Real-Time Trajectory Optimization for High-Performance Guidance & Control

Abhinav Girish Kamath

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

Doctor of Philosophy

University of Washington

2025

Reading Committee:
Behçet Açıkmeşe, Chair
Mehran Mesbahi
Michael Szmuk

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

Abstract

Real-Time Trajectory Optimization for High-Performance Guidance & Control

Abhinav Girish Kamath

Chair of the Supervisory Committee:
Behçet Açıkmeşe
Aeronautics & Astronautics

Autonomous systems of today rely on trajectory planning to achieve complex tasks. With the increasing capabilities of such systems, there is a need for a framework that not only allows for accurate modeling of these tasks, but also enables real-time generation of feasible trajectories to achieve them. This dissertation presents trajectory generation methods, using gradient-based optimization and set-based dynamic programming, for a large class of optimal, robust, and resilient control problems. These methods are intended for adoption onboard agile autonomous systems—such as reusable rockets—that mandate high-performance guidance & control.

This dissertation is organized as follows: Chapter 1 provides an overview of the dissertation, including literature review, motivation, research objectives, and contributions. Chapter 2 delves into the design of the sequential conic optimization (SeCO) framework for trajectory generation and discusses practical considerations for real-time convex optimization, and particularly, quadratic cone programming. Chapter 3 introduces a novel preconditioning procedure for online quadratic cone programming. Chapter 4 develops a set-based dynamic programming framework for optimal, robust, and resilient predictive control. Chapter 5 delves into temporal techniques for guidance & control. Chapter 6 discusses three relevant rocket landing guidance & control applications, the first two using SeCO and the third using set-based dynamic programming. Chapter 7 presents SCvxGEN, an automatic code generation software tool for general-purpose trajectory optimization via successive convexification. Finally, Chapter 8 concludes the dissertation and suggests potential avenues for future work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I owe a debt of gratitude to all those who stood by me throughout my doctoral journey.

First and foremost, I'd like to thank my advisor, Professor Behçet Açıkmeşe. Behçet, you've given me opportunities that I could only dream of, provided me with all the resources I needed to succeed, and helped me every step of the way, without thinking twice. I will cherish all the discussions on rocket landing we've had in your office and at NASA. To get that opportunity—especially after having followed your work for years—was nothing short of a dream come true. Thank you for entrusting me with the rocket landing guidance project at NASA and for all your support and guidance.

Mehran, thank you for welcoming me to UW and always keeping your door open for me. Thank you also for all your words of wisdom. I'm glad I get to continue seeing you at work.

Thank you to Sasha for teaching me optimization and also agreeing to be a part of my committee (and attending my defense at 3 AM from Japan). I'd also like to thank Professor Karen Leung and Professor Ed Habtour for agreeing to be a part of my committee for my qualifying and general exams.

Simply put, I would not be here if not for Danylo. Thank you, Danylo, for taking a chance on me, helping me get into UW, and supporting me throughout my journey here; for being a trusted mentor, friend, and a role model. Thank you for picking me up from the airport when I first moved to Seattle and for being a constant source of support ever since. Your research and work ethic alike inspired me even before I started at UW, and they continue to inspire me today.

Purna, I owe much of my research to you. You are one of the clearest thinkers I've had the pleasure to meet and get to know, and getting to work with you has truly been an honor. I'd

like to say I'm sorry for not letting you off the hook even after you graduated (by following you to MERL as an intern), but I'd be lying. Looking back, it makes me happy to see just how much work we did together. To this day, I try to live by the research principles that you so naturally embody. Thank you for everything you've done for me. Thank you for your friendship and for being there for me throughout. And most of all, thanks to you and Joen for introducing me to real sushi.

Skye, talking to you during the initial stages of my Ph.D. and seeing your passion and enthusiasm for spaceflight gave me some much-needed fuel to embark on this journey. Putting together presentations for ONR, collaborating on our many papers (including the fun last-minute sprints), and all the SciTech dinners with Behçet and Purna were some of the most fun times during my time at UW.

Govind, you are simply one of the highest-agency individuals I have had the pleasure of getting to know. It is inspiring to see you put your mind to something and go after it without looking back. I'm glad we got to work together on many projects and also share many fun moments, both in research and outside of research. Importantly, I'm fortunate that I get to say I have an optimization solver specialist on speed-dial.

Samet, your ingenuity and your ability to dig deep into the problem at hand are impressive. I'm really glad we got to collaborate on our paper towards the end—working with you on it was one of the best writing experiences I've had. I look forward to our next Noon Cafe rendezvous. Taewan, bouncing ideas off you and collaborating with you on various projects have been truly rewarding experiences for me. Sam, getting to work with you on rocket landing at JSC was an amazing experience—Houston was fun.

Thank you to the rest of my fellow ACLers: Dayou, Aman, Jason, Carlos, Natalia, Chris, Justin, Oliver, Kazu, Avi, Fabio, Ben, Sarah, Dan, and all other members, past and present. Thank you also to Carlo—even though I was unable to fly to New Zealand to work with you, I'm glad I got to chat with you on multiple occasions.

Thank you to Yue Yu for always being willing to hop on a call to teach me about optimization and technical writing. The fundamental work that you and Purna did forms the backbone of many of the algorithms and applications in this dissertation. I truly value our collaborations.

Thank you, Behçet, John, and Gavin, for giving me the opportunity to work on the SPLICE program at the NASA Johnson Space Center. John and Gavin, thank you for your continued support and mentorship—I'm lucky I got to work alongside you and learn so much from you. Javier, taking our work from design to onboard implementation would not have been possible without all your hard work and effort; thank you for that and your support in general—I cannot wait to witness the upcoming closed-loop flight tests. Thanks also to Adrian and all the people I had the good fortune of interacting and working with there.

Thank you, Avishai, for giving me the opportunity to work on the precision landing problem at MERL. Thanks also for graciously agreeing to be a part of my defense committee. You and Abraham opened my eyes to the world of set-based control—I'm fortunate to have gotten the opportunity to work with you and publish research. Thanks for bearing with my long tech reports. Abraham, in addition to your foundational work, the software you have released is a gift to the community—a major part of this dissertation would not have been possible without it. Thanks to Purna, again, for all our brainstorming sessions at MERL—I'm glad I got to be your unofficial intern, even if it was only for a couple of weeks.

Thank you, Taylor, for passing down your wisdom to me and taking me under your wing when I first started my internship at Amazon Prime Air. You were always willing to answer my questions on the work you did during your time at UW—your tips and guidance were instrumental in realizing a lot of the work in this dissertation. Your sheer competence is quite something to witness at work—I'm fortunate to be on the same team as you now. Thanks also to Umut, Raghu, Marco, Andrei, Tzanis, Aryslan, Ray, Sankalp, Sujit, and Daniel for making my experience at Prime Air, both as an intern and as a full-timer, truly amazing.

Miki, where do I even begin? I consider myself fortunate to have gotten to attend your lectures on applied guidance & controls (or to me, GNC 101). Lucky are the students who get to learn from the intuition machine™ himself. The effort you put into everything you do is unparalleled. What you have achieved at Prime Air is truly remarkable and awe-inspiring. Thank you for always looking out for me—from all our long walks through Ravenna Park talking about life, to you and Morgan always making sure I was well-fed, to you mentoring me at work. I'm lucky that I get to call you my friend, teacher, mentor, and now, manager—you really are the *world's best boss*.

I would be remiss in my duty if I did not acknowledge the role that Meco and Mo played in my Ph.D.—the last few papers I wrote would not have been completed if not for their thorough code reviews and proofreading.

I would like to acknowledge the role that my time at UC Davis played in my doctoral journey: Thank you to Professor Robinson for molding me into the person I am today. Thanks to Professor Assadian and Professor Moore for getting me excited about controls and dynamics. Thanks to Josh and all of HRVIP for the camaraderie.

Thanks to all the professors and staff at UW who have enabled me to reach this point.

Finally, thank you to Nimmakka, Naresh Uncle, Dinesh Uncle, Teresa Aunty, Annu Maam, Ganesh Rao, Arjun, Anish, Shujju, Mom, Dad, Archana, and all my well-wishers, friends, and family. I certainly wouldn't be here without all your love and support. My gratitude extends beyond what I can write here; I hope to find the right words when I see you in person.

DEDICATION

To my family...

Chapter 1
INTRODUCTION

This chapter provides chapter-wise introductions to the research directions pursued in this dissertation, including the requisite background details and core contributions.

1.1 Background & Contributions

1.1.1 **Chapter 2**: *Sequential Conic Optimization for Trajectory Generation*

Sequential convex programming (SCP) has recently emerged as a powerful tool for solving nonconvex optimal control problems [134]. The method involves iteratively solving a sequence of convex subproblems that approximate the original nonconvex problem. The convergence of SCP algorithms is guaranteed under mild assumptions, and the method has been shown to be effective in practice for a wide range of problems. The primary advantage of SCP is that it only requires first-order information (i.e., gradients and not Hessians), which is in turn beneficial for real-time applications.

SCP falls under the class of trust region methods [155]. Within SCP, there exist both hard trust region [137, 36, 127] and soft trust region [174, 208, 176] methods in the literature. A popular approach in SCP-based trajectory optimization is to impose a soft trust region by augmenting the objective function with a quadratic penalty term, and to use the penalized trust region (PTR) algorithm, which, as the name suggests, penalizes the trust region radius in lieu of constraining it and adopting an outer-loop update rule (which is the case in hard trust region methods).

The PTR algorithm has been shown to work very well in practice, and has been employed to successfully solve a wide range of challenging nonconvex optimal control problems in the context of real-time quadrotor path-planning [206, 205, 204, 219, 65], aircraft approach and

landing [109, 110], spacecraft rendezvous and docking [132, 130, 129, 49], hypersonic entry [143, 146], and rocket landing [176, 174, 208, 108, 103, 102, 42, 51, 65]. A comprehensive survey of the application of SCP methods to optimal control problems that arise in the aerospace domain can be found in [135].

The standard PTR algorithm handles nonconvex constraints via linearization and adding slack variables to them (within each convex subproblem), and by penalizing these slack variables. The penalty function is typically chosen to be the 1-norm in order to promote sparsity in the slack variable solutions [208]. At and around the time of its conception, the convergence properties of PTR remained an open question [208]. However, recently, it was shown that in [65] that the PTR algorithm with a 1-norm penalization of all the nonconvex constraints is practically identical to the prox-linear method for convex composite minimization [60, 61], which is guaranteed to converge to a stationary point of the original nonconvex problem.

The PTR algorithm originally involved a discretization procedure that required matrix inversions. The resulting convex subproblems were solved using interior-point methods (IPMs) [174, 176, 208]. For the 6-DoF powered-descent guidance problem—a motivating application that mandates real-time performance—solve-times were on the order of one second on desktop computers [208]. One such state-of-the-art 6-DoF powered-descent guidance algorithm is the dual quaternion-based powered-descent guidance algorithm (DQG), originally developed in [176], which was chosen as the candidate powered-descent guidance algorithm for NASA's Safe and Precise Landing – Integrated Capabilities Evolution (SPLICE) project, and has been open-loop flight-tested on the Blue Origin New Shepard suborbital rocket onboard the descent and landing computer (DLC) [185, 202, 75].

The implementation of the algorithm for these flight tests made use of a customized version of a second-order IPM-based solver called BSOCP, along with a parser interface called CPRS [63, 62]. As noted in [202], that implementation of DQG onboard the DLC takes over 11 seconds to execute (with an optimization horizon length of 20), and close to 6 seconds to execute (with an optimization horizon length of 10), making it prohibitively slow in terms of

meeting NASA's guidance update-rate requirements for PL&HA in its current state. Further, the customized solver leads to a very large footprint source code—around 600,000 lines of C code. Additionally, the convex subproblems involve positive semidefinite objective function Hessian matrices [49, Equation 26], and are hence not strongly convex.

In light of these challenges and the recent emergence of effective first-order convex optimization methods (see §1.1.2), in this work, we seek to devise a sequential convex programming algorithm—for the optimal control problem at hand—that possesses the following features:

1. The entire sequential convex programming algorithm is *matrix-inverse-free*: this includes both the discretization and convex optimization layers.
2. The resulting convex subproblems are amenable to efficient solution via *first-order methods* (§1.1.2):
   (a) The convex subproblems are *strongly convex*.
   (b) *All* the path constraints are satisfied via *closed-form projections*, the benefits of which are many-fold: exact satisfaction of the constraints at each solver iteration, reduction of the search-space, and reduction in the number of dual variables (and in turn, memory usage).
   (c) The subproblem structure lends itself to *solver customization*.
3. The algorithm is amenable to *real-time implementations* onboard computationally constrained flight hardware (with moderate-accuracy subproblem solves).
4. The algorithm lends itself to a *low-footprint* codebase that is easy to verify and validate.

In order to tackle the challenges of execution speed and code footprint, first-order optimization algorithms can be used instead of (second-order) IPMs. First-order algorithms typically rely on simple linear algebra operations like matrix-vector multiplications and computation of vector norms at each iteration. Unlike second-order methods, they can avoid factorization of larger matrices and can be warm-started easily. In addition to these benefits, the recently introduced first-order algorithm, the proportional-integral projected gradient

(PIPG) method [237, 239], is capable of exploiting the structure of trajectory optimization problems to completely avoid operations on large sparse matrices. As a result, PIPG is readily suitable for onboard, resource-constrained applications. Further, if the problem being solved is strongly convex, PIPG demonstrates faster convergence properties [239].

Recent work on the design and implementation of a customized PIPG solver for the 3-DoF LCvx algorithm demonstrated the capabilities of PIPG in terms of computational efficiency, code footprint, and its viability for easy verification and validation [64]. Moreover, features such as warm-starting and extrapolation [238] for boosting the practical convergence rate have further enhanced PIPG's capabilities as a conic optimization solver for use within sequential convex programming (SCP) algorithms [134]. Further, PIPG is amenable to *customization*, i.e., it directly uses the optimal control problem data (the dynamics matrices, constraint functions, etc) and only incorporates dense linear algebra routines with low-dimensional matrices, i.e., it precludes the need for assembling sparse matrices, performing sparse linear algebra operations, and transforming the problem to a canonical form, all of which are typically required by convex optimization solvers.

We take a bottom-up approach, and devise a high-level sequential convex programming algorithm that meshes well with an existing low-level first-order convex optimization algorithm, the proportional-integral projected gradient method (PIPG). In other words, we develop a specialized SCP algorithm to harness the approach taken in PIPG to solve conic optimization problems, resulting in a framework that possesses all the features mentioned above—we call this *sequential conic optimization* (SeCO) [103, 102, 100].

1.1.2 **Chapter 3**: *Optimal Preconditioning for Conic Optimization*

The class of QCPs given by the template problem, Problem 3.1, appears often in *online* applications wherein optimization problems need to be solved in real-time, in a sequential fashion, with the problem size and structure remaining the same, but with the problem data changing between successive calls to the solver. Such a mode of operation (with the objective function being strongly convex) is observed most notably in model predictive control (MPC) [237] and

nonconvex trajectory optimization using sequential convex programming (SCP)—specifically a variant of SCP called sequential conic optimization (SeCO) [102, 103]. Further, there exist online applications that involve the solution to a single convex problem (with a strongly convex objective function) [64], which would fit well into such a framework if used in conjunction with a line search [2]. We note that several online applications, wherein the objective function is not necessarily strongly convex, exist as well [231, 158, 209, 122, 208, 176, 240, 65], but in this work, we restrict our focus to applications with strongly convex objective functions. While the theoretical results in this work only hold for problems with strongly convex objective functions, problems with ill-conditioned non-strongly convex objective functions might also benefit from the proposed preconditioning procedure in practice.

First-order conic optimization solvers are attractive for: (i) real-time applications, since they are (can be) fast, (ii) implementation onboard resource-constrained embedded systems, since they only require a small code footprint, with low memory requirements in terms of both allocations and the compiled binary executable, (iii) easy *customization*, which refers to exploitation of the sparsity structure of the optimization problem being solved—in a manner that enables low-dimensional matrix-vector multiplications and other dense linear algebra operations with devectorized variables—thus eliminating the need for sparse linear algebra operations entirely [102], (iv) easy verification and validation (V&V), owing to the fact that they only rely on simple linear algebra operations, (v) usage within sequential convex programming algorithms for nonconvex optimization, given their amenability to warm-starting, and (vi) large-scale problems, since their performance scales well with problem size [47, 18, 157, 201, 237, 239, 238]. The performance of first-order methods, however, is highly dependent on conditioning of the problem data [81, 201]—more so than solvers based on interior point methods (IPMs) that utilize second-order (Hessian) information.

Preconditioning is an operation that transforms a given matrix into another matrix of the same size but with a smaller condition number—it is a heuristic often employed to improve the performance of iterative algorithms. Simply put, applying the algorithm in question to a transformed matrix with a smaller condition number typically leads to better performance

in practice (fewer iterations to convergence, for example) [232]. Several preconditioning techniques exist in the literature for various iterative numerical methods [22].

In the context of first-order methods, diagonal preconditioners are popularly used [162, 80]. The computation of an exact diagonal preconditioner can be posed as a (computationally expensive-to-solve) semidefinite program [82, 38]. However, in practice, matrix equilibration [82, 194]—which refers to transforming the matrix in question such that the columns have equal norms and the rows have equal norms—and specifically, Ruiz equilibration [183], is a popularly used *iterative* algorithm for approximate preconditioning in first-order methods that is observed to work well, while also being computationally efficient [157, 201]. Recently, a QR factorization-based constraint matrix preconditioner was also proposed [52], which was shown to accelerate convergence of first-order methods. However, it requires *matrix factorizations* and turns sparse problems dense, thus increasing the per-iteration cost of the solver and limiting customizability. Further, it only applies to QCPs in which $\mathbb{L}$ (in Equation 3.1b) only contains the zero cone. Note that all the aforementioned approaches to preconditioning are either iterative or based on (explicit) matrix factorizations.

In this work, we propose a three-step procedure for preconditioning conic optimization problems that is amenable to a (mostly) analytical, factorization-free, and customizable implementation:

1. Minimizing the condition number of the Hessian of the objective function using hypersphere preconditioning: this step is optimal, in that the condition number of the resulting objective function matrix is unity. This was first introduced in [102], but it was only applied to problems where $\mathbb{L}$ only contains the zero cone.
2. Performing a (block) row-normalization of the constraint matrix, which is a simple, customization-friendly heuristic to make the constraint matrix better conditioned: although not guaranteed to reduce the condition number of the constraint matrix, this procedure has been observed to work well in practice. When used in tandem with a primal-dual conic optimization algorithm, it also provides the added benefit of scaling the

dual variables favorably.

3. Minimizing the condition number of the resulting Karush-Kuhn-Tucker (KKT) matrix by optimally scaling the objective function: in [102], the objective function scaling factor was manually tuned to obtain good performance, whereas in this work, we obtain a closed-form solution for the objective function scaling factor that minimizes the condition number of the KKT matrix—this step is optimal in that sense. A customizable numerical algorithm, the shifted power iteration method [234], can be used to efficiently estimate this value—this is the only iterative component in the entire preconditioning process. Note that this step is important, as the condition number of the KKT matrix is sensitive to the scaling of the objective function—a "bad" scaling can lead to poor convergence behavior, even if the first two steps are followed.

We refer to this three-step procedure as *hypersphere preconditioning* [101]. While the end goal of preconditioning is to minimize the condition number of the KKT matrix of Problem 3.1 itself, we propose the aforementioned three-step procedure to approximately achieve that goal while ensuring that the structure of the preconditioned problem is preserved, thus enabling customization and facilitating online optimization.

1.1.3 **Chapter 4**: *Set-based Dynamic Programming for Predictive Control*

Modern autonomous systems require advanced real-time-capable control methods. In this work, motivated by the autonomous precision landing problem, we develop a computationally tractable and real-time-capable approach to optimal, robust, and resilient predictive control—for a class of systems—within a unified control architecture. Our approach is set-theoretic: we present general results for closed-loop optimal control, for control under modeled state and control uncertainty, and for control in the presence of unmodeled uncertainty. We develop a computational framework for closed-loop control via dynamic programming over a controllable tube, i.e., a time-indexed sequence of controllable sets [225]. For computational tractability, we leverage constrained zonotopes, a representation of compact convex

polytopes that admits efficient and scalable set operations [192].

Robust open-loop control with both state and control uncertainty can be prohibitively conservative, motivating the need for a closed-loop control framework [37]. Global optimality in a closed-loop setting is naturally characterized by dynamic programming [19]. Standard approaches to dynamic programming, however, require discretization of the state space and suffer from the curse of dimensionality, making them intractable for high-dimensional systems [25]. Recently, it was shown that for a class of nonlinear optimal control problems, dynamic programming can be made tractable by lossless convexification, a state-augmentation, and employing controllable tubes [225], which we adopt in this work. The tractability of dynamic programming, consequently, relies on the tractability of controllable tube generation, for which we use constrained zonotopes [192].

The main contributions of this chapter are [105, 106]:

1. Robust control accounting for both control uncertainty and time-varying state uncertainty.
2. A general set-theoretic result for computing the reachable set from the current state (i.e., the instantaneous reachable set).
3. Set-based maximal decision-deferral, i.e., maintaining reachability of multiple targets for as long as possible, which enables large diverts.

All these contributions are presented with computationally tractable implementations, as demonstrated by means of a comprehensive autonomous precision landing case study.

Optimal predictive control for constrained systems is typically achieved by either: (i) model predictive control (MPC) and its variants [76, 170, 142, 131], and (ii) dynamic programming (DP) and its variants [19, 20, 25, 165, 166]. MPC involves the generation of an open-loop control sequence over a time-horizon, only applying the first control input, and repeating the process at each step [76, 37, 233, 4, 131]. Despite providing a feedback action to close the loop, the nominal prediction across the time-horizon itself is feedback-agnostic, i.e., it is open-loop, and robust variants tend to be conservative in terms of handling disturbances

[37]. DP, in contrast, yields globally optimal closed-loop policies, but standard techniques rely on state-space discretization and storage of value function evaluations for each state across the entire time-horizon, the number of which grows exponentially as a function of the state dimension, thus limiting its use to small-dimensional systems [25]. Owing to the closed-loop nature of DP, robust control is less conservative than open-loop approaches.

Resilient control is an emerging field of control [182, 48] that deals with keeping the system under consideration safe in the face of unmodeled uncertainties, or, in other words, *unknown unknowns*, such as disruptions and faults. Landing guidance algorithms with built-in divert capabilities serve as a good example for this [6, 1, 88, 90, 43, 199, 121]. Reachability-steering is an approach to resilient control that has been applied in the context of precision landing [212, 211]. Another framework for open-loop resilient control is that of *deferred-decision trajectory optimization* (DDTO) [67, 90, 68], which ensures (constrained) reachability to a collection of target terminal states for as long as possible.

Set-based methods in control [34, 37] have recently emerged as a popular approach to deterministic optimal control [71, 199] and robust optimal control [121, 228, 229, 225]. Specifically, [71, 199, 121] propose open-loop approaches that involve the construction of a controllable set for the autonomous precision landing problem, and [226, 83, 227, 229, 224, 225] propose closed-loop approaches that involve the construction of a controllable *tube*, i.e., a sequence of controllable sets, applied to spacecraft rendezvous trajectory optimization [226, 227, 228, 224, 229]. Most set-theoretic approaches leverage polytopes, which are often limited to low dimensions due to computational challenges [34]. However, the introduction of *constrained zonotopes* (CZs) has enabled computationally tractable set operations in high dimensions [192], which are largely beneficial in the field of control [192, 228, 229, 225].

1.1.4 **Chapter 5**: *Temporal Techniques for Guidance & Control*

A conditional constraint is one that is only activated if a certain condition is satisfied, best described by an *if-then* statement. One important use-case of conditional constraints is in modern planetary rocket landing missions that require accurate scans of the terrain during

descent, and the ability to avoid hazardous regions. In such missions, the vehicle must maintain a line-of-sight with the target landing site throughout the descent phase, while simultaneously satisfying several other constraints, thus making for a compound constraint that couples the translation and attitude states of the vehicle [177, 176].

In their native form, such constraints happen to be integer constraints, and they require mixed-integer programming solvers to handle them. However, in the context of trajectory optimization, it is desirable to handle such constraints within a continuous optimization framework, to allow for the use of gradient-based solution methods.

State-triggered constraints (STCs) are a powerful tool for modeling such conditional constraints in a direct trajectory optimization framework [207, 208, 176]. They provide an equivalent continuous formulation of inherently mixed-integer constraints, thus allowing for their imposition within continuous optimization frameworks. They allow for *triggering* a certain constraint function based on the satisfaction of a certain condition, given by a trigger function. This idea has also been extended to *compound* STCs, which allow for the triggering of multiple constraints based on the satisfaction of multiple trigger conditions, either with the AND or OR logic [207, 204]. Recently, a homotopy-based approach to handling conditional constraints was also proposed [130].

STCs are general, in that they are *automatic*: once imposed, the optimizer implicitly determines when to trigger the constraints active. However, owing to their formulation, STCs are nonconvex by default, and in their standard form, they can violate the linear independence constraint qualification (LICQ) [96] (they lead to zero rows in the constraint matrix when the trigger function is not satisfied). Further, triggering is not exact in time—it is prone to intersample violations.

The generality of STCs is beneficial when the trigger function is non-monotonic, for example in the case where an angle-of-attack constraint is to be triggered by a velocity-based trigger [208]. That said, we seek to develop a specialized form of compound STCs with the AND logic—in the case wherein the trigger functions are monotonic in a neighborhood around the trigger—that satisfy the following properties:

1. The entire compound STC is *convex* if the trigger and constraint conditions are individually convex.
   (a) Consequently, the compound STC satisfies LICQ if the trigger and constraint conditions satisfy LICQ.
2. Triggering is *exact* in time, i.e., it does not suffer from intersample violations.

We formulate a specialized form of compound STCs, called *single-crossing* compound STCs, that satisfy the above properties. To achieve this, we first introduce the notion of *time-interval dilation*. Further, we demonstrate the efficacy of single-crossing STCs, together with a hybrid control parameterization and discretization technique, in a multi-phase trajectory optimization setting—with free-final-time and free-phase-transition-time capabilities—via the SeCO framework (without introducing any mixed-integer constraints) [103].

Another distinct contribution of this chapter is a general set-theoretic result for closed-loop free-final-time optimal control [105, 106].

1.1.5 **Chapter 6**: *Rocket Landing Guidance & Control*

With robotic and human missions to the Moon and Mars on the horizon, there has been an increased interest in guidance, navigation, and control (GNC) technologies for precision landing [195, 53, 54, 161, 154, 153]. Precision landing and hazard avoidance (PL&HA) have been deemed high-priority capabilities by NASA to facilitate missions of exploration to celestial bodies in the solar system [44]. Critical to achieving PL&HA is powered-descent guidance (PDG), which refers to the generation of a feedforward control profile and the corresponding reference state trajectories for powered-descent and landing.

Historically, missions to the Moon, i.e., the Apollo program, and Mars, i.e., the Mars Science Laboratory (MSL) and Mars 2020 missions, made use of polynomial guidance for the powered-descent and landing phase [111, 187, 46]. While these missions saw incredible success, the resulting descent and landing trajectories were not propellant optimal. Further, the landing dispersion ellipses for the Apollo missions were prohibitively large (on the order

of kilometers) [167] in terms of enabling precision landing, which requires safely landing spacecraft within 100 meters from the target landing site [44].

Rocket landing guidance can be considered to be the generalization of powered-descent guidance (PDG) to include the unpowered phase(s) of flight. Precision landing techniques for orbital rockets harness convex optimization for real-time trajectory generation [2, 32]. One such method, known as *lossless convexification*, was the first convex optimization-based algorithm to compute a rocket landing guidance trajectory for a mid-flight large divert maneuver onboard the vehicle. Sequential convex programming (SCP) techniques for rocket landing have recently emerged as a way to handle more generalized nonconvexities in the dynamics, state constraints, environmental constraints, and path constraints ([137]). Such SCP algorithms have been demonstrated to successfully solve a wide range of 6-DoF rocket landing problems ([207, 176, 208]).

The control problem of autonomous precision landing is particularly challenging owing to payload capacity coming at a premium, estimation and control errors due to uncertainty in navigation and actuation, and the lack of safety due to incomplete or outdated information about the landing zone [200, 32, 44, 135]. At the same time, the overarching goal of autonomous precision landing missions is to maximize the payload capacity by minimizing propellant consumption (optimal control), while remaining robust to navigation and actuation uncertainty (robust control), and maintaining safety in the presence of unforeseen hazards (resilient control). These competing requirements naturally lead to the need for a unified control architecture, which can be realized using set-based dynamic programming [225, 105, 106].

Multi-Phase Rocket Landing Guidance

A traditional approach to entry, descent, and landing guidance would be to compute the trajectories of different phases of the entry, descent, and landing sequences independently, and to then stitch the solutions together a posteriori. However, such an approach would pose the following challenges: (1) the composite trajectory might not be feasible, (2) the

trajectory could be severely suboptimal even if it is feasible, and (3) posing and solving multiple separate optimization problems would not be amenable to real-time computation.

In the optimal control literature, pseudospectral methods have been developed in the last decade for solving multi-phase trajectory optimization problems, particularly in aerospace applications. These methods typically parameterize the state time-history with Chebyshev and Legendre polynomials, operate on nonuniform time-grids with node points that are roots of these polynomials, and use linking conditions to tie together different phases of flight. A multi-phase Radau pseudospectral method was introduced in [77] for ascent and powered-descent guidance. The SPARTAN software package described in [186] has been demonstrated to solve multi-phase problems arising in space applications. Further, relevant details showcasing pseudospectral approaches are provided in [98, 128, 243]. Despite their ability to solve multi-phase problems, pseudospectral methods are typically untenable for real-time implementations [133].

In this work, we present a framework to formulate and solve a multi-phase rocket landing guidance problem in a unified manner [103]. The guidance algorithm has the ability to perform mid-burn engine switching after ignition. The primary features of this framework are as follows: (1) time-interval dilation, which allows for nonuniform time-grids and free-phase-transition-time; (2) a new formulation of compound state-triggered constraints—called *single-crossing* compound state-triggered constraints—that is convex, provided the trigger and constraint conditions are convex; (3) a new approach to handling artificial infeasibility, i.e., *virtual state*, that preserves the shapes of the constraint sets and does not alter the dynamics manifold; and (4) PIPG (proportional-integral projected gradient), a high-performance first-order solver that effectively exploits the structure of trajectory optimization problems, making it well-suited for embedded applications ([237]). Further, we use an inverse-free exact discretization method to generate dynamically feasible solutions. Despite the layers of apparent complexity described, the method that we present is implemented via a low-footprint codebase that is easy to verify and validate.

Dual Quaternion-based 6-DoF Powered-Descent Guidance

The advent of convex optimization methods at the turn of the century, along with pivotal results on the lossless convexification (LCvx) of certain nonconvex constraints, led to the invention of a real-time-capable PDG algorithm with strong guarantees [2, 3, 5]. This algorithm was successfully flight-tested onboard a terrestrial rocket-powered landing testbed, demonstrating the capability of generating propellant-optimal large divert trajectories in real-time [1, 190]. The problem formulation incorporated a point-mass vehicle model and three degree-of-freedom (3-DoF) translation dynamics. While it was demonstrated that such a guidance algorithm was sufficient to successfully guide and land an inherently 6-DoF vehicle (by using the thrust vector profile as a surrogate for the reference attitude of the vehicle), the algorithm itself was incapable of handling coupled translation and attitude constraints.

Technologies such as terrain relative navigation (TRN) and hazard detection and avoidance (HDA) require constraining the flight envelope in a manner that couples the translation and attitude states. Recent work on 6-DoF powered-descent guidance addresses this problem by formulating state-triggered constraints and encoding them in a continuous optimization framework [207, 177, 204, 208, 176]. These are powerful algorithms capable of handling difficult constraints, and have been shown to be amenable to real-time implementation. A real-time implementation of a nonconvex planar landing guidance algorithm has also been proposed [174].

In this work, we demonstrate a real-time-capable implementation of SeCO for solving the nonconvex optimal control problem in DQG [102, 58, 100]. We describe, in detail, the problem formulation, optimization algorithms, solver customization, and then provide numerical results, including offline benchmarks and hardware-in-the-loop results. Many of the following fundamental mathematical definitions and formalisms can be found in [176, 178], and [103], and are presented here for the sake of completeness. The operations pertaining to quaternion and dual quaternion algebra are documented in the Appendix.

Set-based Control for Autonomous Precision Landing

While early work on autonomous precision landing dealt with deterministic optimal control [2, 5], there has been recent work on stochastic guidance and feedback control for landing as well [179, 180, 181]. More recently, [199] proposed a deterministic open-loop control framework for lunar landing with divert capabilities, which was later extended to account for navigational uncertainty in [121]—these papers made use of a convex-optimization-based simplicial "facet-pushing" algorithm to construct the open-loop controllable set, first presented in [71]. The case study we present herein serves as a closed-loop extension to the work on autonomous precision landing with the proposed novel contributions: closed-loop free-final-time optimal control, closed-loop control with robustness to both actuation (control) uncertainties and time-varying navigation (state) uncertainties, and instantaneous-reachable-set- and maximal decision-deferral-based resilient control [105, 106].

1.1.6 **Chapter 7**: SCvxGEN*: Successive Convexification with Automatic Code Generation*

Trajectory optimization refers to the computation of an optimal control profile that generates a dynamically feasible state trajectory, subject to constraints, while optimizing a chosen metric [134]. In motion planning and guidance and control (G&C) applications that rely on onboard trajectory optimization, the optimized control profile serves as a feedforward control input sequence and the corresponding state trajectory serves as the reference trajectory to be tracked [191, 135]. Trajectory optimization also plays a major role in offline mission design [171, 28, 135].

Methods to solve optimal control problems can be classified into indirect methods, which take the *optimize-then-discretize* approach, and direct methods, which take the *discretize-then-optimize* approach. Indirect methods rely on setting up a two-point boundary value problem based on the necessary conditions for optimality that stem from Pontryagin's maximum principle, and solving it to obtain a high-accuracy solution [27]. However, indirect methods require a priori specification of the time-intervals over which constraints are active,

are highly sensitive to initial guesses, and suffer from poor convergence behavior and slow solve-times [135].

Direct methods, on the other hand, rely on parameterizing either the control profile only (as is the case with shooting methods) or both the control profile and the state trajectory (as is the case with collocation methods), by a finite number of variables, to assemble a parameter optimization problem, which is then solved numerically [27, 171]. Owing to the state parameterization, collocation methods typically require a large number of discretization points, which we refer to as *nodes* or *node points*, combined with mesh-refinement heuristics, to ensure that the solution is dynamically feasible, i.e., that the state trajectory satisfies the dynamics of the system. Shooting methods, on the other hand, rely on integration of the true dynamics of the system, and hence produce dynamically feasible solutions.

A notable characteristic of state-of-the-art direct methods for trajectory optimization, including shooting-based methods, is that they typically exhibit inter-sample constraint violation [134]. While multiple shooting itself ensures continuous-time satisfaction of the dynamics, path constraints are only imposed at nodes due to the discrete-time nature of formulations, and the continuous-time trajectory obtained by propagating the system with the optimized controls can violate the imposed constraints in between nodes, as shown in Figure 7.1. This is often mitigated by either increasing the number of nodes or using mesh-refinement heuristics. In [65], however, it was shown that constraint satisfaction in continuous-time can be ensured by using an integral reformulation of constraints [147, 89, 66] within a solution method that ensures continuous-time dynamic feasibility. The method has been used in various applications since [51, 219, 69, 70, 216, 217, 198, 242].

Real-time trajectory optimization requires high-performance numerical optimization solvers under the hood. Code generation has recently emerged as an effective approach to obtaining real-time implementations of optimization algorithms with minimal user effort [141]. The general idea is to provide for a simple user-interface to specify problems that belong to a certain class, typically by means of a domain specific language (DSL), and to generate efficient code, tailored to that problem class (usually in C), that is amenable to real-time

implementation. Code generation tools exist for general convex optimization, including linear programs (LPs) and quadratic programs (QPs) [140, 14], second-order cone programs (SOCPs) [63, 189, 50], and nonconvex optimization [197]. Further, code generation tools have been developed specifically for trajectory optimization, mostly in the context of nonlinear model predictive control (NMPC) using nonlinear programming (NLP) solvers that are either based on interior point methods (IPMs) or sequential quadratic programming (SQP) [95, 222].

There exist many popular software tools for optimal control and embedded optimization: see [135, Table 1] for a catalog of general optimal control problem parsers and [222, Table 1] for an overview of embedded optimization packages. Most modern direct optimal control software, whether based on collocation or shooting methods, provide high-level modeling interfaces and ultimately transcribe the problem into a sparse nonlinear program (NLP), which is then solved using a generic NLP solver such as IPOPT [230] or SNOPT [79].

Among collocation-based methods—and specifically, pseudospectral methods—the MATLAB-based $\mathbb{GPOPS}-\mathbb{II}$ software [159] and its C++ counterpart, $\mathbb{CGPOPS}$ [10], are especially popular for aerospace mission design. Dymos [72] offers similar capabilities in Python, targeting multidisciplinary trajectory optimization. Among shooting-based approaches, ACADO [94] and acados [222] are among the most widely used optimal control software packages. They focus on high-performance implementations of SQP-based algorithms, using structure-exploiting quadratic program (QP) solvers such as HPIPM [74] to solve the resulting QP subproblems. ACADO and acados are the staple for real-time nonlinear model predictive control (NMPC) applications and have seen considerable adoption in embedded and industrial settings [222].

In contrast to pseudospectral and SQP-based optimal control methods, both of which assume twice-differentiability in the nonlinear dynamics and constraints to enable the use of second-order information, a growing class of sequential convex programming (SCP)-based methods has emerged. SCP-based methods only require once-differentiability in the nonconvex constraints, and can natively handle convex conic constraints. An SCP-based tool

that has been growing in popularity is the SCP Toolbox [134], which is written in Julia, and includes SCP algorithms such as the penalized trust region (PTR) algorithm [174] and the guaranteed sequential trajectory optimization (GuSTO) algorithm [36]. We note that the SCvxPyGen package is a recently introduced Python-based tool that automatically generates code for SCP implementations [24]. It represents a preliminary step toward an SCP-based optimal control framework with automatic code generation, but at present only supports a restricted problem template, focusing mainly on obstacle-avoidance-type constraints, and uses an Euler integration scheme for discretization.

SCvxGEN is based on the *successive convexification* framework [66], which combines a multiple-shooting-based approach with sequential convex programming (SCP) to generate trajectories that are not only dynamically feasible, but also satisfy the imposed constraints in continuous-time, while preserving favorable convergence properties.

