Convex Optimization in a Nonconvex World: Applications for Aerospace Systems

Danylo Malyuta

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:
Behçet Açıkmeşe, Chair
Mehran Mesbahi
Maryam Fazel

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

Abstract

Convex Optimization in a Nonconvex World:
Applications for Aerospace Systems

Danylo Malyuta

Chair of the Supervisory Committee:
Professor Behçet Açıkmeşe
Aeronautics & Astronautics

Future aerospace vehicles, and other autonomous systems at large, will operate with higher levels of automation and in more general environments than their predecessors. Counter-intuitively, the sought-after generality of operation of new autonomous systems calls for the consideration of far more constraints during system operation. To name a few examples, these constraints may relate to obstacle avoidance, sensing, or actuator constraints.

This future breed of more-constrained autonomous agents calls for the development of new control methods that seamlessly and reliably deal with a host of nonconvex (in other words, difficult) intrinsic and extrinsic constraints. This dissertation makes the basic assumption that, for safety reasons, the control methods are required to be extremely reliable. With this premise in mind, the chapters of this document focus on developing convex optimization-based methods for dealing with nonconvex feedback and feedforward control tasks. The resulting algorithms are applicable to the broad spectrum of aerospace vehicles. The particular cases considered are satellite orbit station-keeping, spacecraft rendezvous and docking, and planetary rocket landing. In each case, it is shown using numerical examples that the algorithms attain levels of performance above the current state-of-the-art.

The dissertation is organized into four major parts. In Chapter 3, a robust model predictive feedback controller is presented that is based on convex optimization alone, and that

applies to satellite orbit station-keeping. In Chapter 4, an explicit model predictive controller is presented that can be used to precompute optimal control laws for general mixed-integer convex problems, albeit at the price of higher onboard storage memory. In Chapter 6, a novel method is presented for computing feedforward trajectories of hybrid systems using lossless convexification. Applications are shown for spacecraft rendezvous and for rocket landing. Finally, Chapter 7 considers an iterative method to solve even more difficult hybrid system problems. This method is based on the methodologies of sequential convex programming and numerical continuation.

Danylo Malyuta, June 11, 2021.

"Find a Way"

– Diana Nyad, champion long-distance swimmer who, at age 64, became the first person to swim from Cuba to Florida without the aid of a shark cage.

DEDICATION

For Taryn. You matter above all else.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Show me someone who says that they are "self-made [insert anything here]", and I will show you a liar. I truly believe that none of us are self-made, and we all depend and owe much of our lives to the good people that have helped us along the way: some ways big, some ways small, but *all* important. This makes this section the hardest one for me to write. If I think about it for long enough, it would take more than a thesis length for me to acknowledge all the amazing and kind individuals who believed in me and who helped when help was needed. To anyone not mentioned by name, I am so sorry, please forgive me and contact me – I would love to say thank you in person.

The only people I can imagine beginning the acknowledgements with are my parents, Ruslan Malyuta and Olena Krivtsova. Both of you have sacrificed greatly and moving away from Ukraine was not easy, but you did it to give your children a chance at a better life. I am eternally grateful for your unwavering support and dedication to giving my sister and I the best possible education. The love and importance of education is something that your attitude has engrained in me, and I am forever thankful for it. You have been there since the very beginning, I will never forget that.

I have to also thank my two grandfathers, Nikolay Alekseevich Krivtsov and Nikolay Petrovich Malyuta. If my parents carried the load of day-to-day education, you are the ones who inspired me about spaceflight and the stars. Every single time that I would visit, you would tell me stories about your days as an airplane mechanic and as a physicist. I think I listened to these stories 100 times by now – and still I want to hear them again. We would look at the stars, and we would dream up stories of spaceflight. I grew up on those tales and, even though it may seem like I also grew "out" of them, those times and stories lie deep

within me and bring a kindle of warmth whenever needed. Thank you for your inspiration, I'm sure that I owe my space ambition to you.

I was not always a good or ambitious student. Like most young boys, I imagine that I was a bit of a troublemaker. Certainly that was the case in primary school – that is, until I attended Ms. Lau's math and science class at the Cambridgeport School in Boston, MA. My memory goes like this: at one point, I got an "A–" on a test. I don't think I ever got an "A" on a test before that point. At the same time, I was reading an aerodynamics textbook gifted by a family friend (I was around 11 years old, but already with a dream of becoming an airplane designer). So I knew that science had an important role in my future career, and with that "A–" I thought, "Huh, I can do this!" It literally became an obsession to never get less than "A–", ever again. Ever. I wasn't always successful, but Ms. Lau's encouragement was instrumental. Her and other teachers at the school made a small kid (me) feel like the aeronautical engineer "in the house". I wish I had your contact information to tell you how much that meant to me. Being treated as an adult and as a local authority in a beloved subject – that means everything to children. It certainly meant everything to me. I attribute my "getting started" to Ms. Lau. Thank you.

When my family moved to Switzerland, I attended the International Baccalaureate program at the *Campus des Nations* of the International School of Geneva. There I had the privilege of being taught by many amazing teachers, to whom I am thankful beyond measure. I would like to thank in particular Alastair Ronn and John Barr, my Higher Level Physics and Higher Level Math teachers. These amazing people inspired curiosity and taught excellence to their students. They prepared me for the difficult studies at university. Without your instruction, I would have never survived the brutal first year of *École Polytechnique Fédérale de Lausanne*. Thank you for everything, I will never forget the role you played in my life.

When I entered university, my plan was to study aerodynamics, get a Master's degree at

the *SUPAERO* in Toulouse, France, and to work for Airbus. None of it happened thanks to Professor Colin Neil Jones, and I am eternally grateful for how things turned out. Attending Professor Jones' *Control Systems I* course made me fall in love with feedback control. I connected with the subject on a personal level, and I believe it is thanks to Professor Jones' engaging and unassuming teaching style. *Never* underestimate the power of a good teacher – I now think that subjects are not intrinsically interesting, but instead the curiosity in them is largely sparked by a personal connection to the teacher. Professor Jones, thank you for putting me in touch me with Professor Behçet Açıkmeşe and placing me on a path that leads straight to where I am today. I hope I get a chance to repay you for it somehow.

The path from Professor Jones' class to the University of Washington passes first through ETH Zürich and the lecture room for *Control Systems II* taught by Professor Roy Smith. If Professor Jones kindled the fire in me, Professor Smith turned it into a roaring blaze. I am truly humbled by the amazing teachers and people that I have met in my life. Professor Smith, I am lucky to have been your student.

After my family moved away from Boston, MA, in 2007, my first real "comeback" to the United States was for an internship and a Master's thesis at the NASA Jet Propulsion Laboratory in 2017. I am deeply grateful and indebted to Dr. Roland Brockers for giving me the opportunity. That JPL internship opened a million doors and allowed me to meet so many awe-inspiring engineers and scientists working on the final frontier of human exploration. I still remember my first conversation over Skype with Dr. Brockers, where he said something like this, "What I really find special about America is that people give you the chance to try new things. But you better not screw it up." I still reflect on these words, and it is still how I think about this country. This land has given me many unimaginable opportunities, and many people have placed their trust in my abilities. I feel that weight on my shoulders, and I am trying my best to not let anyone down. Perhaps that is my deepest motivation: to not let anyone down.

I finally arrive at my present-day status as a Ph.D. candidate at the Autonomous Controls Laboratory at the University of Washington. My thanks have to begin with my amazing advisor, Professor Behçet Açıkmeşe. There are no words in language to describe how much you have meant for my life – academically, personally, and professionally. I still have a vivid memory of how we met. The plan was to intern with you at the University of Texas at Austin over Summer 2015. But you were away on travels, so I spent most of the time working with other Ph.D. students. Then, in the final two weeks of my visa, there was a knock on the door to my "research room cellar". Suddenly, you were there, "Hey, Danylo!". That is how our relationship started. Straight away we dove into research discussions, and you tried to make me stay in your group and skip ETH. I was too pre-planned of a person to do that, so I left on the understanding that I will "be back" for a Ph.D.. What impressed me most was how quickly we started to work together to solve interesting and challenging problems. Just like Professor Jones from before, I was immediately enamoured by Professor Açıkmeşe's unassuming approach to solving difficult tasks. A fellow Ph.D. from our group, Yue Yu, once said that an advisor is like an academic father. Professor Açıkmeşe, I can say that your approach and the lessons I learned from you will resonate throughout my career. I hope to constantly stay in touch and to give back. Thank you for what you did. More than anyone else professional in the past decade, you have changed my life.

My experience at the University of Washington would not be the same if not for Professors Mehran Mesbahi, Maryam Fazel, and Dr. John Carson III at the NASA Johnson Space Center. In teaching, advising, and guiding my academic development towards being useful for industry, I have so many thanks to give to the three of you. I would not have done what I did if not for meeting and learning from you. I will always remember what you did for me, you are all a constant source of inspiration, and I hope to give back to you in even greater amounts.

I would have to be stripped of my Ph.D. if I did not acknowledge two of the most

important people in it: Michael (Miki) Szmuk and Taylor P. Reynolds. Where to begin, if the amount of things to say is infinite? Miki, you are plainly and simply the kindest, most thoughtful, most insightful, and clearest thinker that I have ever had the chance to call both a friend and a colleague. I respected you from the day we met at UT Austin in Summer 2015. Just for reference, we only overlapped by a few days between the end of your Blue Origin internship and me leaving back to Switzerland. That was enough for me to understand that I stood in the presence of an amazing individual. You are half the reason that I came back for a Ph.D. with Professor Açıkmeşe. I hope that our friendship and collaboration continue through our lives. The results of my Ph.D. owe so much to the foundation you have set.

Taylor, it is a privilege like none other to be your friend and collaborator. Your work ethic, knowledge, capability, interest in reading space history books – all of these and more have become reference points to strive for in my own life. You have much greatness lying ahead of you, and the world is a better place thanks to people like you. If it were Miki's early publications that got me interested in the work that we do, it was your code that I "grew up" on. Our collaboration has been integral to much of this thesis, and I am indebted to you for it.

As the saying attributed to Henry David Thoreau goes, "What you get by achieving your goals is not as important as what you become by achieving your goals." In my case, I thought I was coming to the United Status to get a Ph.D.. What I found in the last three years, however, far exceeds the Ph.D.. I found my life partner, Taryn Zhao. I could not imagine the last three years without you, nor could I imagine the next one hundred. The human language is not expressive enough to convey my level of gratitude for your constant support and belief in me. I love you very much, and this thesis is dedicated to you.

Danylo Malyuta, June 11, 2021.

NOMENCLATURE

JPL: Jet Propulsion Laboratory.

MSL: Mars Science Laboratory.

RAM: Random Access Memory.

NASA: National Aeronautics and Space Administration.

AFRL: Air Force Research Laboratory.

HPSC: High-performance spaceflight computing.

GPS: Global Positioning System.

MAV: Micro Aerial Vehicle.

COTS: Commercial off-the-shelf.

RCS: Reaction Control System.

SCP: Sequential Convex Programming.

STC: state-triggered constraint.

MIB: minimum impulse-bit.

RMPC: robust model predictive control.

QP: quadratic program.

SOC: second-order cone.

SOCP: second-order cone program.

FRCI: feasible robust controlled invariant.

LEO: low Earth orbit.

LVLH: local-vertical local-horizontal.

LQR: linear quadratic regulator.

MICP: mixed-integer convex program.

MINLP: mixed-integer nonlinear program.

NLP: nonlinear program.

DOF: degree of freedom.

AAV: autonomous aerial vehicle.

LTI: linear time-invariant.

G-FOLD: guidance for optimal large divert.

AGL: above ground level.

Chapter 1

INTRODUCTION

This dissertation develops a set of *real-time*, *optimization-based* algorithms with *provable* properties for demanding aerospace control applications. The purpose of this introduction is to break down and to justify the worthiness of this thesis statement, which consists of three key operator phrases that are italicized.

No single book, let alone a thesis, can appropriately address every aspect of algorithm design. This dissertation thus focuses on two must-have properties for algorithms in safety-critical systems: real-time capability and provable (in other words, predictable) behavior. An algorithm is *real-time* if it is able to finish execution in less time than the allotted duration on relevant hardware. For example, heritage planetary lander trajectory generation has targeted computation speeds in the 1 Hz range [1].

A real-time algorithm has little practical use, however, if it requires excessive compute power. This is especially relevant for aerospace, where embedded computing hardware lags far behind of commercial desktop computers. In 2012, the Mars Science Laboratory (MSL) mission landed the 899 kg Curiosity rover on Mars using a RAD750 radiation-hardened processor with just 4 GB of flash memory, 128 MB of random access memory (RAM), and a clock-speed of 133 MHz [2, 3, 4]. Today, the National Aeronautics and Space Administration (NASA), the Air Force Research Laboratory (AFRL), and commercial partners are working on a high-performance spaceflight computing (HPSC) project to deliver a next-generation spaceflight processor featuring an 8-core cluster of ARM A53 processors running at 800 MHz [5, 6, 7]. However, on-board[1] compute power for individual algorithms remains at a premium.

[1]Literature often refers to on-board optimization as *on-line* optimization. Similarly, optimization performed on the ground is called *off-line* optimization.

Algorithms that satisfy system requirements with few computational resources are always preferred.

If on-board compute power is lacking, why not relegate calculation to ground computers? To answer this question, it is first of all necessary to realize that for operations such as planetary landing, being able to recompute the control strategy to adjust for real-world uncertainty is essential for system reliability. The only way then to involve ground computers for computation is if it is possible to communicate calculation results to the real vehicle. Even for lunar flight, the two-way Earth-Moon communication is approximately 2.5 s, which is slower than the typical 1 Hz trajectory computation rate during landing [1]. For Mars, the 8 minute landing sequence was about a third of the two-way Earth-Mars communication delay [8, 9]. Hence, the physics (of the dynamic portions) of spaceflight do not yield themselves well to ground-based calculation. As for Earth-based applications, the curvature of the Earth would require multiple ground stations around the globe while large vehicle fleets would impose communication bandwidth constraints. In summary, on-board computation, when it is possible, is the ideal engineering choice. Already during the Apollo era, the MIT Instrumentation Laboratory recognized the benefits of a fully self-contained on-board autonomy system in order to not compromise the mission due to broken or jammed ground communication [10].

Now that we have established that real-time on-board computation is an important enough feature to be studied, let us ask the following question. What does it take for a real-time algorithm to be trusted in a safety-critical, multi-million (or billion) dollar system? Perhaps a system that human lives depend on? At least part of the answer lies with recognizing two crucial implicit requirements for a "real-time on-board" algorithm: A) the algorithm must be *guaranteed* to execute fast enough, and B) the algorithm must be *guaranteed* to return a correct solution. In the context of this dissertation, to guarantee means to provide a mathematical proof of favorable algorithm behavior under a well-defined set of circumstances. We say that the algorithm behaves *predictably* when the evolution of all governing properties has been mathematically proven. We refer to properties for which mathematical

theorems are available as *provable* properties. Having provable properties is essential for deployment on safety-critical systems.

This dissertation only studies *optimization-based* algorithms. The aim of such algorithms is to solve problems of the form:

$$\min_{x} \; J(x) \tag{1.1a}$$

$$\text{s.t. } x \in \mathcal{C}, \tag{1.1b}$$

where $J : \mathbb{R}^n \to \mathbb{R}$ is a *cost function*, $\mathcal{C} \subseteq \mathbb{R}^n$ is a *feasible set*, and n is the number of *decision variables*. There are two reasons why optimization is a relevant study area for autonomous control: A) appropriateness of formulation, and B) existence of efficient solution methods. Let us first discuss the formulation.

Modern and future aerospace systems seek ever greater levels of performance and efficiency while navigating ever more complex environments. Take, for example, planetary landers. Instead of landing "anywhere" on a planet, or in an open desert-like region, future missions will have to navigate challenging environments such as volcanic vents and jagged blades of ice [11, 12, 13]. Meanwhile, human planetary exploration missions will likely be preceded by cargo missions, requiring landings to occur in close proximity [14]. Motivated by the presence of water ice, future missions to the Moon will target its south pole [15]. The low tilt of the Moon's spin axis, however, creates extreme light-dark lighting conditions at the poles, thus requiring an automated sensor-based landing [16]. If we look at terrestrial applications, unmanned micro aerial vehicles (MAVs), also known as drones, as well as ground and walking robots are being developed for rescue operations in global positioning system (GPS)-denied cluttered and dynamic environments [17], for last-mile delivery in urban landscapes [18, 19, 20], for exploring caves [21], and for mapping farm fields [22], to name just a few applications. At the same time, the vehicle to be controlled for accomplishing these high-level tasks is becoming itself more complicated. New designs feature more flexible structures [23], bio-inspired flight [24], compliant actuation [25], microscopic size, and reconfigurability

[26, 27]. If we were pressed to summarize the current trend in cutting-edge autonomy for aerospace and robotic systems, it is a trend to create a diverse pool of systems that operate at far greater levels of performance and with far more interaction with their surrounding environment.

Perhaps somewhat counterintuitively, robotic systems that can operate in far more *unconstrained* environments require the solution of far more *constrained* decision making problems. Here are a few examples. A flexible aircraft will have to deal with more coupling between its rigid-body and flexible modes [23]. A tunnel-navigating drone will have to avoid infinitely diverse obstacles [21]. A robotic fly will have to be aware of the limitations of its small actuation mechanism, the wings. A planetary lander will have to satisfy a multitude of navigation sensor pointing constraints as it identifies hazardous geography while seeking to land on a dwindling fuel supply [28]. To design a successful control strategy for such systems means to be *feasible* with respect to their constraints. Often, efficient operation requires operation at the boundary of the set of feasible solutions, in other words to "ride" the constraints. By virtue of the feasible set $\mathcal{C}$ in (1.1b), optimization is one of the few suitable methods (and is perhaps the most natural method) to directly impose system constraints [29].

Rarely, however, is there only one feasible solution. For example, there are many ways to drive without crashing – but certain driving behaviors are better than others. Slow acceleration, smooth breaking, lane change indication and respecting stop signs are all parts of good driving. How is this behavior, among all possible behaviors, to be encouraged in an autonomous system? A natural choice is to construct a cost function $J : \mathbb{R}^n \to \mathbb{R}$ which assigns a *cost* to each choice of n decision variables. When the cost is low, the decision is good, and the best decision x^* is one which achieves the lowest possible cost $J(x^*)$. Thus, by including (1.1a), the optimization formulation allows to select the best control strategy. To go back to our driving example, there is ongoing work to use machine learning to determine the "right" cost function for good driving [30, 31]. Summarizing, by combining (1.1a) with (1.1b), optimization can naturally describe favorable control of highly constrained dynamical systems.

An appropriate formulation, however, does not get us far if no algorithm exists to solve Problem 1.1 efficiently. Fortunately, it turns out that efficient algorithms do exist for an important class of so-called *convex* optimization problems (see Chapter 2 for more detail) [32, 33, 34]. A great number of practical instances of Problem 1.1 are convex, and recent applications were demonstrated for planetary rocket landing by the NASA Jet Propulsion Laboratory (JPL) and Masten Space Systems as well as (independently) by SpaceX [9, 35, 36].

Even so, future system requirements surpass what can be done with "vanilla" convex optimization, so that solving nonconvex optimization problems will be required [28]. For nonconvex optimization, fewer real-time algorithms with provable properties are available, their operating assumptions are more limiting, and their computational demand is higher. Nevertheless, by exploiting problem structure, it has been possible to harness convex optimization to solve special cases of difficult nonconvex optimal control problems reliably [37, 38]. The topic of this dissertation is to extend the palette of such methods available to the control engineer. By harnessing existing convex optimization problem solvers, this dissertation devises modeling methods and iterative schemes in order to reliably solve practical instances of nonconvex control problems encountered in aerospace applications.

1.1 Reliable Algorithm Design Avenues

What are the possible avenues to design real-time algorithms when the original problem statement is nonconvex? Three avenues are explored in this dissertation:

1. **Modeling**: Find an equivalent convex problem and solve it instead of the difficult original problem;

2. **Precomputation**: Compute an approximation of the optimal solution in a pre-defined system operating region, and store it in a static database for on-board referencing;

3. **Iteration**: Iteratively compute a local optimum of the nonconvex problem by solving

a sequence of convex approximations to the problem.

The modeling avenue uses mathematical proofs, in particular optimal control theory [39, 40, 41], to show that the optimal solution of a so-called convex "relaxation" (in other words, approximation) exactly matches the optimal solution of the original nonconvex problem. It is also sometimes possible to show via mathematical manipulation that sets which appear nonconvex at first look, may in fact be handled in a fully convex formulation [38]. As may be expected, this is only possible in a very specialized set of circumstances. Nevertheless, the results are useful in certain applications and have in fact been flight-proven for planetary rocket landing [1, 35].

The modeling avenue is in some sense "ideal" since all nonconvexity is removed by paper mathematical calculations and human thought. What is actually written in software is a fully convex optimization for which efficient solution algorithms exist [33, 42, 43, 44, 45]. But as problems grow in nonconvexity, mathematical tools quickly fall short of being able to convexify everything. In such cases, a provable method of solving the nonconvex problem is to precompute (an approximation of) the optimal solution using powerful ground-based computers, and to store the resulting lookup table database on-board the vehicle[2]. This database can be referenced efficiently at a tiny compute cost, albeit with a high static memory storage requirement. Fortunately, for earth-based applications such as in aerial drones and ground robots, terabytes of on-board commercial off-the-shelf (COTS) flash memory are available for purchase [48]. The fundamental research task, then, is to develop sufficiently capable precomputation algorithms that can scale to meaningfully complex high-dimensional real-world systems. Through the algorithm developed in Chapter 4, we shall see that this is a non-trivial task and remains an active area of research.

Many nonconvex problems simply cannot be molded into either the modeling or the precomputation frameworks. For example, their nonconvexities may be too great to be dealt with by smart variable and constraint manipulation. Furthermore, their size may be too

[2]This approach is often called explicit (in other words, precomputed) model predictive control [46, 47].

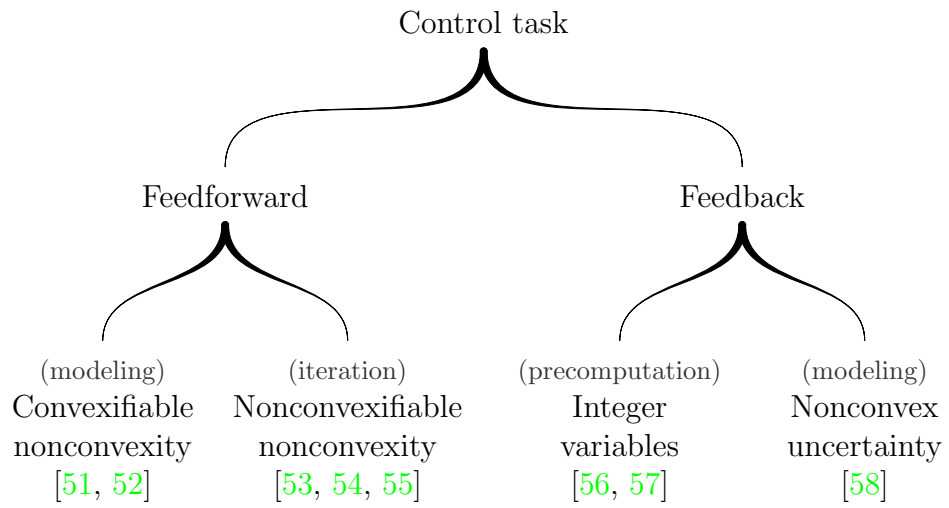
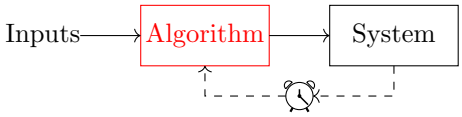


Figure 1.1: Research branches explored in this dissertation.

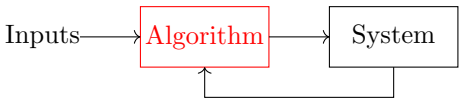
large, leading to intolerably slow runtimes and large memory footprints for precomputation methods (that typically have combinatorial worst-case computation complexity). In these cases, researches and engineers have found that an iterative approach called *sequential convex programming* (SCP) is uniquely well suited to solve problems in real-time along with some hope of guaranteed convergence [49, 50].

This dissertation develops novel algorithms for all three avenues: modeling, precomputation, and iteration. Four problems within control system development are addressed, as shown in Figure 1.1. From a bird's-eye view, control systems engineering can be divided into two branches: feedforward and feedback systems. Let us briefly explain the goal of these systems, with no pretense of covering either field in its full generality.

At the risk of providing an unconvential explanation, we propose the following mental model: feedforward (Figure 1.2a) concerns the command of a dynamical system, while feedback (Figure 1.2b) concerns the modification of system dynamics. Feedback algorithms can stabilize an unstable system, make the outputs of a stable system track desired reference values, or make the system behaviour agnostic to external disturbances. Manipulation of system behaviour is the very purpose of feedback control. Feedforward control has a dif-



(a) Feedforward control architecture.



(b) Feedback control architecture.

Figure 1.2: Comparison of feedforward and feedback algorithms in relation to the system being controlled.

ferent purpose. Within the context of this disseration, feedforward control assumes that the dynamical system is well-behaved in the sense that it can follow the commands given to it. In other words, feedforward assumes that the dynamical system has been feedback-stabilized. Then, the task of a feedforward algorithm is to decide how the system should evolve over time at a high level – in other words to be the decision-making "brain" of the system. Feedforward can tell a rocket what trajectory to follow for landing, tell a robot how to navigate an obstacle course, or provide a strategy for how to optimally manage a supply warehouse. Although feedforward is not about modifying system dynamics, knowing the system state can be useful for *replanning* in order to account for the unforeseen effects of real-world uncertainty. As a result, a feedback loop may be established from the system to the feedforward algorithm, as shown in Figure 1.2a. However, this loop does not serve a stabilization purpose like it does for feedback control, and is not a fundamental part of feedforward algorithm analysis.

1.2 Primary Contributions

Figure 1.1 shows a map of the four algorithms developed in this dissertation. Along the modeling avenue for feedforward control, the rocket landing work started by [35] and generalized by [59, 60, 61], is extended into the domain of mixed-integer programming [51, 52]. It is shown how direct convex optimization can still be used to solve even more difficult rocket landing and satellite docking tasks than was previously thought possible.

Along the modeling avenue for feedback control, it is shown how an important class of

state and input dependent uncertainty can be handled with convex model predictive control [58]. An application is demonstrated for a satellite reaction control system (RCS) with a popular type of uncertainty known as the Gates thruster execution-error model [62, 63, 64].

Along the precomputation avenue for feedback control, a massively parallel algorithm is developed for handling convex model predictive control with integer variables [56, 57]. Mixed-integer convex optimizers are not certifiably real-time due to the combinatorial nature of branch-and-bound solution [65]. Instead, the algorithm presented here computes an arbitrarily close approximation of the optimal solution and stores it in a binary tree, which can be provably referenced in real-time on-board the vehicle.

Perhaps the most complex case considered in this dissertation is the iteration avenue for feedforward problems. Here, a high-dimensional dynamical system is assumed and the nonconvexity of the optimization task exceeds the application domain of [51, 52]. This kind of high-dimensional nonconvex problem is the traditional candidate for being solved with an SCP algorithm [66, 67, 68]. Algorithms based on SCP perform remarkably well for highly nonconvex problems [69] and can achieve real-time execution speeds [70]. However, SCP traditionally allows only for continuous optimization variables. When the problem contains discrete logic elements (such as valves and other on-off events), then traditional SCP cannot be used. To overcome the issue, a continuous embedding was recently introduced under the name of state-triggered constraints (STCs) [67, 68, 69, 71]. However, it was discovered in [54] that STCs can exhibit into a phenomenon called "locking" for an important class of constraints that includes an ever-present minimum impulse-bit (MIB) constraint on RCS thruster firing duration [54, Definition 1]. In brief terms, locking means that once the algorithm chooses a set of decision variable values at a particular iteration, it is unable to change these values at later iterations. The effect is that the algorithm is susceptible to getting into a "corner" where it is unable to use control inputs when they become needed at later refinements of the trajectory. The consequence is failure to generate a feasible, let alone optimal, trajectory. This thesis develops a new breed of STCs in Chapter 7 which are free from the locking phenomenon of previous formulations [55]. The new approach is based

on numerical continuation, whereby a smooth approximation of discrete logic is updated until it becomes arbitrarily accurate. In the process, the algorithm is able to find feasible trajectories that satisfy discrete logic constraints – all without ever using integer variables or mixed-integer programming.

1.3 Document Outline

The dissertation is organized as follows. Because optimal control theory and convex analysis underly all of the algorithms presented in the later chapters, Chapter 2 introduces these topics at a sufficient level. Next, the algorithms mapped in Figure 1.1 are developed in the following order. First, Chapter 3 discusses feedback modeling using the method of robust model predictive control. This is followed by Chapter 4 on feedback precomputation using explicit model predictive control. We then delve into the topic of feedforward control (i.e., trajectory) optimization. We begin with a history of lossless convexification in Chapter 5. Building on this foundation, Chapter 6 presents a novel algorithm for computing feedforward trajectories of hybrid systems using lossless convexification. Finally, Chapter 7 considers an iterative method to solve even more difficult hybrid system problems. This method is based on the methods of sequential convex programming and numerical continuation.

Chapter 2
CONVEXITY AND OPTIMAL CONTROL THEORY

The algorithms developed in this dissertation rely heavily on convex analysis, convex optimization and optimal control theory (in particular, the maximum principle [39]). To make this text more self-contained, this chapter introduces the main notions from convex analysis and optimal control. For a deeper understanding of these vast fields of mathematics, the reader is referred to the cited references.

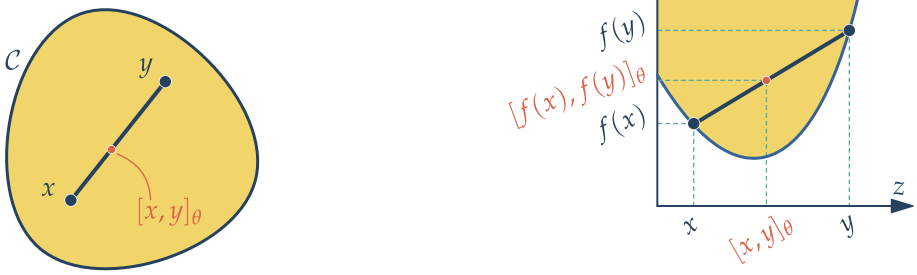
2.1 Convex Optimization

Convex optimization seeks to minimize a convex objective function while satisfying a set of convex constraints. The technique is expressive enough to capture many trajectory generation and control applications, and is appealing due to the availability of solution algorithms with the following properties [32, 34]:

- A globally optimal solution is found if a feasible solution exists;
- A certificate of infeasibility is provided when a feasible solution does not exist;
- The runtime complexity is polynomial in the problem size;
- The algorithms can self-initialize, eliminating the need for an expert initial guess.

The above properties are fairly general and apply to most if not all trajectory generation or control applications. This makes convex programming safer than other optimization methods for autonomous applications with no human presence.

To appreciate what makes an optimization problem convex, we introduce some basic definitions here and defer to [34, 72] for further details. Two fundamental objects must



(a) A convex set contains all line segments connecting its points.

(b) A convex function lies below all line segments connecting its points.

Figure 2.1: Illustration of a notional convex set (a) and convex function (b). In both cases, the variable $\theta \in [0, 1]$ generates a line segment between two points. The epigraph epi $f \subseteq \mathbb{R}^n \times \mathbb{R}$ is the set of points which lie above the function, and itself defines a convex set.

be considered: a convex function and a convex set. For reference, Figure 2.1 illustrates a notional convex set and function. By definition, $\mathcal{C} \subseteq \mathbb{R}^n$ is a convex set if and only if it contains the line segment connecting any two of its points:

$$x, y \in \mathcal{C} \Rightarrow [x, y]_\theta \in \mathcal{C} \tag{2.1}$$

for all $\theta \in [0, 1]$, where $[x, y]_\theta \triangleq \theta x + (1 - \theta)y$. An important property is that convexity is preserved under set intersection. This allows us to build complicated convex sets by intersecting simpler sets. By replacing the word "sets" with "constraints", we can readily appreciate how this property plays into modeling trajectory generation problems using convex optimization.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if dom f is a convex set and the function lies below the line segment connecting any two of its points:

$$x, y \in \text{dom } f \Rightarrow f([x, y]_\theta) \leq [f(x), f(y)]_\theta \tag{2.2}$$

for all $\theta \in [0,1]$. A convex optimization problem is simply the minimization of a convex function subject to a number of convex constraints that act to restrict the search space:

$$\min_{x \in \mathbb{R}^n} \; f_0(x) \tag{2.3a}$$

$$\text{s.t. } f_i(x) \leq 0, \; i = 1, \ldots, n_{\text{ineq}}, \tag{2.3b}$$

$$g_i(x) = 0, \; i = 1, \ldots, n_{\text{eq}}, \tag{2.3c}$$

where $f_0 : \mathbb{R}^n \to \mathbb{R}$ is a convex *cost* function, $f_i : \mathbb{R}^n \to \mathbb{R}$ are convex inequality constraints, and $g_i : \mathbb{R}^n \to \mathbb{R}$ are affine equality constraints. We stress that the equality constraints must be affine, which means that each function g_i is a linear expression in x plus a constant offset. The equations of motion are equality constraints, therefore basic convex optimization restricts the dynamics to be affine (i.e., linear time-varying at most). Handling nonlinear dynamics will be a major topic of discussion throughout this article.

Each constraint defines a convex set so that, together, (2.3b) and (2.3c) form a convex feasible set of values that the decision variable x may take. To explicitly connect this discussion back to the generic convex set introduced in (2.1), we can write the feasible set as:

$$\begin{aligned} \mathcal{C} = \big\{ x \in \mathbb{R}^n \; : \; & f_i(x) \leq 0, \; i = 1, \ldots, n_{\text{ineq}}, \\ & g_i(x) = 0, \; i = 1, \ldots, n_{\text{eq}} \big\}. \end{aligned} \tag{2.4}$$

A fundamental consequence of convexity is that any local minimum of a convex function is a global minimum [34, Section 4.2.2]. More generally, convex functions come with a plethora of properties that allow algorithm designers to obtain global information about function behavior from local measurements. For example, a differentiable convex function is globally lower bounded by its local first-order approximation [34]. Thus, we may see the benefit of convexity in the following light: it is an assumption on function behavior that enables efficient algorithm design. Indeed, a landmark discovery of the twentieth century was that

it is convexity, not linearity, that separates "hard" and "easy" problems [73].

For practitioners, the utility of convex optimization stems not so much from the ability to find the global minimum, but rather from the ability to find it (or indeed any other feasible solution) *quickly*. The field of numerical convex optimization was invigorated by the interior-point method (IPM) family of optimization algorithms, first introduced in 1984 by Karmarkar [74]. Today, convex optimization problems can be solved by primal-dual IPMs in a few tens of iterations [34, 75]. Roughly speaking, we can say that substantially large trajectory generation problems can usually be solved in under one second [1, 42, 70]. In technical parlance, IPMs have a favorable polynomial problem complexity: the number of iterations required to solve the problem to a given tolerance grows polynomially in the number of constraints $n_{\text{ineq}} + n_{\text{eq}}$. With some further assumptions, it is even possible to provide an upper bound on the number of iterations [76, 77]. We defer to [32, Chapters 14 and 19] for futher details on convex optimization algorithms. Throughout this article, our goal will be to leverage existing convex problem solvers to create higher-level frameworks for the solution of trajectory generation problems.

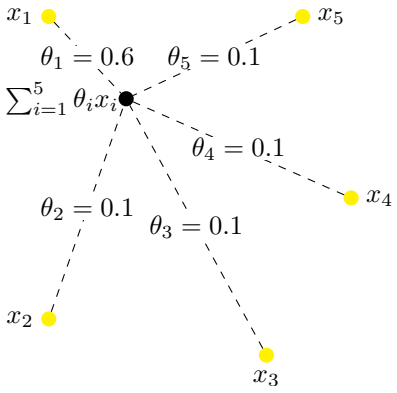
2.2 Important Notions from Convex Geometry

In this section we introduce some important notions from convex geometry that will be used throughout the text. First, given a set of points $x_1, \ldots, x_k$, the *convex combination* of these points is given by $\sum_{i=1}^{k} \theta_i x_i$ where $\theta_i \in [0, 1]$ and $\sum_{i=1}^{k} \theta_i = 1$. We can think of a convex combination as a "mixing" operation where each θ_i represents the "concentration" of x_i in the total mix. This is illustrated in Figure 2.2a.

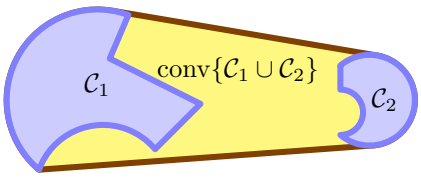
Building on this definition, the *convex hull* of a set $\mathcal{C} \subseteq \mathbb{R}^n$, denoted $\operatorname{conv} \mathcal{C}$, is the set of all convex combinations of its points:

$$\operatorname{conv} \mathcal{C} = \left\{ \sum_{i=1}^{k} \theta_i x_i : x_i \in \mathcal{C}, \theta_i \geq 0, \sum_{i=1}^{k} \theta_i = 1 \right\}, \tag{2.5}$$

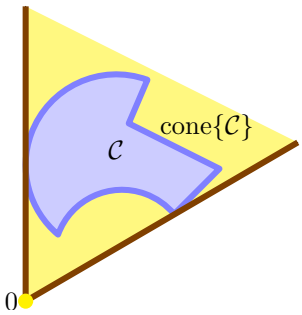
where $k = \infty$ is possible for dense sets. As the name suggests, $\operatorname{conv} \mathcal{C}$ is convex because



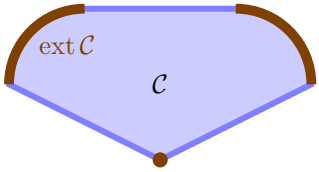
(a) Convex combination of 5 points, with x_1 dominating.



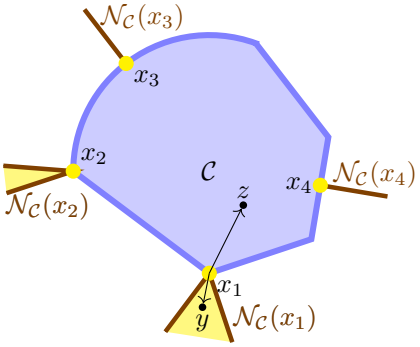
(b) Convex hull of a set $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$, generating a convex set.



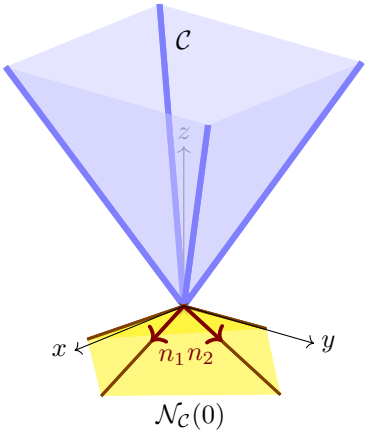
(c) Conic hull of a set $\mathcal{C}$, generating a convex cone.



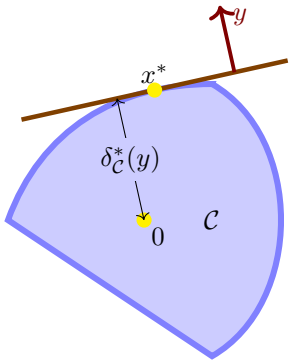
(d) The extreme points of set $\mathcal{C}$ are highlighted by the red lines.



(e) Normal cone to a generic convex set $\mathcal{C}$.



(f) Normal cone to a convex polytopic cone $\mathcal{C}$.



(g) Support function to set $\mathcal{C}$ for a given $y \in \mathbb{R}^n$.

Figure 2.2: Illustration of several convex geometry concepts, which shall be reused throughout the text.

the convex hull operation "fills in" the non-convex areas. The operation is illustrated in Figure 2.2b. Similarly, a *conic hull* of $\mathcal{C}$, denoted $\operatorname{cone}\mathcal{C}$, is the set of all conic combinations of its points:

$$\operatorname{cone}\mathcal{C} = \left\{ \sum_{i=1}^{k} \theta_i x_i : x_i \in \mathcal{C}, \theta_i \geq 0 \right\}. \tag{2.6}$$

Given a set $\mathcal{C} \subseteq \mathbb{R}^n$, we call an *extreme point* of $\mathcal{C}$ a point $v \in \mathcal{C}$ which does not lie in any open line segment joining two points of $\mathcal{C}$. The set of such points shall be denoted $\operatorname{ext}\mathcal{C}$. Intuitively, $\operatorname{ext}\mathcal{C}$ generalized the vertices of a polytope, which are in fact the polytope's extreme points. Figure 2.2d illustrates the set of extreme points.

Give a convex set $\mathcal{C} \subseteq \mathbb{R}^n$, the *normal cone* to $\mathcal{C}$ at $x \in \mathcal{C}$, denoted $\mathcal{N}_{\mathcal{C}}(x)$, is the set of all vectors which form a non-positive inner product with vectors that emanate from x to other points in $\mathcal{C}$. Formally, the normal cone for a convex set is defined as [78]

$$\mathcal{N}_{\mathcal{C}}(x) = \{y \in \mathbb{R}^n : y^{\intercal}(z - x) \leq 0 \ \forall z \in \mathcal{C}\}. \tag{2.7}$$

For a general convex set $\mathcal{C}$, examples of normal cones are illustrated in Figure 2.2e. However, when $\mathcal{C}$ is a polyhedral cone of the form $\mathcal{C} = \{x \in \mathbb{R}^n : n_i^{\intercal} x \leq 0, \ i = 1, \ldots, p\}$ then its normal cone is given by $\operatorname{cone}\{n_1, \ldots, n_m\}$. This is illustrated in Figure 2.2f, where a facet generated by $\operatorname{cone}\{n_1, n_2\}$ is highlighted.

Last but not least, the *support function* $\delta^*_{\mathcal{C}} : \mathbb{R}^n \to \mathbb{R}$ of a set $\mathcal{C} \subseteq \mathbb{R}^n$ is defined as

$$\delta^*_{\mathcal{C}}(y) = \sup_{x \in \mathcal{C}} y^{\intercal} x, \tag{2.8}$$

which can be visualized as the smallest distance from the origin to a supporting hyperplane of $\mathcal{C}$ with normal vector y, in other words a hyperplane which touches $\mathcal{C}$ at some $x \in \mathcal{C}$ and for which $\mathcal{C} \subseteq \{z : y^{\intercal} z \leq y^{\intercal} x\}$. Figure 2.2g illustrates this interpretation.

2.3 The Maximum Principle

The maximum principle states the necessary conditions of optimality for a solution of a so-called optimal control problem. The result was developed in the mid-1950s by Lev Pontryagin in collaboration with his students, Vladimir Boltyanskii and Revaz Gamkrelidze. Notably, the proof of the maximum principle was obtained by Vladimir Boltyanskii [79]. The original maximum principle was published in a joint monograph in Russian in 1961 [80], with an English translation appearing a year later by collaboration with Lucien Neustadt of the University of Southern California [81]. The maximum principle will be used in Chapter 6 to prove that convex relaxations of a certain class of difficult non-convex optimal control problems are optimal for the original non-convex problems.

We shall now present a more modern maximum principle result than the original discovery of the 1950s, known as the *nonsmooth* maximum principle. Its main advantages are to unify mixed state-input constraint handling and to be free of many encumbering hypotheses such as constraint qualifications [82]. This dissertation requires only a restricted presentation of the nonsmooth maximum principle, which we provide by considering the following optimal control problem:

$$\min_{u,t_f} \; m(t_f, x(t_f)) + \int_0^{t_f} \ell(t, u(t), x(t))\mathrm{d}t \tag{2.9a}$$

$$\text{s.t. } \dot{x}(t) = f(t, x(t), u(t)), \quad x(0) = x_0, \tag{2.9b}$$

$$g(t, u(t)) \leq 0, \tag{2.9c}$$

$$b(t_f, x(t_f)) = 0. \tag{2.9d}$$

The state trajectory $x : \mathbb{R} \to \mathbb{R}^n$ is absolutely continuous and the control trajectory $u : \mathbb{R} \to \mathbb{R}^m$ is measurable. The dynamics $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are convex and continuously differentiable. The terminal cost $m : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$, the running cost $\ell : \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$, the input constraint $g : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^{n_g}$, and the terminal constraint $b : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^{n_b}$ are convex. Note a subtle but important difference: Problem 2.9 is a *dynamic* optimization problem

whereas Problem 2.3 is *static* [83, 84]. While the latter can be thought of as optimization "at a point in time", dynamic optimization involves system dynamics (2.9b). The dynamics of mechanical systems are usually given in continuous-time, hence Problem 2.9 is *infinite-dimensional* due to the continuous nature of the sought-after optimal input signal. This motivates the development of new methods for seeking the optimal solution. On the analysis side, the maximum principle (to be given shortly in Theorem 1) is the method of choice. Although typically intractable to solve in real-time, the conditions provided by Theorem 1 are invaluable for exploiting optimal solution structure in efficient real-time solvers. Note that although static and dynamic optimization appear to be quite different at first, approximate solutions to dynamic optimization problems can be obtained via static optimization by the process of discretization, yielding a family of *direct* solution methods [85]. In this dissertation, discretization and direct solution are used to numerically solve optimal control problems.

To state the nonsmooth maximum principle for Problem 2.9, define the terminal manifold as $\mathcal{T} \triangleq \{x \in \mathbb{R}^n : (2.9d) \text{ holds}\}$ and the Hamiltonian function:

$$H(t, x(t), u(t), \alpha, \psi(t)) \triangleq \alpha \ell[t] + \psi(t)^\mathsf{T} f[t], \tag{2.10}$$

where $\alpha \leq 0$ is the *abnormal multiplier* and $\psi : \mathbb{R} \to \mathbb{R}^n$ is the *adjoint variable* trajectory. We now state the nonsmooth maximum principle, due to [86, Theorem 8.7.1] (see also [82, 87]), which specifies the necessary conditions of optimality for Problem 2.9. To understand the background and the development of this theorem, the reader is referred to [41] which provides an excellent introduction. For this dissertation, the theorem is to be taken as a "mathematical recipe" for deriving the necessary conditions of optimality.

Theorem 1. *(Maximum Principle) Let $x : \mathbb{R} \to \mathbb{R}^n$ and $u : \mathbb{R} \to \mathbb{R}^m$ be optimal on the interval $[0, t_f]$. There exist a constant $\alpha \leq 0$ and an absolutely continuous $\psi : \mathbb{R} \to \mathbb{R}^n$ such that the following conditions are satisfied:*

1. *Non-triviality:*

$$(\alpha, \psi(t)) \neq 0 \ \forall t \in [0, t_f]; \tag{2.11}$$

2. Pointwise maximum:

$$u(t) = \underset{v \in (2.9c)}{\operatorname{argmax}} H(t, x(t), v, \alpha, \psi(t)) \ \ a.e. \ [0, t_f]; \tag{2.12}$$

3. The differential equations and inclusions:

$$\dot{x}(t) = \nabla_{\psi} H[t]^{\intercal} \ \ a.e. \ [0, t_f], \tag{2.13a}$$

$$\dot{\psi}(t) \in -\partial_x H[t]^{\intercal} \ \ a.e. \ [0, t_f], \tag{2.13b}$$

$$\dot{H}[t] \in \partial_t H[t] \ \ a.e. \ [0, t_f]; \tag{2.13c}$$

4. Transversality:

$$\psi(t_f) \in \alpha \partial_x m[t_f]^{\intercal} + \mathcal{N}_{\mathcal{T}}(x(t_f)), \tag{2.14a}$$

$$0 \in H[t_f] + \alpha \partial_t m[t_f] + \mathcal{N}_{\mathcal{T}}(t_f). \tag{2.14b}$$

Chapter 3

ROBUST MODEL PREDICTIVE CONTROL WITH NONCONVEX UNCERTAINTY

This chapter presents a computationally efficient robust model predictive control (RMPC) law for discrete linear time invariant systems subject to additive disturbances that may depend on the state and/or input norms. Despite the dependency being non-convex, it is shown that at most a second-order cone convex program is needed for the control task. Both open-loop and semi-feedback planning strategies are presented [88]. The formulation has linear complexity in the planning horizon length, making the approach amenable to efficient real-time implementation with guaranteed recursive feasibility and global optimality. Robust position control of a satellite using the Gates thruster execution-error model is considered as an illustrative example [62].

Within the context of real-time optimization-based control algorithms developed in this dissertation, this algorithm takes the modeling avenue as discussed in Chapter 1. Figure 3.1 situates the algorithm within the research branches of Figure 1.1. This work was partially supported by the National Science Foundation, grant number CMMI-1613235, and was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Government sponsorship acknowledged. Particular thanks go to Martin Cacan, David S. Bayard, Daniel P. Scharf, Jack Aldrich and Carl Seubert for their helpful insight and discussions.

3.1 Introduction

It was mentioned in Chapter 1 that the optimization Problem 1.1 is advantageous in its ability to describe favorable control of highly constrained dynamical systems. The level

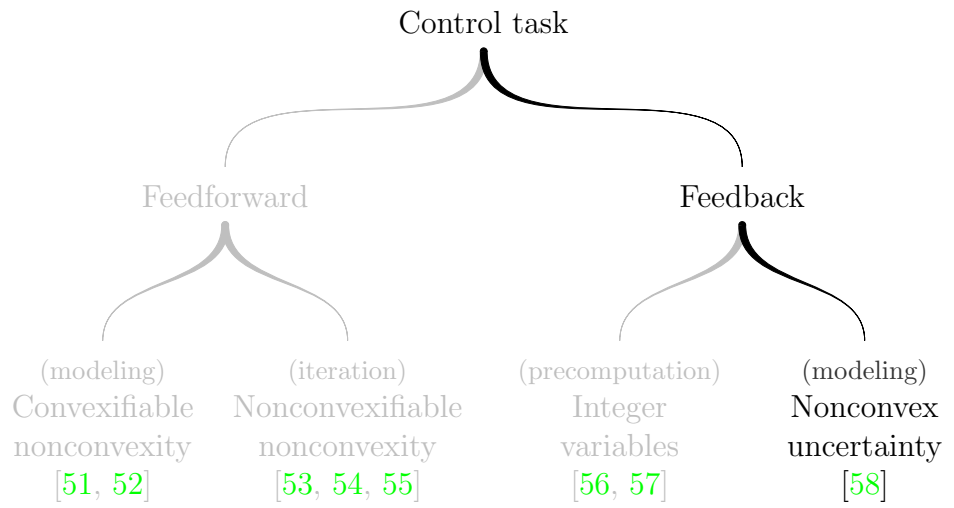


Figure 3.1: This chapter considers the modeling approach to real-time optimization-based feedback control with provable convergence and recursive feasibility.

of uncertainty in the system can be seen as a form of constraint – control must remain adequate despite imperfect knowledge. In this context, a major advantage of RMPC is its ability to guarantee by design that input and state constraints are satisfied for all uncertainty realizations.

Several extensive survey papers cover available RMPC modeling assumptions and solution methods [88, 89]. We restrict our attention to discrete linear time invariant systems and focus on developing a real-time implementable algorithm on computationally constrained hardware. We assume a perfect dynamical model of the system, that is affected by additive bounded uncertainty. The resulting RMPC law takes is given for both open-loop and semi-feedback formulations [88, Section 8]. Note that these strategies merely refer to the method of handling uncertainty within the RMPC optimization problem. In each case, the on-line implementation is done in the traditional feedback sense of re-solving and applying the first value in the optimal control history at every time step. There is evidence that this description is a reasonable choice [90, 91, 92]. Although several authors have considered closed-loop formulations, such approaches typically suffer from combinatorial complexity in

the problem dimension unless certain restrictive assumptions are made in terms of cost norm or feedback type [47, 93, 94, 95, 96, 97].

Closest to our work, [98, 99, 100, 101] use pre-computed constraint tightening factors to guarantee robustness to worst-case uncertainty through a set of linear constraints. The computational cost is marginally higher than that of nominal MPC and the on-board problem is at most a quadratic program (QP). A similar idea is exploited for nonlinear systems in [92]. State and input dependent uncertainty has received some attention in nonlinear MPC [102, 103, 104]. These formulations, however, are conservative due to their use of a Lipschitz constant for constraint tightening, which considers uncertainty magnitude but not direction. A less conservative formulation is considered in [38] via tube MPC [105], where an incrementally conic uncertainty is used to assure robustness via a static feedback component obtained by an off-line linear matrix inequality procedure and an on-line nominal MPC law.

The main contribution of this chapter is to show how a state and input dependent uncertainty model relevant for spaceflight can be included in RMPC while retaining low computational complexity. An open-loop formulation is presented first, followed by a less conservative semi-feedback extension where a static feedback gain is embedded into the finite-horizon planning problem [94, 106]. The state and input dependent uncertainty model is effectively a subset of [38] but has the advantage of using the more computationally efficient second-order cone programming (SOCP) for a robust solution. Unlike [102, 103, 104], the method presented here captures disturbance direction dependency via Hölder's inequality and is therefore less conservative. To demonstrate the method in practice, the popular Gates thruster execution-error uncertainty model for spacecraft RCS systems is used in a satellite robust position control task [62, 63, 64]. Because this method has linear complexity in the planning horizon length and is at most a convex SOCP problem, the algorithm is theoretically amenable to real-time implementation [42, 43, 107, 108].

3.2 Problem Formulation

This section describes the control problem and, in particular, defines the state and input dependent uncertainty model. Consider a discrete linear time invariant system with additive uncertainty:

$$x_{k+1} = Ax_k + Bu_k + Dp_k, \tag{3.1}$$

where $k \in \mathbb{Z}_+$, $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ and $p_k \in \mathbb{R}^d$ are, respectively, the time step, state, input and uncertainty, while A, B and D are constant matrices of commensurate dimension. The following state and input constraints are to be respected for all time:

$$x_k \in \mathcal{X} \subset \mathbb{R}^n \quad \forall k \in \mathbb{Z}_+, \tag{3.2a}$$

$$u_k \in \mathcal{U} \subset \mathbb{R}^m \quad \forall k \in \mathbb{Z}_+, \tag{3.2b}$$

where $\mathcal{X}$ and $\mathcal{U}$ are compact convex polytopes that we express as follows:

$$\mathcal{X} \triangleq \{x \in \mathbb{R}^n : Fx \leq f\}, \tag{3.3a}$$

$$\mathcal{U} \triangleq \{u \in \mathbb{R}^m : Hu \leq h\}, \tag{3.3b}$$

where the rows of F and H, and the elements of f and h, define the constraint polytope facet normals and distances respectively. Consider now that the uncertainty term in (3.1) can be any vector that satisfies

$$p_k \in \mathcal{P}(x_k, u_k) \subset \mathbb{R}^d, \tag{3.4}$$

where the uncertainty set is defined by:

$$\mathcal{P}(x_k, u_k) \triangleq \{Ww + \sum_{i=1}^{n_q} L_i q_i \in \mathbb{R}^d : Rw \leq r, \|q_i\|_{p_{q,i}} \leq \phi_i(\|F_{x,i}x_k\|_{p_{x,i}}, \|F_{u,i}u_k\|_{p_{u,i}}),\ i = 1, \ldots, n_q\}. \tag{3.5}$$

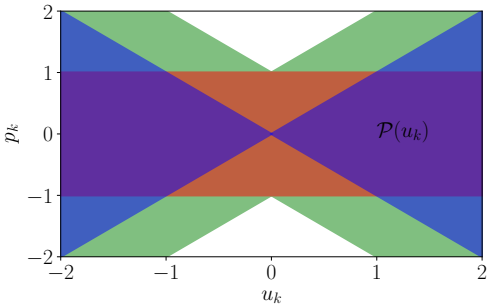


Figure 3.2: Example of a non-convex input dependent uncertainty set $\mathcal{P}(u_k)$ (green). The set is expressed as the Minkowski sum of a polytopic independent component (red) and a conic dependent component (blue).

Each function $\phi_i : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ is convex and non-decreasing such that $\phi_i(\|\cdot\|_{p_{x,i}}, \|\cdot\|_{p_{u,i}})$ is convex [34]. The uncertainty set is effectively the sum of an independent component originating from a polytope and a state and input dependent component that is bounded by a non-convex inequality. Because unbounded disturbances are not practical, we assume that $\mathcal{P}(x_k, u_k)$ is compact $\forall x_k \in \mathcal{X},\ u_k \in \mathcal{U}$. Note that $\mathcal{P}(x_k, u_k)$ is a non-convex set. A simple example of (3.5) for $n = m = d = n_q = 1$ and no state dependency is illustrated in Figure 3.2, where:

$$W = 1,\ R = \begin{bmatrix} 1 \\ -1 \end{bmatrix},\ r = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ L_1 = 1,\ \phi_1(|u_k|) = |u_k|.$$

3.3 Control Law Description

This section develop the RMPC law with planning horizon length $N \in \mathbb{Z}_{++}$ that solves the control problem presented in Section 4.2. We begin with a set theoretic motivation for how the state constraint (3.2a) may be robustly satisfied given the input constraint (A.6) and the uncertainty (3.4).

Definition 1. A compact convex set $\mathcal{I}$ is feasible robust controlled invariant (fRCI) for

system (3.1) and constraints (3.2a), (A.6) and (3.4) if it satisfies the following definition:

$$\mathcal{I} = \{x \in \mathcal{X} : \exists u \in \mathcal{U} \text{ s.t. } Ax + Bu + Dp \in \mathcal{I}\ \forall p \in \mathcal{P}(x, u)\}. \tag{3.6}$$

By constraining $x_k \in \mathcal{I} \subseteq \mathcal{X}$, the state constraint can thus be feasibly satisfied for all time as long as $x_0 \in \mathcal{I}$. Numerous set-based iterative methods are available for computing the maximal volume $\mathcal{I}$ [109, 110, 111]. Approaches like [111] that attempt to deal with state and input dependent uncertainty, however, scale poorly beyond $n = 2$ because they require computing set differences and projections of polytopes. Both operations have $\mathcal{O}(2^n)$ complexity [112]. For this reason we use a more computationally efficient algorithm developed for independent uncertainty and which avoids these two operations [110]. This requires using the convex hull of our uncertainty model:

$$p_k \in \text{conv}\{\mathcal{P}(x_k, u_k) : x_k \in \mathcal{X},\ u_k \in \mathcal{U}\}. \tag{3.7}$$

While this introduces conservatism into the computation of $\mathcal{I}$ by considering a larger uncertainty set, the approach is more scalable to higher dimensions given existing methods for computing fRCI sets. Furthermore, Corollary 1 in Section 3.4 presents a computationally efficient method for checking if $\mathcal{X}$ itself is fRCI, at which point the computation of $\mathcal{I}$ can be avoided altogether as described in Algorithm 1.

The proof of Theorem 2 in Section 3.4 shows that $\mathcal{I}$ is a convex set and may therefore be inner-approximated by a polytope. We henceforth assume that this polytopic description is available:

$$\mathcal{I} \triangleq \{x \in \mathbb{R}^n : Gx \leq g\}, \tag{3.8}$$

where $G \in \mathbb{R}^{n_g \times n}$ and $g \in \mathbb{R}^{n_g}$.

Since $\mathcal{I}$ is fRCI, for each $x_k \in \mathcal{I}$ $\exists u_k \in \mathcal{U}$ such that $x_{k+1} \in \mathcal{I}$ $\forall p_k \in \mathcal{P}(x_k, u_k)$. We compute such a u_k via a tightened state constraint in an optimization problem. Albeit conservative, constraint tightening is an efficient method of inducing robustness in MPC [47]. Letting x_k

be the current state, the system (3.1) has the following impulse response over $t = 1, \ldots, N$ future time steps:

$$x_{k+t} = A^t x_k + \sum_{i=0}^{t-1} B_{t,i}^A u_{k+i} + \sum_{i=0}^{t-1} D_{t,i}^A p_{k+i}, \tag{3.9}$$

where $B_{t,i}^A \triangleq A^{t-1-i}B$ and $D_{t,i}^A \triangleq A^{t-1-i}D$. An invariance-preserving input sequence over the N-step planning horizon satisfies:

$$G\Big(A^t x_k + \sum_{i=0}^{t-1} B_{t,i}^A u_{k+i} + \sum_{i=0}^{t-1} D_{t,i}^A p_{k+i}\Big) \leq g, \tag{3.10}$$

for all sequences $p_{k+i} \in \mathcal{P}(x_{k+i}, u_{k+i})$, $i = 0, \ldots, t-1$ and $t = 1, \ldots, N$. Similarly to [98, 113] we reformulate this requirement by maximizing the left hand side of (3.10):

$$G_j^\intercal \bar{x}_{k+t} + \max_{\substack{p_{k+i} \in \mathcal{P}(x_{k+i}, u_{k+i}) \\ i=0,\ldots,t-1}} \sum_{i=0}^{t-1} G_j^\intercal D_{t,i}^A p_{k+i} \leq g_j, \tag{3.11}$$

for $j = 1, \ldots, n_g$ and $t = 1, \ldots, N$, where $\bar{x}_{k+t}$ denotes the nominal state after t time steps:

$$\bar{x}_{k+t} \triangleq A^t x_k + \sum_{i=0}^{t-1} B_{t,i}^A u_{k+i}. \tag{3.12}$$

The maximization term in (3.11) encodes the worst-case disturbance sequence over t time steps for violating the j-th facet of the state constraint polytope (3.2a). However, the cost function of the problem

$$\max_{\substack{p_{k+i} \in \mathcal{P}(x_{k+i}, u_{k+i}) \\ i=0,\ldots,t-1}} \sum_{i=0}^{t-1} G_j^\intercal D_{t,i}^A p_{k+i} \tag{3.13}$$

is not separable, which makes the selection of the worst-case uncertainty sequence up to time t a fully coupled problem over the time steps $i = 0, \ldots, t-1$. While it is possible to find a conservative upper bound to (3.13) in terms of the initial state x_k and the input sequence u_{k+i} for $i = 0, \ldots, t-1$, simulation experience with the example in Section 3.6 shows that this maximum singular value-based bound is far too conservative. A better approach is to

simply consider the uncertainty set $\mathcal{P}(\bar{x}_{k+i}, u_{k+i})$ based on the nominal state. This makes the cost function of (3.13) separable, allowing to write:

$$\sum_{i=0}^{t-1} \max_{p_{k+i} \in \mathcal{P}(\bar{x}_{k+i}, u_{k+i})} G_j^\intercal D_{t,i}^A p_{k+i} = \sum_{i=0}^{t-1} \delta^*_{D_{t,i}^A \mathcal{P}(\bar{x}_{k+i}, u_{k+i})}(G_j), \tag{3.14}$$

where the last formulation uses the definition of the support function. The following example makes the transition from (3.13) to (3.14) more intuitive by considering a simple case.

Example 1. Let us begin with (3.13) and consider the following simple case: $t = 2$, $A = I \in \mathbb{R}^{n \times n}$, $D = I \in \mathbb{R}^{n \times d}$, and $G_j = e_j \in \mathbb{R}^n$ is the j-th Eucledian basis unit vector. Let the uncertainty model (3.5) be only-state dependent:

$$\mathcal{P}(x_{k+i}, u_{k+i}) = \{q \in \mathbb{R}^d : \|q\|_2 \leq \|x_{k+i}\|_2\}. \tag{3.15}$$

Then, (3.13) simplifies to:

$$\max_{p_k, p_{k+1}} \; e_j^\intercal (p_k + p_{k+1}) \tag{3.16a}$$

$$\text{s.t. } \|p_k\|_2 \leq \|x_k\|_2, \tag{3.16b}$$

$$\|p_{k+1}\|_2 \leq \|x_k + Bu_k + p_k\|_2. \tag{3.16c}$$

Because the decision variables p_k and p_{k+1} are coupled through the constraint (3.16c), the cost (3.16a) is not separable. Furthermore, the constraint (3.16c) is not convex. However, a separable convex approximation exist by replacing (3.16c) with:

$$\|p_{k+1}\|_2 \leq \|x_k + Bu_k\|_2 + \|p_k\|_2, \tag{3.17}$$

which yields:

$$\max_{\text{(3.16b) holds}} e_j^\intercal p_k + \max_{\text{(3.17) holds}} e_j^\intercal p_{k+1}. \tag{3.18}$$

Note that (3.18) gives a more conservative (in other words, larger) constraint tightening

factor to work with for (3.11) than the proposed non-conservative version in (3.14), which would yield:

$$\max_{\text{(3.16b) holds}} e_j^\intercal p_k + \max_{\|p_{k+1}\|_2 \leq \|x_k + Bu_k\|_2} e_j^\intercal p_{k+1}. \tag{3.19}$$

Simulations for the example in Section 3.6 have shown that (3.19) is far less conservative than (3.18). Because it is still robust in an RMPC receding horizon implementation, we prefer (3.19) over (3.18).

Using (3.14) has no impact at $t = 1$ since $x_k = \bar{x}_k$. The same cannot be said over the remaining $N - 1$ steps because perturbed states with a larger norm may occur, inducing a larger uncertainty than the one predicted by $\mathcal{P}(\bar{x}_{k+i}, u_{k+i})$. However, since the control law is implemented in a receding horizon fashion and u_k is anyway the only input to be applied, the implementation remains robust. Furthermore, the formulation remains exact over the entire planning horizon for only-input dependent uncertainty. With this in mind, the support function in (3.14) is simplified using the separable nature of (3.5) and Hölder's inequality [34]:

$$\delta^*_{D^A_{t,i}\mathcal{P}(\bar{x}_{k+i},u_{k+i})}(G_j) = \delta^*_{D^A_{t,i}\mathcal{W}}(G_j) + \sum_{l=1}^{n_q} \|G_j^\intercal D^A_{t,i} L_l\|_{q_{q,l}} \phi_l(\|F_{x,l}\bar{x}_{k+i}\|_{p_{x,l}}, \|F_{u,l}u_{k+i}\|_{p_{u,l}}), \tag{3.20}$$

where the $q_{q,l}$-norm is dual to the $p_{q,l}$-norm in (3.5), so that $1/q_{q,l} + 1/p_{q,l} = 1$. Note that the first term on the right-hand side of (3.20) is the polytopic independent uncertainty part of $\mathcal{P}(\bar{x}_{k+i}, u_{k+i})$. Further note that (3.20) is not conservative since Hölder's inequality is tight [34]. Using (3.20), we can approximate (3.11) as a set of Nn_g convex constraints:

$$G_j^\intercal \bar{x}_{k+t} + \sum_{i=0}^{t-1} \delta^*_{D^A_{t,i}\mathcal{W}}(G_j) + \sum_{i=0}^{t-1}\sum_{l=1}^{n_q} \|G_j^\intercal D^A_{t,i} L_l\|_{q_{q,l}} \phi_l(\|F_{x,l}\bar{x}_{k+i}\|_{p_{x,l}}, \|F_{u,l}u_{k+i}\|_{q_{u_l}}) \leq g_j, \tag{3.21}$$

for $j = 1, \ldots, n_g$ and $t = 1, \ldots, N$. Importantly, $\delta^*_{D^A_{t,i}\mathcal{W}}(G_j)$ can be pre-computed off-line, leading to a faster on-line implementation. The RMPC on-line optimization problem to be implemented in receding horizon fashion can thus be written as:

$$\min_{u_k,\ldots,u_{k+N-1}} J(u_k, \ldots, u_{k+N-1}) \tag{3.22a}$$

$$\text{s.t. Constraints (A.6) and (3.21) hold.} \tag{3.22b}$$

The cost function $J : (\mathbb{R}^m)^N \to \mathbb{R}$ is a design choice which should be convex. If J is linear and the 1- or ∞-norms are used for all ϕ_i in (3.5), this is a linear program (LP). If J is quadratic, it is a QP. In all cases when the 2-norm is used in the ϕ_l functions, Problem 3.22 is an SOCP. Note that all of these problem types are convex and have efficient solvers available that guarantee convergence to the global optimum when the feasible set is non-empty (which it is certified to hold across iterations in Section 3.4). Furthermore, the constraint count of Problem 3.22 is $\mathcal{O}(N)$ owing to the welcome property that the effect of the approximate worst-case uncertainty on a discrete time linear system is explicitly given by (3.20), avoiding a combinatorial search. Algorithm 1 summarizes the off-line and on-line steps that make up the full RMPC controller.

