymbol{v}(\cdot)$	Trajectory of discrete process controls
$\boldsymbol{x}(\cdot)$	Trajectory of ODE system states
$\boldsymbol{x}_0(t_k)$	Observed process state at time t_k for initial value embedding
$\boldsymbol{z}_{\mathrm{LR}}(\cdot)$	LAGRANGIAN function

Greek Symbols

Δ	Prefix for time steps
Ω	Set of admissible choices for a discrete control trajectory
Φ	Objective function/functional
$\boldsymbol{\alpha}(\cdot)$	Trajectory of relaxed controls.
$\boldsymbol{\lambda}$	LAGRANGE multipliers
$\boldsymbol{\mu}$	LAGRANGE multipliers
φ	NCP function
$\boldsymbol{\sigma}$	Switching variables
τ	Auxilliary time descriptions
θ	Control approximation quality
$\boldsymbol{\omega}(\cdot)$	Trajectory of binary controls.
$\boldsymbol{\zeta}$	Min-up times

Calligraphic Symbols

$\mathcal{C}$	Space of differentiable functions
$\mathcal{E}$	Index set
$\mathcal{F}$	Set of fixed variable indices due to branching

$\mathcal{H}$	HAMILTONIAN
$\mathcal{I}$	Index sets
$\mathcal{L}^1$	Space of LEBESQUE integrable functions
$\mathcal{L}^\infty$	Space of essentially bounded functions
$\mathcal{L}^\infty_k$	Space of essentially bounded functions with k degrees of freedom
$\mathcal{U}$	Set of feasible continuous controls

Blackboard Symbols

$\mathbb{G}_m$	Grid with $m+1$ grid points
$\mathbb{N}, \mathbb{N}_0$	Set of natural numbers excluding (including) zero
$\mathbb{R}, \mathbb{R}_0^+, \mathbb{R}_0^-$	Set of real (nonnegative real, nonpositive real) numbers
$\mathbb{R}^n$	Space of n–vectors with elements from the set $\mathbb{R}$
$\mathbb{R}^{m\times n}$	Space of $m \times n$–matrices with elements from the set $\mathbb{R}$

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ACQ	ABADIE Constraint Qualification
BVP	Boundary Value Problem
CSO	Combined Sewer Overflow
DAE	Differential Algebraic Equation
GCQ	GUIGNARD Constraint Qualification
GDP	General Disjunctive Programming
HYSDEL	HYbrid System DEscription Language
IC	Inner Convexification
IP	Integer Program
IVP	Initial Value Problem
KKT	KARUSH-KUHN-TUCKER
LICQ	Linear Independence Constraint Qualification
LP	Linear Program
MLD	Mixed Logical Dynamical
MFCQ	MANGASARIAN-FROMOVITZ Constraint Qualification
MILP	Mixed-Integer Linear Program
MINLP	Mixed-Integer Nonlinear Program
MIOC	Mixed-Integer Optimal Control
MIOCP	Mixed-Integer Optimal Control Problem
MPCC	Mathematical Program with Complementarity Constraints
MPEC	Mathematical Program with Equilibrium Constraints
MPT	Multi-Parametric Toolbox
MPVC	Mathematical Program with Vanishing Constraints
NCP	Nonlinear Complementarity Problem
NFR	Next-forced Rounding
NLP	Nonlinear Program
OC	Outer Convexification
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
SOS	Special Ordered Set type
STL	Standard Template Library
SQP	Sequential Quadratic Programming
SUR	Sum-up Rounding