SCvxGEN is designed to be user-friendly, allowing users to specify their true free-final-time optimal control problems in *continuous-time* via a simple high-level user interface that relies on SymPy [151], a symbolic mathematics and computer algebra system in Python, a high-level programming language [220]. SCvxGEN includes a modeling library for conditional and logical constraints, i.e., *signal temporal logic (STL) specifications*, that allows users to specify constraints in close-to-plain language, which are then internally converted to equivalent smooth forms that are exact [218]. SCvxGEN automatically generates the required Jacobians via SymPy. It can also automatically detect convexity in the optimal control problem, and choose the solution method accordingly (either pure convex optimization or sequential convex programming). Finally, it automatically reformulates the problem to achieve continuous-time constraint satisfaction and generalized time-dilation [66].

SCvxGEN automates the generation of a custom, real-time solver for full-stack trajectory optimization [104]. The code generated by SCvxGEN for the nonconvex solver, including initial guess generation, multiple-shooting discretization/linearization, fixed-step RK4 integrators, problem preconditioning, and convex optimization, is not only embeddable, but also readable and maintainable, unlike typical code generation schemes. While the core solver

is in C, the user can execute the solver and access the solution via the Python interface, without having to interact with the C code.

SCvxGEN also provides the option to interface with different convex optimization solvers, such as QOCO [50], ECOS [59], and OSQP [201] via the CVXPYGEN framework [189], or PIPG [238] directly, and linear algebra libraries using BLAS routines [31, 244], for example. SCvxGEN is designed to be highly efficient, generating library-free C code with static memory allocations only (when used with QOCOGEN or PIPG) and dense linear algebra operations without any matrix inversions or factorizations (when used with PIPG). It has already been tested on both CPUs [66] for real-time trajectory optimization (see Figure 7.10 for a numerical example) and GPUs, via automatically generated CUDA code, for fast Monte Carlo analysis [51].

Some of the features of SCvxGEN are as follows (see Figure 1.1):

1. A simple user-interface in a high-level programming language that accepts continuous-time definitions for constraints and dynamics.
2. A modeling library for signal temporal logic (STL) specifications for conditional and logical constraints.
3. Automatic generation of the required Jacobians.
4. Automatic detection of convexity in the optimal control problem.
5. Automatic reformulation of the problem for continuous-time constraint satisfaction.
6. Automatic reformulation of the problem for generalized time-dilation.
7. Generation of highly readable (with the same variable symbols defined) embeddable C code for full-stack SCP (including initial guess generation, multiple-shooting discretization/linearization, and convex optimization).
8. Automatic scaling of the decision variables, preconditioning of the dynamics and constraints, and tuning of the hyperparameters.
9. The option to interface with different off-the-shelf convex optimization solvers via CVXPYGEN.

10. The option to interface with BLAS routines (aside from the custom linear algebra library provided).
11. The ability to generate C code with static memory allocations only, i.e., no dynamic memory allocations in the entire SCP framework (with QOCOGEN and PIPG).
12. The ability to generate C code with dense linear algebra operations only without any matrix-inversions/factorizations in the entire SCP framework (with PIPG).
13. The ability to modify parameters and initial guesses on-the-fly, via the Python interface, without having to recompile the generated code.

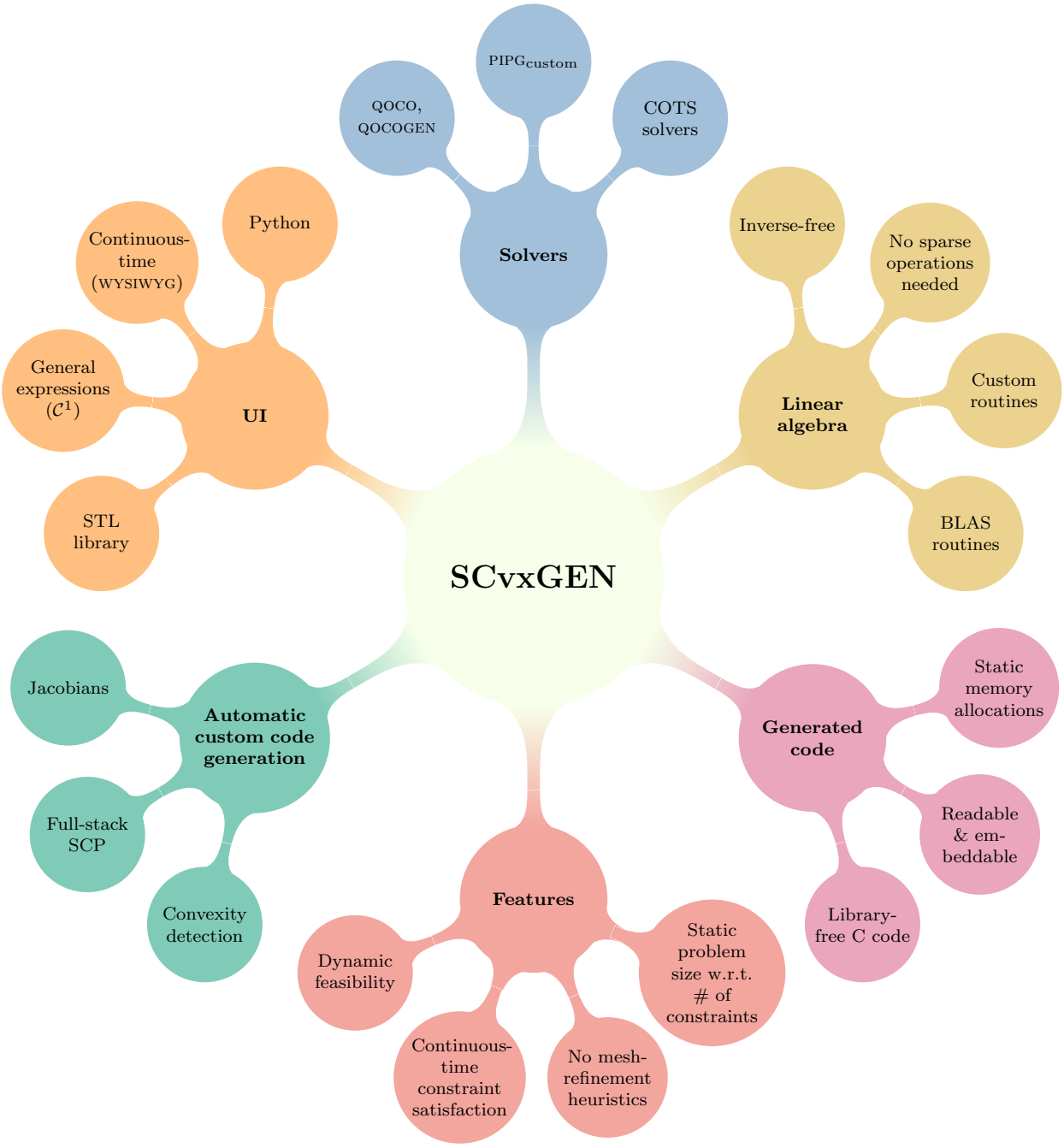

Figure 1.1: An overview of SCvxGEN**.**

Chapter 2

SEQUENTIAL CONIC OPTIMIZATION FOR TRAJECTORY GENERATION

In this chapter, we first describe the sequential conic optimization (SeCO) framework for trajectory generation in §2.1. SeCO involves solving a sequence of conic subproblems, the template for which is described in §2.2. Finally, we describe a specialized conic optimization solver to solve these subproblems in §2.3, and discuss *solver customization* in §2.4.

2.1 Trajectory Optimization Framework

In this section, we discuss the components of the sequential conic optimization (SeCO) framework, shown in Figure 2.1.

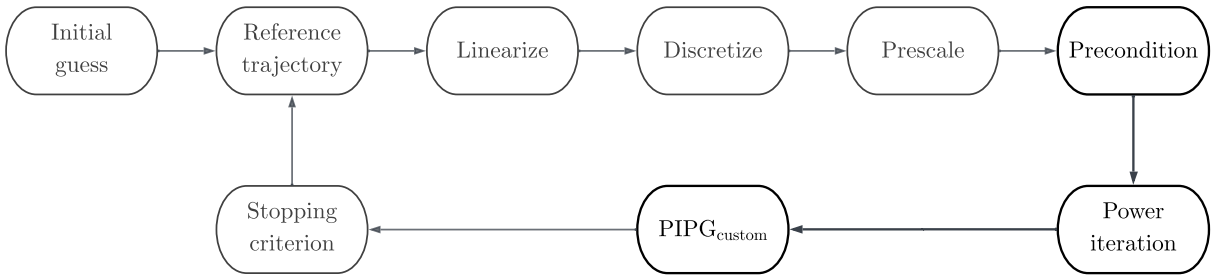

Figure 2.1: An overview of the SeCO framework; the blocks in bold constitute the low-level solver.

2.1.1 Initial guess generation

Sequential convex programming methods require an initial guess to begin the iterative solution procedure. The initial guess generation strategy is a design choice and what constitutes

a "good" initial guess is highly problem dependent. Typically, linearly interpolating between the initial and terminal boundary conditions serves as a reasonable default initial guess strategy. However, given that the quality of the initial guess can significantly impact the convergence behavior and quality of the solution, it is generally best to embed as much information about the problem at hand as possible in its generation [133, 176, 208, 134, 102].

2.1.2 Dynamics

Our approach to treating the dynamics closely follows the methods provided in [208, 176, 174], with some key distinctions, such as the inverse-free propagation step, as described later in this section. All of these approaches yield an *exact* discretization of the linear time-varying (LTV) system under consideration—which means that the discrete-time trajectory exactly coincides with the continuous-time trajectory at the temporal nodes—and are analytically equivalent. In practice, however, the approach we propose herein leads to a much simpler implementation, without the need for any matrix factorizations/inversions, thus also making the implementation more numerically stable and reliable. Further, a very similar approach, albeit in a multi-phase guidance setting, is provided in [103].

Time-dilation

The original nonlinear dynamics over the entire time-horizon are given by Equation 2.1. Without loss of generality, we assume that the initial time is zero.

$$\dot{x}(t) = f(t, x(t), u(t)), \;\; t \in [0, t_f) \tag{2.1}$$

Now, we define an invertible linear map, $\tau : [0, t_f) \to [0, 1)$, as shown in Equation 2.2.

$$\tau(t) = \frac{t}{t_f}, \;\; t \in [0, t_f) \tag{2.2}$$

This mapping is referred to as *time-dilation*, as it dilates (normalizes) the wall-clock time-horizon to a chosen fixed interval—$[0, 1)$, in our case. Next, we invoke the chain-rule to obtain Equation 2.3, where $\mathring{\square}$ denotes the derivative operator with respect to the dilated time, τ.

$$\begin{aligned}\mathring{x}(t) = \frac{d}{d\tau}\, x(t) = \frac{dt}{d\tau}\, \frac{d}{dt}\, x(t) = \frac{dt}{d\tau}\, \dot{x}(t) = \underbrace{t_f^-}_{s}\, \dot{x}(t) \\ = s\, f(t, x(t), u(t)) := F(t, x(t), u(t), s) \end{aligned} \tag{2.3}$$

The multiplier in Equation 2.3, $s := t_f^- \in \mathbb{R}_+$, termed the *dilation factor*, evaluates to the time-of-flight. Given Equations 2.2 and 2.3, t can now be expressed as a function of τ, i.e., $t(\tau) = s\,\tau$, $\tau \in [0, 1)$. Henceforth, we treat τ as the independent variable instead of t, and replace the temporal argument, $t = t(\tau)$, by τ, for notational simplicity.

Linearization

We begin by considering nonlinear systems that can be represented as ordinary differential equations (ODEs), as given by Equation 2.4, where $x(\cdot) \in \mathbb{R}^{n_x}$ is the state vector, $u(\cdot) \in \mathbb{R}^{n_u}$ is the control input vector, $s \in \mathbb{R}$ is the parameter, which in our case, is the time-of-flight, and $F(\cdot) \in \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}^{n_x}$ is the nonlinear function representing the time-dilated dynamics, which is assumed to be at least once continuously differentiable.

$$\mathring{x}(\tau) = F(\tau, x(\tau), u(\tau), s) \tag{2.4}$$

The first-order Taylor series expansion of Equation 2.4 about an arbitrary reference trajectory $(\overline{x}(\tau), \overline{u}(\tau), \overline{s})$yields a linear time-varying (LTV) system, given by Equation 2.5. The equation is approximate as a consequence of linearization via truncation of the higher-order (≥ 2)

terms in the Taylor series expansion.

$$\mathring{x}(\tau) \approx A(\tau)x(\tau) + B(\tau)u(\tau) + S(\tau)s + d(\tau) \tag{2.5}$$

where

$$A(\tau) := \nabla_x F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s}) \tag{2.6a}$$

$$B(\tau) := \nabla_u F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s}) \tag{2.6b}$$

$$S(\tau) := \nabla_s F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s}) \tag{2.6c}$$

$$\begin{aligned} d(\tau) := {} & F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s}) \\ & - A(\tau)\,\overline{x}(\tau) - B(\tau)\,\overline{u}(\tau) - S(\tau)\,\overline{s} \end{aligned} \tag{2.6d}$$

Control parameterization

We adopt a first-order hold (FOH) parameterization of the control input signal, which, in contrast to pseudospectral discretization methods, possesses the following characteristics: (1) inter-sample satisfaction of the convex control constraints is guaranteed (provided they are satisfied at the discrete temporal nodes); and, (2) the resulting conic subproblem has a sparsity pattern that is amenable to real-time implementation [133, 208]. These properties make FOH attractive for optimal control applications.

With FOH, the control input variables are only defined at the discrete temporal nodes, and the continuous-time control input signal is obtained by linearly interpolating between the discrete values at successive nodes. Note that the control input signal is restricted to a continuous piecewise-affine function of time, with only a finite number (N) of discrete control variables, as shown in Equation 2.7.

$$u(\tau) = \sigma_k^-(\tau)\,u_k + \sigma_k^+(\tau)\,u_{k+1}, \quad \forall \tau \in [\tau_k, \tau_{k+1}) \tag{2.7}$$

where

$$\sigma_k^-(\tau) := \frac{\tau_{k+1} - \tau}{\tau_{k+1} - \tau_k}, \;\; \sigma_k^+(\tau) := \frac{\tau - \tau_k}{\tau_{k+1} - \tau_k}, \;\; k = 1\!:\!N\!-\!1$$

As shown in Equation 2.8, the LTV dynamics in terms of deviations from the reference can now be written using the piecewise-affine control input parameterization given by Equation 2.7. $\Delta\square$ denotes the deviation of a quantity from its reference, i.e., $\Delta\square := \square - \overline{\square}$, and $\Delta\mathring{x}(\tau) := \mathring{x}(\tau) - F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s})$. Henceforth, "=" is used in lieu of "≈" for notational simplicity, with the understanding that the LTV system is a first-order approximation of the original nonlinear system.

$$\Delta\mathring{x}(\tau) = A(\tau)\,\Delta x(\tau) + B(\tau)\,\sigma_k^-(\tau)\,\Delta u_k + B(\tau)\,\sigma_k^+(\tau)\,\Delta u_{k+1} + S(\tau)\,\Delta s \tag{2.8}$$

Equation 2.8 has a unique solution [11, 134], given by Equation 2.9, $\forall \tau \in [\tau_k, \tau_{k+1})$.

$$\Delta x(\tau) = \Phi(\tau, \tau_k)\,\Delta x(\tau_k) + \int_{\tau_k}^{\tau} \Phi(\tau, \zeta)\left\{B(\zeta)\,\sigma_k^-(\zeta)\,\Delta u_k + B(\zeta)\,\sigma_k^+(\zeta)\,\Delta u_{k+1} + S(\zeta)\,\Delta s\right\} \mathrm{d}\zeta \tag{2.9}$$

where $\Phi(\tau, \tau_k)$ is a matrix that satisfies the matrix differential equation given by Equation 2.10 such that $\Phi(\tau_k, \tau_k) = I_{n_x}$, and is called the *state transition matrix* (STM).

$$\mathring{\Phi}(\tau, \tau_k) = A(\tau)\,\Phi(\tau, \tau_k) \tag{2.10}$$

Discretization

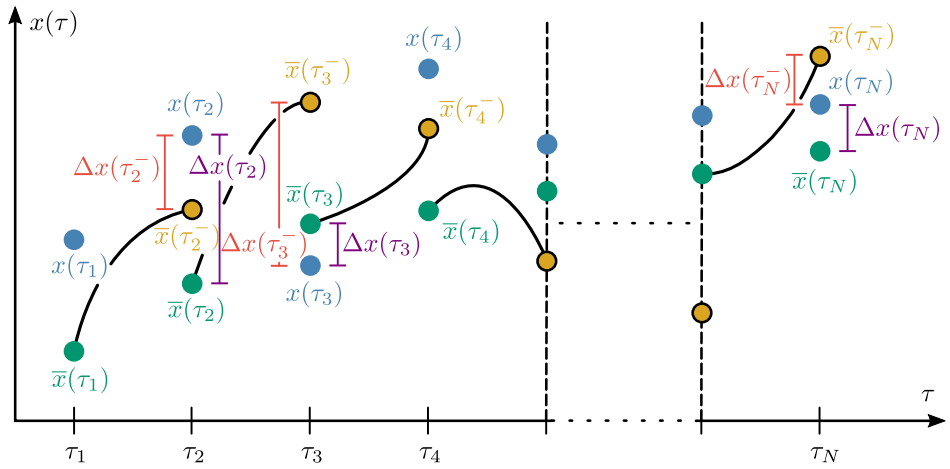

Figure 2.2: Propagation of the state in the discretization procedure.

Evaluating Equation (2.9) at $\tau = \tau_{k+1}^-$, we get Equation (2.11):

$$\Delta x\left(\tau_{k+1}^-\right) = A_k \Delta x(\tau_k) + B_k^- \Delta u_k + B_k^+ \Delta u_{k+1} + S_k \Delta s \tag{2.11}$$

Note that Equation (2.12), which we refer to as the *stitching condition*, holds, as is evident from Figure 2.2:

$$\Delta x\left(\tau_{k+1}^-\right) + \overline{x}\left(\tau_{k+1}^-\right) = \Delta x(\tau_{k+1}) + \overline{x}(\tau_{k+1}) = x(\tau_{k+1}), \quad k = 1\!:\!N-1 \tag{2.12}$$

The discretized dynamics can now be written as shown in Equation (2.13):

$$\Delta x\left(\tau_{k+1}^-\right) = \Delta x_{k+1} + \overline{x}_{k+1} - x_{k+1}^{\text{prop}} \tag{2.13a}$$

$$= A_k \, \Delta x_k + B_k^- \, \Delta u_k + B_k^+ \, \Delta u_{k+1} + S_k \Delta s \tag{2.13b}$$

$$\implies \Delta x_{k+1} = A_k \, \Delta x_k + B_k^- \, \Delta u_k + B_k^+ \, \Delta u_{k+1} + S_k \, \Delta s + d_k \tag{2.13c}$$

where $\Delta x_k := \Delta x(\tau_k)$, $\Delta x(\tau_1) := 0$, $\Delta x_{k+1} := \Delta x(\tau_{k+1})$, $x_{k+1}^{\text{prop}} := \overline{x}\left(\tau_{k+1}^-\right)$, $\overline{x}_{k+1} := \overline{x}(\tau_{k+1})$, and $\overline{u}(\tau) := \sigma_k^-(\tau)\,\overline{u}_k + \sigma_k^+(\tau)\,\overline{u}_{k+1}$, $\forall \tau \in [\tau_k, \tau_{k+1})$.

Here,

$$A_k = I_{n_x} + \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} A(\zeta)\, \Psi_A(\zeta)\, \mathrm{d}\zeta \tag{2.14a}$$

$$B_k^- = \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} \{A(\zeta)\, \Psi_{B^-}(\zeta) + B(\zeta)\, \sigma_k^-(\zeta)\}\, \mathrm{d}\zeta \tag{2.14b}$$

$$B_k^+ = \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} \{A(\zeta)\, \Psi_{B^+}(\zeta) + B(\zeta)\, \sigma_k^+(\zeta)\}\, \mathrm{d}\zeta \tag{2.14c}$$

$$S_k = \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} \{A(\zeta)\, \Psi_S(\zeta) + S(\zeta)\}\, \mathrm{d}\zeta \tag{2.14d}$$

$$d_k = x_{k+1}^{\mathrm{prop}} - \overline{x}_{k+1} \tag{2.14e}$$

where

$$\mathring{\Psi}_A(\tau) = A(\tau)\, \Psi_A(\tau) \tag{2.15a}$$

$$\mathring{\Psi}_{B^-}(\tau) = A(\tau)\, \Psi_{B^-}(\tau) + B(\tau)\, \sigma_k^-(\tau) \tag{2.15b}$$

$$\mathring{\Psi}_{B^+}(\tau) = A(\tau)\, \Psi_{B^+}(\tau) + B(\tau)\, \sigma_k^+(\tau) \tag{2.15c}$$

$$\mathring{\Psi}_S(\tau) = A(\tau)\, \Psi_S(\tau) + S(\tau) \tag{2.15d}$$

x_{k+1}^{prop} is evaluated as follows:

$$x_{k+1}^{\mathrm{prop}} = \overline{x}_k + \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} F(\zeta, \overline{x}(\zeta), \overline{u}(\zeta), \overline{s})\ \mathrm{d}\zeta \tag{2.16}$$

and $\Psi_A(\tau) := \Phi(\tau, \tau_k)$.

2.1.3 Prescaling

The decision variables are prescaled to ensure that the solutions are roughly on the same order of magnitude (between 0 and 1, in our case). This is an important aspect of numerical optimization algorithms, and can significantly improve solution quality and speed up convergence. The interested reader is referred to [134] for more details on variable scaling.

2.2 Conic Subproblem Template

2.2.1 Constraint classification

The template SeCO subproblem is strongly convex, and is given by Equations 2.17, where all the decision variables are stacked into a single high-dimensional vector, z. Here, Q is a positive definite matrix, and $\mathbb{D}$ is a closed convex set that admits a closed-form projection operation. This vectorized problem possesses a sparsity structure that is amenable to customization, as described in Algorithm 8.

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \frac{1}{2} z^\top Q z + \langle q, z \rangle && \text{(2.17a)} \\ \text{subject to} \quad & Hz - h = 0 && \text{(2.17b)} \\ & z \in \mathbb{D} \end{aligned}$$

We prefer to include as many constraints in the form $z \in \mathbb{D}$ as possible for the following reasons: (1) The constraints $z \in \mathbb{D}$ will be satisfied at every iteration of PIPG. Hence, including more constraints in set $\mathbb{D}$ leads to a smaller search space. In contrast, the constraint $Hz - h = 0$ is only satisfied asymptotically and does not affect the search space; (2) Transforming many popular constraint sets in optimal control–such as ℓ_2-norm balls, cylinders, boxes–into conic constraints, requires extra auxiliary variables. Imposing these constraints in the form $z \in \mathbb{D}$ eliminates the need for such auxiliary variables and, as a result, decreases the problem size. As shown in the discretized conic subproblem in §6.2.7, every constraint other than the dynamics is classified into set $\mathbb{D}$—this means that these path constraints are exactly satisfied at every solver iteration (up to constraint approximations), and may prove beneficial in the case of premature termination of the solver in an emergency scenario, for instance. The dynamics constraint belongs to the zero cone, and is satisfied asymptotically as the solver converges to an optimum.

2.2.2 Virtual state

Artificial infeasibility refers to the phenomenon wherein linearization of the nonconvex constraints in a problem can render the convex subproblem infeasible even if there exists a feasible solution to the original nonconvex problem. In order to mitigate this, one approach is to add slack variables to the linearized constraints, thus ensuring that the subproblem is always feasible. These slack variables are usually unconstrained, yet heavily penalized.

The value of these slack variables should go to zero at convergence for a solution to be feasible with respect to the original nonconvex problem. The slack variable is usually referred to as *virtual control* when it is added to the linearized dynamics, and *virtual buffer* when it is added to the linearized constraints [134]. It has been observed that virtual buffer terms are usually not required if virtual control is used [208, 176]. In such cases, however, the intermediate reference solutions will not be feasible with respect to the LTV dynamics unless the value of the virtual control is zero. If dynamic feasibility of the intermediate reference trajectories is of importance, one approach could be to buffer the constraints and leave the dynamics equation unchanged.

We opt to use a third approach, called *virtual state* [103]. The general idea of this approach is to entirely decouple the dynamics and the control constraints from the state constraints, and to exactly satisfy all the constraints at each iteration, while ensuring that the subproblem is always feasible. In order to achieve this, we introduce a new variable, the virtual state, which acts as a *copy* of the original state variable. If x is the actual state vector, u is the control input vector, and ξ is the virtual state vector, the dynamics constraint is imposed on x, the control constraints are imposed on u, and all the state constraints are imposed on ξ.

In order to ensure that both the dynamics and the path constraints are satisfied at convergence, we minimize the error between x and ξ by including the squared distance between them as a quadratic penalty term in the objective function and heavily penalizing it. The virtual state approach has the benefit of not altering the dynamics manifold, unlike the

virtual control approach [208, 176], and preserving the shapes of the conic state constraint sets, unlike the virtual buffer approach [134]. The left superscript, ${}^{\xi}\square$, is used to denote virtual state variables.

Proposition 1. *Given a discrete-time subproblem with $N \in \mathbb{Z}_{++}$ nodes in the temporal grid, the virtual state error penalty, given by $\sum_{k=1}^{N} \|x_k - \xi_k\|_2^2$, is convex, where $x_k, \xi_k \in \mathbb{R}^n$, n being the dimension of the state, and $k = 1, \ldots, N$.*

Proof. Let $y_k := (x_k, \xi_k) \in \mathbb{R}^n$. Therefore, $\|x_k - \xi_k\|_2^2 = y_k^\top M y_k$, where $M = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \otimes I_n$. Since $\operatorname{spec} M \in \{0, 2\}$, M is positive semidefinite (PSD). Therefore, the quadratic form, $y_k^\top M y_k$, is convex. Since the sum of convex functions is convex, $\sum_{k=1}^{N} y_k^\top M y_k = \sum_{k=1}^{N} \|x_k - \xi_k\|_2^2$ is also convex. □

2.2.3 Trust region

The trust region radius is the distance between the solution to a subproblem and the trajectory about which the system and nonconvex constraints are linearized to create that subproblem. The purpose of a trust region is to: (1) make sure that the solution does not venture too far from the reference trajectory so as to ensure that the linearization is sufficiently accurate, thus preserving its validity; and, (2) mitigate *artificial unboundedness*, which refers to the phenomenon wherein linearization can render the cost unbounded from below even if it is bounded in the original nonconvex problem [134].

A quadratic trust region penalty term is added to the original cost function to ensure that the solution to each subproblem does not venture too far from the reference trajectory, preserving the validity of the linearization. This soft trust region mitigates artificial unboundedness.

2.3 First-Order Methods for Convex Optimization

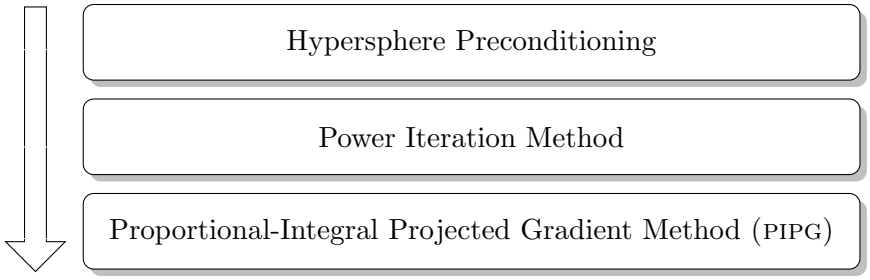

Figure 2.3: The SeCO subproblem solver.

In this section, we describe the development of a high-performance first-order conic optimization solver, as outlined in Figure 2.3, to solve the convex subproblems in SeCO, which we then customize in §2.4.

In this section, we describe the development of a high-performance first-order conic optimization solver, as outlined in Figure 2.3, to solve the convex subproblems in SeCO, which we then customize in §2.4.

2.3.1 Problem Preconditioning

First-order methods are sensitive to problem conditioning and typically perform poorly on ill-conditioned problems [201]. As SCP (SeCO) implementations typically impose a heavy weight on the virtual term penalty relative to the original cost and trust region penalties [134], the objective function of the resulting convex (conic) subproblem is inherently ill-conditioned.

In this work[†], we use the hypersphere preconditioner [101], shown in Algorithm 1, for problems that fit the template of SeCO. This procedure uses an exact Cholesky factorization of the objective function, and a row-normalization of the constraint matrix, while effectively exploiting the structure of the conic subproblem in SeCO with minimal computational overhead—in fact, the Cholesky factor can be computed analytically, without the need for any

[†]A formal treatment of the proposed preconditioning procedure is provided in Chapter 3.

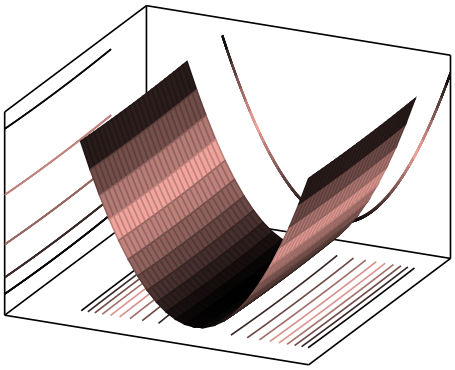

Figure 2.4: An ill-conditioned bivariate quadratic function (prior to preconditioning, for instance).

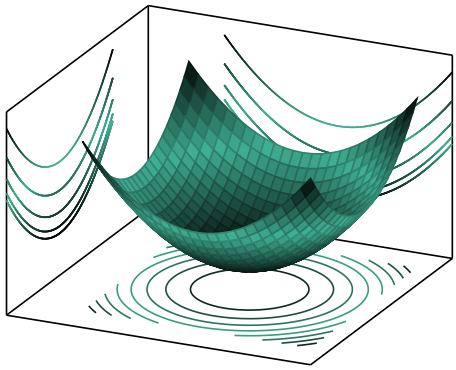

Figure 2.5: A well-conditioned bivariate quadratic function (after preconditioning, for instance).

explicit matrix factorization/inversion operations or iterative heuristics; i.e., the structure of the SeCO subproblem enables the computation of the Cholesky factor of the objective function matrix in closed-form, as shown in the Algorithm 5. Further, to compute the optimal cost-scaling factor in terms of minimizing the condition number of the Karush Kuhn Tucker (KKT) matrix of the problem [101], we adopt the shifted power iteration method, the customized version of which we describe in Algorithm 7. Further, the transformed dynamics matrix still possesses a sparsity structure that is amenable to customization, as shown in Algorithm 8. The preconditioner transforms ill-conditioned quadratic functions to well-conditioned quadratic functions, the level sets of which are ℓ_2-norm hyperspheres; hence the name. The effect of preconditioning on an ill-conditioned quadratic objective function, such as the one in Figure 2.4, is depicted in Figure 2.5. In our initial testing, we observed a reduction in the number of solver iterations to convergence of roughly 5 to 10 times with this preconditioner. Further, we normalize the rows of the constraint matrix—this helps reduce the condition number of the preconditioned constraint matrix in practice [101].

Algorithm 1 Hypersphere Preconditioning

Inputs: Q, q, H, h, $\mathbb{D}$

Require: $Q \succ 0$, $Q = Q^\top$

1: $L^\top L \leftarrow \operatorname{chol} Q$ ▷ Cholesky decomposition
2: $L_{\text{inv}} \leftarrow L^{-1}$
3: $\hat{\mathbb{D}} \leftarrow L\,\mathbb{D}$
4: $\hat{H} \leftarrow H L_{\text{inv}}$
5: $\hat{H} \leftarrow E\hat{H}$ ▷ E: diagonal matrix with reciprocals of row-norms of $\hat{H}$
6: $\hat{h} \leftarrow Eh$
7: $\sigma_{\max} \leftarrow$ Algorithm 2 ▷ power iteration
8: $\sigma_{\min} \leftarrow$ Algorithm 3 ▷ shifted power iteration
9: $\lambda \leftarrow \sqrt{\frac{\sigma_{\min}}{2}}$
10: $\hat{q} \leftarrow \lambda\, L_{\text{inv}}^\top q$

Ensure: $\square \in \hat{\mathbb{D}} \Leftrightarrow L^{-1}\square \in \mathbb{D}$

Return: λ, $\hat{q}$, $\hat{H}$, $\hat{h}$, $\hat{\mathbb{D}}$, L, L_{inv}, $\sigma_{\max}$

As shown in Algorithm 1, we scale the objective function with a positive scalar, λ, that factors into the step-sizes of PIPG and hence serves as a tuning parameter. Scaling the objective function appropriately can also help keep the magnitude of the dual variables in check [201]. The particular choice of λ in Algorithm 1 leads to minimization of the condition number of the KKT matrix of the preconditioned problem [101]. That said, λ can be manually tuned to obtain good performance as well.

The preconditioned conic subproblem is shown in Problem 2.18b:

$$\underset{\hat{z}}{\text{minimize}} \quad \frac{\lambda}{2}\,\hat{z}^\top \hat{z} + \langle \hat{q}, \hat{z} \rangle \tag{2.18a}$$

$$\text{subject to} \quad \hat{H}\hat{z} - \hat{h} = 0 \tag{2.18b}$$

$$\hat{z} \in \hat{\mathbb{D}}$$

The objective function matrix of the preconditioned problem is of the form $\lambda\, I$, with a condition number of 1, which is the minimum attainable condition number (and hence, the

preconditioner is optimal). The transformed projection $\hat{z} \in \hat{\mathbb{D}}$ needs to preserve membership of the original primal variable, z, to set $\mathbb{D}$, i.e., $\hat{z} \in \hat{\mathbb{D}} \Leftrightarrow z \in \mathbb{D}$ [157]. The structure of the conic subproblem in SeCO naturally preserves this, as is apparent from Algorithm 5. Post-solution, the original primal variable can be recovered as follows: $z = L_{\text{inv}}\hat{z}$.

2.3.2 Power Iteration Method

The power iteration method described in Algorithm 2 computes the maximum singular value of a given diagonalizable matrix [214]. We can exploit the structure of matrix $\hat{H}$ in trajectory optimization problems to customize Algorithm 2 to Algorithm 6 so as to avoid sparse linear algebra operations with large dimensional matrices and vectors.

Algorithm 2 Power Iteration Method

Inputs: $\hat{H}$, z, ϵ_{abs}, ϵ_{rel}, ϵ_{buff}, $j_{\max}$

Require: $\|z\|_2 > 0$

$\sigma \leftarrow \|z\|_2$ ▷ initialization
for $j \leftarrow 1 : j_{\max}$ **do**
  $w \leftarrow \frac{1}{\sigma}\hat{H}z$
  $z \leftarrow \hat{H}^\top w$
  $\sigma^\star \leftarrow \|z\|_2$
  if $|\sigma^\star - \sigma| \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\{\sigma^\star, \sigma\}$ **then** ▷ stopping criterion
    break
  else if $j < j_{\max}$ **then**
    $\sigma \leftarrow \sigma^\star$
  end if
end for
$\sigma \leftarrow (1 + \epsilon_{\text{buff}})\,\sigma^\star$ ▷ buffer the (under) estimated maximum singular value

Return: σ ▷ $\approx \max \operatorname{spec} \hat{H}^\top \hat{H} = \sigma_{\max}(\hat{H}^\top \hat{H}) = \|\hat{H}\|_2^2$

2.3.3 Shifted Power Iteration Method

The shifted power iteration method described in Algorithm 3 is similar to the power iteration method, except that it computes an estimate for the smallest singular value of a given diagonalizable matrix instead [234]. See the appendix in [101] for a detailed description of this algorithm. Note that the shifted power iteration method is used solely to estimate the smallest singular value of $\hat{H}\,\hat{H}^\top$ to determine the optimal value for the cost-scaling factor, λ. If λ is manually tuned instead, this algorithm is not required. The customized version of this algorithm is given by Algorithm 7.

Algorithm 3 Shifted Power Iteration Method

Inputs: $\hat{H}$, w, ϵ_{abs}, ϵ_{rel}, ϵ_{buff}, $j_{\max}$, $\sigma_{\max}$

Require: $\|w\|_2 > 0$

$\tilde{\sigma} \leftarrow \|w\|_2$ ▷ initialization
for $j \leftarrow 1 : j_{\max}$ **do**
  $z \leftarrow \hat{H}^\top w$
  $w \leftarrow \frac{1}{\tilde{\sigma}} (\hat{H}\, z - \sigma_{\max}\, w)$ ▷ $\sigma_{\max} := \|\hat{H}\|^2$
  $\tilde{\sigma}^\star \leftarrow \|w\|_2$
  if $|\tilde{\sigma}^\star - \tilde{\sigma}| \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\{\tilde{\sigma}^\star, \tilde{\sigma}\}$ **then** ▷ stopping criterion
    break
  else if $j < j_{\max}$ **then**
    $\tilde{\sigma} \leftarrow \tilde{\sigma}^\star$
  end if
end for
$\sigma_{\min} \leftarrow (1 - \epsilon_{\text{buff}})\,(\sigma_{\max} - \tilde{\sigma}^\star)$ ▷ buffer the (over) estimated minimum singular value

Return: $\sigma_{\min}$ ▷ $\approx \min \operatorname{spec} \hat{H}\,\hat{H}^\top = \sigma_{\min}(\hat{H}\,\hat{H}^\top)$

*2.3.4 Proportional-Integral Projected Gradient Method (*PIPG*)*

The proportional-integral projected gradient method, PIPG, is a first-order primal-dual algorithm for conic optimization [239]. It allows matrix-factorization/inverse-free and easily-verifiable solver implementations for real-time and embedded applications [240]. PIPG achieves the optimal global convergence rates (worst-case) in theory, and performs much faster in practice [238]. It exploits the sparsity structure of conic constraints via parallel matrix operations and the geometric structure of constraint sets via efficient closed-form projections [237]. Unlike most off-the-shelf methods, PIPG allows for warm-starting and enjoys a light computational overhead, as it avoids the cumbersome canonical transformation procedure that standard conic programs are subject to. It is also compatible with extrapolation, which has been shown to accelerate convergence [238]. PIPG not only enables versatile convex optimization, but also has the ability to boost the performance of sequential convex programming (SCP) methods for nonconvex optimization. Moreover, the SeCO framework specializes SCP algorithms to exploit features of PIPG to solve nonconvex optimal control problems in real-time [103].

One of the major applications of PIPG is in solving trajectory optimization problems, given that all of the sparse linear algebra operations in Equations 4 can be devectorized [64, Algorithm 2], as shown in Algorithm 8, and posed as simple, small-dimensional matrix-vector manipulations that are suitable for real-time performance onboard resource-constrained embedded hardware.

PIPG converges to an optimal solution when the difference between two consecutive iterates converges to zero [238, Theorem 1]. Hence, in practice, we terminate the solver when the difference between two consecutive iterates is sufficiently small. In particular, given a relative accuracy tolerance ϵ_{rel} and an absolute accuracy tolerance ϵ_{abs}, we terminate PIPG

when the following conditions are met:

$$\left\|\hat{z}^{j+1} - \hat{z}^{j}\right\|_\infty \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\left\|\hat{z}^{j+1}\right\|_\infty, \left\|\hat{z}^{j}\right\|_\infty\right\},$$
$$\left\|\hat{w}^{j+1} - \hat{w}^{j}\right\|_\infty \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\left\|\hat{w}^{j+1}\right\|_\infty, \left\|\hat{w}^{j}\right\|_\infty\right\}$$

We note that such a combination of absolute and relative accuracy tolerances is popular among first-order solvers [201, 156].