3.4 Main Results

This section proves in Theorem 2 that the control law given by Problem 3.22 is recursively feasible. Furthermore, Corollary 1 provides an alternative sufficient condition for certifying the set $\mathcal{X}$ to be fRCI. This helps to avoid calling the more laborious set-based algorithm of [110] for computing $\mathcal{I}$.

Theorem 2. *The control law given by Problem 3.22 is recursively feasible if and only if it is feasible at the vertices of $\mathcal{I}$.*

Proof. Let $\mathcal{V} \triangleq \{v_1, ..., v_M\}$ be the set of vertices of $\mathcal{I}$. The forward implication is trivial. If Problem 3.22 is recursively feasible then it must be feasible in particular when $x_k \in \mathcal{V}$. For

Algorithm 1 Off-line and on-line RMPC steps.

Off-line:
if Corollary 1 is satisfied for $\mathcal{I} = \mathcal{X}$ **then**
  $\mathcal{I} \leftarrow \mathcal{X}$
else
  Compute $\mathcal{I}$ using the set-based algorithm in [110]
  if Corollary 1 is not satisfied for $\mathcal{I}$ **then**
    Report error "N is too large"
  end if
end if
Store $\delta^*_{D^A_{t,i}\mathcal{W}}(G_j)$ for $j = 1, \ldots, n_g$, $t = 1, \ldots, N$, $i = 0, \ldots, t-1$
On-line:
Obtain the current state x
Set $x_k \leftarrow x$, solve Problem 3.22 and apply u_k
Sleep T_s seconds ▷ Discretization time step

the reverse implication, suppose that we solve Problem 3.22 with x_k set to each vertex v_m, $m = 1, \ldots, M$, and obtain the associated optimal input sequences u^m_{k+t}, $t = 0, \ldots, N-1$, and the nominal state sequences $\bar{x}^m_{k+t}$, $t = 0, \ldots, N$, as given by (3.12):

$$\bar{x}^m_{k+t} = A^t v_m + \sum_{i=0}^{t-1} B^A_{t,i} u^m_{k+i}.$$

Consider now a state $x_k \in \mathcal{I}$. Since $\mathcal{I}$ is a convex polytope, we can express x_k as a convex combination of $\mathcal{I}$'s vertices:

$$x_k = \sum_{m=1}^{M} \theta_m v_m, \quad \sum_{m=1}^{M} \theta_m = 1, \ \theta_m \geq 0 \ \forall m = 1, \ldots, M.$$

It is now possible to sum the constraint (3.21) applied at each vertex, weighted by θ_m, to obtain:

$$G_j^\intercal \sum_{m=1}^{M} \theta_m \bar{x}^m_{k+t} + \sum_{i=0}^{t-1} \delta^*_{D^A_{t,i}\mathcal{W}}(G_j) + \sum_{i=0}^{t-1} \sum_{l=1}^{n_q} \|G_j^\intercal D^A_{t,i} L_l\|_{q_{q,l}} \sum_{m=1}^{M} \theta_m \phi_l(\|F_{x,l} \bar{x}^m_{k+i}\|_{p_{x,l}}, \|F_{u,l} u^m_{k+i}\|_{p_{u,l}}) \leq g_j.$$

Because each ϕ_l is convex, it follows from Jensen's inequality that:

$$\phi_l(\|F_{x,l}\sum_{m=1}^{M}\theta_m\bar{x}_{k+i}^m\|_{p_{x,l}},\|F_{u,l}\sum_{m=1}^{M}\theta_m u_{k+i}^m\|_{p_{u,l}}) \leq \sum_{m=1}^{M}\theta_m\phi_l(\|F_{x,l}\bar{x}_{k+i}^m\|_{p_{x,l}},\|F_{u,l}u_{k+i}^m\|_{p_{u,l}}),$$

and as a result we have:

$$G_j^\intercal\sum_{m=1}^{M}\theta_m\bar{x}_{k+t}^m+\sum_{i=0}^{t-1}\delta^*_{D_{t,i}^A\mathcal{W}}(G_j)+\sum_{i=0}^{t-1}\sum_{l=1}^{n_q}\|G_j^\intercal D_{t,i}^A L_l\|_{q_{q,l}}\phi_l(\|F_{x,l}\sum_{m=1}^{M}\theta_m\bar{x}_{k+i}^m\|_{p_{x,l}},\|F_{u,l}\sum_{m=1}^{M}\theta_m u_{k+i}^m\|_{p_{u,l}})\leq g_j. \quad (3.23)$$

Because $\mathcal{U}$ is convex and $u_{k+i}^m \in \mathcal{U}\ \forall m=1,\dots,M$ and $i=0,\dots,N-1$, we have $\sum_{m=1}^{M}\theta_m u_{k+i}^m \in \mathcal{U}\ \forall i=0,\dots,N-1$. Furthermore:

$$\sum_{m=1}^{M}\theta_m\bar{x}_{k+t}^m = A^t x_k+\sum_{i=0}^{t-1}B_{t,i}^A\sum_{m=1}^{M}\theta_m u_{k+i}^m.$$

We thus recognize that (3.23) is nothing but (3.21) for the initial state x_k and the feasible input sequence $\sum_{m=1}^{M}\theta_m u_{k+i}^m$, $i=0,\dots,N-1$. The feasible set of Problem 3.22 is thus non-empty. Since this control input ensures that $x_{k+1}\in\mathcal{I}$ under the worst-case disturbance, Problem 3.22 continues to be feasible at the next time step, which means that it is recursively feasible. □

The following corollary provides a faster method than computational geometry approaches for verifying that a particular polytope is fRCI. This is used in Algorithm 1 for checking if $\mathcal{X}$ in (3.2a) is fRCI, avoiding an unnecessary call to the more time consuming set-based algorithm in [110]. This is particularly useful for higher dimensional systems.

Corollary 1. *A sufficient condition for $\mathcal{X}$ to be fRCI is for Problem 3.22 to be feasible at its vertices. The condition becomes also necessary when $N=1$.*

Proof. The sufficient condition follows from Theorem 2 and the fact that (3.21) yields the fRCI property by design whenever Problem 3.22 is recursively feasible. Because (3.21) is not conservative when $N = 1$, the condition is necessary in that case. □

3.5 Extension to Semi-feedback RMPC

The RMPC law given by Problem 3.22 can be overly conservative because it does not model feedback action, meaning that the control policy $\{u_k, \dots, u_{k+N-1}\}$ cannot mitigate the effect of disturbances during the planning stage. This is why such an RMPC strategy is sometimes referred to as an "open-loop" formulation [88]. Indeed, the disturbance action terms $\delta^*_{D_i^A \mathcal{W}}(G_j)$ and $\|G_j^\intercal D_i^A L_l\|_{q_{q,l}}$ in (3.21) are independent of the control policy. This section extends Problem 3.22 to semi-feedback RMPC, which introduces feedback action into the planning stage without significantly increasing computational complexity [94, 106]. To achieve this, a static feedback gain $K \in \mathbb{R}^{m \times n}$ is designed and the control input is re-defined as

$$u_k = v_k + K x_k, \tag{3.24}$$

where $v_k \in \mathbb{R}^m$ becomes the optimized decision variable. The nominal state (3.12) becomes:

$$\bar{x}_{k+t} = \tilde{A}^t x_k + \sum_{i=0}^{t-1} B_{t,i}^{\tilde{A}} v_{k+i}, \tag{3.25}$$

where $\tilde{A} = A + BK$. The robust constraint (3.21) becomes:

$$G_j^\intercal \bar{x}_{k+t} + \sum_{i=0}^{t-1} \delta^*_{D_{t,i}^{\tilde{A}} \mathcal{W}}(G_j) + \sum_{i=0}^{t-1} \sum_{l=1}^{n_q} \|G_j^\intercal D_{t,i}^{\tilde{A}} L_l\|_{q_{q,l}} \phi_l(\|F_{x,l} \bar{x}_{k+i}\|_{p_{x,l}}, \\ \|F_{u,l}(v_{k+i} + K \bar{x}_{k+i})\|_{q_{u_l}}) \leq g_j, \tag{3.26}$$

where, by similar reasoning to Example 1, we used the nominal state for the uncertainty's input dependence. Applying the discussion in Section 3.3, this choice has no impact at $t = 1$ so the RMPC law remains robust when implemented in receding horizon fashion. With these

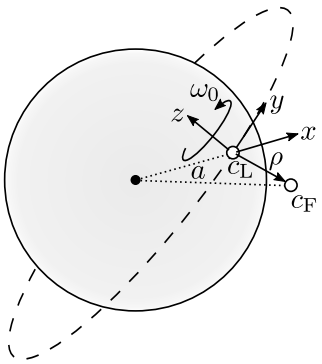


Figure 3.3: LVLH frame showing the leader and follower satellites.

changes, semi-feedback RMPC can be implemented in the same way as Problem 3.22.

3.6 Illustrative Example

In this section the control law presented in Section 3.3 and Section 3.5 is applied to satellite robust position control in low Earth orbit (LEO). Consider a formation of two satellites as shown in Figure 3.3 with a leader c_{L} in a circular orbit with radius a, and a follower c_{F}. The follower's translation in a local-vertical local-horizontal (LVLH) frame is given by the Clohessy-Wiltshire equations (time is omitted for notational simplicity):

$$\ddot{x} = 3\omega_0^2 x + 2\omega_0 \dot{y} + u_x + w_x, \tag{3.27a}$$

$$\ddot{y} = -2\omega_0 \dot{x} + u_y + w_y, \tag{3.27b}$$

$$\ddot{z} = -\omega_0^2 z + u_z + w_z, \tag{3.27c}$$

where $\omega_0 = \sqrt{\mu/a^3}$ and μ is the standard gravitational parameter. Let $\rho = (x, y, z) \in \mathbb{R}^3$ denote the follower's position relative to the leader, where the coordinate x is not to be confused with the state vector. Using the state $x \triangleq (\rho, \dot{\rho}) \in \mathbb{R}^6$, input $u = (u_x, u_y, u_z) \in \mathbb{R}^3$ and exogenous disturbance $w = (w_x, w_y, w_z) \in \mathbb{R}^3$, the dynamics (3.27) take on the familiar linear form:

$$\dot{x} = A_{\mathrm{c}} x + B_{\mathrm{c}}(u + w). \tag{3.28}$$

We use an impulsive control input in which the RCS system can induce every T_s seconds an instantaneous velocity increment $\Delta v(\tau)$ at time τ via the control input $u(t) = \Delta v(\tau)\delta(t-\tau)$, where δ is the Dirac delta. Meanwhile, w is assumed to be a constant acceleration over the T_s period. These assumptions allow (3.28) to be discretized [114]:

$$x_{k+1} = Ax_k + Bu_k + Ew_k, \tag{3.29}$$

$$A = e^{A_c T_s}, \quad B = AB_c, \quad E = \int_0^{T_s} e^{A_c(T_s - t)} B \mathrm{d}t.$$

We include three uncertainty sources:

1. Atmospheric drag, modeled as an independent component $w_k \in \{w : \|w\|_\infty \leq w_{\max}\}$;

2. Additive input error using the Gates thrust execution-error model. This accounts for error in the RCS system's reproduction of a desired Δv velocity increment [62, 63, 64]. Because (3.5) does not capture directional dependency, we confine ourselves to an isotropic description. In fact, this may anyway the best modeling choice if the satellite's design is unknown [62]. The error term is $v_k = v_k^{\text{fix}} + v_k^{\text{prop}}$ where $v_k^{\text{fix}} \in \{v : \|v\|_2 \leq \sigma_{\text{fix}}\}$ and $v_k^{\text{prop}} \in \{v : \|v\|_2 \leq \sigma_{\text{rcs}}\|u_k\|_2\}$;

3. Additive state estimation error $e_k = e_k^{\text{fix}} + \begin{bmatrix} I \\ 0 \end{bmatrix} e_k^{\text{pos}} + \begin{bmatrix} 0 \\ I \end{bmatrix} e_k^{\text{vel}}$ where:

$$\begin{aligned}
e_k^{\text{fix}} &\in \{e \in \mathbb{R}^6 : \| \begin{bmatrix} I & 0 \end{bmatrix} e\|_\infty \leq p_{\max}, \| \begin{bmatrix} 0 & I \end{bmatrix} e\|_\infty \leq v_{\max}\}, \\
e_k^{\text{pos}} &\in \{e \in \mathbb{R}^3 : \|e\|_\infty \leq \sigma_{\text{pos}}\| \begin{bmatrix} I & 0 \end{bmatrix} x_k\|_2\}, \\
e_k^{\text{vel}} &\in \{e \in \mathbb{R}^3 : \|e\|_\infty \leq \sigma_{\text{vel}}\| \begin{bmatrix} 0 & I \end{bmatrix} x_k\|_2\}.
\end{aligned}$$

Altogether, these form a set of independent and dependent polytopic and spherical un-

certainties that area readily described by (3.5). In particular,

$$d = 21,\ n_q = 4,\ w = \begin{bmatrix} w_k \\ e_k^{\text{fix}} \end{bmatrix},\ q_1 = v_k^{\text{fix}},\ q_2 = v_k^{\text{prop}},\ q_3 = e_k^{\text{pos}},\ q_4 = e_k^{\text{vel}},$$

$$p_{q,1} = p_{q,2} = 2,\ p_{q,3} = p_{q,4} = \infty,\ \phi_1 = \sigma_{\text{fix}},\ \phi_2 : u_k \mapsto \sigma_{\text{rcs}} \|u_k\|_2,$$

$$\phi_3 : x_k \mapsto \sigma_{\text{pos}} \| \begin{bmatrix} I & 0 \end{bmatrix} x_k \|_2,\ \phi_4 : x_k \mapsto \sigma_{\text{vel}} \| \begin{bmatrix} 0 & I \end{bmatrix} x_k \|_2,$$

$$D = \begin{bmatrix} E & -A & B & B & -A & -A \end{bmatrix},\ W = \begin{bmatrix} I_9 \\ 0_{18\times 9} \end{bmatrix},\ R = \begin{bmatrix} I_9 \\ -I_9 \end{bmatrix},$$

$$L_1 = \begin{bmatrix} 0_{9\times 3} \\ I \\ 0_{15\times 3} \end{bmatrix},\ L_2 = \begin{bmatrix} 0_{12\times 3} \\ I \\ 0_{12\times 3} \end{bmatrix},\ L_3 = \begin{bmatrix} 0_{15\times 3} \\ I \\ 0_{9\times 3} \end{bmatrix},\ L_4 = \begin{bmatrix} 0_{24\times 3} \\ I \end{bmatrix},$$

$$r = \begin{bmatrix} w_{\max} 1_3 \\ p_{\max} 1_3 \\ v_{\max} 1_3 \\ w_{\max} 1_3 \\ p_{\max} 1_3 \\ v_{\max} 1_3 \end{bmatrix}.$$

We use the following cost:

$$J \triangleq \sum_{t=0}^{N-1} \hat{u}_{k+t}^{\mathsf{T}} \hat{u}_{k+t} + \lambda \hat{\bar{x}}_{k+t+1}^{\mathsf{T}} \hat{\bar{x}}_{k+t+1}, \tag{3.30}$$

where the decision variables $\hat{u}_k$ and $\hat{\bar{x}}_k$ are the scaled input and nominal state respectively. Scaling is performed by diagonal scaling matrices $D_u \in \mathbb{R}^{m\times m}$ and $D_x \in \mathbb{R}^{n\times n}$, such that $u_k = D_u \hat{u}_k$ and $\bar{x}_k = D_x \hat{\bar{x}}_k$. The objective of scaling is to ensure that the scaled quantities $\hat{u}_k$ and $\hat{\bar{x}}_k$ attain ± 1 at the boundary of their respective constraint polytopes $\mathcal{U}$ and $\mathcal{I}$. We choose the trade-off weight $\lambda = 0.003$. The input penalty reflects a minimum-fuel type

problem and the state penalty endows the finite horizon control law with an otherwise lacking long-term knowledge that the origin corresponds to minimal fuel usage, since nominally it takes zero control action to remain there. In this problem, $\mathcal{X}$ in (3.2a) takes on the direct interpretation of a maximum control error specification. We use the following numerical values:

$$\mathcal{X} = \{x \in \mathbb{R}^6 : \| \begin{bmatrix} I & 0 \end{bmatrix} x\|_\infty \leq 10 \text{ cm}, \| \begin{bmatrix} 0 & I \end{bmatrix} x\|_\infty \leq 1 \text{ mm}\,\text{s}^{-1}\},$$
$$\mathcal{U} = \{u \in \mathbb{R}^3 : \|u\|_\infty \leq 2 \text{ mm}\,\text{s}^{-1}\},$$
$$\mu = 3.986 \cdot 10^{14} \text{ m}^3\,\text{s}^{-2},\ a = 6793.137 \text{ km},\ T_\text{s} = 100 \text{ s},$$
$$w_\text{max} = 50 \text{ nm}\,\text{s}^{-2},\ \sigma_\text{fix} = 1 \text{ µm}\,\text{s}^{-1},\ p_\text{max} = 0.4 \text{ cm},\ N = 4,$$
$$v_\text{max} = 4 \text{ µm}\,\text{s}^{-1},\ \sigma_\text{rcs} = \tan\frac{\pi}{180},\ \sigma_\text{pos} = 0.02,\ \sigma_\text{vel} = 0.001.$$

We compare the following four controllers:

1. **Nominal MPC** : ignores the uncertainty, effectively removing the summations in (3.21);

2. **Conservative RMPC**: replaces ϕ_i in (3.5) by its maximum over $x_k \in \mathcal{X}$ and $u_k \in \mathcal{U}$, which results in a conservative independent uncertainty model;

3. **Our open-loop RMPC**: solves Problem 3.22;

4. **Our semi-feedback RMPC**: solves Problem 3.22 with the modifications described in Section 3.5. The static feedback gain K is computed as a linear quadratic regulator (LQR) with weight matrices $Q = I_6$ and $R = 10^5 I$, scaled to $\mathcal{I}$ and $\mathcal{U}$ respectively in the same way as for (3.30).

In each case, the satellite is acted upon by all three of the uncertainty sources described above. It turns out that for this problem, Corollary 1 holds and so $\mathcal{I} = \mathcal{X}$. To demonstrate the conservatism exhibited by the conservative RMPC law, consider Figure 3.4 which

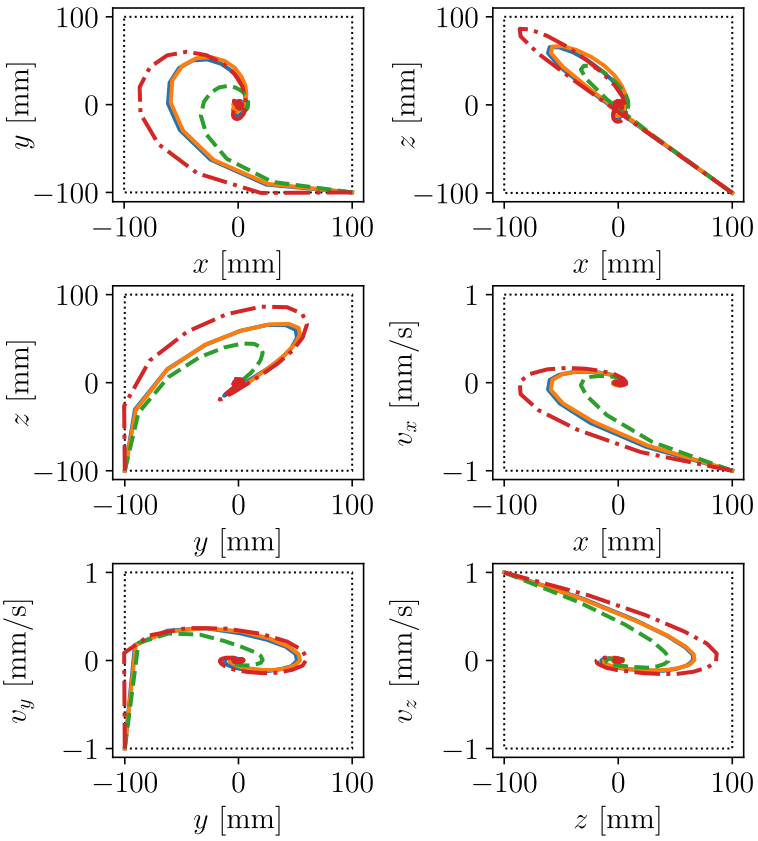


Figure 3.4: Projections of the transient response shown in dash dotted red for nominal MPC, dashed green for conservative RMPC, solid blue for our open-loop RMPC and solid orange for our semi-feedback RMPC. The dotted black rectangle shows the boundary of $\mathcal{I}$.

shows the four controllers' transient responses when starting from a vertex of $\mathcal{I}$. While nominal MPC briefly exits $\mathcal{I}$, conservative RMPC tends to drive the satellite more quickly into the interior of $\mathcal{I}$ because it assumes more uncertainty than necessary. Open-loop and semi-feedback RMPC yield similar responses that are somewhere in between nominal and conservative MPC, staying within $\mathcal{I}$ and not avoiding the boundaries of $\mathcal{I}$ unnecessarily.

The RMPC law Problem 3.22 enables the control engineer to work with a richer set of feasible parameters. For example, the required position accuracy can be increased to 5 cm without changing any other parameter, while doing so with the conservative RMPC law requires using $N \leq 2$. The open-loop RMPC law works for $N \leq 4$ before the passively propagated uncertainty "outgrows" $\mathcal{I}$. The less conservative semi-feedback RMPC law works for $N \leq 6$.

Next, we compare the fuel consumption and computational efficiency of the three controllers. Fuel consumption is quantified by the sum of velocity increment magnitudes commanded per year as obtained from a linear regression, yielding units of $\mathrm{m\,s^{-1}\,yr^{-1}}$. This is common in space applications, where a mission is characterized by a "Δv budget", which is not to be exceeded over the mission duration. Computational efficiency is measured by the time taken to solve Problem 3.22. Since both quantities are affected by uncertainty, 2000 Monte Carlo simulations are run for each controller where the satellite is initialized at the origin and controlled over four orbits.

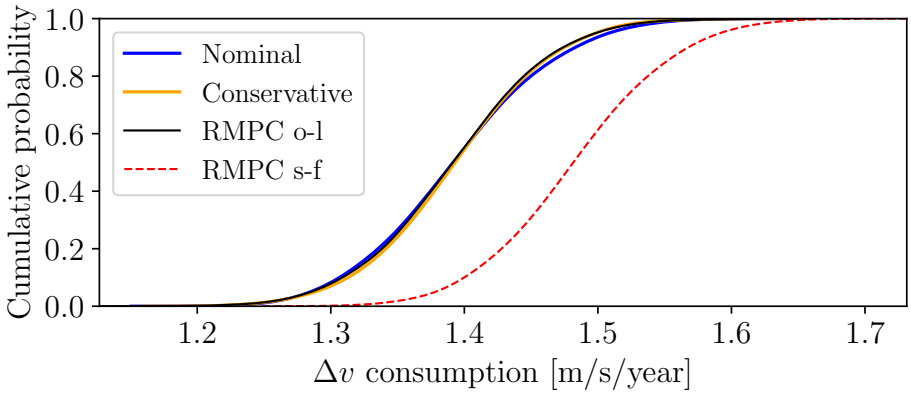
Figure 3.5a shows the fuel consumption cumulative distribution plot using a Gaussian smoothing kernel. There is no statistically significant difference between the nominal MPC, conservative RMPC and open-loop RMPC laws. As expected, the embedded LQR controller increases the fuel consumption of semi-feedback RMPC by an average amount of $0.09\ \mathrm{m\,s^{-1}\,yr^{-1}}$.

Figure 3.5b shows the solver time distribution using Python 2.7.15 with ECOS `2.0.7.post1` [43] in Ubuntu 18.04.1 with a 3.60 GHz Intel i7-6850K CPU and 64 GB of RAM. All four controllers have comparable solver times. The distributions reflect the problem difficulty hierarchy, wherein our RMPC laws are the most difficult due to second order cone (SOC)

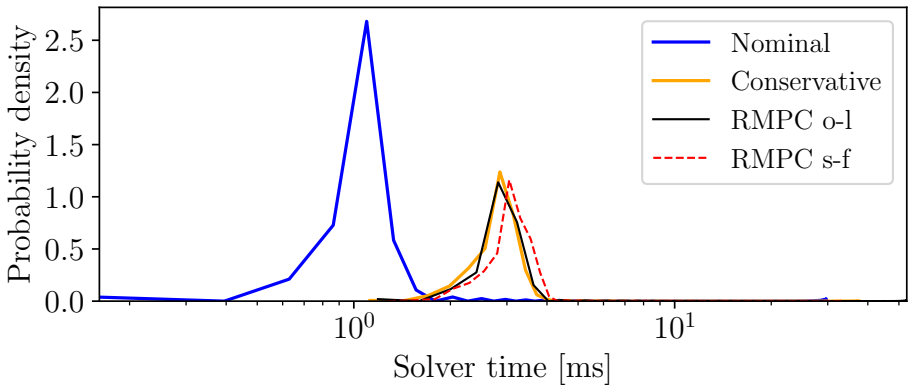
constraints. On average, nominal MPC takes 1.1 ms, conservative and open-loop RMPC take 2.9 ms, and semi-feedback RMPC takes 3 ms. These differences are all statistically significant at the 99.9% confidence level. Generally speaking, SOCP problems are amenable to real-time implementation and we expect this to be the case here [42, 43, 107, 108].

3.7 Future Work

This section introduced a state and input dependent uncertainty model (3.5) and a provable real-time and recursively feasible RMPC law in Problem 3.22, which is at most an SOCP. Both open-loop and semi-feedback implementations are possible. Simulations of a satellite system demonstrate that the approach is more versatile and less conservative than RPMC based on an independent uncertainty model. The approach, however, is currently hampered by the ability to compute a robust controlled invariant set for a high dimensional system. Recent work on controllable set computation for convex optimal control problems has the potential to remove this shortcoming [115].



(a) Fuel usage distribution for each controller from 2000 Monte Carlo simulations. *RMPC o-l* and *s-f* stand for our open-loop and semi-feedback RMPC laws respectively.



(b) Optimization solver time distribution for each controller from 2000 Monte Carlo simulations. *RMPC o-l* and *s-f* stand for our open-loop and semi-feedback RMPC laws respectively.

Figure 3.5: Fuel usage and solver time distributions for the four tested control laws.

Chapter 4

APPROXIMATE MULTIPARAMETRIC MIXED-INTEGER CONVEX PROGRAMMING

This chapter presents an algorithm for generating explicit solutions of multiparametric mixed-integer convex programs to within a given suboptimality tolerance. The algorithm is applicable to a very general class of optimization problems, but is most useful for hybrid model predictive control, where on-line implementation is hampered by the worst-case exponential complexity of mixed-integer solvers. The output is a simplicial partition which defines a static map from the current state to a suboptimal solution. The primary theoretical contribution is to introduce a non-zero optimal cost overlap metric which is necessary and sufficient for convergence. The overlap size is also linked to partition complexity. The algorithm is massively parallelizable and its implementation, which is publicly available, is run on a cluster of several hundred processors. The output of the algorithm is a static database, in the form of a binary tree, which is accessible in deterministic time and which performs faster than on-board optimization by up to three orders of magnitude.

Within the context of real-time optimization-based algorithm development, this research considers the precomputation avenue for creating efficient on-board feedback algorithms whose underlying structure is expressed as a difficult-to-solve optimization problem. Figure 4.1 places this work in the context of the research branches introduced in Figure 1.1. This work was partly supported by the National Science Foundation, grant number CMMI-1613235. The use of advanced computational, storage, and networking infrastructure was provided by the Hyak supercomputer system and funded by the student technology fee (STF) at the University of Washington. Special thanks go to Martin Cacan, David S. Bayard, Daniel P. Scharf, Jack Aldrich and Carl Seubert of the NASA Jet Propulsion Laboratory, California

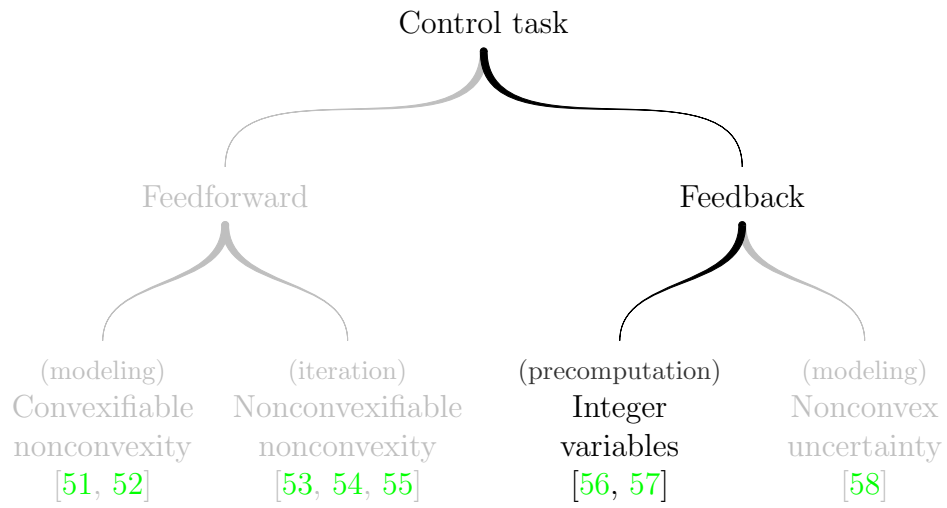


Figure 4.1: This chapter considers the precomputation approach to provably real-time optimization-based feedback control.

Institute of Technology, for their helpful insights and discussions.

4.1 Introduction

Hybrid model predictive control handles systems with discrete switches or piecewise affinely approximated nonlinearities like chemical powerplants, pipelines and aerospace vehicles [65, 116, 117]. This requires solving a mixed-integer convex program (MICP), which is hampered by the worst-case exponential complexity of mixed-integer solvers [65, 118]. In this paper, we present a provably convergent algorithm for computing explicit solutions of MICPs with a specified suboptimality tolerance.

Several approaches have been proposed to improve hybrid MPC performance. By leveraging the polynomial runtime complexity of convex solvers, successive convexification is able to solve nonlinear programs in real-time [119]. Recently, the method was extended to handle binary decision making via state-triggered constraints [54, 120]. Hence, at least some MICPs are solvable in real-time. However, successive convexification is a local method which may not always converge to a feasible solution.

The traditional method of ensuring real-time MPC performance while guaranteeing convergence and global optimality is to pre-compute the optimal solution off-line. Reflecting the approach, the family of methods for doing this are known as *explicit* MPC. Various explicit MPC methodologies have been proposed [46]. For MPC laws more complicated than linear or quadratic programs, however, pre-computation of the exact solution is generally not possible due to non-convexity of common active constraint sets [121]. Instead, approximate solutions have been proposed via local linearization [122] or via optimal cost bounding by affine functions over simplices [121] and hyperrectangles [123]. An approximate explicit solution to mixed-integer quadratic programs has been proposed based on difference-of-convex programming [124] and for MICPs based on local linearization and primal/master subproblems [125].

Our contribution in this paper is twofold. First, we introduce a massively parallelizable algorithm for computing the explicit solution to a very general class of multiparametric MICPs and to within a user-specified suboptimality tolerance. Our implementation is available online[1]. Second, we define a novel "overlap" metric which turns out to be the fundamental driver of partition complexity. A non-zero overlap is necessary and sufficient for convergence of the pre-computation algorithm. To the best of our knowledge, the overlap metric is to-date the most insightful theoretical tool for considering the partition complexity of explicit MPC algorithms similar to the one presented here [121, 123, 126, 127]. Furthermore, parallelization of explicit MPC algorithms has not yet been exploited in existing literature.

The chapter is organized as follows. Section 4.2 defines the class of optimization problems that the pre-computation algorithm can handle. Section 4.4 presents the solution algorithm. Section 4.6 proves the algorithm's convergence and complexity properties. Section 4.8 applies the method to robust control several test problem cases, including the robust control of a satellite's position. Section 4.10 concludes with future research directions.

---

[1]Hosted on GitHub: `https://github.com/dmalyuta/explicit_hybrid_mpc`.

4.2 Problem Formulation

Our algorithm can handle any problem that can be formulated as the following multiparametric MICP:

$$\delta^*, \theta^* = \min_{x,\delta} \; f(\theta, x, \delta) \tag{4.1a}$$

$$\text{s.t. } g(\theta, x, \delta) = 0, \; h(\theta, x, \delta) \in \mathcal{K}. \tag{4.1b}$$

In Problem 4.1, $\theta \in \mathbb{R}^p$ is a parameter, $x \in \mathbb{R}^n$ is a decision vector and $\delta \in \mathbb{I}^m$ is a binary *commutation*. Note that Problem 4.1 is called *multiparametric* because its optimal solution essentially depends on the value of parameter θ [128, 129]. The cost function $f : \mathbb{R}^p \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is jointly convex and the constraint functions $g : \mathbb{R}^p \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^l$ and $h : \mathbb{R}^p \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^d$ are affine in their first two arguments. The convex cone $\mathcal{K} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_q \subset \mathbb{R}^d$ is a Cartesian product of q convex cones. Examples include the positive orthant, the second-order cone and the positive semidefinite cone [34]. If δ is fixed, Problem 4.1 becomes a multiparametric convex program:

$$V_\delta^*(\theta) = \min_{x} \; f(\theta, x, \delta) \tag{4.2a}$$

$$\text{s.t. } g(\theta, x, \delta) = 0, \; h(\theta, x, \delta) \in \mathcal{K}. \tag{4.2b}$$

Problem 4.2 corresponds to Problem 4.1 where δ has been fixed, in other words assigned a specific value. The following definitions are closely related to [121].

Definition 2. The feasible parameter set $\Theta^* \subset \mathbb{R}^p$ is the set of all θ parameters for which Problem 4.1 is feasible.

Definition 3. The fixed-commutation feasible parameter set $\Theta_\delta^* \subset \mathbb{R}^p$ is the set of all θ parameters for which Problem 4.2 is feasible.

Definition 4. The feasible map $f_\delta : \Theta^* \to \mathbb{I}^m$ associates $\theta \in \Theta^*$ to a commutation such that Problem 4.2 is feasible.

Definition 5. The optimal map $f_\delta^* : \Theta^* \to \mathbb{I}^m$ associates $\theta \in \Theta^*$ to any optimal commutation of Problem 4.1, in other words any $\delta \in \{\delta \in \mathbb{I}^m : V^*(\theta) = V_\delta^*(\theta)\}$.

Definition 6. The suboptimal map $f_\delta^\epsilon : \Theta^* \to \mathbb{I}^m$ associates $\theta \in \Theta^*$ to an ϵ-suboptimal commutation δ such that

$$V_\delta^*(\theta) - V^*(\theta) < \max\{\epsilon_\mathrm{a}, \epsilon_\mathrm{r} V^*(\theta)\}, \tag{4.3}$$

where ϵ_a and ϵ_r are the absolute and relative errors.

The ultimate aim of this chapter is to reliably and efficiently compute f_δ^ϵ over a subset $\Theta \subseteq \Theta^*$. It is assumed that Θ is a full-dimensional convex polytope in vertex representation. Section 4.9.1 discusses how one might obtain Θ. It is implicitly assumed that Problem 4.1 and all related problems are appropriately scaled. We suggest a per-axis unit scaling such that $\max_{\theta\in\Theta} |e_i^\mathsf{T}\theta| = 1 \ \forall i = 1, \ldots, p$ where $\{e_i\}_{i=1}^p$ is the standard Eucledian orthonormal basis.

Figure 4.2 illustrates our explicit MPC at a high level. In order to achieve the end goal of computing the suboptimal map f_δ^ϵ, we begin by computing the feasible map f_δ. The result of this "feasible partition" computation shall seed the algorithm for computing the "ϵ-suboptimal" partition.

4.3 Computing the Feasible Map f_δ

The main idea is to express f_δ as a simplicial partition data structure, $\mathcal{R} \triangleq \{(\mathcal{R}_i, \delta_i)\}_{i=1}^{|\mathcal{R}|}$. In this data structure, $\Theta = \bigcup_{i=1}^{|\mathcal{R}|} \mathcal{R}_i$ and each cell $\mathcal{R}_i$ is associated with a fixed commutation δ_i that is feasible everywhere in $\mathcal{R}_i$. In other words, $\mathcal{R}_i \subseteq \Theta_{\delta_i}^*$. This provides sufficient

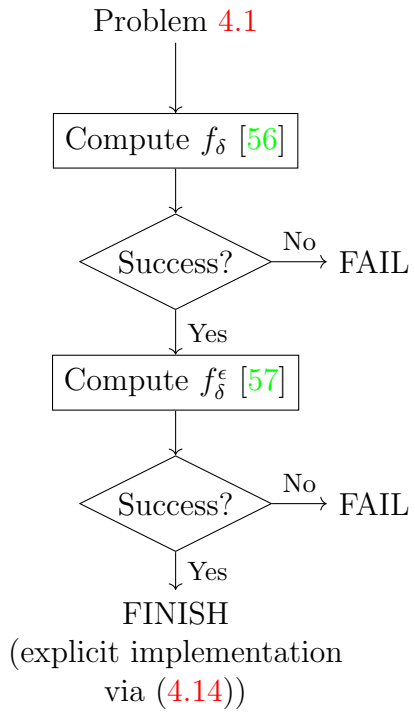


Figure 4.2: Flowchart description of the two main steps in our explicit MPC algorithm: 1) compute the feasible map f_δ, and 2) compute the suboptimal map f_δ^ϵ.

information to evaluate f_δ via:

$$f_\delta(\theta) = \delta_i \text{ such that } \theta \in \mathcal{R}_i. \tag{4.4}$$

Lemma 1. *For any fixed $\delta \in \mathbb{I}^m$, Θ^*_δ is a convex set and V^*_δ is a convex function.*

Proof. Suppose $\theta', \theta'' \in \Theta^*_\delta$. Let $\alpha', \alpha'' \in [0,1]$, $\alpha' + \alpha'' = 1$ and $\theta = \alpha'\theta' + \alpha''\theta''$. Since g, h are affine in their first argument and $\mathcal{K}$ is a convex cone:

$$g(\theta, x, \delta) = \alpha' g(\theta', x, \delta) + \alpha'' g(\theta'', x, \delta) = 0,$$
$$h(\theta, x, \delta) = \alpha' h(\theta', x, \delta) + \alpha'' h(\theta'', x, \delta) \in \mathcal{K},$$

so $\theta \in \Theta^*_\delta$ which is thus a convex set. Next, Problem 4.2 is a minimization of f over a convex set in x which preserves convexity in θ by the joint convexity property of f [34]. Thus, V^*_δ is a convex function. □

Algorithm 2 Brute force f_δ computation.

```
R ← ∅, Θ̄ ← Θ
for all δ ∈ 𝕀^m do
    R ← {(R', δ) : R' ∈ Θ̄ ∩ Θ*_δ} ∪ R
    Θ̄ ← Θ̄ \ Θ*_δ
    if Θ̄ = ∅ then
        STOP
    end if
end for
```

Since Θ^*_δ is convex, an arbitrarily precise inner approximation of it can be found [115]. A conceptually trivial method for generating $\mathcal{R}$ is given by Algorithm 2. The set difference and intersection operations in Algorithm 2 are element-wise [112]. The idea is to exploit the ability to inner approximate Θ^*_δ to procedurally "cover" Θ by intersecting it with fixed-commutation feasible parameter sets.

The filling problem is combinatorial, however, such that in the worst case all 2^m possible values of δ are needed, excepting those that are infeasible directly due to the constraints

in Problem 4.1. Furthermore, accurate polytopic inner approximation of Θ^*_δ in higher-dimensional spaces than about $\mathbb{R}^4$ suffers from excessive vertex count [115]. Last but not least, the set intersection and set difference operations used by Algorithm 2 have poor numerical properties such as creating badly conditioned (in other words, quasi lower-dimensional) polytopes. One would like instead an algorithm which 1) avoids exploring all 2^m combinations for δ, 2) minimizes vertex count and 3) uses only numerically robust operations. We thus propose Algorithm 3 in which the first requirement is addressed via Problem 4.5 discussed below, the second by using only simplices (which have the lowest vertex count among full-dimensional polytopes [130]) and the third by working solely in the vertex representation which is much more numerically robust than the halfspace representation and avoids the numerically fragile operation of converting between the two representations.

Algorithm 3 Proposed computation of f_δ.

Create empty tree with open leaf Θ as root
$\mathcal{S} \leftarrow \texttt{delaunay}(\mathcal{V}(\Theta))$
Add child open leaves $\mathcal{S}_i$ $\forall \mathcal{S}_i \in \mathcal{S}$
while any open leaf exists **do**
    $\mathcal{R} \leftarrow$ the first open leaf
    if Problem 4.1 is infeasible for $\theta = |\mathcal{V}(\mathcal{R})|^{-1} \sum_{v \in \mathcal{V}(\mathcal{R})} v$ **then**
        STOP, $(\Theta^*)^c \cap \Theta \neq \emptyset$
    else
        $\hat{\delta} \leftarrow$ solve Problem 4.5
        if Problem 4.5 is infeasible **then**
            $\bar{v}, \bar{v}' \leftarrow \arg\max_{v,v' \in \mathcal{V}(\mathcal{R})} \|v - v'\|_2$
            $v_{\text{mid}} \leftarrow (\bar{v} + \bar{v}')/2$
            Add child open leaf $\text{conv}\{(\mathcal{V}(\mathcal{R}) \setminus \{\bar{v}\}) \cup \{v_{\text{mid}}\}\}$
            Add child open leaf $\text{conv}\{(\mathcal{V}(\mathcal{R}) \setminus \{\bar{v}'\}) \cup \{v_{\text{mid}}\}\}$
        else
            Replace leaf with closed leaf $(\mathcal{R}, \hat{\delta})$
        end if
    end if
end while

Algorithm 3 creates a simplicial partition of Θ as follows. The partition is stored as a tree whose leaves are cells $(\mathcal{R}, \delta)$ storing the simplex $\mathcal{R}$ and associated commutation δ.

Non-leaf nodes in the tree store just the simplex and make evaluating (4.4) more efficient (see Section 4.7). A "closed leaf" refers to a cell that will be a leaf in the final tree while an "open leaf" will be further partitioned at the next iteration and thus will be merely a non-leaf node in the final tree. In the algorithm, we refer to nodes directly by their contents, in other words $\mathcal{R}$ for open and $(\mathcal{R}, \delta)$ for closed leaves. On Algorithm 3 L3 the tree root is initialized to Θ and, since generally Θ is not a simplex, Delaunay triangulation is first applied [131, Section 9.3]. The tree is then iterated on Algorithm 3 L4 in a depth-first manner until no open leaves are left. By doing a depth-first search, Assumption 1 in Section 4.6 is disproved more quickly in case that it does not hold, such that the algorithm fails with less wasted time.

An open leaf $\mathcal{R}$ in the tree is selected on Algorithm 3 L5. First, it is checked whether Problem 4.1 is feasible at its center of gravity. If not, this point is a certificate of infeasibility of Problem 4.1 in Θ, in which case Section 4.9.1 should be consulted. If however Problem 4.1 is feasible, then Algorithm 3 L9-16 partition $\mathcal{R}$ into at most 2 simplices. First, it is checked whether $\mathcal{R}$ is fully contained inside some particular Θ^*_δ. The following lemma is used for this purpose.

Lemma 2. $\mathcal{R} \subseteq \Theta^*_{\hat{\delta}} \Leftrightarrow$ *Problem 4.2 is feasible* $\forall \theta \in \mathcal{V}(\mathcal{R})$ *and* $\delta = \hat{\delta}$.

Proof. ($\Rightarrow$) Since $\mathcal{R} \subseteq \Theta^*_{\hat{\delta}}$, $\theta \in \mathcal{V}(\mathcal{R}) \Rightarrow \theta \in \Theta^*_{\hat{\delta}}$. Since $\Theta^*_{\hat{\delta}}$ is the fixed-commutation feasible parameter set, Problem 4.2 is by definition feasible. ($\Leftarrow$) Any θ such that Problem 4.2 is feasible satisfies, by definition, $\theta \in \Theta^*_{\hat{\delta}}$. By Lemma 1, since $\Theta^*_{\hat{\delta}}$ is convex, $\text{conv}\{\theta \in \mathcal{V}(\mathcal{R})\} \equiv \mathcal{R} \subseteq \Theta^*_{\hat{\delta}}$. □

Using Lemma 2, one can efficiently check if $\mathcal{R} \subseteq \Theta^*_\delta$ for some δ via the following feasibility MICP:

$$\hat{\delta}(\mathcal{R}) = \text{find} \;\; \delta \tag{4.5a}$$

$$\text{s.t. } g(\theta, x_\theta, \delta) = 0 \;\forall \theta \in \mathcal{V}(\mathcal{R}), \tag{4.5b}$$

$$h(\theta, x_\theta, \delta) \in \mathcal{K} \;\forall \theta \in \mathcal{V}(\mathcal{R}), \tag{4.5c}$$

$$\delta \in \mathbb{I}^m. \tag{4.5d}$$

Variable x_θ in Problem 4.5 denotes a feasible decision vector corresponding to the particular value of θ. Problem Problem 4.5 can be solved in the standard fashion (in other words, with a solver such as Gurobi or MOSEK [132, 133]) as a MICP. If a feasible commutation is found, it can be associated with $\mathcal{R}$ and the leaf can be subsequently closed. If Problem 4.5 is infeasible, however, then $\mathcal{R}$ is not fully contained in any Θ^*_δ. In this case, $\mathcal{R}$ is split into two smaller simplices at the midpoint of its longest edge. As explained in Section 4.6, this yields a volume reduction that necessarily leads to convergence if Assumption 1 holds.

Algorithm 3 enables computing the feasible map f_δ. Moving down the flowchart in Figure 4.2, the next section presents an algorithm which uses f_δ as a starting point for computing the suboptimal map f^ϵ_δ over the same subset $\Theta \subseteq \Theta^*$.

4.4 Computing the Suboptimal Map f^ϵ_δ

Similar to the computation of f_δ in the previous section, we express f^ϵ_δ as a simplicial partition data structure, $\mathcal{R}_\epsilon \triangleq \{(\mathcal{R}_i, \delta_i, \{x^*_{i,j}\}_{j=1}^{|\mathcal{V}(\mathcal{R}_i)|})\}_{i=1}^{|\mathcal{R}_\epsilon|}$. As before, $\Theta = \bigcup_{i=1}^{|\mathcal{R}|} \mathcal{R}_i$. However, in this case each cell $\mathcal{R}_i$ is associated not with a *feasible* δ_i, but an ϵ-suboptimal one. Furthermore $\{x^*_{i,j}\}_{j=1}^{|\mathcal{V}(\mathcal{R}_i)|}$, the optimal decision vectors of Problem 4.2 for the commutation δ_i at the $\mathcal{R}_i$ vertices, are carried along for the fully explicit implementation (refer to Section 4.5).

Algorithm 4 computes the $\mathcal{R}_\epsilon$ data structure and stores it as a binary tree. The initialization step sets $\mathcal{R}_\epsilon = \mathcal{R}$, the feasible partition from the previous section. Subsequently, the leaves of this tree are all set to the open state. Algorithm 4 L4-14 carry out the main work of partitioning the simplex-commutation tuple $(\mathcal{R}, \delta)$. First, the algorithm checks if δ is ϵ-suboptimal in $\mathcal{R}$. If it is not, the following mixed-integer nonlinear program (MINLP) must be feasible due to (4.3):

$$\delta^*, \theta^* = \underset{\theta \in \mathcal{R}}{\text{find}} \;\; \delta' \tag{4.6a}$$

$$\text{s.t. } V^*_\delta(\theta) - V^*_{\delta'}(\theta) \geq \max\{\epsilon_\text{a}, \epsilon_\text{r} V^*_{\delta'}(\theta)\}. \tag{4.6b}$$

Algorithm 4 Proposed computation of f_δ^ϵ.

Run Algorithm 3 and relabel all leaves as nodes
while any open leaf exists **do**
  $(\mathcal{R}, \delta) \leftarrow$ the first open leaf
  if Problem 4.8 is infeasible **then**
    Replace leaf with closed leaf $(\mathcal{R}, \delta, \{x_j^*\}_{j=1}^{|\mathcal{V}(\mathcal{R})|})$
  else
    $\delta^*, \theta^* \leftarrow$ solve Problem 4.9
    if Problem 4.9 is feasible and (4.10) holds **then**
      Change node to $(\mathcal{R}, \delta^*)$
    else
      $\delta^* \leftarrow \delta$ if Problem 4.9 is infeasible
      $v_1, v_2 \leftarrow \arg\max_{v,v' \in \mathcal{V}(\mathcal{R})} \|v - v'\|_2$
      $\mathcal{S}_i \leftarrow \mathrm{conv}\{(\mathcal{V}(\mathcal{R}) \setminus \{v_i\}) \cup \{(v_1 + v_2)/2\}\},\ i = 1, 2$
      Add child open leaves $(\mathcal{S}_1, \delta^*)$ and $(\mathcal{S}_2, \delta^*)$
    end if
  end if
end while

The costs V_δ^* and $V_{\delta'}^*$ are convex due to Lemma 1. However, V_δ^* appears on the wrong side of the inequality. Hence, Problem 4.6 is non-convex and so is not readily solvable. As a remedy, we formulate a conservative convex upper bound.

Definition 7. Let $v_i \in \mathcal{V}(\mathcal{R})$ be the i-th vertex of $\mathcal{R}$ and let $\theta = \sum_{i=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_i v_i$ where $\alpha_i \geq 0$ and $\sum_{i=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_i = 1$. The affine over-approximator of $V_\delta^*(\theta)$ over $\mathcal{R}$ is:

$$\bar{V}_\delta(\theta) \triangleq \sum_{i=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_i V_\delta^*(v_i). \blacksquare \tag{4.7}$$

Since V_δ^* is convex, $V_\delta^*(\theta) \leq \bar{V}_\delta(\theta)\ \forall \theta \in \mathcal{R}$. Hence, the following problem is convex and "conservative" in the sense of Theorem 3:

$$\delta^*, \theta^* = \operatorname*{find}_{\theta \in \mathcal{R}}\ \delta' \tag{4.8a}$$

$$\text{s.t. } \bar{V}_\delta(\theta) - V_{\delta'}^*(\theta) \geq \max\{\epsilon_\mathrm{a}, \epsilon_\mathrm{r} V_{\delta'}^*(\theta)\}. \tag{4.8b}$$

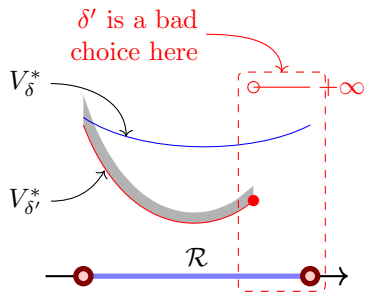


Figure 4.3: Illustration in $\mathbb{R}^2$ of the motivation for constraint (4.9c). Without the constraint, δ' would be wrongly selected as a "locally better" commutation, but one that is not feasible over the entire simplex $\mathcal{R}$. The gray zone above $V^*_{\delta'}$ is the set of ϵ-suboptimal cost values with respect to $V^*_{\delta'}$.

Theorem 3. *If Problem 4.8 is infeasible then δ is ϵ-suboptimal.*

Proof. If δ is not ϵ-suboptimal then Problem 4.6 is feasible, hence Problem 4.8 is feasible. By contraposition, if Problem 4.8 is infeasible then δ must be ϵ-suboptimal. □

Since Problem 4.8 is a MICP, its feasibility can be certified with a mixed-integer solver. If Problem 4.8 is infeasible, Algorithm 4 L5 converts the node to a leaf since no further partitioning is necessary. Otherwise, we cannot conclude about the ϵ-suboptimality of δ. In this case, we search for a potentially better commutation δ^* via the following extension of Problem 4.8:

$$\delta^*, \theta^* = \underset{\theta \in \mathcal{R}}{\text{find}} \;\; \delta' \tag{4.9a}$$

$$\text{s.t. } \bar{V}_\delta(\theta) - V^*_{\delta'}(\theta) \geq \max\{\epsilon_\text{a}, \epsilon_\text{r} V^*_{\delta'}(\theta)\}, \tag{4.9b}$$

$$\delta' \in \{\delta'' \in \mathbb{I}^m \setminus \{\delta\} : \mathcal{R} \subseteq \Theta^*_{\delta''}\}. \tag{4.9c}$$

The last constraint of Problem 4.9 ensures that δ' is feasible in $\mathcal{R}$, and can be embedded via Lemma 2. Figure 4.3 motivates the presence of this constraint, which ensures that $\mathcal{R}$ does not get associated with a commutation that is not feasible everywhere over $\mathcal{R}$.

If Problem 4.9 is infeasible, Section 4.6 shows that a sound strategy is to keep δ and

to split $\mathcal{R}$ in half at the midpoint of its longest edge, as done on Algorithm 4 L14. This may also be done if Problem 4.9 is feasible, since shrinking $\mathcal{R}$ improves the accuracy of the over-approximator (4.7). However, to avoid unnecessary partitioning, Algorithm 4 L8 checks if V_δ^* varies over $\mathcal{R}$ by less than the ϵ-suboptimality threshold with respect to $V_{\delta^*}(\theta^*)$ as output by Problem 4.9:

$$\max_{\theta\in\mathcal{R}} V_\delta^*(\theta) - \min_{\theta\in\mathcal{R}} V_\delta^*(\theta) < \max\{\epsilon_\mathrm{a}, \epsilon_\mathrm{r} V_{\delta^*}^*(\theta^*)\}, \tag{4.10}$$

which can be done with convex optimization. If (4.10) holds, Theorem 6 assures that $\mathcal{R}$ does not need to be subdivided further and δ^* is assigned directly on Algorithm 4 L9. If (4.10) does not hold, Section 4.6 shows that a sound strategy is to split $\mathcal{R}$ in half at the midpoint of its longest edge on Algorithm 4 L14.

Let us now return to the model predictive control scenario. In this context, let θ denote the current state of the system. Suppose that $\theta \in \Theta$ and, in particular, θ belongs to some simplex $\mathcal{R} \subseteq \Theta$. With the data structure $\mathcal{R}_\epsilon$ available, one can show (see Theorem 8) that it is possible to obtain the ϵ-suboptimal commutation in polynomial time (in other words, in real-time):

$$\delta = f_\delta^\epsilon(\theta) \text{ such that } \theta \in \mathcal{R}. \tag{4.11}$$

With δ available, one can then solve the remaining convex problem Problem 4.2 to obtain the optimal control input (which would be part of the optimal decision vector x^*). We call this approach a *semi-explicit* implementation, since it already simplifies the "difficult" mixed-integer solution to a convex solution for on-board implementation.

However, to further speedup the runtime and to decrease the on-board compute requirement, it is possible to remove on-line optimization entirely via a fully *explicit* implementation. To do this, $\mathcal{R}_\epsilon$ stores the optimal solution vectors x_j^* at vertices $j = 1, \ldots, |\mathcal{V}(\mathcal{R})|$ of each partition cell $\mathcal{R}$. In the next section, we show how this information can be used to algebraically obtain an ϵ-suboptimal decision vector $\hat{x}$ corresponding to the current value of θ. By this process, the original MICP optimization problem 4.1 is fully distilled into a sequence

of algebraic, optimization-free operations.

4.5 Explicit Computation of the ϵ-Suboptimal Decision Vector

Consider a cell $(\mathcal{R}, \delta, \{x_j^*\}_{j=1}^{|\mathcal{V}(\mathcal{R})|})$ of the computed data structure $\mathcal{R}_\epsilon$. We first recognize the following property.

Theorem 4. *Suppose that $\theta \in \mathcal{R}$ and let $v_j \in \mathcal{V}(\mathcal{R})$ be the j-th vertex of $\mathcal{R}$. One can write $\theta = \sum_{j=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_j v_j$ where $\alpha_j \geq 0$ and $\sum_{j=1}^{|\mathcal{V}(\mathcal{R}_j)|} \alpha_j = 1$. Let $\hat{x} \triangleq \sum_{j=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_j x_j^*$ and $\hat{V}_\delta(\theta) \triangleq f(\theta, \hat{x}, \delta)$. Then $\hat{x}$ is feasible for Problem 4.2 and*

$$\hat{V}_\delta(\theta) \leq \bar{V}_\delta(\theta). \tag{4.12}$$

Proof. For feasibility, exploit that $g(\cdot, \cdot, \delta)$ and $h(\cdot, \cdot, \delta)$ in Problem 4.2 are affine. For (4.12), exploit that $f(\cdot, \cdot, \delta)$ is convex:

$$\hat{V}_\delta(\theta) = f\Big(v_j, \sum_{j=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_j x_j^*, \delta\Big) \leq \sum_{j=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_j f(v_j, x_j^*, \delta) = \sum_{j=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_j V_\delta^*(v_j) = \bar{V}_\delta(\theta). \blacksquare$$

□

As a result of (4.12), Problem 4.8 continues to be infeasible if $\bar{V}_\delta(\theta)$ is substituted by $\hat{V}_\delta(\theta)$. Therefore $\hat{x}$ is ϵ-suboptimal, in other words

$$\hat{V}_\delta(\theta) - V^*(\theta) < \max\{\epsilon_{\mathrm{a}}, \epsilon_{\mathrm{r}} V^*(\theta)\}. \tag{4.13}$$

Hence, given $\theta \in \mathcal{R} \subseteq \Theta$, $\hat{x}$ in Theorem 4 gives an explicit solution and is obtained by querying the partition via:

$$\hat{x} = \sum_{j=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_j x_j^* \text{ where } \theta = \sum_{j=1}^{|\mathcal{V}(\mathcal{R})|} \alpha_j v_j, \ v_j \in \mathcal{V}(\mathcal{R}). \tag{4.14}$$

To summarize, the full precomputation algorithm proceeds following Figure 4.2. First, the

data structure $\mathcal{R}$ is computed via Algorithm 3, which allows to evaluate f_δ. If this succeeds, next the data structure $\mathcal{R}_\epsilon$ is computed via Algorithm 4, which allows to evaluate f_δ^ϵ. If this succeeds, an explicit on-board implementation is possible by storing $\mathcal{R}_\epsilon$ in static memory and using (4.14) to obtain the optimal control input for the current state in polynomial time (in other words, in real-time). The next two sections analyze the convergence properties of Algorithm 3 and Algorithm 4, and the asymptotic complexities of metrics such as leaf count and storage memory size for the partitions $\mathcal{R}$ and $\mathcal{R}_\epsilon$. If the reader chooses to skip these sections, the key take-away points are:

1. **Guaranteed convergence**: if Problem 4.1 is well-defined, in other words has positive *overlap* (see Definitions 8 and 9), Algorithms 3 and 4 both converge;

2. **Polynomial time complexity**: the evaluation complexity of (4.14) is polynomial time, in other words it is amenable to real-time performance even for high-dimensional systems.

4.6 Precomputation Algorithm Convergence

The first algorithm to execute according to the flowchart in Figure 4.2 is Algorithm 3. Thus, convergence of the entire precomputation procedure is conditioned on the convergence of Algorithm 3. We therefore discuss the convergence of this algorithm first. Without loss of generality, we restrict the discussion to Δ, the set of feasible commutations in Θ:

$$\Delta \triangleq \{\delta \in \mathbb{I}^m : \Theta_\delta^* \cap \Theta \neq \emptyset\}. \tag{4.15}$$

The main convergence result for this phase of the precomputation procedure is Theorem 5, which guarantees convergence of Algorithm 3 under Assumption 1. We define a fundamental quantity for both the convergence and the complexity of Algorithm 3: the "feasibility overlap" between fixed-commutation feasible parameter sets.

Definition 8. The **feasibility overlap** is the largest value $\kappa \in \mathbb{R}_+$ such that for each $\theta \in \Theta$, $\exists \delta \in \Delta$ which is feasible in $(\kappa\mathbb{B} + \theta) \setminus (\Theta^* \cap \Theta)^c$.

Assumption 1. The feasibility overlap is positive, in other words $\kappa > 0$.

The feasibility overlap depends on Problem 4.1 and the choice of Θ. Assumption 1 implies that a non-zero overlap between fixed-commutation feasible parameter sets exists everywhere in the neighborhood of $\partial\Theta^*_\delta \ \forall \delta \in \Delta$. This "fuzzy" commutation transition property is instrumental for the convergence proof in Theorem 5.

Theorem 5. *If Assumption 1 holds then Algorithm 3 either converges or fails in a finite number of iterations.*

Proof. Let $\mathcal{R}_k$ be the leaf chosen at the k-th call of Algorithm 3 L5. Whenever $\mathcal{R}_k$ is not closed, it can be shown that the partition along its longest edge on Algorithm 3 L11-14 halves the volume, therefore $\lim_{k\to\infty} \text{vol}(\mathcal{R}_k) = 0$. Since the longest edge length is also halved, $\exists k$ large enough such that $\mathcal{R}_k \subseteq (\kappa\mathbb{B} + c_{\mathcal{R}_k})$ where $c_{\mathcal{R}_k} = |\mathcal{V}(\mathcal{R})|^{-1} \sum_{i=1}^{\mathcal{V}(\mathcal{R})} v_i$ is the center of gravity of $\mathcal{R}_k$. Two possibilities exist: 1) $\mathcal{R}_k \subseteq (\kappa\mathbb{B} + c_{\mathcal{R}_k}) \setminus (\Theta^* \cap \Theta)^c$ or 2) $\mathcal{R}_k \cap (\Theta^*)^c \neq \emptyset$. In the first case, the $\hat{\delta}$ picked on Algorithm 3 L9 is then the one feasible $\forall \theta \in \mathcal{V}(\mathcal{R}_k)$. Leaf $\mathcal{R}_k$ is therefore closed on Algorithm 3 L16. By this logic, for a large enough (but finite) k all regions that do not intersect the infeasible parameter set $(\Theta^*)^c$ get closed. If the second case does not occur, the algorithm terminates.

In the second case, recall that $\lim_{k\to\infty} \text{vol}(\mathcal{R}_k) = 0$. Since $\mathcal{R}_k \cap (\Theta^*)^c \neq 0$, in a finite number of iterations $c_{\mathcal{R}_k} \notin \Theta^*$ so Algorithm 3 L6 will evaluate to true and the algorithm will fail on Algorithm 3 L7. □

Corollary 2. *If Assumption 1 does not hold and* $\Theta \subseteq \Theta^*$ *then Algorithm 3 does not converge.*

Proof. If Assumption 1 does not hold then there exists a region $\Theta' \subseteq \Theta$ such that $\forall \theta \in \Theta'$, $(\kappa\mathbb{B} + \theta) \setminus (\Theta^* \cap \Theta)^c \subseteq \Theta^*_\delta$ for some $\delta \in \Delta \Leftrightarrow \kappa = 0$. This, however, implies that the only simplex that would validate Lemma 2 is one with coincident vertices, in other words a point. Since this occurs at iteration $k = \infty$, Algorithm 3 does not converge. □

Theorem 5 and Corollary 2 suggest that $\kappa > 0$ is not only necessary and sufficient[2] for convergence but that κ also drives the convergence rate. A small κ implies a high iteration count k until Assumption 1 guarantees leaf closure. We call a MICP with large κ "well-conditioned" and Algorithm 3 will converge more quickly with a rate that is derived in Corollary 3 of Section 4.7.

With Theorem 5 stating the convergence property of Algorithm 3, we now move on to proving convergence of the next precomputation phase: Algorithm 4. The main convergence result here is Theorem 6, which guarantees convergence of Algorithm 4 under Assumption 2. Similarly to Definition 8, we define a fundamental quantity for both the convergence and the complexity of Algorithm 4: the "suboptimality overlap" between regions where particular δ values are ϵ-suboptimal.

Definition 9. The *suboptimality overlap* is the largest $\gamma \in \mathbb{R}_+$ such that for each $\theta \in \Theta$, $\exists \delta \in \Delta$ which is ϵ-suboptimal in $(\gamma\mathbb{B} + \theta) \setminus \Theta^{\mathrm{c}}$.

Assumption 2. The suboptimality overlap is positive, in other words $\gamma > 0$.

The suboptimality overlap γ is this time between sets where a given commutation is ϵ-suboptimal. Its value is a non-trivial property of Problem 4.1 and increases for larger values of ϵ_{a} and ϵ_{r} in (4.3). Because Algorithm 4 requires Algorithm 3 to converge, we know that $\gamma \geq 0$ exists. Figure 4.4 illustrates just three overlap possibilities. For convergence, Algorithm 4 should not "oscillate" between δ choices for the same $\mathcal{R}$. This is guaranteed by the following lemma.

Lemma 3. *Let $(\mathcal{R}, \delta)$ be the node selected on L3 at some iteration of Algorithm 4. If δ is replaced with δ^* on Algorithm 4 L9, the node $(\mathcal{R}, \delta)$ will not reappear in a future iteration.*

Proof. We begin by showing that

$$\min_{\theta \in \mathcal{R}} V^*_{\delta^*}(\theta) < \min_{\theta \in \mathcal{R}} V^*_{\delta}(\theta). \tag{4.16}$$

[2]Provided $\Theta \subseteq \Theta^*$.