Algorithm 4 PIPG

Inputs: $\hat{q}$, $\hat{H}$, $\hat{h}$, $\hat{\mathbb{D}}$, L, L_{inv}, λ, σ, ρ, ϵ_{abs}, ϵ_{rel}, j_{check}, $j_{\max}$,
$z^\star$, $\hat{w}^\star$ ▷ warm start

$\hat{z}^\star \leftarrow L\, z^\star$ ▷ transform previous primal solution
$\zeta^1 \leftarrow \hat{z}^\star$ ▷ initialize transformed primal variable
$\eta^1 \leftarrow \hat{w}^\star$ ▷ initialize transformed dual variable
$\alpha \leftarrow \frac{2}{\lambda + \sqrt{\lambda^2 + 4\sigma}}$ ▷ step-size
for $j \leftarrow 1 : j_{\max}$ **do**
  $\hat{z}^{j+1} = \pi_{\mathbb{D}}[\zeta^j - \alpha\,(\lambda\,\zeta^j + \hat{q} + \hat{H}^\top \eta^j)]$ ▷ projected gradient step
  $\hat{w}^{j+1} = \eta^j + \alpha\,(\hat{H}(2\,\hat{z}^{j+1} - \zeta^j) - \hat{h})$ ▷ PI feedback of affine equality constraint violation
  $\zeta^{j+1} = (1-\rho)\,\zeta^j + \rho\,\hat{z}^{j+1}$ ▷ extrapolate transformed primal variable
  $\eta^{j+1} = (1-\rho)\,\eta^j + \rho\,\hat{w}^{j+1}$ ▷ extrapolate transformed dual variable
  if $j \bmod j_{\text{check}} = 0$ **then** ▷ check stopping criterion every j_{check} iterations
    if $\left\|\hat{z}^{j+1} - \hat{z}^{j}\right\|_\infty \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\left\|\hat{z}^{j+1}\right\|_\infty, \left\|\hat{z}^{j}\right\|_\infty\right\}$ **and**
     $\left\|\hat{w}^{j+1} - \hat{w}^{j}\right\|_\infty \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\left\|\hat{w}^{j+1}\right\|_\infty, \left\|\hat{w}^{j}\right\|_\infty\right\}$ **then** ▷ stopping criterion
      break
    end if
  end if
end for
$z^\star \leftarrow L_{\text{inv}}\, \hat{z}^{j+1}$ ▷ recover original primal variable
$\hat{w}^\star \leftarrow \hat{w}^{j+1}$ ▷ retain transformed dual variable

Return: $z^\star$, $\hat{w}^\star$

2.4 Solver Customization

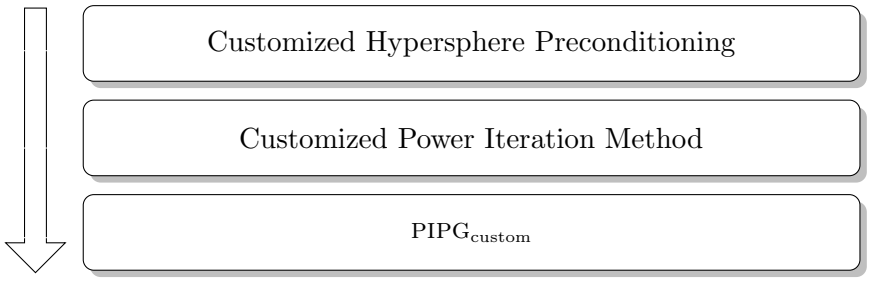

Figure 2.6: The customized SeCO subproblem solver.

We define customization as the exploitation of the sparsity pattern of the optimal control problem at hand, so as to enable low-dimensional matrix-vector multiplications and other dense linear algebra operations with devectorized variables, and thus avoid sparse linear algebra operations. In contrast to dense matrix-vector multiplications, sparse matrix-vector multiplications (SpMV) typically suffer from: (i) additional computational overhead in terms of instructions and storage; (ii) memory access patterns that are indirect and irregular; and, (iii) more cache misses, and are hence inefficient [235, 50]. Customized algorithms preclude these inefficient operations, and are especially effective with optimization problem sizes that are characteristic of onboard guidance [62, 64, 50]. In this section, we describe customization of the algorithms presented in Section 2.3, leading to the customized subproblem solver outlined in Figure 2.6.

2.4.1 Customized Preconditioning

Consider the vectorized conic optimization problem given by Equations (2.17), where

$$z = (x_1, \ldots, x_N, \xi_1, \ldots, \xi_N, u_1, \ldots, u_N, s) \in \mathbb{R}^{n_z}$$
$$q = (q_x, q_\xi, q_u, q_s) \in \mathbb{R}^{n_z}$$
$$Q = \begin{pmatrix} Q_{\text{state}} & & \\ & Q_u & \\ & & Q_s \end{pmatrix} \in \mathbb{S}^{n_z}_{++}$$

$Q_{\text{state}} := W_{\text{state}} \otimes I_{n_x N}$, $Q_u := w_{\text{tr}}\, I_{n_u N}$, $Q_s := w_{\text{tr}_s}$, $W_{\text{state}} := \begin{pmatrix} w_{\text{tr}} + w_{\text{vse}} & -w_{\text{vse}} \\ -w_{\text{vse}} & w_{\text{vse}} \end{pmatrix}$, and n_z is the length of z.

Q is symmetric positive definite (SPD) and therefore has a unique Cholesky decomposition [93, Corollary 7.2.9]. Further, since Q is block diagonal, the blocks are effectively decoupled, and the Cholesky decomposition can be applied directly to the individual blocks, as shown in Equation (2.19):

$$\operatorname{chol} Q = \begin{pmatrix} \operatorname{chol} Q_{\text{state}} & & \\ & \operatorname{chol} Q_u & \\ & & \operatorname{chol} Q_s \end{pmatrix} \tag{2.19}$$

Let L denote the Cholesky factor of Q, such that $Q = L^\top L$, $L_\square$ denote the Cholesky factor of a block partition of Q, such that $Q_\square = L_\square^\top L_\square$, and R denote the Cholesky factor of W_{state}, such that $W_{\text{state}} = R^\top R$.

Given two SPD matrices A and B, it can be shown that $\operatorname{chol}(A \otimes B) = \operatorname{chol} A \otimes \operatorname{chol} B$

[188]. Therefore,

$$\operatorname{chol} Q_{\text{state}} = \operatorname{chol} W_{\text{state}} \otimes \operatorname{chol} I_{n_x N} \tag{2.20}$$

$$= R^\top R \otimes I_{n_x N} \tag{2.21}$$

$$= L_{\text{state}}^\top L_{\text{state}} \tag{2.22}$$

where

$$R = \frac{1}{\sqrt{w_{\text{tr}} + w_{\text{vse}}}} \begin{pmatrix} w_{\text{tr}} + w_{\text{vse}} & -w_{\text{vse}} \\ 0 & \sqrt{w_{\text{tr}} w_{\text{vse}}} \end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{2.23}$$

$$\text{and} \quad L_{\text{state}} = R \otimes I_{n_x N} \tag{2.24}$$

Since Q_u is a diagonal matrix, $\operatorname{chol} Q_u = L_u^\top L_u$, where $L_u = \sqrt{w_{\text{tr}}}\, I_{n_u N}$. Since Q_s is a scalar, $\operatorname{chol} Q_s = L_s^2$, where $L_s = \sqrt{w_{\text{tr}_s}}$.

Finally, the Cholesky factor of Q can be computed as follows: $L = \operatorname{blkdiag}\{L_{\text{state}}, L_u, L_s\}$. Since L is block diagonal, its inverse can be written in terms of the inverses of the individual block partitions L_{state}, L_u, and L_s, as shown in Equation (2.25):

$$L^{-1} = \operatorname{blkdiag}\{L_{\text{state}}^{-1}, L_u^{-1}, L_s^{-1}\} \tag{2.25}$$

Given two nonsingular SPD matrices A and B, it can be shown that $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ [41, Corollary 10]. Therefore,

$$L_{\text{state}}^{-1} = (R \otimes I_{n_x N})^{-1} = R^{-1} \otimes I_{n_x N} \tag{2.26}$$

where

$$R^{-1} = \frac{1}{\sqrt{w_{\text{tr}} + w_{\text{vse}}}} \begin{pmatrix} 1 & w \\ 0 & w + \frac{1}{w} \end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{2.27}$$

$$w = \sqrt{\frac{w_{\text{vse}}}{w_{\text{tr}}}} \tag{2.28}$$

and

$$L_u^{-1} = \frac{1}{\sqrt{w_{\text{tr}}}} I_{n_u N} \text{ and } L_s^{-1} = \frac{1}{\sqrt{w_{\text{tr}_s}}} \tag{2.29}$$

For embedded applications, if the problem data does not need to change, i.e., if Q is fixed, the preconditioning parameters can be computed offline and stored onboard. However, even in cases where the problem data may change, note that effectively, the only matrix factorization/inversion operations required in the proposed preconditioning procedure are one Cholesky decomposition of a 2×2 matrix (W_{state}) and one inversion of a 2×2 upper triangular matrix (R), both of which have closed-form expressions. An efficient implementation of the customized hypersphere preconditioning procedure with no explicit matrix factorizations/inversions is documented in Algorithm 5. Note that all the transformations in the algorithm only involve scaling the problem data by scalars (with the exception of one vector addition operation). Further, these scaling factors only depend on the objective function weights, which are independent of the problem size, thus making the algorithm suitable for large-scale problems as well.

Further, given the optimal control structure, row normalization of the constraint matrix can also be customized, as shown in Algorithm 5.

Algorithm 5 Customized Hypersphere Preconditioning

Inputs: w_{vse}, w_{tr}, w_{tr_s}, q_x, q_ξ, q_u, q_s, $A^-_{[1:N-1]}$, $E^-_{[1:N-1]}$, $B^-_{[1:N-1]}$, $B^+_{[1:N-1]}$, $S_{[1:N-1]}$, $d_{[1:N-1]}$, $\mathbb{D}_\xi$, $\mathbb{D}_u$, $\mathbb{D}_s$, λ

Require: w_{vse}, w_{tr}, $w_{\text{tr}_s} > 0$ — R: Equation (2.23); R^{-1}: Equation (2.27)

$l_{x_1} \leftarrow \sqrt{w_{\text{tr}} + w_{\text{vse}}}$ ▷ $R_{\{1,1\}}$
$l_{x_2} \leftarrow \frac{-w_{\text{vse}}}{l_{x_1}}$ ▷ $R_{\{1,2\}}$
$l_\xi \leftarrow \frac{\sqrt{w_{\text{tr}}\, w_{\text{vse}}}}{l_{x_1}}$ ▷ $R_{\{2,2\}}$
$l_{x_{1_{\text{inv}}}} \leftarrow \frac{1}{l_{x_1}}$ ▷ $R^{-1}_{\{1,1\}} = \frac{1}{R_{\{1,1\}}}$
$l_{x_{2_{\text{inv}}}} \leftarrow \frac{-l_{x_2}}{l_{x_1} l_\xi}$ ▷ $R^{-1}_{\{1,2\}} = \frac{-R_{\{1,2\}}}{R_{\{1,1\}} R_{\{2,2\}}}$
$l_{\xi_{\text{inv}}} \leftarrow \frac{1}{l_\xi}$ ▷ $R^{-1}_{\{2,2\}} = \frac{1}{R_{\{2,2\}}}$
$l_u \leftarrow \sqrt{w_{\text{tr}}}$
$l_{u_{\text{inv}}} \leftarrow \frac{1}{l_u}$
$l_s \leftarrow \sqrt{w_{\text{tr}_s}}$
$l_{s_{\text{inv}}} \leftarrow \frac{1}{l_s}$
$\hat{\mathbb{D}}_\xi \leftarrow l_\xi\, \mathbb{D}_\xi$
$\hat{\mathbb{D}}_u \leftarrow l_u\, \mathbb{D}_u$
$\hat{\mathbb{D}}_s \leftarrow l_s\, \mathbb{D}_s$
$\hat{A}^-_{[1:N-1]} \leftarrow l_{x_{1_{\text{inv}}}} A_{[1:N-1]}$
$\hat{A}^+_{[1:N-1]} \leftarrow -l_{x_{1_{\text{inv}}}} I_{n_x}$
$\hat{E}^-_{[1:N-1]} \leftarrow l_{x_{2_{\text{inv}}}} A_{[1:N-1]}$
$\hat{E}^+_{[1:N-1]} \leftarrow -l_{x_{2_{\text{inv}}}} I_{n_x}$
$\hat{B}^-_{[1:N-1]} \leftarrow l_{u_{\text{inv}}} B^-_{[1:N-1]}$
$\hat{B}^+_{[1:N-1]} \leftarrow l_{u_{\text{inv}}} B^+_{[1:N-1]}$
$\hat{S}_{[1:N-1]} \leftarrow l_{s_{\text{inv}}} S_{[1:N-1]}$
for $k = 1 : N-1$ **do** ▷ row normalization
  for $l = 1 : n_x$ **do**
    $r \leftarrow \left[\hat{A}^-_{[k]}[l,:], \hat{A}^+_{[k]}[l,:], \hat{E}^-_{[k]}[l,:], \hat{E}^+_{[k]}[l,:], \hat{B}^-_{[k]}[l,:], \hat{B}^+_{[k]}[l,:], \hat{S}_{[k]}[l,:]\right]$
    $n \leftarrow \|r\|_\infty$
    $r \leftarrow \frac{1}{n} r$
    $\hat{d}[l,:] \leftarrow \frac{1}{n} d[l,:]$
  end for
end for
$\sigma_{\max} \leftarrow$ Algorithm 6
$\sigma_{\min} \leftarrow$ Algorithm 7
$\lambda \leftarrow \sqrt{\frac{\sigma_{\min}}{2}}$
$\hat{q}_x \leftarrow \lambda\, l_{x_{1_{\text{inv}}}} q_x$
$\hat{q}_\xi \leftarrow \lambda\, (l_{x_{2_{\text{inv}}}} q_x + l_{\xi_{\text{inv}}} q_\xi)$
$\hat{q}_u \leftarrow \lambda\, l_{u_{\text{inv}}} q_u$
$\hat{q}_s \leftarrow \lambda\, l_{s_{\text{inv}}} q_s$

Return: $\hat{q}_x$, $\hat{q}_\xi$, $\hat{q}_u$, $\hat{q}_s$, $\hat{A}^-_{[1:N-1]}$, $\hat{A}^+_{[1:N-1]}$, $\hat{E}^-_{[1:N-1]}$, $\hat{E}^+_{[1:N-1]}$, $\hat{B}^-_{[1:N-1]}$, $\hat{B}^+_{[1:N-1]}$, $\hat{S}_{[1:N-1]}$, $\hat{d}_{[1:N-1]}$, $\hat{\mathbb{D}}_\xi$, $\hat{\mathbb{D}}_u$, $\hat{\mathbb{D}}_s$, l_{x_1}, l_{x_2}, l_ξ, l_u, l_s, $l_{x_{1_{\text{inv}}}}$, $l_{x_{2_{\text{inv}}}}$, $l_{\xi_{\text{inv}}}$, $l_{u_{\text{inv}}}$, $l_{s_{\text{inv}}}$, $\sigma_{\max}$

2.4.2 Customized Power Iteration Method

Algorithm 6 Customized Power Iteration Method

Inputs: $\hat{A}^{-}_{[1:N-1]}$, $\hat{A}^{+}_{[1:N-1]}$, $\hat{E}^{-}_{[1:N-1]}$, $\hat{E}^{+}_{[1:N-1]}$, $\hat{B}^{-}_{[1:N-1]}$, $\hat{B}^{+}_{[1:N-1]}$, $\hat{S}_{[1:N-1]}$,
ϵ_{abs}, ϵ_{rel}, ϵ_{buff}, $j_{\max}$,
$x_{[1:N]}$, $\xi_{[1:N]}$, $u_{[1:N]}$, s, $w_{[1:N-1]}$

Require: $\left\|x_{[1:N]}\right\|_2 > 0,\ \left\|\xi_{[1:N]}\right\|_2 > 0,\ \left\|u_{[1:N]}\right\|_2 > 0,\ s > 0$

$\sigma \leftarrow s^2$
for $k \leftarrow 1{:}N$ **do** ▷ Algorithm 2, Line 1
  $\sigma \leftarrow \sigma + \|x_k\|_2^2 + \|\xi_k\|_2^2 + \|u_k\|_2^2$
end for
$\sigma \leftarrow \sqrt{\sigma}$
for $j \leftarrow 1{:}j_{\max}$ **do**
  for $k \leftarrow 1{:}N{-}1$ **do** ▷ Algorithm 2, Line 4
    $w_k \leftarrow \frac{1}{\sigma}\left(\hat{A}^{-}_k\, x_k + \hat{A}^{+}_k\, x_{k+1} + \hat{E}^{-}_k\, \xi_k + \hat{E}^{+}_k\, \xi_{k+1} + \hat{B}^{-}_k\, u_k + \hat{B}^{+}_k\, u_{k+1} + \hat{S}_k\, s\right)$
  end for
  $x_1 \leftarrow \hat{A}_1^{-\top} w_1$
  $\xi_1 \leftarrow \hat{E}_1^{-\top} w_1$
  $u_1 \leftarrow \hat{B}_1^{-\top} w_1$
  $s \leftarrow \hat{S}_1^{\top} w_1$
  for $k \leftarrow 2{:}N{-}1$ **do** ▷ Algorithm 2, Line 3
    $x_k \leftarrow \hat{A}_k^{-\top} w_k + \hat{A}_{k-1}^{+\top} w_{k-1}$
    $\xi_k \leftarrow \hat{E}_k^{-\top} w_k + \hat{E}_{k-1}^{+\top} w_{k-1}$
    $u_k \leftarrow \hat{B}_k^{-\top} w_k + \hat{B}_{k-1}^{+\top} w_{k-1}$
    $s \leftarrow s + \hat{S}_k^{\top} w_k$
  end for
  $x_N \leftarrow \hat{A}_{N-1}^{+\top} w_{N-1}$
  $\xi_N \leftarrow \hat{E}_{N-1}^{+\top} w_{N-1}$
  $u_N \leftarrow \hat{B}_{N-1}^{+\top} w_{N-1}$
  $\sigma^\star \leftarrow s^2$
  for $k \leftarrow 1{:}N$ **do** ▷ Algorithm 2, Line 5
    $\sigma^\star \leftarrow \sigma^\star + \|x_k\|_2^2 + \|\xi_k\|_2^2 + \|u_k\|_2^2$
  end for
  $\sigma^\star \leftarrow \sqrt{\sigma^\star}$
  if $|\sigma^\star - \sigma| \le \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\{\sigma^\star, \sigma\}$ **then** ▷ stopping criterion
    break
  else if $j < j_{\max}$ **then**
    $\sigma \leftarrow \sigma^\star$
  end if
end for
$\sigma \leftarrow (1 + \epsilon_{\text{buff}})\, \sigma^\star$ ▷ buffer the (under) estimated maximum singular value

Return: σ ▷ $\approx \max \operatorname{spec} \hat{H}^\top \hat{H} = \sigma_{\max}(\hat{H}^\top \hat{H}) = \|\hat{H}\|_2^2$

2.4.3 Customized Shifted Power Iteration Method

Algorithm 7 Customized Shifted Power Iteration Method

Inputs: $\hat{A}^{-}_{[1:N-1]}$, $\hat{A}^{+}_{[1:N-1]}$, $\hat{E}^{-}_{[1:N-1]}$, $\hat{E}^{+}_{[1:N-1]}$, $\hat{B}^{-}_{[1:N-1]}$, $\hat{B}^{+}_{[1:N-1]}$, $\hat{S}_{[1:N-1]}$,
ϵ_{abs}, ϵ_{rel}, ϵ_{buff}, $j_{\max}$, $\sigma_{\max}$
$x_{[1:N]}$, $\xi_{[1:N]}$, $u_{[1:N]}$, s, $w_{[1:N-1]}$

Require: $\|w_{[1:N-1]}\|_2 > 0$

$\tilde{\sigma} \leftarrow \|w_{[1:N-1]}\|_2$
for $j \leftarrow 1 : j_{\max}$ **do**
  $x_1 \leftarrow \hat{A}_1^{-\top} w_1$
  $\xi_1 \leftarrow \hat{E}_1^{-\top} w_1$
  $u_1 \leftarrow \hat{B}_1^{-\top} w_1$
  $s \leftarrow \hat{S}_1^{\top} w_1$
  for $k \leftarrow 2 : N-1$ **do** ▷ Algorithm 10, Line 3
    $x_k \leftarrow \hat{A}_k^{-\top} w_k + \hat{A}_{k-1}^{+\top} w_{k-1}$
    $\xi_k \leftarrow \hat{E}_k^{-\top} w_k + \hat{E}_{k-1}^{+\top} w_{k-1}$
    $u_k \leftarrow \hat{B}_k^{-\top} w_k + \hat{B}_{k-1}^{+\top} w_{k-1}$
    $s \leftarrow s + \hat{S}_k^{\top} w_k$
  end for
  $x_N \leftarrow \hat{A}_{N-1}^{+\top} w_{N-1}$
  $\xi_N \leftarrow \hat{E}_{N-1}^{+\top} w_{N-1}$
  $u_N \leftarrow \hat{B}_{N-1}^{+\top} w_{N-1}$
  for $k \leftarrow 1 : N-1$ **do** ▷ Algorithm 10, Line 4
    $w_k \leftarrow \frac{1}{\tilde{\sigma}} \left(\hat{A}_k^{-} x_k + \hat{A}_k^{+} x_{k+1} + \hat{E}_k^{-} \xi_k + \hat{E}_k^{+} \xi_{k+1} + \hat{B}_k^{-} u_k + \hat{B}_k^{+} u_{k+1} + \hat{S}_k s - \sigma_{\max} w_k \right)$
  end for
  $\tilde{\sigma}^{\star} \leftarrow \|w_{[1:N-1]}\|_2$ ▷ Algorithm 10, Line 5
  if $|\tilde{\sigma}^{\star} - \tilde{\sigma}| \le \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\{\tilde{\sigma}^{\star}, \tilde{\sigma}\}$ **then** ▷ stopping criterion
    break
  else if $j < j_{\max}$ **then**
    $\tilde{\sigma} \leftarrow \tilde{\sigma}^{\star}$
  end if
end for
$\sigma_{\min} \leftarrow (1 - \epsilon_{\text{buff}}) (\sigma_{\max} - \tilde{\sigma}^{\star})$ ▷ buffer the (over) estimated minimum singular value

Return: $\sigma_{\min}$ ▷ $\approx \min \operatorname{spec} \hat{H}\,\hat{H}^{\top} = \sigma_{\min}(\hat{H}\,\hat{H}^{\top})$

2.4.4 Customized PIPG

Algorithm 8 $\text{PIPG}_{\text{custom}}$

Inputs: $q_{x_{[1:N]}}$, $q_{\xi_{[1:N]}}$, $q_{u_{[1:N]}}$, q_s,
$\hat{A}^-_{[1:N-1]}$, $\hat{A}^+_{[1:N-1]}$, $\hat{E}^-_{[1:N-1]}$, $\hat{E}^+_{[1:N-1]}$, $\hat{B}^-_{[1:N-1]}$, $\hat{B}^+_{[1:N-1]}$, $\hat{d}_{[2:N]}$, $\hat{\bar{x}}_{[1:N]}$, $\hat{\bar{u}}_{[1:N]}$, $\hat{\bar{s}}$,
$\hat{\mathbb{D}}_{x_1}$, $\hat{\mathbb{D}}_{\xi_{[2:N]}}$, $\hat{\mathbb{D}}_{u_{[1:N]}}$, $\hat{\mathbb{D}}_s$,
l_{x_1}, l_{x_2}, l_ξ, l_u, l_s, $l_{x_{1_{\text{inv}}}}$, $l_{x_{2_{\text{inv}}}}$, $l_{\xi_{\text{inv}}}$, $l_{u_{\text{inv}}}$, $l_{s_{\text{inv}}}$,
λ, σ, ω, ρ, ϵ_{abs}, ϵ_{rel}, j_{check}, $j_{\max}$,
$\Delta\hat{x}^\star_{[1:N]}$, $\Delta\hat{\xi}^\star_{[1:N]}$, $\Delta\hat{u}^\star_{[1:N]}$, $\Delta\hat{s}^\star$, $w^\star_{[1:N-1]}$ ▷ warm start

$\Delta x^1_{\zeta_{[1:N]}} \leftarrow l_{x_1}\,\Delta\hat{x}^\star_{[1:N]} + l_{x_2}\,\Delta\hat{\xi}^\star_{[1:N]}$ ▷ initialize primal variables
$\Delta\xi^1_{\zeta_{[1:N]}} \leftarrow l_\xi\,\Delta\hat{\xi}^\star_{[1:N]}$
$\Delta u^1_{\zeta_{[1:N]}} \leftarrow l_u\,\Delta\hat{u}^\star_{[1:N]}$
$\Delta s^1_\zeta \leftarrow l_s\,\Delta\hat{s}^\star$
$\eta^1_{[1:N-1]} \leftarrow w^\star_{[1:N-1]}$ ▷ initialize dual variable
$\alpha \leftarrow \frac{2}{\lambda+\sqrt{\lambda^2+4\omega\sigma}}$ ▷ step-sizes
$\beta \leftarrow \omega\alpha$
for $j \leftarrow 1 : j_{\max}$ **do**
  $\Delta\hat{x}^{j+1}_1 \leftarrow \pi_{\hat{\mathbb{D}}_{x_1}}[\Delta x^j_{\zeta_1} - \alpha\,(\lambda\,\Delta x^j_{\zeta_1} + \lambda\,q_{x_1} + \hat{A}_1^{-\top}\eta^j_1) + \hat{\bar{x}}^j_1] - \hat{\bar{x}}^j_1$
  $\Delta\hat{\xi}^{j+1}_1 \leftarrow 0$
  $\Delta\hat{u}^{j+1}_1 \leftarrow \pi_{\hat{\mathbb{D}}_{u_1}}[\Delta u^j_{\zeta_1} - \alpha\,(\lambda\,\Delta u^j_{\zeta_1} + \lambda\,q_{u_1} + \hat{B}_1^{-\top}\eta^j_1) + \hat{\bar{u}}^j_1] - \hat{\bar{u}}^j_1$
  $\Delta\mathcal{S} \leftarrow \hat{S}_1^\top\eta^j_1$
  for $k \leftarrow 2 : N-1$ **do** ▷ projected gradient step
    $\Delta\hat{x}^{j+1}_k \leftarrow \Delta x^j_{\zeta_k} - \alpha\,(\lambda\,\Delta x^j_{\zeta_k} + \lambda\,q_{x_k} + \hat{A}_k^{-\top}\eta^j_k + \hat{A}^{+\top}_{k-1}\eta^j_{k-1})$
    $\Delta\hat{\xi}^{j+1}_k \leftarrow \pi_{\hat{\mathbb{D}}_{\xi_k}}[\Delta\xi^j_{\zeta_k} - \alpha\,(\lambda\,\Delta\xi^j_{\zeta_k} + \lambda\,q_{\xi_k} + \hat{E}_k^{-\top}\eta^j_k + \hat{E}^{+\top}_{k-1}\eta^j_{k-1}) + \hat{\bar{x}}^j_k] - \hat{\bar{x}}^j_k$
    $\Delta\hat{u}^{j+1}_k \leftarrow \pi_{\hat{\mathbb{D}}_{u_k}}[\Delta u^j_{\zeta_k} - \alpha\,(\lambda\,\Delta u^j_{\zeta_k} + \lambda\,q_{u_k} + \hat{B}_k^{-\top}\eta^j_k + \hat{B}^{+\top}_{k-1}\eta^j_{k-1}) + \hat{\bar{u}}^j_k] - \hat{\bar{u}}^j_k$
    $\Delta\mathcal{S} \leftarrow \Delta\mathcal{S} + \hat{S}_k^\top\eta^j_k$
  end for
  $\Delta\hat{x}^{j+1}_N \leftarrow \Delta x^j_{\zeta_N} - \alpha\,(\lambda\,\Delta x^j_{\zeta_N} + \lambda\,q_{x_N} + \hat{A}^{+\top}_{N-1}\eta^j_{N-1})$
  $\Delta\hat{\xi}^{j+1}_N \leftarrow \pi_{\hat{\mathbb{D}}_{\xi_N}}[\Delta\xi^j_{\zeta_N} - \alpha\,(\lambda\,\Delta\xi^j_{\zeta_N} + \lambda\,q_{\xi_N} + \hat{E}^{+\top}_{N-1}\eta^j_{N-1}) + \hat{\bar{x}}^j_N] - \hat{\bar{x}}^j_N$
  $\Delta\hat{u}^{j+1}_N \leftarrow \pi_{\hat{\mathbb{D}}_{u_N}}[\Delta u^j_{\zeta_N} - \alpha\,(\lambda\,\Delta u^j_{\zeta_N} + \lambda\,q_{u_N} + \hat{B}^{+\top}_{N-1}\eta^j_{N-1}) + \hat{\bar{u}}^j_N] - \hat{\bar{u}}^j_N$
  $\Delta\hat{s}^{j+1} \leftarrow \pi_{\hat{\mathbb{D}}_s}[\Delta s^j_\zeta - \alpha\,(\lambda\,\Delta s^j_\zeta + \lambda\,q_s + \Delta\mathcal{S}) + \hat{\bar{s}}^j] - \hat{\bar{s}}^j$
  for $k \leftarrow 1 : N-1$ **do** ▷ PI feedback of affine equality constraint violation
    $w^{j+1}_k \leftarrow \eta^j_k + \beta\,(\hat{A}_k^-\,(2\Delta\hat{x}^{j+1}_k - \Delta x^j_{\zeta_k}) + \hat{A}_k^+\,(2\Delta\hat{x}^{j+1}_{k+1} - \Delta x^j_{\zeta_{k+1}}) + \hat{E}_k^-\,(2\Delta\hat{\xi}^{j+1}_k - \Delta\xi^j_{\zeta_k}) + \hat{E}_k^+\,(2\Delta\hat{\xi}^{j+1}_{k+1} - \Delta\xi^j_{\zeta_{k+1}})$
    $\qquad + \hat{B}_k^-\,(2\Delta\hat{u}^{j+1}_k - \Delta u^j_{\zeta_k}) + \hat{B}_k^+\,(2\Delta\hat{u}^{j+1}_{k+1} - \Delta u^j_{\zeta_{k+1}}) + \hat{S}_k\,(2\Delta\hat{s}^{j+1} - \Delta s^j_\zeta) + \hat{d}_{k+1})$
  end for
  $\Delta x^{j+1}_{\zeta_{[1:N]}} \leftarrow (1-\rho)\,\Delta x^j_{\zeta_{[1:N]}} + \rho\,\Delta\hat{x}^{j+1}_{[1:N]}$ ▷ extrapolate primal variables
  $\Delta\xi^{j+1}_{\zeta_{[1:N]}} \leftarrow (1-\rho)\,\Delta\xi^j_{\zeta_{[1:N]}} + \rho\,\Delta\hat{\xi}^{j+1}_{[1:N]}$
  $\Delta u^{j+1}_{\zeta_{[1:N]}} \leftarrow (1-\rho)\,\Delta u^j_{\zeta_{[1:N]}} + \rho\,\Delta\hat{u}^{j+1}_{[1:N]}$
  $\eta^{j+1}_{[1:N-1]} \leftarrow (1-\rho)\,\eta^j_{[1:N-1]} + \rho\,w^{j+1}_{[1:N-1]}$ ▷ extrapolate dual variables
  if $j \bmod j_{\text{check}} = 0$ **then** ▷ check stopping criterion every j_{check} iterations
    TERMINATE $\leftarrow$ STOPPING($\Delta\hat{x}^{j+1}_{[1:N]}$, $\Delta\hat{\xi}^{j+1}_{[1:N]}$, $\Delta\hat{u}^{j+1}_{[1:N]}$, $\Delta\hat{s}^{j+1}$, $w^{j+1}_{[1:N-1]}$,
    $\qquad\qquad \Delta\hat{x}^j_{[1:N]}$, $\Delta\hat{\xi}^j_{[1:N]}$, $\Delta\hat{u}^j_{[1:N]}$, $\Delta\hat{s}^j$, $w^j_{[1:N-1]}$, ϵ_{abs}, ϵ_{rel})
    if TERMINATE = TRUE **then** ▷ stopping criterion
      break
    end if
  end if
end for
$\Delta\hat{x}^\star_{[1:N]} \leftarrow l_{x_{1_{\text{inv}}}}\,\Delta\hat{x}^{j+1}_{[1:N]} + l_{x_{2_{\text{inv}}}}\,\Delta\hat{\xi}^{j+1}_{[1:N]}$ ▷ update primal variables
$\Delta\hat{\xi}^\star_{[1:N]} \leftarrow l_{\xi_{\text{inv}}}\,\Delta\hat{\xi}^{j+1}_{[1:N]}$
$\Delta\hat{u}^\star_{[1:N]} \leftarrow l_{u_{\text{inv}}}\,\Delta\hat{u}^{j+1}_{[1:N]}$
$\Delta\hat{s}^\star \leftarrow l_{s_{\text{inv}}}\,\Delta\hat{s}^{j+1}$
$w^\star_{[1:N-1]} \leftarrow w^{j+1}_{[1:N-1]}$ ▷ update dual variable

Return: $\Delta\hat{x}^\star_{[1:N]}$, $\Delta\hat{\xi}^\star_{[1:N]}$, $\Delta\hat{u}^\star_{[1:N]}$, $\Delta\hat{s}^\star$, $w^\star_{[1:N-1]}$

Algorithm 9 Stopping Criterion Evaluation:

STOPPING($\Delta\hat{x}^{j+1}_{[1:N]}$, $\Delta\hat{\xi}^{j+1}_{[1:N]}$, $\Delta\hat{u}^{j+1}_{[1:N]}$, $\Delta\hat{s}^{j+1}$, $w^{j+1}_{[1:N-1]}$, $\Delta\hat{x}^{j}_{[1:N]}$, $\Delta\hat{\xi}^{j}_{[1:N]}$, $\Delta\hat{u}^{j}_{[1:N]}$, $\Delta\hat{s}^{j}$, $w^{j}_{[1:N-1]}$, ϵ_{abs}, ϵ_{rel})

Inputs: $\Delta\hat{x}^{j+1}_{[1:N]}$, $\Delta\hat{\xi}^{j+1}_{[1:N]}$, $\Delta\hat{u}^{j+1}_{[1:N]}$, $\Delta\hat{s}^{j+1}$, $w^{j+1}_{[1:N-1]}$, $\Delta\hat{x}^{j}_{[1:N]}$, $\Delta\hat{\xi}^{j}_{[1:N]}$, $\Delta\hat{u}^{j}_{[1:N]}$, $\Delta\hat{s}^{j}$, $w^{j}_{[1:N-1]}$, ϵ_{abs}, ϵ_{rel}

$z^{j+1}_{\infty} \leftarrow \max\Big\{\|\Delta\hat{x}^{j+1}_{[1:N]}\|_\infty, \|\Delta\hat{\xi}^{j+1}_{[1:N]}\|_\infty, \|\Delta\hat{u}^{j+1}_{[1:N]}\|_\infty, |\Delta\hat{s}^{j+1}|\Big\}$

$z^{j}_{\infty} \leftarrow \max\Big\{\|\Delta\hat{x}^{j}_{[1:N]}\|_\infty, \|\Delta\hat{\xi}^{j}_{[1:N]}\|_\infty, \|\Delta\hat{u}^{j}_{[1:N]}\|_\infty, |\Delta\hat{s}^{j}|\Big\}$

$z^{\Delta j}_{\infty} \leftarrow \max\Big\{\|\Delta\hat{x}^{j+1}_{[1:N]} - \Delta\hat{x}^{j}_{[1:N]}\|_\infty, \|\Delta\hat{\xi}^{j+1}_{[1:N]} - \Delta\hat{\xi}^{j}_{[1:N]}\|_\infty, \|\Delta\hat{u}^{j+1}_{[1:N]} - \Delta\hat{u}^{j}_{[1:N]}\|_\infty, |\Delta\hat{s}^{j+1} - \Delta\hat{s}^{j}|\Big\}$

$r^{j+1}_{\infty} \leftarrow \|w^{j+1}_{[1:N-1]}\|_\infty$

$r^{j}_{\infty} \leftarrow \|w^{j}_{[1:N-1]}\|_\infty$

$r^{\Delta j}_{\infty} \leftarrow \|w^{j+1}_{[1:N-1]} - w^{j}_{[1:N-1]}\|_\infty$

if $z^{\Delta j}_{\infty} \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\Big\{z^{j+1}_{\infty}, z^{j}_{\infty}\Big\}$ **and** $r^{\Delta j}_{\infty} \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\Big\{r^{j+1}_{\infty}, r^{j}_{\infty}\Big\}$ **then**

    TERMINATE $\leftarrow$ TRUE

else

    TERMINATE $\leftarrow$ FALSE

end if

Return: TERMINATE

Chapter 3

OPTIMAL PRECONDITIONING FOR CONIC OPTIMIZATION

Consider the following QCP [221] template:

$$\underset{\xi}{\text{minimize}} \quad \frac{1}{2}\xi^\top P\,\xi + p^\top \xi \tag{3.1a}$$

$$\text{subject to} \quad G\,\xi - g \in \mathbb{L} \tag{3.1b}$$

$$\xi \in \mathbb{E} \tag{3.1c}$$

where $\mathbb{L}$ is a closed convex cone, $\mathbb{E}$ is a closed convex set, and P is a positive definite matrix ($P \succ 0$), i.e., the objective function is strongly convex. The cone $\mathbb{L}$ is a Cartesian product of closed convex cones, such as the zero cone, the nonnegative orthant cone, second-order cones (SOCs), and the cone of positive semidefinite (PSD) matrices. The set $\mathbb{E}$ is a Cartesian product of separable closed convex sets—such as halfspaces, boxes, ℓ_2-norm balls, SOCs, etc—that possess closed-form (or easy-to-evaluate) projection operations [16, 15].

The matrix P is assumed to possess a structure that complies with the following requirements:

$$z \in \mathbb{D} \iff \xi \in \mathbb{E} \tag{3.2a}$$

$$(H\,z - h \in \mathbb{K}) \iff (G\,\xi - g \in \mathbb{L}) \tag{3.2b}$$

where R is the upper-triangular Cholesky factor of P, i.e., $R^\top R = P$, $z := R\,\xi$, $\mathbb{D} := R\,\mathbb{E}$, $\begin{bmatrix} H & h \end{bmatrix} := E \begin{bmatrix} G\,R^{-1} & g \end{bmatrix}$, and $\mathbb{K} := E\,\mathbb{L}$, for some positive definite diagonal matrix, E. Here, $S\,\mathbb{F} := \{z \mid S^{-1}z \in \mathbb{F}\}$, where $S \in \mathbb{R}^{n\times n}$ and $\mathbb{F}$ is a convex set or cone, i.e., $z \in$

$S\,\mathbb{F} \iff S^{-1}z \in \mathbb{F}$, and $\begin{bmatrix} A & b \end{bmatrix}$ denotes concatentation, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$.

Further, to enable online *factorization-free* implementations with the elements of P changing between successive calls to the solver, P would need to be diagonal or have a specialized block-diagonal structure—such as the one described in [102]—such that its Cholesky factorization is representable in closed-form. In cases wherein the Hessian of the objective function is static and only the constraint matrix is dynamic, the Cholesky factorization of P (regardless of whether it is representable in closed-form) can be performed offline, such that the online component is factorization-free.

In practice, the template given by Problem 3.1 accounts for the general class of QCPs—including, but not limited to, quadratic programs (QPs), second-order cone programs (SOCPs), and semidefinite programs (SDPs)—with strongly convex quadratic objective functions, subject to the restrictions imposed on P.

This chapter is organized as follows: §3.1 describes the three-step preconditioning procedure that we propose and §3.2 provides a comparison of our method against two state-of-the-art preconditioners, modified Ruiz equilibration [201] and the QR preconditioner [52].

3.1 Preconditioning

3.1.1 Objective Function: Hypersphere Preconditioning

The objective function can be preconditioned using the hypersphere preconditioner [102], which uses the Cholesky factorization of P, i.e., $R^\top R = P$, and scales the objective function with a scalar parameter $\lambda > 0$, to transform Problem 3.1 into the following problem:

$$\begin{aligned} \underset{z}{\text{minimize}} \quad & \frac{\lambda}{2} z^\top z + \lambda\, q^\top z && \text{(3.3a)} \\ \text{subject to} \quad & \hat{G}\, z - g \in \mathbb{L} && \text{(3.3b)} \\ & z \in \mathbb{D} && \text{(3.3c)} \end{aligned}$$

where $z := R\,\xi$, $q := R^{-\top} p$, $\hat{G} := G\,R^{-1}$, and $\mathbb{D} := R\,\mathbb{E}$. This preconditioner is optimal in the sense of minimizing the condition number of the Hessian of the objective function, i.e., the condition number of the resulting objective function matrix, $\lambda\,I$, is unity. Further, its level sets are hyperspheres; hence the name. For ease of reference, we overload the term *hypersphere preconditioner* to refer to the entire three-step preconditioning procedure described herewithin.

3.1.2 Constraints: Block Row-Normalization

The constraint matrix, $\hat{G}$, can be preconditioned using row normalization, i.e., dividing each row of Equation (3.3b) by the norm of the corresponding row of $\hat{G}$, such that the rows of the preconditioned matrix have unit norms. In practice, this procedure often helps reduce the condition number of the constraint matrix, with the added benefit of not requiring transformation of the primal variable, z (since the constraint matrix is only left-multiplied).

Further, row normalization should be carried out in the *block* sense, i.e., by normalizing the rows of $\hat{G}$ such that the requirement given by Equation (3.2b) holds. See [157, Section 5] for a description of this requirement. If $\mathbb{L}$ only contains linear (in)equalities, such as the zero cone or the nonnegative orthant cone, then the diagonal matrix, $E \succ 0$, in $\begin{bmatrix} H & h \end{bmatrix} = E \begin{bmatrix} \hat{G} & g \end{bmatrix}$, can be left unrestricted. Otherwise, for each separable[†] convex set in $\mathbb{L}$, only the maximum magnitude element—among the rows of $\hat{G}$ that correspond to a separable set in $\mathbb{L}$—is considered in the corresponding rows of E.

To illustrate this, consider the following SOC constraint, where $A \in \mathbb{R}^{2\times 2} \succ 0$ (diagonal), $x \in \mathbb{R}^2$, and $t \in \mathbb{R}$:

$$\|A\,x\|_2 \leq t, \text{ i.e., } \tilde{A}\,\tilde{x} \in \mathbb{L}^2_{\text{SOC}} \tag{3.4}$$

where $\|\cdot\|_2$ is the Euclidean norm, $\tilde{A} := \text{blkdiag}\{A,\, 1\}$, $\tilde{x} := (x,\, t)$, and $\mathbb{L}^2_{\text{SOC}} := \{(z_1,\, z_2,\, z_3) \in$

[†]The separable convex sets in $\mathbb{L}$ form a partition, i.e., each set corresponds to unique components of the stacked decision variable vector, z.

$\mathbb{R}^3 \mid \|(z_1, z_2)\|_2 \leq z_3\}$ is the second-order cone. Here, $(a, b) \in \mathbb{R}^{m+n}$, where $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, denotes vector concatenation. The requirement given by Equation (3.2b) implies that a suitable choice for E_{soc}, given $E_{\text{soc}}\,\tilde{A}\,\tilde{x} \in E_{\text{soc}}\,\mathbb{L}^2_{\text{soc}}$, is $c\,I_3$, where $c \in \mathbb{R}_{++}$ and I_3 is the identity matrix in $\mathbb{R}^{3\times 3}$. In block row-normalization, $c := \frac{1}{\max\{|A_{11}|, |A_{22}|, 1\}}$, where A_{ij} is the element of A in the i^{th} row and the j^{th} column.

3.1.3 Optimal Objective Function Scaling Factor

The preconditioned problem is given as follows:

$$\underset{z}{\text{minimize}} \quad \frac{\lambda}{2} z^\top z + \lambda\, q^\top z \tag{3.5a}$$

$$\text{subject to} \quad H\,z - h \in \mathbb{K} \tag{3.5b}$$

$$z \in \mathbb{D} \tag{3.5c}$$

where $\lambda > 0$ is the objective function scaling factor, and Equation (3.5b) is obtained by block row-normalizing Equation (3.3b), i.e., $\begin{bmatrix} H & h \end{bmatrix} = E \begin{bmatrix} \hat{G} & g \end{bmatrix}$, $E \succ 0$ being the diagonal matrix that performs block row-normalization and $\mathbb{K} := E\,\mathbb{L}$ being the corresponding cone.

The following lemma and corollary relate to the KKT matrix of Problem 3.5, given by $K := \begin{bmatrix} \lambda\, I & H^\top \\ H & 0 \end{bmatrix}$.

Lemma 1. *For a given $\lambda > 0$ and $H \in \mathbb{R}^{m\times n}$, where $n > m$ and $\operatorname{rank} H = m$, the spectrum of K is:*

$$\operatorname{spec} K = \left\{ \lambda,\ \tfrac{\lambda \pm \sqrt{\lambda^2 + 4\,\sigma_1}}{2},\ \ldots,\ \tfrac{\lambda \pm \sqrt{\lambda^2 + 4\,\sigma_m}}{2} \right\} \tag{3.6}$$

where $\sigma_1, \ldots, \sigma_m$ are the squares of the singular values of H. The eigenvalue λ is repeated $n - m$ times, and there are a total of $n + m$ eigenvalues.

Proof. Since $\operatorname{rank} H = m$ and $\lambda\, I \succ 0$, the matrix K is nonsingular [40, Section 10.1.1]. Consider a nontrivial eigenvector of K, $p := (u, v) \neq 0$, where $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$. Let

θ be the corresponding eigenvalue. Then, we have $K\,p = \theta\,p$, which leads to the following equations:

$$\lambda\,u + H^\top v = \theta\,u \implies H^\top v = (\theta - \lambda)\,u \tag{3.7}$$

$$H\,u = \theta\,v \tag{3.8}$$

Note that $H^\top$ defines a one-to-one transformation, since it is full column rank. Suppose $v = 0$. This implies that $u \in \ker H$ and $(\theta - \lambda)\,u = 0$, which in turn either implies that $\theta = \lambda$ or that $u = 0$. However, since u cannot be trivial, we have $\theta = \lambda$, i.e., $\theta = \lambda$ is an eigenvalue of K, the corresponding eigenvector being $(u, 0)$, where $u \in \ker H$. Further, there are $n - m$ eigenvectors for $\theta = \lambda$, since $\dim \ker H = n - m$.

Now, multiplying Equation (3.7) on the left by H and substituting Equation (3.8), we get:

$$H\,H^\top v = (\theta - \lambda)\,H\,u = \theta\,(\theta - \lambda)\,v \tag{3.9}$$

Since $H\,H^\top$ is symmetric positive definite, its singular values are equal to its eigenvalues (all positive). Consider the eigenvalues of $H\,H^\top$, $\sigma_k > 0$, $k = 1, \ldots, m$. Note that these m eigenvalues of $H\,H^\top$ have m distinct corresponding eigenvectors. Now, from Equation (3.9), for each σ_k, we have:

$$\theta\,(\theta - \lambda) = \sigma_k \implies \theta^2 - \lambda\,\theta - \sigma_k = 0 \tag{3.10}$$

Solving for θ, we get $2\,m$ eigenvalues of K:

$$\theta_\pm(\lambda, \sigma_k) := \tfrac{\lambda \pm \sqrt{\lambda^2 + 4\,\sigma_k}}{2}, \quad k = 1, \ldots, m \tag{3.11}$$

Therefore, the spectrum of K is given by Equation (3.6), where the algebraic multiplicity of λ is $n - m$, and the total number of eigenvalues of K is $n + m$. $\square$

Corollary 1. *For a given* $\lambda > 0$ *and* $H \in \mathbb{R}^{m\times n}$*, where* $n > m$ *and* rank $H = m$*, the condition number of* K *can be given by:*

$$\kappa(\lambda) := \frac{\frac{\lambda+\sqrt{\lambda^2+4\,\sigma_{\max}}}{2}}{\min\left\{\lambda, \frac{\sqrt{\lambda^2+4\,\sigma_{\min}}-\lambda}{2}\right\}} \tag{3.12}$$

where $\sigma_{\max}$ *and* $\sigma_{\min}$ *are the squares of the largest and smallest singular values of* H*, respectively.*

Proof. Since $\theta_+(\lambda, \sigma_i) \geq \max\{\lambda, |\theta_-(\lambda, \sigma_j)|\}$, $1 \leq i,\, j \leq m$, the condition number of K as a function of λ can be given by $\kappa(\lambda) = \frac{\delta_{\max}(\lambda)}{\delta_{\min}(\lambda)}$, where:

$$\delta_{\max}(\lambda) := \max_k \theta_+(\lambda, \sigma_k) = \max_k \frac{\lambda+\sqrt{\lambda^2+4\,\sigma_k}}{2} \tag{3.13}$$

$$\delta_{\min}(\lambda) := \min\{\lambda, \min_k |\theta_-(\lambda, \sigma_k)|\} = \min\left\{\lambda, \min_k \frac{\sqrt{\lambda^2+4\,\sigma_k}-\lambda}{2}\right\} \tag{3.14}$$

The term $\theta_+(\lambda, \sigma_k)$ in Equation (3.13) attains its maximum when $\sigma_k = \sigma_{\max}$.

$$\therefore\ \delta_{\max}(\lambda) = \frac{\lambda+\sqrt{\lambda^2+4\,\sigma_{\max}}}{2} \tag{3.15}$$

The term $|\theta_-(\lambda, \sigma_k)|$ in Equation (3.14) attains its minimum when $\sigma_k = \sigma_{\min}$.

$$\therefore\ \delta_{\min}(\lambda) = \min\left\{\lambda, \frac{\sqrt{\lambda^2+4\,\sigma_{\min}}-\lambda}{2}\right\} \tag{3.16}$$

$$\therefore\ \kappa(\lambda) = \frac{\frac{\lambda+\sqrt{\lambda^2+4\,\sigma_{\max}}}{2}}{\min\left\{\lambda, \frac{\sqrt{\lambda^2+4\,\sigma_{\min}}-\lambda}{2}\right\}} \qquad \square$$

Theorem 2. *For a given $H \in \mathbb{R}^{m\times n}$, where $n > m$ and* rank $H = m$,

$$\lambda^\star := \underset{\lambda}{\operatorname{argmin}}\, \kappa(\lambda) = \sqrt{\frac{\sigma_{\min}}{2}} \tag{3.17}$$

where $\lambda > 0$, $\sigma_{\min}$ is the square of the smallest singular value of H, and $\kappa(\lambda)$ is given by Equation (3.12).

Proof. Let $f_1(\lambda) := \lambda$ and $f_2(\lambda) := \frac{\sqrt{\lambda^2+4\,\sigma_{\min}}-\lambda}{2}$. From Equation (3.16), $\delta_{\min}(\lambda) = \min\{f_1(\lambda),\, f_2(\lambda)\}$. It is clear that f_1 is a positive and strictly increasing function of λ. Now, taking the derivative of $f_2(\lambda)$ with respect to λ, we get:

$$\tfrac{df_2(\lambda)}{d\lambda} = \tfrac{1}{2}\left(\frac{\lambda}{\sqrt{\lambda^2+4\,\sigma_{\min}}} - 1\right) \tag{3.18}$$

Since $\sqrt{\lambda^2 + 4\,\sigma_{\min}} > \lambda$, we conclude that $\frac{df_2(\lambda)}{d\lambda} < 0$, and hence, f_2 is a positive and strictly decreasing function of λ.

$$\therefore\ \min\{f_1(\lambda),\, f_2(\lambda)\} = \frac{1}{\max\left\{\frac{1}{f_1(\lambda)}, \frac{1}{f_2(\lambda)}\right\}} \tag{3.19}$$

Now, let $f_0(\lambda) := \frac{\lambda+\sqrt{\lambda^2+4\,\sigma_{\max}}}{2}$. We see that $\frac{df_0(\lambda)}{d\lambda} > 0$, and hence, f_0 is a positive and strictly increasing function of λ.

$$\therefore\ \kappa(\lambda) = f_0(\lambda)\, \max\left\{\tfrac{1}{f_1(\lambda)},\, \tfrac{1}{f_2(\lambda)}\right\} = \max\left\{\tfrac{f_0(\lambda)}{f_1(\lambda)},\, \tfrac{f_0(\lambda)}{f_2(\lambda)}\right\} \tag{3.20}$$

$$\therefore\ \min_{\lambda}\, \kappa(\lambda) = \min_{\lambda}\, \max\left\{\tfrac{f_0(\lambda)}{f_1(\lambda)},\, \tfrac{f_0(\lambda)}{f_2(\lambda)}\right\} \tag{3.21}$$

Since $\frac{f_0(\lambda)}{f_1(\lambda)}$ is strictly decreasing and $\frac{f_0(\lambda)}{f_2(\lambda)}$ is strictly increasing, the minimizer of Equation (3.20) satisfies $\frac{f_0(\lambda)}{f_1(\lambda)} = \frac{f_0(\lambda)}{f_2(\lambda)}$, i.e., $f_1(\lambda) = f_2(\lambda)$ (since $f_0(\lambda)$ is positive), as depicted in Fig. 3.1. $\therefore\ \lambda^\star = \frac{\sqrt{{\lambda^\star}^2+4\,\sigma_{\min}}-\lambda^\star}{2} \implies \lambda^\star = \sqrt{\frac{\sigma_{\min}}{2}}$. □

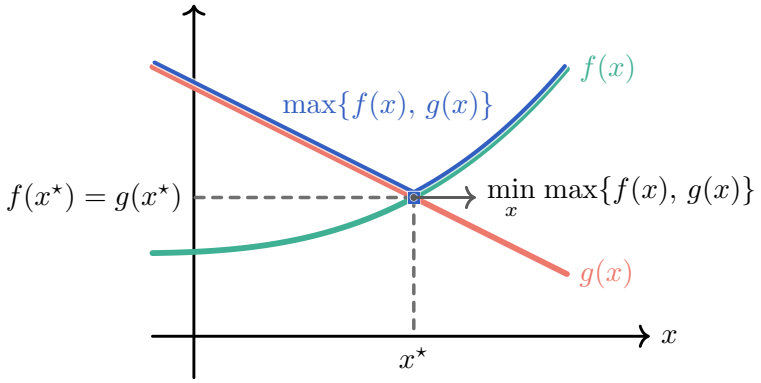

Figure 3.1: The minimizer of the maximum of a strictly increasing function, $f(x)$, and a strictly decreasing function, $g(x)$, satisfies $f(x) = g(x)$.

Corollary 2. *With the three-step preconditioning procedure described, $\kappa(\lambda^\star) \geq 2$, i.e., the condition number of the preconditioned KKT matrix is lower-bounded by 2.*

Proof. By substituting Equation (3.17) in Equation (3.12), we get:

$$\kappa(\lambda^\star) = \frac{1+\sqrt{1+8\,\chi}}{2}$$

where $\chi := \frac{\sigma_{\max}}{\sigma_{\min}}$ is the condition number of $H\,H^\top$. When $\chi = 1$, the lower bound on $\kappa(\lambda^\star)$ is tight. □

Corollary 2 provides a limit on how much we can minimize the condition number of the KKT matrix with the three-step preconditioning procedure described: if $H\,H^\top$ happens to be perfectly conditioned, the condition number of the preconditioned KKT matrix is 2. In practice, row normalization is effective in reducing the condition number of $H\,H^\top$, often by a few orders of magnitude, although rarely to unity. So, although a condition number of unity for the KKT matrix is unattainable (due to being lower-bounded by 2), and despite its lower bound usually not being tight (as a result of $H\,H^\top$ not being perfectly conditioned), the proposed preconditioning procedure can reduce the condition number of the KKT matrix enough to significantly improve the performance of first-order QCP solvers, without sacrificing any of the following features: being (i) mostly analytical (the only iterative

component is shifted power iteration), (ii) entirely factorization-free, and (iii) amenable to customization—all of which are beneficial for online applications.

3.1.4 Optimal Step-Size Ratio in PIPG

Considering Problem 3.5, the primal and dual step-sizes in PIPG [238], α and β, respectively, satisfy the conditions $\alpha,\ \beta > 0$, $\alpha(\|\lambda\, I\| + \beta\|H\|^2) < 1$, where $\|\square\|$ is defined to be the largest singular value of $\square$. One choice for α and β is obtained by parameterizing β in terms of α, i.e., $\beta := \omega^2\alpha$, $\omega \in \mathbb{R}_{++}$, setting the strict inequality to an equality, and solving for α, i.e., $\alpha = \frac{2}{\lambda + \sqrt{\lambda^2 + 4\,\omega^2\,\sigma_{\max}}}$ and $\beta = \omega^2\alpha$, where $\sigma_{\max} := \|H\|^2$. The PIPG iterates for Problem 3.5 are given by (see [238] for more details):

$$z^+ = \Pi_{\mathbb{D}}[\zeta - \alpha\,(\lambda\,\zeta + q + H^\top \eta)] \tag{3.22a}$$

$$w^+ = \Pi_{\mathbb{K}^\circ}[\eta + \beta\,(H(2\,z^+ - \zeta) - h)] \tag{3.22b}$$

$$\zeta^+ = (1-\rho)\,\zeta + \rho\, z^+ \tag{3.22c}$$

$$\eta^+ = (1-\rho)\,\eta + \rho\, w^+ \tag{3.22d}$$

The step-size ratio is given by:

$$\frac{\beta}{\alpha} = \omega^2 \tag{3.23}$$

Typically, the solver parameter, ω, is manually tuned to obtain good performance [238], the tuning process itself being unintuitive in practice. While [52] provides a different step-size rule than the one considered here and adopts an adaptive heuristic based on approximately minimizing the primal-dual gap, in this work, we establish the relationship between the step-size ratio and the objective function scaling factor and find the connection between their optimal values.

Theorem 3. *Scaling the step-size ratio in* PIPG *by a factor of s^2, $s > 0$, is equivalent to scaling the objective function of the corresponding QCP by a factor of $\frac{1}{s}$, i.e., scaling ω by s*

is equivalent to scaling λ *by* $\frac{1}{s}$.

Proof. Scaling ω by s, we get the primal-dual step-sizes $\tilde{\alpha} := \frac{2}{\lambda+\sqrt{\lambda^2+4\,s^2\,\omega^2\,\sigma_{\max}}}$ and $\tilde{\beta} := s^2\,\omega^2\,\alpha$, respectively—this corresponds to scaling the step-size ratio by s^2. Now, we have $\tilde{\alpha} = \frac{\hat{\alpha}}{s}$ and $\tilde{\beta} = s\,\omega^2\,\hat{\alpha}$, where $\hat{\alpha} := \frac{2\,s}{\lambda+\sqrt{\lambda^2+4\,s^2\,\omega^2\,\sigma_{\max}}}$. Let $\hat{\beta} := \omega^2\,\hat{\alpha}$. From [99, Section II, Lemma 3, (ii)], the projection onto a convex cone is nonnegatively homogeneous (and hence, positively homogeneous). Therefore, $\Pi_{\mathbb{K}^\circ}[c\,z] = c\,\Pi_{\mathbb{K}^\circ}[z]$, $c > 0$. Further, defining $\hat{w} := \frac{w}{s}$ and $\hat{\eta} := \frac{\eta}{s}$, dividing Equations (3.22b) and (3.22d) by s, and invoking [99, Section II, Lemma 3, (ii)] on Equation (3.22b), we get:

$$z^+ = \Pi_{\mathbb{D}}\big[\zeta - \hat{\alpha}\,\big(\tfrac{\lambda}{s}\,\zeta + \tfrac{\lambda}{s}\,q + H^\top\hat{\eta}\big)\big] \tag{3.24a}$$

$$\hat{w}^+ = \Pi_{\mathbb{K}^\circ}[\hat{\eta} + \hat{\beta}\,(H(2\,z^+ - \zeta) - h)] \tag{3.24b}$$

$$\zeta^+ = (1-\rho)\,\zeta + \rho\,z^+ \tag{3.24c}$$

$$\hat{\eta}^+ = (1-\rho)\,\hat{\eta} + \rho\,\hat{w}^+ \tag{3.24d}$$

which are the PIPG iterates for the following equivalent problem, Problem 3.25, but with step-sizes $\hat{\alpha}$ and $\hat{\beta}$:

$$\underset{z}{\text{minimize}} \quad \frac{\hat{\lambda}}{2}\,z^\top z + \hat{\lambda}\,q^\top z \tag{3.25a}$$

$$\text{subject to} \quad H\,z - h \in \mathbb{K} \tag{3.25b}$$

$$z \in \mathbb{D} \tag{3.25c}$$

where $\hat{\alpha} = \frac{2}{\hat{\lambda}+\sqrt{\hat{\lambda}^2+4\,\omega^2\,\sigma_{\max}}}$, $\hat{\beta} = \omega^2\,\hat{\alpha}$, and $\hat{\lambda} := \frac{\lambda}{s}$. □

Remark 1. *As a consequence of Theorem 3, tuning* λ *is equivalent to tuning* ω*, i.e., there is a one-to-one mapping between* λ *and* ω*.*

Corollary 3. *For a given* $\lambda > 0$ *in Problem 3.5, the optimal value for the* PIPG *solver parameter,* ω*—in terms of minimizing the condition number of* K*—is given by* $\omega^\star := \lambda\sqrt{\frac{2}{\sigma_{\min}}}$.

Proof. This directly follows from Theorems 2 and 3. □

3.1.5 Shifted Power Iteration

The maximum singular value of $M := H\,H^\top \in \mathbb{R}^{m\times m}$, which is a parameter that factors into the step-sizes of PIPG, can be efficiently estimated using the power iteration method [214], which, in turn, can be customized to the trajectory optimization template for efficient implementation [102]. For the power iteration method to be convergent, the magnitude of the dominant eigenvalue must be strictly greater than the magnitude of every other eigenvalue. We make the assumption that the eigenvalues of matrix M satisfy this condition. The optimal solver parameters in PIPG—given by Corollary 3—however, require an estimate for the *minimum* singular value of M.

General methods to compute the minimum singular value of a matrix, such as the inverse iteration method or the singular value decomposition (SVD), are generally computationally expensive and not amenable to customization, i.e., they are not structure-exploiting, thus making them unsuitable for real-time applications that require online computations. However, since $M \succ 0$, we can use the shifted power iteration method to estimate its smallest singular value [234], which is described in Algorithm 10, where $\sigma_{\min}(A)$ is defined to be the smallest eigenvalue of symmetric matrix $A \succ 0$. First, we perform a spectral shift on M as follows:

$$\widetilde{M} := M - \sigma_{\max}\, I \tag{3.26}$$

where $\sigma_{\max}$ is the largest singular value of M. This shift annihilates the largest eigenvalue of M and consequently *deflates* it to form $\widetilde{M}$. To ensure that the shifted power iteration method is convergent, we make the assumption that the magnitude of the dominant eigenvalue of $\widetilde{M}$ is strictly greater than the magnitude of every other eigenvalue. Let v be the eigenvector corresponding to an arbitrary eigenvalue (singular value) of M, σ, i.e., $M\,v = \sigma\,v$. The matrix-vector product $\widetilde{M}\,v$ yields

$$\widetilde{M}\,v = M\,v - \sigma_{\max}\,I\,v = (\sigma - \sigma_{\max})\,v \tag{3.27}$$

Algorithm 10 Shifted power iteration to estimate $\sigma_{\min}(H\,H^\top)$

Inputs: H, w, ϵ_{abs}, ϵ_{rel}, ϵ_{buff}, $j_{\max}$

Require: $\|w\|_2 > 0$

$\tilde{\sigma} \leftarrow \|w\|_2$ ▷ initialization
for $j \leftarrow 1, \ldots, j_{\max}$ **do**
  $z \leftarrow H^\top w$
  $w \leftarrow \frac{1}{\tilde{\sigma}}\,(H\,z - \sigma_{\max}\,w)$ ▷ $\sigma_{\max} := \|H\|^2$
  $\tilde{\sigma}^\star \leftarrow \|w\|_2$
  if $|\tilde{\sigma}^\star - \tilde{\sigma}| \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}}\,\max\{\tilde{\sigma}^\star, \tilde{\sigma}\}$ **then**
    break
  else if $j < j_{\max}$ **then**
    $\tilde{\sigma} \leftarrow \tilde{\sigma}^\star$
  end if
end for
$\sigma_{\min} \leftarrow (1 - \epsilon_{\text{buff}})\,(\sigma_{\max} - \tilde{\sigma}^\star)$ ▷ buffer the overestimate

Return: $\sigma_{\min}$ ▷ $\approx \sigma_{\min}(H\,H^\top)$

Therefore, $\sigma - \sigma_{\max}$ is an eigenvalue of $\widetilde{M}$. Further, since $\sigma - \sigma_{\max} \leq 0$, matrix $\widetilde{M}$ is symmetric negative semidefinite. Therefore, the power iteration method can be used to find the absolute value of its largest magnitude eigenvalue (largest singular value), which is nothing but $\widetilde{\sigma} := |\sigma_{\min} - \sigma_{\max}| = \sigma_{\max} - \sigma_{\min}$, where $\sigma_{\min}$ is the smallest singular value of M. Finally, $\sigma_{\min}$ can be obtained by subtracting $\widetilde{\sigma}$ from $\sigma_{\max}$, i.e., $\sigma_{\min} = \sigma_{\max} - \widetilde{\sigma}$.

Note that Algorithm 10 is amenable to customization, as shown in [102, Algorithm 5].

3.2 Numerical Results

We first consider the convex optimal control problem from [237] to demonstrate the efficacy of the hypersphere preconditioner. To enable comparison of our method with modified Ruiz equilibration [201], we replace the 2-norm ball constraints with their ∞-norm counterparts.

Further, in compliance with the requirement given by Equation (3.2a), we choose the following state cost matrix: $Q_t := Q = \text{diag}\{1, 1, 0.5, 0.5\},\ t = 1, \ldots, T-1$, where T is chosen to be 50. We scale the terminal state cost matrix by a scaling factor, $\gamma > 0$, i.e., $Q_T := \gamma Q$. By progressively increasing γ, we obtain smaller and smaller terminal state tracking errors, but at the cost of increasing ill-conditioning in the problem.

With that, we compare the hypersphere preconditioner with two state-of-the-art preconditioning techniques: modified Ruiz equilibration [201] and the QR preconditioner [52], in terms of (i) minimizing the condition number of the KKT matrix of the preconditioned problem, and (ii) performance of the proportional-integral projected gradient method (PIPG) [237, 241, 239, 238], a first-order primal-dual method for conic optimization, that has recently gained popularity for problems that fit the trajectory optimization template [240, 125, 103, 102, 51, 49, 136, 58].

All the preconditioners and the solver are implemented in C, the code for which is generated using the MATLAB Coder. All the problems are first solved to high-accuracy using an off-the-shelf interior-point method (constituting the "ground truth"), and PIPG is deemed to have converged if and when the relative error between the PIPG solution and the ground truth drops below 0.5%. The code is available at `https://github.com/UW-ACL/optimal-preconditioning`. Fig. 3.2 shows the effect of the preconditioners on the condition number of the preconditioned KKT matrix, and Fig. 3.3 and Table 3.1 show their effect on the overall solver performance.

To estimate $\sigma_{\min}$ within the hypersphere preconditioner, we use the shifted power iteration method [234], the performance statistics for which are provided in Table 3.2. The presolve time includes the preconditioning time and the time taken by the regular power iteration method to estimate $\sigma_{\max}$, which PIPG uses for step-size determination; however, the presolve step is largely dominated by shifted power iteration. The absolute and relative tolerances for the termination of shifted power iteration are each set to 10^{-9}. The convergence rate of shifted power iteration is proportional to the ratio $\frac{\sigma_{\max}-\sigma_{\min_2}}{\sigma_{\max}-\sigma_{\min}}$, where $\sigma_{\min_2}$ is the second smallest eigenvalue of $H\,H^\top$. We observe that shifted power iteration suffers from slow

convergence when the smallest eigenvalues of $H\,H^\top$ are clustered (explaining the slower convergence for smaller values of γ)—this is a limitation of the method.

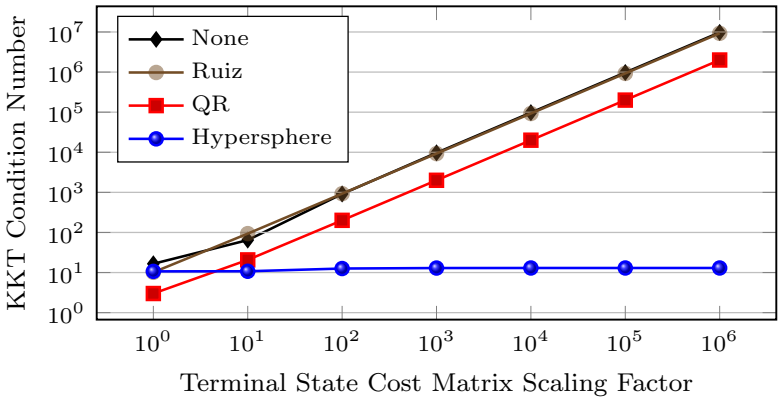

Figure 3.2: The condition number of the preconditioned KKT matrix as a function of the terminal state cost matrix scaling factor, γ. "None" refers to the case where no preconditioning is applied.

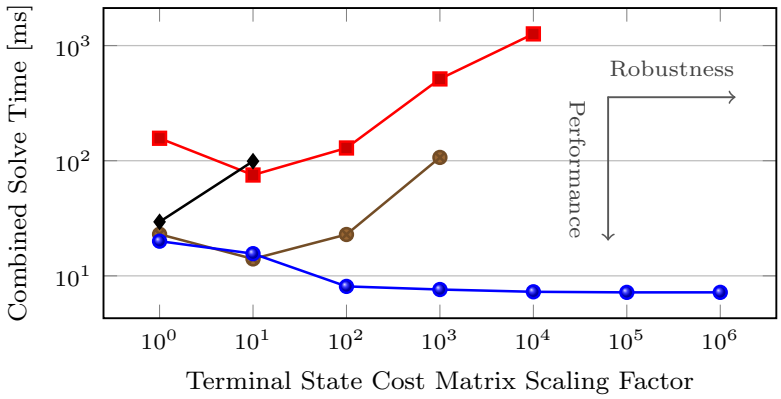

Figure 3.3: The combined solve time (preconditioning and convex solve) as a function of γ. The combined solve time is only plotted if PIPG was able to converge (within 10^5 iterations).

None	3081	11209	10^5	10^5	10^5	10^5	10^5
Ruiz	1101	**237**	1262	10774	10^5	10^5	10^5
QR	2240	2098	2014	3894	28588	10^5	10^5
Hypersphere	**684**	672	**558**	**529**	**525**	**524**	**524**
γ	1	10	10^2	10^3	10^4	10^5	10^6

Table 3.1: The number of PIPG iterations as a function of γ. The green values indicate the least number of PIPG iterations to convergence. The red values indicate that PIPG hit the maximum number of iterations (10^5) and failed to converge.

Iterations	6081	4758	1413	1304	1294	1293	1293
$\frac{\textbf{PT}}{\textbf{CST}}$(%)	68.06	62.74	39.2	37.37	36.91	36.46	36.64
γ	1	10	10^2	10^3	10^4	10^5	10^6

Table 3.2: Performance of shifted power iteration within hypersphere preconditioning. PT: presolve time; CST: combined solve time.

Hypersphere preconditioning was designed with the express goal of mitigating ill-conditioning in the objective function Hessian [102], and we observe that it performs the best, with increasing ill-conditioning, across all metrics. Although PIPG requires longer solve times with the QR preconditioner owing to the loss of sparsity, the QR preconditioner could prove beneficial in solving problems that have dense and ill-conditioned constraint matrices. Finally, modified Ruiz equilibration is a balanced preconditioning approach, and is the only one among the three that does not mandate strong convexity and explicitly accounts for the affine term in the objective function.

Next, to demonstrate the benefit of choosing the optimal objective function scaling factor, we consider a numerical example involving a practical online application. More specifically, we consider the nonconvex multi-phase rocket landing guidance problem from [103], which is solved using sequential conic optimization (SeCO). This involves solving a sequence of QCP subproblems that have strongly convex objective functions, using PIPG, with the application of the hypersphere preconditioner.

The guidance problem is solved in 6 SeCO iterations to a predetermined open-loop terminal-error accuracy for the translation states (< 1 m in position and < 0.5 m s^{-1} in velocity). Fig. 3.4 shows a comparison of cases with no objective function scaling ($\lambda = 1$) and the optimal objective function scaling ($\lambda = \lambda^\star$). We observe a clear reduction in (i) the condition number of the KKT matrix for each subproblem, and (ii) the number of PIPG iterations to convergence for each of the subproblems, thus demonstrating the importance of properly scaling the cost function.

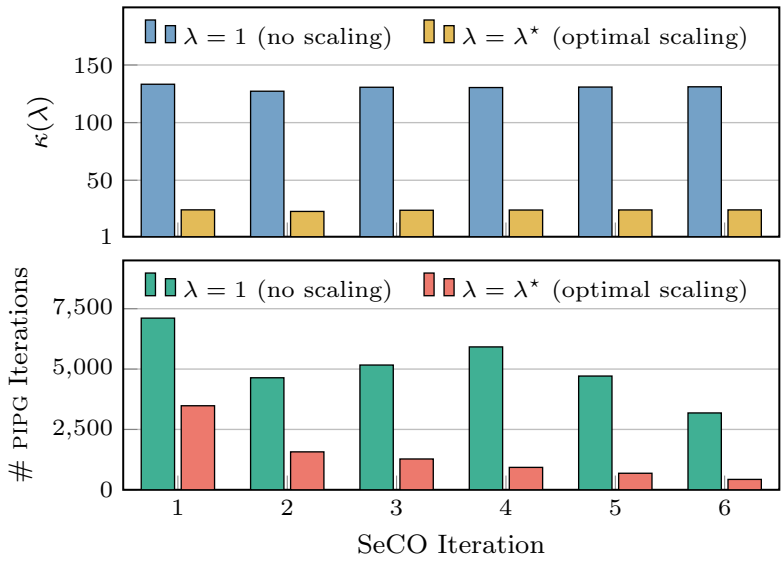

Figure 3.4: The effect of the optimal objective function scaling factor, $\lambda^\star$—on (i) the condition number, $\kappa(\lambda)$, of the KKT matrix, and (ii) the number of PIPG iterations—at each iteration of SeCO for the multi-phase rocket landing guidance problem [103].

Chapter 4

SET-BASED DYNAMIC PROGRAMMING FOR PREDICTIVE CONTROL

In this chapter, we present optimal control and set-based preliminaries in §4.1. We then describe the set-based control architecture for robust control in §4.2 and resilient control in §4.3. Set-based closed-loop free-final-time optimal control is described in §5.4. The proposed set-based framework is demonstrated by means of an autonomous precision landing case study in §6.3.

4.1 Optimal Control

In this section, we present the preliminaries that will aid in the development of the proposed computational framework. In §4.1.1, we formulate the template deterministic optimal control problem and construct a corresponding discrete-time convex optimal control problem with a polytopic feasible set. In §4.1.2, we present a computationally tractable approach to generating a polytopic inner approximation of a second-order (quadratic) cone, intersected with a halfspace, to convert second-order cone constraint sets—which show up in several control problems, including autonomous precision landing—to polytopes. Then, in §4.1.3, we define constrained zonotopes and describe the associated set operations. The use of constrained zonotopes enables the efficient generation of controllable tubes, which we describe in §4.1.4. Finally, we present the one-step optimal control problem—that is executed online as part of the forward rollout—in §4.1.5.

4.1.1 The Deterministic Optimal Control Problem

We first consider a class of free-final-time optimal control problems in continuous-time, in the Lagrange form, with a convex cost functional, a class of nonlinear dynamical systems, and convex state and control constraints, as shown in Problem P1.