Since Problem 4.9 is feasible, $\exists \theta^* \in \mathcal{R}$ such that

$$V^*_{\delta^*}(\theta^*) \leq \bar{V}_\delta(\theta^*) - \max\{\epsilon_\mathrm{a}, \epsilon_\mathrm{r} V^*_{\delta^*}(\theta^*)\}. \tag{4.17}$$

Since (4.10) holds, we have:

$$\begin{aligned} \bar{V}_\delta(\theta^*) &\leq \max_{\theta \in \mathcal{R}} V^*_\delta(\theta) \\ &< \min_{\theta \in \mathcal{R}} V^*_\delta(\theta) + \max\{\epsilon_\mathrm{a}, \epsilon_\mathrm{r} V^*_{\delta^*}(\theta^*)\}. \end{aligned} \tag{4.18}$$

Substituting (4.18) into (4.17) shows that (4.16) holds. If $(\mathcal{R}, \delta)$ reappears in a future iteration, then it must be that all of the preceding iterations finished on Algorithm 4 L9. Consider a future iteration where the node is $(\mathcal{R}, \tilde{\delta})$. Recursively applying (4.16), we have:

$$\min_{\theta \in \mathcal{R}} V^*_{\tilde{\delta}}(\theta) < \cdots < \min_{\theta \in \mathcal{R}} V^*_{\delta^*}(\theta) < \min_{\theta \in \mathcal{R}} V^*_\delta(\theta). \tag{4.19}$$

By contradiction, suppose that δ is chosen on Algorithm 4 L7. This means that $\exists \tilde{\theta} \in \mathcal{R}$ such that

$$V^*_\delta(\tilde{\theta}) \leq \bar{V}_{\tilde{\delta}}(\tilde{\theta}) - \max\{\epsilon_\mathrm{a}, \epsilon_\mathrm{r} V^*_\delta(\tilde{\theta})\}. \tag{4.20}$$

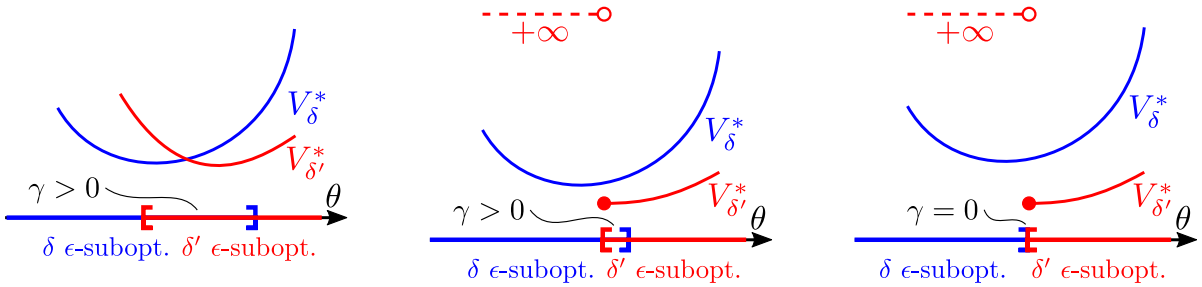
In the same way that we obtained (4.18), we have:

$$\bar{V}_{\tilde{\delta}}(\tilde{\theta}) < \min_{\theta \in \mathcal{R}} V^*_{\tilde{\delta}}(\theta) + \max\{\epsilon_\mathrm{a}, \epsilon_\mathrm{r} V^*_\delta(\tilde{\theta})\}. \tag{4.21}$$

Using (4.21) in (4.20), we have

$$V^*_\delta(\tilde{\theta}) < \min_{\theta \in \mathcal{R}} V^*_{\tilde{\delta}}(\theta) \;\Rightarrow\; \min_{\theta \in \mathcal{R}} V^*_\delta(\theta) < \min_{\theta \in \mathcal{R}} V^*_{\tilde{\delta}}(\theta),$$

which contradicts (4.19), hence δ cannot be more optimal than $\tilde{\delta}$ and thus cannot be re-associated with $\mathcal{R}$. □



(a) Positive overlap from continuity.

(b) Positive overlap from small discontinuity.

(c) Zero overlap.

Figure 4.4: Illustration of the "suboptimality overlap" in Definition 9. In (a) the overlap is positive thanks to local continuity and in (b) it is positive because the downward jump from V^*_δ to $V^*_{\delta'}$ is not too high. In (c) the jump is too high, causing zero overlap.

Theorem 6. *Algorithm 4 terminates if and only if Assumption 2 holds.*

Proof. Suppose that Assumption 2 holds. The algorithm terminates when all nodes become leaves. An iteration of Algorithm 4 can exit on L5, L9 or L14. Since exiting on L5 terminates a branch, it is necessary and sufficient to show that only a finite number of iterations can exit on L9 or L14. Since V^*_δ is convex, it is continuous and therefore (4.10) holds for a small enough $\mathcal{R}$ but with a non-empty interior [34, 134]. If an iteration exits on L14, the volume of $\mathcal{R}$ is halved and its size is reduced, so after a finite number of iterations $\mathcal{R}$ will be small enough such that (4.10) holds and $\mathcal{R} \subset \gamma\mathbb{B} + \theta$ for some $\theta \in \Theta$. Once this occurs, by Definition 9 $\exists \delta^* \in \Delta$ such that Problem 4.9 is feasible. As a result, for any $\delta \in \Delta$ it will take a finite number of iterations until all iterations persistently exit on L9. However, by Lemma 3 and since $|\Delta|$ is finite, this can only occur a finite number of times. Thus, after a finite number of iterations there will remain only one possible choice of δ and the iteration will exit on L5, so the algorithm terminates. If Assumption 2 does not hold, it will take infinite iterations until $\mathcal{R} \subset \gamma\mathbb{B} + \theta$, so the algorithm does not terminate. □

Theorem 6 indicates that the partition complexity is driven by γ and the required "smallness" of $\mathcal{R}$ such that (4.10) holds. We formally define the latter quantity below.

Definition 10. The (conservative) variability is the largest $\nu > 0$ such that (4.10) holds for any $\delta \in \Delta$, any $\mathcal{R} \subset \nu\mathbb{B} + \theta$, any $\theta \in \Theta$ and any $\theta^* \in \mathcal{R}$.

Combining Definitions 9 and 10, we can state an overall *condition number*:

$$\psi \triangleq \min\{\gamma, \nu\}^{-1}, \tag{4.22}$$

which is positively correlated to how much $\mathcal{R}$ must be subdivided until Theorem 6 assures convergence. We call Problem 4.1 with small ψ "well-conditioned" and Algorithm 4 will converge faster. Note that larger ϵ_a and ϵ_r decrease ψ.

In summary, this section introduced two fundamental quantities, the feasible and suboptimal overlaps (Definitions 8 and 9), which are used in Theorems 5 and 6 to guarantee the convergence of Algorithms 3 and 4 respectively when the corresponding overlap quantities are positive. However, knowing that an algorithm converges is in general not enough, as it may take arbitrarily (albeit finitely) many iterations to do so. In the next section, we thus provide theoretical order of magnitude guarantees (in other words, "computational complexity" [135]) on how the iteration count grows with problem size.

4.7 Precomputation Algorithm Complexity

We begin by analyzing the computational complexity of Algorithm 3, as it is the first algorithm to be executed in the precomputation procedure (see Figure 4.2). Complexity results are given both in terms of the partition cell count of the $\mathcal{R}$ data structure as well as the on-line evaluation complexity.

Theorem 7. *The maximum tree depth τ of Algorithm 3 is $\mathcal{O}(p^2 \log(\kappa^{-1}))$.*

Proof. Algorithm 3 reduces search space volume by halving the longest edge length on Algorithm 3 L11-14. Suppose that l_0 is the longest edge length of a simplex $\mathcal{R} \subset \mathbb{R}^p$, then its length is $l_k \triangleq l_0/2^k$ after k divisions. We wish to determine the number of divisions necessary until $\mathcal{R} \subseteq \kappa\mathbb{B} + c^\mathcal{R}$ and gets closed by Theorem 5. This approximately requires

$l_k \leq \kappa \Rightarrow k \geq \log_2(l_0/\kappa)$. Since $\mathcal{R}$ has $p(p+1)/2$ edges then an approximate number of required subdivisions, in other words the depth of the partition tree, is given by:

$$\tau = \left\lceil \frac{p(p+1)\log_2(l_0/\kappa)}{2} \right\rceil, \tag{4.23}$$

which yields $\tau = \mathcal{O}(p^2 \log(\kappa^{-1}))$. □

Corollary 3. *The maximum tree leaf count η of Algorithm 3 is $\mathcal{O}(2^{p^2 \log(\kappa^{-1})})$.*

Proof. In the worst case, Algorithm 3 generates a perfect binary tree of depth $\tau = \mathcal{O}(p^2 \log(\kappa^{-1}))$ according to Theorem 7. Note that we neglected the first layer where the node count depends on $|\mathcal{S}|$ from Algorithm 3 L2. Such a tree contains $\eta = 2^\tau = \mathcal{O}(2^{p^2 \log(\kappa^{-1})})$ leaves. □

Corollary 3 tells us that the leaf count is exponential in the parameter dimension and polynomial in the overlap. However, if we assume that the algorithm terminates with a given finite leaf count then the following lemma states that a linearly proportional number of problems will have had to be solved (known as "output complexity" [121, Section 4.1.1]).

Lemma 4. *The iteration count ν of Algorithm 3 is $\mathcal{O}(\eta)$.*

Proof. A perfect binary tree with η leaves has $2\eta - 1$ nodes, thus at most $\nu = \mathcal{O}(\eta)$ iterations will have occured. □

We have analyzed the tree complexity alone, with disregard for the complexity of the optimization problems that need to be solved at each iteration of Algorithm 3. Unlike [121] where convex nonlinear programs (NLPs) need to be solved at each iteration (due to their MPC law being non-hybrid), we must solve MICPs whose solution time is $\mathcal{O}(2^m)$ in the worst case. However, the basic assumption is that Problem 4.1 *is* solvable in the first place, and that the basic desire is to simply offload on-line computation to an off-line solution. Therefore, we do not consider solving Problem 4.1 for a given $\theta \in \Theta$ to be an issue in practice.

Theorem 8. *The on-line evaluation complexity of f_δ is $\mathcal{O}(p^4)$.*

Proof. Algorithm 3 outputs a tree with η_0 nodes in the first level followed by a binary tree thereafter, where $\eta_0 = |\mathcal{S}|$ from Algorithm 3 L2. Checking if θ is contained in a given simplex can be done via a matrix-vector product in vertex representation, which is $\mathcal{O}(p^2)$. Given a tree depth τ, there are $\eta_0 + \tau - 2$ containment checks to perform. Since $\tau = \mathcal{O}(p^2)$ by Theorem 7, the overall evaluation complexity of f_δ is $\mathcal{O}(p^4)$. □

In conclusion, Theorem 8 guarantees that the evaluation complexity of f_δ is polynomial in the parameter size. Thus, the evaluation of (4.4) is expected to be amenable to real-time implementation even for high-dimensional systems (in other words, control systems with a large state vector for MPC applications). In this manner, the computed feasible δ may be used to warm-start a mixed-integer solver if a globally optimal solution is desired and the time and compute resources are available[3]. Otherwise, we now continue by analyzing the computational complexity of evaluating f_δ^ϵ, which will give us guarantees on real-time performance of semi-explicit or explicit MPC implementations. The main result here is to show that evaluating (4.14) has polynomial complexity (see Theorem 9). We assume that Θ is a simplex, so the partition $\mathcal{R}_\epsilon$ is a binary tree.

Lemma 5. *The depth τ of the tree output by Algorithm 4 is $\mathcal{O}(p^2 \log(\psi))$.*

Proof. In the worst case, Algorithm 4 has to reduce the size of $\mathcal{R}$ until $\mathcal{R} \subset \psi\mathbb{B} + \theta$ for some $\theta \in \Theta$. Once this occurs, Theorem 6 assures that a future iteration will close the corresponding branch without further subdivision. It was shown in Theorem 7 that reducing $\mathcal{R}$ until $\mathcal{R} \subset \psi^{-1}\mathbb{B} + \theta$ takes $\tau = \mathcal{O}(p^2 \log(\psi))$ subdivisions. □

Theorem 9. *The evaluation complexity of (4.14) is $\mathcal{O}(p^4)$.*

Proof. As explained in Section 4.9.2, checking if $\theta \in \mathcal{R}$ can be done via a matrix-vector product, which is $\mathcal{O}(p^2)$. Since there are τ such checks to perform and since $\tau = \mathcal{O}(p^2)$ due to Lemma 5, it takes $\mathcal{O}(p^4)$ operations to find the $\mathcal{R}$ which contains θ. It subsequently

[3]This point is important. It emphasizes that the output of Algorithm 3 is already useful in itself, namely in warm starting a mixed-integer solver to achieve faster solution times or an anytime implementation.

takes $\mathcal{O}(np)$ operations to compute $\hat{x}$ via (4.14), hence the overall evaluation complexity is $\mathcal{O}(p^4)$. □

Theorem 9 guarantees polynomial time evaluation complexity of for the explicit implementation (4.14). Thus, even for problems with a high-dimensional state vector, we expect the on-board explicit implementation to be amenable to real-time performance[4]. This stands in contrast to implementing Problem 4.1 directly with a mixed-integer solver, which has an exponential runtime $\mathcal{O}(2^m)$.

We shall corroborate the theoretic convergence and complexity results of Sections 4.6 and 4.7 on several practical examples, including a more complicated (mixed-integer) formulation of satellite robust position control using RCS, a milder version of which was solved at the end of Chapter 3.

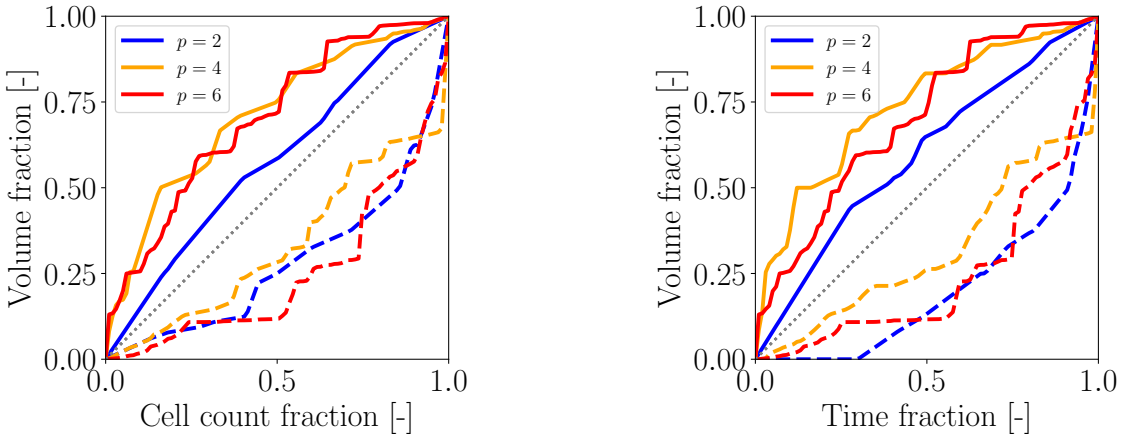
4.8 Numerical Examples

This section presents two numerical tests of Algorithms 3 and 4:

1. **Control of an uncertain multiple degree of freedom oscillator**: this tests only the capability of Algorithm 3 to generate the feasible partition data structure $\mathcal{R}$ for computing f_δ. The output can be used in an aforementioned warm start scheme. The multiple-DoF oscillators are randomly generated, allowing us to test the algorithm for a whole suite of systems with growing state dimension;

2. **Position control of a satellite with RCS minimum impulse-bit constraint**: this test is primarily for Algorithm 4, however it implicitly also tests Algorithm 3 as it will have had to converge prior to calling Algorithm 4 (as illustrated in Figure 4.2).

[4]Note that this statement relates strictly to on-board implementation, that is once the data structure $\mathcal{R}_\epsilon$ has been computed. Since the maximum tree leaf count is exponential in the parameter size (see Corollary 3), computing $\mathcal{R}_\epsilon$ itself is an exponential hard task in the parameter dimension. This downside is characteristic of all explicit MPC methods [46].



(a) Volume versus cell count.

(b) Volume versus runtime.

Figure 4.5: Normalized convergence plots. Cumulative closed volume, cumulative closed leaf count and runtime are normalized by their final respective values. The solid/dashed lines show the envelope max/min while the dotted reference line shows linear convergence.

The aim of these tests, and the focus of the following discussion, is to analyze the quality of the theoretical bounds established in Section 4.7. It will be seen that the theoretical bounds predict the algorithm behaviour well. In particular, Section 4.8.2 the behaviour as a function of an estimator for the condition number ψ in (4.22).

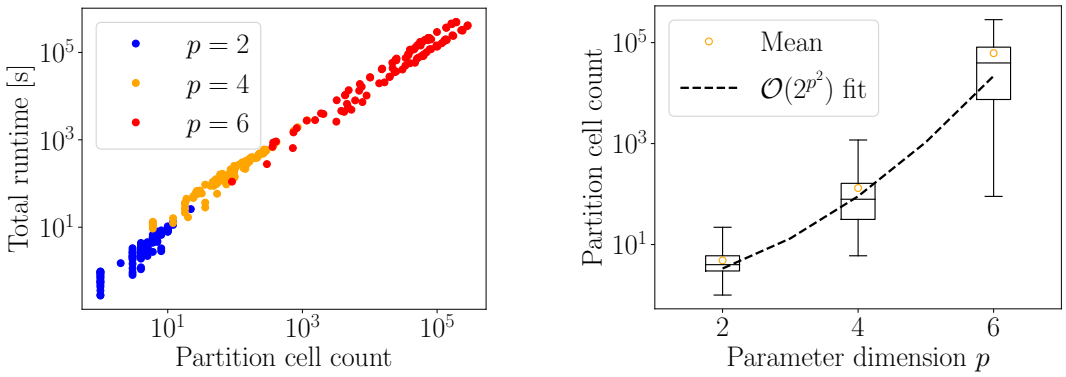
4.8.1 Performance for Multiple-DoF Oscillator Control

Appendix A constructs a model predictive control law (Problem A.7) for a randomly generated instance of a multiple-DoF oscillator system. This allows us to perform a Monte Carlo analysis of Algorithm 3 on a whole class of systems, rather than on one or a small number of specific examples. Thus, Algorithm 3 is applied to 100 randomly generated instances of Problem A.7 with state dimensions of 2, 4 and 6. This yields tested parameter dimensions of $p = 2, 4, 6$, and commutation dimensions $m = 9, 15, 21$. This demonstrates that Algorithm 3 can scale up to at least $p = 6$ and $m = 21$, with higher dimensions likely possible via massive parallelization (discussed for the next example in Section 4.8.2).

Figure 7.19 shows convergence plots using the volume fraction of Θ made up by the closed leafs as the metric. Since at first none and in the end all leaves are closed, this metric goes from 0 at the start to 1 at the end of Algorithm 3 and is easily evaluated since the volume of a simplex is well known [136]. The algorithm has a favorable convergence characteristic in that no convergence curve in our tests deviated significantly from a linear rate. The practical significance of this is that the algorithm progresses steadily towards filling up the entire volume of Θ rather than being very slow at the beginning and very fast at the end or vice versa. Both cases would be poor for the user to supervise since, especially for $p > 3$, it becomes difficult to diagnose the reason for slow convergence.

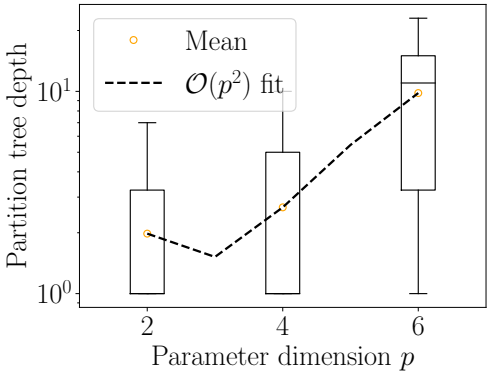
Figure 4.6a shows the wall-clock total runtime corresponding to each run, which appears to increase linearly and with unity slope as a function of the final partition cell count. The linear trend agrees with the linear output complexity of Lemma 4 while the unity slope may be interpreted as that, regardless of p, it takes the current implementation on average 1 second to add 1 closed leaf to the partition tree. Note that this measurement includes the time taken to traverse potentially many layers of the tree until adding a closed leaf (up to about 20 layers according to Figure 4.6c).

Figures 4.6b, 4.6c, and 4.6d show statistics on the final tree leaf count and depth along with fitted complexity curves resulting from Section 4.7. Figure 4.6b shows clearly that the partition tree leaf count increases exponentially with p as stipulated by Corollary 3. Interestingly, we note from Figure 4.6c that for some instances of Problem A.7 the tree depth does not go beyond the first layer. In other words, sometimes the Delaunay triangulation of Θ on Algorithm 3 L2 of Algorithm 3 suffices. Note that because the complexities in Theorem 7 and Corollary 3 depend also on the overlap κ, which currently cannot be computed a priori, the regressions in Figure 4.6b and Figure 4.6c carry an omitted variable bias. However, Corollary 3 allows to compute normalized values for κ by assuming that the deviations in Figure 4.6b of the actual cell count from the fitted one are due to κ alone. This effectively captures the normalized variation required from κ in order to explain the deviation of observed results from the regressed theoretical values. This is shown in Figure 4.6d where

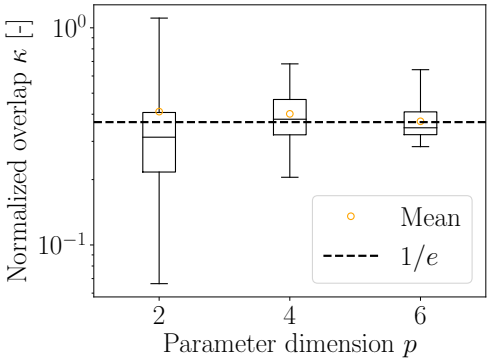


(a) Total runtime statistics.

(b) Cell count statistics.



(c) Tree depth statistics.



(d) Normalized overlap statistics.

Figure 4.6: Final partition statistics. Whiskers show the full range.

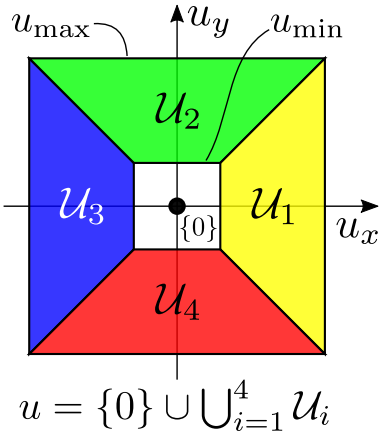


Figure 4.7: A non-convex input lower-bound constraint for satellite MPC can be modeled as the union of convex sets. This induces a mixed-integer program.

$\kappa = 1/e$ if the match between the fitted and predicted cell counts is perfect. As expected, the effect of κ diminishes for higher p where the exponential complexity in p dominates over the polynomial complexity in κ.

4.8.2 Performance for Satellite Robust Position Control

This section presents two examples to corroborate the effectiveness of Algorithm 4 and the conclusions of Sections 4.6 and 4.7. We consider robust hybrid MPC of a satellite's out-of-plane (`cwh_z`) and in-plane (`cwh_xy`) position, as an extension of the MPC problem solved in Section 3.6. Explicit MPC is relevant for satellite control because the conservative design of space systems typically prohibits the use of on-line optimization. We split the Clohessy-Wiltshire equations into the out-of-plane and in-plane components (whose motions are decoupled):

$$\texttt{cwh\_xy:} \quad \begin{cases} \ddot{x} = 3\omega_0^2 x + 2\omega_o \dot{y} + u_x + w_x, \\ \ddot{y} = -2\omega_o \dot{x} + u_y + w_u, \end{cases} \tag{4.24}$$

$$\texttt{cwh\_z:} \quad \ddot{z} = -\omega_o^2 z + u_z + w_z, \tag{4.25}$$

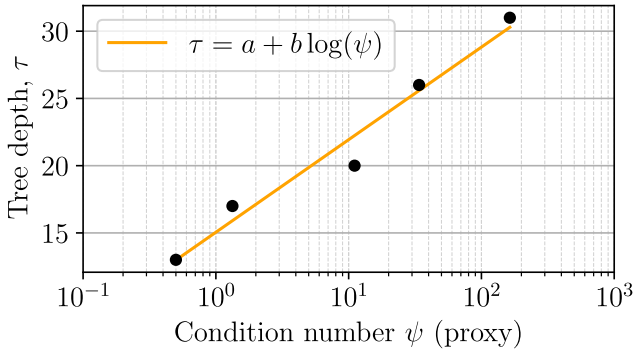


Figure 4.8: The tree depth for `cwh_z` is approximately logarithmic in the condition number ψ, as predicted by Lemma 5.

where ω_o is the orbital rate in rad/s and w are disturbance terms. The full explanation of this model was already provided in Section 3.6 of Chapter 3, so will not be repeated here. For `cwh_xy`, the state is $\begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^4$ and the input is $\begin{bmatrix} u_x & u_y \end{bmatrix}^\mathsf{T} \in \mathbb{R}^2$. For `cwh_z`, the state is $\begin{bmatrix} z & \dot{z} \end{bmatrix} \in \mathbb{R}^2$ and the input is $u_z \in \mathbb{R}$. Assuming an impulsive input, the system is discretized at a $T_\mathrm{s} = 100$ s thruster firing period. On top of the example problem in Section 3.6, we add a lower-bound constraint $\|u\|_\infty \geq u_\mathrm{min}$ on the thrust magnitude, which arises from the thruster minimum impulse-bit. The constraint is non-convex but can be modeled as the union of convex sets, as illustrated in Figure 4.7, yielding a mixed-integer second order cone program. We use a prediction horizon $N = 4$ and, letting $\mathcal{X} \triangleq [-10, 10]$ cm $\times$ $[-1, 1]$ mm/s, choose $\Theta = \mathcal{X}$ for `cwh_z` and $\Theta = \mathcal{X} \times \mathcal{X}$ for `cwh_xy`. It was shown in Section 3.6 that this Θ choice is a robust controlled invariant set, hence the partition will be sufficient for controlling the satellite.

Our implementation is available online (see Footnote 1). The code is written in Python 3.7.2, using CVXPY 1.0.21 [137] and MOSEK 9.0.87 [132]. Recognizing that Algorithm 4 L4-14 can run in parallel across tree branches, we used MPICH 3.2 in CentOS 7 on a cluster of up to 420 2.4 GHz Intel E5-2680 CPU cores with 20 GB of RAM per compute node (28 cores). The code can also run locally.

Table 4.1 summarizes the output of Algorithm 4 using a sequence of increasingly tight

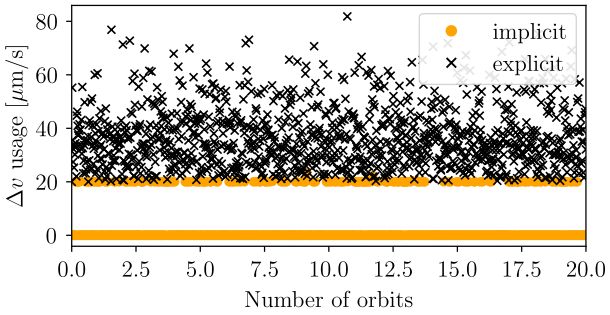
Example	s_a	ϵ_r	τ	λ	T_wall [hr]	T_cpu [hr]	M [MB]
`cwh_z`	0.50	2.00	13	101	0.01	0.09	< 0.01
`cwh_z`	0.25	1.00	17	978	0.06	0.96	< 0.01
`cwh_z`	0.10	0.10	20	13500	0.31	7.72	11
`cwh_z`	0.03	0.05	26	235231	1.91	154.19	202
`cwh_z`	0.01	0.01	31	3322941	6.37	2516.98	2916
`cwh_xy`	0.50	2.00	32	30448	0.57	53.44	36
`cwh_xy`	0.25	1.00	49	884323	3.38	1297.35	1069

Table 4.1: Numerical results for several ϵ-suboptimality settings; τ is the tree depth, λ is the leaf count, T_wall is the Algorithm 4 runtime, T_cpu is the computation time summed across parallel processors, and M is the tree file size.

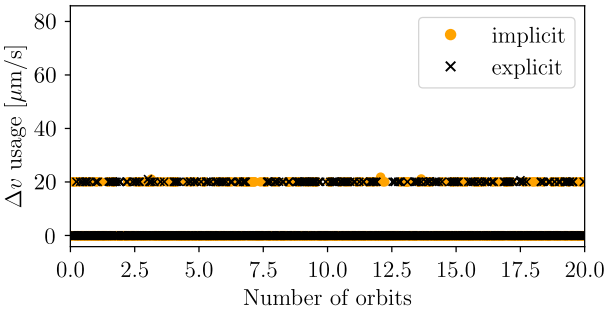
ϵ-suboptimality settings. We compute ϵ_a as the largest cost among the vertices of a shrunk Θ, i.e. $\epsilon_\mathrm{a} = \max\{V^*(\theta) \mid \theta \in \mathcal{V}(s_\mathrm{a}\Theta)\}$. The smaller s_a is, the more dense the partition will be in a neighborhood of the origin. Figure 4.11 shows the progress of Algorithm 4 for the last row of Table 4.1. We can see that the progress is mostly linear. Because multiple cores do not participate in evaluating Algorithm 4 L4-14 for the same node $(\mathcal{R}, \delta)$, progress slows down near the end when only a few nodes are left.

Figure 4.8 confirms that $\tau = \mathcal{O}(\log(\psi))$ using the proxy $\psi = (\epsilon_\mathrm{a}/\bar{\epsilon}_\mathrm{a} + \epsilon_\mathrm{r}/\bar{\epsilon}_\mathrm{r})^{-1}$, where $\bar{\epsilon}_\mathrm{a}$ and $\bar{\epsilon}_\mathrm{r}$ are the largest of the tested values. The leaf count is exponential in ψ. It follows that T_cpu and M are also exponential in ψ. Note that the exponential increase in T_cpu can be offset by an exponential increase in parallel core count, until a certain limit. Thus, our method allows for reasonable T_wall runtimes.

Figure 4.10a show statistics for the on-line control input computation time. For implict MPC, this is the mixed-integer solver time. For explicit MPC, it is the time to evaluate (4.14), which involves querying the partition tree. Statistics are computed by uniformly randomly sampling 1000 values of $\theta \in \Theta$. As expected from Theorem 9, explicit MPC can be up to three orders of magnitude faster. Importantly, explicit MPC provides a real-time guarantee given by the time that it takes to traverse the tree to the deepest leaf. The implicit approach



(a) Input 2-norm history for `cwh_z` with $s_a = 0.5$ and $\epsilon_r = 2$.



(b) Input 2-norm history for `cwh_z` with $s_a = 0.01$ and $\epsilon_r = 0.01$.

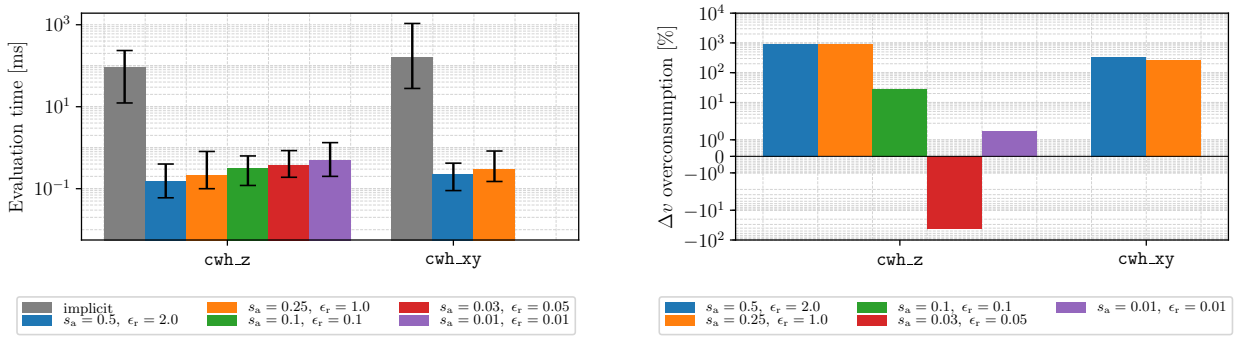
Figure 4.9: Comparison of control input histories for a coarse and a refined ϵ-suboptimal partition. By reducing ϵ_a and ϵ_r, explicit MPC approaches the behavior of implicit MPC.

may be arbitrarily slower for some values of θ, subject to the success of the mixed-integer solver's heuristics.

Figure 4.10b quantifies the fuel consumption suboptimality with respect to implicit MPC. The data is collected based on a 20 orbit simulation where the satellite is initialized at the origin, i.e. with zero control error. As expected, partitions with a tighter ϵ-suboptimality setting perform better. Importantly, explicit MPC can *outperform* implicit MPC since the control scheme is finite horizon while fuel is an integrated quantity. This is the case for `cwh_z` with $s_a = 0.03$ and $\epsilon_r = 0.05$, which achieves ≈ 40 % fuel reduction. The source of this reduced fuel consumption is clearly visible in Figure 4.9, where one can see that with a tighter ϵ-suboptimality setting, explicit MPC better reproduces the optimal behavior of implicit MPC.

4.9 Comments on Implementation

The previous section discussed several numerical results for the implementation of Algorithms 3 and 4. In this section, we discuss some best practices and give general advice for how to implement the precomputation procedure illustrated in Figure 4.2.



(a) MPC on-line evaluation time. Bars show the mean while error bars shown the minimum and maximum values.

(b) Overconsumption of fuel with respect to implicit MPC due to ϵ-suboptimality. Implicit MPC uses ≈ 4 mm/s over 20 orbits.

Figure 4.10: Comparison of the proposed semi-explicit and explicit implementations to implicit MPC in terms of (a) on-line control input computation time and (b) total fuel consumption over 20 orbits.

4.9.1 Choice of Θ

Throughout this paper we have assumed that Θ is available. We now explain possible methods of obtaining it. First of all, $\Theta \subseteq \Theta^*$ should hold. If it does not, per Theorem 5 Algorithm 3 will report it in a finite number of iterations and a different Θ must be chosen. Assuming the task of Problem 4.1 is to drive θ (e.g. the current state) to the origin, a Θ in a small enough neighborhood of $0 \in \mathbb{R}^p$ should satisfy this property as long as Problem 4.1 is a well-defined controller.

Since f_δ is defined only over Θ, in practice it must be ensured that $\theta \notin \Theta$ is never encountered during runtime. This requires Θ to be a robust controlled invariant set of Problem 4.1. A straightforward method for ensuring this is to include the constraint $\Theta^+ \subseteq \Theta$ in Problem 4.1 where Θ^+ models all possible future values of θ. This is a constraint in some RMPC methods [58, 98], in which case Θ is explicitly known. Section 4.8.1 leverages this fact in Problem A.7. Note that convergence of Algorithm 3 then certifies recursive feasibility of Problem 4.1.

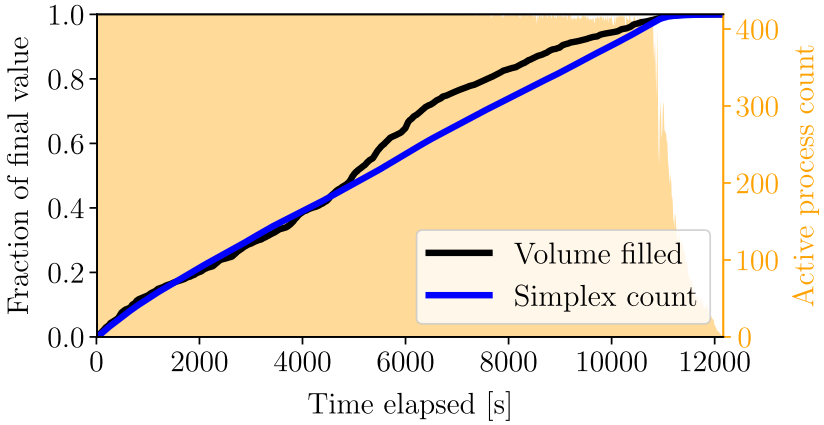


Figure 4.11: Algorithm 4 progress plot. The partition complexity and the volume fraction of Θ comprised by completed tree branches steadily increase. The partitioning becomes serialized near the end.

Finally, note from the proof of Theorem 8 that the practical complexity of f_δ is $\mathcal{O}(\eta_0 p^2)$ since in practice $\eta_0 \gg \tau$. There is therefore a considerable interest to make the Delaunay triangulation of Θ yield a small η_0, e.g. by using a simplex or a small number of simplices to define Θ.

4.9.2 Improving the Storage Memory Requirement

Our implementation (see Footnote 1) is not optimized for storage size, so the M values in Table 4.1 are far larger than necessary. A more efficient storage model is as follows. Given a simplex $\mathcal{R} \subset \mathbb{R}^p$ and its vertices $v_i \in \mathcal{V}(\mathcal{R})$, a parameter $\theta \in \mathcal{R}$ if an only if $\alpha = H_{\mathcal{R}}^{-1}(x - v_1) \geq 0$, $\mathbf{1}^\intercal \alpha \leq 1$, where the i-th column of $H_{\mathcal{R}}$ equals $v_{i+1} - v_1$. Since Θ and hence $\mathcal{R}$ are full-dimensional, $H_{\mathcal{R}}$ is invertible. By leveraging mutual exclusivity of the partition cells, we can thus store a matrix $H \in \mathbb{R}^{p \times p}$ and a vector $v_1 \in \mathbb{R}^p$ for each "left" child node. For each leaf, in order to evaluate (4.14) we store the $p+1$ ϵ-suboptimal decision vectors x_j^* at its vertices. Assuming a perfect binary tree and that Θ is a simplex,

the improved storage size is:

$$M^* \approx \frac{3}{2}\lambda\mu_{\mathrm{f}}p(p+1) + \lambda(p+1)\hat{n}\mu_{\mathrm{f}}, \tag{4.26}$$

where μ_{f} is the floating point size and $\hat{n} \leq n$ is the dimension of the part of the decision vector that is necessary to compute the control input (i.e. the first control input for MPC). For the examples in Table 4.1, M^* is 3 to 10 times less than M. Greater economy is possible by eliminating further redundancy in the stored vertices and optimal decision vectors.

4.10 Future Work

This chapter presented a partitioning algorithm for generating explicit solutions of a very general class of multiparametric mixed-integer convex programs to within a given suboptimality tolerance. Figure 4.2 illustrates the method, which relies on the sequential solution of: 1) Algorithm 3, then 2) Algorithm 4, and finally the application of (4.14) on-board the target system.

From a theoretical standpoint, the chapter showed in Sections 4.6 and 4.7 that the positivity of novel "feasibility overlap" and "suboptimality overlap" metrics is necessary and sufficient for convergence. To the best of our knowledge, this is the first deep theoretical insight into the fundamental driver of convergence rate and partition complexity of suboptimal explicit MPC precomputation algorithms similar to the one in this chapter [121, 123, 126, 127].

In future work it will be interesting to prove the stability of the resulting control law along the lines of [126] and [127], and to see if the selection of ϵ_{a} and ϵ_{r} could be automated to ensure convergence. Furthermore, the method was presented as a way for partitioning some set $\Theta \subseteq \Theta^*$. A possible extension is to partition the entire Θ^*. Since by Lemma 1 Θ^*_δ is convex, it may be inner-approximated with arbitrary precision [115]. Doing so for each δ, one can run the algorithm for each commutation $\delta \in \mathbb{I}^m$ by taking $\Theta = \Theta^*_\delta$. To remove overlapping regions, the intersection of Θ^*_δ with all previously considered fixed-commutation

feasible parameter sets can be removed.

Chapter 5

HISTORY OF LOSSLESS CONVEXIFICATION

Lossless convexification (LCvx) is a modeling technique that solves nonconvex optimal control problems through a convex relaxation. In this method, Pontryagin's maximum principle [39] is used to show that a convex relaxation of a nonconvex problem finds the globally optimal solution to the original problem, hence the name "lossless". To date, the method has been extended as far as relaxing certain classes of nonconvex control constraints, such as an input norm lower bound (see Example 2) and a nonconvex pointing constraint (see Example 3).

The LCvx method has been shown to work for a large class of state-constrained optimal control problems, however a working assumption is that state constraints are convex. Lossless relaxation of nonconvex state constraints remains under active research, and some related results are available [116] (which we will cover in this section).

As the reader goes through this section, it is suggested to keep in mind that the main concerns of lossless convexification are:

- To find a convex lifting of the feasible input set;
- To show that the optimal input of the lifted problem projects back to a feasible input of the non-lifted problem.

Figure 5.1 chronicles the development history of LCvx. The aim of this section is to provide a tutorial overview of the key results, so theoretical proofs are omitted in favor of a more practical and action-oriented description. Ultimately, our aim is for the reader to come away with a clear understanding of how LCvx can be applied to their own problems.

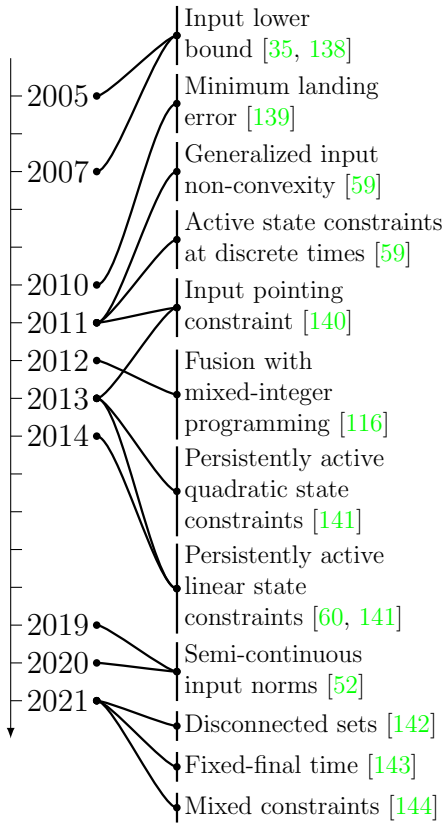


Figure 5.1: Chronology of lossless convexification theory development. Note the progression from state-unconstrained problems to those with progressively more general state constraints and, finally, to problems that contain integer variables.

In the following sections, we begin by introducing LCvx for the input norm lower bound and pointing nonconvexities using a baseline problem with no state constraints. Then, the method is extended to handle affine and quadratic state constraints, followed by general convex state constraints. We then describe how the LCvx method can also handle a class of dynamical systems with nonlinear dynamics. For more general applications, embedding LCvx into nonlinear optimization algorithms is also discussed. At the very end of this section we cover some of the newest LCvx results from the past year, and provide a toy example that gives a first taste of how LCvx can be used in practice.

5.1 No State Constraints

We begin by stating perhaps the simplest optimal control problem for which an LCvx result is available. Its salient features are a distinct absence of state constraints (except for the boundary conditions), and its only source of nonconvexity is a lower-bound on the input given by (5.1c). A detailed description of LCvx for this problem may be found in [35, 59, 138].

$$\min_{u,t_f} \; m(t_f, x(t_f)) + \zeta \int_0^{t_f} \ell(g_1(u(t)))\,\mathrm{d}t \tag{5.1a}$$

$$\text{s.t. } \dot{x}(t) = A(t)x(t) + B(t)u(t) + E(t)w(t), \tag{5.1b}$$

$$\rho_{\min} \leq g_1(u(t)), \;\; g_0(u(t)) \leq \rho_{\max}, \tag{5.1c}$$

$$x(0) = x_0, \;\; b(t_f, x(t_f)) = 0. \tag{5.1d}$$

In Problem 5.1, $t_f > 0$ is the terminal time, $x(\cdot) \in \mathbb{R}^n$ is the state trajectory, $u(\cdot) \in \mathbb{R}^m$ is the input trajectory, $w(\cdot) \in \mathbb{R}^p$ is an exogenous additive disturbance, $m : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ is a convex terminal cost, $\ell : \mathbb{R} \to \mathbb{R}$ is a convex and non-decreasing running cost modifier, $\zeta \in \{0, 1\}$ is a fixed user-chosen parameter to toggle the running cost, $g_0, g_1 : \mathbb{R}^m \to \mathbb{R}_+$ are convex functions, $\rho_{\min} > 0$ and $\rho_{\max} > \rho_{\min}$ are user-chosen bounds, and $b : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^{n_b}$ is an affine terminal constraint function. Note that the dynamics in Problem 5.1 define a linear time varying (LTV) system. For all the LCvx discussion that follows, we will also make the following two assumptions on the problem data.

Assumption 3. If any part of the terminal state is constrained then the Jacobian $\nabla_x b[t_f] \in \mathbb{R}^{n_b \times n}$ is full row rank, i.e., $\operatorname{rank} \nabla_x b[t_f] = n_b$. This implies that the terminal state is not over-constrained.

Assumption 4. The running cost is positive definite, in other words $\ell(z) > 0$ for all $z \neq 0$.

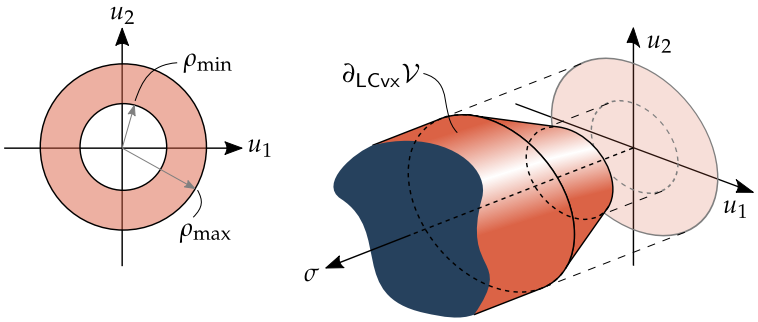


Figure 5.2: Relaxation of the non-convex input constraint set (5.3) to the convex set (5.4) by using a slack input σ. If the optimal input $(u^*, \sigma^*) \in \partial_{\text{LCvx}}\mathcal{V}$ (5.6), then u^* is feasible for the non-convex constraint (5.3).

When faced with a nonconvex problem like Problem 5.1, an engineer has two choices. Either devise a nonlinear optimization algorithm, or solve a simpler problem that is convex. The mantra of LCvx is to take the latter approach by "relaxing" the problem until it is convex. In the case of Problem 5.1, let us propose the following relaxation, which introduces a new variable $\sigma(\cdot) \in \mathbb{R}$ called the *slack input*.

$$\min_{\sigma,u,t_f} \; m(t_f, x(t_f)) + \zeta \int_0^{t_f} \ell(\sigma(t))\,\mathrm{d}t \tag{5.2a}$$

$$\text{s.t. } \dot{x}(t) = A(t)x(t) + B(t)u(t) + E(t)w(t), \tag{5.2b}$$

$$\rho_{\min} \le \sigma(t), \;\; g_0(u(t)) \le \rho_{\max}, \tag{5.2c}$$

$$g_1(u(t)) \le \sigma(t), \tag{5.2d}$$

$$x(0) = x_0, \;\; b(t_f, x(t_f)) = 0. \tag{5.2e}$$

Example 2. A typical rocket engine's thrust is upper bounded by the engine's performance and lower bounded by combustion instability issues for small thrusts. Thus, (5.1c) can be written as

$$\rho_{\min} \le \|u\|_2 \le \rho_{\max}, \tag{5.3}$$

so that $g_0(u) = g_1(u) \triangleq \|u\|_2$. The relaxation (5.2c)-(5.2d) then becomes:

$$\rho_{\min} \leq \sigma, \ \ \|u\|_2 \leq \min(\sigma, \rho_{\max}). \tag{5.4}$$

Figure 5.2 shows how for $u \in \mathbb{R}^2$, going from (5.3) to (5.4) lifts a non-convex annulus to a convex volume $\mathcal{V}$,

$$\mathcal{V} \triangleq \{(u, \sigma) \in \mathbb{R}^3 : \rho_{\min} \leq \sigma, \|u\|_2 \leq \min(\sigma, \rho_{\max})\}. \tag{5.5}$$

The drawback is that inputs that were not feasible for (5.3) become feasible for (5.4). In particular, any u for which $\|u\|_2 < \rho_{\min}$ is now feasible. However, for the special case of $(u, \sigma) \in \partial_{\text{LCvx}}\mathcal{V}$, where:

$$\partial_{\text{LCvx}}\mathcal{V} \triangleq \{(u, \sigma) \in \mathbb{R}^3 : \rho_{\min} \leq \sigma, \|u\|_2 = \min(\sigma, \rho_{\max})\}, \tag{5.6}$$

the input u is feasible. The goal of lossless convexification is to show that this occurs at the global optimum of Problem 5.2, in other words $(u^*(t), \sigma^*(t)) \in \partial_{\text{LCvx}}\mathcal{V}$. Theorem 10 guarantees this.

The relaxation of the nonconvex input constraint (5.1c) to the convex constraints (5.2c)-(5.2d) is illustrated in Example 2 for the case of a rocket engine. Note that if (5.2d) is replaced with equality, then Problem 5.2 is equivalent to Problem 5.1. Indeed, the entire goal of LCvx is to *prove* that (5.2d) holds with equality at the globally optimal solution of Problem 5.2. Because of the clear importance of constraint (5.2d) to the LCvx method, we shall call it the *LCvx equality constraint*. This special constraint will be highlighted in red in all subsequent LCvx optimization problems.

Consider now the following set of conditions, which arise naturally when using the maximum principle to prove lossless convexification. The theoretical details are provided in [59, Theorem 2].

Condition 1. The pair $\{A(\cdot), B(\cdot)\}$ must be totally controllable. This means that any initial state can be transferred to any final state by a bounded control trajectory in any finite time interval $[0, t_f]$ [116, 145]. If the system is time invariant, this is equivalent to $\{A, B\}$ being controllable, and can be verified by checking that the controllability matrix is full rank or, more robustly, via the PBH test [114].

Condition 2. Define the quantities:

$$m_{\text{LCvx}} = \begin{bmatrix} \nabla_x m[t_f] \\ \nabla_t m[t_f] + \zeta \ell(\sigma(t_f)) \end{bmatrix} \in \mathbb{R}^{n+1}, \tag{5.7a}$$

$$B_{\text{LCvx}} = \begin{bmatrix} \nabla_x b[t_f]^{\mathsf{T}} \\ \nabla_t b[t_f]^{\mathsf{T}} \end{bmatrix} \in \mathbb{R}^{(n+1)\times n_b}. \tag{5.7b}$$

The vector m_{LCvx} must be linearly independent from the columns of B_{LCvx}.

We can now state Theorem 10, which is the main lossless convexification result for Problem 5.1. The practical implication of Theorem 10 is that the solution of Problem 5.1 can be found in polynomial time by solving Problem 5.2 instead.

Theorem 10. *The solution of Problem 5.2 is globally optimal for Problem 5.1 if Conditions 1 and 2 hold.*

There is a partial generalization of Theorem 10. Let us restrict Problem 5.1 to the choices $\zeta = 1$ and $g_0 = g_1$, and let us introduce a new pointing-like input constraint (5.8d). The quantities $\hat{n}_u \in \mathbb{R}^m$ and $\hat{n}_g \in \mathbb{R}$ are user-chosen parameters. The new problem takes the following form:

$$\min_{u, t_f} \; m(t_f, x(t_f)) + \int_0^{t_f} \ell(g_0(u(t)))\, \mathrm{d}t \tag{5.8a}$$

$$\text{s.t. } \dot{x}(t) = A(t)x(t) + B(t)u(t) + E(t)w(t), \tag{5.8b}$$

$$\rho_{\min} \leq g_0(u(t)) \leq \rho_{\max}, \tag{5.8c}$$

$$\hat{n}_u^{\mathsf{T}} u(t) \geq \hat{n}_g g_0(u(t)), \tag{5.8d}$$

$$x(0) = x_0,\ b(t_f, x(t_f)) = 0. \tag{5.8e}$$

Using (5.8d) one can, for example, constrain an airborne vehicle's tilt angle. This constraint, however, is nonconvex for $\hat{n}_g < 0$. We take care of this nonconvexity, along with the typical nonconvexity of the lower bound in (5.8c), by solving the following relaxation of the original problem:

$$\min_{\sigma,u,t_f}\ m(t_f, x(t_f)) + \int_0^{t_f} \ell(\sigma(t))\,\mathrm{d}t \tag{5.9a}$$

$$\text{s.t. } \dot{x}(t) = A(t)x(t) + B(t)u(t) + E(t)w(t), \tag{5.9b}$$

$$\rho_{\min} \leq \sigma(t) \leq \rho_{\max}, \tag{5.9c}$$

$$g_0(u(t)) \leq \sigma(t), \tag{5.9d}$$

$$\hat{n}_u^{\mathsf{T}} u(t) \geq \sigma(t)\hat{n}_g, \tag{5.9e}$$

$$x(0) = x_0,\ b(t_f, x(t_f)) = 0. \tag{5.9f}$$

Just like in Problem 5.2, we introduced a slack input $\sigma(\cdot) \in \mathbb{R}$ to strategically remove nonconvexity. Note once again the appearance of the LCvx equality constraint (5.9d). Meanwhile, the relaxation of (5.8d) to (5.9e) corresponds to a halfspace input constraint in the $(u, \sigma) \in \mathbb{R}^{m+1}$ space. A geometric intuition about the relaxation is illustrated in Example 3 for a typical vehicle tilt constraint. Lossless convexification can again be shown under an extra Condition 3, yielding Theorem 11. Theoretical details are provided in [140, 146].

Condition 3. Let $N \in \mathbb{R}^{m\times(m-1)}$ be a matrix whose columns span the nullspace of $\hat{n}_u$ in (5.8d). The pair $\{A(\cdot), B(\cdot)N\}$ must be totally controllable.

Theorem 11. *The solution of Problem 5.9 is globally optimal for Problem 5.8 if Conditions 1, 2, and 3 hold.*

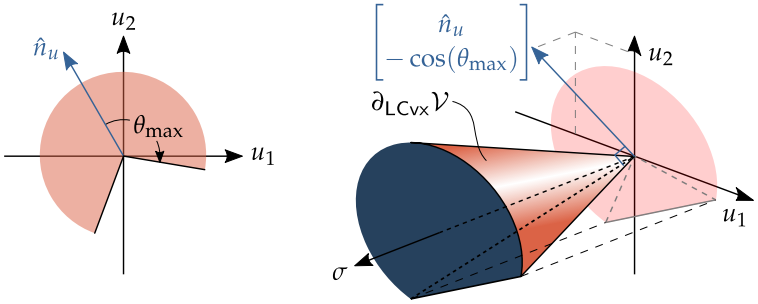


Figure 5.3: Relaxation of the non-convex input set (5.10) to a convex set (5.11) via intersection with a halfspace in the lifted (u, σ) space. If the optimal input $(u^*, \sigma^*) \in \partial_{\text{LCvx}}\mathcal{V}$ then u^* is feasible for the non-convex constraint (5.10).

Example 3. Practical mechanical systems, such as Segways and rockets, may have a constraint on their tilt angle away from an upright orientation. To model such cases, the inequality (5.8d) can be specialized to an input pointing constraint. By choosing $\hat{n}_g = \cos(\theta_{\max})$ and $g_0(u) = \|u\|_2$, the constraint becomes

$$\hat{n}_u^\mathsf{T} u \geq \|u\|_2 \cos(\theta_{\max}), \tag{5.10}$$

which constrains u to be no more than $\theta_{\max}$ radians away from a nominal pointing direction $\hat{n}_u$. The relaxed constraint (5.9e) becomes

$$\hat{n}_u^\mathsf{T} u \geq \sigma \cos(\theta_{\max}), \tag{5.11}$$

which is a halfspace constraint and is thus convex even for $\theta_{\max} > \pi/2$, when (5.10) becomes non-convex.

Figure 5.3 shows how for $u \in \mathbb{R}^2$, relaxing (5.10) to (5.11) lifts an obtuse "pie slice" to a convex halfspace $\mathcal{V}$,

$$\mathcal{V} \triangleq \{(u, \sigma) \in \mathbb{R}^3 : \hat{n}_u^\mathsf{T} u \geq \sigma \cos(\theta_{\max}),\ \|u\|_2 \leq \sigma\}. \tag{5.12}$$

The drawback is that inputs that were not feasible for (5.10) become feasible for (5.11). In particular, low magnitude inputs for which $\hat{n}_u^{\mathsf{T}} u < \|u\|_2 \cos(\theta_{\max})$ become feasible. However, if $(u, \sigma) \in \partial_{\text{LCvx}} \mathcal{V}$,

$$\partial_{\text{LCvx}} \mathcal{V} \triangleq \{(u, \sigma) \in \mathbb{R}^3 : \hat{n}_u^{\mathsf{T}} u \geq \sigma \cos(\theta_{\max}),\ \|u\|_2 = \sigma\}, \tag{5.13}$$

then the input u satisfies the original constraint (5.10). The goal of lossless convexification is to show that this occurs at the global optimum, in other words $(u^*, \sigma^*) \in \partial_{\text{LCvx}} \mathcal{V}$. Theorem 11 guarantees this.

5.2 Affine State Constraints

The logical next step after Theorems 10 and 11 is to ask whether Problem 5.1 can incorporate state constraints. It turns out that this is possible under a fairly mild set of extra conditions. The results presented in this section originate from [60, 61, 141].

Affine inequality constraints are the simplest class of state constraints that can be handled in LCvx. The nonconvex statement of the original problem is a close relative of Problem 5.1:

$$\min_{u, t_f} \ m(x(t_f)) + \zeta \int_0^{t_f} \ell(g_1(u(t)))\, \mathrm{d}t \tag{5.14a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + Bu(t) + Ew, \tag{5.14b}$$

$$\rho_{\min} \leq g_1(u(t)),\ g_0(u(t)) \leq \rho_{\max}, \tag{5.14c}$$

$$Cu(t) \leq c, \tag{5.14d}$$

$$Hx(t) \leq h, \tag{5.14e}$$

$$x(0) = x_0,\ b(x(t_f)) = 0. \tag{5.14f}$$

First, we note that Problem 5.14 is *autonomous*, in other words the terminal cost in (5.14a), the dynamics (5.14b), and the boundary constraint (5.14f) are all independent of

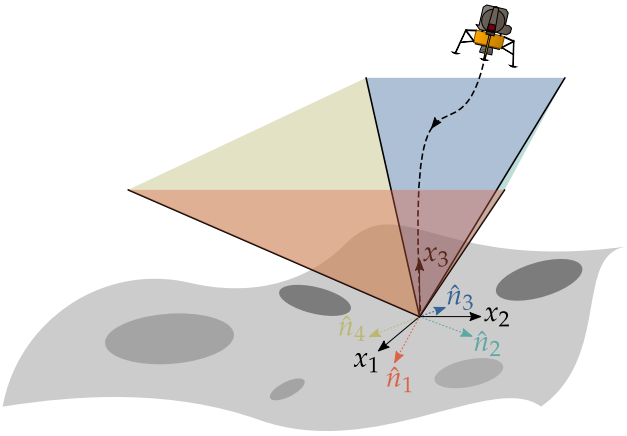


Figure 5.4: A spacecraft planetary landing glideslope cone can be approximated as an affine state constraint (5.14e). By adding more facets, a cone with circular cross-sections can be approximated to arbitrary precision.

time. Note that a limited form of time variance can still be included through an additional time integrator state whose dynamics are $\dot{z}(t) = 1$. The limitation here is that time variance must not introduce nonconvexity in the cost, in the dynamics, and in the terminal constraint (5.14f). The matrix of facet normals $H \in \mathbb{R}^{n_h \times n}$ and the vector of facet offsets $h \in \mathbb{R}^{n_h}$ define a new polytopic (affine) state constraint set. A practical use-case for this constraint is described in Example 4. Similarly, $C \in \mathbb{R}^{n_c \times m}$ and $c \in \mathbb{R}^{n_c}$ define a new polytopic subset of the input constraint set, as illustrated in Example 5.

Example 4. A typical planetary landing problem may include a glideslope constraint to ensure sufficient elevation during approach and to prevent the spacecraft from colliding with nearby terrain [35]. Letting $\hat{e}_3 \in \mathbb{R}^3$ represent the local vertical unit vector at the landing site, the glideslope requirement can be expressed as a convex second-order cone constraint:

$$\hat{e}_3^{\mathsf{T}} x \geq \|x\|_2 \cos(\gamma_{\text{gs}}), \tag{5.15}$$

where $\gamma_{\text{gs}} \in [0, \pi/2]$ is the glideslope angle, i.e. the maximum angle that the spacecraft position vector is allowed to make with the local vertical. As illustrated in Figure 5.4,

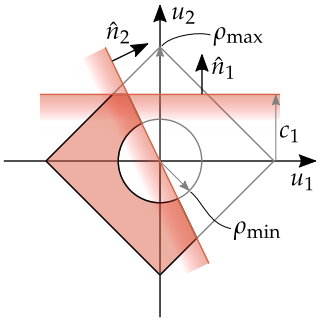


Figure 5.5: Halfspace constraints from (5.14d) can be used to select only portions of the non-convex input set defined by (5.14c). The remaining feasible input set is filled in red.

we can approximate (5.15) as an intersection of four halfspaces with the following outward normal vectors:

$$\hat{n}_1 \triangleq \begin{bmatrix} \cos(\gamma_{\text{gs}}) \\ 0 \\ -\sin(\gamma_{\text{gs}}) \end{bmatrix}, \quad \hat{n}_2 \triangleq \begin{bmatrix} 0 \\ \cos(\gamma_{\text{gs}}) \\ -\sin(\gamma_{\text{gs}}) \end{bmatrix}, \tag{5.16a}$$

$$\hat{n}_3 \triangleq \begin{bmatrix} -\cos(\gamma_{\text{gs}}) \\ 0 \\ -\sin(\gamma_{\text{gs}}) \end{bmatrix}, \quad \hat{n}_4 \triangleq \begin{bmatrix} 0 \\ -\cos(\gamma_{\text{gs}}) \\ -\sin(\gamma_{\text{gs}}) \end{bmatrix}. \tag{5.16b}$$

Thus, (5.15) can be written in the form (5.14e) by setting:

$$H = \begin{bmatrix} \hat{n}_1{}^{\mathsf{T}} \\ \hat{n}_2{}^{\mathsf{T}} \\ \hat{n}_3{}^{\mathsf{T}} \\ \hat{n}_4{}^{\mathsf{T}} \end{bmatrix}, \quad h = 0 \in \mathbb{R}^4. \tag{5.17}$$

Example 5. The input constraint set defined by (5.14c) generally lacks the ability to describe constraints on individual input components, or on combinations thereof. To enhance the descriptiveness of the input constraint set, halfspace constraints in (5.14d) may be used.

This is illustrated in Figure 5.5 where $g_0(\cdot) = \|\cdot\|_1$, $g_1(\cdot) = \|\cdot\|_2$, and

$$C = \begin{bmatrix} \hat{n}_1^\mathsf{T} \\ \hat{n}_2^\mathsf{T} \end{bmatrix}, \ c = \begin{bmatrix} c_1 \\ 0 \end{bmatrix}. \tag{5.18}$$

For example, u_1 and u_2 may describe the concentrations of two chemical reactants – hydrochloric acid (HCl) and ammonia (NH_3) – to produce ammonium chloride (NH_4Cl). Then, the first affine constraint may describe the maximum concentration of NH_3 while the second affine constraint may describe an inter-dependency in the concentrations of both reactants. Meanwhile, the by-now familiar constraint (5.14c) describes the joint reactant concentration upper and lower bounds.

Let us propose the following convex relaxation of Problem 5.14, which takes the familiar form of Problem 5.2:

$$\min_{\sigma,u,t_f} \ m(x(t_f)) + \zeta \int_0^{t_f} \ell(\sigma(t))\,\mathrm{d}t \tag{5.19a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + Bu(t) + Ew, \tag{5.19b}$$

$$\rho_{\min} \leq \sigma(t), \ g_0(u(t)) \leq \rho_{\max}, \tag{5.19c}$$

$$g_1(u(t)) \leq \sigma(t), \tag{5.19d}$$

$$Cu(t) \leq c, \tag{5.19e}$$

$$Hx(t) \leq h, \tag{5.19f}$$

$$x(0) = x_0, \ b(x(t_f)) = 0. \tag{5.19g}$$

To guarantee lossless convexification for this convex relaxation, Condition 1 can be modified to handle the new state and input constraints (5.14d) and (5.14e). To this end, use the following notion of cyclic coordinates from mechanics [147].

Definition 11. For a dynamical system $\dot{x} = f(x)$, with state $x \in \mathbb{R}^n$, any components of x that do not appear explicitly in $f(\cdot)$ are said to be *cyclic coordinates*. Without loss of

generality, we can decompose the state as follows:

$$x = \begin{bmatrix} x_c \\ x_{nc} \end{bmatrix}, \tag{5.20}$$

where $x_c \in \mathbb{R}^{n_c}$ are the cyclic and $x_{nc} \in \mathbb{R}^{n_{nc}}$ are the non-cyclic coordinates, such that $n_c + n_{nc} = n$. We can then write $\dot{x} = f(x_{nc})$.

Many mechanical systems have cyclic coordinates. For example, quadrotor drone and fixed-wing aircraft dynamics do not depend on the position or yaw angle [148]. Satellite dynamics in a circular low Earth orbit, frequently approximated with the Clohessy-Wiltshire-Hill equations [149], do not depend on the true anomaly angle which locates the spacecraft along the orbit.

With Definition 11 in hand, we call a *cyclic transformation* $\Psi_c : \mathbb{R}^n \to \mathbb{R}^n$ any mapping from the state space to itself which translates the state vector along the cyclic coordinates. In other words, assuming without loss of generality that the state is given by (5.20), we can write:

$$\Psi_c(x) = \begin{bmatrix} x_c + \Delta x_c \\ x_{nc} \end{bmatrix} \tag{5.21}$$

for some translation $\Delta x_c \in \mathbb{R}^n$. Let us now consider the polytopic state constraint (5.19f) and, in particular, let $\mathcal{F}_i = \{x \in \mathbb{R}^n : H_i^\intercal x = h_i\}$ be its i-th facet ($H_i^\intercal$ is the i-th row of H). The following condition must then hold.