Problem P1: Template Continuous-Time Optimal Control Problem (Nonconvex)

$$
\begin{aligned}
&\underset{t_\mathrm{f},\, u}{\text{minimize}} && \textit{Cost Functional}: && \int_0^{t_\mathrm{f}} \beta\, h(u(t))\, dt && && \text{(P1a)}\\
&\text{subject to} && \textit{Dynamics}: && \dot{x}(t) = A_\mathrm{c}\, x(t) + B_\mathrm{c}\, u(t) + G_\mathrm{c}\, h(u(t)) + d_\mathrm{c}, && \forall t \in [0, t_\mathrm{f}] && \text{(P1b)}\\
&&& \textit{State Constraints:} && x(t) \in \mathcal{X}_\mathrm{CVX}, && \forall t \in [0, t_\mathrm{f}] && \text{(P1c)}\\
&&& \textit{Control Constraints:} && u(t) \in \mathcal{U}_\mathrm{CVX}, && \forall t \in [0, t_\mathrm{f}] && \text{(P1d)}\\
&&& \textit{Initial Condition:} && x(0) = x_\mathrm{i} && && \text{(P1e)}\\
&&& \textit{Final Condition:} && x(t_\mathrm{f}) \in \mathcal{X}_{\mathrm{f}_\mathrm{cvx}} && && \text{(P1f)}
\end{aligned}
$$

In Problem P1, $t_\mathrm{f} \in \mathbb{R}_{++}$ is the final time (a free variable), $x(t) \in \mathbb{R}^{n_x}$ is the state, $u(t) \in \mathbb{R}^{n_u}$ is the control input, $\beta \in \mathbb{R}_{++}$ is a constant, and $h : \mathbb{R}^{n_u} \to \mathbb{R}$ is a convex function; $\mathcal{X}_\mathrm{CVX} \subset \mathbb{R}^{n_x}$ and $\mathcal{U}_\mathrm{CVX} \subset \mathbb{R}^{n_u}$ are the convex state and control constraint sets, respectively; $\mathcal{X}_{\mathrm{f}_\mathrm{cvx}}$ is the convex final state set; and $x_\mathrm{i} \in \mathbb{R}^{n_x}$ is the fixed initial condition. In the dynamics, $A_c \in \mathbb{R}^{n_x \times n_x}$ is the system matrix, $B_c \in \mathbb{R}^{n_x \times n_u}$ is the control input matrix, $G_c \in \mathbb{R}^{n_x}$ is a vector that governs the influence of the convex function, h, on the state evolution, and $d_c \in \mathbb{R}^{n_x}$ is the affine term. All the constraint sets considered in this work are assumed to be compact. Note that Problem P1 is nonconvex due to the presence of convex functions, h, in the equality constraint given by Equation (P1b).

Problem P1 can be relaxed to Problem P2, which is convex. In Problem P2, $\sigma(t) \in \mathbb{R}$, which will be treated as an auxiliary control input, is the slack variable introduced to convexify the problem, and $\hat{\mathcal{U}}_\mathrm{CVX} \subset \mathbb{R}^{n_u+1}$ is the augmented control constraint set. Note that Problem P2 bears resemblance to the problem templates considered in the vast body

Problem P2: Template Continuous-Time Optimal Control Problem (Convexified)

$$
\begin{aligned}
&\underset{t_\mathrm{f},\, u,\, \sigma}{\text{minimize}} && \textit{Cost Functional}: && J := \int_0^{t_\mathrm{f}} \beta\, \sigma(t)\, dt && && \text{(P2a)}\\
&\text{subject to} && \textit{Dynamics}: && \dot{x}(t) = A_\mathrm{c}\, x(t) + B_\mathrm{c}\, u(t) + G_\mathrm{c}\, \sigma(t) + d_\mathrm{c}, && \forall t \in [0, t_\mathrm{f}] && \text{(P2b)}\\
& && \textit{Convex Relaxation:} && h(u(t)) \leq \sigma(t), && \forall t \in [0, t_\mathrm{f}] && \text{(P2c)}\\
& && \textit{State Constraints:} && x(t) \in \mathcal{X}_\mathrm{cvx}, && \forall t \in [0, t_\mathrm{f}] && \text{(P2d)}\\
& && \textit{Control Constraints:} && (u(t), \sigma(t)) \in \hat{\mathcal{U}}_\mathrm{cvx}, && \forall t \in [0, t_\mathrm{f}] && \text{(P2e)}\\
& && \textit{Initial Condition:} && x(0) = x_\mathrm{i} && && \text{(P2f)}\\
& && \textit{Final Condition:} && x(t_\mathrm{f}) \in \mathcal{X}_{\mathrm{f}_\mathrm{cvx}} && && \text{(P2g)}
\end{aligned}
$$

of literature on lossless convexification in optimal control [33, 134, 225], which are shown to recover globally optimal solutions to the original nonconvex problems.

Cost-To-Go Augmentation Recently, in the discrete-time setting, a general version of Problem P1 was shown to be losslessly convexifiable under some conditions [225]. Specifically, [225] leverages dynamic programming for the synthesis of the online controller to achieve global optimality in a closed loop using the so-called *controllable tube* (see §4.1.4). A key insight in [225] is that the controllable tube must explicitly account for the cost-to-go, which can be subsequently minimized in the forward rollout. Next, we discuss augmentation of the dynamical system with the cost-to-go dynamics.

From the cost functional in Equation (P2a), J, the cost-to-go at any $t \in [0, t_\mathrm{f}]$ is given by:

$$
c(t) := \int_t^{t_\mathrm{f}} \beta\, \sigma(\tau)\, d\tau \tag{4.1}
$$

Note that $J = c(0)$. Taking the time-derivative of the cost-to-go yield the auxiliary dynamical

system:

$$\dot{c}(t) = \frac{d}{dt}\int_t^{t_\mathrm{f}} \beta\,\sigma(\tau)\,d\tau = \frac{d}{dt}\int_{t_\mathrm{f}}^{t} -\beta\,\sigma(\tau)\,d\tau = -\beta\,\sigma(t) \tag{4.2}$$

From Equation (4.1), the boundary conditions for this auxiliary dynamical system are given by:

$$c(0) = \int_0^{t_\mathrm{f}} \beta\,\sigma(t)\,dt = J = \text{free} \tag{4.3a}$$

$$c(t_\mathrm{f}) = 0 \tag{4.3b}$$

Augmenting the dynamical system with the *cost-to-go dynamics* (Equation (4.2)) and writing the cost functional in terms of the cost-to-go state yields Problem P3, which is equivalent to Problem P2.

Problem P3: Template Continuous-Time Optimal Control Problem (Augmented)

minimize t_f, u, σ, $c(0)$	*Cost Functional*:	$c(0)$		(P3a)
subject to	*Dynamics*:	$\dot{x}(t) = A_\mathrm{c}\,x(t) + B_\mathrm{c}\,u(t) + G_\mathrm{c}\,\sigma(t) + d_\mathrm{c},$	$\forall t \in [0, t_\mathrm{f}]$	(P3b)
	Cost-to-go Dynamics:	$\dot{c}(t) = -\beta\,\sigma(t),$	$\forall t \in [0, t_\mathrm{f}]$	(P3c)
	State Constraints:	$(x(t), c(t)) \in \tilde{\mathcal{X}}_\mathrm{CVX},$	$\forall t \in [0, t_\mathrm{f}]$	(P3d)
	Control Constraints:	$(u(t), \sigma(t)) \in \tilde{\mathcal{U}}_\mathrm{CVX},$	$\forall t \in [0, t_\mathrm{f}]$	(P3e)
	Initial Condition:	$x(0) = x_\mathrm{i}$		(P3f)
	Final Condition:	$x(t_\mathrm{f}) \in \mathcal{X}_{\mathrm{f}_\mathrm{CVX}},\ c(t_\mathrm{f}) = 0$		(P3g)

In Problem P3, $\tilde{\mathcal{X}}_\mathrm{CVX} \subset \mathbb{R}^{n_x+1}$ is the augmented state constraint set and $\tilde{\mathcal{U}}_\mathrm{CVX} \subset \mathbb{R}^{n_u+1}$ is the augmented control constraint set, subsuming the constraint set defined by the convex relaxation in Equation (P2c).

We adopt a zero-order hold (ZOH), i.e., a piecewise-constant control parameterization,

and time-discretize Problem P3. Assuming the true optimal control is a continuous signal, the control parameterization may lead to suboptimality, but since the dynamics themselves are exactly discretized, the continuous-time trajectory will exactly pass through the discrete temporal nodes. The state and control constraint sets are only considered at the nodes. Given (i) the convexity of the control constraint sets, and (ii) the ZOH control parameterization, the control constraints are guaranteed to be satisfied at and in between temporal nodes. For the state, however, these constraints are only imposed at (finitely many) temporal nodes and inter-sample constraint satisfaction is not guaranteed in general. The resulting discrete-time optimal control problem is shown in Problem P4.

Problem P4: Template Discrete-Time Optimal Control Problem (Conic)

$$
\begin{aligned}
&\underset{\substack{N,\, c_1,\\ u_1,\ldots,u_{N-1},\\ \sigma_1,\ldots,\sigma_{N-1}}}{\text{minimize}} && \textit{Cost Function:} && c_1 && && \text{(P4a)}\\
&\text{subject to} && \textit{Dynamics:} && (x_{k+1}, c_{k+1}) = A\cdot(x_k, c_k) + B\cdot(u_k, \sigma_k) + d, && k = 1,\ldots,N-1 && \text{(P4b)}\\
&&& \textit{State Constraints:} && (x_k, c_k) \in \tilde{\mathcal{X}}_{\text{CVX}}, && k = 1,\ldots,N && \text{(P4c)}\\
&&& \textit{Control Constraints:} && (u_k, \sigma_k) \in \tilde{\mathcal{U}}_{\text{CVX}}, && k = 1,\ldots,N-1 && \text{(P4d)}\\
&&& \textit{Initial Condition:} && x_1 = x_{\text{i}} && && \text{(P4e)}\\
&&& \textit{Final Condition:} && (x_N, c_N) \in \tilde{\mathcal{X}}_{\text{f}_{\text{CVX}}} && && \text{(P4f)}
\end{aligned}
$$

In Problem P4, N is the number of temporal nodes (the horizon length), A and d are the discrete-time counterparts of $\text{blkdiag}\{A_{\text{c}}, 0\} \in \mathbb{R}^{(n_x+1)\times(n_x+1)}$ and $(d_{\text{c}}, 0) \in \mathbb{R}^{n_x+1}$, respectively, B is the discrete-time counterpart of $\begin{bmatrix} B_{\text{c}} & G_{\text{c}} \\ 0_{n_u} & -\beta \end{bmatrix} \in \mathbb{R}^{(n_x+1)\times(n_u+1)}$, and $\tilde{\mathcal{X}}_{\text{f}_{\text{CVX}}} := \mathcal{X}_f \times \{0\}$. We assume A is given by a matrix exponential and is hence always invertible. Problem P4 is conic since $\tilde{\mathcal{X}}_{\text{CVX}}$ and $\tilde{\mathcal{U}}_{\text{CVX}}$ can be arbitrary conic constraint sets.

The polytopic approximation of Problem P4 is given by Problem P5.

Problem P5: Template Discrete-Time Optimal Control Problem (Polytopic)

$$
\begin{aligned}
\underset{\substack{N,\, c_1,\\ u_1,\ldots,u_{N-1},\\ \sigma_1,\ldots,\sigma_{N-1}}}{\text{minimize}} \quad & \textit{Cost Function:} && c_1 && && \text{(P5a)}\\
\text{subject to} \quad & \textit{Dynamics:} && (x_{k+1}, c_{k+1}) = A\cdot(x_k, c_k) + B\cdot(u_k, \sigma_k) + d, && k = 1,\ldots,N-1 && \text{(P5b)}\\
& \textit{State Constraints:} && (x_k, c_k) \in \tilde{\mathcal{X}}, && k = 1,\ldots,N && \text{(P5c)}\\
& \textit{Control Constraints:} && (u_k, \sigma_k) \in \tilde{\mathcal{U}}, && k = 1,\ldots,N-1 && \text{(P5d)}\\
& \textit{Initial Condition:} && x_1 = x_{\mathrm{i}} && && \text{(P5e)}\\
& \textit{Final Condition:} && (x_N, c_N) \in \tilde{\mathcal{X}}_{\mathrm{f}} && && \text{(P5f)}
\end{aligned}
$$

In Problem P5, $\tilde{\mathcal{X}}$, $\tilde{\mathcal{U}}$, and $\tilde{\mathcal{X}}_{\mathrm{f}}$ are the polytopic approximations of $\tilde{\mathcal{X}}_{\mathrm{CVX}}$, $\tilde{\mathcal{U}}_{\mathrm{CVX}}$, and $\tilde{\mathcal{X}}_{\mathrm{f_{CVX}}}$, respectively. Note that there is no approximation involved if a constraint set is polytopic to begin with.

The optimal and resilient control approaches we describe in this work are *exact* for the template polytopic optimal control problem given by Problem P5. However, in the realm of optimal control, most problems are better represented by the template of Problem P1, which may involve second-order cone constraints that need to be approximated to fit the template of Problem P5, as shown. In §4.1.2, we provide a numerically tractable method to obtain "good" polytopic approximations to these convexified conic programs (Problem P4) for the special case wherein the conic constraint sets are quadratic cones, which are also known as standard/unit second-order cones (SOCs), ice-cream cones, or Lorentz cones [123].

4.1.2 Polytopic Inner Approximation of a Compact Quadratic Cone

For controllable tube generation as described in §4.1.4, the constraint sets are required to be polytopic, i.e., convex, closed, *and* bounded. There exist methods in the literature to obtain polyhedral *outer* approximations of an n-dimensional (unbounded, i.e., non-compact) quadratic cone [21, 223]. However, in order to ensure that the generated controllable tube is conservative rather than infeasible, i.e., to guarantee feasibility of all points in the set, the polytopic constraint set must be an *inner* approximation of the original constraint set.

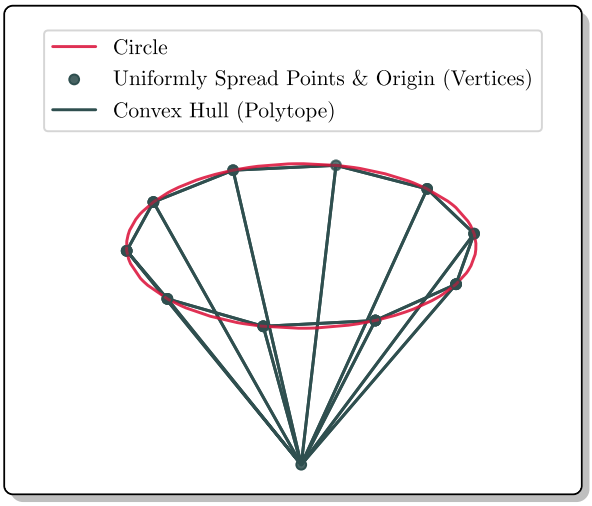

Figure 4.1: Polytopic inner approximation of a 3-dimensional compact quadratic cone.

Motivated by these reasons, we propose a computationally tractable approach to obtaining a polytopic *inner* approximation of a closed and bounded, i.e., compact, n-dimensional quadratic cone, which is a special case of a quadratic cone, intersected with a halfspace.

We define a compact quadratic cone (CQC) as follows:

$$\mathbb{L}^n_{\text{CQC}} := \left\{(z,\, t) \;\middle|\; \|z\|_2 \leq t,\; 0 \leq t \leq t_{\max}\right\} \tag{4.4}$$

where $z \in \mathbb{R}^{n-1}$ and $t \in \mathbb{R}$.

To obtain a polytopic approximation of $\mathbb{L}^n_{\text{CQC}}$, we take a slice of $\mathbb{L}^n_{\text{CQC}}$ along $t = t_{\max}$:

$$\widetilde{\mathbb{L}}^n_{\text{CQC}} := \left\{(z,\, t) \;\middle|\; \|z\|_2 \leq t_{\max},\; t = t_{\max}\right\} \tag{4.5}$$

Here, $\mathcal{Z} := \left\{z \;\middle|\; \|z\|_2 \leq t_{\max}\right\}$ defines an $(n-1)$-dimensional hyperball.

Now, we proceed to uniformly spread points on the boundary of the $(n-1)$-dimensional hyperball by (approximately) solving a difference-of-convex (DC) optimization problem via

the convex-concave procedure[†] [224, 83, 120]. The accuracy of the approximation can be improved by increasing the number of points. These $(n-1)$-dimensional points are then lifted to n dimensions with $t = t_{\max}$.

The convex hull of the union of these points and the origin (in n dimensions) is a polytope that is guaranteed to be an inner approximation of the original CQC. Figure 4.1 shows the polytopic approximation of a 3-dimensional CQC with 10 points spread on the boundary of the 2-dimensional hyperball, i.e., a circle.

4.1.3 Constrained Zonotopes

Polytopes are sets with a finite number of facets (flat sides). Convex polytopes that are closed and bounded, i.e., compact, can be represented either as the convex hull of a finite number of points/vertices (vertex representation: V-REP) or as the intersection of a finite number of halfspaces (halfspace representation: H-REP). In this work, we use the term *polytopes* to refer to compact convex polytopes.

Zonotopes are sets that are given by the Minkowski sum of line segments, called generators, with unit-box-constrained weights, known as latent variables, which are typically higher-dimensional than the set itself. *Constrained* zonotopes (CZs) are zonotopes with additional affine equality constraints on the weights (constrained generator representation: CG-REP)—they are a recently introduced class of sets that can be used to describe arbitrary polytopes, with distinct computational advantages over standard representations (V-REP, H-REP) [192]. In fact, a set is a constrained zonotope if and only if it is a polytope [192, Theorem 1]. While V-REP and H-REP polytopes suffer from the curse of dimensionality with exponential blow-up [210], CG-REP CZs do not (see Table 4.1). This key characteristic of CZs, in addition to the efficiency and accuracy of the corresponding set operations, has led to significant adoption in set-based methods, especially in the fields of estimation and control [192, 229].

[†]This is for $n \geq 3$; for $n \leq 2$, the uniform spreading problem can be solved in closed-form.

Operation	V-REP Polytope	H-REP Polytope	CG-REP CZ
Intersection	×	✓	✓
Minkowski Sum	×	×	✓
Affine Map	✓	×†	✓

Table 4.1: Computational complexity for the required set operations in Algorithm 11: ✓ indicates that the operation has polynomial complexity and × indicates that it has exponential complexity (†exponential complexity in general, but polynomial complexity if the map is invertible).

A CZ in CG-REP is defined by the following set [192, 169, 229, 224]:

$$\mathcal{Z} := \mathcal{Z}(G, c, A, b) = \{x \mid \exists\, \xi,\ x = G\,\xi + c,\ \|\xi\|_\infty \leq 1,\ A\,\xi = b\} \tag{4.6}$$

where $x \in \mathbb{R}^n$, $\xi \in \mathbb{R}^{n_g}$, $G \in \mathbb{R}^{n \times n_g}$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{n_e \times n_g}$, and $b \in \mathbb{R}^{n_e}$; n is the dimension of the CZ, n_g is the number of generators (columns of G), and n_e is the number of equality constraints. Some useful set operations in terms of constrained zonotopes are presented in Table 4.2, where we use the following shorthand to denote a CG-REP CZ: $\mathcal{Z}_i := \{G_i, c_i, A_i, b_i\}$, where $i \in \mathbb{Z}_{++}$ is used to describe operations that involve more than one CZ. The subscript is dropped in the description of operations that only involve one CZ. A selector matrix is defined to be a matrix with one entry per row, which is unity.

4.1.4 Controllable Tube Generation (Offline)

We refer to the target final set as the *terminal set*. We refer to the set of all initial conditions from which a given final set is reachable, in the presence of state and control constraints, for a given trajectory time, as a *controllable set*. The union of all such controllable sets over time is referred to as the *controllable tube*.

We start with deterministic optimal control problems. For problems that fit the template of Problem P5, the exact controllable tube can be constructed by means of a simple backward set-recursion, given by Algorithm 11, which is executed offline. Algorithm 11 is

Operation	**Method**	**Description**
Affine Map $R\,\mathcal{Z}+r$	$\{R\,G,\, R\,c+r,\, A,\, b\}$	$R\,\mathcal{Z}+r := \{R\,x+r \mid x \in \mathcal{Z}\}$, where $R \in \mathbb{R}^{n_r \times n}$ and $r \in \mathbb{R}^{n_r}$
Intersection $\mathcal{Z}_1 \cap \mathcal{Z}_2$	$\left\{ \begin{bmatrix} G_1 & 0 \end{bmatrix},\, c_1,\, \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \\ G_1 & -G_2 \end{bmatrix},\, \begin{bmatrix} b_1 \\ b_2 \\ c_2 - c_1 \end{bmatrix} \right\}$	$\mathcal{Z}_1 \cap \mathcal{Z}_2 := \{x \in \mathcal{Z}_1 \mid x \in \mathcal{Z}_2\}$
Minkowski Sum $\mathcal{Z}_1 \oplus \mathcal{Z}_2$	$\left\{ \begin{bmatrix} G_1 & G_2 \end{bmatrix},\, c_1+c_2,\, \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix},\, \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right\}$	$\mathcal{Z}_1 \oplus \mathcal{Z}_2 := \{x_1+x_2 \mid x_1 \in \mathcal{Z}_1,\, x_2 \in \mathcal{Z}_2\}$
Pontryagin Difference $\mathcal{Z}_1 \ominus \mathcal{Z}_2$	[229, Algorithm 3] (inner approximation)	$\mathcal{Z}_1 \ominus \mathcal{Z}_2 := \{x_1 \mid \forall x_2 \in \mathcal{Z}_2,\, x_1+x_2 \in \mathcal{Z}_1\}$
Intersection with Affine Set $\mathcal{Z} \cap \{x \mid H\,x = h\}$	$\left\{ G,\, c,\, \begin{bmatrix} A \\ H\,G \end{bmatrix},\, \begin{bmatrix} b \\ h - H\,c \end{bmatrix} \right\}$	$x \in \mathbb{R}^n$, $H \in \mathbb{R}^{n_h \times n}$, and $h \in \mathbb{R}^{n_h}$
Slice $\mathcal{Z} \cap \{x \mid E\,x = \bar{x}\}$	$\left\{ G,\, c,\, \begin{bmatrix} A \\ E\,G \end{bmatrix},\, \begin{bmatrix} b \\ \bar{x} - E\,c \end{bmatrix} \right\}$	special case of intersection with affine set, where $\bar{x} \in \mathbb{R}^{n_{\bar{x}}}$, $n_{\bar{x}} \leq n$, and $E \in \mathbb{R}^{n_{\bar{x}} \times n}$ is a selector matrix
Projection $E\,\mathcal{Z}$	$\{E\,G,\, E\,c,\, A,\, b\}$	special case of affine map, where $E \in \mathbb{R}^{n_{\bar{x}} \times n}$ is a selector matrix, $n_{\bar{x}} \leq n$
Distance	$\underset{x \in \mathcal{Z}}{\text{minimize}} \; \|y-x\|_p$	$y \in \mathbb{R}^n$ is the point to be projected onto the set to compute the p-norm distance
Containment	$\underset{x \in \mathcal{Z}}{\text{minimize}} \; \|y-x\|_p$	$y \in \mathcal{Z}$ if the distance is zero; $y \notin \mathcal{Z}$ otherwise
Emptiness	$\underset{x \in \mathcal{Z}}{\text{minimize}} \; 0$	the set is empty if the problem is infeasible; the set is nonempty otherwise
Support	$\underset{x \in \mathcal{Z}}{\text{maximize}} \; \eta^\top x$	$\eta \in \mathbb{R}^n$ is the direction vector along which to evaluate the support function
Extreme Point	$\underset{x \in \mathcal{Z}}{\text{argmax}} \; \eta^\top x$	point on the boundary of the set along the direction vector, η

Table 4.2: Useful set operations in terms of constrained zonotopes, which can be found in [192, 169, 229, 224].

computationally tractable even in high dimensions when the sets in question are represented as constrained zonotopes (CG-REP). This can be extended to handle problems with uncertainties, which will be discussed in §4.2. Note that the generated controllable (backward reachable) sets are guaranteed to be feasible with respect to Problem P4. Here, N is the number of temporal nodes (the horizon length) and CS_k is the k^{th} controllable set, the union

over $k = 1, \ldots, N$ of which is the controllable tube. If the cost-to-go is strictly monotonically decreasing (and since the cost-to-go is bounded by construction), this set recursion is guaranteed to terminate for a finite N, i.e., the set recursion will yield empty sets after N iterations, in which case the first nonempty controllable set, CS_1, corresponds to the maximum-feasible trajectory time. In practice, since N is not known a priori, this backward recursion is performed starting with index 1 until it terminates at an index, N, after which the indices are reversed.

4.1.5 One-Step Optimal Control (Online)

As shown in [225], closed-loop optimal control synthesis may be done via a *forward rollout*, i.e., by solving a sequence of one-step optimal control problems. The forward rollout involves minimization of the current cost-to-go, and one-step set-containment constraints: the next-step controllable set for the propagated state, and the current-step control constraint set for the control input. The optimal control problem—given by Problem P6—itself is a linear program (LP), and can be solved using efficient LP solvers [148, 78, 97, 12].

Algorithm 11 Set Recursion for Controllable Tube Generation

Inputs: N, A, B, d, $\tilde{\mathcal{X}}$, $\tilde{\mathcal{X}}_\text{f}$, $\tilde{\mathcal{U}}$

1: $\text{CS}_N \leftarrow \tilde{\mathcal{X}}_\text{f}$
2: **for** $k = (N-1), (N-2), \ldots, 2, 1$ **do**
3: $\quad \text{CS}_k = \tilde{\mathcal{X}} \cap A^{-1}\left(\text{CS}_{k+1} \oplus \left(-B\tilde{\mathcal{U}} - d\right)\right)$ ▷ $\oplus$: Minkowski sum
4: **end for**

Return: $\text{CS}_{1,\ldots,N}$

Problem P6: One-Step Optimal Control Problem (Polytopic)

$$\begin{aligned}
&\underset{u_k,\,\sigma_k,\,c_k}{\text{minimize}} && \textit{Cost Function}: && c_k && \text{(P6a)}\\
&\text{subject to} && \textit{Dynamics}: && (x_{k+1}, c_{k+1}) = A \cdot (x_k, c_k) + B \cdot (u_k, \sigma_k) + d && \text{(P6b)}\\
&&& \textit{State Constraints (Controllable Set)}: && (x_{k+1}, c_{k+1}) \in \mathrm{CS}_{k+1} && \text{(P6c)}\\
&&& \textit{Control Constraints (Polytopic)}: && (u_k, \sigma_k) \in \tilde{\mathcal{U}} && \text{(P6d)}
\end{aligned}$$

Remark 1. *The control constraint set, $\tilde{\mathcal{U}}$, in the one-step optimal control problem, Problem P6, need not be the polytopic approximation. Instead, we can use the original conic control input set—since it is larger than the polytopic approximation by construction—and solve Problem P7. This has the two-fold benefit of: (i) rendering the convex relaxation tight with respect to the original conic constraint, and (ii) producing solutions with a lower cost. Consequently, instead of solving a sequence of* 1*-step linear programs, we would solve a sequence of* 1*-step conic programs, for which there exist efficient solvers [59, 238, 84, 50].*

Problem P7: One-Step Optimal Control Problem

$$\begin{aligned}
&\underset{u_k,\,\sigma_k,\,c_k}{\text{minimize}} && \textit{Cost Function}: && c_k && \text{(P7a)}\\
&\text{subject to} && \textit{Dynamics}: && (x_{k+1}, c_{k+1}) = A \cdot (x_k, c_k) + B \cdot (u_k, \sigma_k) + d && \text{(P7b)}\\
&&& \textit{State Constraints (Controllable Set)}: && (x_{k+1}, c_{k+1}) \in \mathrm{CS}_{k+1} && \text{(P7c)}\\
&&& \textit{Control Constraints (Conic)}: && (u_k, \sigma_k) \in \tilde{\mathcal{U}}_{\mathrm{CVX}} && \text{(P7d)}
\end{aligned}$$

Remark 2. *The recursive feasibility of Problems P6 and P7 is guaranteed by Equations* (P6c) *and* (P7c)*, respectively.*

4.2 Robust Control

We describe the robust optimal control problem and distinguish between robust open-loop optimal control and robust closed-loop optimal control in §4.2.1. We assume that the state

measurements and the control input are subject to stochastic uncertainty, and further, that the uncertainties in the state are time-varying; we describe the modeling of these uncertainties in §4.2.2. The robust control approach we propose, however, requires the characterization of bounded disturbance sets, such that the closed-loop system is robust to any state and control disturbances that lie within these sets.

Rather than arbitrarily choosing bounded sets for this purpose, we are interested in the systematic construction of bounded sets that allow us to make the claim that the trajectory, even in the presence of random disturbance vectors sampled from known (possibly unbounded) probability distributions, is guaranteed to stay within the controllable tube with a user-specified probability. We describe this procedure in generality in §4.2.3. In the case that the uncertainty is bounded to begin with, the corresponding bounded disturbance sets can be directly used within our framework.

In the presence of disturbances that are coupled in time, the construction of the bounded disturbance sets requires additional considerations. Once the disturbance sets are constructed, the state and control constraint sets, and in turn, the controllable tube, are robustified, i.e., tightened, such that they are robust to the worst-case disturbances from the bounded sets; we describe the construction of the bounded disturbance sets and the robustification of the control and state constraints in §4.2.4. Note that these steps are performed offline. Within the confines of this shrunken feasible space, the controller synthesized online is cost-optimal, making this approach one of *robust optimal control*, in that we prioritize robustness to disturbances first, and then optimize performance among the admissible controls.

4.2.1 The Robust Optimal Control Problem

In the deterministic case, the open-loop and closed-loop optimal control problems are equivalent [25]. In the robust case, however, there are differences—specifically, as shown in §4.2.1, the robust open-loop optimal control problem described in §4.2.1 is more conservative than the robust closed-loop optimal control problem described in §4.2.1.

Robust Open-Loop Optimal Control

Consider the optimal control problem given by Problem P5, but with the inclusion of bounded additive disturbances, $w_k \in \mathcal{W}_k \subset \mathbb{R}^{n_x+1}$, where $\mathcal{W}_k$, $k = 1, \ldots, N$, are compact; see Problem P8. This problem represents the robust *open-loop* optimal control problem, since any control input along the future horizon only depends on the state at the beginning of the horizon.

Problem P8: Robust Open-Loop Optimal Control Problem (Polytopic)

$$\min_{\substack{N,\, c_1 \\ u_1,\ldots,u_{N-1},\\ \sigma_1,\ldots,\sigma_{N-1}}} \quad \max_{w_1,\ldots,w_N} \quad c_1 \tag{P8a}$$

$$\text{s.t.} \quad (x_{k+1}, c_{k+1}) = A \cdot (x_k, c_k) + B \cdot (u_k, \sigma_k) + d + w_k, \quad k = 1, \ldots, N-1 \tag{P8b}$$

$$(x_k, c_k) \in \tilde{\mathcal{X}}, \qquad k = 1, \ldots, N \tag{P8c}$$

$$(u_k, \sigma_k) \in \tilde{\mathcal{U}}, \qquad k = 1, \ldots, N-1 \tag{P8d}$$

$$w_k \in \mathcal{W}_k, \qquad k = 1, \ldots, N \tag{P8e}$$

$$x_1 = x_{\mathrm{i}} \tag{P8f}$$

$$(x_N, c_N) + w_N \in \tilde{\mathcal{X}}_{\mathrm{f}} \tag{P8g}$$

In practice, the optimal control problem is first robustified (i.e., the constraints are tightened) with respect to the worst-case disturbances from the bounded sets (owing to the maximization over them), and then the entire sequence of controls is solved for at once.

Robust Closed-Loop Optimal Control

Consider Problem P5 again, with the same bounded additive disturbances, $w_k \in \mathcal{W}_k \subset \mathbb{R}^{n_x+1}$, $k = 1, \ldots, N$. Now, instead of sequential maximization over disturbance and minimization over controls, we leverage the additional information provided by the value of the current state [25], and formulate an interleaved minimization and maximization problem; see Problem P9. This problem represents the robust *closed-loop* optimal control problem, since the control input at any given point in time explicitly depends on the state at the same time.

Problem P9: Robust Closed-Loop Optimal Control Problem (Polytopic)

$$\min_{N,\,c_1} \; \min_{u_1} \max_{w_1} \ldots \min_{u_{N-1}} \max_{w_{N-1}} \max_{w_N} \quad c_1 \tag{P9a}$$

$$\text{s.t.} \quad (x_{k+1}, c_{k+1}) = A \cdot (x_k, c_k) + B \cdot (u_k, \sigma_k) + d + w_k, \quad k = 1, \ldots, N-1 \tag{P9b}$$

$$(x_k, c_k) \in \tilde{\mathcal{X}}, \qquad k = 1, \ldots, N \tag{P9c}$$

$$(u_k, \sigma_k) \in \tilde{\mathcal{U}}, \qquad k = 1, \ldots, N-1 \tag{P9d}$$

$$w_k \in \mathcal{W}_k, \qquad k = 1, \ldots, N \tag{P9e}$$

$$x_1 = x_{\mathrm{i}} \tag{P9f}$$

$$(x_N, c_N) + w_N \in \tilde{\mathcal{X}}_{\mathrm{f}} \tag{P9g}$$

In practice, Problem P9 is solved by means of dynamic programming—first, a backward recursion, which accounts for robustification, as shown in Algorithm 12, and then a forward rollout, i.e., a sequence of one-step optimal control problems, akin to Problem P7, but with the robustified constraint sets instead.

Robust Closed-Loop Optimal Control vs. Robust Open-Loop Optimal Control

Consider the *max-min inequality* [40, Equation (5.46)]:

$$\sup_{z \in Z} \inf_{y \in Y} f(y, z) \leq \inf_{y \in Y} \sup_{z \in Z} f(y, z) \tag{4.7}$$

for any $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, any $Y \subseteq \mathbb{R}^n$, and any $Z \subseteq \mathbb{R}^m$. When the maximum and the minimum exist, Equation (4.7) can be written as follows:

$$\max_{z \in Z} \min_{y \in Y} f(y, z) \leq \min_{y \in Y} \max_{z \in Z} f(y, z) \tag{4.8}$$

Repeated application of the max-min inequality in Equation (4.8) to Problem P9 (closed-loop) results in the minimization and maximization operations grouped together, as in Problem P8 (open-loop). Thus, we conclude that (i) the robust closed-loop optimal control prob-

lem has a lower cost than that of its open-loop counterpart, and (ii) the open-loop approach is *more conservative* than the closed-loop approach.

4.2.2 State and Control Uncertainty Modeling

Let $y := (x, c) \in \mathbb{R}^{n_x+1}$ be the augmented state and $s := (u, \sigma) \in \mathbb{R}^{n_u+1}$ be the augmented controls.

Consider the following deterministic discrete-time dynamics (Equation (P5b)) in terms of y and s:

$$y_{k+1} = A\, y_k + B\, s_k + d, \quad k = 1, \ldots, N-1 \tag{4.9}$$

We first consider the case where some (or all) of the control inputs are subject to an additive disturbance that is Gaussian and independent and identically distributed (i.i.d.):

$$\tilde{s}_k := s_k + E^u_w\, w^u_k, \quad w^u_k \sim \mathcal{N}(0, \Sigma^u), \quad k = 1, \ldots, N-1 \tag{4.10}$$

where $\Sigma^u \in \mathbb{S}^{\tilde{n}_u}$ is the control disturbance covariance and $\tilde{n}_u \leq n_u + 1$ is the dimension of the uncertain controls. We define an *embedding matrix* to be a matrix that maps/embeds the vector it left-multiplies to/in a (potentially) higher dimension—it only has one entry per column, and that entry is unity (and thus an embedding matrix is full column rank); an embedding matrix is akin to the transpose of a selector matrix. Here, $E^u_w \in \mathbb{R}^{(n_u+1)\times\tilde{n}_u}$ is the uncertain-control embedding matrix, i.e., it lifts w^u_k from $\tilde{n}_u$ dimensions to n_u+1 dimensions. For $k = 1, \ldots, N-1$, we have:

$$y_{k+1} = A\, y_k + B\, \tilde{s}_k + d \tag{4.11a}$$

$$= A\, y_k + B\, (s_k + E^u_w\, w^u_k) + d \tag{4.11b}$$

$$= A\, y_k + B\, s_k + B\, E^u_w\, w^u_k + d \tag{4.11c}$$

Now, additionally, consider the case where some (or all) of the state measurements, $\tilde{y}_k, k = 1, \ldots, N-1$, are uncertain with independent additive Gaussian disturbances with time-varying covariances:

$$\tilde{y}_k := y_k + E_w^x \, w_k^x, \quad w_k^x \sim \mathcal{N}(0, \Sigma_k^x), \quad k = 1, \ldots, N \tag{4.12}$$

where $\Sigma_k^x \in \mathbb{S}^{\tilde{n}_x}$, $k = 1, \ldots, N$, are the (time-varying) state covariances, where $\tilde{n}_x \leq n_x + 1$ is the dimension of the uncertain component of the state and $E_w^x \in \mathbb{R}^{(n_x+1)\times\tilde{n}_x}$ is the uncertain-state embedding matrix. Therefore, for $k = 1, \ldots, N-1$, we have:

$$\begin{aligned}
\tilde{y}_{k+1} &= y_{k+1} + E_w^x \, w_{k+1}^x && \text{(4.13a)} \\
&= A \, y_k + B \, s_k + B \, E_w^u \, w_k^u + d + E_w^x \, w_{k+1}^x && \text{(4.13b)} \\
&= A \, (\tilde{y}_k - E_w^x \, w_k^x) + B \, s_k + B \, E_w^u \, w_k^u + d + E_w^x \, w_{k+1}^x && \text{(4.13c)} \\
&= A \, \tilde{y}_k + B \, s_k + d + (B \, E_w^u \, w_k^u + E_w^x \, w_{k+1}^x - A \, E_w^x \, w_k^x) && \text{(4.13d)}
\end{aligned}$$

Finally, we have:

$$\tilde{y}_{k+1} = A \, \tilde{y}_k + B \, s_k + d + w_k \tag{4.14}$$

where, for $k = 1, \ldots, N-1$, we have:

$$w_k \; = B \, E_w^u \, w_k^u + E_w^x \, w_{k+1}^x - A \, E_w^x \, w_k^x \; = \begin{bmatrix} B \, E_w^u & E_w^x & -A \, E_w^x \end{bmatrix} \begin{bmatrix} w_k^u \\ w_{k+1}^x \\ w_k^x \end{bmatrix} := M \, \hat{w}_k \tag{4.15}$$

where $\hat{w}_k := (w_k^u, w_{k+1}^x, w_k^x) \in \mathbb{R}^{\tilde{n}_u + 2\,\tilde{n}_x}$ is the concatenation of the disturbance vectors influencing the dynamics, and $M := \begin{bmatrix} B \, E_w^u & E_w^x & -A \, E_w^x \end{bmatrix} \in \mathbb{R}^{(n_x+1)\times(\tilde{n}_u + 2\,\tilde{n}_x)}$ is a linear map. Further, note that $w_N = E_w^x \, w_N^x$. Equation (4.14) represents the uncertain dynamical system.

4.2.3 Representing Stochastic Uncertainty as a Bounded Disturbance Set

For Gaussian disturbances, which have unbounded distributions, ellipsoids are the best option to represent bounded disturbance sets, since they are the natural sets of confidence regions due to the shape of the Gaussian kernel. Each bounded set is the p-level set of a Gaussian distribution, which, in turn, guarantees that the true state at a given step, in the presence of a disturbance sampled from this Gaussian distribution, will lie within the corresponding robust controllable set with probability at least p [83].

The logical interpretation of this is as follows:

(i) The probability that a random disturbance vector sampled from the Gaussian distribution lies within the corresponding bounded disturbance set is p (by construction of the bounded disturbance set).

(ii) The true state at a given step is guaranteed to lie within the corresponding controllable set for any disturbance from within the bounded disturbance set (by construction/robustification of the controllable tube).

(iii) From (i) and (ii), the probability that the true state at a given step will lie inside the corresponding controllable set for a random disturbance sampled from the Gaussian distribution is at least p.

We assume the instantaneous disturbance, $w \in \mathbb{R}^n$, is Gaussian, with mean $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in \mathbb{S}^n$, i.e., $w \sim \mathcal{N}(\mu, \Sigma)$.

Consider the ellipsoid with shape matrix Σ, centered around $\mu \in \mathbb{R}^n$ and parameterized by size parameter $R^2 \in [0, \infty)$ [83]:

$$\mathcal{E}(R^2; \mu, \Sigma) := \{q \in \mathbb{R}^n \mid (q - \mu)^\top \Sigma^{-1} (q - \mu) \leq R^2\} \tag{4.16}$$

Now, let $\eta \in \mathbb{R}^n$ be a standard normal random vector, i.e., $\eta \sim \mathcal{N}(0, I_n)$. Then, w can be

written in terms of η as follows [30]:

$$w = \Sigma^{\frac{1}{2}}\,\eta + \mu \tag{4.17}$$

Then,

$$\begin{aligned}
w \in \mathcal{E}(R^2;\mu,\Sigma) &\implies (w-\mu)^\top \Sigma^{-1}(w-\mu) \leq R^2 && \text{(4.18a)}\\
&\implies \left(\Sigma^{\frac{1}{2}}\,\eta + \mu - \mu\right)^\top \Sigma^{-1}\left(\Sigma^{\frac{1}{2}}\,\eta + \mu - \mu\right) \leq R^2 && \text{(4.18b)}\\
&\implies \left(\eta^\top \Sigma^{\frac{1}{2}}\right)\Sigma^{-\frac{1}{2}}\,\Sigma^{-\frac{1}{2}}\left(\Sigma^{\frac{1}{2}}\,\eta\right) \leq R^2 && \text{(4.18c)}\\
&\implies \eta^\top \eta \leq R^2 && \text{(4.18d)}\\
&\implies \eta \in \mathcal{E}(R^2;0,I_n) && \text{(4.18e)}
\end{aligned}$$

Consequently, we have [83, §3.3.2]:

$$\begin{aligned}
\mathbb{P}_w\big(w \in \mathcal{E}(R^2;\mu,\Sigma)\big) &= \mathbb{P}_\eta\big(\eta \in \mathcal{E}(R^2;0,I_n)\big) && \text{(4.19a)}\\
&= \mathbb{P}_\eta\big(\chi^2(n) \leq R^2\big) && \text{(4.19b)}\\
&= F_{\chi^2(n)}(R^2) && \text{(4.19c)}
\end{aligned}$$

where, since η is a normal random variable, $\eta^\top\eta = \|\eta\|_2^2$ can be replaced with $\chi^2(n)$, which is a chi-squared random variable with n degrees of freedom, $F_{\chi^2(n)}(R^2)$ being its cumulative distribution function.

Given a user-specified probability, p, we set:

$$R^2 = F^{-1}_{\chi^2(n)}(p) \tag{4.20}$$

which gives us:

$$F_{\chi^2(n)}(R^2) = p \tag{4.21}$$

and hence, gives us the following guarantee:

$$\mathbb{P}_w\big(w \in \mathcal{E}(R^2; \mu, \Sigma)\big) = p \tag{4.22}$$

4.2.4 Robust Controllable Tube Generation

Robust controllable tube generation involves the contruction of bounded disturbance sets, which we present in §4.2.4, and the tightening, i.e., *robustification*, of the control and state constraint sets, which we present in §4.2.4 and §4.2.4, respectively. The set recursion algorithm for robust controllable tube generation is provided in §4.2.4.

Bounded Disturbance Set Construction

While w_k^x and w_k^u in Equations (4.12) and (4.10), respectively, are independent across time, $\hat{w}_k$ is not, due to the coupling of w_k^x between time-steps. For the purpose of bounded disturbance set construction, and, consequently, robust controllable tube generation, however, we make the assumption that $\hat{w}_k$ are independent, i.e., we assume that $\hat{w}_k \sim \mathcal{N}(0, \Sigma_k^{\hat{w}})$, where $\Sigma_k^{\hat{w}} := \text{blkdiag}\{\Sigma^u, \Sigma_{k+1}^x, \Sigma_k^x\} \in \mathbb{S}^{\tilde{n}_u + 2\,\tilde{n}_x}$, $k = 1, \ldots, N-1$. Note that this assumption is conservative, since it ignores the temporal correlation in $\hat{w}_k$ (the temporal correlation manifests as additional affine constraints on w_k, which are ignored, leading to robustification with respect to a larger disturbance set). Further, since a Gaussian distribution is closed under linear transformations of random variables [213, Theorem 3.3.3], w_k is also Gaussian.

The corresponding bounded disturbance sets are constructed as follows. First, we consider the following ellipsoids (see Equation (4.16)) for $k = 1, \ldots, N-1$:

$$\mathcal{E}_k(R_k^2; 0, \Sigma_k^{\hat{w}}) = \{q \in \mathbb{R}^{\tilde{n}_u + 2\,\tilde{n}_x} \mid q^\top (\Sigma_k^{\hat{w}})^{-1} q \leq R_k^2\} \tag{4.23}$$

and for $k = N$:

$$\mathcal{E}_N(R_N^2; 0, \Sigma_N^x) = \{q \in \mathbb{R}^{\tilde{n}_x} \mid q^\top (\Sigma_N^x)^{-1} q \leq R_N^2\} \tag{4.24}$$

Since the robust controllable tube generated using these bounded disturbance sets will be tightened with respect to the worst-case disturbances from these sets, the tube will be guaranteed to be robust with respect to the original disturbance vectors, $\hat{w}_k := (w_k^u, w_{k+1}^x, w_k^x) \in \mathbb{R}^{\tilde{n}_u + 2\,\tilde{n}_x}$, $k = 1, \ldots, N-1$, as well.

Next, to get the effective disturbance sets for $k = 1, \ldots, N-1$, each in $\mathbb{R}^{n_x+1}$, we take linear transformations of $\mathcal{E}_k(R_k^2; 0, \Sigma_k^{\hat{w}})$ with respect to M, which are ellipsoids themselves [113], i.e.,

$$\mathcal{W}_k := \{Mq \mid M \in \mathbb{R}^{(n_x+1)\times(\tilde{n}_u+2\,\tilde{n}_x)},\ q \in \mathcal{E}_k(R_k^2; 0, \Sigma_k^{\hat{w}})\} \tag{4.25a}$$

$$= \{l \in \mathbb{R}^{n_x+1} \mid l \in \mathcal{E}_k(R_k^2; 0, M\Sigma_k^{\hat{w}} M^\top)\} \tag{4.25b}$$

For $k = N$, we have:

$$\mathcal{W}_N := \{E_w^x\, q \mid E_w^x \in \mathbb{R}^{(n_x+1)\times\tilde{n}_x},\ q \in \mathcal{E}_N(R_N^2; 0, \Sigma_N^x)\} \tag{4.26a}$$

$$= \{l \in \mathbb{R}^{n_x+1} \mid l \in \mathcal{E}_N(R_N^2; 0, E_w^x\, \Sigma_N^x (E_w^x)^\top)\} \tag{4.26b}$$

Further, in accordance with Equation (4.20), we set $R_k^2 = F^{-1}_{\chi^2(\tilde{n}_u+2\,\tilde{n}_x)}(p)$, $k = 1, \ldots, N-1$, and $R_N^2 = F^{-1}_{\chi^2(\tilde{n}_x)}(p)$, wherein, in order to guarantee that the probability of the trajectory at the final-step N lying inside the terminal set CS_N is λ, we choose p as follows [83]:

$$p = \lambda^{\frac{1}{N}} \tag{4.27}$$

where $\lambda \in (0, 1)$ and N is the horizon length. That is to say that with this choice of p, the probability that the trajectory at the terminal time-step, N, will lie inside the terminal set, is at least $p^N = \lambda$.

Robustification of Control Constraints

For the closed-loop system to be robust to control disturbances, we need to tighten the control constraint set such that the commanded control lying in the tightened constraint set implies that the true uncertain control lies in the original control constraint set, for all possible control disturbances within the bounded control disturbance set. Again, the tightened constraints need to be polytopic for robust controllable tube construction, but can be conic for forward rollout. Note that the robustified control constraint set needs to be considered in the forward rollout in the robust case.

Tightening of convex constraints in general is problem-dependent, since it depends on the specific structure of the constraint set under consideration; for example, a robustified halfspace constraint can be modeled as an SOC constraint [40, §4.4.2]. In §6.3.4, we provide one such way to achieve robustification of the convex control constraints with respect to the worst-case control disturbance for the autonomous precision landing problem.

Robustification of State Constraints and Robust Set Recursion

While we explicitly robustify the control constraint set, as shown in §4.2.4, the state constraint sets are implictly robustified via the backward set-recursion for robust controllable tube generation, given by Algorithm 12. Note that Algorithm 12 is similar to Algorithm 11, with the key differences being (i) the use of the robustified control constraint set, $\tilde{\mathcal{U}}_{\text{robust}}$, and (ii) the Pontryagin difference step, which implicitly robustifies the state constraints with respect to the effective disturbance sets, $\mathcal{W}_k$, $k = 1, \ldots, N$. Note that $\mathcal{W}_k$ (at each time-step) accounts for the influence of both the state and control disturbances on the evolution of the state.

Algorithm 12 Set Recursion for Robust Controllable Tube Generation

Inputs: N, A, B, d, $\tilde{\mathcal{X}}$, $\tilde{\mathcal{X}}_{\mathrm{f}}$, $\tilde{\mathcal{U}}_{\mathrm{robust}}$, $\mathcal{W}_{[1:N]}$

1: $\mathrm{CS}_N \leftarrow \tilde{\mathcal{X}}_{\mathrm{f}} \ominus \mathcal{W}_N$ ▷ $\ominus$: Pontryagin difference
2: **for** $k = (N-1), (N-2), \ldots, 2, 1$ **do**
3: $\quad \mathrm{CS}_k = \tilde{\mathcal{X}} \cap A^{-1}\left((\mathrm{CS}_{k+1} \ominus \mathcal{W}_k) \oplus \left(-B\tilde{\mathcal{U}}_{\mathrm{robust}} - d\right)\right)$ ▷ $\oplus$: Minkowski sum
4: **end for**

Return: $\mathrm{CS}_{1,\ldots,N}$

4.3 Resilient Control

When uncertainties/disturbances that affect the system can be modeled, controllers can be synthesized to ensure that the closed-loop system is robust to the worst-case disturbance. However, we would like for the system to remain safe even in the face of unmodeled uncertainties, i.e., *unknown unknowns*, such as disruptions and faults. In other words, we want the system to be *resilient*. This section deals with control in such situations, referred to as resilient control—it is a deterministic framework, designed to enable uncertainty-agnostic "backup" maneuvers to ensure safety.

Note that resilient control is distinct from robust control, in that decisions such as choosing the appropriate backup maneuver to execute need to be made in real-time to ensure safety with respect to unforeseen events, as opposed to characterizing the (known) sources of uncertainties that constantly (but mildly) affect the system, and synthesizing controllers offline.

Specifically, we describe two forms of resilient control in this section: (i) instantaneous reachability (§4.3.1), and (ii) maximal decision-deferral (§4.3.2). In both cases, the controller synthesized online is cost-optimal, and hence, this approach is one of *resilient optimal control*.

A key benefit of our set-based framework is the ability to seamlessly incorporate robustness to modeled uncertainty into the resilient control framework—with resilience to unmodeled uncertainty—by considering robust controllable tubes.

4.3.1 Instantaneous Reachability

In real-time control applications, it may be desirable to determine feasibility of a set of pre-selected target terminal points in a computationally tractable manner, i.e., to be able to quickly assess whether or not they are feasible (reachable), in the event that the nominal target becomes infeasible or unsafe due to an unforeseen event during the maneuver. Further, in certain applications, there could be a priori-determined safety maps corresponding to a set of targets, with a requirement that the trajectory of the system terminate in a safe region—in this case, any chosen target would have to be in the intersection of the set of safe regions (known a priori) and the physically-reachable envelope of the system (to be computed in real-time).

Consider a feasible state trajectory with an associated sequence of controls. A translation-invariant subspace (of the state space) is one wherein any translation of the entire state trajectory in that subspace is also a valid state trajectory for the same sequence of controls. States that lie in a translation-invariant subspace do not show up on the right-hand side of the dynamics. A good example of a translation-invariant subspace is the position space for systems that adhere to Newton's second law of motion. Translation-invariant coordinates are also known as cyclic coordinates [114, 134].

Assumption 1. *The state, x, is assumed to be partitioned as $x = (\hat{x}, \hat{x}^{\complement})$, where $\hat{x}$ are cyclic coordinates corresponding to a translation-invariant subspace of interest, and $\hat{x}^{\complement}$ are the remaining coordinates.*

Note that $\hat{x}^{\complement}$ may contain translation-invariant coordinates.

The instantaneous forward reachable set in a subspace of interest is the set of all terminal state coordinates in that subspace, reachable from the current state at time-index k, in $N-k$

steps. We use the term "instantaneous" to indicate that the reachable set of terminal states at the horizon endpoint is computed with respect to the *current state*, rather than solely with respect to the initial condition of the overall trajectory. Using the controllable tube, it is possible to obtain a closed-form expression for the instantaneous reachable set in any translation-invariant subspace of the state space.

Now, we are ready to present the main instantaneous reachability result:

Notation. *For a set* $\mathcal{A} \subset \mathbb{R}^n$ *and vector* $a \in \mathbb{R}^n$, $\mathcal{A} \oplus a := \mathcal{A} \oplus \{a\}$, *and* $-\mathcal{A} := \{-z \mid z \in \mathcal{A}\}$.

Theorem 1. *Let Assumption 1 hold. Let* $x_k = (\hat{x}_k, \hat{x}_k^{\complement})$ *be the current state at time-index* $k \in \{1, \ldots, N\}$. *Let* $\text{CS}_k \subset \mathbb{R}^{n_x+1}$ *be the controllable set at time-index* k *of the controllable tube, and let* $\mathcal{P} \subset \mathbb{R}^{n_x}$ *be the projection of* CS_k *onto the state coordinates, i.e.:*

$$\mathcal{P} := \{x \mid (x, c) \in \text{CS}_k\} \tag{4.28}$$

Assume $x_k \in \mathcal{P}$. *Let the singleton* $\{\hat{x}_{\text{f}}\}$ *be the target final set corresponding to the* $\hat{x}$*-subspace of* $\mathcal{P}$. *Consider the slice of* $\mathcal{P}$ *along* $\hat{x}_k^{\complement}$, *projected onto the* $\hat{x}$*-coordinates:*

$$\mathcal{S}(\hat{x}_k^{\complement}) := \left\{ s \mid (s, \hat{x}_k^{\complement}) \in \mathcal{P} \right\} \tag{4.29}$$

Then, the instantaneous reachable set in the $\hat{x}$*-subspace is:*

$$\mathcal{R}(\hat{x}_k, \hat{x}_k^{\complement}) = \hat{x}_k \oplus (-\mathcal{S}(\hat{x}_k^{\complement})) \oplus \hat{x}_{\text{f}} \tag{4.30}$$

Proof. We prove the theorem in two steps.

(i) $\hat{x}_k \oplus (-\mathcal{S}(\hat{x}_k^{\complement})) \oplus \hat{x}_{\text{f}} \subseteq \mathcal{R}(\hat{x}_k, \hat{x}_k^{\complement})$

Consider any $s \in \mathcal{S}(\hat{x}_k^{\complement})$. By the definitions of $\mathcal{P}$ and $\mathcal{S}(\hat{x}_k^{\complement})$, $\hat{x}_{\text{f}}$ is reachable from $(s, \hat{x}_k^{\complement})$. Given the translation-invariance of the $\hat{x}$-subspace, translate the trajectory by $\tau = \hat{x}_k - s$. Then the translated current state is now the current state, $(s + \tau, \hat{x}_k^{\complement}) = (\hat{x}_k, \hat{x}_k^{\complement})$, and the

corresponding translated final condition in the $\hat{x}$-subspace is $\hat{x}_\mathrm{f} + \tau = \hat{x}_\mathrm{f} + \hat{x}_k - s$. Since s is arbitrary, we have:

$$\begin{aligned} & \hat{x}_\mathrm{f} + \hat{x}_k - s \in \mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}), \quad \forall\, s \in \mathcal{S}(\hat{x}_k^{\complement}) \\ \iff & \left\{ \hat{x}_\mathrm{f} + \hat{x}_k - s \mid s \in \mathcal{S}(\hat{x}_k^{\complement}) \right\} \subseteq \mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}) \\ \iff & \hat{x}_\mathrm{f} \oplus \hat{x}_k \oplus (-\mathcal{S}(\hat{x}_k^{\complement})) \subseteq \mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}) \end{aligned}$$

(ii) $\mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}) \subseteq \hat{x}_k \oplus (-\mathcal{S}(\hat{x}_k^{\complement})) \oplus \hat{x}_\mathrm{f}$

Consider any $r \in \mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement})$. By the definition of $\mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement})$, r is reachable from $(\hat{x}_k,\, \hat{x}_k^{\complement})$. Given the translation-invariance of the $\hat{x}$-subspace, translate the trajectory by $\tau = \hat{x}_\mathrm{f} - r$. Then the translated final condition in the subspace is now the final condition in the $\hat{x}$-subspace, $r + \tau = \hat{x}_\mathrm{f}$, and the corresponding translated initial condition is $(\hat{x}_k + \tau,\, \hat{x}_k^{\complement}) = (\hat{x}_k + \hat{x}_\mathrm{f} - r,\, \hat{x}_k^{\complement})$. By the definitions of $\mathcal{P}$ and $\mathcal{S}(\hat{x}_k^{\complement})$, and since the target final set in the $\hat{x}$-subspace is a singleton, we have that $\hat{x}_\mathrm{f}$ is reachable from $(s,\, \hat{x}_k^{\complement})$ if and only if $s \in \mathcal{S}(\hat{x}_k^{\complement})$. Now, since r is arbitrary, we have:

$$\begin{aligned} & \hat{x}_k + \hat{x}_\mathrm{f} - r \in \mathcal{S}(\hat{x}_k^{\complement}), \quad \forall\, r \in \mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}) \\ \iff & \left\{ \hat{x}_\mathrm{f} + \hat{x}_k - r \mid r \in \mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}) \right\} \subseteq \mathcal{S}(\hat{x}_k^{\complement}) \\ \iff & \hat{x}_\mathrm{f} \oplus \hat{x}_k \oplus (-\mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement})) \subseteq \mathcal{S}(\hat{x}_k^{\complement}) \\ \iff & -\mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}) \subseteq (-\hat{x}_k) \oplus \mathcal{S}(\hat{x}_k^{\complement}) \oplus (-\hat{x}_\mathrm{f}) \\ \iff & \mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}) \subseteq \hat{x}_k \oplus (-\mathcal{S}(\hat{x}_k^{\complement})) \oplus \hat{x}_\mathrm{f} \end{aligned}$$

Since both (i) and (ii) hold, $\mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement}) = \hat{x}_k \oplus (-\mathcal{S}(\hat{x}_k^{\complement})) \oplus \hat{x}_\mathrm{f}$. □

Remark 3. *The instantaneous reachable set in the chosen translation-invariant subspace, $\mathcal{R}(\hat{x}_k,\, \hat{x}_k^{\complement})$ (Equation* (4.30)*), is nothing but a reflection of $\mathcal{S}(\hat{x}_k^{\complement})$ (Equation* (4.29)*) about the origin, translated by $(\hat{x}_k + \hat{x}_\mathrm{f})$.*

Remark 4. *If the controllable set is convex and represented as a constrained zonotope, Equation* (4.30) *can be efficiently computed in an optimization-free manner, only making use of simple set operations. This procedure is described in Algorithm 13, and is demonstrated by means of a simple example in Figure 4.2.*

Once the instantaneous reachable envelope is computed, a list of pre-selected target terminal points can be assessed for feasibility by checking for containment within the envelope, i.e., if a target terminal point is contained within the envelope, it is guaranteed to be reachable; if it is not contained in the envelope, it is guaranteed to be unreachable. Also, given a pre-determined safety map corresponding to the terminal set, a target terminal point can be chosen such that it lies in the intersection of the set of safe regions and the instantaneous reachable envelope. Further, there could be situations where the goal is to move as far away from the nominal target terminal point as possible, in which case the system can be commanded to divert to an appropriately chosen point on the boundary of the envelope.

Remark 5. *If the target final set is not a singleton and has a nonempty interior, then the corresponding* $\mathcal{R}(\hat{x}_k, \hat{x}_k^{\complement})$ *is no longer the instantaneous reachable set (in the* $\hat{x}$*-subspace). However, it is guaranteed to contain—i.e., it is a superset/an outer-approximation of—the true instantaneous reachable set (in the* $\hat{x}$*-subspace). Specifically, condition (i) in the proof of Theorem 1 will hold, but condition (ii) need not. However, this outer-approximation may still be useful, especially in cases where the goal is to ensure that the target final set is free of debris, for example. Further, this bears resemblance to the notion of the* obstacle region *in robot motion planning [115, §4.3].*

Algorithm 13 Computation of the Instantaneous Reachable Set in a Translation-Invariant Subspace

Inputs: (1) Current controllable set: CS_k
(2) Current states for which to compute the reachable set: $\hat{x}_k$
(3) Target final states for which to compute the reachable set: $\hat{x}_\text{f}$
(4) Complement of the current initial states for which to compute the reachable set: $\hat{x}_k^\complement$

1: $\mathcal{P} \leftarrow$ project onto the state coordinates in CS_k
2: $\mathcal{S} \leftarrow$ slice $\mathcal{P}$ along $\hat{x}_k^\complement$ and project it onto the $\hat{x}$ coordinates
3: $\mathcal{R} \leftarrow \hat{x}_k \oplus (-\mathcal{S}) \oplus \hat{x}_\text{f}$

Return: $\mathcal{R}$

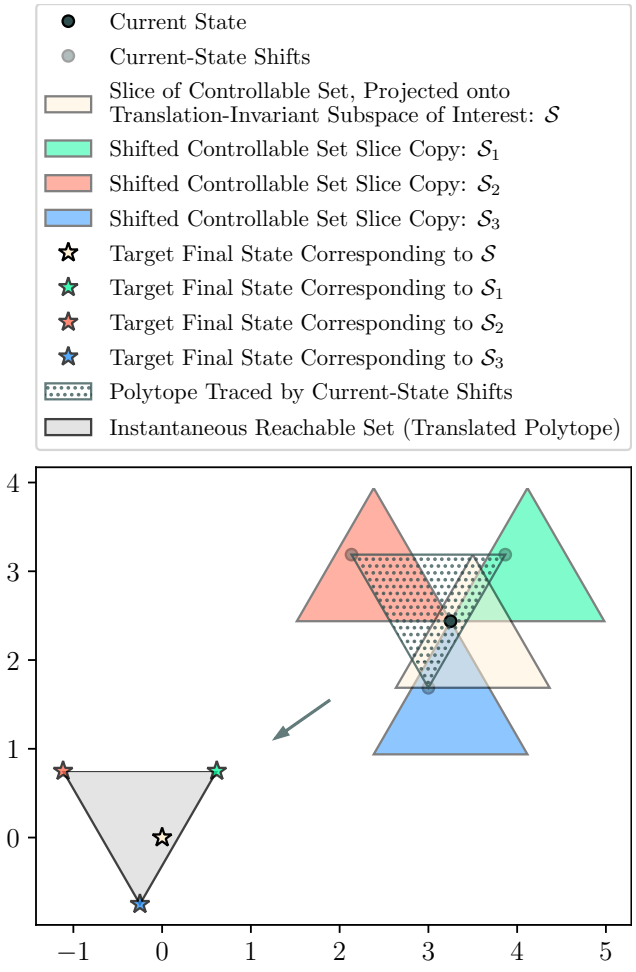

Figure 4.2: Computation of the instantaneous reachable set in a translation-invariant subspace of interest, given a slice of a current-state-containing controllable set projected onto that subspace.

Given a controllable set that contains the current state, i.e., the current initial condition, $x_k \in \mathbb{R}^{n_x}$, we first project away the cost-to-go dimension in the $(n_x + 1)$-dimensional controllable set to obtain an n_x-dimensional polytope, $\mathcal{P} \in \mathbb{R}^{n_x}$.

Let $\hat{x}_k \in \mathbb{R}^{n_{\hat{x}}}$ be the initial condition of the translation-invariant states of interest, where $n_{\hat{x}} < n_x$, and let $\hat{x}_k^{\complement} \in \mathbb{R}^{n_x - n_{\hat{x}}}$ be the initial condition of the remaining states, i.e., of the states that are complementary to the considered translation-invariant states.

We first take a slice of $\mathcal{P}$ along $\hat{x}_k^{\complement}$ to obtain a degenerate polytope in $\mathbb{R}^{n_x}$, after which we project away the collapsed dimensions to obtain a lower-dimensional polytope in $\mathbb{R}^{n_{\hat{x}}}$, $\mathcal{S} \in \mathbb{R}^{n_{\hat{x}}}$.

The polytope $\mathcal{S}$ lives in a translation-invariant subspace. Recall that the controllable set is the set of all initial conditions of the full state, $x \in \mathbb{R}^{n_x}$, from which the target terminal set is reachable, and $\mathcal{S}$ is the set of all initial conditions of the translation-invariant states of interest, $\hat{x} \in \mathbb{R}^{n_{\hat{x}}}$, from which the corresponding projected target terminal set (in $\mathbb{R}^{n_{\hat{x}}}$), is reachable. Note that we require this projected terminal set, $\{\hat{x}_\mathrm{f}\} \subset \mathbb{R}^{n_{\hat{x}}}$, to be a singleton.

The question we are trying to answer is the following: if the system can reach a fixed terminal point $\hat{x}_\mathrm{f}$ from all points in $\mathcal{S}$, what is the set of all points that are reachable from a fixed initial point $\hat{x}_k$ in $\mathcal{S}$? Given the translation invariance of the controllable tube with respect to $\mathcal{S}$, we can consider the (fixed) initial condition, $\hat{x}_k$, and translate $\mathcal{S}$ around it (such that the fixed initial condition point remains inside all translates) to trace the instantaneous feasible reachable envelope in that subspace, which is exact for the template of Problem P5.

This procedure is demonstrated in Figure 4.2 by means of a simple and intuitive example, and is described in Algorithm 13, wherein Line 3 is derived as follows: given slice $\mathcal{S}$,

$$\mathcal{C} := \hat{x}_k \oplus (\hat{x}_k \oplus (-\mathcal{S})) \tag{4.31}$$

gives the "hula-hooped" polytope around the initial condition point (yielding the black poly-

tope in Figure 4.2), and,

$$\mathcal{R} := \mathcal{C} \oplus (\hat{x}_{\mathrm{f}} - \hat{x}_k) \tag{4.32}$$

is the translated polytope, i.e., the polytope obtained by translating $\mathcal{C}$ such that the initial condition point coincides with the terminal point (yielding the solid gray polytope in Figure 4.2). Putting Equations (4.31) and (4.32) together, we get:

$$\mathcal{R} = \hat{x}_k \oplus (\hat{x}_k \oplus (-\mathcal{S})) \oplus (\hat{x}_{\mathrm{f}} - \hat{x}_k) \tag{4.33a}$$

$$= \hat{x}_k \oplus (-\mathcal{S}) \oplus \hat{x}_{\mathrm{f}} \tag{4.33b}$$

4.3.2 Maximal Decision-Deferral

In real-time trajectory generation applications, it is often the case that more information comes in (through hazard detection sensors, for instance) as the system gets closer to the nominal target. While the robust control approach we propose can account for bounded and well-characterized state and control uncertainties (see §4.2), it is not equipped to mitigate *unmodeled* uncertainties/disruptions—such as a dangerous event occurring or a hazard/obstacle being detected at the last minute—that might lead to mission failure if the system were to continue proceeding towards the nominal target.

In such cases, it is desirable to have a collection of pre-selected targets as a contingency measure. While §4.3.1 dealt with determining feasibility of these targets, in this subsection, we determine how to ensure reachability to a collection of targets for as long as possible, which in turn enables divert decision-making as late as possible. Deferred-decision trajectory optimization (DDTO) was recently introduced as a deterministic open-loop framework to tackle such problems [67, 68]. In this work, we propose a tractable set-based, closed-loop implementation of DDTO, that can also seamlessly incorporate robustness to modeled uncertainties.

For this, we construct controllable tubes associated with each of the targets offline. The

controllable tubes can either be deterministic, or robust to state and/or control uncertainties, as described in §4.2. In the case where the targets (final conditions) differ only in a translation-invariant subspace, it suffices to construct one controllable tube and consider translated copies of it. In the one-step OCP solved online, we require the system to stay in the *intersection* of controllable sets, with a prioritized ordering of targets to determine the intersection logic.

If the chosen intersection is nonempty, the one-step OCP is solved. If the OCP returns a feasible solution, we proceed to the next intersection check. If the OCP is infeasible, we select the next best intersection or target based on the prioritized ordering, and proceed. The forward rollout algorithm with decision-deferral for the case of two targets—one nominal and one backup—is given by Algorithm 14. The nominal target is the highest-priority target.

In Algorithm 14, we assume that the controllable tubes associated with the two targets have the same number of controllable sets (which would be the case if the controllable tube for the backup target is a translated copy of the controllable tube for the nominal target, for instance). The containment check in the optimal horizon computation also checks for nonemptiness of the intersection.

We make use of the key insight that optimal trajectories form a tree-like structure and do not exhibit arbitrary clumping, i.e., once the trajectory *trunk* (common segment) branches out, the *branch* trajectories do not intersect again [68, §4 and Figure 3], and only check for emptiness until either an intersection is empty or the problem becomes infeasible—once an empty intersection or an infeasible problem is encountered, another optimal horizon computation is triggered, and the system proceeds to the nominal target (for which feasibility is guaranteed) for the remainder of the forward rollout. If the nominal target becomes unsafe while the system is in a valid intersection of controllable tubes, the system can feasibly divert to the backup target, since reachability to it was maintained by design.

Within the constraints imposed by the problem structure resulting from the chosen intersection logic, the resulting trajectory maintains reachability to the targets for as long as possible, i.e., it maintains maximal decision-deferrability, and further, since we directly leverage

Algorithm 14 Forward Rollout with Decision-Deferral

Inputs: (1) Initial condition: x_{i}
(2) Controllable tube for the nominal target: $\mathrm{CS}^{\mathrm{nominal}}_{1,\dots,N}$
(3) Controllable tube for the backup target: $\mathrm{CS}^{\mathrm{backup}}_{1,\dots,N}$

$\mathrm{CS}^{\mathrm{effective}}_{[1,\dots,N]} \leftarrow \mathrm{CS}^{\mathrm{nominal}}_{[1,\dots,N]} \cap \mathrm{CS}^{\mathrm{backup}}_{[1,\dots,N]}$
$k^\star \leftarrow$ `optimal_horizon`$(x_{\mathrm{i}}, \mathrm{CS}^{\mathrm{effective}}_{[1,\dots,N]})$ ▷ Algorithm 15
BRANCH $\leftarrow$ FALSE
$x_{k^\star} \leftarrow x_{\mathrm{i}}$
for $k = k^\star, \dots, N-1$ **do**
  if BRANCH is FALSE **then**
    if $\mathrm{CS}^{\mathrm{effective}}_{k+1}$ is empty **OR** one-step OCP is infeasible **then**
      BRANCH $\leftarrow$ TRUE
      $\mathrm{CS}^{\mathrm{effective}}_{[1,\dots,N]} \leftarrow \mathrm{CS}^{\mathrm{nominal}}_{[1,\dots,N]}$
      $k^\star \leftarrow$ `optimal_horizon`$(x_{\mathrm{i}}, \mathrm{CS}^{\mathrm{effective}}_{[1,\dots,N]})$ ▷ Algorithm 15
    else
      $x_{k+1}, u_k \leftarrow$ `one_step_optimal_control`$(x_k, \mathrm{CS}^{\mathrm{effective}}_{k+1})$ ▷ solve Problem P7
    end if
  else
    $x_{k+1}, u_k \leftarrow$ `one_step_optimal_control`$(x_k, \mathrm{CS}^{\mathrm{effective}}_{k+1})$ ▷ solve Problem P7
  end if
end for

Return: $u_{k^\star,\dots,N-1}$ ▷ optimal control sequence

the optimal control approach developed in §5.4, in the deterministic case, the (free-final-time) trajectory thus obtained is globally optimal with respect to the chosen cost metric.

Chapter 5

TEMPORAL TECHNIQUES FOR GUIDANCE & CONTROL

In this chapter, we first discuss the notions of time-interval dilation and hybrid control parameterization/discretization in §5.1 and §5.2, respectively. These concepts aid in the development of *single-crossing state-triggered constraints*, which are described in §5.3. Finally, we present a set-based approach to closed-loop free-final-time optimal control in §5.4.

5.1 Time-Interval Dilation

The original nonlinear dynamics, governing the evolution of state $x(t) \in \mathbb{R}^{n_x}$ with control input $u(t) \in \mathbb{R}^{n_u}$, over the entire time-horizon, are given by Equation (5.1).

$$\dot{x}(t) = f(t, x(t), u(t)), \;\; t \in [0, t_f) \tag{5.1}$$

Now, we consider the dynamics in the sub-interval $[t_k, t_{k+1})$, where $0 \leq t_k < t_{k+1} \leq t_N = t_f^-$, $k = 1 : N-1$—where $a : b$ denotes the range of integers between (and including) integers a and b—and define an affine map, $\tau_k(t)$, as shown in Equation (5.2).

$$\tau_k(t) := \frac{t - t_k}{t_{k+1}^- - t_k} \ni \tau_k(t) : [t_k, t_{k+1}) \to [0, 1) \tag{5.2}$$

This mapping is referred to as *time-interval dilation*, as it normalizes the wall-clock time-interval to a known fixed interval—in our case $[0, 1)$—by either shrinking or expanding—and hence *dilating*—the original time-interval. Next, we apply the derivative operator with respect to the dilated time τ_k, denoted by $\overset{\circ}{\square}$, to Equation (5.1), and invoke the chain-rule, as

shown in Equation (5.3), where $t \in [t_k, t_{k+1})$.

$$\begin{aligned}\mathring{x}(t) = \frac{d}{d\tau_k}x(t) = \frac{dt}{d\tau_k}\frac{d}{dt}x(t) = \frac{dt}{d\tau_k}\dot{x}(t) = \underbrace{\left(t_{k+1}^{-} - t_k\right)}_{s_k}\dot{x}(t) \\ = s_k\, f(t, x(t), u(t)) := F(t, x(t), u(t), s_k) \end{aligned} \tag{5.3}$$

The multiplier in Equation (5.3), $s_k := t_{k+1}^{-} - t_k \in \mathbb{R}_+$, which is nothing but the length of the k^{th} wall-clock time-interval, is termed the *dilation factor*. By treating a phase-based subset of s_k, $k = 1{:}N{-}1$, i.e., such that $k \subseteq 1{:}N{-}1$, as decision variables and discretizing the system over them, we allow the optimizer to decide what the temporal spacing of discrete nodes should be in each phase rather than use a uniform temporal grid over the entire horizon. In the approach we propose, this is the key to enabling free-transition-time multi-phase trajectory optimization within a single-shot optimization framework, without requiring any mixed-integer or nonconvex constraints to handle the discrete temporal events/switching.

The number of discrete temporal nodes allocated to each phase is a user-defined parameter. As long as there is a sufficient number of nodes/degrees of freedom per phase for the optimizer to work with, this method is fairly insensitive to node allocation. Although it is possible to allow each dilation factor to be an independent decision variable, we choose to partition the temporal grid based on the phases of flight, and evenly space the temporal nodes within each phase. This measure is taken to mitigate extreme inter-sample constraint violation, which tends to occur when fully adaptive grids are used. The time-dilated dynamics given by Equation (5.3) will be used henceforth. Note that using the time-dilated dynamics given by Equation (5.3) in lieu of Equation (5.1) converts the original *free-final-time* optimal control problem to an equivalent *fixed-final-time* optimal control problem, with the effective horizon being $[0, N-1)$. When compared with the uniform temporal grid case of time-dilation, only $n_{\text{phase}} - 1$ additional variables are needed, n_{phase} being the number of phases. Furthermore, the dynamics are not rendered any more nonlinear than they would be with uniform spacing.

5.2 Hybrid Control Parameterization & Discretization

We adopt a hybrid control parameterization: a first-order hold (FOH) on the control input signal throughout the horizon and a zero-order hold (ZOH) on the control input signal only during discrete switching events such as engine startup and downselection. Both of these control parameterizations have two key properties that make them attractive for optimal control applications: (1) inter-sample satisfaction of the convex control constraints is guaranteed (provided they are satisfied at the discrete temporal nodes), which is in contrast to pseudospectral methods; and, (2) the resulting conic subproblem has a sparsity pattern that is amenable to real-time implementation [133, 208].