Condition 4. Let $\mathcal{F}_i = \{x \in \mathbb{R}^n : H_i^\intercal x = h_i\}$ denote the i-th facet of the polytopic state constraint (5.19f), for any $i \in \{1, \ldots, n_h\}$. If $h_i \neq 0$, then there must exist a cyclic transformation Ψ_c such that

$$H_i^\intercal \, \Psi_c(x) = 0. \tag{5.22}$$

To visualize the implication of Condition 4, simply consider again the case of the landing glideslope constraint from Example 4. Because the position of a spacecraft in a constant gravity field is a cyclic coordinate, Condition 4 confirms our intuitive understanding that

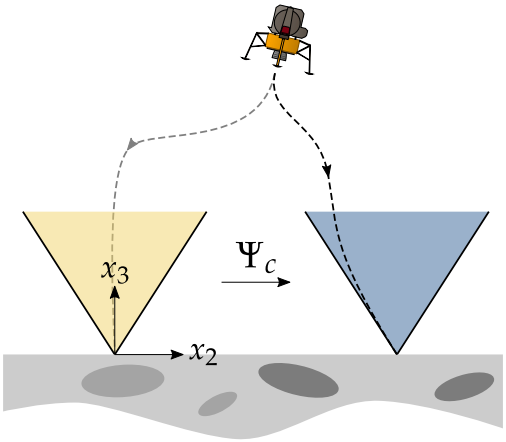


Figure 5.6: Illustration of a landing glideslope constraint (orange, sideview of Figure 5.4) that undergoes a cyclic shift in position along the positive x_2 axis, to arrive at a new landing location (blue). Thanks to Condition 4, the LCvx guarantee continues to hold for the new glideslope constraint, even though the new constraint facets are not subspaces.

we can impose landing at a coordinate other than the origin without compromising lossless convexification. Figure 5.6 provides an illustration.

From linear systems theory [60, 150, 151], it turns out that $x(t) \in \mathcal{F}_i$ can hold for a non-zero time interval (i.e. the state "sticks" to a facet) if and only if there exists a triplet of matrices $\{F_i \in \mathbb{R}^{m\times n}, G_i \in \mathbb{R}^{m\times n_v}, H_i \in \mathbb{R}^{m\times p}\}$ such that:

$$u(t) = F_i x(t) + G_i v(t) + H_i w. \tag{5.23}$$

The "new" control input $v(t) \in \mathbb{R}^{n_v}$ effectively gets filtered through (5.23) to produce a control that maintains $x(\cdot)$ on the hyperplane $\mathcal{F}_i$. The situation is illustrated in Figure 5.7 in a familiar block diagram form. While the matrix triplet is not unique, a valid triplet may be computed via standard linear algebra operations. The reader may consult these operations directly in the source code of the LCvx examples provided at the end of this article.

The proof of LCvx for Problem 5.14 was originally developed in [60, 141]. The theory behind equation (5.23) is among the most abstract in all of LCvx, and we shall not attempt a proof here. The ultimate outcome of the proof is that the following condition must hold.

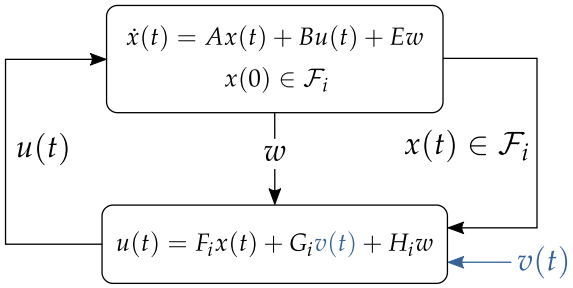


Figure 5.7: Given $x(0) \in \mathcal{F}_i$, the dynamical system (5.14b) evolves on $\mathcal{F}_i$ if and only if $u(t)$ is of the form (5.23).

Condition 5. For each facet $\mathcal{F}_i \subseteq \mathbb{R}^n$ of the polytopic state constraint (5.19f), the following "dual" linear system has no transmission zeros:

$$\dot{\lambda}(t) = -(A + BF_i)^\intercal \lambda(t) - (CF_i)^\intercal \mu(t),$$
$$y(t) = (BG_i)^\intercal \lambda(t) + (CG_i)^\intercal \mu(t).$$

Transmission zeros are defined in [152, Section 4.5.1]. Roughly speaking, if there are no trasmission zeros then there cannot exist an initial condition $\lambda(0) \in \mathbb{R}^n$ and an input trajectory $\mu \in \mathbb{R}^{n_c}$ such that $y(t) = 0$ for a non-zero time interval.

We can now state when lossless convexification holds for Problem 5.19. Note that the statement is very similar to Theorem 10. Indeed, the primary contribution of [60, 141] was to introduce Condition 5 and to show that LCvx holds by using a version of the maximum principle that includes state constraints [87, 153]. The actual lossless convexification procedure, meanwhile, does not change.

Theorem 12. *The solution of Problem 5.19 is globally optimal for Problem 5.14 if Conditions 2 and 5 hold.*

Most recently, a similar LCvx result was proved for problems with affine equality state constraints that can furthermore depend on the input [144]. These so-called mixed con-

straints are of the following form:

$$y(t) = C(t)x(t) + D(t)u(t), \tag{5.24}$$

where y, C, and D are problem data.

5.3 Quadratic State Constraints

In the last section, we showed an LCvx result for state constraints that can be represented by the affine description (5.14e). The natural next question is whether LCvx extends to more complicated state constraints. It turns out that a generalization of LCvx exists for quadratic state constraints, if one can accept a slight restriction to the system dynamics. This result was originally presented in [61, 151]. The nonconvex problem statement is as follows:

$$\min_{u,t_f} \; m(x(t_f)) + \zeta \int_0^{t_f} \ell(g_1(u(t)))\,\mathrm{d}t \tag{5.25a}$$

$$\text{s.t. } \dot{x}_1(t) = Ax_2(t), \tag{5.25b}$$

$$\dot{x}_2(t) = Bu(t) + w, \tag{5.25c}$$

$$\rho_{\min} \leq g_1(u(t)),\; g_0(u(t)) \leq \rho_{\max}, \tag{5.25d}$$

$$x_2(t)^\mathsf{T} H x_2(t) \leq 1, \tag{5.25e}$$

$$x_1(0) = x_{1,0},\; x_2(0) = x_{2,0},\; b(x(t_f)) = 0, \tag{5.25f}$$

where $(x_1, x_2) \in \mathbb{R}^n \times \mathbb{R}^n$ is the state that has been partitioned into two distinct parts, $u \in \mathbb{R}^n$ is an input of the same dimension, and the exogenous disturbance $w \in \mathbb{R}^n$ is some fixed constant. The matrix $H \in \mathbb{R}^{n\times n}$ is symmetric positive definite such that (5.25e) maintains the state in an ellipsoid. An example of such a constraint is illustrated in Example 6.

Example 6. If we set $A = I_n$ and $B = I_n$ in Problem 5.25 then the state x_2 can be interpretated as a vehicle's velocity. A maximum velocity constraint $\|x_2\|2 \leq v_{\max}$ can then

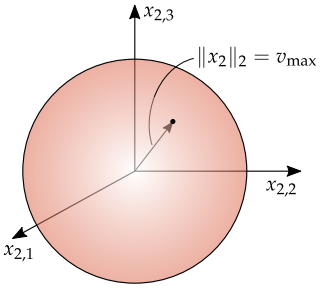


Figure 5.8: Illustration of the boundary of a maximum velocity constraint, which can be expressed as (5.25e).

equivalently be written in the form (5.25e) as

$$x_2^{\mathsf{T}}(v_{\max}^{-2} I_n) x_2 \leq 1. \tag{5.26}$$

Figure 5.8 illustrates the maximum velocity constraint.

Although the dynamics (5.25b)-(5.25c) are less general than (5.14b), they can still accommodate problems related to vehicle trajectory generation. In such problems, the vehicle is usually closely related to a double integrator system, for which $A = I_n$ and $B = I_n$ such that x_1 is the position and x_2 is the velocity of the vehicle. The control u in this case is the acceleration.

The following assumption further restricts the problem setup, and is a consequence of the lossless convexification proof [141].

Assumption 5. The matrices A, B, and H in Problem 5.14 are invertible. The functions $g_0(\cdot)$ and $g_1(\cdot)$ satisfy $\rho_{\min} \leq g_1(-B^{-1}w)$ and $g_0(-B^{-1}w) \leq \rho_{\max}$.

Assumption 5 has the direct interpretation of requiring that the disturbance w can be counteracted by an input that is feasible with respect to (5.25d). The relaxed problem once again simply convexifies the nonconvex input lower bound by introducing a slack input:

$$\min_{\sigma,u,t_f} \; m(x(t_f)) + \zeta \int_0^{t_f} \ell(\sigma(t))\,\mathrm{d}t \tag{5.27a}$$

$$\text{s.t. } \dot{x}_1(t) = Ax_2(t), \tag{5.27b}$$

$$\dot{x}_2(t) = Bu(t) + w, \tag{5.27c}$$

$$\rho_{\min} \leq \sigma(t), \;\; g_0(u(t)) \leq \rho_{\max}, \tag{5.27d}$$

$$g_1(u(t)) \leq \sigma(t), \tag{5.27e}$$

$$x_2(t)^\mathsf{T} H x_2(t) \leq 1, \tag{5.27f}$$

$$x_1(0) = x_{1,0}, \;\; x_2(0) = x_{2,0}, \;\; b(x(t_f)) = 0. \tag{5.27g}$$

Thanks to the structure of the dynamics (5.27b)-(5.27c), it can be shown that Condition 1 is automatically satisfied. On the other hand, Condition 2 must be modified to account for the quadratic state constraint.

Condition 6. If $\zeta = 0$, the vector $\nabla_x m[t_f] \in \mathbb{R}^{2n}$ must be linearly independent from the columns of the following matrix:

$$\tilde{B}_{LCvx} = \begin{bmatrix} \nabla_{x_1} b[t_f]^\mathsf{T} & 0 \\ \nabla_{x_2} b[t_f]^\mathsf{T} & 2Hx_2(t_f) \end{bmatrix} \in \mathbb{R}^{2n \times (n_b+1)}. \tag{5.28}$$

Note that Condition 6 carries a subtle but important implication. Recall that due to Assumption 3, $\nabla_x b[t_f]^\mathsf{T}$ must be full column rank. Hence, if $\zeta = 0$ then the vector $\big(0, \; 2Hx_2(t_f)\big) \in \mathbb{R}^{2n}$ and the columns of $\nabla_x b[t_f]^\mathsf{T}$ must be linearly dependent. Otherwise, $\tilde{B}_{LCvx}$ is full column rank and Condition 6 cannot be satisfied. With this in mind, the following LCvx result was proved in [151, Theorem 2].

Theorem 13. *The solution of Problem 5.27 is globally optimal for Problem 5.25 if Condition 6 holds.*

5.4 General Convex State Constraints

The preceding two sections discussed problem classes where an LCvx guarantee is available even in the presence of affine or quadratic state constraints. For obvious reasons, an engineer may want to impose more exotic constraints than afforded by Problems 5.14 and 5.25. Luckily, there exists an LCvx guarantee for general convex state constraints.

As may be expected, generality comes at the price of a somewhat weaker result. In the preceding sections, the affine and quadratic state constraints could be activated as they please: instantaneously, for period of time, and even for the entire optimal trajectory duration. In contrast, for the case of general convex state constraints, an LCvx guarantee will only hold as long as the state constraints are active *pointwise* in time. In other words, they get activated at isolated time instances and never persistently over a time interval. This result was originally provided in [59]. The nonconvex problem statement is:

$$\min_{u,t_f} \; m(x(t_f)) + \zeta \int_0^{t_f} \ell(g_1(u(t)))\,\mathrm{d}t \tag{5.29a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + Bu(t) + Ew, \tag{5.29b}$$

$$\rho_{\min} \leq g_1(u(t)),\; g_0(u(t)) \leq \rho_{\max}, \tag{5.29c}$$

$$x(t) \in \mathcal{X}, \tag{5.29d}$$

$$x(0) = x_0,\; b(x(t_f)) = 0, \tag{5.29e}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is a convex set that defines the state constraints. Without the state constraint, Problem 5.29 is nothing but the autonomous version of Problem 5.1. As for Problem 5.14, time variance can be introduced in a limited way by using a time integrator state, as long as this does not introduce nonconvexity. The relaxed problem uses the by-now familiar slack variable relaxation technique (5.29c):

$$\min_{\sigma,u,t_f} \; m(x(t_f)) + \zeta \int_0^{t_f} \ell(\sigma(t))\,\mathrm{d}t \tag{5.30a}$$

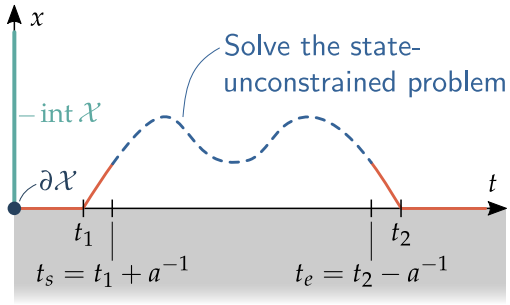


Figure 5.9: The blue dotted curve represents any segment of the optimal state trajectory for Problem 5.29 which evolves in the interior of the state constraint set. Because the optimal control problem is autonomous, any such segment is the solution to the state-unconstrained Problem 5.1. When $\zeta = 1$ and in the limit as $a \to \infty$, LCvx applies to the entire segment in $\text{int}\,\mathcal{X}$ [59].

$$\text{s.t. } \dot{x}(t) = Ax(t) + Bu(t) + Ew, \tag{5.30b}$$

$$\rho_{\min} \leq \sigma(t),\ g_0(u(t)) \leq \rho_{\max}, \tag{5.30c}$$

$$g_1(u(t)) \leq \sigma(t), \tag{5.30d}$$

$$x(t) \in \mathcal{X}, \tag{5.30e}$$

$$x(0) = x_0,\ b(x(t_f)) = 0. \tag{5.30f}$$

The LCvx proof is provided in [59, Corollary 3], and relies on recognizing two key facts:

1. When $x(t) \in \text{int}\,\mathcal{X}$ for any time interval $t \in [t_1, t_2]$, the state of the optimal control problem is unconstrained along that time interval;

2. For autonomous problems (recall the description after Problem 5.14), every segment of the trajectory is itself optimal [151].

As a result, whenever $x(t) \in \text{int}\,\mathcal{X}$, the solution of Problem 5.30 is equivalent to the solution of Problem 5.2. Consider an *interior* trajectory segment, as illustrated in Figure 5.9. The optimal trajectory for the dashed portion in Figure 5.9 is the solution of the following

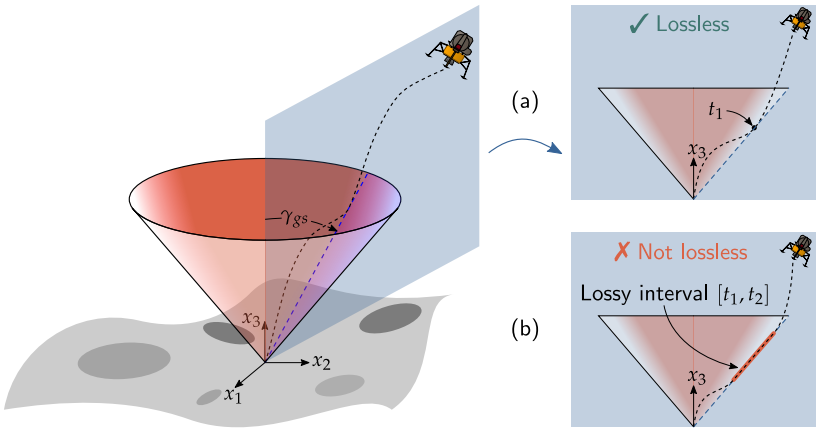


Figure 5.10: Illustration of when lossless convexification with general convex state constraints may fail. The two figures on the right show an in-plane projection of the 3D figure on the left. In (a) the glideslope constraint is only activated once prior to landing, hence the solution is lossless. In (b) the constraint is activated for a non-trivial duration and the solution may be infeasible over that interval.

fixed final state, free final time problem:

$$\min_{u,t_f} \; m(z(t_e)) + \zeta \int_{t_s}^{t_e} \ell(g_1(u(t)))\,\mathrm{d}t \tag{5.31a}$$

$$\text{s.t. } \dot{z}(t) = Az(t) + Bu(t) + Ew, \tag{5.31b}$$

$$\rho_{\min} \leq g_1(u(t)),\; g_0(u(t)) \leq \rho_{\max}, \tag{5.31c}$$

$$z(t_s) = x(t_s),\; z(t_e) = x(t_e). \tag{5.31d}$$

We recognize that Problem 5.31 is an instance of Problem 5.1 and, as long as $\zeta = 1$ (in order for Condition 2 to hold), Theorem 10 applies. Because $a > 0$ can be arbitrarily large in Figure 5.9, lossless convexification applies over the open time interval (t_1, t_2). Thus, the solution segments of the relaxed problem that lie in the interior of the state constraint set are feasible and globally optimal for the original Problem 5.29.

Example 7. Let us go back to the landing glideslope constraint for a spacecraft, which was previously motivated in Example 4. Because Problem 5.29 allows us to use any convex state

constraint, we can directly use the second-order cone constraint (5.15).

As illustrated in Figure 5.10, lossless convexification in this case will only hold if the spacecraft touches the "landing cone" a finite number of times. In Figure 5.10a, the glideslope constraint is activated only once at time t_1 prior to landing. Hence, $\mathcal{T} = \{t \in [0, t_f] : x(t) \in \partial\mathcal{X}\} = \{0, t_1\}$ is a discrete set and Theorem 14 holds. Note that the constraint may be activated at other times, e.g. $\mathcal{T} = \{0, t_1, t_2\}$, as long as these times are all isolated points.

In Figure 5.10b there is a non-trivial interval for which the glideslope constraint is active. This results in a non-discrete set of activation times $\mathcal{T} = \{0\} \cup [t_1, t_2]$. For $t \in [t_1, t_2]$, there is not LCvx guarantee, so the solution of Problem 5.30 may be infeasible for Problem 5.29 over that interval.

For historical context, LCvx theory was initially developed for the express use in planetary rocket landing. For this application, glide slope constraint activation behaves like Figure 5.10a, so LCvx holds in that important case. Indeed, the constraint (5.15) has been part of NASA Jet Propulsion Laboratory's optimization-based rocket landing algorithm flight tests [1, 9, 154, 155, 156].

The same cannot be said when $x(t) \in \partial\mathcal{X}$. During these segments, the solution can become infeasible for Problem 5.29. However, as long as $x(t) \in \partial\mathcal{X}$ at isoalted time instances, LCvx can be guaranteed to hold. This idea is further illustrated in Example 7.

When $\zeta = 0$, the situation becomes more complicated because Condition 2 does not hold for Problem 5.31. This is clear from the fact that the terms defined in Condition 2 become:

$$m_{\text{LCvx}} = \begin{bmatrix} \nabla_z m[t_e] \\ 0 \end{bmatrix}, \quad B_{\text{LCvx}} = \begin{bmatrix} I_n \\ 0 \end{bmatrix},$$

which are clearly not linearly independent. Thus, even for interior segments the solution may be infeasible for Problem 5.29. To remedy this, [59, Corollary 4] suggests Algorithm 5. At its core, the algorithm relies on the following simple idea. By solving Problem 5.30 with the suggested modifications in Algorithm 5 L3, every interior segment once again becomes

Algorithm 5 Solution algorithm for Problem 5.29. When $\zeta = 0$, a two-step procedure is used where an auxiliary problem with $\zeta = 1$ searches over the optimal solutions to the original problem.

1: Solve Problem 5.30 to obtain $x^*(t_f^*)$
2: **if** $\zeta = 0$ **then**
3: Solve Problem 5.30 again, with the modifications:
- Use the cost $\int_0^{t_f} \ell(\sigma(t))\mathrm{d}t$
- Set $b(x(t_f)) = x(t_f) - x^*(t_f^*)$

4: **end if**

an instance of Problem 5.1 for which Theorem 10 holds. Furthermore, due to the constraint $x(t_f) = x^*(t_f^*)$, any solution to the modified problem will be optimal for the original formulation where $\zeta = 0$ (since $m(x(t_f)) = m(x^*(t_f^*))$).

This modification can be viewed as a search for an equivalent solution for which LCvx holds. As a concrete example, Problem 5.30 may be searching for a minimum miss distance solution for a planetary rocket landing trajectory. The ancillary problem in Algorithm 5 can search for a minimum fuel solution that achieves the same miss distance. Clearly, other running cost choices are possible. Thus, the ancillary problem's running cost becomes an extra tuning parameter.

We are now able to summarize the lossless convexification result for problems with general convex state constraints.

Theorem 14. *Algorithm 5 returns the globally optimal solution of Problem 5.29 if the state constraint* (5.29d) *is activated at isolated time instances, and Conditions 1 and 2 hold.*

5.5 Nonlinear Dynamics

A unifying theme of the previous sections is the assumption that the system dynamics are linear. In fact, across all LCvx results that we have mentioned so far, the dynamics did not vary much from the first formulation in (5.1b). Many engineering applications, however, involve non-negligible nonlinearities. A natural question is then whether the theory of lossless

convexification be extended to systems with general nonlinear dynamics.

An LCvx result is available for a class of nonlinear dynamical systems. The groundwork for this extension was presented in [116]. The goal here is to show that the standard input set relaxation based on the LCvx equality constraint is also lossless when the dynamics are nonlinear. Importantly, note that the dynamics themselves are not convexified, so the relaxed optimization problem is still nonlinear, and it is up to the user to solve the problem to global optimality. This is possible in special cases, for example if the nonlinearities are approximated with piecewise affine functions. This yields a mixed-integer convex problem whose globally optimal solution can be found via mixed-integer programming [157].

With this introduction, let us begin by introducing the generalization of Problem 5.1 that we shall solve using LCvx:

$$\min_{u,t_f} \; m(t_f, x(t_f)) + \zeta \int_0^{t_f} \ell(g(u(t)))\,\mathrm{d}t \tag{5.32a}$$

$$\text{s.t. } \dot{x}(t) = f(t, x(t), u(t), g(u(t))), \tag{5.32b}$$

$$\rho_{\min} \leq g(u(t)) \leq \rho_{\max}, \tag{5.32c}$$

$$x(0) = x_0, \;\; b(t_f, x(t_f)) = 0, \tag{5.32d}$$

where $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ defines the nonlinear dynamics. Just as for Problem 5.8, it is required that $g_0(\cdot) = g_1(\cdot) \triangleq g(\cdot)$. Consider the following convex relaxation of the input constraint by using a slack input:

$$\min_{\sigma,u,t_f} \; m(t_f, x(t_f)) + \zeta \int_0^{t_f} \ell(\sigma(t))\,\mathrm{d}t \tag{5.33a}$$

$$\text{s.t. } \dot{x}(t) = f(t, x(t), u(t), \sigma(t)), \tag{5.33b}$$

$$\rho_{\min} \leq \sigma(t) \leq \rho_{\max}, \tag{5.33c}$$

$$g(u(t)) \leq \sigma(t), \tag{5.33d}$$

$$x(0) = x_0, \;\; b(t_f, x(t_f)) = 0. \tag{5.33e}$$

Note that the slack input σ makes a new appearance in the dynamics (5.33b). The more complicated dynamics call for an updated version of Condition 1 in order to guarantee that LCvx holds.

Condition 7. The pair $\{\nabla_x f[t], \nabla_u f[t]\}$ must be totally controllable on $[0, t_f]$ for all feasible sequences of $x(\cdot)$ and $u(\cdot)$ for Problem 5.33 [116, 145].

Using the above condition, we can state the following quite general LCvx guarantee for problems that fit the Problem 5.32 template.

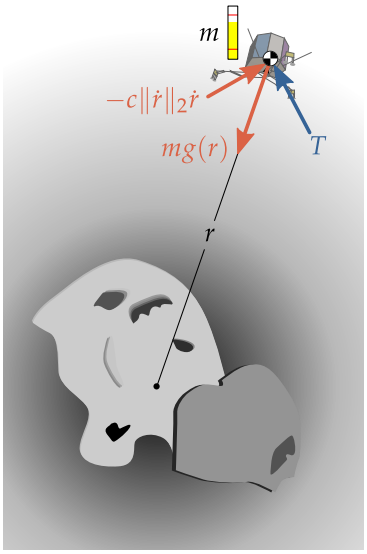
Theorem 15. *The solution of Problem 5.33 is globally optimal for Problem 5.32 if Conditions 2 and 7 hold.*

Alas, Condition 7 can in general be quite difficult to check. Nevertheless, two general classes of systems have been shown to automatically satisfy this condition thanks to the structure of their dynamics [116]. These classes accommodate vehicle trajectory generation problems with double integrator dynamics and nonlinearities like mass depletion, aerodynamic drag and nonlinear gravity. The following discussion of these system classes can appear hard to parse at first sight. For this reason, we provide two practical examples of systems that belong to each class in Example 8.

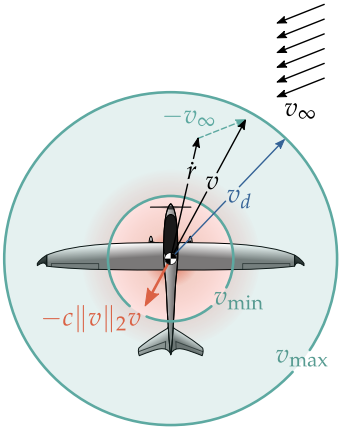
Example 8. At first sight, the discussion around Corollaries 4 and 5 may be hard to parse into something useful. However, we will now show two concrete and very practical examples of dynamical systems that satisfy these corollaries. Both examples originate from [116].

Nonlinear Rocket Landing. First, LCvx can be used for rocket landing with nonlinear gravity and aerodynamic drag. Both effects are important for landing either on small celestial bodies with a weak gravitational pull, or on planets with a thick atmosphere (such as Earth). In this case, the lander dynamics can be written as:

$$\ddot{r}(t) = g(r(t)) - \frac{c}{m(t)}\|\dot{r}(t)\|_2\dot{r}(t) + \frac{T(t)}{m(t)}, \tag{5.34a}$$



(a) Landing a rocket in the presence of nonlinear gravity and atmospheric drag.



(b) Unmanned aircraft trajectory generation with a stall constraint and wind drag.

Figure 5.11: Illustrations of two nonlinear system examples for which the lossless convexification result of Theorem 15 can be applied.

$$\dot{m}(t) = -\alpha\|T(t)\|_2, \tag{5.34b}$$

where r denotes position, m is mass, T is thrust, c is a drag coefficient, α relates to the rocket engine's specific impulse, and $g : \mathbb{R}^3 \to \mathbb{R}^3$ is the nonlinear gravity model. An illustration is given in Figure 5.11a.

We can rewrite the above dynamics using the template of (5.32b) by defining the state $x \triangleq (r; \dot{r}; m) \in \mathbb{R}^7$, the input $u \triangleq T \in \mathbb{R}^3$, and the input penalty function $g(u) \triangleq \|u\|_2$. The equations of motion can then be written as (omitting the time argument from x and u for conciseness):

$$f(t, x, u, g(u)) = \begin{bmatrix} f_r(x, u) \\ f_{\dot{r}}(x) \\ f_m(g(u)) \end{bmatrix},$$

$$= \begin{bmatrix} \dot{r} \\ g(r) - cm^{-1}\|\dot{r}\|_2\dot{r} + Tm^{-1} \\ -\alpha\|T\|_2 \end{bmatrix}.$$

This system belongs to the class in Corollary 5. In particular, let $f_1 = (f_r; f_{\dot{r}})$ and $f_2 = f_m$. Working out the algebra in (5.39), we have:

$$M = \begin{bmatrix} 0 & m^{-1}I \\ m^{-1}I & M_{22} \end{bmatrix}, \tag{5.35}$$

where $M_{22} = -\frac{c}{m^2}\left(\|\dot{r}\|_2 - \frac{\dot{r}\dot{r}^\mathsf{T}}{\|\dot{r}\|_2}\right) + \alpha\frac{TT^\mathsf{T}}{m^2\|T\|_2}$. Thanks to the off-diagonal terms, $\text{null}(M) = \{0\}$ unconditionally, so the rocket lander dynamics satisfy the requirements of Corollary 5.

AAV Trajectory Generation Without Stalling. An autonomous aerial vehicle (AAV) can lose control and fall out of the sky if its airspeed drops below a certain value. This occurs because the wings fail to generate enough lift, resulting in an aerodynamic *stall*. It was shown in [116] that a stall speed constraint of the form $v_{\min} \leq \|v\|_2$ can be handled by LCvx via the following dynamics:

$$\dot{r}(t) = v(t) + v_\infty(t), \tag{5.36a}$$

$$\dot{v}(t) = \kappa\big(v_d(t) - v(t)\big) - c\|v(t)\|_2 v(t), \tag{5.36b}$$

where v is the airspeed, $\dot{r}$ is the ground velocity, and v_∞ is the velocity of the air mass relative to the ground (also known as the freestream velocity). An illustration is given in Figure 5.11b.

The important transformation in (5.36) is to make the desired airspeed v_d the control variable, and to include a velocity control inner-loop via a proportional controller with gain κ. Because the velocity dynamics are first order, v converges to v_d along a straight path in the velocity phase plane, hence the stall speed constraint is closely approximated for small

control errors $\|v_d - v\|_2$.

We can rewrite the dynamics using the template of (5.32b) by defining the state $x \triangleq (r; v) \in \mathbb{R}^6$ and the input $u \triangleq v_d \in \mathbb{R}^3$. The equations of motions can then be written as:

$$\begin{aligned} f(t, x, u, g(u)) &= \begin{bmatrix} f_r(t, x) \\ f_v(x, u) \end{bmatrix}, \\ &= \begin{bmatrix} v + v_\infty \\ -\kappa v - c\|v\|_2 v + \kappa v_d \end{bmatrix}. \end{aligned}$$

This system belongs to the class in Corollary 4. In particular, let $f_1 = f_r$ and $f_2 = f_{\dot{r}}$. We then have $\nabla_u f_2 = \kappa I$ and $\nabla_{x_2} f_1 = I$. Therefore, $\text{null}(\nabla_u f_2) = \{0\}$ and $\text{null}(\nabla_{x_2} f_1) = \{0\}$, so the aircraft dynamics satisfy the requirements of Corollary 4.

The first corollary to Theorem 15 introduces the first class of systems. A key insight is that the nullspace conditions of the corollary require that $2m \geq n$, in other words there are at least twice as many control variables as there are state variables. This is satisfied by some vehicle trajectory generation problems where $2m = n$, for example when the state consists of position and velocity while the control is an acceleration that acts on all the velocity states.

Corollary 4. *Suppose that the dynamics* (5.32b) *are of the form:*

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad f(t, x, u, g(u)) = \begin{bmatrix} f_1(t, x) \\ f_2(t, x, u) \end{bmatrix}, \tag{5.37}$$

where $\text{null}(\nabla_u f_2) = \{0\}$ *and* $\text{null}(\nabla_{x_2} f_1) = \{0\}$. *Then Theorem 15 applies if Condition 2 holds.*

The next corollary to Theorem 15 introduces the second class of systems, for which $2m < n$ is allowed. This class is once again useful for vehicle trajectory generation problems where the dynamics are given by (5.38) and $g(u)$ is a function that measures control effort. A practical example is when the state x_2 is mass, which is depleted as a function of the

control effort (such as thrust for a rocket).

Corollary 5. *Suppose that the dynamics* (5.32b) *are of the form:*

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad f(t, x, u, g(u)) = \begin{bmatrix} f_1(t, x, u) \\ f_2(t, g(u)) \end{bmatrix}. \tag{5.38}$$

Define the matrix:

$$M \triangleq \begin{bmatrix} \nabla_u f_1 \\ \dfrac{\mathrm{d}\nabla_u f_1}{\mathrm{d}t} - (\nabla_u f)\nabla_x f_1 \end{bmatrix}. \tag{5.39}$$

Furthermore, suppose that the terminal constraint function b is affine and $x_2(t_f)$ is unconstrained, such that $\nabla_{x_2} b = 0$. Then Theorem 15 applies if $\operatorname{null}(M) = \{0\}$ *and Condition 2 holds.*

It must be emphasized that Problem 5.33 is still a nonlinear program and that for Theorem 15 to hold, a globally optimal solution of Problem 5.33 must be found. Although this cannot be done for general nonlinear programming, if the dynamics f are piecewise affine then the problem can be solved to global optimality via mixed-integer programming [118, 157, 158]. In this case, convexification of the nonconvex input lower bound reduces the number of disjunctions in the branch-and-bound tree, and hence lowers the problem complexity [116]. Several examples of nonlinear systems that can be modeled in this way, and which comply with Corollaries 4 and 5, are illustrated in Example 9.

Example 9. Piecewise affine functions can be used for arbitrarily accurate approximation of any nonlinear function. This is the technique used by [116] to write the nonlinear dynamics (5.33b) in piecewise affine form. Doing so enables solving Problem 5.33 to global optimality via mixed-integer programming, which is imperative for the LCvx guarantee of Theorem 15. We now show how a piecewise affine approximation can be obtained in the general case, and concretely in the case of the dynamics from Example 8.

General Case. We begin by finding a first order approximation of the nonlinear equation $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ from (5.33b) about an operating point $(t, \bar{x}^i, \bar{u}^i) \in \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m$, where i is an integer index that shall become clear in a moment. Without loss of generality, assume that f is decomposable into an affine and a non-affine part:

$$f = f_a + f_{na}. \tag{5.40}$$

The first order Taylor expansion of f is given by:

$$\begin{aligned} f &\approx f_a + f_{na}^i, \\ f_{na}^i &\triangleq \bar{f}_{na} + \big(\nabla_x \bar{f}_{na}\big)(x - \bar{x}) + \big(\tilde{\nabla}_u \bar{f}_{na}\big)(u - \bar{u}), \end{aligned} \tag{5.41}$$

where we have used the shorthand:

$$f_{na}^i = f_{na}^i(t, x, u, g(u)), \tag{5.42a}$$

$$\bar{f}_{na} = f_{na}(t, \bar{x}^i, \bar{u}^i, g(\bar{u}^i)), \tag{5.42b}$$

$$\tilde{\nabla}_u f_{na} = \nabla_u f_{na} + (\nabla_g f_{na}) \nabla g. \tag{5.42c}$$

In (5.42c), we understand $\nabla_g f_{na}$ as the Jacobian of f_{na} with respect to its fourth argument. Suppose that the linearization in (5.41) is sufficiently accurate only over the hyper-rectangular region $\mathcal{R}^i \subset \mathbb{R}^n \times \mathbb{R}^m$ defined by the following inequalities:

$$\bar{x}^i + b_{L,x}^i \leq x \leq \bar{x}^i + b_{U,x}^i, \tag{5.43a}$$

$$\bar{u}^i + b_{L,u}^i \leq u \leq \bar{u}^i + b_{U,u}^i, \tag{5.43b}$$

where $b_{L,x}^i$ and $b_{U,x}^i$ represent the upper and lower bounds on the state, while $b_{L,u}^i$ and $b_{U,u}^i$ relate to the input. The index i denotes the i-th validity region. Without loss of generality, we assume $\mathcal{R}^i \cap \mathcal{R}^j = \emptyset$ if $i \neq j$. In general, $i = 1, \ldots, N$, which means that f is approximated

by affine functions over N regions. The piecewise affine approximation of f can then be written as:

$$f_{pwa} \triangleq f_a + f_{na}^i, \text{ for } i \text{ such that } (x,u) \in \mathcal{R}^i. \tag{5.44}$$

The big-M formulation can be used to write (5.44) in a form that is readily used in a mixed-integer program [158]. To this end, let $M > 0$ be a sufficiently large fixed scalar parameter, and let $z^i \in \{0,1\}$ be a binary variable indicating that $(x,u) \in \mathcal{R}^i$ if and only if $z^i = 1$. Then, the piecewise affine dynamics in mixed-integer programming form are encoded by the following set of constraints:

$$\dot{x} = f_a(t,x,u,g(u)) + f_{na}^i(t,x,u,g(u)), \tag{5.45a}$$
$$x \geq \bar{x}^i + b_{L,x}^i - M(1-z^i), \tag{5.45b}$$
$$x \leq \bar{x}^i + b_{U,x}^i + M(1-z_i), \tag{5.45c}$$
$$u \geq \bar{u}^i + b_{L,u}^i - M(1-z^i), \tag{5.45d}$$
$$u \leq \bar{u}^i + b_{U,u}^i + M(1-z_i), \tag{5.45e}$$
$$1 = \textstyle\sum_{i=1}^N z^i. \tag{5.45f}$$

Nonlinear Rocket Landing. Consider the rocket dynamics in (5.34a) and, for simplicity, suppose that the mass is constant. Define the state $x \triangleq (r;\dot{r}) \in \mathbb{R}^6$ and the input $u \triangleq T \in \mathbb{R}^3$. The non-affine part of the dynamics (5.40) then takes the particular form:

$$f_{na} = \begin{bmatrix} 0 \\ g(r) - cm^{-1}\|\dot{r}\|_2\dot{r} \end{bmatrix}. \tag{5.46}$$

For simplicity, let us consider only the nonlinear gravity term, $g(r)$. We will consider atmospheric drag for AAV trajectory generation below. Suppose that $g(r) = (0;0;g_z(r_z))$, which means that we only need to consider the vertical component $g_z : \mathbb{R} \to \mathbb{R}$. A typical profile is $g_z(r_z) = -\mu/r_z^2$, where μ is the gravitational parameter. Given a reference point

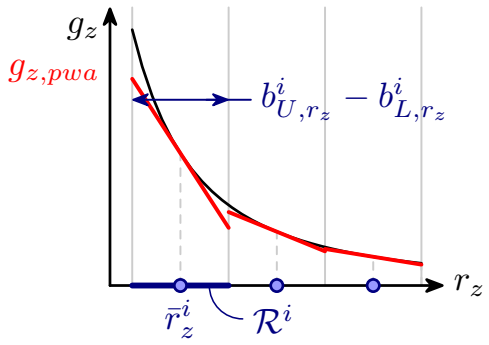


Figure 5.12: Illustration of a piecewise affine approximation of g_z, the z-component of the nonlinear gravity force, using (5.49).

$\bar{r}^i_z$, we have:

$$g^i_z = g_z(\bar{r}^i_z) + g'_z(\bar{r}^i_z)(r_z - \bar{r}_z). \tag{5.47}$$

Using (5.44) and (5.47), Figure 5.12 draws $g_{z,pwa}$.

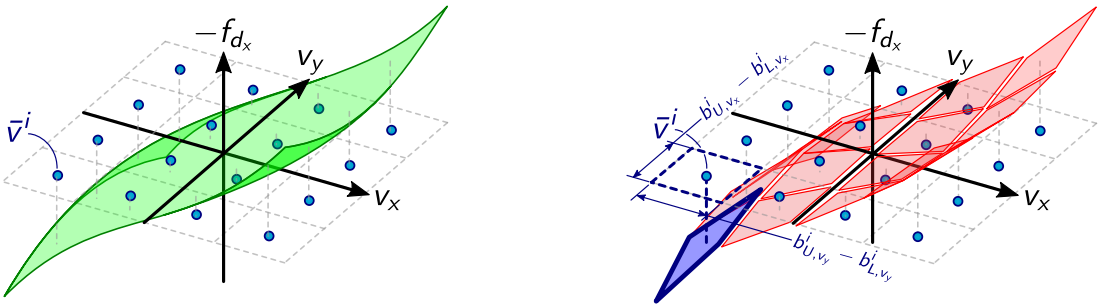
AAV Trajectory Generation Without Stalling. Consider the AAV dynamics in (5.36) and, for simplicity, assume constant altitude flight. Define the satte $x \triangleq (r; v) \in \mathbb{R}^4$ and the input $u = v_d \in \mathbb{R}^2$. The non-affine part of the dynamics (5.40) then takes the particular form:

$$f_{na} = \begin{bmatrix} 0 \\ -c^{-1}\|v\|_2 v \end{bmatrix}. \tag{5.48}$$

Let us focus on the aerodynamic drag term, $f_d = -c\|v\|_2 v$. The first order Taylor expansion about a reference airspeed $\bar{v}^i$ is:

$$f^i_d = -c^{-1}\|\bar{v}^i\|_2 \left(\left[I + \|\bar{v}^i\|_2^{-2} \bar{v}^i \bar{v}^{i\mathsf{T}} \right] v - \bar{v}^i \right). \tag{5.49}$$

Using (5.44) and (5.49), Figure 5.13 draws $f_{d,pwa}$.



(a) Surface of the continuous function $-f_{d,x}$. The blue dots display operating points at which the gradient is evaluated.

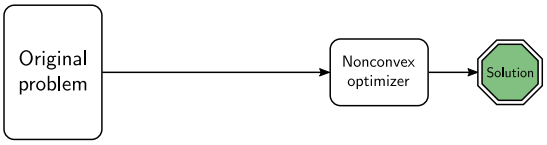
(b) Surface of the discontinuous piecewise affine approximation $-f_{d_x,pwa}$. The blue dashed rectangle in the airspeed space shows the boundary of the approximation validity region $\mathcal{R}^i$.

Figure 5.13: Illustration of a piecewise affine approximation of f_{d_x}, the x-component of the aerodynamic drag force, using (5.49).

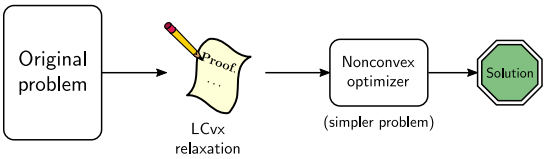
5.6 Embedded Lossless Convexification

The reader will notice that the LCvx theory of the previous sections deals with special case problems that are simple enough, and whose nonconvexity is "just right" for an LCvx guarantee to be provable by the maximum principle. While such problems have found their practical use in problems like spaceflight [1] and quadrotor path planning [159], this leaves out many trajectory generation applications that do not fit the tight mold of original problems and conditions of the previous sections.

Despite this apparent limitation, LCvx is still highly relevant for problems that simply do not conform to one of the forms given in the previous sections. In this case, we will assume that the reader is facing the challenge of solving a nonconvex optimal control problem that fits the mold of Problem 7.42 (presented in Chapter 7), and is considering whether LCvx can help. There is evidence that the answer can be affirmative, by using LCvx theory only on the constraints that are losslessly convexifiable. We call this *embedded LCvx*, because it is used to convexify only part of the problem, while the rest is handled by other nonconvex optimization methods such as those presented in the next section. Because LCvx reduces the



(a) The solution process for a nonconvex optimal control problem, without using LCvx.



(b) The solution process for a nonconvex optimal control problem, where LCvx is embedded to convexify part of the original problem.

Figure 5.14: Illustration of how embedded LCvx can be used to solve an optimal control problem that does not fit one of the templates presented in this section.

nonconvexity present in the original problem, it can significantly improve the convergence properties and reduce the computational cost to solve the remaining problem.

The basic procedure for applying embedded LCvx is illustrated in Figure 5.14. As shown in Figure 5.14b, we reiterate that LCvx is not a computation scheme, but rather it is a convex relaxation with an accompanying proof of equivalence to the original problem. As such, it happens prior to the solution and simply changes the problem description seen by the subsequent numerical optimization algorithm.

There are a number of examples of embedded LCvx that we can mention. First, the previous section can also be intepreted as embedded LCvx, since [116] solves a mixed-integer optimal control problem where the nonconvex input constraint (5.32c) is convexified. Another example is [160], lossless convexification was embedded in a mixed-integer unmanned aerial vehicle (AAV) trajectory generation problem in order to convexify a stall speed constraint if the form:

$$0 < v_{\min} \leq \|v_{cmd}(t)\|_2 \leq v_{\max}, \tag{5.50}$$

where the input $v_{cmd}(\cdot) \in \mathbb{R}^3$ is the commanded velocity, while $v_{\min}$ and $v_{\max}$ are lower and upper bounds to guarantee a stable flight envelope. The same constraint is also considered in [116]. In [161], the authors consider a highly nonlinear planetary entry trajectory optimization where the control input is the bank angle $\beta \in \mathbb{R}$, parametrized via two inputs

$u_1 \triangleq \cos(\beta)$ and $u_2 \triangleq \sin(\beta)$. The associated constraint $\|u\|_2^2 = 1$ is convexified to $\|u\|_2^2 \leq 1$, and equality at optimality is proven in LCvx-like fashion (the authors call it "assurance of active control constraint"). Similar methods are used in [162] in the context of rocket landing with aerodynamic controls. A survey of related methods is available in [37]. Finally, we will mention [159, 163], where embedded LCvx was used to convexify the nonconvex input lower bound constraint and an attitude pointing constraint in a sequential convex programming framework for solving nonlinear optimal control problems for rockets and quadrotor drones. The quadrotor application in particular is demonstrated as a numerical example at the end of this article.

As a result of the success of these applications, we foresee there being further opportunities to use LCvx as a strategy to derive simpler problem formulations. The result would be a speed up in computation for optimization-based trajectory generation.

5.7 Toy Example

The following example provides a simple illustration of how LCvx can be used to solve a nonconvex problem. This example is meant to be a "preview" of the practical application of LCvx. More challenging and realistic examples are given in Part III.

The problem that we will solve is minimum-effort control of a double integrator system (such as a car) with a constant "friction" term g. This can be written as a linear time-invariant instance of Problem 5.1:

$$\min_u \quad \int_0^{t_f} u(t)^2 \, \mathrm{d}t \tag{5.51a}$$

$$\text{s.t. } \dot{x}_1(t) = x_2(t), \tag{5.51b}$$

$$\dot{x}_2(t) = u(t) - g, \tag{5.51c}$$

$$1 \leq |u(t)| \leq 2, \tag{5.51d}$$

$$x_1(0) = x_2(0) = 0, \tag{5.51e}$$

$$x_1(t_f) = s, \ x_2(t_f) = 0, \ t_f = 10. \tag{5.51f}$$

The input $u(\cdot) \in \mathbb{R}$ is the acceleration of the car. The constraint (5.51d) is a nonconvex one-dimensional version of the constraint (5.3). Assuming that the car has unit mass, the integrand in (5.51a) has units of Watts. The objective of Problem 5.51 is therefore to move a car by a distance s in $t_f = 10$ seconds while minimizing the average power. Following the relaxation template provided by Problem 5.2, we propose the following convex relaxation to solve Problem 5.51:

$$\min_{\sigma,u} \quad \int_0^{t_f} \sigma(t)^2 \, \mathrm{d}t \tag{5.52a}$$

$$\text{s.t. } \dot{x}_1(t) = x_2(t), \tag{5.52b}$$

$$\dot{x}_2(t) = u(t) - g, \tag{5.52c}$$

$$1 \leq \sigma(t) \leq 2, \tag{5.52d}$$

$$|u(t)| \leq \sigma(t), \tag{5.52e}$$

$$x_1(0) = x_2(0) = 0, \tag{5.52f}$$

$$x_1(t_f) = s, \ x_2(t_f) = 0, \ t_f = 10. \tag{5.52g}$$

To guarantee that LCvx holds, in other words that Problem 5.52 finds the globally optimal solution of Problem 5.51, let us first attempt to verify the conditions of Theorem 10. In particular, we need to show that Conditions 1 and 2 hold. First, from (5.51b)-(5.51c), we can extract the following state-space matrices:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{5.53}$$

We can verify that Condition 1 holds by either showing that the controllability matrix is full rank, or by using the PBH test [114]. Next, from (5.52a) and (5.52f)-(5.52g), we can

extract the following terminal cost and terminal constraint functions:

$$m(t_f, x(t_f)) = 0, \ b(t_f, x(t_f)) = \begin{bmatrix} t_f - 10 \\ x_1(t_f) - s \\ x_2(t_f) \end{bmatrix}. \tag{5.54}$$

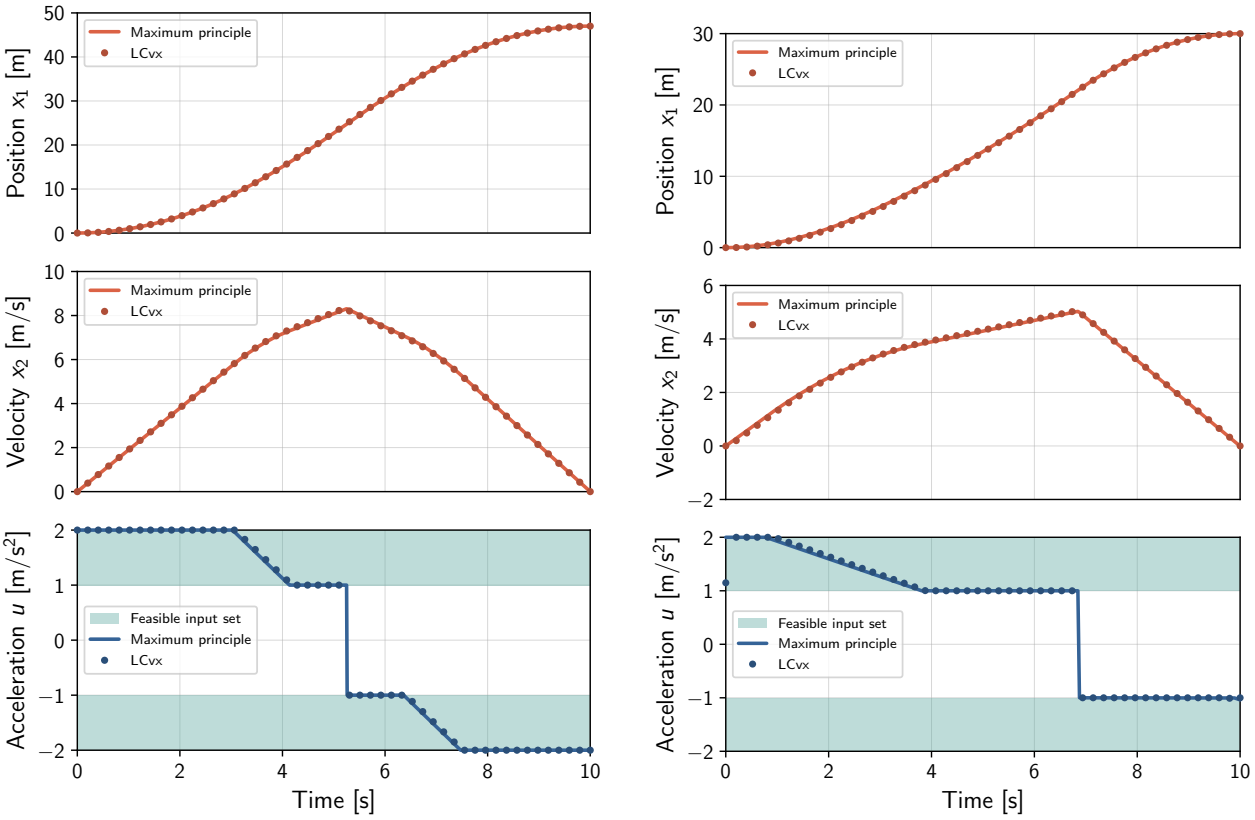
We can now substitute (5.54) into (5.7) to obtain:

$$m_{\text{LCvx}} = \begin{bmatrix} 0 \\ 0 \\ \sigma(t_f)^2 \end{bmatrix}, \ B_{\text{LCvx}} = I_3. \tag{5.55}$$

Thus, B_{LCvx} is full column rank and its columns cannot be linearly independent from m_{LCvx}. We conclude that Condition 2 does not hold, so Theorem 10 cannot be applied. In fact, Problem 5.51 has both a fixed final time and a fixed final state. This is exactly the edge case for which traditional LCvx does not apply, as was mentioned in the previous section on future LCvx. Instead, we fall back on Theorem 20 which says that Condition 2 is not needed as long as t_f is between the minimum and optimal times for Problem 5.51. It turns out that this holds for the problem parameters used in Figure 5.15. The minimum time is just slightly below 10 s and the optimal time is ≈ 13.8 s for Figure 5.15a and ≈ 13.3 s for Figure 5.15b. Most interestingly, lossless convexification fails (i.e., (5.52e) does not hold with equality) for t_f values almost exactly past the optimal time for Figure 5.15a, and just slightly past it for Figure 5.15b.

Although Problem 5.52 is convex, it has an infinite number of solution variables because time is continuous. To be able to find an approximation of the optimal solution using a numerical convex optimization algorithm, the problem must be temporally discretized. To this end, we apply a first-order hold (FOH) discretization with $N = 50$ temporal nodes, as explained in [49, 53].

Looking at the solutions in Figure 5.15 for two values of the friction parameter g, we



(a) Solution of Problem 5.52 for $g = 0.1$ m/s^2 and $s = 47$ m.

(b) Solution of Problem 5.52 for $g = 0.6$ m/s^2 and $s = 30$ m.

Figure 5.15: LCvx solutions of Problem 5.52 for two scenarios. The close match of the analytic solution using the maximum principle (drawn as a continuous line) and the discretized solution using LCvx (drawn as discrete dots) confirms that LCvx finds the globally optimal solution of the problem.

can see that the nonconvex constraint (5.51d) holds in both cases. We emphasize that this is despite the trajectories in Figure 5.15 coming from the solution of Problem 5.52, where $|u(t)| < 1$ is feasible. The fact that this does not occur is the salient feature of LCvx theory, and for this problem it is guaranteed by Theorem 20. Finally, we note that Figure 5.15 also plots the analytical globally optimal solution obtained via the maximum principle, where no relaxation nor discretization is made. The close match between this solution and the numerical LCvx solution further confirms the theory, as well as the accuracy of the FOH discretization method. Note that the mismatch at $t = 0$ in the acceleration plot in Figure 5.15b is a benign single-time-step discretization artifact that is commonly observed in LCvx numerical solutions.

Chapter 6

LOSSLESS CONVEXIFICATION OF MIXED-INTEGER OPTIMAL CONTROL PROBLEMS

This chapter presents a convex optimization-based method for finding the globally optimal solution for a class of mixed-integer non-convex optimal control problems. The optimization problems in this class are non-convex in the input norm, which is a semi-continuous variable that can be zero or lower- and upper-bounded. Using the method of *lossless convexification*, the non-convex problem is relaxed to a convex problem whose optimal solution is proved to be optimal almost everywhere for the original problem. The relaxed problem can be solved using second-order cone programming, which is a subclass of convex optimization for which there exist numerically reliable solvers with convergence guarantees and polynomial time complexity [33, 42, 43, 44, 45].

The results of this chapter represent a significant extension of traditional lossless convexification theory from Chapter 5. To the best of our knowledge, our publication [52] was the first work to propose a rigorous and fully convex solution method to a fairly broad class of mixed-integer optimal control tasks. Most recently, an interesting addition to mixed-integer lossless convexification theory appeared in a related work [142]. We discuss how the two contributions are complementary at the end of this chapter in Section 6.7.

To corroborate the effectiveness and the utility of the results presented herein, two examples are given. First, docking trajectories are generated for an in-orbit rendezvous operation between a spacecraft and a rotating space station. Second, landing trajectories are generated for rocket landing with a coupled thrust-gimbal constraint (in other words, the allowable gimbal angle shrinks for larger main rocket engine thrust values). In both cases, the approach of this chapter significantly outpaces classical mixed-integer programming, which either fails

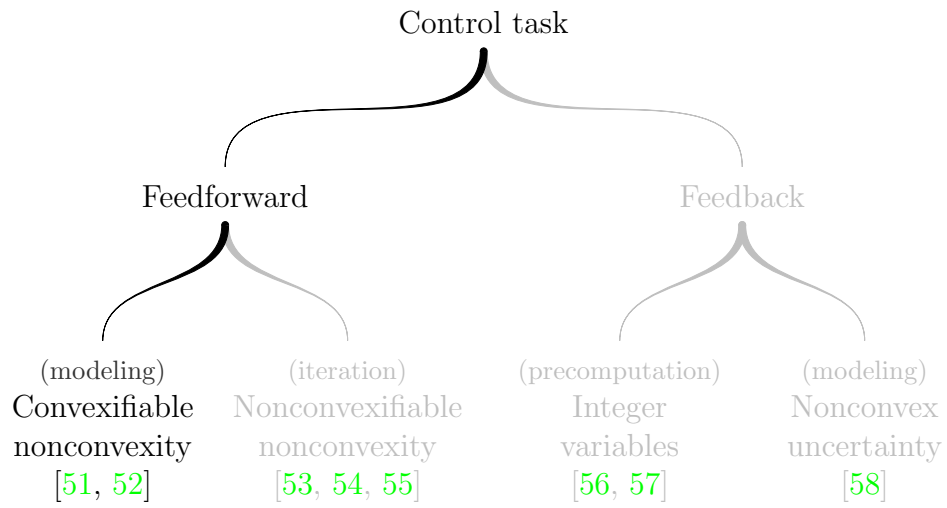


Figure 6.1: This chapter considers the modeling approach for provable real-time optimization-based trajectory generation with guaranteed convergence and global optimality.

to find a solution or finds it too slowly for practical application.

Within the context of real-time optimization-based control algorithm development, the approach of this chapter concerns feedforward control and takes the modeling avenue in which the original problem's non-convexity is completely removed for on-board implementation via guarantees provided by optimal control theory (see Section 2.3 for an introduction). Figure 6.1 situates the algorithm within the research branches of Figure 1.1. Particular thanks go to Michael Szmuk and Taylor P. Reynolds for their technical insight and helpful suggestions.

6.1 Introduction

This chapter is all about developing a *fully convex* programming solution to a class mixed-integer (in other words, "strongly" non-convex[1]) optimal control problems. In this context,

[1]This is an informal statement to emphasize the core difficulty: it is not that some continuous function or set is non-convex, but that fundamentally part of the decision space is discrete. Traditionally, this requires an entirely different class of *integer* optimization algorithms (based on the branch-and-bound method [164]).

fully convex means that the original non-convex problem is converted into an entirely convex relaxed problem. This latter problem is then solved in a one-shot convex optimization to recover the globally optimal solution of the original problem. It is the *one-shot* nature of the approach that distinguishes it from the iterative "sequential convex programming" methods of Chapter 7.

The mixed-integer optimal control problems that we consider have semi-continuous control input norms. Semi-continuous variables are a particular type of binary non-convexity.

Definition 12. Variable $x \in \mathbb{R}$ is *semi-continuous* if $x \in \{0\} \cup [a, b]$ with $0 < a \leq b$ [165].

The constraint $az \leq x \leq bz$ with $z \in \{0, 1\}$ models semi-continuity. We consider systems that have multiple inputs which may not all be simultaneously active, which point in dissimilar directions in the input space, and whose norms are semi-continuous. Although mixed-integer convex programming is applicable, it is an $\mathcal{NP}$-hard optimization class [166, 167] and solving a practical path planning problem such as rocket landing or spacecraft rendezvous can take hours. This paper proposes an algorithm based on lossless convexification that solves these problems to global optimality in seconds.

6.2 Problem Statement

This section presents the class of optimal control problems that is to be solved via convex optimization by our method. We consider mixed-integer non-convex optimal control problems with linear time-invariant (LTI) dynamics and semi-continuous input norms:

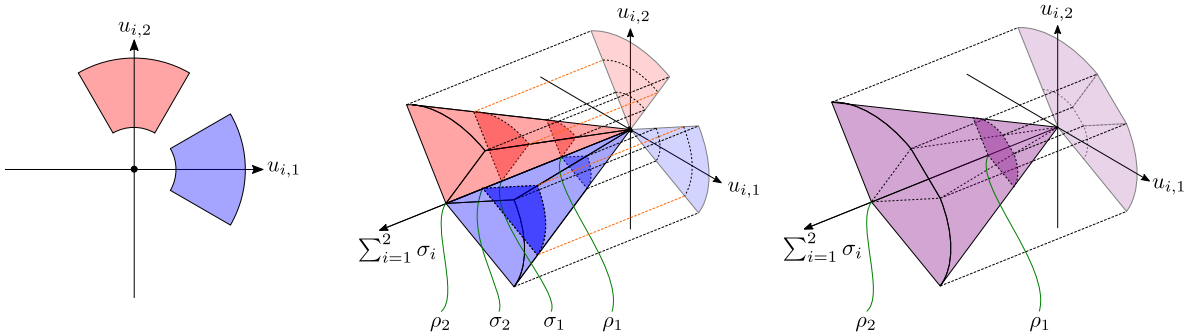
$$\min_{u_i, \gamma_i, t_f} \; m(t_f, x(t_f)) + \int_0^{t_f} \ell(x(t)) + \zeta \textstyle\sum_{i=1}^{M} \|u_i(t)\|_2 \mathrm{d}t \tag{6.1a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + B \textstyle\sum_{i=1}^{M} u_i(t) + w, \;\; x(0) = x_0, \tag{6.1b}$$

$$\gamma_i(t)\rho_1^i \leq \|u_i(t)\|_2 \leq \gamma_i(t)\rho_2^i \quad i = 1, \ldots, M, \tag{6.1c}$$

$$\gamma_i(t) \in \{0, 1\} \quad i = 1, \ldots, M, \tag{6.1d}$$

$$\textstyle\sum_{i=1}^{M} \gamma_i(t) \leq K, \tag{6.1e}$$



(a) Original non-convex disjoint input sets defined by (6.3c)-(6.3f).

(b) The non-convexity is removed by relaxing (6.3c) to (6.4c)-(6.4d).

(c) The mutual exclusivity is removed by relaxing (6.3d) to (6.4e).

Figure 6.2: Illustration of the convex relaxation steps for the restricted Problem 6.3, here shown for $M = 2$, $K = 1$ and $m = 2$.

$$C_i u_i(t) \leq 0 \quad i = 1, \ldots, M, \tag{6.1f}$$

$$x(t) \in \mathcal{X}, \tag{6.1g}$$

$$b(x(t_f)) = 0. \tag{6.1h}$$

In Problem 6.1, $x(t) \in \mathbb{R}^n$ is the state, $u_i(t) \in \mathbb{R}^m$ is the i-th input, and $w \in \mathbb{R}^n$ is a known external input. Convex functions $m : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$, $\ell : \mathbb{R}^n \to \mathbb{R}$ and $b : \mathbb{R}^n \to \mathbb{R}^{n_b}$ define the terminal cost, the state running cost and the terminal manifold respectively. The binary coefficient $\zeta \in \{0, 1\}$ toggles the input running cost. The state must lie in the convex set $\mathcal{X} \subseteq \mathbb{R}^n$. The input directions are constrained to polytopic cones called *input pointing sets*:

$$\mathcal{U}_i \triangleq \{u \in \mathbb{R}^m : C_i u \leq 0\}, \tag{6.2}$$

where $C_i \in \mathbb{R}^{p_i \times m}$ is a matrix with $C_{i,j} \in \mathbb{R}^m$ the j-th row. The problem statement satisfies the following assumption.

Assumption 6. The control norm bounds in (6.1c) are distinct, i.e. $\rho_1^i < \rho_2^i$.

For clarity of presentation as well as to follow the development history of this extension to lossless convexification [51, 52], we will first consider a simpler "restriction" (in other words, a special case) of Problem 6.1, given by the following problem.

$$\min_{u_i,\gamma_i,t_f} \; m(t_f, x(t_f)) \tag{6.3a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + B\textstyle\sum_{i=1}^{M} u_i(t) + w, \;\; x(0) = x_0, \tag{6.3b}$$

$$\gamma_i(t)\rho_1 \leq \|u_i(t)\|_2 \leq \gamma_i(t)\rho_2 \quad i = 1, \ldots, M, \tag{6.3c}$$

$$\gamma_i(t) \in \{0, 1\} \quad i = 1, \ldots, M, \tag{6.3d}$$

$$\textstyle\sum_{i=1}^{M} \gamma_i(t) \leq K, \tag{6.3e}$$

$$C_i u_i(t) \leq 0 \quad i = 1, \ldots, M, \tag{6.3f}$$

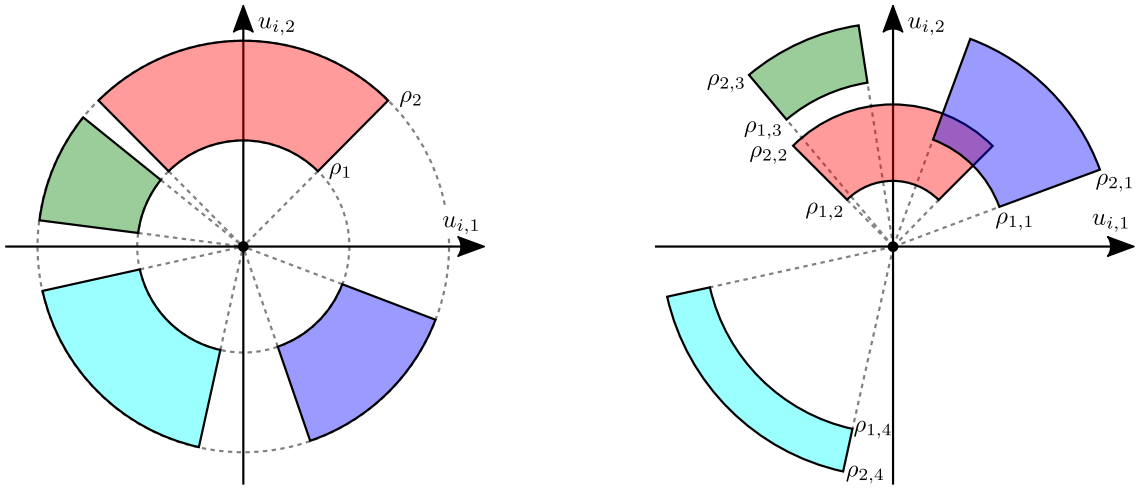
$$b(t_f, x(t_f)) = 0, \tag{6.3g}$$

Problem 6.3 is simpler than Problem 6.1 in the following ways. First, there are no input and state integral costs (compare (6.3a) to (6.1a)). Second, the input norm bounds are the same for each input (compare (6.3c) to (6.1c)). Third, the state is unconstrained (constraint (6.1g) is missing). Last and most important, Problem 6.3 makes the following additional assumption which prevents the input pointing sets from overlapping. Figure 6.3 shows how this enables richer input set geometry in Problem 6.1.

Assumption 7. The pointing set interiors in Problem 6.3 do not overlap, i.e. $\operatorname{int}\mathcal{U}_i \cap \operatorname{int}\mathcal{U}_j = \emptyset$ for all $i \neq j$.

Although Problem 6.1 contains[2] Problem 6.3, discussing the latter problem first makes for a gentler and more intuitive explanation of the algorithm. Thus, the next section states the lossless convexification result for Problem 6.3. The result is then generalized to Problem 6.1 in Section 6.4, and proved in Section 6.5.

[2] A minor technical detail, of little practical significance, is that (6.3g) may depend on the final time t_f, whereas (6.1h) cannot.



(a) Disjoint same-bound input sets allowed by Problem 6.3 [51].

(b) Overlapping multiple-bound input sets allowed by Problem 6.1 [52].

Figure 6.3: Input set example for $m = 2$. As shown in (a), Problem 6.3 does not allow different input norm bounds or overlapping input pointing sets. As shown in (b), Problem 6.1 allows both features.

6.3 Lossless Convexification for the Restricted Problem

This section describes lossless convexification for the restricted Problem 6.3. The main result is Theorem 16, which states that Problem 6.3 under certain conditions is solved via convex optimization by Problem 6.4. Note that, like the problem, the theorem is a restricted statement of the two theorems in the next section. As a result, the primary aim of this section is to elucidate the idea of lossless convexification on a simpler problem, and the theorem shall be proved in the more general context of Problem 6.1. The interested reader may refer to the original publication for the specific proof of Theorem 16 [51].