In the FOH case, the control profile is parameterized as shown in Equation (5.4), where $t \in [t_k, t_{k+1})$ and $\tau_k \in [0, 1)$, as given by Equation (5.2).

$$u(\tau_k) = (1 - \tau_k)\, u_k + \tau_k\, u_{k+1}, \;\; k = 1\!:\!N-1 \tag{5.4}$$

The LTV dynamics can now be written: (1) using the piecewise-affine control input parameterization given by Equation (5.4); and, (2) in terms of deviations from the reference, as shown in Equation (5.5). The reference quantities are denoted by $\overline{\square}$, and $\Delta\square$ denotes the deviation of a quantity from its reference, i.e., $\Delta\square := \square - \overline{\square}$, and $\Delta\mathring{x}(\tau_k) := \mathring{x}(\tau_k) - F(\tau_k, \overline{x}(\tau_k), \overline{u}(\tau_k), \overline{s}_k)$. The approximate nature of the equation is an artifact of linearization of the original nonlinear dynamics via truncation of the higher-order (≥ 2) terms in the Taylor series expansion.

$$\begin{aligned} \Delta\mathring{x}(\tau_k) \approx\; & A(\tau_k)\, \Delta x(\tau_k) + B(\tau_k)\, (1 - \tau_k)\, \Delta u_k \\ & + B(\tau_k)\, \tau_k\, \Delta u_{k+1} + S(\tau_k)\, \Delta s_k \end{aligned} \tag{5.5}$$

The *state transition matrix* (STM) associated with Equation (5.5), denoted by $\Phi(\tau_k, 0)$, $\tau_k \in [0, 1)$, satisfies the following matrix differential equation: $\mathring{\Phi}(\tau_k, 0) = A(\tau_k)\, \Phi(\tau_k, 0)$, with

$\Phi(0,0) = I_{n_x}$. The unique solution to Equation (5.5) is given by Equation (5.6) [11, 134].

$$\Delta x(\tau_k) = \Phi(\tau_k, 0)\,\Delta x(0) + \int_0^{\tau_k} \Phi(\tau_k, \zeta)\cdot \tag{5.6}$$
$$\cdot\,\{B(\zeta)\,(1-\tau_k)\,\Delta u_k + B(\zeta)\,\tau_k\,\Delta u_{k+1} + S(\zeta)\,\Delta s_k\}\,\mathrm{d}\zeta$$

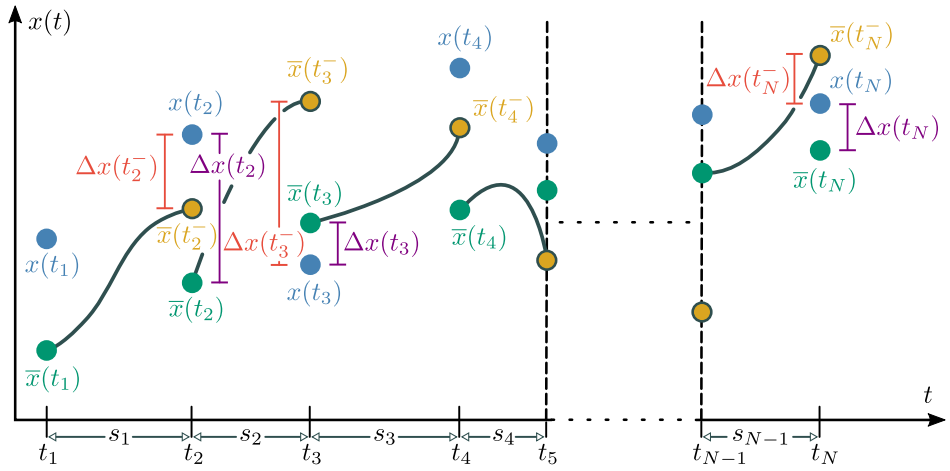

Figure 5.1: Propagation of the state with time-interval dilation. $\Delta x(t_1) := 0$**, where** $t_1 := 0$**. The *stitching condition* for** $k = 1 : N-1$ **is given by** $\Delta x(t_{k+1}^-) + \overline{x}(t_{k+1}^-) = \Delta x(t_{k+1}) + \overline{x}(t_{k+1}) = x(t_{k+1})$**.**

Evaluating Equation (5.6) at $\tau_k = 1^-$, we get Equation (5.7). We replace the limits 0 and 1 with 0_k and 1_k, respectively, to explicitly indicate the dependence on index k, i.e., $\tau_k(t_k) := 0_k$ and $\tau_k\big(t_{k+1}^-\big) := 1_k^-$.

$$\Delta x\big(1_k^-\big) = A_k \Delta x(0_k) + B_k^- \Delta u_k + B_k^+ \Delta u_{k+1} + S_k \Delta s_k \tag{5.7}$$

A_k, B_k^-, B_k^+, and S_k can be computed as the solution to the initial value problem (IVP)

given by Equations (5.8), respectively, integrated from 0_k to 1_k^-.

$$\mathring{\Psi}_A(\zeta) = A(\zeta)\,\Psi_A(\zeta) \tag{5.8a}$$

$$\mathring{\Psi}_{B^-}(\zeta) = A(\zeta)\,\Psi_{B^-}(\zeta) + B(\zeta)\,(1-\zeta) \tag{5.8b}$$

$$\mathring{\Psi}_{B^+}(\zeta) = A(\zeta)\,\Psi_{B^+}(\zeta) + B(\zeta)\,\zeta \tag{5.8c}$$

$$\mathring{\Psi}_S(\zeta) = A(\zeta)\,\Psi_S(\zeta) + S(\zeta) \tag{5.8d}$$

where function $\Psi_A(\zeta)$ is defined such that $\zeta \mapsto \Phi(\zeta, 0_k)$, and the initial conditions for Equations (5.8) are: $\Psi_A(0_k) = I_{n_x}$, $\Psi_{B^-}(0_k) = \Psi_{B^+}(0_k) = 0_{n_x \times n_u}$, and $\Psi_S(0_k) = 0_{n_x}$. Note that Equation (5.9), which we refer to as the *stitching condition*, holds, as is evident from Figure 5.1.

$$\Delta x\big(1_k^-\big) + \overline{x}\big(1_k^-\big) = \Delta x(1_k) + \overline{x}(1_k) \tag{5.9}$$

The discretized dynamics can now be given by Equation (5.10), where $\Delta x_k := \Delta x(0_k)$, $x_{k+1}^{\text{prop}} := \overline{x}\big(1_k^-\big)$, $\overline{x}_{k+1} := \overline{x}(1_k)$, and $\overline{u}(\tau_k) := (1-\tau_k)\,\overline{u}_k + \tau_k\,\overline{u}_{k+1}$, for $1 \le k \le N-1$.

$$\begin{aligned}\Delta x_{k+1} = A_k \Delta x_k + B_k^- \Delta u_k + B_k^+ \Delta u_{k+1} + S_k \Delta s_k \\ + x_{k+1}^{\text{prop}} - \overline{x}_{k+1}\end{aligned} \tag{5.10}$$

The discretized dynamics in terms of the absolute variables are recovered from Equation (5.10), as shown in Equation (5.11).

$$\begin{aligned}x_{k+1} = A_k\,x_k + B_k^-\,u_k + B_k^+\,u_{k+1} + S_k\,s_k + \\ x_{k+1}^{\text{prop}} - \big(A_k\,\overline{x}_k + B_k^-\,\overline{u}_k + B_k^+\,\overline{u}_{k+1} + S_k\,\overline{s}_k\big)\end{aligned} \tag{5.11}$$

Equation (5.11) represents an *exact* discretization of the LTV dynamics, which means that the error between the continuous-time trajectory and the discrete-time trajectory at the discrete temporal nodes is analytically zero.

In order to obtain the corresponding expressions for the ZOH case, the following changes are incorporated: (1) Equation (5.8b) is replaced by Equation (5.12) (and accordingly, B_k^- is replaced by B_k); and, (2) B_k^+ is set to a zero matrix with the same dimensions.

$$\mathring{\Psi}_B(\zeta) = A(\zeta)\,\Psi_B(\zeta) + B(\zeta) \tag{5.12}$$

5.3 Single-Crossing State-Triggered Constraints

We introduce a specialized formulation of compound state-triggered constraints (STCs) for problems in which the trigger functions are activated only once and are strictly monotonic in a neighborhood around the trigger. We refer to these constraints as *single-crossing* compound state-triggered constraints.

Let $g(\cdot)$ be a trigger function that is said to be *activated* on the set $\{x \,|\, g(x) \leq g^\star\}$ for some trigger value $g^\star$. Further, let $g^{-1}(g^\star)$ be a well-defined pre-image. Then, $g(\cdot)$ is called a *single-crossing* trigger function if $x^\star := g^{-1}(g^\star)$ is a singleton and $g(\cdot)$ is strictly monotonic in a neighborhood around $x^\star$. Such a formulation is especially useful in applications such as rocket landing that require certain STCs to be satisfied for mission success, wherein it is reasonable to expect the trigger conditions to be activated once and only once. For instance, it is reasonable to expect/require a rocket in descent from a certain initial altitude, h_i, with its target landing location at the origin, to certainly cross a trigger altitude $0 < h_{\text{trigger}} < h_i$ once during its descent, and not surpass that altitude after.

For the purpose of demonstration, we consider a compound STC with one trigger condition and multiple constraint conditions to be imposed with the AND logic. Let $g(\cdot)$ be the trigger function and $c^j(\cdot)$, $j = 1\!:\!n_c$, be the constraint functions, where n_c is the number of constraint conditions. The purpose of the compound STC [207, Section II.B] is to satisfy the condition given in Equation (5.13), where $x(t)$ is the state. $\forall t \in [0, t_f)$,

$$g(x(t)) \leq 0 \Rightarrow \bigwedge_{j=1}^{n_c} c^j(x(t)) \leq 0 \tag{5.13}$$

In maneuvers with a single-crossing trigger condition, the trigger condition is activated once and only once, i.e., $g(x(t)) = 0$ is guaranteed to activate at some $t = t_{\text{trigger}}$, $g(x(t)) > 0 \; \forall t \in [0, t_{\text{trigger}})$, and $g(x(t)) < 0 \; \forall t \in (t_{\text{trigger}}, t_f)$. Using this fact, the single-crossing compound STC is formulated as shown in Equation (5.14). We emphasize that t_{trigger} itself is free, and hence the compound constraint is state-triggered and not time-triggered.

$$g(x(t)) \geq 0 \qquad \forall t \in [0, t_{\text{trigger}}) \tag{5.14a}$$

$$g(x(t)) = 0, \qquad t = t_{\text{trigger}} \tag{5.14b}$$

$$\left.\begin{aligned} g(x(t)) &\leq 0 \\ c^1(x(t)) &\leq 0 \\ &\vdots \\ c^{n_c}(x(t)) &\leq 0 \end{aligned}\right\} \quad \forall t \in (t_{\text{trigger}}, t_f) \tag{5.14c}$$

If the trigger conditions and the constraint conditions above are individually convex, the compound STC is entirely convex. This is in contrast to existing formulations of STCs in the literature that are inherently nonconvex, regardless of the convexity of the trigger and constraint conditions [207, 208, 176]. However, those methods are more general, in that they do not mandate the trigger condition to be single-crossing—we trade off this generality for convexity/simplicity in our method. Further, we note that the single-crossing STC formulation bears resemblance to the method adopted in [29].

The aforementioned approaches in the literature use uniform temporal spacing of discrete nodes over the entire horizon. As a result, along with the fact that inter-sample constraint satisfaction is typically not guaranteed in general, these approaches do not guarantee the imposition of the constraints exactly at the specified trigger conditions. The triggering of these constraints (in time) is only accurate up to the spacing of the grid, and the corresponding solutions usually demonstrate violation of these constraints with respect to the triggers. This issue becomes more prevalent in maneuvers over very long time horizons, especially when the set of feasible trigger windows is much smaller in comparison, and can be detrimental to

mission success if accurate triggering is required.

We leverage time-interval dilation to impose single-crossing STCs, and treat the windows within which these constraints need to be imposed as distinct phases of flight. This allows for a fine grid in phases that involve critical constraints that need to be satisfied to ensure mission success, and a coarser grid in the more benign phases of flight—thus enabling one-shot multi-phase trajectory optimization. If the solution converges to a feasible trajectory, the STCs are guaranteed to be satisfied at the triggers (since the triggers are imposed as waypoints as in Equation (5.14b)), and at every discrete temporal node within the trigger window.

5.4 Set-based Free-Final-Time Optimal Control

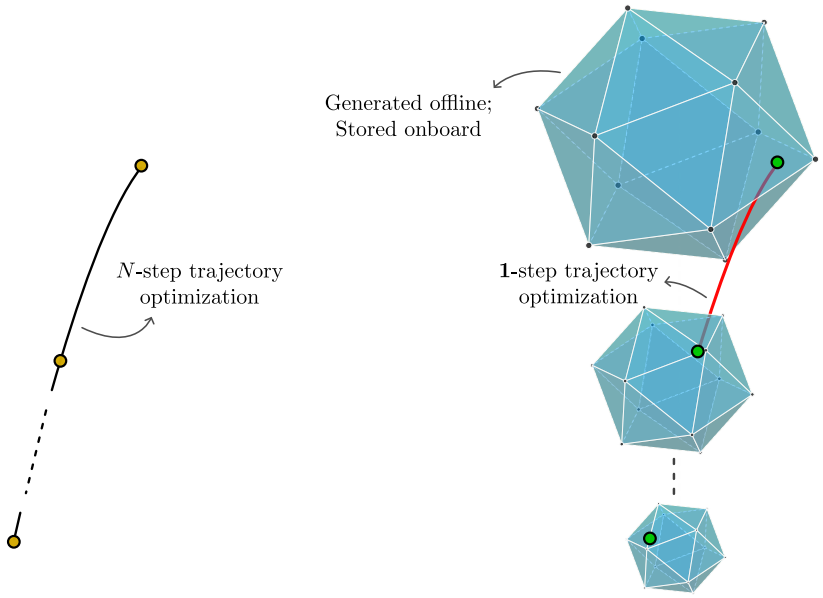

Figure 5.2: The open-loop approach (left) vs. the closed-loop approach using a controllable tube, i.e., a collection of controllable sets (right): the closed-loop approach (Algorithm 16) recovers a globally optimal solution to the open-loop problem (Problem P5).

In this section, we present a computationally tractable set-based approach to free-final-time optimal control. We present the computation of the optimal horizon length in §5.4.1, and the corresponding free-final-time forward rollout algorithm in §5.4.2.

5.4.1 Optimal Horizon Computation

With the existing controllable tube architecture (see Chapter 4), we can compute the free-final-time cost-optimal closed-loop trajectory, for a given initial condition, directly. The proposed approach requires simple and cheap set operations and the solution to small-dimensional linear programs. A notable contribution here is that this method can apply to problems that fit the template of Problem P5 in general, as long as the backward set-

recursion terminates for a finite N; i.e., we make no assumptions on the relationship between the cost metric and the trajectory time. However, additional information about this relationship, if known, can be leveraged to accelerate the search. For example, if the relationship is known to be unimodal [7, 8], a golden section search can be employed [26, 112].

During online execution, the initial condition of the system may be contained in several of the controllable sets. In other words, there may exist feasible solutions to the terminal set starting from many different controllable sets—if the system can reach the terminal set in $M \leq N$ steps (with $M-1$ control actions) for example, it could potentially reach the terminal set in $M+1$ steps (with M control actions) and/or $M-1$ steps (with $M-2$ control actions), and so on, as well.

A natural approach then is to check for containment of the initial condition point in the controllable sets starting from the terminal set, going backward in time, and to then stop when the first containment is achieved. This would correspond to the minimum-time solution [34]. However, rather than choosing the minimum-time controllable set (or any other time-slice based on such heuristics), we can instead choose the time-slice that corresponds to the least possible cost-to-go, which is easy to evaluate in real-time.

Now, we present a general set-theoretic result for free-final-time optimal control that we then leverage within our computational closed-loop control framework.

Theorem 2. *Let $x = x_{\mathrm{i}} \in \mathbb{R}^{n_x}$ be the initial condition (current state). Further, let:*

$$\mathcal{K} := \{k \in \{1, \ldots, N\} \mid x_{\mathrm{i}} \in \{x \mid (x,\, c) \in \mathrm{CS}_k\}\} \tag{5.15}$$

$$\mathcal{C}_k(x_{\mathrm{i}}) := \{c \mid (x_{\mathrm{i}},\, c) \in \mathrm{CS}_k\}, \quad k \in \mathcal{K} \tag{5.16}$$

$$c_k := \min_{c \in \mathcal{C}_k(x_{\mathrm{i}})} c \tag{5.17}$$

where $\mathcal{K}$ is the set of indices of controllable sets CS_k, $k = 1, \ldots, N$, that contain x_{i}, and, for any $k \in \mathcal{K}$, $\mathcal{C}_k(x_{\mathrm{i}})$ is the slice of CS_k at x_{i} projected onto the cost-to-go coordinate, and c_k is the minimum of $\mathcal{C}_k(x_{\mathrm{i}})$. Assume $\mathcal{K}$ is nonempty and, for all $k \in \mathcal{K}$, CS_k is compact. Then

the globally optimal horizon length is $N - k^\star$, where:

$$k^\star = \operatorname{argmin}_{k \in \mathcal{K}} c_k \tag{5.18}$$

and the corresponding globally optimal cost is:

$$c^\star := c_{k^\star} \tag{5.19}$$

Proof. Consider any $k \in \mathcal{K}$. Let the optimal trajectory cost to go from x_i to $\mathcal{X}_\mathrm{f}$ in $N - k$ steps be J_k. By construction, $(x,\, c) \in \textsc{cs}_k$ if and only if there exists a sequence of $N - k$ feasible one-step control actions from x to $\mathcal{X}_\mathrm{f}$, with cost less than or equal to c, i.e.:

$$J_k \leq c,\ \ \forall\, c \in \mathcal{C}_k(x_\mathrm{i}) \iff J_k \leq \min_{c \in \mathcal{C}_k(x_\mathrm{i})} c \iff J_k \leq c_k \tag{5.20}$$

Next, since the last dimension in $\textsc{cs}_k$ is the cost-to-go state (by construction), we have that $(x_\mathrm{i},\, J_k) \in \textsc{cs}_k$, which implies $J_k \in \mathcal{C}_k(x_\mathrm{i})$. Therefore, we have:

$$\min_{c \in \mathcal{C}_k(x_\mathrm{i})} c \leq J_k \iff c_k \leq J_k \tag{5.21}$$

From Equations (5.20) and (5.21), we have:

$$J_k = c_k \tag{5.22}$$

Taking the minimum of Equation (5.22) over $k \in \mathcal{K}$, we get:

$$\min_{k \in \mathcal{K}} J_k = \min_{k \in \mathcal{K}} c_k \tag{5.23}$$

which is attained at $k^\star = \operatorname{argmin}_{k \in \mathcal{K}} c_k$. Hence, the globally optimal horizon length is $N - k^\star$ and the corresponding globally optimal cost is $c^\star = c_{k^\star}$. □

Remark 6. *If* CS_k, $k \in \mathcal{K}$, *are compact convex sets, then each* $\mathcal{C}_k(x_\text{i})$ *is a closed interval in* $\mathbb{R}$, *and* $\min_{c \in \mathcal{C}_k(x_\text{i})} c$ *is its left endpoint.*

To compute the minimum cost-to-go, we first slice the $(n_x + 1)$-dimensional controllable tube along the n_x-dimensional initial condition point, project it onto the cost-to-go coordinate (to obtain an interval, i.e., a set in one dimension), and then compute the left extreme point of the remaining one-dimensional cost-to-go set (interval). This value corresponds to the minimum possible cost-to-go from that particular controllable set. Note that this set, although 1-dimensional, is a constrained zonotope, so computing the left extreme point requires solving a linear program.

We perform this computation for each of the time-slices that contain the initial condition, and pick the time-slice with the least cost-to-go among them. Then, the forward rollout algorithm, as described in §5.4.2, starting from this set, will produce the globally optimal trajectory.

Algorithm 15 Optimal Horizon Computation

Inputs: (1) Initial condition: x_i
(2) Controllable tube: $\text{CS}_{1,\ldots,N}$

1: $\mathcal{K} \leftarrow$ `containment_check`$(x_\text{i}, \text{CS}_{1,\ldots,N})$ ▷ Equation (5.15)
2: **for** $k \in \mathcal{K}$ **do**
3: $\quad \mathcal{C}_k \leftarrow$ slice CS_k along x_i and project it onto the cost-to-go coordinate
4: $\quad c_k \leftarrow \min_{c \in \mathcal{C}_k} c$
5: **end for**
6: $k^\star \leftarrow \text{argmin}_{k \in \mathcal{K}} c_k$

Return: $k^\star$ ▷ optimal start index

Algorithm 15 gives us the optimal starting index, $k^\star$, i.e., the controllable set index from

which to initiate the sequence of $N - k^\star$ control actions, which is guaranteed to give us the globally optimal trajectory.

5.4.2 Forward Rollout

The *forward rollout* of one-step optimal control problems, from the current controllable set (with index k) to the next (with index $k+1$), is described in Algorithm 16. Note that this is distinct from an *open-loop* approach, wherein the sequence of optimal control actions is determined in one shot—unlike that, the sequence of optimal control actions is determined one step at a time here, allowing for the incorporation of feedback at every step, making it a *closed-loop* approach; see Figure 5.2 for a depiction of the same.

Algorithm 16 Forward Rollout

Inputs: (1) Initial condition: x_i
(2) Controllable tube: $\mathrm{CS}_{1,\ldots,N}$

1: $k^\star \leftarrow$ `optimal_horizon`$(x_\mathrm{i}, \mathrm{CS}_{1,\ldots,N})$ ▷ Algorithm 15
2: $x_{k^\star} \leftarrow x_\mathrm{i}$
3: **for** $k = k^\star, \ldots, N-1$ **do**
4: $\quad x_{k+1}, u_k \leftarrow$ `one_step_optimal_control`(x_k, CS_{k+1}) ▷ solve Problem P7
5: **end for**

Return: $u_{k^\star,\ldots,N-1}$ ▷ globally optimal control sequence

Chapter 6
ROCKET LANDING GUIDANCE & CONTROL

In this chapter, we discuss three rocket landing guidance & control applications: multi-phase rocket landing guidance via SeCO in §6.1, dual quaternion-based 6-DoF powered-descent guidance via SeCO in §6.2, and set-based autonomous precision landing in §6.3.

6.1 Multi-Phase Rocket Landing Guidance

As a representative example for multi-phase rocket landing guidance via SeCO, we consider the problem of terrestrial precision landing of a vehicle akin to Starship, which is designed to be a fully reusable rocket and currently in development [193].

We consider a nonlinear planar model of the vehicle, given by Equations (6.1), with $\hat{x}$ being the vertical axis and $\hat{z}$ being the horizontal axis in the inertial frame (and the longitudinal and lateral axes in the body frame, respectively). We use a simple, tractable aerodynamic model to account for the aerodynamic effects on the vehicle during descent [208]. In practice, however, numerical databases can be used along with SeCO [143]. Further, with our approach, aerodynamic-free polynomial coast-phase ballistic trajectory predictions, such as the one provided in [208], are not required. We note that [116] provide an SCP-based 6-DoF multi-phase rocket landing guidance implementation, but impose nonconvex STCs and implement inexact (trapezoidal) discretization, unlike our approach.

The *original* state vector is defined as follows: $x(t) := (m(t), r(t), v(t), \theta(t), \omega(t))$, where $m(t) \in \mathbb{R}_+$ is the mass, $r(t) \in \mathbb{R}^2$ is the position, $v(t) \in \mathbb{R}^2$ is the velocity, $\theta(t) \in \mathbb{R}$ is the body tilt angle with respect to the inertial vertical, and $\omega(t) \in \mathbb{R}$ is the angular velocity of the body.

The control input vector is defined as follows: $u(t) := (T(t), \delta(t))$, where $T(t) \in \mathbb{R}$ is the

thrust magnitude and $\delta(t) \in \mathbb{R}$ is the gimbal deflection angle.

Further, we define a *virtual state* vector and impose all the state constraints, including the boundary conditions, on this variable, in accordance with §2.2.2: $\xi(t) := \big(m^\xi(t), r^\xi(t), v^\xi(t), \theta^\xi(t), \omega^\xi(t)\big)$.

$$\dot{m}(t) = -\alpha_{\mathrm{e}} T(t) \tag{6.1a}$$

$$\dot{r}(t) = v(t) \tag{6.1b}$$

$$\dot{v}(t) = \frac{1}{m(t)}(F_{\mathcal{I}}(t) + A_{\mathcal{I}}(t)) + g \tag{6.1c}$$

$$\dot{\theta}(t) = \omega(t) \tag{6.1d}$$

$$\dot{\omega}(t) = \frac{1}{J(t)}\big(F_{\mathcal{B}_{\hat{z}}}(t)\, l_{\mathrm{cm}} - A_{\mathcal{B}_{\hat{z}}}(t)\, l^j_{\mathrm{cp}}\big) \tag{6.1e}$$

where

$$\alpha_{\mathrm{e}} := \frac{1}{I_{\mathrm{sp}} g_0} \tag{6.2a}$$

$$F_{\mathcal{I}}(t) := T(t) \begin{pmatrix} \cos(\theta(t) + \delta(t)) \\ -\sin(\theta(t) + \delta(t)) \end{pmatrix} \tag{6.2b}$$

$$A_{\mathcal{I}}(t) := R_{\mathcal{I} \leftarrow \mathcal{B}}(t) A_{\mathcal{B}}(t) \tag{6.2c}$$

$$F_{\mathcal{B}}(t) := T(t) \begin{pmatrix} \cos(\delta(t)) \\ -\sin(\delta(t)) \end{pmatrix} \tag{6.2d}$$

$$A_{\mathcal{B}}(t) := -\frac{\rho_{\mathrm{air}} S_{\mathrm{area}} \|v(t)\|_2}{2} C_{\mathrm{aero}} R^\top_{\mathcal{I} \leftarrow \mathcal{B}}(t) v(t) \tag{6.2e}$$

$$R_{\mathcal{I} \leftarrow \mathcal{B}}(t) := \begin{pmatrix} \cos\theta(t) & \sin\theta(t) \\ -\sin\theta(t) & \cos\theta(t) \end{pmatrix} \tag{6.2f}$$

Here, $\alpha_{\mathrm{e}} \in \mathbb{R}_{++}$ is the thrust-specific fuel consumption (TSFC), $I_{\mathrm{sp}} \in \mathbb{R}_{++}$ is the specific impulse of the rocket engine, $g_0 \in \mathbb{R}_{++}$ is standard Earth gravitational acceleration, $g := (-g_0, 0)$, $R_{\mathcal{I} \leftarrow \mathcal{B}}(t) \in \mathrm{SO}(2)$ is the rotation matrix that maps coordinates in the body frame to

the inertial frame, $\rho_{\text{air}} \in \mathbb{R}_{++}$ is the ambient atmospheric density, $S_{\text{area}} \in \mathbb{R}_{++}$ is the reference area, $C_{\text{aero}} := \text{diag}\{c_{\hat{x}}, c_{\hat{z}}\}$ is the aerodynamic coefficient matrix, where $c_{\hat{x}}, c_{\hat{z}} \in \mathbb{R}_{++}$ are the aerodynamic coefficients along the body $\hat{x}$ and $\hat{z}$ axes, respectively, and $J(t) \in \mathbb{R}_{++}$ is the moment of inertia of the vehicle about the body $\hat{y}$ axis (out-of-plane). The body of the vehicle is assumed to be a uniform solid cylinder, and hence, the moment of inertia about its central diameter is given by $J(t) := m(t)\left(\frac{l_{\text{r}}^2}{4} + \frac{l_{\text{h}}^2}{12}\right)$, where $l_{\text{r}} \in \mathbb{R}_{++}$ and $l_{\text{h}} \in \mathbb{R}_{++}$ are the radius and height of the fuselage, respectively.

The engines are assumed to be co-located, and the location of the vehicle mass-center is assumed to be fixed in the body frame. The thrust moment-arm (the distance between the vehicle mass-center and the engine gimbal hinge point) is denoted by $l_{\text{cm}} \in \mathbb{R}_{++}$, and $l_{\text{cp}}^j \in \mathbb{R}_+$ is the aerodynamic moment-arm (the distance between the vehicle mass-center and the center-of-pressure), where $j \in \{0, 1\}$; $j = 0$ for the unpowered phase of flight and $j = 1$ for the powered phases of flight—l_{cp}^0 is assumed to be maintained at zero via independent aerodynamic controls by means of forward and aft flaps (aerodynamic control surfaces), i.e., the center-of-pressure and the mass-center are assumed to be coincident when the vehicle is in the coast phase.

At the engine ignition (PDI) epoch, it is assumed that the following events occur: (1) the forward flaps are fully extended (to maximize drag towards the nose-cone); and, (2) the aft flaps are fully folded (to minimize drag towards the aft section of the vehicle). As a result, the center-of-pressure shifts away from the mass-center, towards the nose-cone, and induces an aerodynamic torque (pitching moment) on the vehicle. Hence, l_{cp}^1 is set to a nonzero value, and it is kept fixed for the remainder of the trajectory. This, along with gimbaling of the rocket engines, is used to induce the flip maneuver to get the vehicle upright in preparation for terminal descent.

The guidance problem is partitioned into four phases: (1) the unpowered, subsonic coast phase; (2) the high-thrust (3-engine) burn phase; (3) the low-thrust (1-engine) burn phase; and, (4) the altitude-triggered terminal descent phase, separated by the following important discrete events/epochs: (1) powered-descent initiation (PDI) or engine ignition, t_{ignition};

(2) engine downselection or switching from a triple-engine burn to a single-engine burn, t_{switch}; and, (3) altitude-based triggering of the terminal descent phase, t_{trigger}. We emphasize that putting all of these phases together implicitly leads to a free-ignition-time, free-engine-switching-time, *and* free-final-time optimal control problem (subject to the temporal constraints imposed).

For the discretized problem, the temporal grid we choose is as given by Equation (6.3), where N is the number of discrete temporal nodes and k_{ignition}, k_{switch}, and k_{trigger} are the nodes at which the discrete events occur.

$$k \in \{1, \ldots, k_{\text{ignition}}, \ldots, k_{\text{switch}}, \ldots, k_{\text{trigger}}, \ldots, N\} \tag{6.3}$$

6.1.1 Common constraints

The dynamics and temporal constraints are imposed over the entire horizon. The dynamics constraint is given by Equation (6.15b). The constraint given by Equation (6.4) is imposed on the dilation factors to ensure they are bounded.

$$s_{\min} \leq s_l \leq s_{\max}, \;\; l \in \{1{:}n_{\text{phase}}\} \tag{6.4}$$

In order to enable the imposition of a single-crossing compound STC in the altitude-triggered terminal descent phase, we impose a minimum altitude constraint in the first three phases, as given by Equation (6.5).

$$r^{\xi}_{x_k} \geq h_{\text{trigger}}, \;\; k \in \{1{:}k_{\text{trigger}}-1\} \tag{6.5}$$

6.1.2 The unpowered coast phase, $k \in \{1{:}k_{\text{ignition}}-1\}$

The center-of-pressure is coincident with the mass-center, i.e., $j = 0$ and $l^j_{\text{cp}} = l^0_{\text{cp}} = 0$ in the dynamics. In implementation, the thrust magnitude is set to zero (the gimbal angle is inconsequential), and a zero-order hold (ZOH) is assumed on the control input signal for

this phase, in order to avoid control input constraint violation when the engines are ignited in the next phase. A first-order hold (FOH) can also be assumed here, if the engine startup time is significant and needs to be taken into account (with appropriate constraints on the dilation factor).

The following initial conditions are imposed:

$$\begin{aligned} m_1^\xi = m_i,\, r_1^\xi = r_i,\, v_1^\xi = v_i, \\ \theta_1^\xi = \theta_i,\, \omega_1^\xi = \omega_i,\, T_1 = 0 \end{aligned} \tag{6.6}$$

where m_i, r_i, v_i, θ_i, and ω_i are the initial values for the mass, position, velocity, body tilt angle, and angular velocity, respectively. The thrust magnitude at the first node is constrained to be zero.

6.1.3 The high-thrust burn phase, $k \in \{k_{\text{ignition}} : k_{\text{switch}} - 1\}$

The center-of-pressure shifts towards the nose-cone of the vehicle, i.e., $j = 1$ and $l_{\text{cp}}^j = l_{\text{cp}}^1 > 0$ in the dynamics. We model this as a discrete change in its value, which is then held constant for the remainder of the trajectory. The following constraints are imposed on the control variables:

$$3\,T_{\min} \leq T_k \leq 3\,T_{\max} \tag{6.7}$$

$$\begin{aligned} \max\{-\delta_{\max}, -\dot{\delta}_{\max}\bar{s}_{k-1} + \bar{\delta}_{k-1}\} \leq \delta_k \\ \leq \min\{\delta_{\max}, \dot{\delta}_{\max}\bar{s}_{k-1} + \bar{\delta}_{k-1}\} \end{aligned} \tag{6.8}$$

where $T_{\min}$ and $T_{\max}$ are the lower and upper bounds on the thrust magnitude for a single engine, respectively, $-\delta_{\max}$ and $\delta_{\max}$ are the lower and upper bounds on the gimbal deflection angle, respectively, and $-\dot{\delta}_{\max}$ and $\dot{\delta}_{\max}$ are the lower and upper bounds on the gimbal rate, respectively. Equation (6.8) combines the gimbal angle and rate constraints, by using reference values in the lower and upper bounds to make them constants and hence, avoid overlapping projections on the same variable. This constraint is, however, exact at conver-

gence, and this formulation has been observed to work well in practice. Although there are only n_{phase} dilation factor decision variables, we consider the length of $\bar{s}$ to be $N-1$, such that the dilation factors are repeated to span each phase.

ZOH is assumed on the control input signal between $k_{\text{switch}}-1$ and k_{switch}, in preparation for engine downselection, which marks the beginning of the next phase. FOH can also be assumed here, if the engine shutdown time is significant and needs to be accounted for. In order to ensure that the gimbal deflection angle profile does not have any discontinuities, we set the gimbal rate to zero between these nodes, as shown in Equation (6.9).

$$\delta_{k_{\text{switch}}} = \bar{\delta}_{k_{\text{switch}}-1} \tag{6.9}$$

6.1.4 The low-thrust burn phase, $k \in \{k_{\text{switch}} : k_{\text{trigger}}-1\}$

This phase is similar to the high-thrust burn phase, apart from the fact that the switch from 3 to 1 engines occurs at k_{switch}, and the bounds on the thrust magnitude are changed accordingly, as shown in Equation (6.10). The combined gimbal constraint is left unchanged from the previous phase, and is shown in Equation (6.11). FOH is assumed on the control input signal in this phase, and for the remainder of the trajectory.

$$T_{\min} \leq T_k \leq T_{\max} \tag{6.10}$$

$$\begin{aligned} \max\{-\delta_{\max}, -\dot{\delta}_{\max}\bar{s}_{k-1} + \bar{\delta}_{k-1}\} &\leq \delta_k \\ &\leq \min\{\delta_{\max}, \dot{\delta}_{\max}\bar{s}_{k-1} + \bar{\delta}_{k-1}\} \end{aligned} \tag{6.11}$$

6.1.5 The terminal descent phase, $k \in \{k_{\text{trigger}} : N\}$

The final phase, the terminal descent phase, is the most heavily and tightly constrained phase of flight. This is designed as such in order to enable closed-loop precision landing, i.e, to ensure that the generated guidance trajectories (the feedforward control input signal and the reference state profiles) are amenable to tight tracking via feedback controllers. Such a phase would be especially useful if sub-meter touchdown accuracy is required, for instance,

if the vehicle is to be retrieved by the launch tower itself [13].

The constraint on the thrust magnitude is left unchanged from the previous phase, and is shown in Equation (6.12).

$$T_{\min} \leq T_k \leq T_{\max} \tag{6.12}$$

An altitude-triggered single-crossing compound STC, with five constraint conditions, is imposed, as shown in Equations (6.13). The constraint conditions include the following: maximum speed, maximum tilt, maximum angular speed, glideslope, and tighter gimbal deflection bounds. The altitude constraint forms the trigger condition. All of these constraints are imposed in the interval $k \in \{k_{\text{trigger}} : N-1\}$.

$$r^{\xi}_{x_k} \begin{cases} = h_{\text{trigger}}, & \text{if } k = k_{\text{trigger}} \\ \leq h_{\text{trigger}}, & \text{otherwise} \end{cases} \tag{6.13a}$$

$$|r^{\xi}_{z_k}| \leq \begin{cases} \tan\gamma_{\text{gs}}\, h_{\text{trigger}}, & \text{if } k = k_{\text{trigger}} \\ \tan\gamma_{\text{gs}}\, \bar{r}^{\xi}_{x_k}, & \text{otherwise} \end{cases} \tag{6.13b}$$

$$\|v^{\xi}_k\|_2 \leq v_{\max} \tag{6.13c}$$

$$|\theta^{\xi}_k| \leq \theta_{\max} \tag{6.13d}$$

$$|\omega^{\xi}_k| \leq \omega_{\max} \tag{6.13e}$$

$$\begin{aligned} \max\{-\delta_{\max_{\text{TD}}}, -\dot{\delta}_{\max}\bar{s}_{k-1} + \bar{\delta}_{k-1}\} &\leq \delta_k \\ &\leq \min\{\delta_{\max_{\text{TD}}}, \dot{\delta}_{\max}\bar{s}_{k-1} + \bar{\delta}_{k-1}\} \end{aligned} \tag{6.13f}$$

The altitude and glideslope constraints are treated differently at the trigger epoch and after. The altitude constraint is posed as an equality at the trigger—this ensures that the constraint conditions are exactly satisfied at the trigger. Further, the glideslope constraint is cast in the form of box constraints in terms of the reference values, in order to enable closed-form projections (without this measure, there would be two constraints on $r^{\xi}_{x_k}$ at every temporal

node, thus precluding closed-form projections). Similar to the combined gimbal constraint, this constraint is exact at convergence. The bounds on the gimbal deflection angle are tightened in this phase as well, i.e., $\delta_{\max_{\text{TD}}} < \delta_{\max}$.

The following terminal boundary conditions are imposed:

$$\begin{aligned} m_N^{\xi} \geq m_{\text{dry}},\, r_N^{\xi} = r_f,\, v_N^{\xi} = v_f, \\ \theta_N^{\xi} = \theta_f,\, \omega_N^{\xi} = \omega_f,\, \delta_N = 0 \end{aligned} \tag{6.14}$$

where m_{dry} is the dry mass of the vehicle, and r_f, v_f, θ_f, and ω_f are the terminal values for the position, velocity, body tilt angle, and angular velocity, respectively. The gimbal angle at the final node is constrained to be zero to avoid plume-impingement on the retrieval structure.

6.1.6 The discrete SeCO *subproblem*

We impose a soft trust region on the decision variable and use the penalized trust region (PTR) algorithm [208, 174, 176]. The discretized conic subproblem with virtual state(s) and a soft trust region is shown in Problem (6.15), which is strongly convex.

$$\min_{u,\,s} \quad w_{\text{c}} J(x_N) + \tfrac{1}{2}(w_{\text{tr}} J_{\text{tr}} + w_{\text{vse}} J_{\text{vse}}) \tag{6.15a}$$

$$\text{s.t.} \quad x_{k+1} = \text{RHS of DT dynamics}, \quad k = 1{:}N{-}1 \tag{6.15b}$$

$$\xi_k \in \mathcal{X}_\nabla, \qquad k = 1{:}N \tag{6.15c}$$

$$u_k \in \mathcal{U}_\nabla, \qquad k = 1{:}N \tag{6.15d}$$

$$s_k \in \mathcal{S}_\nabla, \qquad k = 1{:}n_{\text{phase}} \tag{6.15e}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, $\xi \in \mathbb{R}^{n_x}$ is the *virtual state* vector, $u \in \mathbb{R}^{n_u}$ is the control input vector, and $s \in \mathbb{R}_{++}$ is the dilation factor (vector); $\mathcal{X}_\nabla$, $\mathcal{U}_\nabla$, and $\mathcal{S}_\nabla$ are the state, control, and temporal constraint sets, respectively, which are assumed to be closed and convex; $J(x_N)$ is the original cost function, assumed to be in the Mayer form [23],

$J_{\text{tr}} := \sum_{k=1}^{N} \left(\|x_k - \overline{x}_k\|_2^2 + \|u_k - \overline{u}_k\|_2^2 \right) + \sum_{k=1}^{n_{\text{phase}}} \|s_k - \overline{s}_k\|_2^2$ is the trust region penalty, and $J_{\text{vse}} := \sum_{k=1}^{N} \|x_k - \xi_k\|_2^2$ is the virtual state error penalty, w_c, w_{tr}, and w_{vse} being their respective weights.