Problem 6.3 is non-convex due to the input norm lower-bound in (6.3c) and is mixed-integer due to (6.3d). This is made clear in Figure 6.2a, where an example in $\mathbb{R}^2$ for two inputs shows how the input set non-convex and disjoint. Consider the following convex relaxation of Problem 6.3:

$$\min_{u_i,\gamma_i,\sigma_i,t_f} m(t_f, x(t_f)) \tag{6.4a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + B\sum_{i=1}^{M} u_i(t) + w,\ x(0) = x_0, \tag{6.4b}$$

$$\gamma_i(t)\rho_1 \leq \sigma_i(t) \leq \gamma_i(t)\rho_2 \quad i = 1, \ldots, M, \tag{6.4c}$$

$$\|u_i(t)\|_2 \leq \sigma_i(t) \quad i = 1, \ldots, M, \tag{6.4d}$$

$$0 \leq \gamma_i(t) \leq 1 \quad i = 1, \ldots, M, \tag{6.4e}$$

$$\sum_{i=1}^{M} \gamma_i(t) \leq K, \tag{6.4f}$$

$$C_i u_i(t) \leq 0 \quad i = 1, \ldots, M, \tag{6.4g}$$

$$b(t_f, x(t_f)) = 0. \tag{6.4h}$$

The action of the convexified constraints (6.4c)-(6.4e) on the original problem set is illustrated for a simple example in Figure 6.2a. The relaxation consists of three steps. First, by relaxing (6.3c) to (6.4c)-(6.4d), individual input sets are convexified to 3D slices (Figure 6.2b). Next by relaxing (6.3d) to (6.4e), a convex hull is obtained (Figure 6.2c). Take note of that the constraint (6.4d) is the familiar LCvx equality constraint from Chapter 5.

By replacing (6.3c)-(6.3d) with (6.4c)-(6.4e), the input set of Problem 6.4 becomes the convex hull of the Minkowski sums of every combination of at most K of the relaxed individual input sets of Problem 6.3. Figure 6.4 shows the case of $M = 3$ and $K = 2$, where the input set is projected onto the u_i space. It will be shown in Section 6.5 that the optimal solution is extremal, hence it will take values among the extreme points of the input set of Problem 6.4 with at most K inputs active.

Consider the following conditions, which are sufficient to eliminate degenerate optimal solutions of Problem 6.4 that may be infeasible for Problem 6.3. To state the conditions, define an *adjoint system* whose output $y(t) \in \mathbb{R}^m$ is called the *primer vector*:

$$\dot{\lambda}(t) = -A^{\intercal}\lambda(t), \tag{6.5a}$$

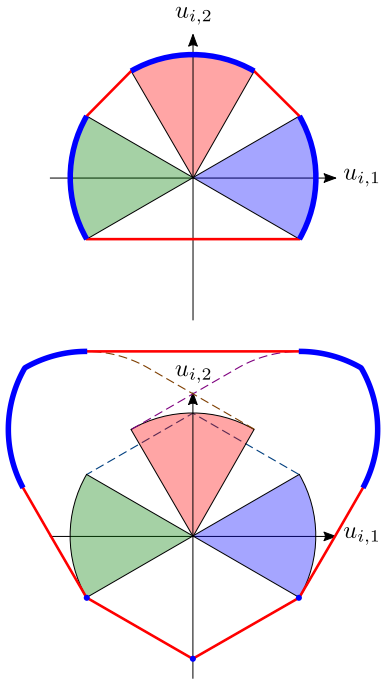


Figure 6.4: The relaxed input set of Problem 6.4 is the convex hull of the Minkowski sums for every combination of K or fewer of the individual input sets (each one relaxed via (6.4c)-(6.4e)). For $M = 3$, $K = 2$ and $m = 2$, (left) shows the case of one- and (right) shows the case of two-input set combinations, with the relaxed set shown in bold red and the boundaries of the constituent Minkowski sums shown as dashed lines. The optimal solution takes values from the extreme points, shown as blue segments. The origin is also an extreme point, corresponding to a combination of zero input sets (not shown).

$$y(t) = B^{\mathsf{T}}\lambda(t). \tag{6.5b}$$

A generalized version of (6.5) will be seen in the next section, and its origin shall become clear from the lossless convexification proof in Section 6.5. Essentially, we are interested in "how much" $y(t)$ projects onto the i-th input pointing set. This is given by the following input *gain* measure:

$$\Gamma_i(t) \triangleq \operatorname{proj}(U_i)\, y(t).$$

Condition 8. The adjoint system (6.5) is observable.

Condition 9. The adjoint system (6.5) and pointing cone geometry (6.3f) satisfy either:

(a) $\Gamma_i(t) \neq 0$ a.e. $[0, t_f]$ $\forall i$ s.t. $y(t) \notin \operatorname{int} \mathcal{N}_{\mathcal{U}_i}(0)$;

(b) on any interval where $\Gamma_i(t) = 0$, $\Gamma_j(t) > 0$ for at least K other inputs.

Condition 10. The adjoint system (6.5) and pointing cone geometry (6.3f) satisfy either:

(a) $\Gamma_i(t) \neq \Gamma_j(t)$ a.e. $[0, t_f]$ $\forall i$ s.t. $y(t) \notin \operatorname{int} \mathcal{N}_{\mathcal{U}_i}(0)$;

(b) on any interval where $\Gamma_i(t) = \Gamma_j(t)$, there exist K inputs with $\Gamma_k(t) > \Gamma_i(t)$ or $M - K$ inputs where $\Gamma_k(t) < \Gamma_i(t)$.

Condition 11. The following intersection holds:

$$\operatorname{range} \begin{bmatrix} \nabla_x b[t_f]^\intercal \\ \nabla_t b[t_f]^\intercal \end{bmatrix} \cap \operatorname{cone} \begin{bmatrix} \nabla_x m[t_f]^\intercal \\ \nabla_t m[t_f] \end{bmatrix} = \{0\}.$$

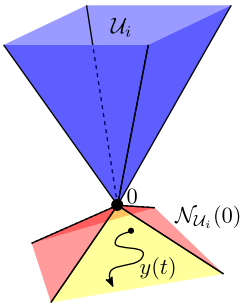
We now state the main lossless convexification result for the restricted Problem 6.3, which claims that Problem 6.4 solves Problem 6.3 under the above conditions. The next section presents a generalized version of this theorem, applied to the broader class of problems defined by Problem 6.1.

Theorem 16. *The solution of Problem 6.4 is globally optimal a.e.* $[0, t_f]$ *for Problem 6.3 if Conditions 8-11 hold.*

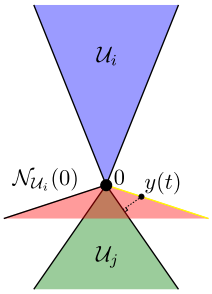
6.3.1 Discussion of Conditions 8-11 and Special Cases

This section describes situations when Conditions 8-11 are easy to verify. First, Condition 8 is readily verified by checking if the pair $\{-A^\intercal, B^\intercal\}$ in (6.5) is observable [114].

Condition 11 originates from the maximum principle transversality requirement (2.14). Figure 6.6a illustrates the general requirement. As shown in Figure 6.6b, the condition is satisfied for a minimum-time problem with a fixed or (partially) free terminal state. As

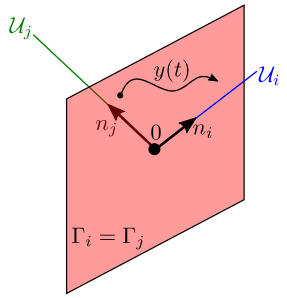


(B.1)

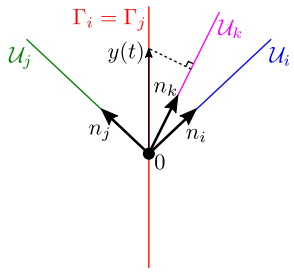


(B.2)

(a) When case (a) of Condition 9 fails, $y(t)$ can evolve on the normal cone boundary (A.1). Case (b) then holds if $y(t) \in \partial\mathcal{N}_{\mathcal{U}_i}(0)$ projects positively onto at least K other input pointing sets. For $K = 1$, (A.2) illustrates a case where $y(t)$ projects positively onto $\mathcal{U}_j$.



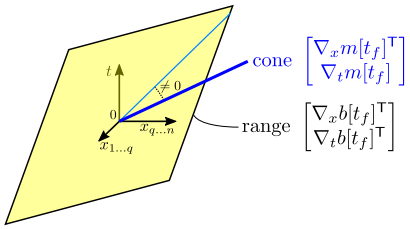
(A.1)



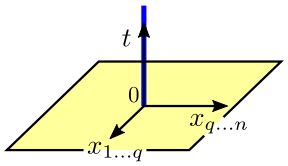
(A.2)

(b) When case (a) of Condition 10 fails, $y(t)$ can evolve on the manifold $\Gamma_i(t) = \Gamma_j(t)$ as shown in (B.1) for ray cones. Case (b) then holds if, for example, $y(t)$ projects more positively onto K other cones. For $K = 1$, (B.2) shows a situation where $y(t)$ projects more positively onto $\mathcal{U}_k$.

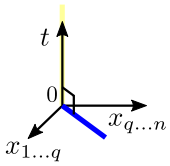
Figure 6.5: Visualization of Condition 9 and Condition 10 when their case (a) fails. In such circumstances, the conditions can nevertheless hold given the right input pointing set geometry.



(a) General Condition 11 requirement.



(b) Minimum-time Condition 11 requirement.



(c) Minimum-error Condition 11 requirement.

Figure 6.6: Condition 11 for the general and two special cases.

shown in Figure 6.6c, the condition is also satisfied for a fixed- or free-time problem with a (partially) penalized terminal state.

Conditions 9 and 10 are illustrated in Figure 6.5. Both conditions can be checked via matrix algebra in the special case of *ray cones*, in other words cones of the form $\mathcal{U}_i = \operatorname{cone} n_i$ for some *direction* $n_i \in \mathbb{R}^m$. In this case, $\partial \mathcal{N}_{\mathcal{U}_i}(0) = \{u \in \mathbb{R}^m : n_i^\intercal u = 0\}$ which is a hyperplane. Consider the adjoint system (6.5) with the "projected" primer vector:

$$n_i^\intercal y(t) = (Bn_i)^\intercal \lambda(t).$$

If the pair $\{-A^\intercal, (Bn_i)^\intercal\}$ is observable then Condition 9 case (a) holds. If not, let $\mathcal{V}_i$ be the unobservable subspace and consider the special case $\operatorname{range} B^\intercal(-A^\intercal)^k \mathcal{V}_i \subseteq \operatorname{range} z_i$ $\forall k = 0, \ldots, n-1$ for some $z_i \in \mathbb{R}^m$. If $\operatorname{proj}(U_k) z_i > 0$ and $\operatorname{proj}(U_k) -z_i > 0$ for K or more input pointing sets, then Condition 9 case (b) holds. Similarly, for Condition 10 we consider the projected primer vector:

$$(n_i - n_j)^\intercal y(t) = \big(B(n_i - n_j)\big)^\intercal \lambda(t).$$

If the pair $\{-A^\intercal, \big(B(n_i - n_j)\big)^\intercal\}$ is observable then Condition 10 case (a) holds. If not, let $z_i \in \mathbb{R}^m$ be defined as before with $\mathcal{V}_i$ the unobservable subspace for this new system. If $\operatorname{proj}(U_k) z_i > \operatorname{proj}(U_i) z_i$ for K other inputs or $\operatorname{proj}(U_k) z_i < \operatorname{proj}(U_i) z_i$ for $M - K$ other

inputs, then Condition 10 case (b) holds.

6.4 Lossless Convexification for the Full Problem

We now address lossless convexification for the general Problem 6.1. At the end of the section, two main results are gives in Theorems 17 and 18, for the free-state and state-constrained cases respectively. The latter result is reminiscent of [59] in the sense of the state constraint being permitted to activate only at discrete time instances (in other words, never on a continuous time interval).

The input magnitude in Problem 6.1 is semi-continuous, i.e. $\|u_i(t)\|_2 \in \{0\} \cup [\rho_1^i, \rho_2^i]$. This makes the problem mixed-integer and non-convex, which is readily apparent from Figure 6.3. Consider the following convex relaxation:

$$\min_{u_i,\gamma_i,\sigma_i,t_f} m(t_f, x(t_f)) + \zeta\xi(t_f) + \int_0^{t_f} \ell(x(t))\mathrm{d}t \tag{6.6a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + B\textstyle\sum_{i=1}^M u_i(t) + w,\ x(0) = x_0, \tag{6.6b}$$

$$\dot{\xi}(t) = \textstyle\sum_{i=1}^M \sigma_i(t), \tag{6.6c}$$

$$\gamma_i(t)\rho_1^i \le \sigma_i(t) \le \gamma_i(t)\rho_2^i \quad i = 1, \dots, M, \tag{6.6d}$$

$$\|u_i(t)\|_2 \le \sigma_i(t) \quad i = 1, \dots, M, \tag{6.6e}$$

$$0 \le \gamma_i(t) \le 1 \quad i = 1, \dots, M, \tag{6.6f}$$

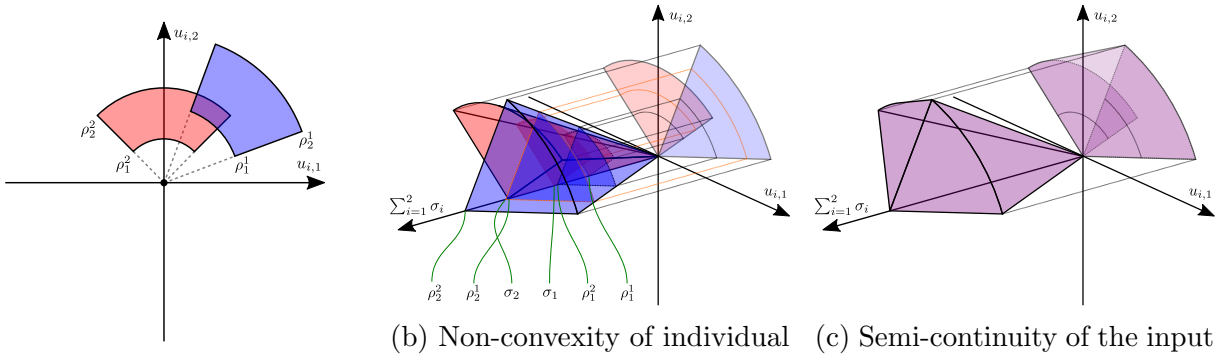
$$\textstyle\sum_{i=1}^M \gamma_i(t) \le K, \tag{6.6g}$$

$$C_i u_i(t) \le 0 \quad i = 1, \dots, M, \tag{6.6h}$$

$$x(t) \in \mathcal{X}, \tag{6.6i}$$

$$b(x(t_f)) = 0. \tag{6.6j}$$

Replacing (6.1c)-(6.1d) with (6.6d)-(6.6f) convexifies the input set of Problem 6.1. Figure 6.7 illustrates an example. The relaxation consists of two steps. First (Figure 6.7b), by relaxing (6.1c) to (6.6d)-(6.6e), individual input sets are convexified. Second (Figure 6.7c),



(a) Original non-convex input set defined by (6.1c)-(6.1f).

(b) Non-convexity of individual input sets is removed by relaxing (6.1c) to (6.6d)-(6.6e).

(c) Semi-continuity of the input norm is convexified by relaxing (6.1d) to (6.6f).

Figure 6.7: Illustration of the convex relaxation steps for the general Problem 6.1, here shown for $M = 2$, $K = 1$ and $m = 2$.

by relaxing (6.1d) to (6.6f), a convex hull is obtained.

We now state conditions which remove degenerate solutions of Problem 6.6 that may be infeasible for Problem 6.1. The conditions depend once again on the following generalization of the aforementioned adjoint system (6.7):

$$\dot{\lambda}(t) = -A^{\mathsf{T}}\lambda(t) + v(t), \;\; v(t) \in \partial\ell(x(t))^{\mathsf{T}}, \tag{6.7a}$$

$$y(t) = B^{\mathsf{T}}\lambda(t), \tag{6.7b}$$

where we note that the primer vector is unchanged. It will be seen in Section 6.5 that we are interested in "how much" $y(t)$ projects onto the i-th input pointing set. The measure of projection amount is given by the following input gain measure, which is again a generalization of (6.8):

$$\Gamma_i(t) \triangleq (\text{proj}\,(U_i)\, y(t) - \zeta)\rho_2^i. \tag{6.8}$$

Condition 12. The adjoint system (6.7) is strongly observable [150].

Condition 13. The adjoint system (6.7) and pointing cone geometry (6.1f) satisfy either:

(a) $\Gamma_i(t) \neq 0$ a.e. $[0, t_f]$ $\forall i$ s.t. $y(t) \notin \operatorname{int} \mathcal{N}_{\mathcal{U}_i}(0)$;

(b) on any interval where $\Gamma_i(t) = 0$, $\Gamma_j(t) > 0$ for at least K other inputs.

Condition 14. The adjoint system (6.7) and pointing cone geometry (6.1f) satisfy either:

(a) $\Gamma_i(t) \neq \Gamma_j(t)$ a.e. $[0, t_f]$ $\forall i$ s.t. $y(t) \notin \operatorname{int} \mathcal{N}_{\mathcal{U}_i}(0)$;

(b) on any interval where $\Gamma_i(t) = \Gamma_j(t)$, there exist K inputs with $\Gamma_k(t) > \Gamma_i(t)$ or $M - K$ inputs where $\Gamma_k(t) < \Gamma_i(t)$.

Condition 15. The following equation holds:

$$\ell[t] + \zeta \sum_{i=1}^{M} \sigma_i(t) + \nabla_t m[t_f] \neq 0 \ \forall t \in [0, t_f). \tag{6.9}$$

We now state the two main lossless convexification results, which claim that Problem 6.6 solves Problem 6.1 under the above conditions. The theorems are proved in the next section.

Theorem 17. *The solution of Problem 6.6 is globally optimal a.e.* $[0, t_f]$ *for Problem 6.1 if Conditions 12-15 hold and the state constraint* (6.1g) *is never activated.*

Theorem 18. *The solution of Problem 6.6 is globally optimal a.e.* $[0, t_f]$ *for Problem 6.1 if Conditions 12-15 hold and the state constraint* (6.1g) *is activated at discrete times.*

6.4.1 Discussion of Condition 12 and Strong Observability

This section describes Condition 12 and its verification. Strong observability extends the concept of observability to the case of non-zero inputs. A strongly observable system does not have transmission zeroes. To be precise, let us state strong observability in the context of (6.7).

Definition 13. (*[150, Definition 7.8]*) A point $\lambda_0 \in \mathbb{R}^n$ is *weakly unobservable* if there exists an interval $\mathcal{T} = [\tau_1, \tau_2]$ and an input trajectory $v(t) \in \partial \ell[t]^\intercal$ for $t \in \mathcal{T}$ such that if $\lambda(\tau_1) = \lambda_0$ then the primer vector satisfies $y(t) = 0$ $\forall t \in \mathcal{T}$. The set of all weakly unobservable points is denoted $\mathcal{V}$, which is called the weakly unobservable set.

Theorem 19. *([150, Theorem 7.16]) The adjoint system (6.7) is strongly observable if* $\mathcal{V} = \{0\}$.

To verify Condition 12 via simple matrix algebra, it is sufficient to apply the algorithm for computing $\mathcal{V}$ in [150, Section 7.3] using the following alternative to (6.7a):

$$\dot{\lambda}(t) = -A^{\mathsf{T}}\lambda(t) + Dv(t), \tag{6.10}$$

where $v(t) \in \mathbb{R}^n$ and range $D = \text{span} \bigcup_{t\in[\tau_1,\tau_2]} \partial\ell(x(t))^{\mathsf{T}}$. This conservative approximation assumes that the input can come from a subspace spanned by the subdifferentials. Section 6.6.2 uses this approximation to verify Conditions 12 and 14 for the rocket landing problem.

6.5 Lossless Convexification Proof

This section proves Theorems 17 and 18. The general outline is as follows. We first prove Theorem 17 by showing that (step 1) the solution of Problem 6.6 is feasible for Problem 6.1, and (step 2) the solution is globally optimal. We then show Theorem 18 via a proof by contradiction in which Theorem 17 is applied on each interval where the state constraint is inactive.

Lemma 6. *The solution of Problem 6.6 is feasible a.e.* $[0, t_f]$ *for Problem 6.1 if* $x(t) \in \text{int}\,\mathcal{X}$ *and Conditions Conditions 12-15 hold.*

Proof. The proof uses the maximum principle from Theorem 1. Since there are two states, partition the adjoint variable as $\psi(t) = (\lambda(t) \in \mathbb{R}^n, \eta(t) \in \mathbb{R})$. For Problem 6.6 and $x(t) \in \text{int}\,\mathcal{X}$, the adjoint and Hamiltonian dynamics follow from (2.13b) and (2.13c):

$$\dot{\lambda}(t) = -A^{\mathsf{T}}\lambda(t) - \alpha v(t),\ v(t) \in \partial\ell[t]^{\mathsf{T}},\ \text{a.e. } [0, t_f], \tag{6.11a}$$

$$\dot{\eta}(t) = 0 \text{ a.e. } [0, t_f], \tag{6.11b}$$

$$\dot{H}[t] = 0 \text{ a.e. } [0, t_f], \tag{6.11c}$$

Using the subdifferential basic chain rule [78, Theorem 10.6], the transversality condition (2.14) yields:

$$\lambda(t_f) = \nabla_x m[t_f]^\intercal \alpha + \nabla_x b[t_f]^\intercal \beta, \tag{6.12a}$$

$$\eta(t_f) = \alpha\zeta, \tag{6.12b}$$

$$H[t_f] = -\nabla_t m[t_f]\alpha, \tag{6.12c}$$

for some $\beta \in \mathbb{R}^{n_b}$. Due to (6.11b)-(6.11c), (6.12b)-(6.12c) and absolute continuity, we have [168, Theorem 9]:

$$\eta(t) = \alpha\zeta, \ \forall t \in [0, t_f], \tag{6.13a}$$

$$H[t] = -\nabla_t m[t_f]\alpha, \ \forall t \in [0, t_f]. \tag{6.13b}$$

We claim that the primer vector $y(t) \neq 0$ a.e. $[0, t_f]$. By contradiction, suppose there exists an interval $[\tau_1, \tau_2] \subseteq [0, t_f]$ for which $y(t) = 0$. Condition 12 implies that $\lambda(\tau_1) = 0$. Due to (6.13), this implies $\alpha(\ell[\tau_1] + \zeta \sum_{i=1}^M \sigma_i(\tau_1) + \nabla_t m[t_f]) = 0$. Due to Condition 15, it must be that $\alpha = 0$ which implies $(\alpha, \psi(\tau_1)) = 0$. Since this violates non-triviality (2.11), it must be that $y(t) \neq 0$ a.e. $[0, t_f]$. Having eliminated the pathological case, assume $\alpha < 0$. In particular, since the necessary conditions in Theorem 1 are scale-invariant, we can set $\alpha = -1$ without loss of generality. The pointwise maximum condition (2.12) implies that the following must hold a.e. $[0, t_f]$:

$$\underset{u_i, \gamma_i, \sigma_i}{\text{argmax}} \ \sum_{i=1}^M y(t)^\intercal u_i(t) - \zeta\sigma_i(t) \tag{6.14a}$$

$$\text{s.t. constraints (6.6d)-(6.6h) hold.} \tag{6.14b}$$

We shall now analyze the optimality conditions of Problem 6.14. For concise notation, the time argument t shall be omitted. Problem 6.14 as a minimization and treating constraints

(6.6f) and (6.6g) implicitly, we can write the Lagrangian of Problem 6.14 [34]:

$$\mathcal{L}(u_i, \gamma_i, \sigma_i, \lambda^i_{1\ldots4}) = \textstyle\sum_{i=1}^{M} \zeta\sigma_i - y^\intercal u_i + \lambda^i_1(\|u_i\|2 - \sigma_i) + \\ \lambda^i_2(\gamma_i\rho^i_1 - \sigma_i) + \lambda^i_3(\sigma_i - \gamma_i\rho^i_2) + {\lambda^i_4}^\intercal C_i u_i, \quad (6.15)$$

where $\lambda^i_j \geq 0$ are Lagrange multipliers satisfying the following complementarity conditions:

$$\lambda^i_1(\|u_i\|_2 - \sigma_i) = 0, \quad (6.16a)$$
$$\lambda^i_2(\gamma_i\rho^i_1 - \sigma_i) = 0, \quad (6.16b)$$
$$\lambda^i_3(\sigma_i - \gamma_i\rho^i_2) = 0, \quad (6.16c)$$
$$\lambda^i_4 \circ C_i u_i = 0. \quad (6.16d)$$

Next, the Lagrange dual function is given by:

$$\begin{aligned} g(\lambda^i_{1\ldots4}) &= \inf_{u_i,\gamma_i,\sigma_i} \mathcal{L}(u_i, \gamma_i, \sigma_i, \lambda^i_{1\ldots4}) \\ &= \textstyle\sum_{i=1}^{M} \inf_{\sigma_i} [(\zeta + \lambda^i_3 - \lambda^i_2 - \lambda^i_1)\sigma_i] - \\ &\quad \textstyle\sum_{i=1}^{M} \sup_{u_i} [(y - C_i^\intercal \lambda^i_4)^\intercal u_i - \lambda^i_1\|u_i\|_2] + \\ &\quad \inf_{(6.6f),(6.6g)} \textstyle\sum_{i=1}^{M} (\lambda^i_2\rho^i_1 - \lambda^i_3\rho^i_2)\gamma_i. \end{aligned} \quad (6.17)$$

The dual function bounds the primal optimal cost from above. A non-trivial upper-bound requires:

$$\|y - C_i^\intercal \lambda^i_4\|_2 \leq \lambda^i_1, \quad (6.18a)$$
$$\zeta + \lambda^i_3 - \lambda^i_2 - \lambda^i_1 = 0, \quad (6.18b)$$

where the first inequality is akin to the $\|\cdot\|_2$ conjugate function [34, Example 3.26]. However, note that if (6.18a) is strict then $\|u_i\|_2 = 0$ is optimal, which is trivially feasible for Problem 6.1. Substituting (6.18b) into (6.18a) gives the following condition for non-trivial

solutions:

$$\|y - C_i^\intercal \lambda_4^i\|_2 = \zeta + \lambda_3^i - \lambda_2^i. \tag{6.19}$$

Further simplification is possible by recognizing that a non-trivial solution implies $\gamma_i > 0$. By Assumption 6, (6.16b) and (6.16c), this means $\lambda_2^i > 0$ and $\lambda_3^i > 0$ cannot occur simultaneously. Furthemore, (6.17) reveals that $\gamma_i > 0$ is not sub-optimal if and only if $\lambda_2^i \rho_1^i - \lambda_3^i \rho_2^i \leq 0$. By this reasoning, $\lambda_2^i = 0$ and $\lambda_3^i \geq 0$ are necessary for optimality. Thus, (6.19) simplifies to:

$$\|y - C_i^\intercal \lambda_4^i\|_2 = \zeta + \lambda_3^i. \tag{6.20}$$

Next, note that at optimality the left-hand side of (6.20) equals the Eucledian projection onto $\mathcal{U}_i$, i.e. $\|y - C_i^\intercal \lambda_4^i\|_2 = \text{proj}\,(U_i)\, y$. This can be shown by contradiction using (6.16d) and that $u_i = \|u_i\|_2 (y - C_i^\intercal \lambda_4^i)/\|y - C_i^\intercal \lambda_4^i\|_2$ in (6.17). Note that the degenerate case of $u_i \neq 0$ and $\|y - C_i^\intercal \lambda_4^i\|_2 = 0$ is eliminated in the discussion below which leverages Condition 13. Thus (6.20) simplifies to the following relationship, which we call the *characteristic equation* of non-trivial solutions to Problem 6.6:

$$\text{proj}\,(U_i)\, y = \zeta + \lambda_3^i. \tag{6.21}$$

Substituting (6.21) into (6.17) yields:

$$g(\lambda_{1\ldots4}^i) = -\sup_{(6.6f),(6.6g)} \textstyle\sum_{i=1}^{K'} (\text{proj}\,(U_i)\, y - \zeta) \rho_2^i \gamma_i, \tag{6.22}$$

where we assume that the characteristic equation (6.21) does not hold for $i = K'+1, \ldots, M$ such that $\gamma_{i>K'} = 0$. To facilitate discussion, define the i-th input *gain* as in (6.8). Note that $\Gamma_i \geq 0$ due to (6.21). Thus (6.22) becomes:

$$g(\lambda_{1\ldots4}^i) = -\sup_{(6.6f),(6.6g)} \textstyle\sum_{i=1}^{K'} \Gamma_i \gamma_i. \tag{6.23}$$

Without loss of generality, assume a descending ordering $\Gamma_i \geq \Gamma_j$ for $i > j$. Let $K'' \triangleq$

$\min\{K, K'\}$. By inspection of (6.23), the condition:

$$\Gamma_{K''} > 0 \ \wedge \ \Gamma_{K''} > \Gamma_{K''+1}, \tag{6.24}$$

is sufficient to ensure that it is optimal to set

$$\gamma_i = \begin{cases} 1 & \text{if } i \leq K'', \\ 0 & \text{otherwise.} \end{cases} \tag{6.25}$$

The lemma holds if (6.24) holds a.e. $[0, t_f]$. This is assured by Conditions 13 and 14. Condition 13 case (a) assures $\Gamma_{K''} > 0$ a.e. $[0, t_f]$. If on some interval $\Gamma_k = 0$, Condition 13 case (b) assures that $k > K''$. If $K'' < K$ then due to $\Gamma_{K''} > 0$ and the definition of K', it must be that $\Gamma_{K''+1} = 0 \Rightarrow \Gamma_{K''} > \Gamma_{K''+1}$. Else if $K'' = K$, Condition 14 case (a) assures $\Gamma_K > \Gamma_{K+1}$ a.e. $[0, t_f]$. If on some interval $\Gamma_k = \Gamma_{k+1}$, Condition 14 case (b) assures that $k \neq K$.

Thus, (6.24) holds a.e. $[0, t_f]$ and the lemma is proved. From (6.25), the structure of the optimal solution is **bang-bang with at most K inputs active** a.e. $[0, t_f]$. □

Lemma 6 guarantees that Problem 6.6 produces a feasible solution of Problem 6.1. We will now show that this solution is globally optimal, thus proving Theorem 17.

Proof of Theorem 17. The solution of Problem 6.6 is feasible a.e. $[0, t_f]$ for Problem 6.1 due to Lemma 6. Furthermore, if $\zeta = 0$ then the cost functions of Problems 6.1 and 6.6 are the same. This is also true when $\zeta = 1$ because Lemma 6 guarantees that $\|u_i(t)\|_2 = \sigma_i(t)$. The optimal costs thus satisfy $J^*_{\mathcal{O}} \leq J^*_{\mathcal{R}}$. However, any solution of Problem 6.1 is feasible for Problem 6.6 by setting $\sigma_i(t) = \|u_i(t)\|_2$, thus $J^*_{\mathcal{R}} \leq J^*_{\mathcal{O}}$. Therefore $J^*_{\mathcal{R}} = J^*_{\mathcal{O}}$ so the Problem 6.6 solution is globally optimal for Problem 6.1 a.e. $[0, t_f]$. □

Theorem 17 implies that Problem 6.1 is solved in polynomial time by an SOCP solver applied to Problem 6.6. This can be done efficiently with several numerically reliable SOCP

solvers [107]. Therefore the class of $\mathcal{NP}$-hard problems defined by Problem 6.1 is in fact of $\mathcal{P}$ complexity if $x(t) \in \operatorname{int} \mathcal{X}$ and Conditions 12-15 hold.

6.5.1 The Case of Active State Constraints

So far it has been assumed that the state constraint (6.1g) is inactive. This section guarantees lossless convexification in a limited setting when (6.1g) is activated at a discrete set of times. To begin, define the *interior time* and *contact time* sets as follows:

$$\mathcal{T}_i \triangleq \{t \in (0, t_f) : x(t) \in \operatorname{int} \mathcal{X}\}, \tag{6.26a}$$

$$\mathcal{T}_c \triangleq [0, t_f] \setminus \mathcal{T}_i. \tag{6.26b}$$

A point τ of $\mathcal{T}_c$ is called an *isolated point* if there exists a neighborhood of τ not containing other points of $\mathcal{T}_c$ [169]. A set of isolated points is called a *discrete set* and any discrete subset of a Eucledian space has measure zero [59]. We can now prove Theorem 18.

Proof of Theorem 18. The proof is similar to [59, Corollary 3]. To begin, let $\Sigma_O = \{t_f^*, x^*, \xi^*, u_i^*, \gamma_i^*, \sigma_i^*\}$ be the *original solution* returned by Problem 6.6, which achieves the optimal cost value $J_{\mathcal{R}}^*$. Since $\mathcal{T}_c$ is a discrete set, for any consecutive contact times $\tau_1 < \tau_2$ there exists a large enough real $a > 0$ such that $\tau_1 + 1/a < \tau_2 - 1/a$. Let $\tau_e = \tau_1 + 1/a$ and $\tau_f = \tau_2 - 1/a$. Now consider solving Problem 6.6 over $[\tau_e, \tau_e + \Delta\tau]$ with $t_f = \Delta\tau$, $x_0 = x(\tau_e)$, $b[t_f] = x(\Delta\tau) - x(\tau_f)$. Call the solution to this problem the *subproblem solution* $\Sigma_S = \{\tilde{\Delta}\tau, \tilde{x}, \tilde{\xi}, \tilde{u}_i, \tilde{\gamma}_i, \tilde{\sigma}_i\}$, and let J_S^* be the achieved optimal cost. We claim that the corresponding portion of Σ_O must also achieve J_S^*. If it does not, the *modified solution* $\Sigma_M = \{\hat{t}_f, \hat{x}, \hat{\xi}, \hat{u}_i, \hat{\gamma}_i, \hat{\sigma}_i\}$ such that $\hat{t}_f = t_f^* + \tilde{\Delta}\tau - (\tau_f - \tau_e)$ and $\{\hat{x}, \hat{\xi}, \hat{u}_i, \hat{\gamma}_i, \hat{\sigma}_i\} =$

$$\begin{cases} \{x^*(t), \xi^*(t), u_i^*(t), \gamma_i^*(t), \sigma_i^*(t)\} & \text{for } t \in [0, \hat{t}_f] \setminus \\ & [\tau_e, \tau_e + \tilde{\Delta}\tau], \\ \{\tilde{x}(t), \tilde{\xi}(t), \tilde{u}_i(t), \tilde{\gamma}_i(t), \tilde{\sigma}_i(t)\} & \text{for } t \in [\tau_e, \tau_e + \tilde{\Delta}\tau], \end{cases}$$

is also feasible for Problem 6.6 and achieves a lower cost than $J^*_{\mathcal{R}}$, which contradicts that the $[\tau_e, \tau_f]$ segment of Σ_O is optimal. Thus, Σ_S must be optimal for the original problem. By Theorem 17, Σ_S must be globally optimal for Problem 6.1. Since a is arbitrarily large, Σ_S must be optimal for Problem 6.1 over $t \in (t_1, t_2)$. Let $\mathcal{T}_c = \{\tau_i,\ i = 1, 2, \dots\}$, $\tau_i < \tau_{i+1}$ $\forall i$. Hence $\operatorname{int} \mathcal{T}_i = \bigcup_i (\tau_i, \tau_{i+1})$ and Σ_O is globally optimal for Problem 6.1 a.e. $\mathcal{T}_i$. Since $\mathcal{T}_c$ is a discrete set, $\operatorname{cl} \mathcal{T}_i = [t_0, t_f]$ and so the Problem 6.6 solution is globally optimal for Problem 6.1 a.e. $[0, t_f]$. □

The main lossless convexification results of Theorems 17 and 18 have now been proven. These extend classical lossless convexification theory, which can only handle a continuous signal, to input of semi-continuous nature that encapsulate fundamentally binary decision making. Thus, by solving Problem 6.6, a significant class of optimal control problems modeled by Problem 6.1 under Assumption 6 and Conditions 12-15, *convex optimization can be used to find the global optimum of mixed-integer optimal control problems in provable real-time*. The next section corroborates the effectiveness of this approach on two examples: spacecraft rendezvous and rocket landing.

6.6 Numerical Examples

This section presents two numerical validations of Theorems 17 and 18, comparing both accuracy and performance to standard solutions using a MICP solver:

1. **Spacecraft docking to a rotating space station**: for this example, the restricted result of Section 6.3 is sufficient. An identical solution is achieved to a mixed-integer solver for a minimum-time maneuver, but with a runtime that is almost three orders of magnitude faster. The intuitive interplay between the input gain measure (6.3) and the optimal input selection (given directly by (6.25) in the lossless convexification core proof) is also plotted;

2. **Rocket landing with a coupled thrust-gimbal angle constraint**: this example tests the full extent of the theory developed in this chapter. The system under study is

a single-engine planetary rocket lander with the following complication: when thrust is above a threshold then the maximum gimbal angle is restricted, but for low thrust values the gimbal angle may be higher. We call this a "coupled thrust-gimbal" constraint, and the theory developed in this chapter allows to effectively handle more general cases than what is allowed by previous lossless convexification results (discussion continued in Section 6.6.2). Once again, a traditional mixed-integer approach is shown to be either much slower, or incapable within reasonable runtimes of finding any solution at all .

The source code for both examples is publicly available[3]. In both cases, Python 2.7.15 with ECOS 2.0.7.post1 [43] is used for implementation on a Ubuntu 18.04.1 64-bit platform with a 2.5 GHz Intel Core i5-7200U CPU and 8 GB of RAM. The solution and runtime are compared to a mixed-integer formulation where (6.3d) is implemented directly as a binary constraint using Gurobi 8.1 [133].

6.6.1 Spacecraft Docking to a Rotating Space Station

This section shows how a trajectory for spacecraft docking to a rotating space station can be computed more efficiently via Problem 6.4 versus a standard MICP approach. The spacecraft's dynamics are described in the rotating frame by:

$$\dot{x}(t) = A(\omega)x(t) + B\sum_{i=1}^{M} u_i(t), \tag{6.27}$$

where $x(t) = (r(t), v(t)) : \mathbb{R}_+ \to \mathbb{R}^6$ is the position and velocity state, $\omega \in \mathbb{R}^3$ is the space station's constant angular velocity vector, and

$$A(\omega) \triangleq \begin{bmatrix} 0 & I \\ -S(\omega)^2 & -2S(\omega) \end{bmatrix}, \; B \triangleq \begin{bmatrix} 0 \\ I \end{bmatrix}, \tag{6.28}$$

[3]Hosted on GitHub: https://github.com/dmalyuta/lcvx.

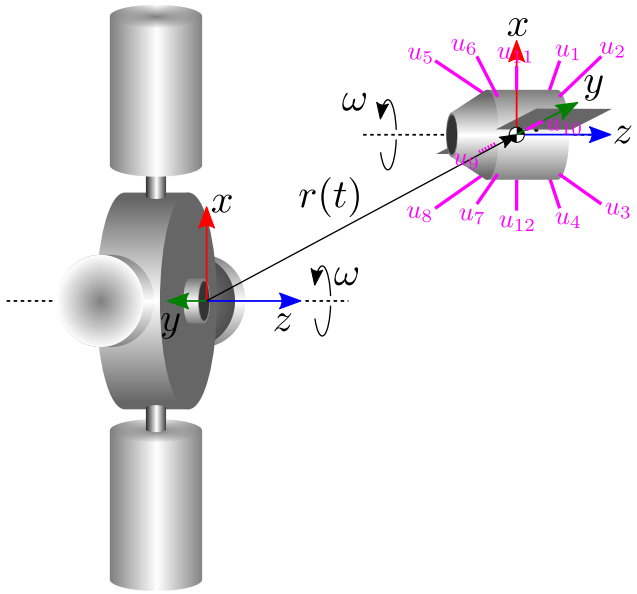


Figure 6.8: Spacecraft docking to a rotating space station is modelled in the space station's rotating frame. The spacecraft is assumed to have matched the space station angular velocity.

where $S(\omega) \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix representation of the cross product $\omega \times (\cdot)$.

We assume that the spacecraft is equipped with a reaction control system capable of producing up to $K = 4$ acceleration vectors from a total of $M = 12$ distinct directions, as illustrated in Figure 6.9. Acceleration vectors along the positive z-axis point with a pitch and roll of 40 degrees. Along the negative z-axis, the pitch and roll is 30 degrees. The docking port is positioned at the origin and rotates with the space station. The spacecraft is assumed to rotate with the same angular velocity ω and is tasked to perform translation control to berth with the docking port. We use the parameters:

$$\omega = (0, 0, 1) \text{ rpm}, \ \rho_1 = 1 \text{ mm s}^{-2}, \ \rho_2 = 10 \text{ mm s}^{-2},$$
$$m[t_f] = t_f, \ r(0) = (5, 5, 100) \text{ m}, \ v(0) = (0, 0, 0) \text{ m s}^{-1},$$
$$r(t_f) = (0, 0, 0) \text{ m}, \ v(t_f) = (0, 0, -0.01) \text{ m s}^{-1},$$

where the initial velocity choice makes the spacecraft's inertial velocity $\omega \times r(0)$ m s^{-1}. A

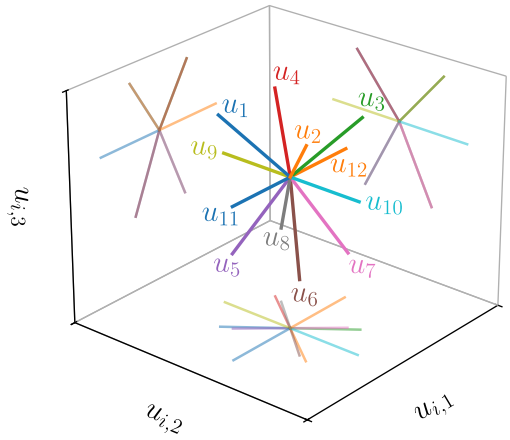


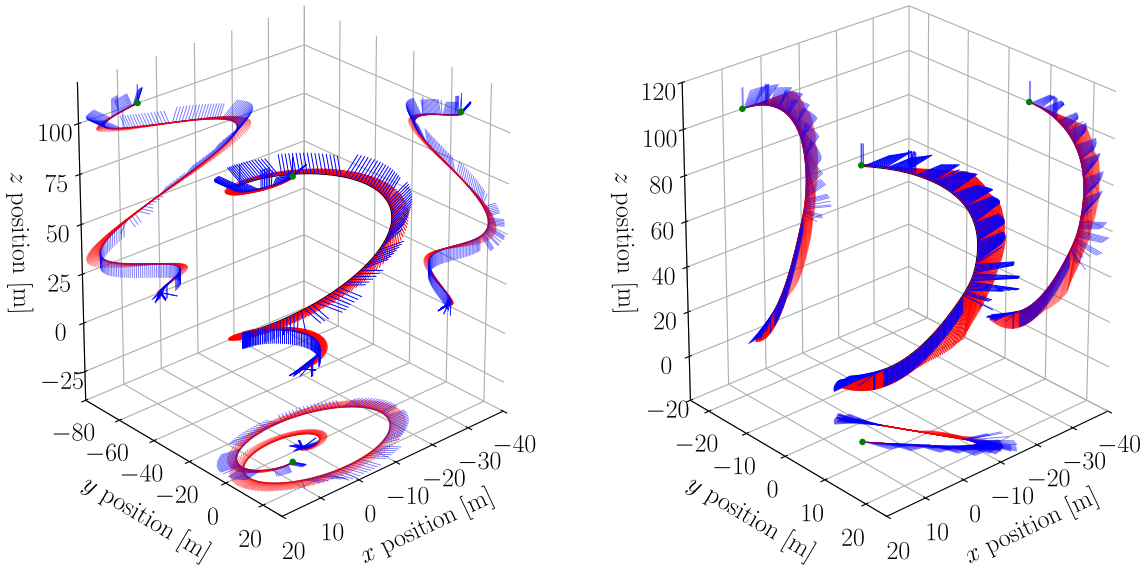
Figure 6.9: The spacecraft's reaction control system is capable of producing up to $K = 4$ out of $M = 12$ acceleration vectors, acting through the center of mass.

more realistic setting would be $v(0) = -\omega \times r(0)$ which makes the inertial velocity zero. The motivation for the present choice is to enrich the solution. Since the cost is linear in t_f, both bisection and golden search can be applied to find the minimum t_f [139, 170].

Conditions 8-11 are satisfied by this problem. Condition 8 is satisfied since $\{-A(\omega)^\intercal, B^\intercal\}$ is observable. Condition 11 is satisfied according to the minimum-time special case in Figure 6.6b. Conditions 9 and 10 hold following the discussion in Section 6.3.1 and noting that the primer vector $y(\cdot)$ in (6.5) can only be persistently normal to ω or to vectors normal to ω. The conditions are checked automatically via a script in the public source code.

The dynamics (6.27) are discretized using zeroth-order hold over a uniform temporal grid of 300 nodes. Figure 6.11 shows the resulting state, input and input gain trajectories for the globally optimal solution. The solution is obtained in 20 s via Problem 6.4, whereas MICP takes an intractable 6200 s. This is expected, since solving Problem 6.4 relies on an SOCP solver with polynomial time complexity, whereas MICP has exponential time complexity.

Figure 6.11a confirms that the constraints (6.3c)-(6.3e) are satisfied. In particular, the thrust magnitude is bang-bang as predicted in Lemma 6. The intermediate thrusts occuring at rising and falling edges are discretization artifacts since the lossless convexification guarantee is only "almost everywhere" in nature. These artifacts have been observed since



(A) View in rotating frame of the space station.

(B) View in inertial frame.

Figure 6.10: Optimal docking trajectory in (A) space station rotating frame, and (B) the inertial frame. The time of flight is $t_f = 135$ s. Red vectors show the scaled velocity, blue vectors show the scaled thrust (negative of acceleration), and the green marker shows the initial position.

the early days of the lossless convexification method [35]. Figure 6.11b confirms the optimal input structure (6.25). In particular, for the minimum-time solution it is always the inputs corresponding to the largest $K = 4$ gain values $\Gamma_i(t)$ that are active.

6.6.2 Rocket Landing with Coupled Thrust-Gimbal Constraint

This section performs a kind of stress-test of the most general lossless convexification results derived in this section, namely Theorems 17 and 18. The system considered for this purpose is a rocket-powered planetary lander. This example motivated the original lossless convexification studies at NASA JPL in 2005-2007 [35, 138], whose practical results have been since incorporated into JPL's guidance for optimal large divert (G-FOLD) software [171]. The latter underwent a successful 7-flight, 3-year flight test campaign in 2012-2014 [155, 156], demonstrated on-board convex optimization and expanded the "divert[4]" capability of Masten Space Systems' Xombie sounding rocket from 75 m to 750 m with horizontal speeds in excess of 25 $\mathrm{m\,s^{-1}}$ [1, 9, 154].

Classical lossless convexification theory allows to handle a *pointing constraint* (in other words, a constraint on the rocket engine gimbal angle) of the following form, as illustrated in Figure 6.13 (and which we have introduced back in Examples 4 and 7):

$$\hat{n}^{\mathsf{T}} u(t) \geq \|u(t)\|_2 \cos(\bar{\theta}), \tag{6.29}$$

where $\hat{n} \in \mathbb{R}^3$ is the nominal (unit-norm) pointing direction "up", and $\bar{\theta} \in [0, \pi]$ is a fixed maximum pointing angle. In [140, 146], a lossless convexification result is shown where the pointing constraint is convexified (for $\bar{\theta} > \pi/2$) to:

$$\hat{n}^{\mathsf{T}} u(t) \geq \sigma(t) \cos(\bar{\theta}), \tag{6.30}$$

[4]The divert maneuver redirects a planetary lander located at point A and intending to land at point B, to an alternative landing location at point C (presumably due to the determination by on-board sensors that landing location B is hazardous).

where $\sigma(t)$ is a slack variable in the same was as in Problems 6.4 and 6.6. Fortunately, the lossless convexification proof in [146] can be generalized to the case where

$$\cos(\bar{\theta}(t)) = \frac{\rho_2 - \|u(t)\|_2}{\rho_2 - \rho_1}\cos(\theta_{\max}) + \frac{\|u(t)\|_2 - \rho_1}{\rho_2 - \rho_1}\cos(\theta_{\min}), \tag{6.31}$$

where $\theta_{\min}, \theta_{\max} \in [0, \pi]$, $\theta_{\min} \leq \theta_{\max}$, as the minimum and maximum gimbal angles. This can be equivalently written as the affine relationship:

$$\cos(\bar{\theta}(t)) = \alpha_0 + \alpha_1\|u(t)\|_2, \tag{6.32}$$

where

$$\alpha_0 \triangleq \frac{\rho_2\cos(\theta_{\max}) - \rho_1\cos(\theta_{\min})}{\rho_2 - \rho_1}, \ \alpha_1 \triangleq \frac{\cos(\theta_{\min}) - \cos(\theta_{\max})}{\rho_2 - \rho_1}. \tag{6.33}$$

Thus, (6.30) can be generalized to the (still convex) constraint:

$$\hat{n}^{\mathsf{T}}u(t) \geq \alpha_0\sigma(t) + \alpha_1\sigma^2(t), \tag{6.34}$$

which is a lossless convexification of the constraint:

$$\theta(t) \leq \arccos(\alpha_0 + \alpha_1\|u(t)\|_2), \tag{6.35}$$

where $\theta(t)$ is the gimbal angle as shown in Figure 6.13. An example for $\theta_{\min} = 20$ °, $\theta_{\max} = 80$ °, $\rho_1 = 2\ \mathrm{m\,s^{-2}}$, and $\rho_2 = 10\ \mathrm{m\,s^{-2}}$ yields a constraint illustrated in Figure 6.14. However, an affine dependence of the gimbal angle *cosine* on the rocket engine thrust can be a restrictive assumption, and certainly similar dependencies may appear in other applications requiring more complicated descriptions.

This section shows how the general coupled thrust-gimbal angle constraint can be losslessly convexified via the theory presented in this chapter, and solved much faster via Prob-

lem 6.6 than MICP. Consider the in-plane rocket dynamics:

$$\dot{x}(t) = A(\omega)x(t) + B\sum_{i=1}^{M} u_i(t) + w, \tag{6.36}$$

where the vehicle is treated as a point mass with $x(t) = \begin{bmatrix} r(t)^\intercal & v(t)^\intercal \end{bmatrix}^\intercal \in \mathbb{R}^4$ the position and velocity state and $\omega \in \mathbb{R}$ the planet rotation rate, which is assumed to be constant and perpendicular to the trajectory plane[5]. The input $u_i(t) \in \mathbb{R}^2$ represents an acceleration imparted on the rocket by a gimballed thruster. The LTI matrices are:

$$A(\omega) = \begin{bmatrix} 0 & I \\ \omega^2 I & 2\omega S \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ I \end{bmatrix}, w = \begin{bmatrix} 0 \\ \omega^2 l + g \end{bmatrix}, \tag{6.37}$$

where $S = [0\ 1; -1\ 0] \in \mathbb{R}^{2\times 2}$, $I \in \mathbb{R}^{2\times 2}$ is identity, $l \in \mathbb{R}^2$ is the landing pad position with respect to the planet's center of rotation, and $g \in \mathbb{R}^2$ is the gravity vector. Note that the dynamics assume constant mass and gravity for concision, but both can be made variable within the lossless convexification framework [35, 116].

The rocket is equipped with a single gimballed thruster which operates in two modes: 1) low-thrust high-gimbal, and 2) high-thrust low-gimbal. A maximum gimbal angle range of $\theta_i \in (0, \pi)$ is enforced via (6.1f) by setting:

$$C_i = \begin{bmatrix} -\cos(\theta_i/2) & -\sin(\theta_i/2) \\ \cos(\theta_i/2) & -\sin(\theta_i/2) \end{bmatrix}. \tag{6.38}$$

We also impose a glide slope constraint as in [139] which prevents the rocket from approaching the ground too closely prior to touchdown:

$$\mathcal{X} = \{x = (r, v) \in \mathbb{R}^4 : \hat{e}_y^\intercal r \geq \|r\| 2\sin(\gamma_{gs})\}, \tag{6.39}$$

[5]This is done for simplicity in order to keep the motion planar. A general 3-dimensional angular velocity vector can also be considered.

where $\hat{e}_y = (0,1) \in \mathbb{R}^2$ is the unit vector along the altitude axis. We choose the following parameters, corresponding to a Martian divert maneuver similar to [35]:

$$
\begin{gathered}
M = 2,\ K = 1,\ \omega = 2\pi/88775 \text{ rad}\,\text{s}^{-1},\ \rho_1^1 = 4 \text{ m}\,\text{s}^{-2}, \\
\rho_1^2 = 8 \text{ m}\,\text{s}^{-2},\ \rho_2^1 = 8 \text{ m}\,\text{s}^{-2},\ \rho_2^2 = 12 \text{ m}\,\text{s}^{-2},\ \theta_1 = 120 \text{ °}, \\
\theta_2 = 10 \text{ °},\ \gamma_{gs} = 10 \text{ °},\ l = (0, 3396.2) \text{ km},\ \zeta \in \{0,1\}, \\
g = (0, -3.71) \text{ m}\,\text{s}^{-2},\ m[t_f] = (1-\zeta) t_f \xi_{\max}/t_{f,\max}, \\
\ell(x(t)) = 10^{-3}\xi_{\max}(|r_1(t)| \tan(\gamma_{gs}) + |r_2(t)|)/h_0, \\
r(0) = (1500, h_0) \text{ m},\ v(0) = (50, -70) \text{ m}\,\text{s}^{-1}, \\
r(t_f) = (0,0) \text{ m},\ v(t_f) = (0,0) \text{ m}\,\text{s}^{-1},
\end{gathered}
$$

where $t_{f,\max} = 100$ s is the time of flight upper-bound and $\xi_{\max} = t_{f,\max}\rho_2^2$ is the maximum input integral cost. The optimal cost is verified to be unimodal in t_f such that golden search can be applied to find the optimal t_f [139, 170]. The initial altitude above ground level (AGL) h_0 and $\zeta \in \{0,1\}$ are independent variables that we shall vary. When $\zeta = 0$, we solve for a minimum-time trajectory, while for $\zeta = 1$ we solve for a minimum-fuel trajectory.

The problem satisfies Conditions 12-15 under a few light assumptions. Because the glide slope (6.39) maintains the rocket above zero altitude, $\ell[t] > 0\ \forall t \in [0, t_f)$ such that Condition 15 holds irrespective of m. To check Condition 12, recognize that for our choice of ℓ:

$$
\partial\ell[t]^\intercal = D\partial_r\ell[t]^\intercal,\ D \triangleq \begin{bmatrix} I \\ 0 \end{bmatrix}. \tag{6.40}
$$

Following the discussion in Section 6.4.1, we confirm that the LTI system $\{-A^\intercal, D, B^\intercal, 0\}$ is strongly observable, hence Condition 12 holds. To check Conditions 13 and 14, we need to make the following assumption because replacing $\partial_r\ell[t]^\intercal$ with $\mathbb{R}^2$ is too conservative.

Assumption 8. The downrange and altitude are non-zero almost everywhere, i.e. $r_1(t) \neq 0$ and $r_2(t) \neq 0$ a.e. $[0, t_f]$.

Leveraging Assumption 8 yields a piecewise constant input to the adjoint system:

$$\partial_r \ell[t]^\intercal = \frac{10^{-3}\xi_{\max}}{h_0} \left\{ \begin{bmatrix} \tan(\gamma_{gs}) \\ 1 \end{bmatrix}, \begin{bmatrix} -\tan(\gamma_{gs}) \\ 1 \end{bmatrix} \right\}. \tag{6.41}$$

Leveraging (6.41), consider the following LTI system where a constant input is modelled as a static state, yielding an augmented state $\lambda'(t) \in \mathbb{R}^6$:

$$\dot{\lambda}'(t) = \begin{bmatrix} -A^\intercal & D \\ 0 & 0 \end{bmatrix} \lambda'(t) = A'\lambda'(t), \tag{6.42a}$$

$$y(t) = \begin{bmatrix} B^\intercal & 0 \end{bmatrix} \lambda'(t) = C'\lambda'(t). \tag{6.42b}$$

When $\zeta = 0$, checking Conditions 13 and 14 reduces to ensuring that $\dot{y}(t) = C'A'\lambda'(t)$ cannot evolve perpendicular to certain constant vectors $\hat{n} \in \mathbb{R}^2$. The values of $\hat{n}$ that need to be checked are illustrated in Figure 6.15a. To verify Conditions 13 and 14, we check the observability properties of the pair $\{A', \hat{n}^\intercal C'A'\}$, as in Section 6.6.1. Let $V_{\hat{n}}$ be a matrix whose columns span the unobservable subspace. It turns out for the rocket landing problem that $A'V_{\hat{n}} = 0 \ \forall \hat{n}$. Conditions 13 and 14 can thus be violated only by a constant primer vector. If this occurs, the input is constrained to point in the directions shown in Figure 6.15b. Notice that this constrains the downrange acceleration to always have the same sign. The following assumption requires the rocket to experience both acceleration *and* deceleration. The assumption is satisfied if, for example, the rocket is initially travelling away from the landing site and has to reverse its velocity.

Assumption 9. The downrange acceleration $\sum_{i=1}^{M} u_{i,1}(t)$ changes sign at least once over $[0, t_f]$.

The assumption is sufficient for Theorem 17 but not Theorem 18, because a discontinuity in $\dot{y}(t)$ may occur at $t \in \mathcal{T}_c$ (6.26b) [87]. If state constraints are activated, a "sufficiently rich" gimbal history may be assumed or Conditions 13 and 14 may be verified *a posteriori*,

Table 6.1: Optimal cost and solver runtime when solving Problem 6.6 versus MICP. Dashes show when MICP took too long to converge (> 10 min per iteration).

h_0 [m]	ζ	$J^*_{\mathcal{R}}$	J^*_{MICP}	$t_{\mathcal{R}}$ [s]	t_{MICP} [s]
650	0	636.2	–	2.9	–
650	1	374.5	–	2.4	–
800	0	577.7	577.8	2.4	232.3
800	1	350.8	350.9	2.3	269.9
1000	0	548.9	–	3.9	–
1000	1	333.7	333.7	2.3	566.8
1500	0	493.4	–	2.5	–
1500	1	316.1	316.1	2.2	177.3
3000	0	558.0	558.0	2.5	73.1
3000	1	323.0	323.1	1.8	505.9

i.e. the solution is lossless if they hold.

When $\zeta = 1$, Condition 13 requires $\|y(t)\|2 \neq 1$ a.e. $[0, t_f]$. Modal shape analysis for the pair $\{A', C'\}$ reveals that, given a constant input in (6.41), $\|y(t)\|2 = 1$ for an interval is only possible if $y(t)$ is constant. This is eliminated by Assumption 9 with the same caveat about state constraint activation. Checking Condition 14 is not possible *a priori* when $\zeta = 1$. The condition is verified *a posteriori*.

The dynamics (6.36) are discretized via zeroth-order hold on a uniform temporal grid of 150 nodes. Figure 6.16 shows the resulting state, input and input gain trajectories. Let us first discuss Figures 6.16a and 6.16b. The top row shows the overall trajectory, from which we note that Assumptions 8 and 9 are satisfied. The second and third rows show that the input norm is feasible almost everywhere for Problem 6.1. In particular, the thrust magnitude is bang-bang as predicted in Lemma 6. The intermediate thrusts occuring at the rising and falling edges in the third row are discretization artifacts. Recall that the lossless convexification guarantee is only "almost everywhere" in nature. These artifacts have been observed since the early days of lossless convexification theory [35]. Note the kink that occurs in the $y(t)$ trajectory in the second row, which coincides with the glide slope state

constraint activation as highlighted by the red dot in the first row. Looking at the third row, $\sigma_i(t) \neq \|u_i(t)\|2$ as expected when $\zeta = 0$ and both inputs are off, since there is no cost incentive to minimize $\sigma_i(t)$. Note that optimality nevertheless requires $u_i(t) = 0$, as predicted by Lemma 6. Finally, the fourth row shows the $\Gamma_i(t)$ trajectories. As predicted by (6.25), when $\Gamma_i(t) > \Gamma_j(t)$, optimality forces input $\gamma_i(t) = 1$ and $\gamma_j(t) = 0$.

Table 6.1 compares the achieved optimal cost and solver runtimes of lossless convexification versus a direct MICP implementation of (6.1d). One can see that the optimal cost values are quasi-identical, with some slightly lower values for lossless convexification due to the "intermediate thrusts" discussed above. More importantly, solving Problem 6.6 is up to two orders of magnitude faster than using MICP. This is expected, since SOCP has polynomial time complexity in the problem size while MICP has exponential time complexity. Furthermore, MICP was not able to find a trajectory in several cases (the computation was aborted when runtime exceeded 10 min for a single golden search iteration). The third column of Figure 6.16 shows a sequence of 50 landing trajectories for a sweep over $h_0 \in [650, 6000]$ m AGL. Computing this sequence of 50 trajectories with $N = 150$ takes 130 s, which is less than the average MICP solution time for a single trajectory.

6.7 The Future of Lossless Convexification

The previous sections presented a novel lossless convexification result that applies to a new and relatively broad class of mixed-integer problems. It is appropriate that we conclude this chapter with an outlook for the future of lossless convexification as an trajectory optimization method at large.

Lossless convexification is a method that solves nonconvex trajectory generation problems with one or a small number of calls to a convex solver. This places it among the most reliable and robust methods for nonconvex trajectory generation. The future of LCvx therefore has an obvious motivation: to expand the class of problems that can be losslessly convexified. In the past two years, LCvx research has been rejuvenated by several fundamental discoveries and practical methods that expand the method to new and interesting problem types. This

section briefly surveys these new results.

Fixed-final Time Problems

The first new LCvx result applies to a fixed-final time and fixed-final state version of Problem 5.1 with no state constraints. To begin, recognize that the classical LCvx result from Theorem 10 does not apply when both t_f and $x(t_f)$ are fixed. In this case, $B_{\text{LCvx}} = I_{n+1}$ in (5.7b) and therefore its columns, which span all of $\mathbb{R}^{n+1}$, cannot be linearly independent from m_{LCvx}. Thus, traditionally one could not fix the final time and the final state simultaneously. Very recently, Kunhippurayil et al. [143] showed that Condition 2 is in fact not necessary for the following version of Problem 5.1:

$$\min_{u,t_f} \int_0^{t_f} \ell(g(u(t)))\,\mathrm{d}t \tag{6.43a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + Bu(t), \tag{6.43b}$$

$$\rho_{\min} \leq g(u(t)) \leq \rho_{\max}, \tag{6.43c}$$

$$x(0) = x_0,\ x(t_f) = x_f, \tag{6.43d}$$

where t_f is fixed and $x_f \in \mathbb{R}^n$ specifies the final state. The lossless relaxation is the usual one, and is just a specialization of Problem 5.2 for Problem 6.43:

$$\min_{\sigma,u,t_f} \int_0^{t_f} \ell(\sigma(t))\,\mathrm{d}t \tag{6.44a}$$

$$\text{s.t. } \dot{x}(t) = Ax(t) + Bu(t)w(t), \tag{6.44b}$$

$$\rho_{\min} \leq \sigma(t) \leq \rho_{\max}, \tag{6.44c}$$

$$g(u(t)) \leq \sigma(t), \tag{6.44d}$$

$$x(0) = x_0,\ x(t_f) = x_f. \tag{6.44e}$$

The following result is then proved in [143]. By dropping Condition 2, the result generalizes Theorem 10 and significantly expands the reach of LCvx for problems without state

constraints.

Theorem 20. *The solution of Problem 6.44 is globally optimal for Problem 6.43 if Condition 1 holds and t_f is between the minimum feasible time and the time which minimizes (6.43a). For longer trajectory durations, there exists a solution to Problem 6.44 that is globally optimal for Problem 6.43.*

Perhaps the most important part of Theorem 20, and a significant future direction for LCvx, is in its final sentence. Although a lossless solution "exists", how does one find it? An *algorithm* is provided in [143] to find the lossless solution, that is, one solution among many others which may not be lossless. This is similar to Theorem 14 and Algorithm 5: we know that slackness in (6.44d) may occur, so we devise an algorithm that works around the issue and is able to recover an input for which (6.44d) holds with equality. Most traditional LCvx results place further restrictions on the original problem in order to "avoid" slackness, but this by definition limits the applicability of LCvx. By instead providing algorithms which recover lossless inputs from problems that do not admit LCvx naturally, we can tackle lossless convexification "head on" and expand the class of losslessly convexifiable problems. An approach that is similar in spirit is taken for spacecraft rendezvous in [173], where an iterative algorithm that modifies the dynamics is devised to extract bang-bang controls from a solution that exhibits slackness.

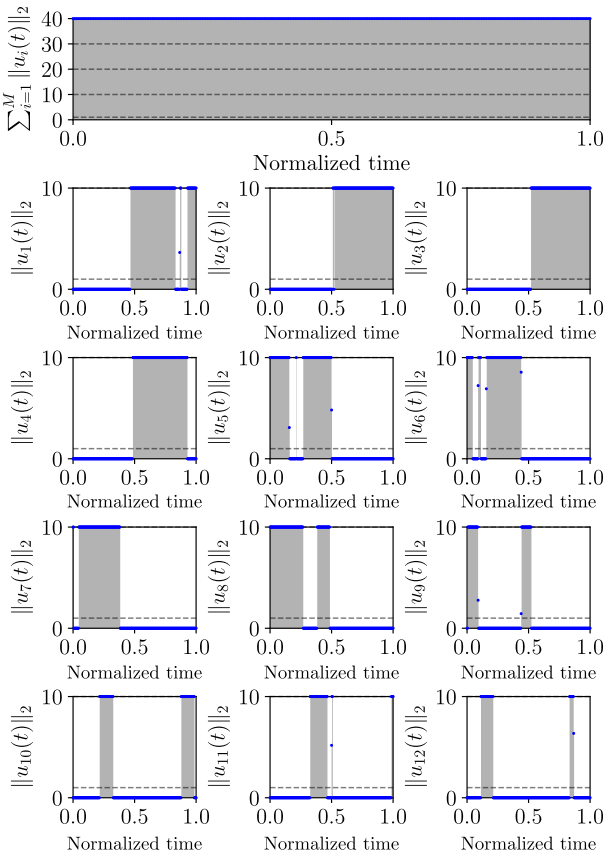
Hybrid System Problems

A very recently published result by Harris treats the lossless convexification of a similar problem to the one in this chapter (i.e., Problem 6.1) from a slightly different angle. In [142] it is recognized that a version of Problem 6.6 will be a lossless convexification of Problem 6.1 if the dynamical system is "normal" due to the so-called bang-bang principle [40, 174]. Normality is related to, but much stronger than, the notion of controllability from Condition 1. Nevertheless, it is shown that the dynamical system can be perturbed by an arbirarily small amount to induce normality. This phenomenon was previously observed in a

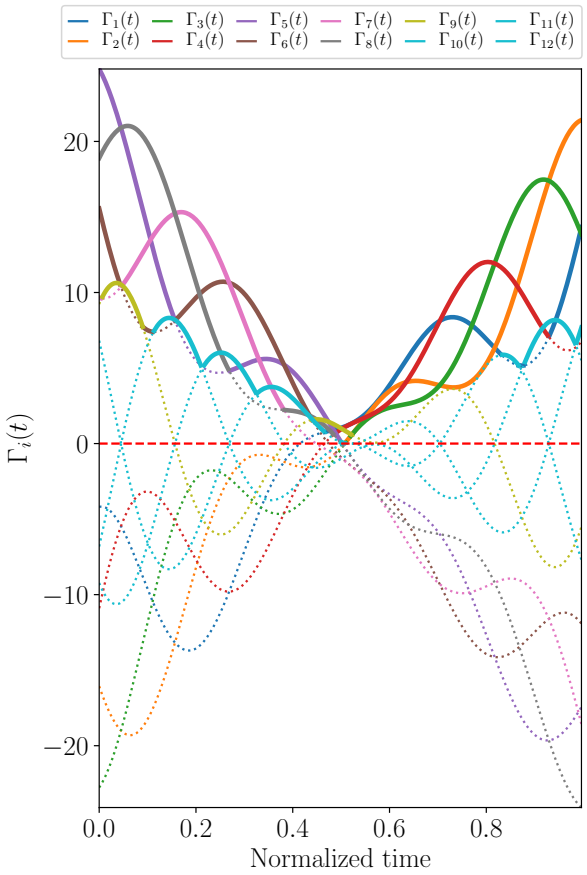
practical context for rocket landing LCvx with a pointing constraint, which we discussed for Problem 5.8 [140]. Practical examples are shown for spacecraft orbit reshaping, minimum-energy transfer, and cubesat differential drag and thrust maneuvering. It is noted that while mixed-integer programming fails to solve the latter problem, the convex relaxation is solved in < 0.1 seconds.

In this chapter and the corresponding publications [51, 52], on the other hand, we bypass normality and directly deadl with the nonsmooth maximum principle [82, 86, 87]. The result is that we come up with a set of conditions for which LCvx holds (i.e., Conditions 12-15). These conditions are an interesting mix of problem geometry and Conditions 1 and 2. Notably, they are more general than normality, so they can be verified by systems which are not normal.

Our contribution and that of [142] can thus be seen as complementary: we show that for some systems, the perturbation proposed by [142] is not necessary. On the other hand, [142] provides a method to recover LCvx when our conditions fail. Altogether, the fact that an arbitrarily small perturbation of the dynamics can recover LCvx suggests a deeper underlying theory for how and why problems can be losslessly convexified. We feel that the search for this reason will be a running theme of future LCvx developments, and its eventual discovery will lead to more general lossless convexification algorithms.



(a) Optimal input norm history in $\mathrm{mm\,s^{-2}}$. The bang-bang nature of the solution is clearly visible. Blue markers show $\sigma_i(t)$.



(b) Time history of the input gain (6.3). Bold lines show when the corresponding input is active.

Figure 6.11: Optimal docking trajectory RCS acceleration vector (a) and gain measure (b) histories.

Figure 6.12: A Xombie technology demonstrator from Masten Space Systems, Mojave, Calif., ascends from its pad at Mojave Air and Space Port on a test for NASA's Jet Propulsion Laboratory. The vehicle is a vertical-takeoff, vertical-landing experimental rocket. It is being used in collaboration with NASA Dryden Flight Research Center to evaluate performance of JPL's Fuel Optimal Large Divert Guidance (G-FOLD), a new algorithm for planetary pinpoint landing of spacecraft. *Image and caption credit: NASA/Masten [172].*

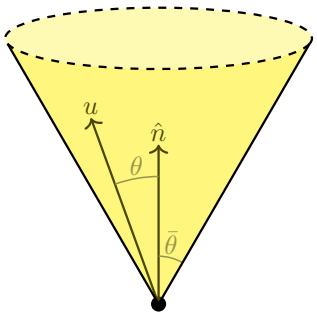


Figure 6.13: Illustration of the pointing constraint that can be handled by classical lossless convexification results.

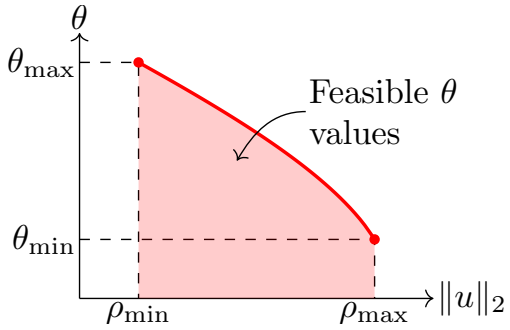
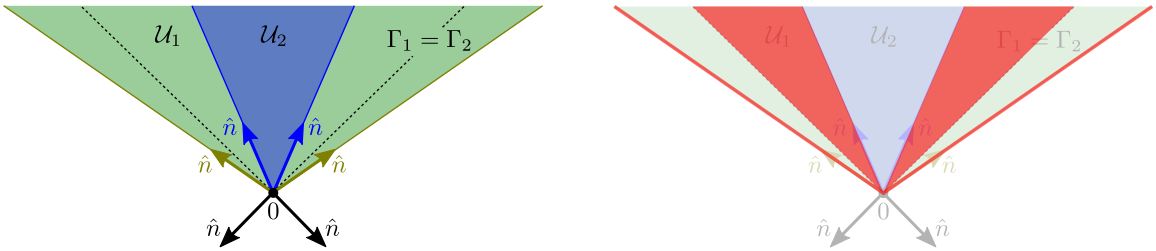


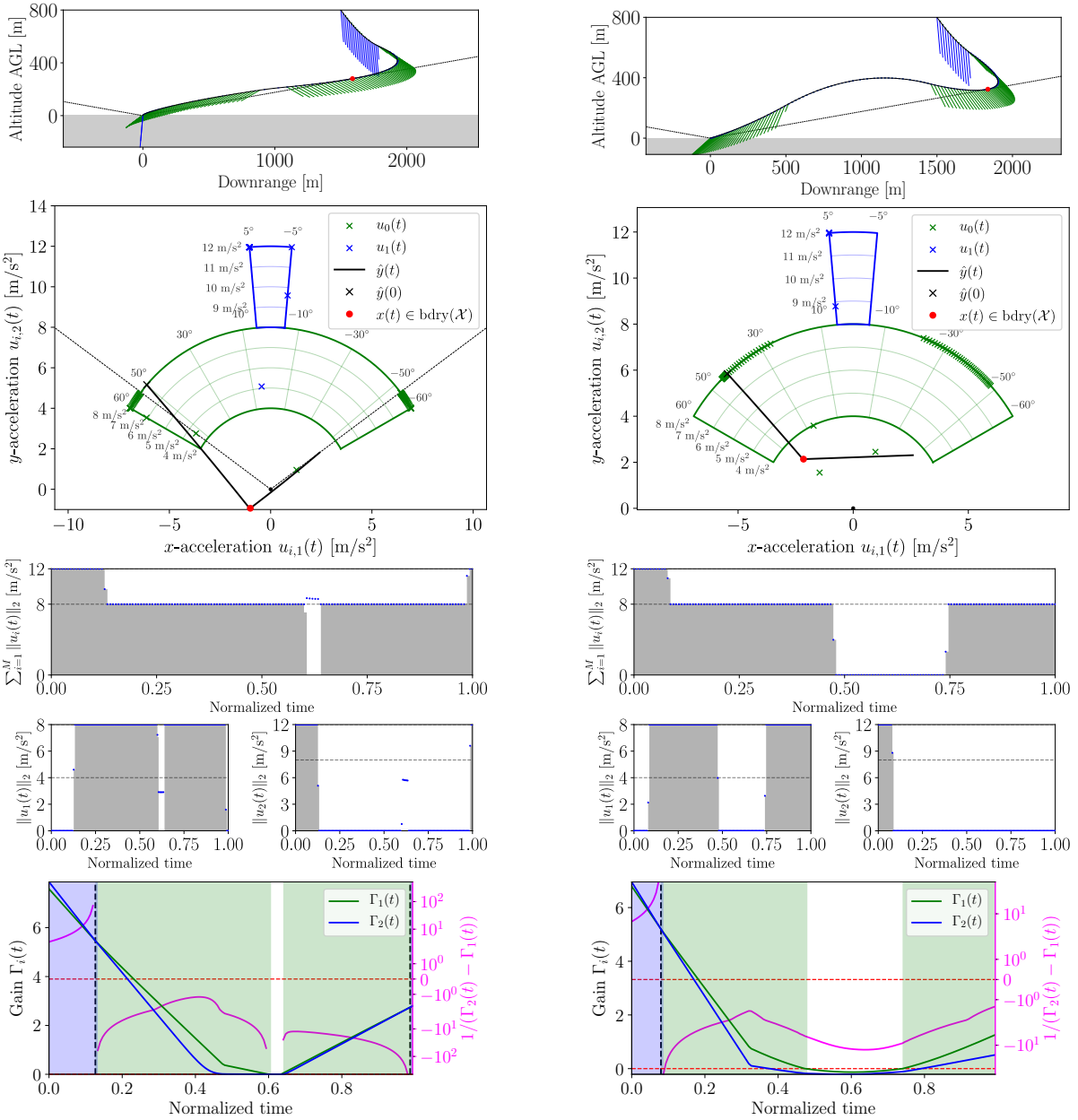
Figure 6.14: Illustration of the pointing constraint that can be handled by a simple extension to classical lossless convexification result [140, 146]. Assumes an affine dependence of the gimbal angle *cosine* on the thrust.



(a) Illustration of the six vectors $\hat{n}$ that $\dot{y}(t)$ must not be normal to for Conditions 13 and 14 to hold.

(b) If the normality check fails, the optimal input could point in the directions highlighted in red.

Figure 6.15: Illustrated verification of Conditions 13 and 14 when $\zeta = 0$. If $\dot{y}(t)$ can evolve normal to any vector in (a), the input can point in the directions shown in (b) while violating (6.1d).



(a) Landing from $h_0 = 800$ m AGL, $\zeta = 0$. Time of flight $t_f = 46.93$ s.

(b) Landing from $h_0 = 800$ m AGL, $\zeta = 1$. Time of flight $t_f = 53.97$ s.

Figure 6.16: Landing trajectories computed by Problem 6.6. Green shows the high-gimbal low-thrust mode and blue shows the low-gimbal high-thrust mode. In (a) and (b), the top row shows the position trajectory with overlaid thrusts ($-u_i(t)$). Dotted lines show glide slope (6.39). The second row shows the input with the (normalized) primer vector (6.7b). Dotted lines show the equal-gain manifold $\Gamma_1(t) = \Gamma_2(t)$. The third row shows the input magnitude history. The bottom row shows each input's gain (6.8) and their difference. The background colour shows when the corresponding input is active..

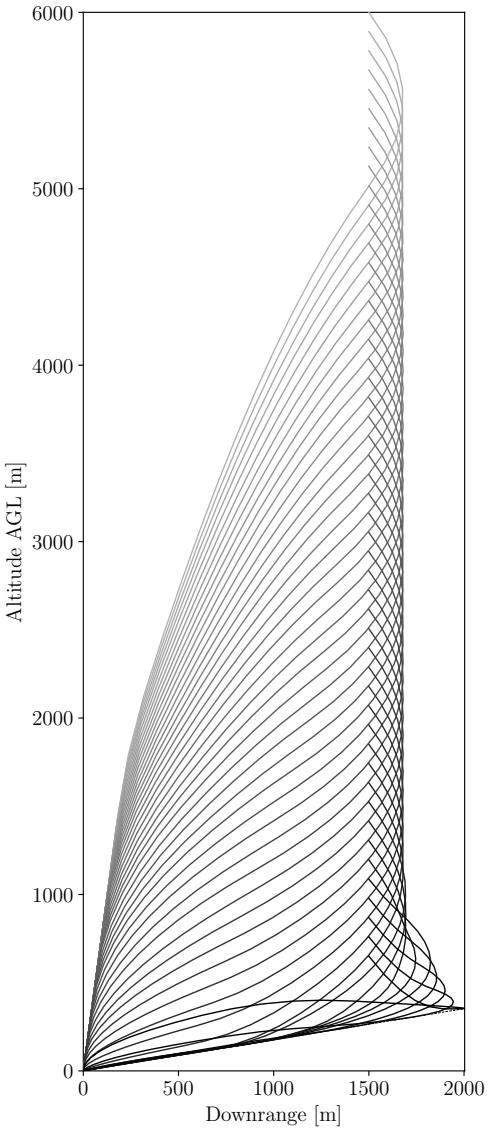


Figure 6.17: Trajectory sweep over initial altitudes $h_0 \in [650, 6000]$ m AGL. Using $\zeta = 0$ and $N = 30$.

Chapter 7

SEQUENTIAL CONVEX PROGRAMMING WITH DISCRETE LOGIC

This chapter presents a nonconvex optimization algorithm for solving optimal control problems that involve discrete logic constraints. Traditional models of these constraints require binary variables and mixed-integer programming, which is prohibitively slow and computationally expensive. We target a fast solution that is capable of real-time implementation onboard spacecraft. To do so, we introduce a novel algorithm that blends sequential convex programming and numerical continuation into a single iterative solution process. Inside the algorithm, discrete logic constraints are approximated as smooth functions, and a homotopy parameter governs the accuracy of this approximation. As the algorithm converges, the homotopy parameter is updated such that the smooth approximations enforce the exact discrete logic. The effectiveness of this approach is numerically demonstrated for a realistic rendezvous scenario inspired by the Apollo Transposition and Docking maneuver. In under 15 seconds of cumulative solver time, the algorithm is able to reliably find difficult fuel optimal trajectories that obey the following discrete logic constraints: thruster minimum impulse-bit, range-triggered approach cone, and range-triggered plume impingement. The optimized trajectory uses significantly less fuel than NASA design targets. Within the context of this thesis (see Figure 7.1), this chapter falls into the iterative algorithm category for the most difficult kind of problems: nonconvex and not convexifiable through mathematical modeling (as in Chapter 6).

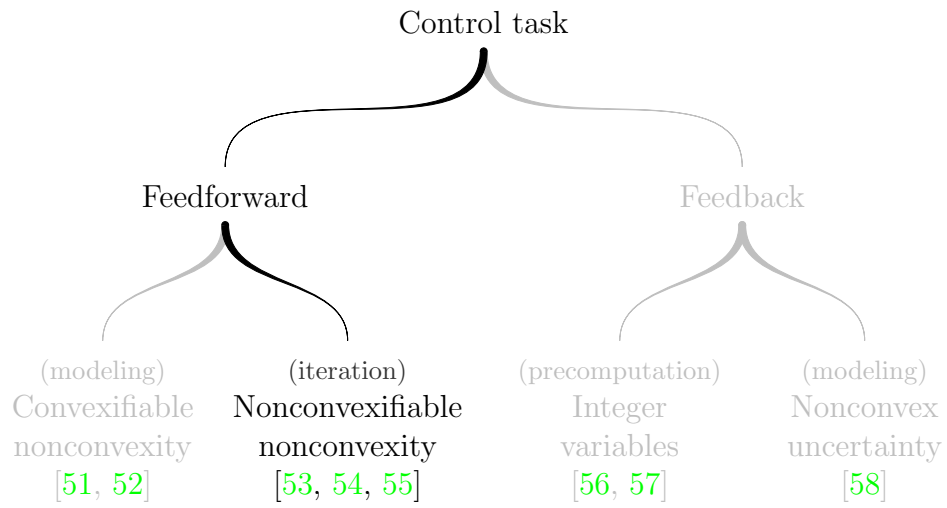


Figure 7.1: This chapter presents an iterative approach for real-time optimization-based trajectory generation for the most difficult of systems: neither convex nor convexifiable.

7.1 Background

Space programs have historically been deemed mature once they establish the ability to perform rendezvous and docking operations [175]. Some of the earliest programs of the United States and the Soviet Union (e.g., Gemini and Soyuz) had as their explicit goal to demonstrate the capability of performing rendezvous, proximity operations, and docking maneuvers. The ultimate objective to land humans on the moon drove the need for these capabilities. Beyond the lunar missions of the 1960s, rendezvous and docking continued to be a core technology required to construct and service space stations that were built in low Earth orbit [176]. The Shuttle program was comprised of dozens of missions for which rendezvous (and more generally, proximity operations) was an explicit mission objective. The core technology used to achieve rendezvous and docking has remained largely unchanged in the decades since the earliest maneuvers were successful. While this heritage technology is far from obsolete, it has been stated that it may be unable to meet the requirements of future missions [175]. A driving force that will require new methods is the need for a system that can perform *fully autonomous* rendezvous in several domains (e.g., low Earth orbit, low

lunar orbit, etc.) [177]. Several vehicles capable of autonomous docking are either already operational or in development, ranging from large vehicles such as the SpaceX Crew Dragon, Soyuz, and Orion [175, 177, 178], to smaller robotic vehicles for clearing orbital debris [179, 180, 181].

The objective of this chapter is to present a framework for designing autonomous docking trajectories that accurately reflect the capabilities and constraints that have been historically prevalent for proximity operation missions. We view the problem as a trajectory generation problem, and compute what would be implemented as a guidance solution. In particular, we show how to model challenging *discrete logic* constraints within a continuous optimization framework. The resulting open-source algorithm is numerically demonstrated to be sufficiently fast for ground-based use, and has the potential to be real-time capable if implemented in a compiled programming language.

The open-loop generation of spacecraft docking trajectories using optimization-based methods is a relatively new field spawned by the shift towards autonomy [49]. Open-loop trajectory generation computes a complete start-to-finish trajectory, and leaves robust tracking to closed-loop feedback control. This is slightly different from receding horizon or model predictive control (MPC), which has received more attention in the rendezvous and docking literature [50, 182, 183, 184]. In [185, 186] the authors discuss both time- and fuel-optimal solutions with a focus on problem formulations that are conducive to on-board implementation. Their study offers an insightful view on the structure of optimality at the cost of a simplified problem statement and omission of state constraints. In [187], lossless convexification is used to generate fuel-optimal docking trajectories which account for non-convex thrust and plume impingement constraints, albeit the thrust is not allowed to turn off. In [173], lossless convexification allows to generate bang-bang controls for minimum-time spacecraft rendezvous using differential drag, however without state constraints or spacecraft attitude dynamics. In [188], an optimization framework is used to impose safety-based constraints in the case of anomalous behaviour (including thruster failure) by introducing a suboptimal convex program to design safe trajectories which approximate a non-convex mixed-integer problem

using a new set of "safe" inputs. Along the same lines of mixed-integer programming, [183] solves a fuel-optimal problem subject to thrust plume and collision avoidance constraints. The authors introduce several heuristic techniques in order to fit the problem within the scope of mixed-integer linear programming, but still observe rather long solve times (over 40 minutes in some cases). More recently, [189] studied a multi-phase docking problem with several state constraints. The authors use binary variables to impose different constraints during each phase, and propose an iterative solution method with closed-form update rules. Beyond the use of mixed-integer methods, [190] proposes a randomized optimization method similar to the A^* method, while [184] proposes a convex one-norm regularized MPC solution.

Notably, the aforementioned references do not consider the spacecraft attitude during trajectory generation and do not explicitly account for what is referred to as the minimum impulse-bit (MIB) of the reaction control thrusters that are used to realize the trajectories. The latter constraint refers to the fact that impulsive chemical thrusters cannot fire for an arbitrarily short duration, since there is some minimum pulse width that is inherent to the hardware. Hartley et al. [184] acknowledge this issue, but instead of explicitly enforcing the constraint, the authors use a one-norm penalty term to discourage violation of the constraint (i.e., a soft constraint). Our view is that both attitude and the MIB constraint are critical for close proximity operations such as the terminal phase of rendezvous and docking, where two spacecraft are maneuvering close to each other. We thus target an algorithm that can efficiently incorporate both effects.

7.1.1 Contributions

This chapter's contribution is a numerical optimization algorithm to solve trajectory generation problems involving a general class of discrete logic constraints. The algorithm is based on a novel arrangement of two core technologies: sequential convex programming (SCP) and numerical continuation. SCP is a trust region method for solving general nonconvex optimal control problems [49]. However, it is incapable of handling discrete constraints in their pure (integer) form. By using a homotopy map based on the multinomial logit function, we

embed continuous approximations of discrete constraints into the SCP framework, a process known as continuous embedding [191]. The homotopy map is then updated via a numerical continuation scheme, which transforms an initial coarse approximation into an artbirarily precise representation of the discrete logic. Herein lies our key innovation: we run SCP and numerical continuation *in parallel*, rather than the traditional sequenced approach where one homotopy update is followed by a full SCP solve. The resulting algorithm is shown to converge quickly and reliably for a representative terminal rendezvous problem inspired by the Apollo Transposition and Docking maneuver. The problem involves the following major constraints: full six degree of freedom (DOF) dynamics, thruster minimum impulse-bit, range-triggered approach cone, and range-triggered plume impingement.

This chapter represents a significant upgrade in terms of both runtime performance and convergence reliability over the same authors' previous publication on SCP-based rendezvous [54]. In relation to existing literature, the method of this chapter is closest to the recently published relaxed autonomous switched hybrid system (RASHS) and composite smooth control (CSC) algorithms [192, 193, 194]. Both methods also use homotopy and numerical continuation to enforce discrete logic constraints. While they handle Boolean `and` logic, our approach models Boolean `or` logic. In combination with RASHS and CSC, our work thus extends homotopy to general Boolean logic using any combination of logic gates. Another difference is that RASHS and CSC are devised for the indirect family of optimization methods that solve Pontryagin's necessary conditions of optimality [39, 40]. Our method is devised for SCP, which is a direct method that solves a discretized version of the original optimal control problem [85]. A more detailed comparison of the methods is given in Section 7.3.2.

7.1.2 Outline

The chapter is structured as follows. In Section 7.2 we formulate the rendezvous problem that is to be solved, but which is not efficiently solvable in its raw form. Section 7.3 then describes the homotopy map which can model a generic class of discrete logic in a smooth way. Using this smoothing, Section 7.4 describes our key contribution: an algorithm that

can solve nonconvex optimal control problems with discrete logic. The effectiveness of the approach is numerically demonstrated in Section 7.5 for a realistic scenario based on the historical Apollo Transposition and Docking maneuver.

7.2 Rendezvous Problem Formulation

In this section we formulate a trajectory generation problem where the objective is to guide a chaser spacecraft to dock with a passive target spacecraft in a predetermined orbit. We assume that the maneuver happens in low Earth orbit (LEO) and that the target's orbit is circular. The chaser's dynamics are defined in Section 7.2.1, the actuator model is described in Section 7.2.2, and the rendezvous constraints are given in Sections 7.2.3, 7.2.4, and 7.2.5. Section 7.2.6 gives a complete formulation of the free-final time nonconvex optimal control problem which, if solved, generates a fuel-optimal rendezvous trajectory. Most notably, because the constraints in Sections 7.2.2, 7.2.3, and 7.2.4 involve discrete logic, the problem is not readily solvable by traditional continuous optimization methods.

7.2.1 Chaser Spacecraft Dynamics

We begin by writing down the equations of motion for the chaser spacecraft. It is assumed that the chaser is a 6-DOF rigid body vehicle with constant mass. The latter assumption is accurate for our ultimate numerical application to the Apollo Transposition and Docking maneuver, whose fuel mass allocation is 32 kg, corresponding to about 0.1% of the total Apollo Command and Service Module (CSM) vehicle mass [195].

The general setup is illustrated in Figure 7.2. First, a Local-Vertical Local-Horizontal (LVLH) frame is placed at the target's center of mass (COM). Assuming that the target is in a circular orbit, and because separation distances during the final stages of rendezvous are relatively small, we can write the translation dynamics in this frame according to the Clohessy-Wiltshire-Hill equations [196]. For the attitude dynamics, a body frame is affixed to the chaser's COM. Apart from the non-inertial forces of the relative motion dynamics in the LVLH frame, the only forces acting on the chaser are the ones generated by its system

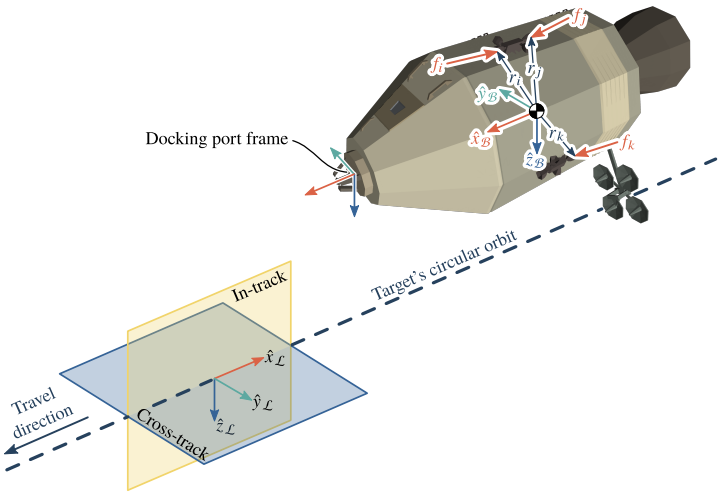


Figure 7.2: Illustration of the LVLH and body frames, and several of the spacecraft RCS thrusters. Each thruster is represented by its position r_i relative to the COM and its thrust vector f_i, both quantities being expressed in the body frame. The docking port also has an attached frame that is used for computing the terminal state.

of reaction control system (RCS) thrusters. As shown in Figure 7.2, the force produced by each thruster is defined by its point of application r_i and its vector f_i, both of which are expressed in the $\mathcal{F}_\mathcal{B}$ frame. Altogether, the 6-DOF equations of motion of the chaser in the LVLH frame are written as follows:

$$\dot{p}(t) = v(t), \tag{7.1a}$$

$$\dot{v}(t) = \frac{1}{m}\sum_{i=1}^{n_{\text{rcs}}} q(t) \otimes f_i \otimes q(t)^* + a_{\text{LVLH}}\big(p(t), v(t)\big), \tag{7.1b}$$

$$\dot{q}(t) = \frac{1}{2} q(t) \otimes \omega(t), \tag{7.1c}$$

$$\dot{\omega}(t) = J^{-1}\Big[\sum_{i=1}^{n_{\text{rcs}}} r_i \times f_i(t) - \omega(t) \times \big(J\omega(t)\big)\Big], \tag{7.1d}$$

where the acceleration due to relative motion is given by:

$$a_{\text{LVLH}}(p, v) = \big(-2n_o \hat{z}_{\mathcal{L}}^{\mathsf{T}} v \big) \hat{x}_{\mathcal{L}} + \big(-n_o^2 \hat{y}_{\mathcal{L}}^{\mathsf{T}} r \big) \hat{y}_{\mathcal{L}} + \big(3n_o^2 \hat{z}_{\mathcal{L}}^{\mathsf{T}} r + 2n_o \hat{x}_{\mathcal{L}}^{\mathsf{T}} v \big) \hat{z}_{\mathcal{L}}, \tag{7.2}$$

where $n_o \in \mathbb{R}$ is the orbital mean motion. The translation dynamics are encoded by $p \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$, which are LVLH frame vectors denoting the position and velocity of $\mathcal{F}_{\mathcal{B}}$ with respect to $\mathcal{F}_{\mathcal{L}}$. The attitude dynamics are encoded by a quaternion $q \in \mathbb{Q}$ and an angular velocity $\omega \in \mathbb{R}^3$. We use the Hamilton quaternion convention and represent q as a four-element vector [197]. The quaternion thus represents a frame transformation from $\mathcal{F}_{\mathcal{B}}$ to $\mathcal{F}_{\mathcal{L}}$, or (equivalently) the rotation of a vector in the $\mathcal{F}_{\mathcal{L}}$ frame. The ω vector corresponds to the angular velocity of $\mathcal{F}_{\mathcal{B}}$ with respect to $\mathcal{F}_{\mathcal{L}}$, expressed as a vector in the $\mathcal{F}_{\mathcal{B}}$ frame. Altogether, the vehicle state is encoded by $x = \big[p; v; q; \omega\big] \in \mathbb{R}^{13}$.

7.2.2 Impulsive Thrust Model

As mentioned in the previous section, the chaser is controlled by a system of n_{rcs} chemical RCS thrusters. In accordance with our ultimate application to the Apollo CSM spacecraft, we assume that each thruster is able to deliver a constant thrust for a variable duration of time [198, 199, 200]. This is known as pulse-width modulation (PWM).

Let us temporarily focus the discussion on the force produced by the i-th thruster. Let F_{rcs} denote the constant thrust level generated when the thruster is active (i.e., "firing"), and let Δt_i be the firing or pulse duration. If the thruster fires for a very short duration relative to the bandwidth of the chaser's dynamics, then we can approximate the state as being constant over the firing interval. We can furthermore shrink the firing interval to zero, as long as we increase the thrust level to maintain a constant net impulse that is imparted on the chaser. This is illustrated in Figure 7.3, where an original 500 ms rectangular pulse is reduced down to 100 ms. In the limit as Δt_i is reduced to zero, the thrust signal becomes a scaled Dirac delta function:

$$f_i(t) = \Delta t_i F_{\text{rcs}} \delta(t). \tag{7.3}$$

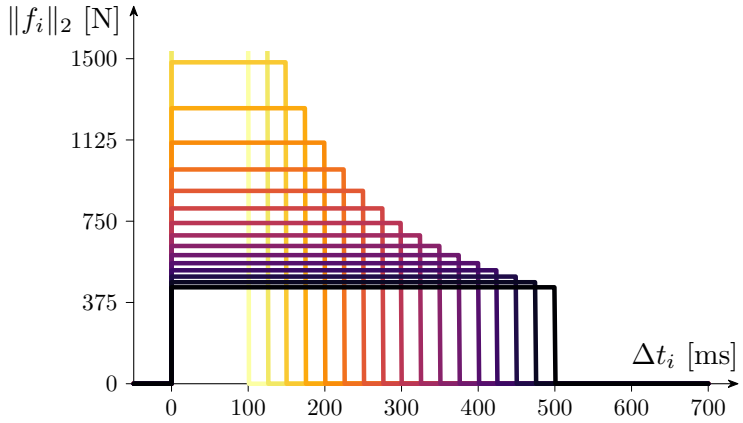


Figure 7.3: Illustration of reducing the rectangular thrust pulse duration from an original value of $\Delta t_i = 500$ ms, while maintaining a constant net impulse. The result is an increasing thrust level, while the area below the curve remains constant.

This model is an accurate enough approximation for generating long duration trajectories with relatively few intermittent control interventions. By neglecting state variation over the firing duration, the model furthermore has a significant computational advantage when it comes to linearizing, discretizing, and simulating the dynamics for the solution process in Section 7.4. We emphasize, however, that (7.3) is a model which we use for computation alone. In the physical world, we still expect the thrusters to fire for a finite duration and at their design (finite) thrust level.

The discussion so far has centered around a single pulse that occurs at $t = 0$ s. We now generalize this model to the trajectory generation context. Begin by fixing a control interval $t_c > 0$ that corresponds to the "silent" time interval between thruster firings. Furthermore, let N_c be the total number of control opportunities during the trajectory. This means that the trajectory lasts for $N_c t_c$ seconds. Note that no firing occurs at the final time instant, since that would lead to undesirable control at the moment of docking. Thus, a thruster can be activated only at the time instances $(k-1)t_c$ where $k = 1, 2, \ldots, N_c$. To keep the notation short, we define $k' \equiv k - 1$ for any general index k. Thus, the thrust signal for the

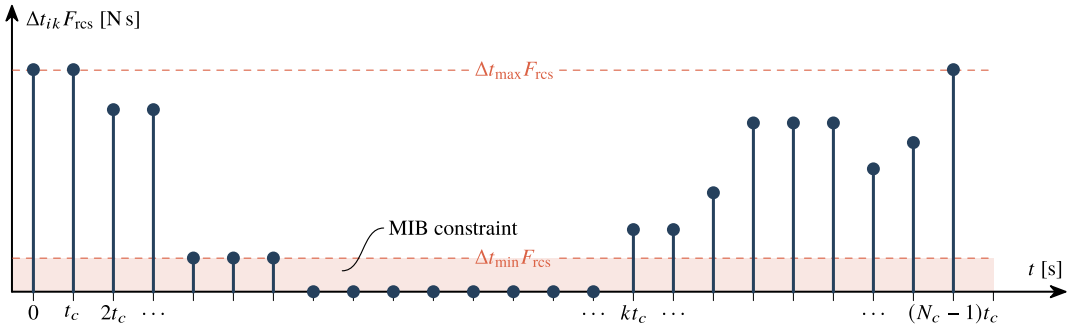


Figure 7.4: Control history example that is compatible with the impulsive thrust model (7.4) and the pulse duration constraint (7.5). Most notably, no impulse occurs in the MIB keep-out zone between 0 and $\Delta t_{\min} F_{\text{rcs}}$ N s. This region represents impulses which the RCS system cannot reproduce. A feasible rendezvous trajectory cannot ask for $\Delta t_{ik} < \Delta t_{\min}$.

i-th thruster can be formally written as:

$$f_i(t) = \sum_{k=1}^{N_c} \Delta t_{ik} F_{\text{rcs}} \delta\big(t - k' t_c\big) \hat{f}_i, \tag{7.4}$$

where Δt_{ik} is the pulse duration for the i-th thruster at the k-th control interval, and $\hat{f}_i$ is the thrust direction unit vector in the $\mathcal{F}_{\mathcal{B}}$ frame. Due to delays in on-board electronics and residual propellant flow downstream of the injector valves [200, pp. 2.5-16 to 2.5-18], the pulse duration is lower bounded such that $\Delta t_{ik} \geq \Delta t_{\min}$. This is known as a minimum impulse-bit (MIB) constraint. Other propulsion and RCS parameters, such as engine service life and damage to engine materials, impose an upper bound $\Delta t_{ik} \leq \Delta t_{\max}$. As a result, the pulse duration must satisfy the following nonconvex constraint:

$$\Delta t_{ik} \in \{0\} \cup [\Delta t_{\min}, \Delta t_{\max}]. \tag{7.5}$$

Figure 7.4 illustrates a typical control history that we can expect from the model (7.4) subject to the constraint (7.5). The salient feature of this control history is that the thruster is either silent, or firing with a minimum impulse.

7.2.3 Plume Impingement Constraint

A plume impingement constraint prevents the RCS thrusters from firing and potentially damaging the target spacecraft. Naturally, this constraint is only required once the chaser is close enough to the target. Let $\mathcal{I}_{\text{fr}}$ denote the indices of forward-facing thrusters that are physically pointed along the $+\hat{x}_{\mathcal{B}}$ axis in Figure 7.2. Due to the physics of rendezvous and the approach cone constraint of the next section, it is reasonable to assume that large-angle maneuvering is finished by the time the spacecraft is close to the target. Thus, when the plume impingement constraint is relevant, the chaser is approximately facing the target. This yields a simple plume impingement heuristic: shut off the $\mathcal{I}_{\text{fr}}$ thrusters when the chaser is inside a so-called plume impingement sphere of radius r_{plume} centered at the target. This can be formally stated as the following implication:

$$\|p(k't_c)\|_2 \leq r_{\text{plume}} \;\Rightarrow\; \Delta t_{ik} = 0 \text{ for all } i \in \mathcal{I}_{\text{fr}}. \tag{7.6}$$

7.2.4 Approach Cone Constraint

The approach cone constraint bounds how much the chaser spacecraft can maneuver once it gets close enough to the target. It has the direct effect of bounding transverse motion along the $\hat{y}_{\mathcal{L}}$ and $\hat{z}_{\mathcal{L}}$ LVLH axes in Figure 7.2. In practice, it also bounds all other maneuvering, including attitude rates, except for translation motion along $-\hat{x}_{\mathcal{L}}$.

Figure 7.5 illustrates our implementation of an approach cone. Because we do not want to restrict the chaser's motion far away from the target, the constraint only gets applied once the chaser enters a so-called approach sphere of radius r_{appch}. When this condition is satisfied, the chaser's position is constrained to lie in a cone that emanates from the target along $+\hat{x}_{\mathcal{L}}$ and has an opening half-angle θ_{appch}. Formally, the approach cone constraint can be written as the following implication:

$$\|p(t)\|_2 \leq r_{\text{appch}} \;\Rightarrow\; \hat{x}_{\mathcal{L}}^{\mathsf{T}} p(t) \geq \|p(t)\|_2 \cos(\theta_{\text{appch}}). \tag{7.7}$$

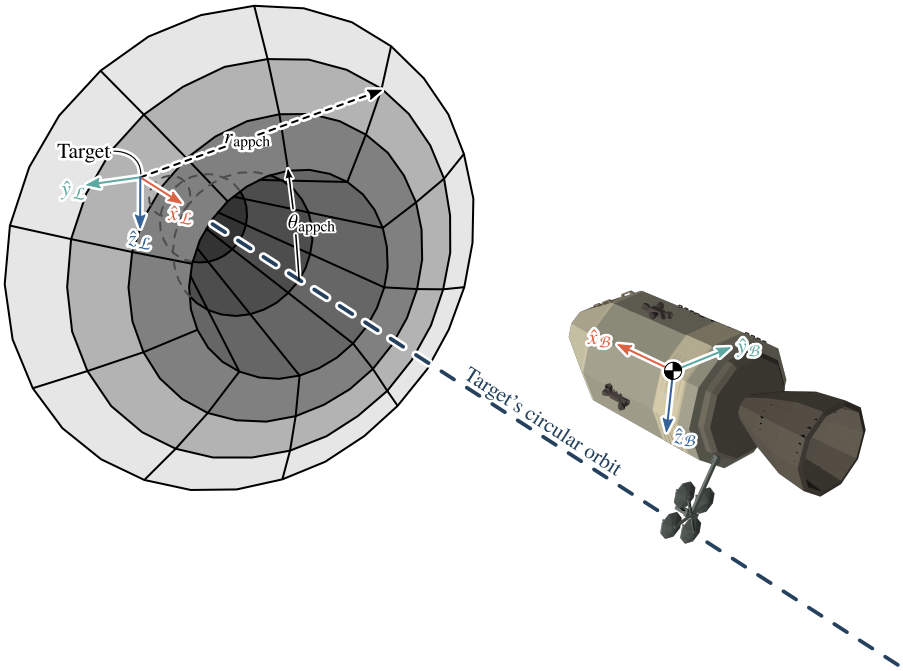


Figure 7.5: Illustration of an approach cone with its apex at the target spacecraft. The approach cone half-angle is θ_{appch}. The chaser's position is constrained to lie inside the cone when the chaser enters an approach sphere of radius r_{appch} centered at the target (only part of the sphere is drawn).

7.2.5 Boundary Conditions

We consider the case of terminal rendezvous between two fixed boundary conditions: some initial chaser state and a terminal "docked" state. In particular, let $x_0 = \left[p_0; v_0; q_0; \omega_0\right] \in \mathbb{R}^{13}$ and $x_f = \left[p_f; v_f; q_f; \omega_f\right] \in \mathbb{R}^{13}$ correspond to the initial and terminal desired states. The terminal position and attitude are computed according to the relative geometry of the target and chaser docking ports and the chaser COM. For simplicity, assume that the target docking port is centered at the origin of $\mathcal{F}_{\mathcal{L}}$ and points along $+\hat{x}_{\mathcal{L}}$. Generalizing this assumption to a non-collocated docking port is possible, but does not represent anything particularly novel or challenging for our algorithm. When docked, let $q_\ell \in \mathbb{Q}$ denote the chaser docking port's attitude with respect to the target docking port. As illustrated in Figure 7.2, q_ℓ is a simple yaw around $+\hat{z}_{\mathcal{L}}$ by 180°. Furthermore, let $q_{\text{dp}} \in \mathbb{Q}$ and $p_{\text{dp}} \in \mathbb{R}^3$ be the rotation and position

of the chaser docking port relative to $\mathcal{F}_\mathcal{B}$. The terminal position and attitude are then given by:

$$q_f = q_\ell \otimes q_{\text{dp}}^*, \tag{7.8a}$$

$$p_f = -q_f \otimes p_{\text{dp}} \otimes q_f^*. \tag{7.8b}$$

For a rendezvous trajectory that lasts t_f seconds, the boundary conditions we impose are:

$$x(0) = x_0, \quad x(t_f) + \Delta x_f = x_f, \tag{7.9}$$

where $\Delta x_f = \left[\Delta p_f; \Delta v_f; \Delta q_f; \Delta \omega_f\right] \in \mathbb{R}^{13}$ relaxes of the terminal boundary condition. This is necessary because the MIB constraint from Figure 7.4 makes it impossible to fine-tune the trajectory to arbitrary precision. In general, some terminal error has to occur. As long as this error is small, it will be safely absorbed by the mechanical design of the docking port. The required tolerances can be found in the spacecraft's manual. For example, for the Apollo CSM the complete list is given in [198, Section 3.8.2.3]. Because it is good practice to leave a margin of error for feedback controllers, we will constrain Δx_f to a much smaller value than what the docking mechanism can tolerate. The following constraints restrict the size of Δx_f to user-specified tolerances:

$$\|\Delta p_f\|_\infty \leq \varepsilon_{p_f}, \quad \hat{x}_\mathcal{L}^\mathsf{T} \Delta p_f = 0, \tag{7.10a}$$

$$\|\Delta v_f\|_\infty \leq \varepsilon_{v_f}, \tag{7.10b}$$

$$q(t_f)^\mathsf{T} q_f \geq \cos(\varepsilon_{q_f}/2), \tag{7.10c}$$

$$\|\Delta \omega_f\|_\infty \leq \varepsilon_{\omega_f}. \tag{7.10d}$$

The terminal position along $\hat{x}_\mathcal{L}$ is made exact since contact along $\hat{x}_\mathcal{L}$ is required for docking. Furthermore, it is always possible to satisfy by adjusting t_f. The terminal attitude is constrained by (7.10c) in terms of an error quaternion, and says that the angular deviation

from q_f about any axis must be no larger than angle ε_{q_f}.

7.2.6 Basic Rendezvous Problem

Our goal is to compute a fuel-optimal rendezvous trajectory, which means that it is desirable to keep the pulse durations Δt_{ik} as short and as sparse as possible. An appropriate optimization cost function is simply the sum of pulse durations for all thrusters and control intervals:

$$J_{\text{fuel}} = \Delta t_{\max}^{-1} \sum_{i=1}^{n_{\text{rcs}}} \sum_{k=1}^{N_c} \Delta t_{ik}, \tag{7.11}$$

where the normalization by $\Delta t_{\max}$ is useful when (7.11) is mixed with other costs for the solution process in Section 7.4. Note that (7.11) is effectively a one-norm penalty on the pulse durations. This encourages the optimal pulse history to be sparse, which goes part of the way towards discouraging MIB constraint violation [34, 184].

We can now summarize the above sections by writing the full rendezvous optimization problem that has to be solved. We call this the basic rendezvous problem (BRP). Starting now and throughout the rest of the article, the time argument will be omitted whenever it does not introduce ambiguity.

$$\min_{x, \Delta t, t_f} \; J_{\text{fuel}} \tag{7.12a}$$

$$\text{s.t. } \dot{p} = v, \tag{7.12b}$$

$$\dot{v} = \frac{1}{m} \sum_{i=1}^{n_{\text{rcs}}} q \otimes f_i \otimes q^* + a_{\text{LVLH}}\big(p, v\big), \tag{7.12c}$$

$$\dot{q} = \frac{1}{2} q \otimes \omega, \tag{7.12d}$$

$$\dot{\omega} = J^{-1}\Big[\sum_{i=1}^{n_{\text{rcs}}} r_i \times f_i - \omega \times \big(J\omega\big)\Big], \tag{7.12e}$$

$$\Delta t_{ik} \in \{0\} \cup [\Delta t_{\min}, \Delta t_{\max}], \tag{7.12f}$$

$$\|p(k' t_c)\|_2 \leq r_{\text{plume}} \;\Rightarrow\; \Delta t_{ik} = 0 \text{ for all } i \in \mathcal{I}_{\text{fr}}, \tag{7.12g}$$

$$\|p\|_2 \leq r_{\text{appch}} \;\Rightarrow\; \hat{x}_{\mathcal{L}}^{\mathsf{T}} p \geq \|p\|_2 \cos(\theta_{\text{appch}}), \tag{7.12h}$$

$$x(0) = x_0, \quad x(t_f) + \Delta x_f = x_f, \tag{7.12i}$$

$$\|\Delta p_f\|_\infty \leq \varepsilon_{p_f},\; \hat{x}_{\mathcal{L}}^{\mathsf{T}} \Delta p_f = 0,\; \|\Delta v_f\|_\infty \leq \varepsilon_{v_f}, \tag{7.12j}$$

$$q(t_f)^{\mathsf{T}} q_f \geq \cos(\varepsilon_{q_f}/2),\; \|\Delta \omega_f\|_\infty \leq \varepsilon_{\omega_f}. \tag{7.12k}$$

The BRP is a continuous-time, free-final time, nonconvex optimal control problem. It is not efficiently solvable on a computer for three main reasons [49]:

1. Continuous-time problems have an infinite number of DOFs in the optimized control signal. However, numerical optimization algorithms are restricted to a finite number of DOFs;

2. The problem has nonlinear dynamics, which means that a nonconvex numerical solver must be used. However, nonconvex solvers require expert initial guesses and generally do not converge quickly and reliably enough for safety-critical applications [32, 34];

3. The constraints (7.12f)-(7.12h) contain discrete `if-else` logic. This is traditionally handled by mixed-integer programming (MIP), which has poor computational complexity and does not scale well to large problems [157].

We will begin by resolving the third issue through a continuous embedding approach in the next section. The first two issues will then be tackled in Section 7.4.

7.3 Continuous Embedding of Discrete Logic

We now consider the problem of computationally efficient modeling the discrete logic constraints (7.12f)-(7.12h). This model along with the associated numerical continuation solution method in Section 7.4 are the main contributions of this chapter. We begin in Section 7.3.1 with a motivation for why a new approach to handling discrete logic is necessary.

Our continuous embedding algorithm is then described in general terms in Section 7.3.2. Finally, Sections 7.3.3, 7.3.4, and 7.3.5 specialize the approach to the discrete logic constraints (7.12f)-(7.12h).

7.3.1 Motivation

The traditional way of handling discrete logic in an optimization problem is through the use of binary variables [117, 158]. As a concrete example, consider the plume impingement constraint (7.12g). Let $z_{\text{plume}}(t) : [0, t_f] \to \{0, 1\}$ denote a binary variable trajectory that is also to be optimized. Let M_{plume} be a large positive value that bounds all possible values of $\|p(t)\|_2$ that can occur during a rendezvous trajectory. For example, $M_{\text{plume}} = 10\|p_0\|_2$ is a reasonable choice. The plume impingement constraint (7.12g) can then be equivalently written as:

$$z_{\text{plume}}(k't_c) r_{\text{plume}} \leq \|p(k't_c)\|_2 \leq r_{\text{plume}} + z_{\text{plume}}(k't_c) M_{\text{plume}}, \tag{7.13a}$$

$$0 \leq \Delta t_{ik} \leq z_{\text{plume}}(k't_c) \Delta t_{\max} \text{ for all } i \in \mathcal{I}_{\text{fr}}. \tag{7.13b}$$

Looking at (7.13), z_{plume} can be interpreted as follows: the chaser is outside the plume impingement sphere if and only if $z_{\text{plume}} = 1$. When the chaser is inside this sphere, the only feasible choice is $z_{\text{plume}} = 0$, and (7.13b) shuts off the forward-facing thrusters.

A similar formulation can be used to model the MIB and approach cone constraints (7.12f) and (7.12h), resulting in a MIP formulation. Unfortunately, this approach has an issue when it comes to actually solving Problem 7.12: mixed-integer optimization algorithms are generally too slow for real-time applications, are computationally expensive, and do not scale well to large problem sizes [57, 157]. When compounded by the fact that this formulation introduces new nonconvex constraints (e.g., the position norm lower bound in (7.13a)), it becomes clear that the MIP approach is not a workable real-time solution method for Problem 7.12.

Several methods have been proposed in recent years to replace MIP with a real-time

capable approach. As we covered in Chapter 6, recent theoretical results (which include this thesis) have demonstrated that a lossless relaxation can solve certain classes of problems with discrete logic constraints on the control variable [52, 142]. This approach is lucrative because it requires solving only a single convex problem. Some versions of the method can handle restricted forms of nonlinear dynamics and convex constraints [49, 61, 116]. However, the method does not apply to the full generality of Problem 7.12, which involves more complicated nonlinear dynamics as well as discrete logic constraints on the state.

A separate family of solution methods has been proposed to handle discrete logic constraints using sequential convex programming (SCP) [49]. The methods define so-called state triggered constraints (STCs) that can embed general discrete logic into a continuous optimization framework [50, 201, 202]. Two equivalent forms of STCs have been proposed, based on a slack variable [67] and based on a multiplicative coefficient that is motivated by the linear complementarity problem [70]. STCs have also been extended to handle quite general logical combinations of `and` and `or` gates [120, 203]. In fact, the author has applied STCs to solve a version of Problem 7.12, with the results available in [54]. In the latter work it was observed that STCs run into an issue called *locking* for the MIB constraint (7.12f) [54, Definition 1]. As described in Section 7.4, SCP works by iteratively refining an approximate solution of Problem 7.12. In brief terms, locking means that once the algorithm chooses $\Delta t_{ik} = 0$ at a particular iteration, it is unable to change the value to $\Delta t_{ik} \in [\Delta t_{\min}, \Delta t_{\max}]$ at later iterations. The effect is that the algorithm is susceptible to getting into a "corner" where it is unable to use thrusters if they become needed at later refinements of the rendezvous trajectory. The consequence is failure to generate a feasible trajectory. There is currently no known remedy for constraints that exhibit locking in the STC formulations of [67, 70].

For reasons that are well documented in past literature, we view SCP as one of the best frameworks for the real-time solution of nonconvex trajectory generation problems [49, 50, 70]. Thus, our primary motivation is to devise a new general method that is free from locking and that can embed discrete logic into an SCP-based continuous optimization framework.

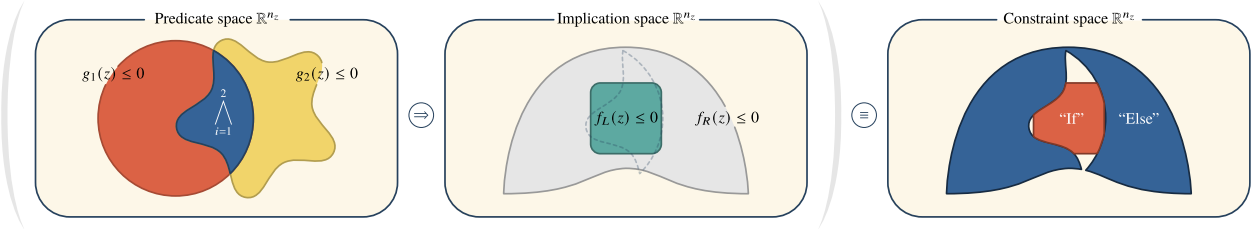


Figure 7.6: Pictorial representation of the `if-else` discrete logic constraint (7.14). In the predicate space, the functions g_i individually form sublevel sets where they take nonpositive values. In the implication space, the constraint functions f_L and f_R also form their own sublevel sets. Note that these sets can generally be disjoint. The net `if` constraint is obtained by intersecting the sublevel set of f_L with the `and` combination. The net `else` constraint is obtained by intersecting the sublevel set of f_R with the set complement of the `and` combination.

7.3.2 Continuous Embedding Algorithm

We now develop a homotopy-based method to systematically handle `if-else` discrete logic constraints of the following form:

$$\text{"If"} \quad L(z) \triangleq \bigwedge_{i=1}^{n_g} \big(g_i(z) \leq 0\big) \;\Rightarrow\; f_L(z) \leq 0, \tag{7.14a}$$

$$\text{"Else"} \quad R(z) \triangleq \bigvee_{i=1}^{n_g} \big(g_i(z) > 0\big) \;\Rightarrow f_R(z) \leq 0, \tag{7.14b}$$

where $z \in \mathbb{R}^{n_z}$ is a generic placeholder for one or several optimization variables. The functions $g_i : \mathbb{R}^{n_z} \to \mathbb{R}$ are called predicates, and the functions $f_L : \mathbb{R}^{n_z} \to \mathbb{R}^{n_f}$ and $f_R : \mathbb{R}^{n_z} \to \mathbb{R}^{n_f}$ are implication constraints to be enforced when the corresponding expression's left-hand side is `true`. For (7.14a) this is a combination of `and` gates, whereas for (7.14b) it is a combination of `or` gates with the predicate inequalities reversed. We may thus see (7.14) in the following light: enforce $f_L \leq 0$ when *all* the predicates are nonpositive, or enforce $f_R \leq 0$ when *any* predicate is positive.

One can show using De Morgan's theorem that $R(z) = \neg L(z)$. As a result, the implications in (7.14) indeed form an `if-else` pair in the sense that exactly one of f_L and f_R is

enforced at any given instant. The situation is illustrated in Figure 7.6. By using 1 to denote `true`and 0 to denote `false`, we have the complementarity relationship $R(z) = 1 - L(z)$. Using this property, (7.14) can be stated in the following equivalent ways:

$$L(z)f_L(z) + \big[1 - L(z)\big]f_R(z) \leq 0, \tag{7.15a}$$

$$\big[1 - R(z)\big]f_L(z) + R(z)f_R(z) \leq 0. \tag{7.15b}$$

Because (7.15) involves discrete elements (i.e., the `and` and `or` gates), it cannot be readily included in a continuous optimization problem. As mentioned in the previous section, STCs are one possible way to circumvent the issue, however they exhibit locking in the particular case of the MIB constraint (7.12f). An alternative approach is to replace either L or R by a smooth approximation, and to apply a numerical continuation scheme to iteratively improve the approximation until some arbitrary precision [32]. We take this latter approach, and begin with a brief description of two existing methods.

Existing Homotopy Methods

Homotopy is the core idea behind the recent relaxed autonomous switched hybrid system (RASHS) and composite smooth control (CSC) algorithms [192, 193, 194]. Both algorithms model the constraint (7.15a) whereby the `and` combination is approximated by a sigmoid. In brief terms, let $\sigma_\kappa(w) : \mathbb{R} \to \mathbb{R}$ represent a sigmoid function which approaches one for negative arguments and zero for positive arguments. The transition point occurs at $w = 0$ and the homotopy parameter $\kappa > 0$ (also known as a sharpness parameter) regulates how quickly the transition happens. As κ increases, σ_κ approaches a "step down" function. This allows RASHS and CSC to model L as follows:

$$L(z) \approx \tilde{L}_\kappa(z) \triangleq \prod_{i=1}^{n_g} \sigma_\kappa\big(g_i(z)\big). \tag{7.16}$$

By replacing L with $\tilde{L}_\kappa$ in (7.15a), the RASHS and CSC methods can model discrete

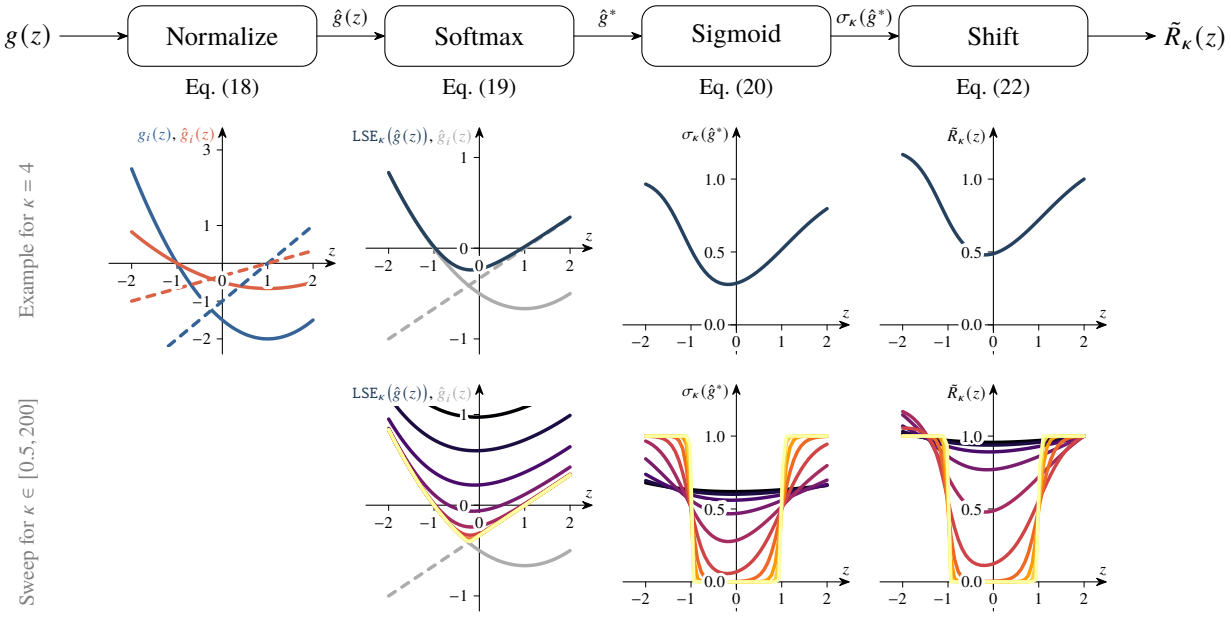


Figure 7.7: Illustration of the four stages comprising our smooth approximation of the `or` combination R in (7.14b). The top row of plots shows a particular example of the smoothing pipeline for homotopy parameter $\kappa = 4$ and a simple set of two predicates. The second row of plots shows how the approximation becomes arbitrarily precise as κ increases.

logic in a smooth way that is conducive for continuous optimization. By using numerical continuation to progressively increase κ, the methods can enforce the discrete logic constraint (7.15a) with arbitrary accuracy.

Our Homotopy Method

Our method for imposing (7.14) is centered around a smooth approximation of the alternative constraint (7.15b) using a multinomial logit function [204]. We thus view our approach as a "dual" formulation to RASHS and CSC: instead of modeling the `and` combination of (7.14a), we model its complement (7.14b). A noteworthy benefit of this approach is the ability to model `or` combinations, whereas RASHS and CSC both appear to be compatible only with `and` logic. Our method is therefore an extension of the ideas in RASHS and CSC. Although we do not develop the full theory here, our method together with (7.16) can model arbitrary

combinations of `and` and `or` logic. This extends smooth modeling of discrete logic to its full generality.

We break down the smooth approximation of R into four computational "stages". The reader may follow along with the help of the illustration in Figure 7.7. Begin with the raw data, which are the individual predicate values $g_i(z)$. For convenience, let $g(z) : \mathbb{R}^{n_z} \to \mathbb{R}^{n_g}$ be the concatenated vector of predicates. The first stage is to normalize $g(z)$ by the expected maximum value of the predicates:

$$g_{\max} \triangleq \max_z \|g(z)\|_\infty, \tag{7.17}$$

where z is understood to be taken from the set of all reasonable values for Problem 7.12. We can then define a normalized predicate vector:

$$\hat{g}(z) \triangleq g_{\max}^{-1} g(z). \tag{7.18}$$

Normalization ensures that $\hat{g}(z)$ takes values in a $[-1, 1]^{n_g}$ hypercube. This helps to standardize the parameter choices for the numerical continuation solution method, which we will describe in Section 7.4. The second stage is to pick out the maximum predicate value. Because we want a smooth approximation, we find an approximate maximum using the log-sum-exp function, also known as a softmax. For a given homotopy parameter $\kappa > 0$, the softmax function $\texttt{LSE}_\kappa : \mathbb{R}^{n_g} \to \mathbb{R}$ is defined by:

$$\texttt{LSE}_\kappa\big(\hat{g}(z)\big) \triangleq \kappa^{-1} \log\Big(\sum_{i=1}^{n_g} e^{\kappa \hat{g}_i(z)}\Big). \tag{7.19}$$

Let us denote the resulting value by $\hat{g}^* \equiv \texttt{LSE}_\kappa\big(\hat{g}(z)\big)$. As κ grows, this value approaches the true $\max_i \hat{g}_i(z)$. In the third stage, the value is passed to a sigmoid function which maps it to the $[0, 1]$ interval. This function approaches zero for negative arguments and one for

positive arguments. We define it as follows:

$$\sigma_\kappa(\hat{g}^*) \triangleq 1 - \left[1 + e^{\kappa \hat{g}^*}\right]^{-1}. \tag{7.20}$$

Note that by subtituting (7.19) into (7.20), we obtain the familiar multinomial logit function [204]:

$$\sigma_\kappa(\hat{g}^*) = 1 - \left[1 + \sum_{i=1}^{n_g} e^{\kappa \hat{g}_i}\right]^{-1}. \tag{7.21}$$

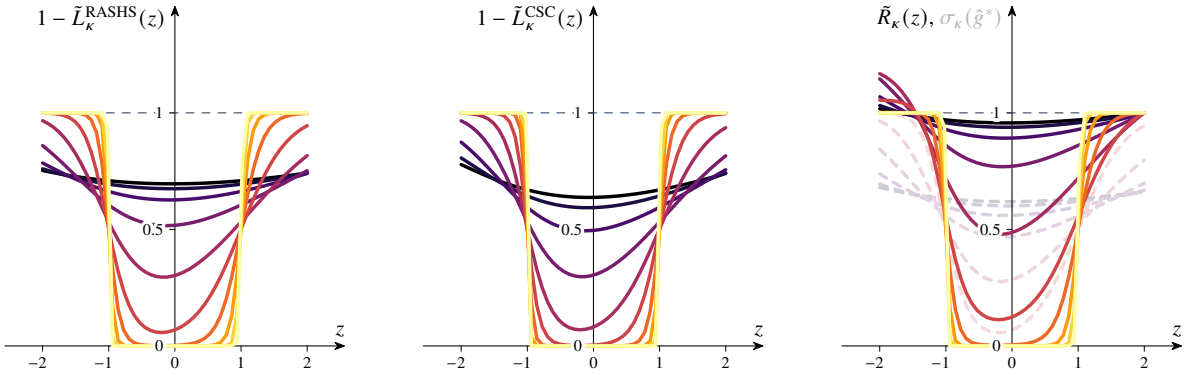
For this reason, we call our approach *multinomial logit smoothing*. When κ is large and the time comes to computing the derivatives of (7.21) for the solution process in Section 7.4, we have noted that there are important numerical stability advantages to breaking the logistic function into separate steps (7.19) into (7.20). This is why we keep the second and third stages separate.

The fourth and last stage of approximating R is to vertically shift the sigmoid function so that it matches its exact value at some specified predicate value $g_c \in \mathbb{R}^{n_g}$, where we require at least one element to be positive (such that $R(z) > 0$). We typically choose $g_c = g(z^*)$ where $z^* = \text{argmax}_z \|g(z)\|_\infty$ from (7.17). Shifting carries the benefit of not over-restricting the solution variables early in the solution process, when κ is small and $\sigma_\kappa \approx n_g/(n_g + 1)$. The latter effect is visible in the bottom row, third column of Figure 7.7. Ultimately, the smooth approximation of R is defined as follows, and is the direct counterpart of the RASHS and CSC model (7.16):

$$R(z) \approx \tilde{R}_\kappa(z) \triangleq \sigma_\kappa(\hat{g}^*) + \big(1 - \sigma_\kappa(g_c)\big). \tag{7.22}$$

The discrete logic constraint (7.14) can then be written as the following smooth approximation, which is obtained by substituting R in (7.15b) with $\tilde{R}_\kappa$ from (7.22):

$$\big[1 - \tilde{R}_\kappa(z)\big] f_L(z) + \tilde{R}_\kappa(z) f_R(z) \leq 0. \tag{7.23}$$



(a) The RASHS method [192]. (b) The CSC method [193, 194]. (c) Our proposed method (7.22).

Figure 7.8: Comparison of smoothed discrete `or` logic (7.14b) obtained using three smoothing methods. The primary difference between our method and RASHS/CSC is the shift by $1 - \sigma_\kappa(g_c)$ in (7.22). Figure (c) shows in faint dashed lines the multinomial logit function (7.21) without shifting, which is very similar to RASHS in (a).

In the following sections, we will show how to use (7.23) to model the discrete logic constraints (7.12f)-(7.12h). For the sake of comparison, the RASHS and CSC smooth approximations (7.16) are given by [192, 193]:

$$\tilde{L}_\kappa^{\text{RASHS}}(z) = \prod_{i=1}^{n_g} \left(1 + e^{\kappa \hat{g}_i(z)}\right)^{-1}, \tag{7.24a}$$

$$\tilde{L}_\kappa^{\text{CSC}}(z) = \prod_{i=1}^{n_g} \frac{1}{2}\left(1 - \tanh\left(\kappa \hat{g}_i(z)\right)\right). \tag{7.24b}$$

Figure 7.8 compares the smooth logic (7.24) with our approach (7.22). Without the shifting operation in (7.22), all three methods are remarkably similar. Multinomial logit smoothing without shifting is most similar to RASHS: the two are identical for $n_g = 1$, and slightly different for $n_g > 1$. Thus, shifting is the critical difference in our method. As we shall see below, it is most critical for constraints like the MIB (7.12f), where it is important that $\tilde{R}_\kappa(z) \approx 1$ for small κ (this effectively removes the MIB constraint from the early solution algorithm iterations in Section 7.4).

7.3.3 Modeling the Approach Cone

We begin by modeling the approach cone constraint (7.12h) in the framework of (7.14) and its smooth approximation (7.23). Comparing (7.12h) with (7.14a), we have $n_g = 1$, $z = p$, and the predicate:

$$g_1(p) = p^\mathsf{T} p - r_{\text{appch}}^2, \tag{7.25}$$

where we use the two-norm squared to conveniently make the predicate everywhere smooth. This predicate is then used in (7.22) to form $\tilde{R}_\kappa^{\text{appch}}$, the smooth `or` approximation for the approach cone predicate. The `if` implication can be written as:

$$f_L(p) = \cos(\theta_{\text{appch}}) - \hat{x}_\mathcal{L}^\mathsf{T} p \|p\|_2^{-1}. \tag{7.26}$$

When the chaser is outside of the approach sphere, we wish to allow the chaser's trajectory to assume any approach angle. By the Cauchy-Schwarz inequality, this can be expressed as the inequality $\hat{x}_\mathcal{L}^\mathsf{T} p \geq -\|p\|_2$. As a result, the `else` implicationx can be written as:

$$f_R(p) = -1 - \hat{x}_\mathcal{L}^\mathsf{T} p \|p\|_2^{-1}. \tag{7.27}$$

We can now use (7.26) and (7.27) directly in (7.23), which yields a smooth approximation of the approach cone constraint:

$$\cos(\theta_{\text{appch}}) - \big(1 + \cos(\theta_{\text{appch}})\big) \tilde{R}_\kappa^{\text{appch}}(p) - \hat{x}_\mathcal{L}^\mathsf{T} p \|p\|_2^{-1} \leq 0. \tag{7.28}$$

7.3.4 Modeling Plume Impingement

The plume impingement constraint (7.12g) is modeled in a very similar way. Recall that the rendezvous trajectory has N_c control opportunities and the chaser has $\mathcal{I}_{\text{fr}}$ forward-facing thrusters. Let us focus on the k-th control opportunity for thruster $i \in \mathcal{I}_{\text{fr}}$. Comparing (7.12g) with (7.14a), we have $n_g = 1$ and $z = \big[p(k't_c); \Delta t_{ik}\big]$. The predicate takes after

(7.25):

$$g_1\big(p(k't_c)\big) = p(k't_c)^\intercal p(k't_c) - r_{\text{plume}}^2, \tag{7.29}$$

This predicate is then used in (7.22) to form $\tilde{R}_\kappa^{\text{plume}}$, the smooth `or` approximation for the plume impingement predicate. The `if` implication for plume impingement is an equality constraint, whereas our standard formulation (7.14) requires an inequality. To reconcile the two situations, one possible approach is to leverage (7.12f) and to realize that $\Delta t_{ik} \in [0, \Delta t_{\max}]$. Thus, we can impose the constraint:

$$0 \leq \Delta t_{ik} \leq \Delta t_{\max}, \tag{7.30}$$

and we can write the following `if` implication:

$$f_L(\Delta t_{ik}) = \Delta t_{ik}. \tag{7.31}$$

Equation (7.31) together with (7.30) enforce $0 \leq \Delta t_{ik} \leq 0$ when the predicate (7.29) is `true`, which is equivalent to (7.12g). When the chaser is outside of the plume impingement sphere, the forward-facing thrusters are free to fire. We can express this as the following `else` implication:

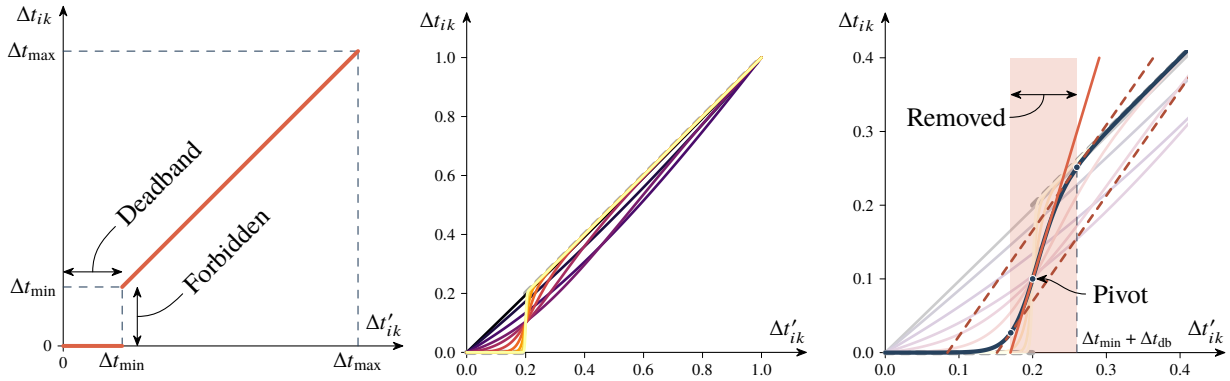
$$f_R(\Delta t_{ik}) = \Delta t_{ik} - \Delta t_{\max}. \tag{7.32}$$

Equations (7.31) and (7.32) can now be substituted into (7.23), yielding a smooth approximation of the plume impingement constraint:

$$\Delta t_{ik} \leq \tilde{R}_\kappa^{\text{plume}}\big(p(k't_c)\big)\Delta t_{\max}. \tag{7.33}$$

7.3.5 Modeling the Minimum Impulse Bit

The MIB constraint (7.12f) is the most intricate one to model effectively, and has been the core motivation behind developing a new way to handle discrete logic constraints. Our



(a) Exact deadband curve.

(b) Deadband approximation using (7.23).

(c) Zoomed-in view of (b) around $\Delta t_{\min}$.

Figure 7.9: Smooth approximation example for the MIB constraint (7.12f). The deadband discontinuity illustrated in (a) can be modeled as a discrete logic constraint using (7.14). Thus, we can approximate it using multinomial logit smoothing (7.23). The result for various values of the homotopy parameter κ is shown in (b). Figure (c) illustrates our gradient-based approach in Section 7.3.5 for preventing the optimization from exploiting the approximated jump discontinuity "wall".

past work used STCs, which exhibited locking and prevented the algorithm from converging effectively in some cases [54]. Among the several possible ways of fitting the MIB constraint into the discrete logic framework of (7.14), we present one way that yields good convergence performance across a wide variety of instances of Problem 7.12.

Let us focus the discussion on pulse Δt_{ik}, in other words the i-th thruster at the k-th control opportunity. We view the thruster as an actuator with a deadband, as illustrated in Figure 7.9a. The "input" or "reference" pulse duration is given by a *continuous* variable $\Delta t'_{ik} \in [0, \Delta t_{\max}]$. When this value falls below $\Delta t_{\min}$, the "obtained" or "output" pulse duration which the thruster actually executes is zero. Thus, while $\Delta t'_{ik}$ is a continuous variable that can take any value in $[0, \Delta t_{\max}]$, the obtained pulse duration Δt_{ik} exhibits a jump discontinuity at $\Delta t_{\min}$. Modeling this jump discontinuity is precisely the focus of our smooth approximation strategy.

Comparing Figure 7.9a with the standard model (7.14), we can write the following

`if-else` logic:

$$\Delta t'_{ik} \leq \Delta t_{\min} \;\Rightarrow\; \Delta t_{ik} = 0, \tag{7.34a}$$

$$\Delta t'_{ik} > \Delta t_{\min} \;\Rightarrow\; \Delta t_{ik} = \Delta t'_{ik}. \tag{7.34b}$$

We can thus define $n_g = 1$, $z = \left[\Delta t_{ik}; \Delta t'_{ik}\right]$, and use the predicate:

$$g_1(\Delta t'_{ik}) = \Delta t'_{ik} - \Delta t_{\min}. \tag{7.35}$$

This predicate is used in (7.22) to form $\tilde{R}^{\text{mib}}_{\kappa}$, the smooth `or` approximation for the MIB predicate. As for the implications on the right-hand side of (7.34), we can use pairs of inequalities to represent equality constraints as required by (7.14). This yields the following `if` and `else` implications:

$$f_L(\Delta t_{ik}) = \begin{bmatrix} \Delta t_{ik} \\ -\Delta t_{ik} \end{bmatrix}, \quad f_R(\Delta t_{ik}, \Delta t'_{ik}) = \begin{bmatrix} \Delta t_{ik} - \Delta t'_{ik} \\ \Delta t'_{ik} - \Delta t_{ik} \end{bmatrix}. \tag{7.36}$$

Just like for the approach cone and plume impingement constraints, (7.36) can now be substituted into (7.23) to obtain a smooth approximation of the deadband behavior in Figure 7.9a. Simplifying the result, we obtain the following constraint:

$$\Delta t_{ik} = \tilde{R}^{\text{mib}}_{\kappa}(\Delta t'_{ik})\Delta t'_{ik}. \tag{7.37}$$

The smooth approximation is shown in Figure 7.9b for a number of homotopy parameter κ values. We call this approximation the smooth deadband curve (SDC). As κ increases, the approximation converges to the exact deadband curve *with one significant exception*: the "forbidden" region (i.e., the jump discontinuity) from Figure 7.9a becomes part of the SDC as a quasi-vertical "wall" for large κ in Figure 7.9b. This raises the followig question: can a rendezvous trajectory exploit this wall and therefore "get around" the MIB constraint?

Alas, the answer is yes, and our numerical tests show that this happens quite regularly. Generally, this adversarial exploitation of the model feeds into a long-standing pain point of optimization. As Betts writes in [205, p. 701], "If there is a flaw in the problem formulation, the optimization algorithm will find it." To fix this side effect and forbid Δt_{ik} from exploiting the wall, we introduce a new constraint to the optimization problem.

The Wall Avoidance Constraint

We now develop an extra constraint to ensure that no Δt_{ik} can exploit the wall part of the SDC (7.37). We ask ourselves the following question: what makes the wall different from the other parts of the SDC? One property stands out above all others: for large κ values the wall has a very large gradient, as opposed to other parts of the curve where the gradient is approximately zero or one. There is another favorable property of (7.37): in the limit as κ increases, the smooth approximation converges to a function whose gradient monotonically increases for $\Delta t_{ik} \in [0, \Delta t_{\min})$, and monotonically decreases for $\Delta t_{ik} \in (\Delta t_{\min}, \Delta t_{\max}]$. In other words, (7.37) has an *inflection point* at $\Delta t_{\min}$ for large κ, where its gradient takes its maximum value. We call this the "pivot" since the SDC appears to revolve around this point as κ increases. This is visible in Figures 7.9b and 7.9c for the brighter colored curves that correspond to larger κ values.

We develop the following intuition from the above discussion: if we constrain Δt_{ik} such that the SDC's gradient is sufficiently less than its value at the pivot, then Δt_{ik} cannot exploit the wall. To put this into practice, define Δt_{db} to be a "buffer" around $\Delta t_{\min}$. We want the gradient at Δt_{ik} to be less than its value at the buffered pulse duration $\Delta t_{\min} + \Delta t_{\text{db}}$. The SDC gradient at $\Delta t_{\min} + \Delta t_{\text{db}}$ is computed as follows using (7.37):

$$G_{\text{db},\kappa} \triangleq \frac{\mathrm{d}\tilde{R}^{\text{mib}}_{\kappa}(\Delta t_{\min} + \Delta t_{\text{db}})}{\mathrm{d}\Delta t'_{ik}}(\Delta t_{\min} + \Delta t_{\text{db}}) + \tilde{R}^{\text{mib}}_{\kappa}(\Delta t_{\min} + \Delta t_{\text{db}}). \tag{7.38}$$

This allows us to impose the following *wall avoidance constraint*, which prevents Δt_{ik}

from taking values along the wall of the SDC:

$$\frac{\mathrm{d}\tilde{R}_{\kappa}^{\mathrm{mib}}(\Delta t'_{ik})}{\mathrm{d}\Delta t'_{ik}}(\Delta t'_{ik}) + \tilde{R}_{\kappa}(\Delta t'_{ik}) \leq G_{\mathrm{db},\kappa}. \tag{7.39}$$

Figure 7.9c illustrates an example region of $\Delta t'_{ik}$ and Δt_{ik} values that is effectively removed by (7.39). In the figure, $\Delta t_{\min} = 0.2$ s and $\Delta t_{\mathrm{db}} = 0.06$ s. The gradients of all points inside the red region are larger than $G_{\mathrm{db},\kappa}$, hence the corresponding choices of Δt_{ik} are infeasible. Because the aforementioned monotonicity property guarantees that this region contains the wall, the net effect is that the SDC wall can no longer be utilized by the optimization.

Improving Convergence

The smoothed MIB constraint (7.37) introduced a new input variable $\Delta t'_{ik}$ to represent a reference pulse duration. This variable was necessary to model the deadband curve in Figure 7.9a. If we compare the deadband curve to the original MIB constraint (7.12f), we realize that the only "useful" parts of the curve in Figure 7.9a that we actually need are the origin (i.e., $[\Delta t_{ik}; \Delta t'_{ik}] = 0$) and the continuous trace $\Delta t_{ik} = \Delta t'_{ik}$ where $\Delta t_{ik} > \Delta t_{\min}$. In both cases, we have the simple relationship $\Delta t_{ik} = \Delta t'_{ik}$. Our numerical experience shows that encouraging this equality significantly improves the convergence process of the algorithm in Section 7.4. We do this by adding the following regularization term to the original cost (7.12a):

$$J_{\mathrm{eq}} = w_{\mathrm{eq}} \Delta t_{\min}^{-1} \sum_{i=1}^{n_{\mathrm{rcs}}} \sum_{k=1}^{N_c} \|\Delta t_{ik} - \Delta t'_{ik}\|_1, \tag{7.40}$$

where $w_{\mathrm{eq}} > 0$ is some small weight for the cost. We view (7.40) as penalizing the choice $\Delta t_{ik} \neq \Delta t'_{ik}$. The use of the one-norm encourages sparsity in the number of Δt_{ik} that violate the equality. This choice traces back to theory from lasso regression, sparse signal recovery, and basis pursuit to find sparse solutions [34].

7.3.6 Smoothed Rendezvous Problem

We are now in a position to restate Problem 7.12 as a continuous optimization problem by using the smoothed discrete logic constraints from the previous sections. The process is straightforward: simply replace each discrete logic constraint with its smooth approximation. We call the result the smooth rendezvous problem (SRP), stated below.

$$\min_{x,\Delta t,\Delta t',t_f} J_{\text{fuel}} + J_{\text{eq}} \tag{7.41a}$$

$$\text{s.t. Dynamics (7.12b)-(7.12e)}, \tag{7.41b}$$

$$0 \leq \Delta t_{ik} \leq \Delta t_{\max},\ 0 \leq \Delta t'_{ik} \leq \Delta t_{\max}, \tag{7.41c}$$

$$\Delta t_{ik} = \tilde{R}_\kappa^{\text{mib}}(\Delta t'_{ik})\Delta t'_{ik}, \tag{7.41d}$$

$$\frac{\mathrm{d}\tilde{R}_\kappa^{\text{mib}}(\Delta t'_{ik})}{\mathrm{d}\Delta t'_{ik}}(\Delta t'_{ik}) + \tilde{R}_\kappa(\Delta t'_{ik}) \leq G_{\text{db},\kappa}, \tag{7.41e}$$

$$\Delta t_{ik} \leq \tilde{R}_\kappa^{\text{plume}}\big(p(k' t_c)\big)\Delta t_{\max} \text{ for all } i \in \mathcal{I}_{\text{fr}}, \tag{7.41f}$$

$$\cos(\theta_{\text{appch}}) - \big(1 + \cos(\theta_{\text{appch}})\big)\tilde{R}_\kappa^{\text{appch}}(p) - \hat{x}_{\mathcal{L}}^{\mathsf{T}} p \|p\|_2^{-1} \leq 0, \tag{7.41g}$$

$$\text{Boundary conditions (7.12i)-(7.12k)}. \tag{7.41h}$$

The key difference between Problem 7.12 and the new Problem 7.41 is that the latter no longer contains integer variables to solve. Instead, there is a single homotopy parameter κ that regulates how accurately the smoothed constraints (7.41d), (7.41f), and (7.41g) approximate the original discrete logic. Thus, we have eliminated the third difficulty mentioned in Section 7.2.6 (i.e., the mixed-integer programming aspect). However, we are now faced with solving a nonconvex optimization problem, and there remains the question of how to set the value of κ. In the next section we answer both questions using sequential convex programming and numerical continuation.

7.4 Sequential Convex Programming with Numerical Continuation

We now present a numerical optimization algorithm that solves Problem 7.41. This algorithm combines two key technologies: sequential convex programming (SCP) and numerical continuation. SCP is an iterative scheme traditionally designed to solve Problem 7.41 for a given value of κ. The *raison d'être* for numerical continuation is to greatly expand the region of convergence of iterative schemes [206]. Due to the vanishing gradient problem and the very large gradients at the "step" transition points of discrete logic (see, for example, Figures 7.7, 7.8, and 7.9), SCP is unlikely to converge if a large κ value is used right away together with an initial guess that is not already almost optimal [49]. As a result, numerical continuation is used to aid SCP convergence. This is done by providing an algorithm to update κ starting from a small value where the smooth approximation is coarse, and increasing it until a large value where the approximation is quasi-exact.

Our core contribution is to *merge* these two methods. In other words, the algorithm that we present is not SCP with a numerical continuation "outer loop". Rather, the methods run simultaneously. The numerical results in Section 7.5 show that this can dramatically decrease the total number of required iterations without sacrificing optimality.

7.4.1 The Penalized Trust Region Algorithm

We begin by describing the penalized trust region (PTR) algorithm. This is a particular SCP method that has been widely used for fast and even real-time solution of nonconvex problems like Problem 7.41, where the value of κ is fixed [70, 201, 202]. This section provides a brief overview of PTR and identifies locations where the method is changed in order to embed numerical continuation. These changes are then described in the sections that follow. For the standard portions of the PTR algorithm, we will refer the reader to existing literature which already provides detailed explanations.

The goal of SCP in general, and PTR in particular, is to solve continuous-time optimal

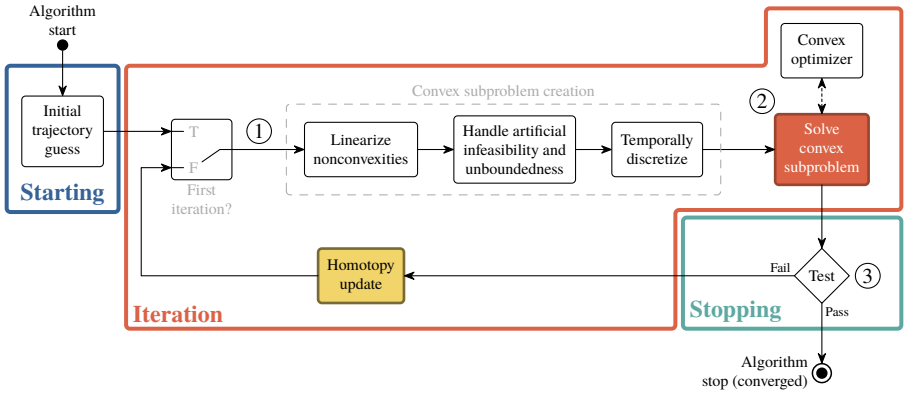


Figure 7.10: Block diagram illustration of the PTR sequential convex programming algorithm. The method is composed of three major parts: a way to guess the initial trajectory (Starting), an iteration scheme which refines the trajectory until it is feasible and locally optimal (Iteration), and an exit criterion to stop once the trajectory has been computed (Stopping). Our novel contribution is the highlighted "homotopy update" block, which implements numerical continuation to solve Problem 7.41.

control problems of the following form:

$$\min_{x,u,p,t_f} \; J(x,u,p) \tag{7.42a}$$

$$\text{s.t. } \dot{x}(t) = f\big(t, x(t), u(t), p\big), \tag{7.42b}$$

$$\big(x(t), p\big) \in \mathcal{X}(t), \;\; \big(u(t), p\big) \in \mathcal{U}(t), \tag{7.42c}$$

$$s\big(t, x(t), u(t), p\big) \leq 0, \tag{7.42d}$$

$$g_{\text{ic}}\big(x(0), p\big) = 0, \;\; g_{\text{tc}}\big(x(t_f), p\big) = 0, \tag{7.42e}$$

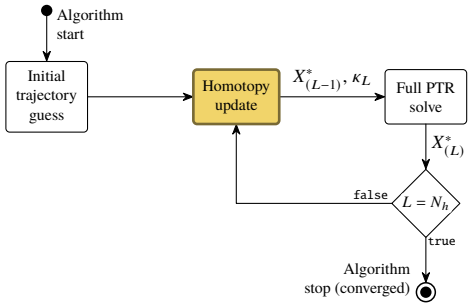
where $x(\cdot) \in \mathbb{R}^{n_x}$ is the state trajectory, $u(\cdot) \in \mathbb{R}^{n_u}$ is the control trajectory, and $p \in \mathbb{R}^{n_p}$ is a vector of parameters. The function $f : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ encodes the nonlinear equations of motion, which are assumed to be at least once continuously differentiable. Initial and terminal boundary conditions are enforced by using the continuously differentiable functions $g_{\text{ic}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_{\text{ic}}}$ and $g_{\text{tc}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_{\text{tc}}}$. The convex and nonconvex path (i.e., state and control) constraints are imposed using the convex sets $\mathcal{X}(t)$, $\mathcal{U}(t)$, and the

continuously differentiable function $s : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_s}$. Finally, a continuously differentiable cost function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}$ encodes some trajectory property that is to be minimized. Without giving the explicit details here, we note that Problem 7.41 can be fit into the mold of Problem 7.42 for any fixed value of κ. The interested reader may consult our open-source implementation for details (see Figure 7.15), and may refer to [49] for a broad tutorial on the modeling process.

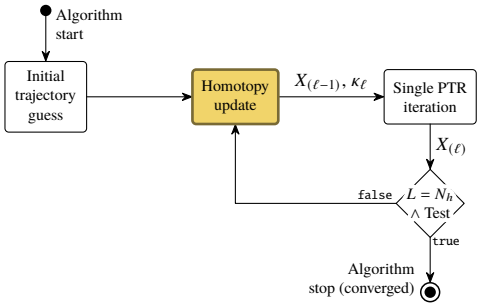
At the core of PTR is the idea of solving Problem 7.42 through iterative convex approximation. The algorithm can be represented in the block diagram form of Figure 7.10. Strictly speaking, PTR is a nonlinear local optimization algorithm known as a trust region method [32, 170, 207].

Let us begin by assuming that the homotopy parameter κ is fixed to a specific value. PTR solves Problem 7.41 using a sequence of convex approximations called *subproblems*. Roughly speaking, the convex approximation is improved each time that a new solution is obtained. Going around the loop of Figure 7.10, all algorithms start with a user-supplied initial guess, which can be very coarse (more on this later). At ①, the SCP algorithm has available a so-called reference trajectory, which may be infeasible with respect to the problem dynamics and constraints. The nonconvexities of the problem are removed by a local linearization around the reference trajectory, while convex elements are kept unchanged. To ensure that linearization does not cause the subproblems to become infeasible, extra terms are added which are known as virtual controls (for the dynamics (7.42b)) and virtual buffers (for the constraints (7.42d) and (7.42e)). The resulting convex continuous-time subproblem is temporally discretized to yield a finite-dimensional convex optimization problem. The optimal solution to the discretized subproblem is computed at ②, where the SCP algorithm makes a call to any appropriate convex optimization solver. The solution is tested at ③ against stopping criteria. If the test passes, the algorithm has converged and the most recent solution from ② is returned. Otherwise, the solution becomes the new reference trajectory for the next iteration of the algorithm.

The traditional PTR method as described above is covered in great depth in existing



(a) The non-embedded approach. Each homotopy update is followed by a full PTR solve.



(b) Our proposed embedded approach. Homotopy updates are interspersed with individual PTR iterations.

Figure 7.11: Block diagram illustration of the proposed numerical continuation scheme. Figure (a) demonstrates the non-embedded "standard" approach. Our proposed method, which embeds the homotopy update into PTR itself by placing it "in between" individual PTR iterations, is shown in (b). The "Test" in (b) corresponds to the stopping criterion from Figure 7.10.

literature. We refer the reader to a recent expansive tutorial [49], and to papers which describe PTR in the context of rocket landing, rendezvous and docking, and quadrotor flight [54, 67, 70, 201, 202]. In this chapter we will focus our attention on the novel "homotopy update" block in Figure 7.10. This block implements a numerical continuation method in order to update κ until the smooth approximations of discrete logic from Section 7.3 become quasi-exact.

7.4.2 Non-embedded Numerical Continuation

In order to arrive at the embedded numerical continuation approach, we begin by motivating a non-embedded scheme which we will then generalize to the embedded algorithm. As shown in Figure 7.11a, the basic idea is to update the homotopy parameter κ after each time that Problem 7.41 is solved for the current value of κ. Furthermore, each new call to PTR is "warm started" by providing the most recent solution as the initial guess in Figure 7.10.

In formal terms, let L denote the iteration number of the non-embedded algorithm.

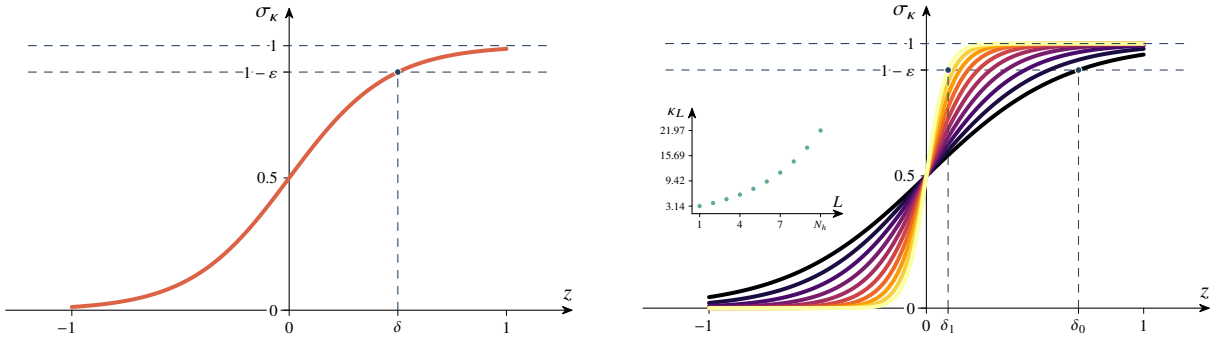
Effectively, L corresponds to the number of full PTR solves of Problem 7.41 that have occured up until the end of that iteration. If we place ourselves at iteration L, then let κ_L denote the homotopy parameter chosen by the "homotopy update" block, and let $X^*_{(L)}$ be the corresponding solution of Problem 7.41 computed by PTR. Importantly, PTR is warm started with the initial guess $X^*_{(L-1)}$. When $L > 1$, this corresponds to the PTR solution from the previous iteration (i.e., the solution of Problem 7.41 for the previous value of κ). For the first iteration $L = 1$, $X^*_{(0)}$ corresponds to the user-chosen initial trajectory guess. The job of the homotopy update is the following: compute κ_L given $X^*_{(L-1)}$ and κ_{L-1}. While we describe the details below, the basic idea is as follows: κ_L grows with L and, eventually, the smooth approximations from Section 7.3 become quasi-exact representations of the original discrete logic (e.g., see the example in Figure 7.7). Once κ_L reaches some large user-defined value that yields an accurate enough approximation of the discrete logic, the algorithm terminates.

The remaining task for the non-embedded numerical continuation approach is to define the internals of the homotopy update block in Figure 7.11a. Our method stems from viewing the sigmoid function (7.20) as a smooth model for a step function. As we increase the homotopy parameter κ, we want to explicitly control how "sharply" the sigmoid approximates the step function's discontinuity. This leads us to the following update rule, which is illustrated in Figure 7.12. As shown in Figure 7.12a, we define two parameters: a precision $\varepsilon \in (0, 1)$ and a smoothness $\delta > 0$. The sigmoid function σ_κ in (7.20) is then required to satisfiy the following *interpolation condition*: it must equal $1 - \varepsilon$ when its argument equals δ. An exact step function corresponds to $\varepsilon = 0$ and $\delta = 0$, so we view ε and δ as defining how much the sigmoid deviates from the exact step function.

For the homotopy update rule, we hold ε constant and define two bounds on δ: a "smoothest" value δ_0 and a "sharpest" value $\delta_1 < \delta_0$. We then sweep δ as follows:

$$\delta_\alpha = \rho^\alpha \delta_0, \qquad \rho = \delta_1/\delta_0, \tag{7.43}$$

where $\alpha \in [0, 1]$ is an interpolation parameter. The homotopy value that satisfies the inter-



(a) Each sigmoid is formed by fixing the values of ε and δ, which determine the homotopy parameter κ.

(b) The smoothness parameter δ is decreased from δ_0 to δ_1. The sigmoid is sharpened as a result.

Figure 7.12: Our approach to updating the homotopy parameter κ is to define a precision ε and a smoothness δ where the sigmoid attains the value $1-\varepsilon$. The smoothness parameter is then decreased from an initial value δ_0 to a final value of δ_1. In the process, the sigmoid becomes a close approximation of the step function.

polation condition is given by:

$$\kappa_\alpha = \frac{\ln\left(\varepsilon^{-1}-1\right)}{\rho^\alpha \delta_0}. \tag{7.44}$$

Equation (7.44) defines a continuous range of homotopy values from the smoothest ($\alpha = 0$) to the sharpest ($\alpha = 1$) case. In practice, we set a fixed number of updates N_h and let $\alpha = (L-1)/(N_h-1)$ for $L = 1, 2, \ldots, N_h$. Thus, N_h defines the number of iterations in the non-embedded numerical continuation algorithm of Figure 7.11a. Using (7.44), the homotopy value κ_L at iteration L is given by:

$$\kappa_L = \frac{\ln\left(\varepsilon^{-1}-1\right)}{\rho^{(L-1)/(N_h-1)} \delta_0}. \tag{7.45}$$

7.4.3 Embedded Numerical Continuation

We are now ready to describe the embedded numerical continuation algorithm shown in Figure 7.11b. One key difference distinguishes this algorithm from the non-embedded approach:

PTR does not have to run to completion before the homotopy parameter κ is increased. As shown in Figure 7.11b, the full PTR solve of the non-embedded method is replaced with a *single* PTR iteration (which corresponds to the top half of the PTR block diagram in Figure 7.10). We use ℓ to denote the PTR iteration counter. At each iteration ℓ, a homotopy update rule is called that potentially changes the value of κ. This new value and the most recent PTR iterate (i.e., subproblem solution) are used for the next PTR iteration. The process shown in Figure 7.11b works exactly like in Figure 7.10, with the blocks rearranged.

Now that we understand how the algorithm is structured, we need to describe the homotopy update. This is composed of two parts: deciding *whether* to update κ, and then updating it. The latter piece works just like in the previous section. Once we know that κ should be updated, we use (7.46) to compute its new value:

$$\kappa_\ell = \frac{\ln\left(\varepsilon^{-1} - 1\right)}{\rho^{L/(N_h-1)}\delta_0}, \quad L \leftarrow L + 1, \tag{7.46}$$

where L now represents the number of times that the homotopy parameter has been updated so far. The core of the embedded homotopy update rule is the first piece: deciding whether to update κ. For this, let J_ℓ denote the subproblem cost achieved at PTR iteration ℓ. If the following condition holds, then we update κ:

$$\beta_{\text{worse}} \leq \frac{J_{\ell-1} - J_\ell}{|J_{\ell-1}|} \leq \beta_{\text{trig}} \quad \wedge \quad L < N_h. \tag{7.47}$$

The second half of the condition is simple: don't update κ if this is already its highest value. The first half is a condition on relative cost decrease over the past iteration. If the cost in the current iteration decreased by less than β_{trig} relative to the last iteration, then the algorithm is "converging" for the current value of κ and it is time to update it. However, the cost is not guaranteed to decrease monotonically with PTR iterations. Thus, the relative cost decrease may be negative, which means that the cost increased over the past iteration. In this case, we may specify a certain (small) tolerance $\beta_{\text{worse}} < 0$. This means that we will

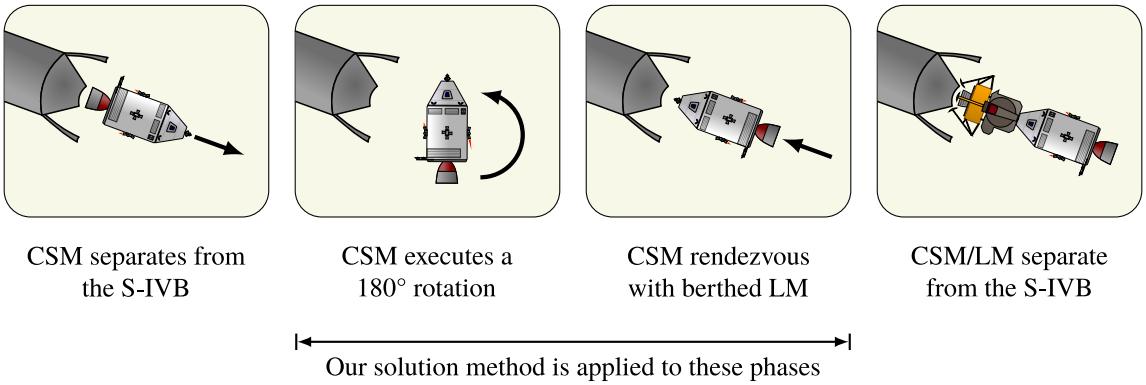


Figure 7.13: Illustration of the Apollo CSM Transposition and Docking maneuver with the LM housed inside the Saturn S-IVB third stage [195, Figure 2-11]. We evaluate our solution method by computing an open-loop reference trajectory for the middle two phases.

still update κ if the cost did not increase by more than β_{worse} allows. In the numerical results of Section 7.5 we set $\beta_{\text{worse}} = -10^{-3}$ (i.e., a 0.1% tolerance).

In summary, the embedded numerical continuation scheme is given by Figure 7.11b. In the "homotopy update" block, a decision is made using (7.47) whether to update the current κ value. If the answer is `true`, then κ is updated using (7.46). Otherwise, it is maintained at its present value.

7.5 Numerical Results

In this section we apply our solution method to a more challenging variant of the historical docking maneuver between the Apollo Command and Service Module (CSM) and the Lunar Module (LM). Section 7.5.1 defines the problem parameters and Section 7.5.2 discusses the solved trajectory and various computational aspects. The key takeaways are as follows. Our algorithm is able to consistently find rendezvous trajectories which satisfy the discrete logic constraints from Section 7.2. The algorithm is insensitive to the β_{trig} tolerance parameter in (7.47), and in fact increasing this value can dramatically reduce the total number of iterations. The total convex solver time is approximately 13.5 s, which is fast for an implementation that is not optimized for speed. We discuss how the algorithm can be made to run in under

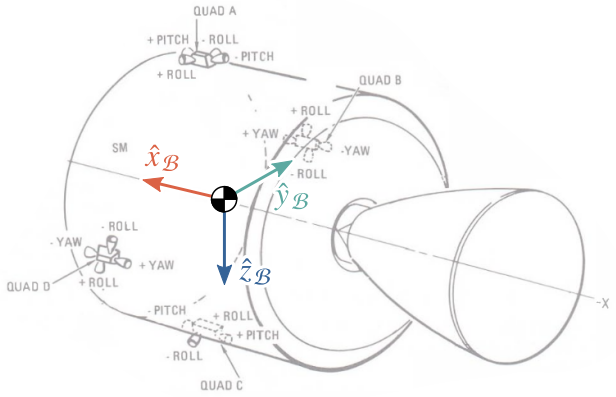


Figure 7.14: Layout of the Apollo SM RCS thrusters [200, Figure 2.5-1]. There are $n_{\text{rcs}} = 16$ thrusters grouped into four "quads" labeled A/B/C/D and each having four independent hypergolic pressure-fed thrusters.

10 s of total solution time.

7.5.1 Problem Parameters

The numerical example is inspired by the Apollo CSM "Transposition and Docking" (TD) maneuver [200, Section 2.13.1.1]. As illustrated in Figure 7.13, this maneuver uses the RCS thrusters of the CSM in order to dock with the LM, which is housed inside the Saturn S-IVB third stage. The most interesting feature of this maneuver is the 180° rotation of the CSM, which allows us to stress-test our solution method for the CSM's coupled translation and rotation dynamics from Section 7.2.1.

The SM RCS system is composed of four similar, independent "quads" located 90° apart around the Service Module (SM) circumference, as illustrated in Figure 7.14. Each quad is composed of four independent hypergolic pressure-fed pulse-modulated thrusters, yielding a total of $n_{\text{rcs}} = 16$ control inputs [200, Section 2.5.1]. We model the complete, high-fidelity CSM geometry according to public NASA documentation [195, 200]. The CSM mass and

inertia are specified in [195, Table 3.1-2]:

$$m \approx 30323 \text{ kg}, \quad J \approx \begin{bmatrix} 49249 & 2862 & -370 \\ 2862 & 108514 & -3075 \\ -370 & -3075 & 110772 \end{bmatrix} \text{ kg m}^2. \tag{7.48}$$

The RCS thrusters are capable of producing approximately $\|\hat{f}_i\|_2 = 445$ N of thrust in steady-state operation [200, Figure 2.5-8, Table 4.3-1]. However, during transposition and docking they are pulse-fired in bursts [199, 200, Section 2.5.1]. In this mode of operation, the minimum electric on-off pulse width is 12 ms [200, Section 2.5.2.3.1], generating an irregular burst of thrust that lasts for upwards of 65 ms and with a peak of 300 to 350 N [200, Figure 2.5-9]. To buffer the thrust away from this irregular region, we set $\Delta t_{\min} = 112$ ms (which corresponds to a 50 N s impulse) and $\Delta t_{\max} = 1$ s. On a system architecture level, we assume that irregularity in the thrust profile is going be corrected by a feedback control system that tracks our open-loop rendezvous trajectory.

Table 7.1 summarizes the major numerical values used to obtain the results of the following section. Other parameters not mentioned (such as the CSM geometry) can be consulted directly in our open-source implementation linked in Figure 7.15. Note that the maneuver we are seeking to optimize is more complicated than the original Apollo TD concept of operations. The Apollo initial position p_0 was almost purely along the $\hat{x}_{\mathcal{L}}$ axis, whereas we add significant $\hat{y}_{\mathcal{L}}$ and $\hat{z}_{\mathcal{L}}$ displacement in order to stress our algorithm. Furthermore, the original TD maneuver takes place after translunar injection whereas we use a LEO circular orbit. This allows us to use the Clohessy-Wiltshire-Hill dynamics (7.2), which adds further complexity compared to our previous work [54].

Our algorithm from Section 7.4 is implemented using the framework introduced in [49]. The Julia programming language is used because it is simple to read like Python, yet it can be as fast as C/C++ [208]. The timing results in the next section correspond to a Dell XPS 13 9260 laptop powered by an Intel Core i5-7200U CPU clocked at 2.5 GHz. The computer

Parameter	Value
Rendezvous parameters	
$\Delta t_{\min}$	112 ms
$\Delta t_{\max}$	1 s
F_{rcs}	445 N
r_{plume}	20 m
r_{appch}	30 m
θ_{appch}	10°
t_f	$\in [100, 1000]$ s
n_o	400 km LEO
p_0	$100\hat{x}_{\mathcal{L}} - 20\hat{z}_{\mathcal{L}} + 20\hat{y}_{\mathcal{L}}$ m
v_0	0 m s^{-1}
q_0	$[0; 0; 0; 1]$
$\omega_0,\ \omega_f$	0 rad s^{-1}
p_f	$4.48\hat{x}_{\mathcal{L}} - 0.05\hat{y}_{\mathcal{L}} + 0.17\hat{z}_{\mathcal{L}}$ m
v_f	$-0.1\hat{x}_{\mathcal{L}}$ m s^{-1}
q_f	$[0; 0.26; 0.97; 0]$
ε_{r_f}	0.1 m
ε_{v_f}	1 cm s^{-1}
ε_{q_f}	1°
ε_{ω_f}	0.01° s^{-1}
Algorithm parameters	
ε	10^{-2}
δ_0	10
δ_1	0.01
N_h	10
w_{eq}	1
β_{worse}	-10^{-3}
β_{trig}	0.1
Δt_{db}	11.2 ms
PTR convex subproblem size	
Variables	2956
Equalities	339
Affine inequalities	2227
One-norm cones	52
Inf-norm cones	104

Table 7.1: Numerical parameters for the Apollo CSM/LM Transposition and Docking maneuver trajectory optimization.

has 8 GiB LPDD3 RAM and 128 KiB L1, 512 KiB L2, and 3 MiB L3 cache. The ECOS software (written in C) is used as the low-level numerical convex solver at location ② in Figure 7.10 [43]. Note that the convex subproblem size in Table 7.1 is considerably large. Modern interior point methods routinely handle problems of this size, as the next section confirms for the rendezvous problem.

7.5.2 Computed Trajectory

Figures 7.16-7.21 exhibit our algorithm's solution as well as its associated properties for Problem 7.41 with the parameters in Table 7.1. The initial guess provided to the algorithm in Figure 7.10 is a straight-line interpolation in position and a spherical linear interpolation for the atittude quaternion [197]. The initial RCS thruster pulse durations are all set to zero.

github.com/dmalyuta/scp_traj_opt/tree/jgcd

Figure 7.15: The complete implementation source code for the Apollo Transposition and Docking numerical example can be found in the `jgcd` branch of our open-source GitHub repository.

We begin by discussing the position trajectory, whose 2D projections onto the LVLH axes are shown in Figure 7.16. The trajectory has two salient features. First, the RCS thrusters fire mostly at the start to initiate motion, and near the end to cancel vehicle rates just prior to docking. This resembles the classical two-impulse rendezvous maneuver [196], modified to account for 6-DOF dynamics, the RCS system geometry, and the discrete logic constraints (7.41c)-(7.41g), all of which are absent in the classical setup. Secondly, recall that negative $\hat{z}_{\mathcal{L}}$ positions correspond to lower orbits where objects move faster relative to the target. The chaser exploits this "gift" from orbital mechanics by dipping into the negative $\hat{z}_{\mathcal{L}}$ positions (see the top and bottom plots) where it benefits from a zero-fuel acceleration to the target. Furthermore, note how the chaser stays within the approach cone when it is inside the approach sphere, as required by (7.41g). The time histories of the chaser's states for this trajectory are shown in Figure 7.17. The quaternion attitude was converted to the more informative Euler angles using the Tait-Bryan yaw-pitch-roll convention. The continuous-time trajectory shown in the figure is propagated through the original nonlinear dynamics (see Section 7.2.1) using a numerical integrator. Because it passes through the discrete-time solution at each temporal node, we conclude that the trajectory is dynamically feasible for the chaser.

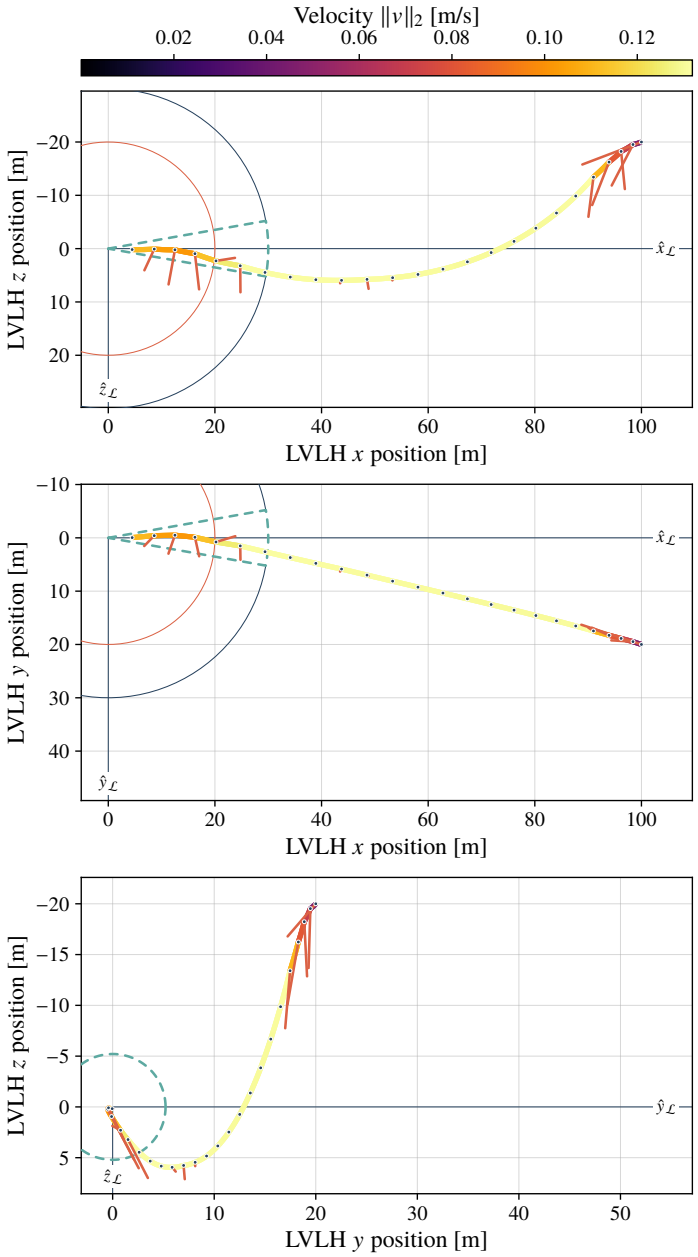


Figure 7.16: Projected views of the optimized trajectory in the LVLH frame centered at the docking port of the target spacecraft. Projections are also shown of the approach and plume impingement spheres, as well as of the approach cone. The red vectors represent the net thrust generated by the combined action of the RCS thrusters, scaled for size (so only *relative* magnitudes are correct). The circular markers show the chaser's COM for the discrete-time solution, while the continuous trajectory is obtained by integrating the optimal control through the original nonlinear dynamics of Section 7.2.1. Because the two coincide, we conclude that the converged trajectory is dynamically feasible.

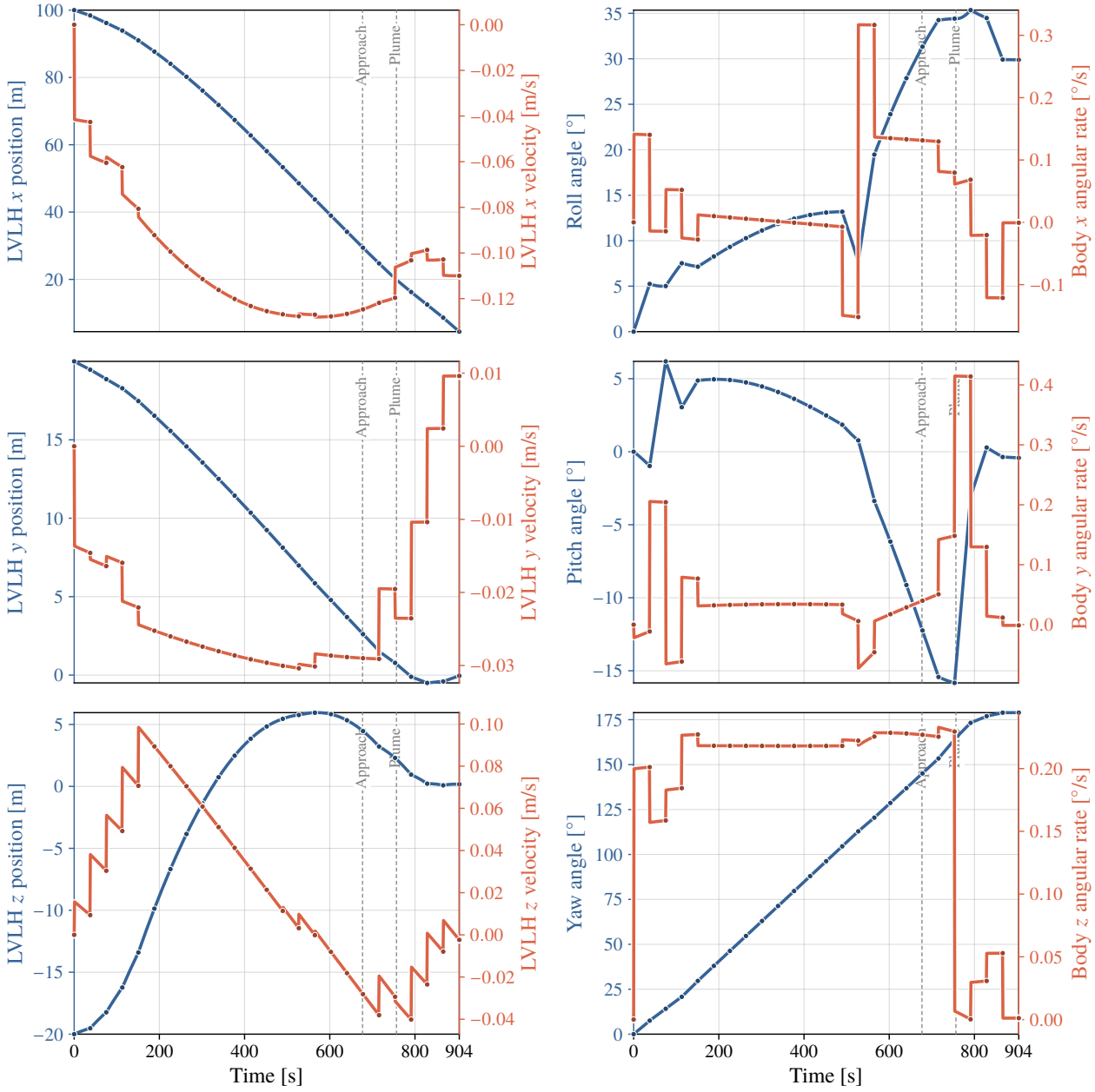


Figure 7.17: Evolution of the chaser's 6-DOF state vector for the rendezvous trajectory in Figure 7.16. Green vertical lines demarcate the times at which the chaser enters the approach and plume impingement spheres. Velocity and angular rate states exhibit jumps according to our impulsive thruster model in Section 7.2.2. Note that the chaser assumes a 30° roll orientation at docking, as required by the CSM/LM geometry [195, Figure 2-4].

The RCS thruster pulse history is shown in Figure 7.18. The pulses are relatively sparse and clustered around the start and end of the trajectory. As required by the plume impingement constraint (7.41f), the forward thrusters are silent once the chaser is inside the plume impingement sphere. Furthermore, some thrusters fire for almost exactly $\approx \Delta t_{\min}$ s. This shows that the smoothed discrete logic (7.41d) actively enforces the MIB constraint (7.5). The constraint (7.41d) is therefore indispensable for satisfying the MIB requirement.

We can estimate the total fuel consumption of the rendezvous trajectory using NASA charts for RCS thruster performance [198, Figure 4.3-6 and 4.3-7]. These charts map pulse duration to the corresponding amount of fuel consumed by a single thruster. By applying these charts to the pulse history in Figure 7.18, we obtain a fuel consumption of 2.63 kg. Unfortunately, NASA documentation on the actual fuel consumption achieved by the Apollo missions is unclear; [195, Table 3.1-7] suggests that it was 32 kg, but this confounds the other phases of the TD maneuver which we do not consider (see Figure 7.13). In any case, it appears that our trajectory uses considerably less fuel, not to mention that its initial conditions are more challenging than those of the Apollo concept of operations due to the initial position offsets along $\hat{y}_{\mathcal{L}}$ and $\hat{z}_{\mathcal{L}}$.

The convergence process of our algorithm and the runtime performance of its implementation are shown in Figure 7.19. As the iterations progress and the homotopy parameter approaches its largest value, the algorithm appears to attain a superlinear convergence rate (noticeable over iterations $\ell \in [19, 30]$). A small spike in solver time appears around the iterations where the homotopy parameter changes rapidly (see Figure 7.20 ahead). Otherwise, the subproblem difficulty stays roughly constant over the iterations. While our Julia implementation takes a median time of 50 s, the cumulative median time for solving the subproblems at location ② in Figure 7.10 is 13.5 s (which is the sum of the red "solve" bars in Figure 7.19). This corresponds to the time taken by the ECOS convex solver, which is written in C. We view this as a favorable runtime result for the following reasons. ECOS is a generic solver, and a custom solver is likely to run at least twice as fast [42, 107]. Coupled with other implementation efficiencies, we expect that the total solver time can be reduced

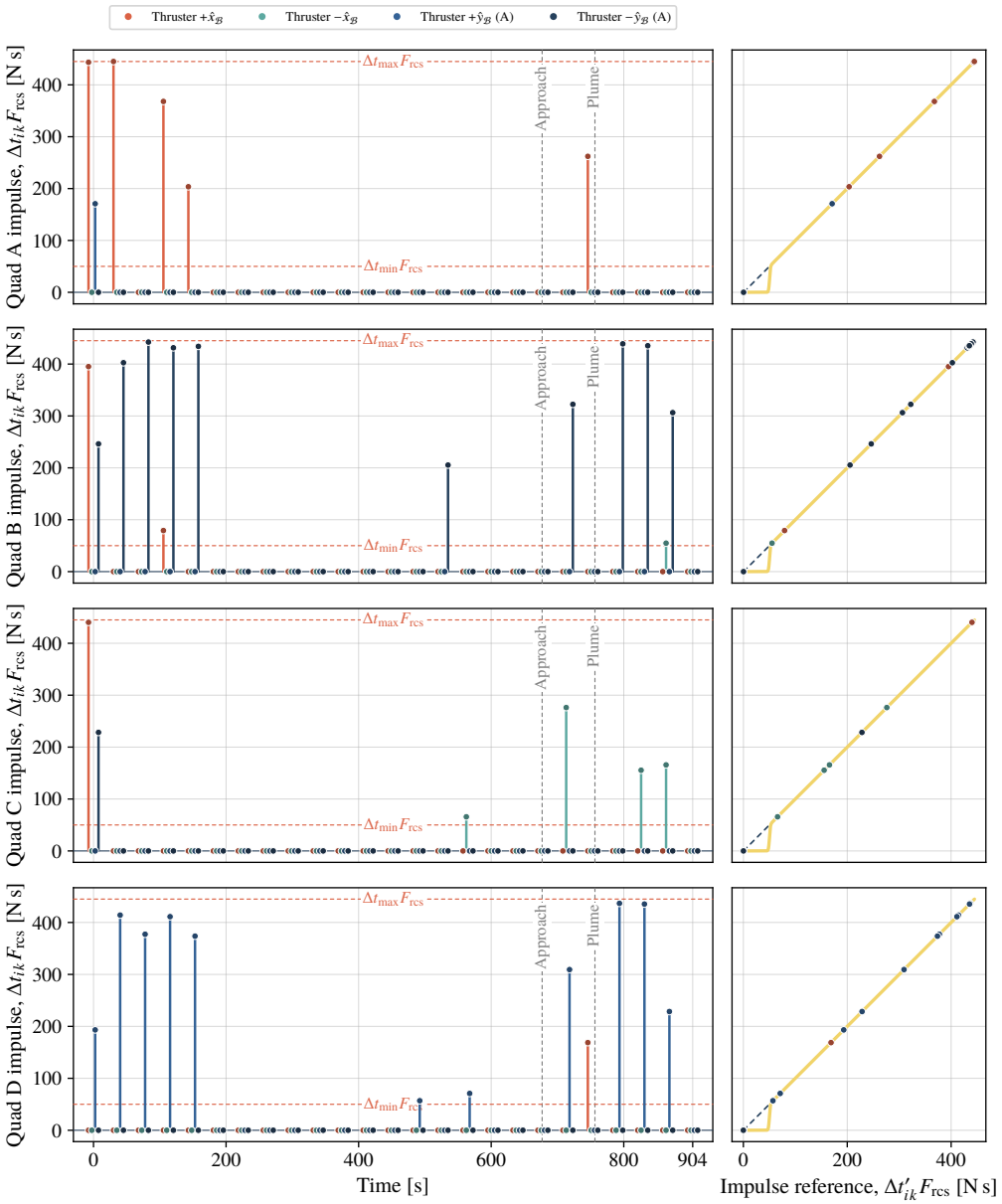


Figure 7.18: RCS thruster firing history, shown in terms of the impulse $F_{\text{rcs}}\Delta t_{ik}$. Green vertical lines demarcate the times at which the chaser enters the approach and plume impingement spheres. In the left column, the four thrusters of each quad are "spread" along the time axis in order to not overlap. See Figure 7.14 for help associating quads and thrusters to their physical locations on the CSM. Note that the legend for the roll thrusters corresponds to quad A (the thrusters on other quads are simply rotated by 90° in succession).

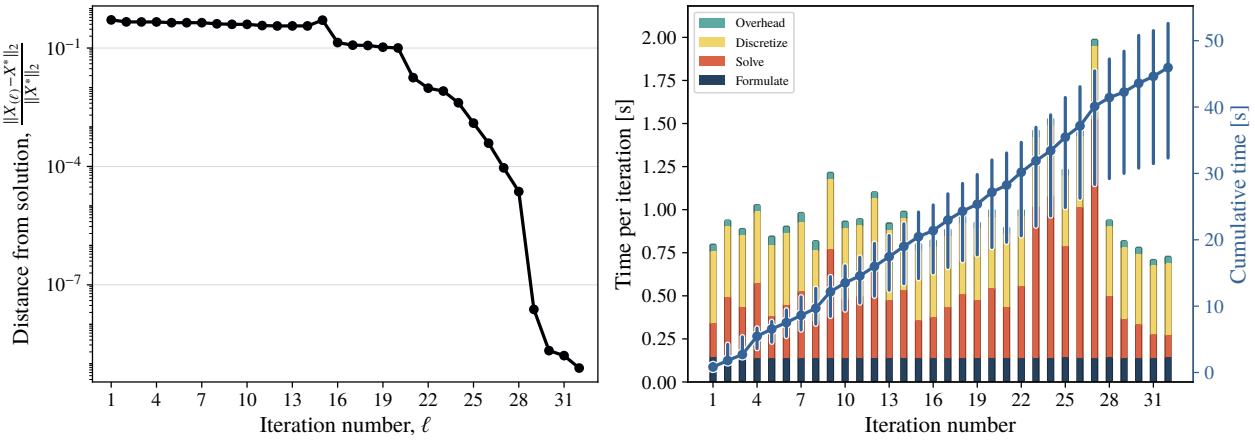


Figure 7.19: Convergence and runtime performance of our algorithm. The runtimes in the right plot show statistics over 20 executions. Formulate measures the time taken to parse the subproblem into the input format of the convex optimizer; Solve measures the time taken by the core convex numerical optimizer; Discretize measures the time taken to temporally discretize the linearized dynamics from Section 7.2.1; and Overhead measures the time taken by all other supporting tasks during a single PTR iteration. Each bar shows the median time. The blue trace across the diagonal shows the cumulative time obtained by summing the runtimes of all preceding iterations. Its markers are placed at the median time, and the error bars show the 10% (bottom) and 90% (top) quantiles. Because small runtime differences accumulate over iterations, the error bars grow with iteration count.

to < 5 s. Furthermore, our code is optimized for readability. By writing other parts of the algorithm in a compiled language and optimizing for speed, we can expect to shrink the other 36 s of runtime down to < 5 s as well. Thus, a speed-optimized implementation of our algorithm may solve the rendezvous problem in under 10 s, which is quite acceptable for rendezvous applications since the actual trajectory can last for several thousand seconds.

Figure 7.20 shows the evolution of the cost function value over the PTR iterations. Every time the cost improvement falls within the decision range of (7.47), the homotopy parameter is updated. The update is followed by a spike in the cost, with fast subsequent improvement to an equal or better (i.e., smaller) value. During the final stages of the optimization (iterations $\ell \geq 18$), increases in κ no longer cause appreciable spikes in cost. This is remarkable, given that it is at this time that the homotopy parameter experiences its largest growth

(since it grows exponentially, as seen in Figure 7.12b and the log scale of the bottom plot in Figure 7.20). This means that, well before convergence occurs, our algorithm already finds a solution that is feasible with respect to the final "sharp" approximation of the discrete logic. This analysis is corroborated by the left plot in Figure 7.19, where it can be seen that past iteration $\ell \approx 20$ the amount by which the solution changes drops off quickly.

Finally, Figure 7.21 analyzes the depence of the optimal solution and of our algorithm's performance on the choice of homotopy update tolerance β_{trig} in (7.47). This reveals several favorable properties of the algorithm. First, by increasing β_{trig} we can dramatically lower the total iteration count and speed up solution time. A very low value of β_{trig} emulates the non-embedded numerical continuation scheme from Figure 7.11a, since κ does not update until PTR has quasi-converged for its current value. By increasing β_{trig}, we can lower the iteration count by over 60% for this rendezvous example. We observe this behavior consistenly across different initial conditions. At the same time as lowering the iteration count, we basically maintain a consistent level of fuel optimality. The fuel consumption goes up and down slightly, but on balance there is no perceptible trend. A notable downside of using a larger β_{trig} is an increased danger of not converging to a feasible trajectory, since we have "rushed" the algorithm too much. This does not happen in the present example, but we have noticed the issue for other initial conditions. Our future work plans to investigate what is the theoretically safe upper bound for the β_{trig} value.

7.6 Conclusion

This chapter presents a novel algorithm that merges sequential convex programming and numerical continuation. By applying multinomial logit smoothing, the algorithm is able to model a general class of discrete logic constraints in a continuous optimization framework. This makes the approach amenable to fast and reliable solution methods for trajectory optimization problems commonly encountered in spaceflight. In particular, the algorithm is motivated and tested for the terminal phase of a rendezvous and docking maneuver, where a chaser spacecraft must dock with a target while satisfying the following discrete logic

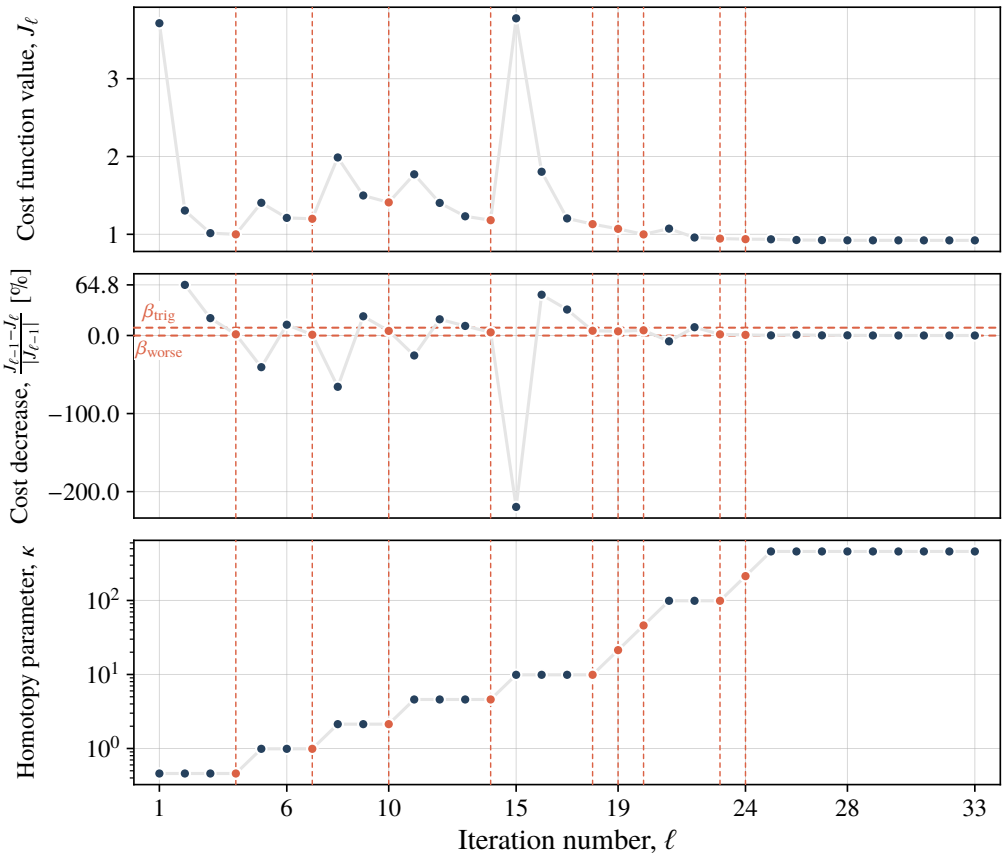


Figure 7.20: Evaluation of the cost function (7.41a) over the PTR iterations. Each time that the cost decrease falls within the decision range of (7.47), the homotopy parameter is updated (the corresponding iterations are marked by red dashed vertical lines and markers). Note the log scale for κ.

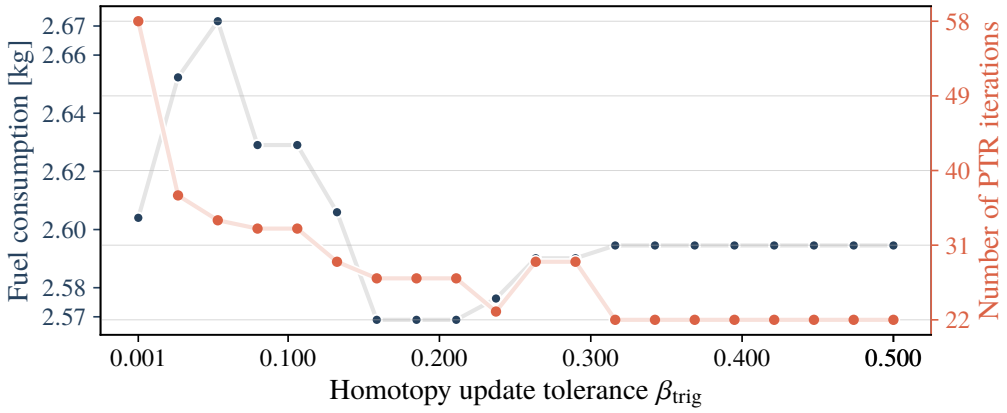


Figure 7.21: Dependence of the converged trajectory's fuel consumption and of our algorithm's total iteration count on the value of β_{trig} in (7.47). As the homotopy update tolerance increases, the iteration count falls dramatically (by over 60%) with negligible change in total fuel consumption.

constraints: RCS thruster minimum impulse-bit, approach cone, and plume impingement. We demonstrate the algorithm on a realistic setup inspired by the Apollo Transposition and Docking maneuver. The algorithm is able to consistently find fuel optimal trajectories with favorable runtimes that indicate a potential for real-time performance. Our view is that this method can already be used on the ground as a useful engineering and research tool, and has the potential to be an onboard autonomous docking guidance algorithm.

Chapter 8

OUTLOOK

This thesis develops four algorithms based on convex optimization-based techniques for the next generation of aerospace control systems. In Chapter 3, a robust model predictive control method is presented that is based purely on convex optimization and that is applicable to satellite position control in low Earth orbit, where the Gates thruster execution error model is used to represent actuation inaccuracy. In Chapter 4, a massively parallel explicit model predictive control law is presented that applies to a general class of mixed-integer convex optimization problems. In Chapter 6, again looking at problems with integer variables, a lossless convexification is introduced. The maximum principle is used to show that the convexification is indeed lossless for a class of mixed-integer optimal control problems that includes certain satellite docking and rocket landing applications. Finally, Chapter 7 uses sequential convex programming and numerical continuation to solve a six degree of freedom terminal rendezvous trajectory generation problem that includes several "discrete logic" integer constraints representing "if-then" statements embedded into the trajectory specification. In all of these applications, convex optimization is used either alone or in an iterative scheme in order to achieve real-time execution speeds and/or levels of robustness that are otherwise rare to come by in general nonconvex programming.

It is only appropriate to conclude this thesis by surveying some interesting and important future directions for optimization-based space vehicle control.

8.1 Guaranteed Performance

When proposing a new control algorithm for a real system, it is sobering to remember that the vehicle's survival, along with that of its occupants, literally hangs in the balance

[209]. The modern controls engineer has immense responsibility both to mission success and to upholding the foundation of trust created by the high reliability of traditional control methods. If we cannot guarantee an equal or greater level of reliability for optimization-based control, then these methods will quite certainly be relegated to a ground support role [210].

A direction of great importance for optimization-based space vehicle control, then, is to rigorously certify that this new family of algorithms converges to solutions which yield safe and robust operation in the real world. Active research is being done in the area, but general results are limited and many promising optimization-based methods lack proper guarantees. Today, researchers are looking at real-time performance [54, 55, 70], optimality [211], and convergence [66, 212]. Perhaps the most important yet difficult guarantee is on algorithm convergence, which is imperative for control. In the convex setting, algorithms with guaranteed convergence are available and have been flight-tested [1, 42], so one direction to explore is how to convexify more general types of nonlinearity [52, 213, 214]. For more difficult nonlinearities which are not convexifiable, an emerging subject of *funnel libraries* is being investigated [202, 215, 216, 217]. The idea, akin to explicit MPC, is to pre-compute a lookup table of trajectories and invariant controllers in order to replace on-board optimization with the more reliable task of numerical integration.

8.2 Machine Learning

Impressive advances in machine learning, and particularly in reinforcement learning (RL), could not side-ste space vehicle control without due consideration [218]. The main advantage of RL is that it is able to optimize over a stochastic data stream rather than assuming a particular description of a dynamic model [219, 220]. As an optimization tool for nonlinear stochastic systems, it is not surprising that the RL method is attractive for aerospace control.

Although RL for space vehicle control is less than a decade old, a certain amount of literature is now available which addresses many relevant spaceflight applications. The reader is referred to [221] for a dedicated survey. In powered descent guidance, [222] use deep

RL (DRL) for lunar landing, [223] improve ZEM/ZEV guidance via DRL, and [224] use recursive RL for Mars landing. For spacecraft rendezvous, [225] use actor-critic RL (ACRL) in near-rectilinear orbits, [226] consider cluttered environments, and [227, 228] use inverse RL to learn the target's behaviour. In re-entry guidance, [229] aim for real-time computation by training a deep neural network (DNN) to learn the functional relationship between state-action pairs obtained from a high-fidelity optimizer. Alternatively, [230] use a DNN to provide a numerical predictor-corrector guidance algorithm with a range prediction based on the current vehicle state. This method improves runtime performance by replacing traditional propagation-based trajectory prediction with a neural network. A different line of work is presented in[231], where the attitude of a re-entry vehicle with model uncertainty and external disturbances is controlled by a robust adaptive fuzzy PID-type sliding mode controller designed using a radial basis function neural network. For small-body landing, [232, 233, 234] use RL meta-learning for greater adaptability, and [235] train several DNNs to approximate a nonlinear gravity field as well as the optimal solution obtained by an indirect optimal control solver. Another interesting approach was proposed in [236], where DNNs are used to supply good costate initial guesses, while an accurate trajectory is obtained by a downstream shooting method and a homotopy process. In orbit insertion and transfer applications, [237] develop a multiscale DNN architecture to approximate the optimal solution for a solar-sail mission, [238] use ACRL for low-thrust trajectory optimization under changing dynamics, [239] use RL for libration point transfer in lunar applications, and [240, 241] use proximal policy optimization.

A promising modern direction for spacecraft trajectory RL is to learn a small number of "behind the scenes" parameters (called *solution hyperparameters*) that govern the optimal solution, instead of directly learning the high-dimensional optimal state-input map. Most importantly, the relationship between these parameters and the control policy is much more predictable, and hence can be learned more easily and with less training data. This thesis makes it clear that most if not all spaceflight trajectory generation problems can be formulated as a variable of the optimal control problem Problem 7.42. Hence, the solution

hyperparameters are often the maximum principle costates, or combinations thereof, which completely define the optimal control policy. Among these, we find aforementioned concepts of a *primer vector* [242, 243, 244], and switching functions for bang-bang controls [193]. This RL approach was shown to be effective for 3-DoF PDG in [245, 246], where the authors learn 10 hyperparameters instead of the map from a 7D state to a 3D input. Most importantly, only $\approx 10^3$ training trajectories were required. In comparison, the state-input map learning approach of [247] also achieved good results, but required $\approx 10^7$ training samples. A slightly different approach was taken for 3-DoF small-body landing in [236], where homotopy and coordinate transforms were used to learn a 5D costate vector instead of the map from a 7D state to a 3D input. The DNN's output was then used to provide accurate initial guesses and to improve the convergence of a downstream shooting method. To summarize, the fact that learning hyperparameters works better than learning the optimal state-input mapping is just an observation of the fact that application domain knowledge can go a long way towards improving learning performance [248]. In the case of spacecraft trajectory optimization, this knowledge often comes from applying Pontryagin's maximum principle.

As discussed the previous section, performance guarantees for an RL-based controller will have to be provided before serious on-board consideration. This may be harder to achieve for RL, since controllers are typically based on neural networks whose out-of-sample performance is still very difficult to characterize. Nevertheless, RL and other machine learning approaches are appealing for adaptive control systems. Future research will likely see the aerospace control community search for the right opportunities where RL can be embedded to improve traditional control systems.

BIBLIOGRAPHY

[1] Daniel P. Scharf et al. "Implementation and Experimental Demonstration of Onboard Powered-Descent Guidance". In: *Journal of Guidance, Control, and Dynamics* 40.2 (Feb. 2017), pp. 213–229. DOI: 10.2514/1.g000399.

[2] Max Bajracharya, Mark W. Maimone, and Daniel Helmick. "Autonomy for Mars Rovers: Past, Present, and Future". In: *Computer* 41.12 (Dec. 2008), pp. 44–50. DOI: 10.1109/mc.2008.479.

[3] A. Steltzner et al. "Mars Science Laboratory Entry, Descent, and Landing System". In: *2006 IEEE Aerospace Conference*. IEEE, 2006. DOI: 10.1109/aero.2006.1655796.

[4] Gerard J. Holzmann. "Landing a Spacecraft on Mars". In: *IEEE Software* 30.2 (Mar. 2013), pp. 83–86. DOI: 10.1109/ms.2013.32.

[5] David Rutishauser et al. "High Performance Computing for Precision Landing and Hazard Avoidance and Co-Design Approach". In: *2019 IEEE Aerospace Conference*. IEEE, Mar. 2019. DOI: 10.1109/aero.2019.8741888.

[6] Wesley A. Powell. "High-Performance Spaceflight Computing (HPSC) Project Overview". In: *Radiation Hardened Electronics Technology Conference (RHET) 2018*. NASA Goddard Space Flight Center, Nov. 2018.

[7] Alan Cudmore. "High-Performance Spaceflight Computing (HPSC) Middleware Overview". In: *2018 Flight Software Workshop*. NASA Goddard Space Flight Center, Dec. 2019.

[8] NASA Jet Propulsion Laboratory. *Curiosity's Seven Minutes of Terror*. https://www.jpl.nasa.gov/video/details.php?id=1090. June 2012.

[9] Daniel P. Scharf et al. "ADAPT demonstrations of onboard large-divert Guidance with a VTVL rocket". In: *2014 IEEE Aerospace Conference*. IEEE, Mar. 2014. DOI: 10.1109/aero.2014.6836462.

[10] David A. Mindell. *Digital Apollo: Human and Machine in Spaceflight*. The MIT Press, 2011. ISBN: 9780262516105.

[11] A. Miguel San Martin and Edward C. Lee Steven W. Wong. "The development of the MSL guidance, navigation, and control system for entry, descent, and landing". In: *23rd Space Flight Mechanics Meeting*. 2013.

[12] Edward A. Robertson. "Synopsis of Precision Landing and Hazard Avoidance (PL&HA) Capabilities for Space Exploration". In: *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, Jan. 2017. DOI: 10.2514/6.2017-1897.

[13] Europa Study Team. *Europa Study 2012 Report: Europa Lander Mission*. Tech. rep. Task Order NMO711062 Outer Planets Flagship Mission. Jet Propulsion Laboratory, 2012.

[14] Alicia M. Dwyer-Cianciolo et al. "Defining Navigation Requirements for Future Missions". In: *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: 10.2514/6.2019-0661.

[15] NASA Science. *Moon's South Pole in NASA's Landing Sites*. https://solarsystem.nasa.gov/news/907/moons-south-pole-in-nasas-landing-sites/. Accessed: 11/26/2019. Apr. 2019.

[16] Mark Robinson. *Lunar Reconnaissance Orbiter Camera (LROC)*. http://lroc.sese.asu.edu/posts/993. Accessed: 11/26/2019. Feb. 2018.

[17] Jeffrey Delmerico et al. "The current state and future outlook of rescue robotics". In: *Journal of Field Robotics* 36.7 (Aug. 2019), pp. 1171–1191. DOI: 10.1002/rob.21887.

[18] Raffaello D'Andrea. "Guest Editorial Can Drones Deliver?" In: *IEEE Transactions on Automation Science and Engineering* 11.3 (July 2014), pp. 647–648. DOI: 10.1109/tase.2014.2326952.

[19] Amazon. *Amazon Prime Air*. https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

[20] Google. *Google Wing*. https://x.company/projects/wing/.

[21] Yi-Wei Huang et al. "Duckiefloat: a Collision-Tolerant Resource-Constrained Blimp for Long-Term Autonomy in Subterranean Environments". In: *arXiv e-prints*, arXiv:1910.14275 (Oct. 2019), arXiv:1910.14275. arXiv: 1910.14275 [cs.RO].

[22] Danylo Malyuta et al. "Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition". In: *Journal of Field Robotics* (Aug. 2019), arXiv:1908.06381. DOI: 10.1002/rob.21898.

[23] Christopher D. Regan and Christine V. Jutte. *Survey of Applications of Active Control Technology for Gust Alleviation and New Challenges for Lighter-weight Aircraft*. Tech. rep. Edwards, CA, United States: NASA Dryden Flight Research Center, Apr. 2012.

[24] Yufeng Chen et al. "Controlled flight of a microrobot powered by soft artificial muscles". In: *Nature* 575.7782 (Nov. 2019), pp. 324–329. DOI: 10.1038/s41586-019-1737-7.

[25] Thomas George Thuruthel et al. "Control Strategies for Soft Robotic Manipulators: A Survey". In: *Soft Robotics* 5.2 (Apr. 2018), pp. 149–163. DOI: 10.1089/soro.2017.0007.

[26] Zhenishbek Zhakypov et al. "Designing minimal and scalable insect-inspired multi-locomotion millirobots". In: *Nature* 571.7765 (July 2019), pp. 381–386. DOI: 10.1038/s41586-019-1388-8.

[27] Jungwon Seo, Jamie Paik, and Mark Yim. "Modular Reconfigurable Robotics". In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (May 2019), pp. 63–88. DOI: 10.1146/annurev-control-053018-023834.

[28] John M. Carson et al. "The SPLICE Project: Continuing NASA Development of GN&C Technologies for Safe and Precise Landing". In: *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: 10.2514/6.2019-0660.

[29] D.Q. Mayne et al. "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6 (June 2000), pp. 789–814. DOI: 10.1016/s0005-1098(99)00214-9.

[30] Dylan Hadfield-Menell et al. "Inverse Reward Design". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 6765–6774.

[31] Rohin Shah et al. "On the Feasibility of Learning, Rather than Assuming, Human Biases for Reward Inference". In: *Thirty-sixth International Conference on Machine Learning*. ICML, June 2019.

[32] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

[33] Dimitri P. Bertsekas. *Convex Optimization Algorithms*. Athena Scientific, 2015. ISBN: 1886529280.

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

[35] Behçet Açıkmeşe and Scott R. Ploen. "Convex Programming Approach to Powered Descent Guidance for Mars Landing". In: *Journal of Guidance, Control, and Dynamics* 30.5 (Sept. 2007), pp. 1353–1366. DOI: 10.2514/1.27553.

[36] Lars Blackmore. "Autonomous Precision Landing of Space Rockets". In: *The Bridge* 4.46 (Dec. 2016), pp. 15–20.

[37] Xinfu Liu, Ping Lu, and Binfeng Pan. "Survey of convex optimization for aerospace applications". In: *Astrodynamics* 1.1 (Sept. 2017), pp. 23–40. DOI: 10.1007/s42064-017-0003-8.

[38] Behçet Açıkmeşe, John M. Carson, and David S. Bayard. "A robust model predictive control algorithm for incrementally conic uncertain/nonlinear systems". In: *International Journal of Robust and Nonlinear Control* 21.5 (July 2010), pp. 563–590. DOI: 10.1002/rnc.1613.

[39] L. S. Pontryagin et al. *The Mathematical Theory of Optimal Processes*. Pergamon Press Limited, 1964.

[40] Leonard D. Berkovitz. *Optimal Control Theory*. Springer New York, 1974. DOI: 10.1007/978-1-4757-6097-2.

[41] Daniel Liberzon. *Calculus of variations and optimal control theory : a concise introduction*. Princeton, N.J: Princeton University Press, 2012. ISBN: 9781400842643.

[42] Daniel Dueri et al. "Customized Real-Time Interior-Point Methods for Onboard Powered-Descent Guidance". In: *Journal of Guidance, Control, and Dynamics* 40.2 (Feb. 2017), pp. 197–212. DOI: 10.2514/1.g001480.

[43] Alexander Domahidi, Eric Chu, and Stephen Boyd. "ECOS: An SOCP solver for embedded systems". In: *2013 European Control Conference (ECC)*. IEEE, July 2013. DOI: 10.23919/ecc.2013.6669541.

[44] Anders Forsgren, Philip E. Gill, and Margaret H. Wright. "Interior Methods for Nonlinear Optimization". In: *SIAM Review* 44.4 (Jan. 2002), pp. 525–597. DOI: 10.1137/s0036144502414942.

[45] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, Jan. 1994. DOI: 10.1137/1.9781611970791.

[46] Alessandro Alessio and Alberto Bemporad. "A Survey on Explicit Model Predictive Control". In: *Nonlinear Model Predictive Control*. Springer Berlin Heidelberg, 2009, pp. 345–369. DOI: 10.1007/978-3-642-01094-1_29.

[47] D.Q. Mayne. "Robust and Stochastic MPC: Are We Going In The Right Direction?" In: *IFAC-PapersOnLine* 48.23 (2015), pp. 1–8. DOI: 10.1016/j.ifacol.2015.11.255.

[48] Samsung. *Samsung Breaks Terabyte Threshold for Smartphone Storage with Industry's First 1TB Embedded Universal Flash Storage*. https://news.samsung.com/global/samsung-breaks-terabyte-threshold-for-smartphone-storage-with-industrys-first-1tb-embedded-universal-flash-storage. Jan. 2019.

[49] Danylo Malyuta et al. "Convex Optimization for Trajectory Generation". In: *IEEE Control Systems Magazine (work in progress)* (2021).

[50] Danylo Malyuta et al. "Advances in Trajectory Optimization for Space Vehicle Control". In: *Annual Reviews in Control* (2021). accepted.

[51] Danylo Malyuta and Behçet Açıkmeşe. "Lossless convexification of non-convex optimal control problems with disjoint semi-continuous inputs". In: *Preprint* (Nov. 2019), arXiv:1902.02726.

[52] Danylo Malyuta and Behçet Açıkmeşe. "Lossless Convexification of Optimal Control Problems with Semi-continuous Inputs". In: *21st IFAC World Congress (submitted)*. Nov. 2019, arXiv:1911.09013.

[53] Danylo Malyuta et al. "Discretization Performance and Accuracy Analysis for the Rocket Powered Descent Guidance Problem". In: *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: 10.2514/6.2019-0925.

[54] Danylo Malyuta et al. "Fast Trajectory Optimization via Successive Convexification for Spacecraft Rendezvous with Integer Constraints". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020, arXiv:1906.04857. DOI: 10.2514/6.2020-0616.

[55] Danylo Malyuta and Behçet Açıkmeşe. "Fast Homotopy for Spacecraft Rendezvous Trajectory Optimization with Discrete Logic". In: *Journal of Guidance, Control, and Dynamics (submitted)* (June 2021).

[56] Danylo Malyuta et al. "Partition-based Feasible Integer Solution Pre-computation for Hybrid Model Predictive Control". In: *2019 18th European Control Conference (ECC)*. IEEE, June 2019, arXiv:1902.10989. DOI: 10.23919/ecc.2019.8795790.

[57] Danylo Malyuta and Behçet Açıkmeşe. "Approximate Multiparametric Mixed-Integer Convex Programming". In: *IEEE Control Systems Letters* 4.1 (Jan. 2020), arXiv:1902.10994. DOI: 10.1109/lcsys.2019.2922639.

[58] Danylo Malyuta, Behcet Acikmese, and Martin Cacan. "Robust Model Predictive Control for Linear Systems with State and Input Dependent Uncertainties". In: *2019 American Control Conference (ACC)*. IEEE, July 2019, arXiv:1902.10984. DOI: 10.23919/acc.2019.8815343.

[59] Behçet Açıkmeşe and Lars Blackmore. "Lossless convexification of a class of optimal control problems with non-convex control constraints". In: *Automatica* 47.2 (Feb. 2011), pp. 341–347. DOI: 10.1016/j.automatica.2010.10.037.

[60] Matthew W. Harris and Behçet Açıkmeşe. "Lossless convexification of non-convex optimal control problems for state constrained linear systems". In: *Automatica* 50.9 (Sept. 2014), pp. 2304–2311. DOI: 10.1016/j.automatica.2014.06.008.

[61] Matthew Wade Harris. "Lossless Convexification of Optimal Control Problems". PhD thesis. Austin: The University of Texas at Austin, 2014.

[62] C. R. Gates. *A Simplified Model of Midcourse Maneuver Execution Errors*. Tech. rep. 32-504. Pasadena, CA: JPL, Oct. 1963.

[63] T. D. Goodson. "Execution-Error Modeling and Analysis of the GRAIL Spacecraft Pair". In: *AAS/AIAA Space Flight Mechanics Meeting*. Aug. 2013.

[64] S. V. Wagner. "Maneuver performance assessment of the Cassini spacecraft through execution-error modeling and analysis". In: *AAS/AIAA Space Flight Mechanics Meeting*. Jan. 2014.

[65] Alberto Bemporad and Manfred Morari. "Control of systems integrating logic, dynamics, and constraints". In: *Automatica* 35.3 (Mar. 1999), pp. 407–427. DOI: 10.1016/s0005-1098(98)00178-2.

[66] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems". In: *arXiv e-prints*, arXiv:1804.06539 (Apr. 2018), arXiv:1804.06539. arXiv: 1804.06539 [math.OC].

[67] Michael Szmuk, Taylor P. Reynolds, and Behçet Açıkmeşe. "Successive Convexification for Real-Time Six-Degree-of-Freedom Powered Descent Guidance with State-Triggered Constraints". In: *Journal of Guidance, Control, and Dynamics* 43.8 (Aug. 2020), pp. 1399–1413. DOI: 10.2514/1.g004549.

[68] Taylor P. Reynolds et al. "Dual Quaternion-Based Powered Descent Guidance with State-Triggered Constraints". In: *Journal of Guidance, Control, and Dynamics* 43.9 (Sept. 2020), pp. 1584–1599. DOI: 10.2514/1.g004536.

[69] Michael Szmuk et al. "Real-Time Quad-Rotor Path Planning Using Convex Optimization and Compound State-Triggered Constraints". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Nov. 2019, pp. 7666–7673. DOI: 10.1109/iros40897.2019.8967706.

[70] Taylor Reynolds et al. "A Real-Time Algorithm for Non-Convex Powered Descent Guidance". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-0844.

[71] Michael Szmuk et al. "Successive Convexification for 6-DoF Powered Descent Guidance with Compound State-Triggered Constraints". In: (Jan. 2019), pp. 1–16. DOI: 10.2514/6.2019-0926.

[72] Ralph Tyrell Rockafellar. *Convex Analysis*. Princeton University Press, 1996. ISBN: 0691015864.

[73] R. Tyrrell Rockafellar. "Lagrange Multipliers and Optimality". In: *SIAM Review* 35.2 (June 1993), pp. 183–238. DOI: 10.1137/1035044.

[74] N. Karmarkar. "A new polynomial time algorithm for linear programming". In: *Combinatorics* 4 (1984), pp. 373–395.

[75] Stephen J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, Jan. 1997. DOI: 10.1137/1.9781611971453.

[76] Jiming Peng, Cornelis Roos, and Tamás Terlaky. *Self-Regularity: A New Paradigm for Primal-Dual Interior-Point Algorithms*. Princeton University Press, 2002. ISBN: 9780691091938.

[77] Jiming Peng, Cornelis Roos, and Tamás Terlaky. "Primal-Dual Interior-Point Methods for Second-Order Conic Optimization Based on Self-Regular Proximities". In: *SIAM Journal on Optimization* 13.1 (Jan. 2002), pp. 179–203. DOI: 10.1137/s1052623401383236.

[78] R. Tyrrell Rockafellar and Roger J. B. Wets. *Variational Analysis*. Springer Berlin Heidelberg, 1998. DOI: 10.1007/978-3-642-02431-3.

[79] R. V. Gamkrelidze. "History of the Discovery of the Pontryagin Maximum Principle". In: *Proceedings of the Steklov Institute of Mathematics* 304.1 (Jan. 2019), pp. 1–7. DOI: 10.1134/s0081543819010012.

[80] L. S. Pontryagin et al. *The Mathematical Theory of Optimal Processes*. [in Russian]. Moscow: Fizmatgiz, 1961.

[81] L. S. Pontryagin et al. *The Mathematical Theory of Optimal Processes*. New York: Interscience, 1962.

[82] Francis Clarke. "The Pontryagin maximum principle and a unified theory of dynamic optimization". In: *Proceedings of the Steklov Institute of Mathematics* 268.1 (Apr. 2010), pp. 58–69. DOI: 10.1134/s0081543810010062.

[83] Michael Carter. *Foundations of Mathematical Economics*. The MIT Press, 2001. ISBN: 0262531925.

[84] Leon S. Lasdon. *Optimization Theory for Large Systems*. Dover Publications, 2013.

[85] John T. Betts. "Survey of Numerical Methods for Trajectory Optimization". In: *Journal of Guidance, Control, and Dynamics* 21.2 (Mar. 1998), pp. 193–207. DOI: 10.2514/2.4231.

[86] Richard Vinter. *Optimal Control*. Birkhauser, 2000. ISBN: 0817640754.

[87] Richard F. Hartl, Suresh P. Sethi, and Raymond G. Vickson. "A Survey of the Maximum Principles for Optimal Control Problems with State Constraints". In: *SIAM Review* 37.2 (June 1995), pp. 181–218. DOI: 10.1137/1037043.

[88] Alberto Bemporad and Manfred Morari. "Robust model predictive control: A survey". In: *Robustness in identification and control*. Springer London, 2007, pp. 207–226. DOI: 10.1007/bfb0109870.

[89] David Q. Mayne. "Model predictive control: Recent developments and future promise". In: *Automatica* 50.12 (Dec. 2014), pp. 2967–2986. DOI: 10.1016/j.automatica.2014.10.128.

[90] M. Milanese and A. Vicino. "Information-Based Complexity and Nonparametric Worst-Case System Identification". In: *Journal of Complexity* 9.4 (Dec. 1993), pp. 427–446. DOI: 10.1006/jcom.1993.1028.

[91] P.M. Mäkilä, J.R. Partington, and T.K. Gustafsson. "Worst-case control-relevant identification". In: *Automatica* 31.12 (Dec. 1995), pp. 1799–1819. DOI: 10.1016/0005-1098(95)00106-3.

[92] D.L. Marruedo, T. Alamo, and E.F. Camacho. "Input-to-state stable MPC for constrained discrete-time nonlinear systems with bounded additive uncertainties". In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* IEEE, Dec. 2002. DOI: 10.1109/cdc.2002.1185106.

[93] J.H. Lee and Zhenghong Yu. "Worst-case formulations of model predictive control for systems with bounded parameters". In: *Automatica* 33.5 (May 1997), pp. 763–781. DOI: 10.1016/s0005-1098(96)00255-5.

[94] A. Bemporad. "Reducing conservativeness in predictive control of constrained systems with disturbances". In: *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171).* IEEE, 1998. DOI: 10.1109/cdc.1998.758479.

[95] P.O.M. Scokaert and David Q. Mayne. "Min-max feedback model predictive control for constrained linear systems". In: *IEEE Transactions on Automatic Control* 43.8 (1998), pp. 1136–1142. DOI: 10.1109/9.704989.

[96] Eric C. Kerrigan and J. M. Maciejowski. "Feedback min-max model predictive control using a single linear program: robust stability and the explicit solution". In: *International Journal of Robust and Nonlinear Control* 14.4 (2004), pp. 395–413.

[97] Johan Löfberg. "Minimax approaches to robust model predictive control". Dissertation (Ph.D.) Linköping University, 2003.

[98] Franco Blanchini. "Feedback control for linear time-invariant systems with state and control bounds in the presence of disturbances". In: *IEEE Transactions on Automatic Control* 35.11 (1990), pp. 1231–1234. DOI: 10.1109/9.59808.

[99] Franco Blanchini. "Control synthesis for discrete time systems with control and state bounds in the presence of disturbances". In: *Journal of Optimization Theory and Applications* 65.1 (Apr. 1990), pp. 29–40. DOI: 10.1007/bf00941157.

[100] Franco Blanchini. "Constrained control for perturbed linear systems". In: *29th IEEE Conference on Decision and Control*. IEEE, 1990. DOI: 10.1109/cdc.1990.203446.

[101] Franco Blanchini. "Ultimate boundedness control for uncertain discrete-time systems via set-induced Lyapunov functions". In: *IEEE Transactions on Automatic Control* 39.2 (1994), pp. 428–433. DOI: 10.1109/9.272351.

[102] G. Pin et al. "Robust Model Predictive Control of Nonlinear Systems With Bounded and State-Dependent Uncertainties". In: *IEEE Transactions on Automatic Control* 54.7 (July 2009), pp. 1681–1687. DOI: 10.1109/tac.2009.2020641.

[103] Y.I. Lee, B. Kouvaritakis, and M. Cannon. "Constrained receding horizon predictive control for nonlinear systems". In: *Automatica* 38.12 (Dec. 2002), pp. 2093–2102. DOI: 10.1016/s0005-1098(02)00133-4.

[104] D. Limon, T. Alamo, and E.F. Camacho. "Robust stability of min-max MPC controllers for nonlinear systems with bounded uncertainties". In: *Proceedings of the 16th Mathematical Theory of Networks and Systems Conference.* 2004.

[105] Saša V. Raković. "Set Theoretic Methods in Model Predictive Control". In: *Nonlinear Model Predictive Control*. Springer Berlin Heidelberg, 2009, pp. 41–54. DOI: 10.1007/978-3-642-01094-1_3.

[106] M. Cannon and B. Kouvaritakis. "Optimizing prediction dynamics for robust MPC". In: *IEEE Transactions on Automatic Control* 50.11 (Nov. 2005), pp. 1892–1897. DOI: 10.1109/tac.2005.858679.

[107] Daniel Dueri, Jing Zhang, and Behçet Açıkmeşe. "Automated Custom Code Generation for Embedded, Real-time Second Order Cone Programming". In: *IFAC Pro-*

ceedings Volumes 47.3 (2014), pp. 1605–1612. DOI: 10.3182/20140824-6-za-1003.02736.

[108] Melanie N. Zeilinger et al. "On real-time robust model predictive control". In: *Automatica* 50.3 (Mar. 2014), pp. 683–694. DOI: 10.1016/j.automatica.2013.11.019.

[109] Franco Blanchini and Stefano Miani. *Set-Theoretic Methods in Control*. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-17933-9.

[110] Michal Kvasnica et al. "Reachability Analysis and Control Synthesis for Uncertain Linear Systems in MPT". In: *IFAC-PapersOnLine* 48.14 (2015), pp. 302–307. DOI: 10.1016/j.ifacol.2015.09.474.

[111] Saša V. Raković et al. "Reachability Analysis of Discrete-Time Systems With Disturbances". In: *IEEE Transactions on Automatic Control* 51.4 (Apr. 2006), pp. 546–561. DOI: 10.1109/tac.2006.872835.

[112] Mato Baotić. "Polytopic computations in constrained optimal control". In: *Automatika, Journal for Control, Measurement, Electronics, Computing and Communications* 50 (2009), pp. 119–134.

[113] Ilya Kolmanovsky and Elmer G. Gilbert. "Theory and computation of disturbance invariant sets for discrete-time linear systems". In: *Mathematical Problems in Engineering* 4.4 (1998), pp. 317–367. DOI: 10.1155/s1024123x98000866.

[114] Panos J. Antsaklis and Anthony N. Michel. *A Linear Systems Primer*. Birkhäuser Boston, 2007. DOI: 10.1007/978-0-8176-4661-5.

[115] Daniel Dueri, Saša V. Raković, and Behçet Açıkmeşe. "Consistently improving approximations for constrained controllability and reachability". In: *2016 European Control Conference (ECC)*. IEEE, June 2016. DOI: 10.1109/ecc.2016.7810523.

[116] Lars Blackmore, Behçet Açıkmeşe, and John M. Carson III. "Lossless convexification of control constraints for a class of nonlinear optimal control problems". In: *Systems & Control Letters* 61.8 (Aug. 2012), pp. 863–870. DOI: 10.1016/j.sysconle.2012.04.010.

[117] Tom Schouwenaars. "Safe trajectory planning of autonomous vehicles". Dissertation (Ph.D.) Massachusetts Institute of Technology, 2006.

[118] Tobias Achterberg. "Constrained Integer Programming". Dissertation (Ph.D.) Technischen Universität Berlin, 2007.

[119] Yuanqi Mao et al. "Successive Convexification of Non-Convex Optimal Control Problems with State Constraints". In: *IFAC-PapersOnLine* 50.1 (July 2017), pp. 4063–4069. DOI: 10.1016/j.ifacol.2017.08.789.

[120] Michael Szmuk et al. "A Tutorial on Successive Convexification for Real-Time Rocket Landing Guidance with State-Triggered Constraints". In: *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: 10.2514/6.2019-0926.

[121] Alberto Bemporad and Carlo Filippi. "An Algorithm for Approximate Multiparametric Convex Programming". In: *Computational Optimization and Applications* 35.1 (Mar. 2006), pp. 87–108. DOI: 10.1007/s10589-006-6447-z.

[122] Efstratios N. Pistikopoulos, Michael C. Georgiadis, and Vivek Dua, eds. *Multi-Parametric Programming: Theory, Algorithms, and Applications*. Vol. 1. Wiley-VCH Verlag GmbH & Co. KGaA, Feb. 2007. DOI: 10.1002/9783527631216.

[123] Tor A. Johansen. "Approximate explicit receding horizon control of constrained nonlinear systems". In: *Automatica* 40.2 (Feb. 2004), pp. 293–300. DOI: 10.1016/j.automatica.2003.09.021.

[124] Alessandro Alessio and Alberto Bemporad. "Feasible mode enumeration and cost comparison for explicit quadratic model predictive control of hybrid systems". In: *IFAC Proceedings Volumes* 39.5 (2006), pp. 302–308. DOI: 10.3182/20060607-3-it-3902.00057.

[125] V. Dua and E.N. Pistikopoulos. "An outer-approximation algorithm for the solution of multiparametric MINLP problems". In: *Computers & Chemical Engineering* 22 (Mar. 1998), S955–S958. DOI: 10.1016/s0098-1354(98)00189-6.

[126] Munoz de la Pena, A. Bemporad, and C. Filippi. "Robust explicit MPC based on approximate multi-parametric convex programming". In: *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*. IEEE, 2004. DOI: 10.1109/cdc.2004.1428788.

[127] D. Munoz De La Pena, A. Bemporad, and C. Filippi. "Robust Explicit MPC Based on Approximate Multiparametric Convex Programming". In: *IEEE Transactions on Automatic Control* 51.8 (Aug. 2006), pp. 1399–1403. DOI: 10.1109/tac.2006.878755.

[128] A. Bemporad et al. "The explicit solution of model predictive control via multiparametric quadratic programming". In: *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*. IEEE, 2000. DOI: 10.1109/acc.2000.876624.

[129] Tomas Gal. *Postoptimal Analyses, Parametric Programming, and Related Topics*. 2nd ed. De Gruyter, 2010. ISBN: 3110140608.

[130] Günter M. Ziegler. *Lectures on Polytopes*. Springer New York, 1995. DOI: 10.1007/978-1-4613-8431-1.

[131] Mark de Berg et al. *Computational Geometry*. 3rd ed. Springer Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-77974-2.

[132] MOSEK ApS. *MOSEK Optimizer API for Python 9.0.87*. 2019.

[133] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2018.

[134] Halsey Royden. *Real Analysis*. 3rd ed. Pearson, 1988. ISBN: 0024041513.

[135] Thomas H. Cormen et al. *Introduction to Algorithms*. MIT Press, 2010. ISBN: 9788120340077.

[136] P. Stein. "A Note on the Volume of a Simplex". In: *The American Mathematical Monthly* 73.3 (Mar. 1966), p. 299. DOI: 10.2307/2315353.

[137] Steven Diamond and Stephen Boyd. "CVXPY: A Python-Embedded Modeling Language for Convex Optimization". In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[138] Behcet Açıkmeşe and Scott Ploen. "A Powered Descent Guidance Algorithm for Mars Pinpoint Landing". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. AIAA, Aug. 2005. DOI: 10.2514/6.2005-6288.

[139] Lars Blackmore, Behçet Açıkmeşe, and Daniel P. Scharf. "Minimum-Landing-Error Powered-Descent Guidance for Mars Landing Using Convex Optimization". In: *Journal of Guidance, Control, and Dynamics* 33.4 (July 2010), pp. 1161–1171. DOI: 10.2514/1.47202.

[140] John M. Carson III, Behçet Açıkmeşe, and Lars Blackmore. "Lossless convexification of Powered-Descent Guidance with non-convex thrust bound and pointing constraints". In: *Proceedings of the 2011 American Control Conference*. IEEE, June 2011. DOI: 10.1109/acc.2011.5990959.

[141] Matthew W. Harris and Behçet Açıkmeşe. "Lossless convexification for a class of optimal control problems with quadratic state constraints". In: *2013 American Control Conference*. IEEE, June 2013. DOI: 10.1109/acc.2013.6580359.

[142] Matthew W. Harris. "Optimal Control on Disconnected Sets using Extreme Point Relaxations and Normality Approximations". In: *IEEE Transactions on Automatic Control* (2021), pp. 1–1. DOI: 10.1109/tac.2021.3059682.

[143] Sheril Kunhippurayil, Matthew W. Harris, and Olli Jansson. "Lossless Convexification of Optimal Control Problems with Annular Control Constraints". In: *Automatica* (2021).

[144] Sheril Kunhippurayil and Matthew W. Harris. "Strong Observability as a Sufficient Condition for Non-singularity in Optimal Control with Mixed Constraints". In: *Automatica* (2021).

[145] Henry D'Angelo. *Linear Time-Varying Systems: Analysis and Synthesis*. Allyn and Bacon, 1970.

[146] Behçet Açıkmeşe, John M. Carson III, and Lars Blackmore. "Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem". In: *IEEE Transactions on Control Systems Technology* 21.6 (Nov. 2013), pp. 2104–2113. DOI: 10.1109/tcst.2012.2237346.

[147] L. D. Landau and E. M. Lifshitz. *Mechanics*. 2nd ed. Bristol, UK: Pergamon Press, 1969.

[148] Warren F. Phillips. *Mechanics of Flight*. Hoboken, NJ: Wiley, 2010. ISBN: 9780470539750.

[149] Anton H. J. de Ruiter, Christopher J. Damaren, and James R. Forbes. *Spacecraft Dynamics and Control: An Introduction*. Wiley, 2013. ISBN: 9781118342367.

[150] H. L. Trentelman, A. A. Stoorvogel, and M. Hautus. *Control Theory for Linear Systems*. Springer, 2001.

[151] Matthew W. Harris and Behçet Açıkmeşe. "Lossless convexification for a class of optimal control problems with linear state constraints". In: *52nd IEEE Conference on Decision and Control*. IEEE, Dec. 2013. DOI: 10.1109/cdc.2013.6761017.

[152] Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. Wiley, 2005. ISBN: 9780470011676.

[153] A. Milyutin and N. Osmolovskii. *Calculus of Variations and Optimal Control*. American Mathematical Society, 1998.

[154] Behçet Açıkmeşe et al. "Flight Testing of Trajectories Computed by G-FOLD: Fuel Optimal Large Divert Guidance Algorithm for Planetary Landing". In: *23rd AAS/AIAA Space Flight Mechanics Meeting, Kauai, HI, 2013*. AAS/AIAA, Feb. 2013.

[155] JPL and Masten Space Systems. "750 meter divert Xombie test flight for G-FOLD, Guidance for Fuel Optimal Large Divert, validation". In: *http://www.youtube.com/watch?v=jl6pw2o* (July 2012).

[156] JPL and Masten Space Systems. "500 meter divert Xombie test flight for G-FOLD, Guidance for Fuel Optimal Large Divert, validation". In: *http://www.youtube.com/watch?v=1GRwi* (Aug. 2012).

[157] Tobias Achterberg and Roland Wunderling. "Mixed Integer Programming: Analyzing 12 Years of Progress". In: *Facets of Combinatorial Optimization*. Springer Berlin Heidelberg, 2013, pp. 449–481. DOI: 10.1007/978-3-642-38189-8_18.

[158] Tom Schouwenaars et al. "Mixed integer programming for multi-vehicle path planning". In: *2001 European Control Conference (ECC)*. IEEE, Sept. 2001. DOI: 10.23919/ecc.2001.7076321.

[159] Michael Szmuk et al. "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2017. DOI: 10.1109/iros.2017.8206363.

[160] Zhe Zhang, Jun Wang, and Jianxun Li. "Lossless convexification of nonconvex MINLP on the UAV path-planning problem". In: *Optimal Control Applications and Methods* 39.2 (Dec. 2017), pp. 845–859. DOI: 10.1002/oca.2380.

[161] Xinfu Liu, Zuojun Shen, and Ping Lu. "Entry Trajectory Optimization by Second-Order Cone Programming". In: *Journal of Guidance, Control, and Dynamics* 39.2 (Feb. 2016), pp. 227–241. DOI: 10.2514/1.g001210.

[162] Xinfu Liu. "Fuel-Optimal Rocket Landing with Aerodynamic Controls". In: *Journal of Guidance, Control, and Dynamics* 42.1 (Jan. 2019), pp. 65–77. DOI: 10.2514/1.g003537.

[163] Michael Szmuk, Behçet Açıkmeşe, and Andrew W. Berning. "Successive Convexification for Fuel-Optimal Powered Landing with Aerodynamic Drag and Non-Convex Constraints". In: *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, Jan. 2016. DOI: 10.2514/6.2016-0378.

[164] Laurence A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998. ISBN: 0471283665.

[165] MOSEK ApS. *MOSEK Modeling Cookbook*. 3.1. July 2019.

[166] Alberto Bemporad and Manfred Morari. "Control of systems integrating logic, dynamics, and constraints". In: *Automatica* 35.3 (Mar. 1999), pp. 407–427. DOI: 10.1016/s0005-1098(98)00178-2.

[167] Thomas H. Cormen et al. *Introduction to Algorithms*. 3rd ed. MIT Press, 2009. ISBN: 9780262033848.

[168] Dale E. Varberg. "On Absolutely Continuous Functions". In: *The American Mathematical Monthly* 72.8 (Oct. 1965), p. 831. DOI: 10.2307/2315025.

[169] Elias M. Stein and Rami Shakarchi. *Real Analysis: Measure Theory, Integration, and Hilbert Spaces*. Princeton University Press, 2005. ISBN: 9780691113869.

[170] Mykel J. Kochenderfer and Tim A. Wheeler. *Algorithms for Optimization*. Cambridge, Massachusetts: The MIT Press, 2019. ISBN: 9780262039420.

[171] Monroe Conner and NASA Jet Propulsion Laboratory. *Masten's Xombie Tests JPL's G-FOLD Precision Landing Software*. https://www.nasa.gov/centers/dryden/Features/gfold_tests.html. Jan. 2014.

[172] NASA. *Masten Xombie for Testing of JPL Spacecraft-Landing Algorithm*. https://www.nasa.gov/centers/jpl/multimedia/pia17986.html. 2013.

[173] Matthew W. Harris and Behçet Açıkmeşe. "Minimum Time Rendezvous of Multiple Spacecraft Using Differential Drag". In: *Journal of Guidance, Control, and Dynamics* 37.2 (Mar. 2014), pp. 365–373. DOI: 10.2514/1.61505.

[174] Henry Hermes and Joseph P. LaSalle. *Functional Analysis and Time Optimal Control*. Elsevier, 1969. ISBN: 9780080955650.

[175] David Charles Woffinden and David Keith Geller. "Navigating the road to autonomous orbital rendezvous". In: *Journal of Spacecraft and Rockets* 44.4 (2007), pp. 898–909. ISSN: 0022-4650. DOI: 10.2514/1.30734.

[176] John Louis Goodman. "History of space shuttle rendezvous and proximity operations". In: *Journal of Spacecraft and Rockets* 43.5 (2006), pp. 944–959. ISSN: 0022-4650. DOI: 10.2514/1.19653.

[177] Christopher N D'Souza et al. "Orion rendezvous, proximity operations, and docking design and analysis". In: *AIAA Guidance, Navigation and Control Conference*. Hilton Head, SC, 2007, pp. 1–13. DOI: 10.2514/6.2007-6683.

[178] Hanneke Weitering. *SpaceX's 1st upgraded Dragon cargo ship docks itself at space station with science, goodies and new airlock*. https://www.space.com/spacex-cargo-dragon-crs-21-docks-at-space-station. Dec. 2020.

[179] Shin-Ichiro Nishida and Satomi Kawamoto. "Strategy for capturing of a tumbling space debris". In: *Acta Astronautica* 68.1-2 (Jan. 2011), pp. 113–120. DOI: 10.1016/j.actaastro.2010.06.045.

[180] Marshall Kaplan. "Survey of Space Debris Reduction Methods". In: *AIAA SPACE 2009 Conference & Exposition*. American Institute of Aeronautics and Astronautics, June 2009, pp. 1–11. DOI: 10.2514/6.2009-6619.

[181] Kenneth Chang. *An Orbital Rendezvous Demonstrates a Space Junk Solution*. https://www.nytimes.com/2020/02/26/science/mev-1-northrop-grumman-space-junk.html. Feb. 2020.

[182] A. Weiss et al. "Model predictive control of three dimensional spacecraft relative motion". In: *American Control Conference* (2012), pp. 173–178. DOI: 10.1109/acc.2012.6314862.

[183] Arthur Richards et al. "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming". In: *Journal of Guidance, Control, and Dynamics* 25.4 (2002), pp. 755–764. ISSN: 0731-5090. DOI: 10.2514/2.4943.

[184] Edward N. Hartley, Marco Gallieri, and Jan M. Maciejowski. "Terminal spacecraft rendezvous and capture with LASSO model predictive control". In: *International Journal of Control* 86.11 (2013), pp. 2104–2113. ISSN: 00207179. DOI: 10.1080/00207179.2013.789608.

[185] A. Miele, M. W. Weeks, and M. Ciarcià. "Optimal trajectories for spacecraft rendezvous". In: *Journal of Optimization Theory and Applications* 132.3 (2007), pp. 353–376. ISSN: 00223239. DOI: 10.1007/s10957-007-9166-4.

[186] A. Miele, M. Ciarcià, and M. W. Weeks. "Guidance trajectories for spacecraft rendezvous". In: *Journal of Optimization Theory and Applications* 132.3 (2007), pp. 377–400. ISSN: 00223239. DOI: 10.1007/s10957-007-9165-5.

[187] Carlo Alberto Pascucci, Michael Szmuk, and Behçet Açıkmeşe. "Optimal real-time force rendering for on-orbit structures assembly". In: *10th International ESA Conference on Guidance, Navigation & Control Systems*. ESA, May 2017.

[188] Louis S. Breger and Jonathan P. How. "Safe trajectories for autonomous rendezvous of spacecraft". In: *Journal of Guidance, Control, and Dynamics* 31.5 (2008), pp. 1478–1489. ISSN: 0731-5090. DOI: 10.2514/1.29590.

[189] Chuangchuang Sun, Ran Dai, and Ping Lu. "Multi-phase spacecraft mission optimization by quadratically constrained quadratic programming". In: *AIAA Scitech Forum*. San Diego, CA, 2019, pp. 1–15. ISBN: 978-1-62410-578-4. DOI: 10.2514/6.2019-1667.

[190] Jeff Phillips, Lydia Kavraki, and Nazareth Bedrossian. "Spacecraft Rendezvous and Docking with Real-Time, Randomized Optimization". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, June 2003, pp. 1–11. DOI: 10.2514/6.2003-5511.

[191] Sorin C. Bengea and Raymond A. DeCarlo. "Optimal control of switching systems". In: *Automatica* 41.1 (Jan. 2005), pp. 11–27. DOI: 10.1016/j.automatica.2004.08.003.

[192] Harish Saranathan and Michael J. Grant. "Relaxed Autonomously Switched Hybrid System Approach to Indirect Multiphase Aerospace Trajectory Optimization". In: *Journal of Spacecraft and Rockets* 55.3 (May 2018), pp. 611–621. DOI: 10.2514/1.a34012.

[193] Ehsan Taheri et al. "A novel approach for optimal trajectory design with multiple operation modes of propulsion system, part 1". In: *Acta Astronautica* 172 (July 2020), pp. 151–165. DOI: 10.1016/j.actaastro.2020.02.042.

[194] Vishala Arya, Ehsan Taheri, and John L. Junkins. "A composite framework for co-optimization of spacecraft trajectory and propulsion system". In: *Acta Astronautica* 178 (Jan. 2021), pp. 773–782. DOI: 10.1016/j.actaastro.2020.10.007.

[195] *CSM/LM Spacecraft Operation Data Book, Volume 3: Mass Properties*. SNA-8-D-027(III) REV 2. National Aeronautics and Space Administration. 1969.

[196] Howard D. Curtis. *Orbital Mechanics for Engineering Students*. 3rd ed. Waltham, MA: Butterworth-Heinemann, 2014. ISBN: 9780080977478.

[197] Joan Solà. "Quaternion kinematics for the error-state Kalman filter". In: *CoRR* (2017).

[198] *CSM/LM Spacecraft Operation Data Book, Volume 1: CSM Data Book, Part 1: Constraints and Performance*. SNA-8-D-027(I) REV 3. National Aeronautics and Space Administration. 1970.

[199] *Apollo CSM and LM News Reference: Reaction Control Subsystem*. Space Division of North American Rockwell Corp. 1969.

[200] *Apollo Operations Handbook, Block II Spacecraft, Volume 1: Spacecraft Description*. SM2A-03-Block II-(1). National Aeronautics and Space Administration. 1969.

[201] Michael Szmuk. "Successive Convexification & High Performance Feedback Control for Agile Flight". PhD thesis. Seattle: University of Washington, 2019.

[202] Taylor P. Reynolds. "Computation Guidance and Control for Aerospace Systems". PhD thesis. Seattle, WA: University of Washington, 2021.

[203] Michael Szmuk, Taylor P. Reynolds, and Behçet Açıkmeşe. "Successive Convexification for Real-Time 6-DoF Powered Descent Guidance with State-Triggered Constraints". In: *arXiv e-prints*, arXiv:1811.10803 (Nov. 2018), arXiv:1811.10803. arXiv: 1811.10803 [math.OC].

[204] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. 2nd ed. Springer New York, 2009. ISBN: 9780387848570. DOI: 10.1007/978-0-387-84858-7.

[205] John T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, 2020. ISBN: 9781611976182.

[206] Layne T. Watson. "Numerical Linear Algebra Aspects of Globally Convergent Homotopy Methods". In: *SIAM Review* 28.4 (Dec. 1986), pp. 529–545. DOI: 10.1137/1028157.

[207] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods*. SIAM, Philadelphia, PA, 2000. DOI: 10.1137/1.9780898719857.

[208] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671.

[209] Gunter Stein. "Respect the unstable". In: *IEEE Control Systems* 23.4 (Aug. 2003), pp. 12–25. DOI: 10.1109/mcs.2003.1213600.

[210] Scott Ploen, Behçet Açıkmeşe, and Aron Wolf. "A Comparison of Powered Descent Guidance Laws for Mars Pinpoint Landing". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2006. DOI: 10.2514/6.2006-6676.

[211] Taylor P. Reynolds and Mehran Mesbahi. "Optimal Planar Powered Descent with Independent Thrust and Torque". In: *Journal of Guidance, Control, and Dynamics* 43.7 (July 2020), pp. 1225–1231. DOI: 10.2514/1.g004701.

[212] Riccardo Bonalli et al. "GuSTO: Guaranteed Sequential Trajectory Optimization via Sequential Convex Programming". In: *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019.

[213] Xinfu Liu and Ping Lu. "Solving Nonconvex Optimal Control Problems by Convex Optimization". In: *Journal of Guidance, Control, and Dynamics* 37.3 (May 2014), pp. 750–765. DOI: 10.2514/1.62110.

[214] Unsik Lee and Mehran Mesbahi. "Dual Quaternion based Spacecraft Rendezvous with Rotational and Translational Field of View Constraints". In: *AIAA/AAS Astrodynamics Specialist Conference*. American Institute of Aeronautics and Astronautics, Aug. 2014. DOI: 10.2514/6.2014-4362.

[215] Anirudha Majumdar and Russ Tedrake. "Funnel libraries for real-time robust feedback motion planning". In: *The International Journal of Robotics Research* 36.8 (June 2017), pp. 947–982. DOI: 10.1177/0278364917712421.

[216] Taylor P. Reynolds et al. "Temporally-Interpolated Funnel Synthesis for Nonlinear Systems". In: *2nd RSS Workshop on Robust Autonomy*. RSS, July 2020.

[217] Behçet Açıkmeşe et al. "Enhancements on the Convex Programming Based Powered Descent Guidance Algorithm for Mars Landing". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. American Institute of Aeronautics and Astronautics, Aug. 2008. DOI: 10.2514/6.2008-6426.

[218] Panagiotis Tsiotras and Mehran Mesbahi. "Toward an Algorithmic Control Theory". In: *Journal of Guidance, Control, and Dynamics* 40.2 (Feb. 2017), pp. 194–196. DOI: 10.2514/1.g002754.

[219] Lucian Buşoniu et al. "Reinforcement learning for control: Performance, stability, and deep approximators". In: *Annual Reviews in Control* 46 (2018), pp. 8–28. DOI: 10.1016/j.arcontrol.2018.09.005.

[220] Kai Arulkumaran et al. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (Nov. 2017), pp. 26–38. ISSN: 1053-5888. DOI: 10.1109/msp.2017.2743240.

[221] Dario Izzo, Marcus Märtens, and Binfeng Pan. "A survey on artificial intelligence trends in spacecraft guidance dynamics and control". In: *Astrodynamics* 3.4 (July 2019), pp. 287–299. DOI: 10.1007/s42064-018-0053-6.

[222] Lin Cheng, Zhenbo Wang, and Fanghua Jiang. "Real-time control for fuel-optimal Moon landing based on an interactive deep reinforcement learning algorithm". In: *Astrodynamics* 3.4 (July 2019), pp. 375–386. DOI: 10.1007/s42064-018-0052-2.

[223] Roberto Furfaro et al. "Adaptive generalized ZEM-ZEV feedback guidance for planetary landing via a deep reinforcement learning approach". In: *Acta Astronautica* 171 (June 2020), pp. 156–171. DOI: 10.1016/j.actaastro.2020.02.051.

[224] Brian Gaudet and Richard Linares. *Adaptive Guidance with Reinforcement Meta-Learning*. 2019. arXiv: 1901.04473 [cs.SY].

[225] Andrea Scorsoglio et al. "Actor-Critic Reinforcement Learning Approach to Relative Motion Guidance in Near-rectilinear Orbit". In: *AAS/AIAA Space Flight Mechanics Meeting* (2019).

[226] Brian Gaudet, Richard Linares, and Roberto Furfaro. "Spacecraft rendezvous guidance in cluttered environments via artificial potential functions and reinforcement learning". In: *AAS/AIAA Astrodynamics Specialist Conference*. 2018.

[227] Bryce G. Doerr, Richard Linares, and Roberto Furfaro. "Space Objects Maneuvering Prediction via Maximum Causal Entropy Inverse Reinforcement Learning". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-0235.

[228] Richard Linares and Joseph B. Raquepas. "Physically-Constrained Inverse Optimal Control for Satellite Maneuver Detection". In: *2018 AAS/AIAA Astrodynamics Specialist Conference*. 2018.

[229] Yang Shi and Zhenbo Wang. "A Deep Learning-Based Approach to Real-Time Trajectory Optimization for Hypersonic Vehicles". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-0023.

[230] Lin Cheng et al. "Multi-Constrained Real-Time Entry Guidance Using Deep Neural Networks". In: *IEEE Transactions on Aerospace and Electronic Systems* (2020). DOI: 10.1109/taes.2020.3015321.

[231] Zhen Jin et al. "Neural network based adaptive fuzzy PID-type sliding mode attitude control for a reentry vehicle". In: *International Journal of Control, Automation and Systems* 15.1 (Dec. 2016), pp. 404–415. DOI: 10.1007/s12555-015-0181-1.

[232] Brian Gaudet, Richard Linares, and Roberto Furfaro. "Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations". In: *Acta Astronautica* 171 (June 2020), pp. 1–13. DOI: 10.1016/j.actaastro.2020.02.036.

[233] Brian Gaudet, Richard Linares, and Roberto Furfaro. "Six degree-of-freedom body-fixed hovering over unmapped asteroids via LIDAR altimetry and reinforcement meta-learning". In: *Acta Astronautica* 172 (July 2020), pp. 90–99. DOI: 10.1016/j.actaastro.2020.03.026.

[234] Brian Gaudet, Richard Linares, and Roberto Furfaro. *Seeker based Adaptive Guidance via Reinforcement Meta-Learning Applied to Asteroid Close Proximity Operations*. 2019. arXiv: 1907.06098 [eess.SY].

[235] Lin Cheng et al. "Real-time optimal control for irregular asteroid landings using deep neural networks". In: *Acta Astronautica* 170 (May 2020), pp. 66–79. DOI: 10.1016/j.actaastro.2019.11.039.

[236] Lin Cheng et al. "Fast Generation of Optimal Asteroid Landing Trajectories Using Deep Neural Networks". In: *IEEE Transactions on Aerospace and Electronic Systems* 56.4 (Aug. 2020), pp. 2642–2655. DOI: 10.1109/taes.2019.2952700.

[237] Lin Cheng et al. "Real-Time Optimal Control for Spacecraft Orbit Transfer via Multiscale Deep Neural Networks". In: *IEEE Transactions on Aerospace and Electronic Systems* 55.5 (Oct. 2019), pp. 2436–2450. DOI: 10.1109/taes.2018.2889571.

[238] Harry Holt et al. "Low-Thrust Trajectory Design Using Closed-Loop Feedback-Driven Control Laws and State-Dependent Parameters". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-1694.

[239] Nicholas B. LaFarge et al. "Guidance for Closed-Loop Transfers using Reinforcement Learning with Application to Libration Point Orbits". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-0458.

[240] Daniel Miller and Richard Linares. "Low-Thrust Optimal Control via Reinforcement Learning". In: *29th AAS/AIAA Space Flight Mechanics Meeting*. 2019.

[241] Daniel Miller, Jacob A. Englander, and Richard Linares. "Interplanetary Low-Thrust Design Using Proximal Policy Optimization". In: *AAS/AIAA Astrodynamics Specialist Conference*. American Astronautical Society, 2019.

[242] Behçet Açıkmeşe and Scott R. Ploen. "Convex Programming Approach to Powered Descent Guidance for Mars Landing". In: *Journal of Guidance, Control, and Dynamics* 30.5 (Sept. 2007), pp. 1353–1366. DOI: 10.2514/1.27553.

[243] Ping Lu and Binfeng Pan. "Highly Constrained Optimal Launch Ascent Guidance". In: *Journal of Guidance, Control, and Dynamics* 33.2 (Mar. 2010), pp. 404–414. DOI: 10.2514/1.45632.

[244] D.F. Lawden. *Optimal Trajectories for Space Navigation*. London: Butterworths, 1963.

[245] Sixiong You et al. "Learning-based Optimal Control for Planetary Entry, Powered Descent and Landing Guidance". In: *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-0849.

[246] Sixiong You et al. "Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems". In: *Journal of Guidance, Control, and Dynamics* (2020). Accepted.

[247] Carlos Sánchez-Sánchez and Dario Izzo. "Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems". In: *Journal of Guidance, Control, and Dynamics* 41.5 (May 2018), pp. 1122–1135. DOI: 10.2514/1.g002357.

[248] Paulo Tabuada and Lucas Fraile. "Data-Driven Stabilization Of SISO Feedback Linearizable Systems". In: *59th IEEE Conference on Decision and Control*. IEEE, Dec. 2020, arXiv:2003.14240.

[249] A. Srikantha Phani. "On the necessary and sufficient conditions for the existence of classical normal modes in damped linear dynamic systems". In: *Journal of Sound and Vibration* 264.3 (July 2003), pp. 741–745. DOI: 10.1016/s0022-460x(02)01506-7.

[250] Jean-Claude Hennet. "Discrete Time Constrained Linear Systems". In: *Control and Dynamic Systems*. Elsevier, 1995, pp. 157–213. DOI: 10.1016/s0090-5267(06)80018-6.

[251] Paul Trodden. “A One-Step Approach to Computing a Polytopic Robust Positively Invariant Set”. In: *IEEE Transactions on Automatic Control* 61.12 (Dec. 2016), pp. 4100–4105. DOI: 10.1109/tac.2016.2541300.

Appendix A

RANDOM GENERATION OF A MULTIPLE-DOF OSCILLATOR MPC PROBLEM

This appendix presents how one can generate a random MPC problem instance for a $2n_r$-dimensional ($n_r \in \mathbb{Z}_+$) instance of a multiple-DoF oscillator. To begin, we discuss random generation of the system equations of motion (in other words, the dynamics). In continuous time (time is omitted for notational simplicity) and in its configuration basis, the multiple-DoF oscillator dynamics are written as:

$$M\ddot{r} + C\dot{r} + Kr = Lu, \tag{A.1}$$

where $M \in \mathbb{R}^{n_r \times n_r}$, $M \succ 0$, is the mass matrix, $C \in \mathbb{R}^{n_r \times n_r}$ is the damping matrix, $K \in \mathbb{R}^{n_r \times n_r}$, $K \succeq 0$, is the stiffness matrix, $L \in \mathbb{R}^{n_r \times n_u}$ is an input map, $r \in \mathbb{R}^{n_r}$ is a vector of generalized coordinates and $u \in \mathbb{R}^{n_u}$ is the input. The task is to generate M, C, K and L such that the system is controllable and has poles located in a prescribed region of the complex plane. Assuming Caughey's condition holds such that M, C and K are simultaneously diagonalizable [249], (A.1) can be written in its modal basis:

$$\ddot{\eta} + \Lambda\dot{\eta} + \Omega\eta = \Gamma u, \tag{A.2}$$

where $\eta = T^\top M^{1/2} r \in \mathbb{R}^{n_r}$ are the modal coordinates, $T \in \mathbb{R}^{n_r \times n_r}$ is the modal matrix, $\Lambda = T^\top M^{-1/2} C M^{-1/2} T$, $\Omega = T^\top M^{-1/2} K M^{-1/2} T$ and $\Gamma = T^\top M^{-1/2} L$. The matrices Λ and Ω are diagonal such that each row of (A.2) is a 1-DoF oscillator which contributes two poles

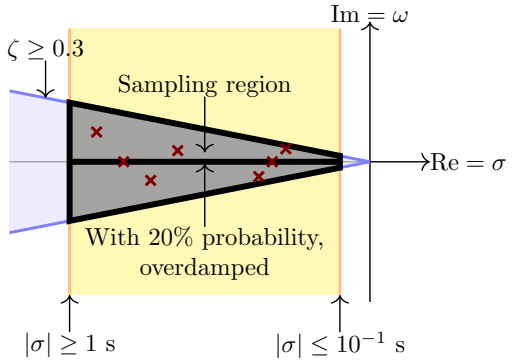


Figure A.1: Visualization of the s-plane with constraints on the randomly generated system poles. The feasible sampling region is highlighted and several sampled poles are shown.

to the overall system:

$$\ddot{\eta}_i + 2\zeta_i \omega_{\mathrm{n},i} \dot{\eta}_i + \omega_{\mathrm{n},i}^2 \eta_i = u_i \quad i = 1, \ldots, n_r, \tag{A.3}$$

where we chose $\Gamma = I_{n_r}$ which ensures that (A.1) is controllable, ζ_i is the damping ratio and $\omega_{\mathrm{n},i}$ is the natural frequency of the i-th mode. Note that this implies $n_u = n_r$. To generate (A.1), it remains to choose M, T, Λ and Ω. We choose a uniform random diagonal $M = \mathrm{diag}(\{m_i \in [0.1, 1]\}_{i=1}^{n_r})$, T as the orthogonal matrix from QR decomposition of a Gaussian random matrix and Λ, Ω from uniformly randomly generating poles in the s-plane such that 1) the damping rate is $\in [1, 10]$ s, and 2) the damping ratio $\zeta \geq 0.3$ with a probability of 0.2 that $\zeta \geq 1$. This set of requirements is illustrated in Figure A.1.

Once generated, the system (A.1) is written in state space form $\dot{x} = Ax + Bu + Ew$:

$$\begin{bmatrix} \dot{r} \\ \ddot{r} \end{bmatrix} = \begin{bmatrix} 0 & I_{n_r} \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} r \\ \dot{r}g \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}L \end{bmatrix} u + \begin{bmatrix} 0 \\ I_{n_r} \end{bmatrix} w, \tag{A.4}$$

where $w \in \mathbb{R}^{n_r}$ is an exogenous disturbance force acting along each generalized coordinate. This system is discretized via zero-order hold with sampling frequency $\omega_{\mathrm{s}} = 10 \max_{\lambda \in \mathrm{spec}(A)} |\lambda|$, i.e. ten times faster than the fastest natural frequency present in the

system [114].

A.1 MPC Problem Instance

We now present how the generate system dynamics (A.4) are used in an MPC problem instance. The control objective is to drive the state $\begin{bmatrix}\dot{r}^\mathsf{T} & \ddot{r}^\mathsf{T}\end{bmatrix}^\mathsf{T} \in \mathbb{R}^{2n_r}$ to the origin.

Following Section 4.9.1, the set Θ over which the explicit MPC solution is to be computed, is chosen to be the smallest robust controlled invariant set for (A.4) using the uncertainty set $\mathcal{W} \triangleq \{w : \|w\|_\infty \leq 10^{-3}\}$ and an LQR controller with a $Q_{\text{lqr}} = 0.1 I_{2n_r}$ state penalty and an $R_{\text{lqr}} = I_{n_r}$ input penalty [250, 251]. For the MPC law, the uncertainty model is changed to be norm-bounded:

$$\mathcal{W}' \triangleq \{w \in \mathbb{R}^{n_r} : \|w\|_2 \leq 0.4 \cdot \frac{10^{-3}\|x\|_2}{\max_{v \in \mathcal{V}(\Theta)} \|v\|_2}\}, \tag{A.5}$$

which is a smaller uncertainty but, importantly, introduces second-order cone constraints into Problem 4.1 [58]. The ad hoc factor of 0.4 is used to reduce uncertainty such that a planning horizon of $N = 3$ is feasible for the robust MPC law, whereas only $N = 1$ is guaranteed by the computation method for Θ [251]. Finally, to make the control problem mixed-integer, the control input is constrained to lie in a non-convex set:

$$\begin{aligned} u \in \mathcal{U} &= \{0\} \cup (\mathcal{U}_{\text{ext}} \setminus \mathcal{U}_{\text{int}}), \\ \mathcal{U}_{\text{ext}} &\triangleq \{u \in \mathbb{R}^{n_r} : -u_{\max} \leq u \leq u_{\max}\}, \\ \mathcal{U}_{\text{int}} &\triangleq \{u \in \mathbb{R}^{n_r} : -10^{-3} u_{\max} \leq u \leq 10^{-3} u_{\max}\}, \end{aligned} \tag{A.6}$$

where $u_{\max,i} = \max_{v \in \mathcal{V}(\Theta)} |e_i^\mathsf{T} K_{\text{lqr}} v|$ is the largest input magnitude required by the LQR controller along the i-th generalized coordinate. Note that since $\mathcal{U}_{\text{ext}}$ and $\mathcal{U}_{\text{int}}$ are origin-centered hyperrectangles, one can write $\mathcal{U} = \cup_{i=1}^{2n_r+1} \mathcal{U}_i$ where $\mathcal{U}_i$ are convex polytopes and $\mathcal{U}_1 = \{0\}$ (by the same partitioning method as illustrated in Figure 4.7). There are then $N(2n_r + 1)$ degrees of freedom to choose which convex subsets of $\mathcal{U}$ the control inputs are

to be in, which makes for a commutation vector of dimension $m = N(2n_r + 1)$. The robust MPC law is then:

$$\min_{x_k, u_k} \sum_{k=0}^{N-1} \hat{u}_k^\mathsf{T} R_{\text{lqr}} \hat{u}_k + \hat{x}_{k+1}^\mathsf{T} Q_{\text{lqr}} \hat{x}_{k+1} \tag{A.7a}$$

$$\text{s.t. } x_0 = \theta, \tag{A.7b}$$

$$x_{k+1} = Ax_k + Bu_k + Ew_k, \ k = 0 : N-1, \tag{A.7c}$$

$$x_k \in \Theta \ \forall w_j \in \mathcal{W}', \ j = 0 : k-1, \ k = 1 : N, \tag{A.7d}$$

$$u_k \in \mathcal{U}_i \text{ for some } i \in \{1 : 2n_r + 1\}, \ k = 0 : N-1. \tag{A.7e}$$

In Problem A.7, $\hat{u}_k$ and $\hat{x}_k$ are the scaled input and state such that they attain plus or minus unity at the boundary of $\mathcal{U}_{\text{ext}}$ and Θ respectively (the idea is the same as for (3.30) in Chapter 3). We transform Problem A.7 into the form Problem 4.1 via Hölder's inequality via the method developed in Chapter 3 [58]. Note that the parameter vector dimension is $p = 2n_r$, therefore Problem A.7 is constrained to even parameter dimensions.

VITA

Danylo Malyuta received a B.Sc. in Mechanical Engineering from EPFL and an M.Sc. in Robotics, Systems and Control from ETH Zürich. He is currently a Ph.D. candidate in the Autonomous Controls Lab at the Department of Aeronautics and Astronautics of the University of Washington. His research is primarily focused on computationally efficient optimization-based control of dynamical systems. Danylo has held internship positions at the NASA Jet Propulsion Laboratory, NASA Johnson Space Center, and Amazon Prime Air.