Problem (6.15) can be vectorized, i.e., assembled into the form of Problem (6.16), by stacking all the decision variables into a single vector, z. For more details, see [238].

$$\min_{z} \quad \frac{1}{2} z^\top Q z + \langle q, z \rangle \tag{6.16a}$$

$$\text{s.t.} \quad Hz - h = 0 \tag{6.16b}$$

$$z \in \mathbb{D} \tag{6.16c}$$

We define $J(x_N)$ in Equation (6.15a) to be $-m_N$, i.e., the final mass of the vehicle is maximized (thus minimizing propellant consumption). The discrete SeCO subproblem, which is a second-order cone program (SOCP), can now be given as follows:

min *Objective function*: Equation (6.15a)

s.t. *Common constraints*, §6.1.1:

Equations (6.15b), (6.4), and (6.5)

Coast phase, §6.1.2:

Equations (6.6)

High-thrust burn phase, §6.1.3:

Equations (6.7), (6.8), and (6.9)

Low-thrust burn phase, §6.1.4:

Equations (6.10) and (6.11)

Terminal descent phase, §6.1.5:

Equations (6.12), (6.13), and (6.14)

6.1.7 Numerical results

For our numerical implementation of the multi-phase rocket landing guidance algorithm, we choose a grid of $N = 16$ discrete temporal nodes, with one node allocated to the unpowered coast phase, and 5 nodes allocated to each of the remaining three phases of flight, i.e, $k_{\text{ignition}} = 2$, $k_{\text{switch}} = 7$, $k_{\text{trigger}} = 12$. The following parameters are used, where the ones pertaining to the vehicle/maneuver were either estimated or obtained from public sources [56, 186, 134].

$$g_0 = 9.81\,\mathrm{m\,s^{-2}},\ I_{\text{sp}} = 330\,\mathrm{s},\ l_r = 4.5\,\mathrm{m},\ l_h = 50\,\mathrm{m},$$

$$l_{\text{cm}} = 0.4\,l_h,\ l^0_{\text{cp}} = 0\,\mathrm{m},\ l^1_{\text{cp}} = 0.2\,l_h,\ \rho_{\text{air}} = 1.225\,\mathrm{kg\,m^{-3}},$$

$$S_{\text{area}} = 545\,\mathrm{m^2},\ v_{\text{terminal}} = 85\,\mathrm{m\,s^{-1}},\ m_i = 100000\,\mathrm{kg},$$

$$c_{\hat{x}} = 0.0522,\ c_{\hat{z}} = 0.4068,\ T_{\max} = 2200\,\mathrm{kN},\ T_{\min} = 880\,\mathrm{kN},$$

$$\delta_{\max} = 10^\circ,\ \dot{\delta}_{\max} = 15^\circ\mathrm{s^{-1}},\ h_{\text{trigger}} = 100\,\mathrm{m},\ \gamma_{\text{gs}} = 5^\circ,$$

$$v_{\max} = 20\,\mathrm{m\,s^{-1}},\ \theta_{\max} = 5^\circ,\ \omega_{\max} = 2.5^\circ\mathrm{s^{-1}},\ \delta_{\max_{\text{TD}}} = 1^\circ,$$

$$m_{\text{dry}} = 85000\,\mathrm{kg},\ r_i = (1000, 100)\,\mathrm{m},\ v_i = (-90, 0)\,\mathrm{m\,s^{-1}},$$

$$\theta_i = 90^\circ,\ \omega_i = 0^\circ\mathrm{s^{-1}},\ r_f = (0, 0)\,\mathrm{m},\ v_f = (0, 0)\,\mathrm{m\,s^{-1}},$$

$$\theta_f = 0^\circ,\ \omega_f = 0^\circ\mathrm{s^{-1}},\ s_{\min} = 0.6\,\mathrm{s},\ s_{\max} = 10\,\mathrm{s}$$

Averaged over 100 full SeCO solves, we report a mean run-time of the PIPG solver (to solve the entire nonconvex problem) of 13.7 ms. In comparison, ECOS[†] requires 37.1 ms, on average, to solve the problem. For the comparison, we assess the quality of the converged solutions in terms of the difference in propellant consumption (0.02%) and the number of SeCO iterations required to converge (7). A real-time guidance solution obtained via PIPG is shown in Figure 6.1. Here, we observe that the optimizer chooses to initiate the powered-descent phase at an altitude of 490.34 m and a speed of 86.28 $\mathrm{m\,s^{-1}}$ (which is very close to the terminal velocity). Further, we note that the numerous prototype flight tests and independent analyses corroborate many of our observations [56].

[†]For this problem, ECOS is faster than both MOSEK and GUROBI.

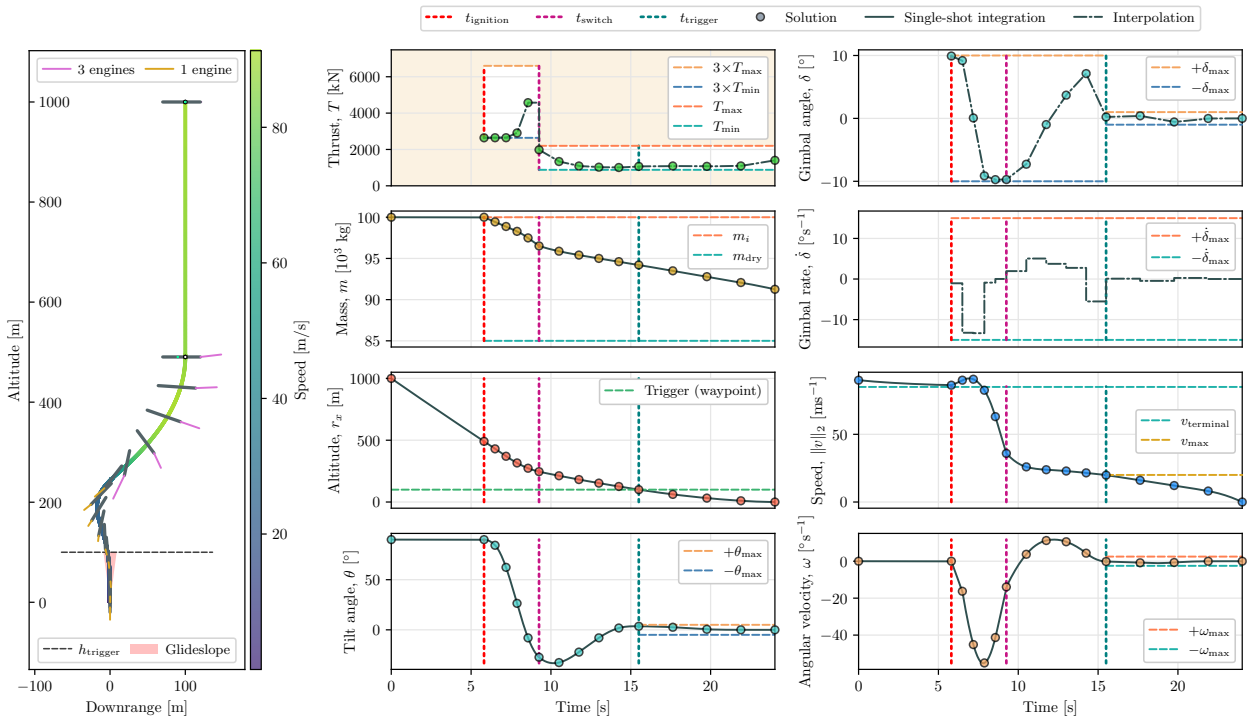

Figure 6.1: A real-time multi-phase rocket landing guidance solution obtained via SeCO**.**

6.2 Dual Quaternion-based 6-DoF Powered-Descent Guidance

In this section, we formulate the dual quaternion-based 6-DoF rocket landing guidance problem in compliance with the SeCO framework, and provide offline and onboard benchmarking results.

6.2.1 Equations of Motion

One of the defining characteristics of DQG is the representation of the 6-DoF equations of motion using unit dual quaternions, which yield an elegant parameterization of the dynamics by naturally coupling the translational and rotational states and enabling the representation of certain key operational constraints, such as the line-of-sight constraint, as convex constraints (in theory) [118, 176]. However, the use of the dual quaternion parameterization is ultimately a design choice, and other parameterizations can be adopted as well, such as Cartesian coordinates for the translational states and unit quaternions for attitude [208]. The interested reader is referred to [117, 119, 175, 176] for detailed descriptions of parameterizing rigid body dynamics via unit dual quaternions.

States

The state vector is 15-dimensional, and consists of mass, m, the 8-dimensional unit dual quaternion that couples translation and attitude, $\boldsymbol{q}$, and the (reduced-order) 6-dimensional dual velocity, $\boldsymbol{\omega}$, as shown in Equations (6.17e). The vector $\widetilde{\boldsymbol{\omega}}$ represents the 8-dimensional dual velocity, in which the fourth and eighth terms are zero; q is the attitude (unit) quaternion. In this work, we adopt the scalar-last convention to represent quaternions. See the Appendix for definitions of unit quaternions and unit dual quaternions. The subscripts $\mathcal{I}$ and $\mathcal{B}$ denote that the quantity in question is expressed in the inertial frame or the body frame, respectively. Further, $r \in \mathbb{R}^3$ is the position, $\omega \in \mathbb{R}^3$ is the angular velocity, and

$v \in \mathbb{R}^3$ is the velocity; the aforementioned representations are summarized as follows:

$$m \in \mathbb{R} \tag{6.17a}$$

$$\boldsymbol{q} := \begin{pmatrix} q \\ \frac{1}{2}\begin{pmatrix} r_{\mathcal{I}} \\ 0 \end{pmatrix} \otimes q \end{pmatrix} = \begin{pmatrix} q \\ \frac{1}{2} q \otimes \begin{pmatrix} r_{\mathcal{B}} \\ 0 \end{pmatrix} \end{pmatrix} \in \mathbb{R}^8 \tag{6.17b}$$

$$\widetilde{\boldsymbol{\omega}} := \begin{pmatrix} \begin{pmatrix} \omega_{\mathcal{B}} \\ 0 \end{pmatrix} \\ q^* \otimes \begin{pmatrix} v_{\mathcal{I}} \\ 0 \end{pmatrix} \otimes q \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \omega_{\mathcal{B}} \\ 0 \end{pmatrix} \\ \begin{pmatrix} v_{\mathcal{B}} \\ 0 \end{pmatrix} \end{pmatrix} \in \mathbb{R}^8 \tag{6.17c}$$

$$\boldsymbol{\omega} := \begin{pmatrix} \omega_{\mathcal{B}} \\ v_{\mathcal{B}} \end{pmatrix} \in \mathbb{R}^6 \tag{6.17d}$$

$$x := \begin{pmatrix} m \\ \boldsymbol{q} \\ \boldsymbol{\omega} \end{pmatrix} \in \mathbb{R}^{15} \tag{6.17e}$$

Controls

The control input vector is 6-dimensional, with three parameters describing the thrust vector: the thrust magnitude, $T \in \mathbb{R}$, the gimbal deflection angle, $\delta \in \mathbb{R}$, and the gimbal azimuth angle, $\phi \in \mathbb{R}$, and a 3-dimensional body torque vector, $\tau \in \mathbb{R}^3$, as shown in Equations (6.18). The thrust vector is effected by means of a gimbaled main engine and the torque input is assumed to be effected by means of reaction control system (RCS) thrusters.

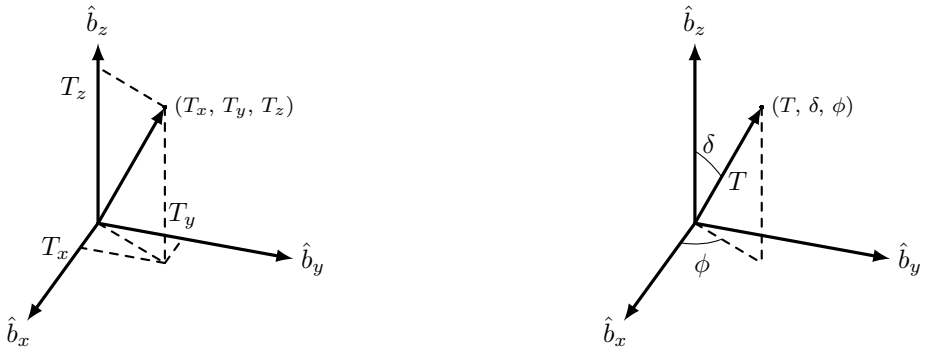

Figure 6.2: Parameterization of the thrust vector (expressed in the body frame) in terms of Cartesian coordinates (left) and spherical coordinates (right). In this work, we use the latter.

We choose to parameterize the thrust vector in terms of spherical coordinates, as shown in Figure 6.2, for the following reasons: (1) all the control constraints in DQG become naturally convex, and hence, in combination with the first-order hold (FOH) parameterization (described in §2.1.2), intersample satisfaction of the control constraints is guaranteed; (2) the control rate constraints can be imposed exactly; and, (3) with a mild assumption, both the magnitude and rate constraints (throttle and gimbaling) can be combined and made projection-friendly, which is beneficial in terms of implementing the solver, as described in §2.2.1. Further, with the spherical coordinate parameterization, we note that the thrust magnitude solution possesses a piecewise-affine profile, which will not be the case if the Cartesian coordinate parameterization is adopted; these parameterizations are summarized as follows:

$$\mathcal{T}_{\mathcal{B}} = \begin{pmatrix} T_x \\ T_y \\ T_z \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{pmatrix} T\sin\delta\cos\phi \\ T\sin\delta\sin\phi \\ T\cos\delta \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} \in \mathbb{R}^6 \tag{6.18a}$$

$$u = \begin{pmatrix} T \\ \delta \\ \phi \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} \in \mathbb{R}^6 \tag{6.18b}$$

T: thrust magnitude

δ: gimbal deflection angle defined from the body vertical (z) axis

ϕ: gimbal azimuth angle defined from the body x-axis

τ_x, τ_y, τ_z: body torque inputs

Here, $\mathcal{T}_{\mathcal{B}}$ is the wrench vector expressed in the body frame [207], and u is the control input vector. The convexity and simplicity of the resulting control constraints come at the cost of additional trigonometric nonlinearities in the dynamics, as shown in Equation (6.18a).

Mass-Depletion

In addition to accounting for thrust due to the main engine, we consider the effect of thrust due to the RCS thrusters on mass-depletion. In order to do so, we assume that two diagonally opposite RCS thrusters fire at any given instant to achieve the desired net torque, such that the thrust due to each thruster is orthogonal to the body vertical (z) axis.

Assuming the mass-center of the vehicle is equidistant from the top-mounted RCS thrusters and the bottom-mounted RCS thrusters/gimbaled main engine, the mass-depletion dynamics can be given by Equation (6.19):

$$\dot{m}(t) = -\left(\alpha_{\text{ME}}\, T(t) + \alpha_{\text{RCS}} \frac{\|\tau(t)\|_2}{l_{\text{CM}}} \right) \tag{6.19}$$

Here, $\alpha_{\text{ME}} \in \mathbb{R}_+$ and $\alpha_{\text{RCS}} \in \mathbb{R}_+$ are the thrust-specific fuel consumption (TSFC) parameters for the main engine and an RCS thruster, respectively, and $l_{\text{CM}} \in \mathbb{R}_{++}$ is the length of the

moment-arm of the vehicle, i.e., the distance between the mass-center of the vehicle and the bottom-mounted RCS thrusters/gimbaled main engine.

Kinematics

This dual quaternion kinematic equation requires the use of the 8-dimensional dual velocity vector, where the fourth and eighth terms are zero, and is given by Equation (6.20):

$$\dot{\boldsymbol{q}}(t) = \frac{1}{2}\boldsymbol{q}(t) \otimes \widetilde{\boldsymbol{\omega}}(t) \tag{6.20}$$

Dynamics

For the dynamics, we assume that the moment of inertia is a linear function of the vehicle mass (as opposed to assuming a constant value as in [176]), and thereby account for the effect of mass-depletion on the attitude of the vehicle. The moment of inertia can be assumed to be an affine function of the vehicle mass as well, if required [177]. The dynamics can be given by Equation (6.21):

$$\dot{\boldsymbol{\omega}}(t) = \boldsymbol{J}(t)^{-1}\left[\begin{pmatrix} 0_{3\times3} & -\omega_{\mathcal{B}}^{\times}(t) \\ -\omega_{\mathcal{B}}^{\times}(t) & 0_{3\times3} \end{pmatrix}\boldsymbol{J}(t)\boldsymbol{\omega}(t) + \Phi\mathcal{T}_{\mathcal{B}}(t) + m(t)\boldsymbol{g}_{\mathcal{B}}(t)\right] \tag{6.21}$$

where

$$\boldsymbol{J}(t) := \left(\begin{array}{c|c} 0_{3\times3} & m(t)I_3 \\ \hline m(t)J & 0_{3\times3} \end{array}\right)_{6\times6} \qquad \Phi := \left(\begin{array}{c|c} I_3 & 0_{3\times3} \\ \hline l^{\times} & I_3 \end{array}\right)_{6\times6} \qquad \boldsymbol{g}_{\mathcal{B}}(t) := \left(\begin{array}{c} g_{\mathcal{B}}(t) \\ \hline 0_{3\times1} \end{array}\right)_{6\times1}$$

$$g_{\mathcal{I}} := [0,\, 0,\, -g]^{\top}$$

$$\begin{pmatrix} g_{\mathcal{B}}(t) \\ 0 \end{pmatrix} = q^{*}(t) \otimes \begin{pmatrix} g_{\mathcal{I}} \\ 0 \end{pmatrix} \otimes q(t)$$

where $J \in \mathbb{S}^3_{++}$ is the inertia tensor of the vehicle about its mass-center, $l = [0, 0, -l_{\text{CM}}]^{\top}$ is the body-fixed moment-arm vector, and $g \in \mathbb{R}_+$ is the acceleration due to gravity at the celestial body under consideration.

6.2.2 Control Constraints

All components of the thrust vector are bounded, as shown in Equations (6.22), where $T_{\min} \in \mathbb{R}_{++}$ and $T_{\max} \in \mathbb{R}_{++}$ are the lower and upper bounds on the thrust magnitude, respectively, and $\delta_{\max} \in \mathbb{R}_{++}$ is the upper bound on the gimbal deflection angle. Further, they are rate-limited, as shown in Equations (6.23), where $\dot{T}_{\max} \in \mathbb{R}_{++}$, $\dot{\delta}_{\max} \in \mathbb{R}_{++}$, and $\dot{\phi}_{\max} \in \mathbb{R}_{++}$ are the rate-limits on the thrust magnitude, the gimbal deflection angle, and the gimbal azimuth angle, respectively. An upper bound is levied on the magnitude of the body torque input about each body axis, as shown in (6.24). We emphasize that every single control constraint is naturally convex, owing to the spherical coordinate parameterization.

Thrust Vector Bounds

$$T_{\min} \leq T(t) \leq T_{\max} \tag{6.22a}$$

$$0 \leq \delta(t) \leq \delta_{\max} \tag{6.22b}$$

$$0 \leq \phi(t) \leq 2\pi \tag{6.22c}$$

Thrust Vector Rate-Limits

$$\left|\dot{T}(t)\right| \leq \dot{T}_{\max} \tag{6.23a}$$

$$\left|\dot{\delta}(t)\right| \leq \dot{\delta}_{\max} \tag{6.23b}$$

$$\left|\dot{\phi}(t)\right| \leq \dot{\phi}_{\max} \tag{6.23c}$$

Torque Bounds

$$\left\|\tau(t)\right\|_{\infty} \leq \tau_{\max} \tag{6.24}$$

6.2.3 State Constraints

We classify state constraints into global state constraints, state-triggered constraints, initial condition constraints, and terminal condition constraints, which we describe in this subsection.

Global State Constraints

Global state constraints involve constraints that are imposed on the state over the entire time-horizon. These constraints include a maximum tilt constraint, a maximum angular body rate constraint, a maximum speed constraint, and a minimum altitude constraint, as shown in Equations (6.25), respectively:

$$\forall t \in [0, t_f)\,,$$

$$\left\|\boldsymbol{q}^{[1:2]}(t)\right\|_2 \leq \sin\frac{\theta_{\max}}{2} \tag{6.25a}$$

$$\left\|\boldsymbol{\omega}^{[1:3]}(t)\right\|_{\infty} \leq \omega_{\max} \tag{6.25b}$$

$$\left\|\boldsymbol{\omega}^{[4:6]}(t)\right\|_2 \leq v_{\max} \tag{6.25c}$$

$$\boldsymbol{q}(t)^{\top} M_g\, \boldsymbol{q}(t) \geq h_{\min} \tag{6.25d}$$

where

$$M_g := \begin{pmatrix} \mathbf{0}_{4\times4} & [\widetilde{z}_{\mathcal{I}}]_{\otimes}^{\top} \\ [\widetilde{z}_{\mathcal{I}}]_{\otimes} & \mathbf{0}_{4\times4} \end{pmatrix}$$

Here, $t_f \in \mathbb{R}_+$ is the time-of-flight, $\theta_{\max} \in \mathbb{R}_+$ is the maximum tilt angle from the inertial vertical (z) axis, $\omega_{\max} \in \mathbb{R}_+$ is the maximum angular speed about any body axis, $v_{\max} \in \mathbb{R}_+$ is the maximum speed, $h_{\min} \in \mathbb{R}_+$ is the minimum altitude, $z_{\mathcal{I}} := [0, 0, 1]^{\top}$, and $\widetilde{z}_{\mathcal{I}} := [z_{\mathcal{I}}^{\top}, 0]^{\top}$ (pure quaternion).

State-Triggered Constraints

State-triggered constraints (STCs) include the constraints that are to be activated only when the vehicle is within the prescribed trigger window. Here, we consider slant-range-based triggering, and tightly constrain the body tilt angle, angular body rates, maximum speed, and the maximum line-of-sight angle to the target landing site—which is assumed to be at the origin, without loss of generality—as shown in Equations (6.26), respectively:

$$\forall t \ni \rho_{\min} \leq \left\| 2\,\boldsymbol{q}^{[5:8]}(t) \right\|_2 \leq \rho_{\max},$$

$$\left\| \boldsymbol{q}^{[1:2]}(t) \right\|_2 \leq \sin \frac{\theta_{\mathrm{STC}_{\max}}}{2} \tag{6.26a}$$

$$\left\| \boldsymbol{\omega}^{[1:3]}(t) \right\|_{\infty} \leq \omega_{\mathrm{STC}_{\max}} \tag{6.26b}$$

$$\left\| \boldsymbol{\omega}^{[4:6]}(t) \right\|_2 \leq v_{\mathrm{STC}_{\max}} \tag{6.26c}$$

$$\boldsymbol{q}(t)^{\top} M_l\, \boldsymbol{q}(t) + \left\| 2\,\boldsymbol{q}^{[5:8]}(t) \right\|_2 \cos \mu_{\mathrm{STC}_{\max}} \leq 0 \tag{6.26d}$$

where

$$M_l := \begin{pmatrix} \mathbf{0}_{4\times4} & [\widetilde{p}_{\mathcal{B}}]_{\otimes}^{*\top} \\ [\widetilde{p}_{\mathcal{B}}]_{\otimes}^{*} & \mathbf{0}_{4\times4} \end{pmatrix}$$

Here, $\rho_{\max} \in \mathbb{R}_{++}$ and $\rho_{\min} \in \mathbb{R}_{++}$ are the maximum (activation) and minimum (deactivation) trigger distances from the target landing site, respectively; $\theta_{\text{STC}_{\max}}$, $\omega_{\text{STC}_{\max}}$, and $v_{\text{STC}_{\max}}$ are assumed to be smaller than their counterparts in Equations (6.25); $\mu_{\text{STC}_{\max}} \in \mathbb{R}_{+}$ is the maximum line-of-sight angle; and $\widetilde{p}_{\mathcal{B}} := [p_{\mathcal{B}}^{\top}, 0]^{\top}$ (pure quaternion), where $p_{\mathcal{B}} \in \mathbb{R}^3$ is a unit vector in the body frame that represents the body-fixed sensor-pointing direction. We choose not to impose the minimum altitude constraint in the trigger window, with the observation that the simultaneous satisfaction of the maximum tilt and maximum line-of-sight angle constraints implicitly precludes subsurface solutions if $\theta_{\text{STC}_{\max}} \leq \frac{\pi}{2} - \mu_{\text{STC}_{\max}} - \gamma_{\text{boresight}}$, where $\gamma_{\text{boresight}}$ is the angle made by the body-fixed sensor with the body vertical (z) axis, i.e., the (acute) angle between $p_{\mathcal{B}}$ and $\hat{b}_z$.

These constraints are imposed to enable accurate scans of the potential landing site during descent using the hazard detection LiDAR (HDL) [172, 173], for instance, and to initiate diverts if necessary.

Initial Conditions

The initial condition constraints are given by Equations (6.27):

$$m(0) = m_i \tag{6.27a}$$

$$\boldsymbol{q}(0) = \begin{pmatrix} q_i \\ \frac{1}{2}\begin{pmatrix} r_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \tag{6.27b}$$

$$\boldsymbol{\omega}(0) = \begin{pmatrix} \omega_{\mathcal{B}_i} \\ q_i^* \otimes \begin{pmatrix} v_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \tag{6.27c}$$

where $m_i \in \mathbb{R}_{++}$ is the initial mass of the vehicle, $q_i \in \mathbb{R}_u^4$ is the initial attitude quaternion, $r_{\mathcal{I}_i} \in \mathbb{R}^3$ is the initial position expressed in the inertial frame, $\omega_{\mathcal{B}_i} \in \mathbb{R}^3$ is the initial angular velocity expressed in the body frame, and $v_{\mathcal{I}_i} \in \mathbb{R}^3$ is the initial velocity expressed in the

inertial frame.

Terminal Conditions

The terminal conditions subsume the following details—at the final time t_f: (1) the vehicle is upright (zero pitch and yaw); (2) the roll is free; (3) the angular body rates are zero; and, (4) the (inertial) horizontal components of velocity are zero. By infusing these details into the expressions rather than treating them as problem parameters, the terminal condition constraints are rendered convex. Here, $m_f \in \mathbb{R}_{++}$ is the final mass of the vehicle, $r_{\mathcal{I}_f} \in \mathbb{R}^3$ is the final position expressed in the inertial frame, and $v_{z_{\mathcal{I}_f}} \in \mathbb{R}$ is the final velocity along the inertial vertical (z) axis. If desired, the entire final dual quaternion can be fixed as well. The terminal condition constraints are given by Equations (6.28):

$$m(t_f) \geq m_f \tag{6.28a}$$

$$\boldsymbol{q}^{[1:2]}(t_f) = \mathbf{0}_{2\times 1} \tag{6.28b}$$

$$\begin{bmatrix} \left(\frac{1}{2}[r_{\mathcal{I}_f}]_{\otimes}\begin{pmatrix}\mathbf{0}_{2\times 2}\\ I_2\end{pmatrix}\right) & -I_4 \end{bmatrix}_{4\times 6} \boldsymbol{q}^{[3:8]}(t_f) = \mathbf{0}_{4\times 1} \tag{6.28c}$$

$$\widetilde{\boldsymbol{\omega}}(t_f) = \left(\begin{pmatrix}\mathbf{0}_{2\times 1}\\ q^{[3:4]}(t_f)\end{pmatrix}^* \otimes \begin{pmatrix}\mathbf{0}_{4\times 1}\\ \mathbf{0}_{2\times 1}\\ v_{z_{\mathcal{I}_f}}\\ 0\end{pmatrix} \otimes \begin{pmatrix}\mathbf{0}_{2\times 1}\\ q^{[3:4]}(t_f)\end{pmatrix}\right) = \begin{pmatrix}\mathbf{0}_{4\times 1}\\ \mathbf{0}_{2\times 1}\\ v_{z_{\mathcal{I}_f}}\\ 0\end{pmatrix} \tag{6.28d}$$

$$\therefore \boldsymbol{\omega}(t_f) = \begin{pmatrix}\mathbf{0}_{5\times 1}\\ v_{z_{\mathcal{I}_f}}\end{pmatrix} \tag{6.28e}$$

6.2.4 The Continuous-Time Nonconvex Optimal Control Problem

$$
\begin{aligned}
&\underset{t_f,\, u(t)}{\text{minimize}} && -m(t_f) \\
&\text{subject to} && \forall t \in [0, t_f) \\
&\boxed{\text{Dynamics}} && \dot{x}(t) = f(t, x(t), u(t)) \\
&\boxed{\text{Control constraints}} && T_{\min} \leq T(t) \leq T_{\max} \\
& && 0 \leq \delta(t) \leq \delta_{\max} \\
& && 0 \leq \phi(t) \leq 2\pi \\
& && \left|\dot{T}(t)\right| \leq \dot{T}_{\max} \\
& && \left|\dot{\delta}(t)\right| \leq \dot{\delta}_{\max} \\
& && \left|\dot{\phi}(t)\right| \leq \dot{\phi}_{\max} \\
& && \left\|\tau(t)\right\|_\infty \leq \tau_{\max} \\
&\boxed{\text{Global state constraints}} && \left\|\boldsymbol{q}^{[1:2]}(t)\right\|_2 \leq \sin\frac{\theta_{\max}}{2} \\
& && \left\|\boldsymbol{\omega}^{[1:3]}(t)\right\|_\infty \leq \omega_{\max} \\
& && \left\|\boldsymbol{\omega}^{[4:6]}(t)\right\|_2 \leq v_{\max} \\
& && \boldsymbol{q}(t)^\top M_g\, \boldsymbol{q}(t) \geq h_{\min} \\
&\boxed{\text{State-triggered constraints}} && \left\|\boldsymbol{q}^{[1:2]}(t)\right\|_2 \leq \sin\frac{\theta_{\text{STC}_{\max}}}{2} \\
&\forall t \ni \rho_{\min} \leq \left\|2\,\boldsymbol{q}^{[5:8]}(t)\right\|_2 \leq \rho_{\max} && \left\|\boldsymbol{\omega}^{[1:3]}(t)\right\|_\infty \leq \omega_{\text{STC}_{\max}} \\
& && \left\|\boldsymbol{\omega}^{[4:6]}(t)\right\|_2 \leq v_{\text{STC}_{\max}} \\
& && \boldsymbol{q}(t)^\top M_l\, \boldsymbol{q}(t) + \left\|2\,\boldsymbol{q}^{[5:8]}(t)\right\|_2 \cos\mu_{\text{STC}_{\max}} \leq 0 \\
&\boxed{\text{Initial conditions}} && m(0) = m_i \\
& && \boldsymbol{q}(0) = \begin{pmatrix} q_i \\ \frac{1}{2}\begin{pmatrix} r_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \\
& && \boldsymbol{\omega}(0) = \begin{pmatrix} \omega_{\mathcal{B}_i} \\ q_i^* \otimes \begin{pmatrix} v_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \\
&\boxed{\text{Terminal conditions}} && m(t_f) \geq m_f \\
& && \boldsymbol{q}^{[1:2]}(t_f) = \mathbf{0}_{2\times 1} \\
& && \begin{bmatrix} \left(\frac{1}{2}[r_{\mathcal{I}_f}]_\otimes \begin{pmatrix} \mathbf{0}_{2\times 2} \\ I_2 \end{pmatrix}\right) & -I_4 \end{bmatrix}_{4\times 6} \boldsymbol{q}^{[3:8]}(t_f) = \mathbf{0}_{4\times 1} \\
& && \boldsymbol{\omega}(t_f) = \begin{pmatrix} \mathbf{0}_{5\times 1} \\ v_{z_{\mathcal{I}_f}} \end{pmatrix}
\end{aligned}
$$

The continuous-time nonconvex optimal control problem is given by §6.2.4, where we minimize propellant consumption (by maximizing the final mass of the vehicle).

6.2.5 Constraint Reformulations

We choose to *combine* certain intersecting path constraints for the following reasons: (1) to reduce the number of constraints imposed (and in turn, reduce the number of operations that the solver needs to carry out); and, (2) to ensure that all the path constraint sets possess closed-form projection operations. Closed-form expressions only exist for the projection onto the intersection of convex sets in special cases—such as the intersection of a cone and a ball, and the intersection of two halfspaces—but this is not the case in general, even if the individual constraint sets can be projected onto [15]. Although iterative methods such as the alternating direction method of multipliers (ADMM) [39] can be used to compute projections onto the intersection of such convex sets, we opt to reformulate the intersecting constraint sets so as to enable closed-form projections instead.

Combined Thrust Vector Constraints

Given the FOH parameterization of the control input, the thrust vector magnitude constraints, in discrete-time, can be expressed as shown in Equations (6.29):

$$T_{\min} \leq T_k \leq T_{\max}, \quad k = 1:N \tag{6.29a}$$

$$0 \leq \delta_k \leq \delta_{\max}, \quad k = 1:N \tag{6.29b}$$

$$0 \leq \phi_k \leq 2\pi, \quad k = 1:N \tag{6.29c}$$

where N is the size of the chosen temporal grid, i.e., the number of discrete temporal nodes in the discrete-time subproblem. Further, the thrust vector rate constraints can be expressed

as shown in Equations (6.30):

$$-\dot{T}_{\max} \leq \frac{T_k - T_{k-1}}{(\frac{s}{N-1})} \leq \dot{T}_{\max}, \quad k = 2\!:\!N \tag{6.30a}$$

$$-\dot{\delta}_{\max} \leq \frac{\delta_k - \delta_{k-1}}{(\frac{s}{N-1})} \leq \dot{\delta}_{\max}, \quad k = 2\!:\!N \tag{6.30b}$$

$$-\dot{\phi}_{\max} \leq \frac{\phi_k - \phi_{k-1}}{(\frac{s}{N-1})} \leq \dot{\phi}_{\max}, \quad k = 2\!:\!N \tag{6.30c}$$

which are exact, owing to the FOH parameterization [146]. The variable s is the dilation factor, and $\frac{s}{N-1}$ is the length of each of the uniformly spaced time-intervals.

These thrust vector magnitude and rate constraints can be combined by means of an approximation leveraging the reference solution, as shown in §6.2.7, such that T_k, δ_k, and ϕ_k, $k = 1\!:\!N$, are the sole decision variables. This form of the constraints enables closed-form projections, ensures that the original thrust vector constraints are never violated, and is exact at convergence.

Combined State Constraints

We propose a new approach to modeling state-triggered constraints (STCs) [207, 208, 177, 135] that allows for the combination of the global state constraints and the STCs, thus avoiding intersecting constraint sets on the state variables and, in turn, enabling closed-form projection operations onto the constraint sets. For the dual quaternion variable specifically, we reformulate the constraints so as to enable (closed-form) projections onto the intersection of halfspaces.

With the assumption that the global state bound on any given variable, $\square_{\max}$, is greater than its STC counterpart, $\square_{\text{STC}_{\max}}$, i.e., $\square_{\max} > \square_{\text{STC}_{\max}}$, we observe that the bounds can be expressed in a single expression as follows:

$$g(\square) \leq \max\{-\psi(t)\,\square_{\max},\, \square_{\text{STC}_{\max}}\} \tag{6.31}$$

where $g(\cdot)$ is the constraint function under consideration. The trigger function, $\psi(t)$, given by Equation (6.32):

$$\psi(t) := \operatorname{sgn}\left\{\left(\rho_{\max} - \left\|2\,\boldsymbol{q}^{[5:8]}(t)\right\|_2\right)\left(\left\|2\,\boldsymbol{q}^{[5:8]}(t)\right\|_2 - \rho_{\min}\right)\right\} \ni \psi : [0, t_f) \to \{-1,\, 0,\, +1\} \tag{6.32}$$

takes the value -1 if the vehicle is outside the trigger window, $+1$ if the vehicle is inside the trigger window, and 0, if the vehicle is at either of the triggers. With this formulation, the RHS of Equation (6.31) can automatically switch between the global bound and the STC bound based on the value that the trigger function assumes.

Further, an approximation, similar to the one made with the thrust vector constraints, is made to the trigger function, and the global state constraints and the STCs are combined, as shown in §6.2.7. Note that the combined state constraints are exact at convergence.

6.2.6 Projections

As shown in the discrete-time conic subproblem in §6.2.7, which is a second-order cone program (SOCP), every single path constraint admits a closed-form projection operation. The constraint sets listed in green possess direct closed-form projection operations. The ones listed in blue and yellow involve closed-form projections onto the intersection of halfspaces; the maximum tilt constraint is linearized to enable that. The interested reader is referred to [17] for a description of these closed-form projection operations.

6.2.7 *The Discretized Conic Subproblem*

$$\underset{s,\,u_{[1:N]}}{\text{minimize}} \quad \underbrace{-w_m\, m_N}_{\text{cost term}} + \underbrace{\frac{1}{2}\left(w_{\text{tr}} \sum_{k=1}^{N} \left(\|x_k - \overline{x}_k\|_2^2 + \|u_k - \overline{u}_k\|_2^2\right) + w_{\text{tr}_s}\|s - \overline{s}\|_2^2\right)}_{\text{soft trust region term}} + \underbrace{\frac{1}{2}\, w_{\text{vse}} \sum_{k=1}^{N} \|x_k - \xi_k\|_2^2}_{\text{virtual state penalty term}}$$

subject to **Dynamics**

zero cone $x_{k+1} = A_k\, x_k + B_k^-\, u_k + B_k^+\, u_{k+1} + S_k\, s \; + d_k$ $(\mathbb{K}_0)$ $k = 1\!:\!N-1$

Combined control constraints

box $T_{\min} \le T_1 \le T_{\max}$ $(\mathbb{D})$

box $0 \le \delta_1 \le \delta_{\max}$ $(\mathbb{D})$

box $0 \le \phi_1 \le 2\pi$ $(\mathbb{D})$

box $\max\left(T_{\min},\, -\dot{T}_{\max}\dfrac{\overline{s}}{N-1} + \overline{T}_{k-1}\right) \le T_k \le \min\left(T_{\max},\, \dot{T}_{\max}\dfrac{\overline{s}}{N-1} + \overline{T}_{k-1}\right)$ $(\mathbb{D})$ $k = 2\!:\!N$

box $\max\left(0,\, -\dot{\delta}_{\max}\dfrac{\overline{s}}{N-1} + \overline{\delta}_{k-1}\right) \le \delta_k \le \min\left(\delta_{\max},\, \dot{\delta}_{\max}\dfrac{\overline{s}}{N-1} + \overline{\delta}_{k-1}\right)$ $(\mathbb{D})$ $k = 2\!:\!N$

box $\max\left(0,\, -\dot{\phi}_{\max}\dfrac{\overline{s}}{N-1} + \overline{\phi}_{k-1}\right) \le \phi_k \le \min\left(2\pi,\, \dot{\phi}_{\max}\dfrac{\overline{s}}{N-1} + \overline{\phi}_{k-1}\right)$ $(\mathbb{D})$ $k = 2\!:\!N$

box $\|\tau_k\|_\infty \le \tau_{\max}$ $(\mathbb{D})$ $k = 1\!:\!N$

Combined state constraints

box $\left\|{}^{\xi}\boldsymbol{\omega}_k^{[1:3]}\right\|_\infty \le \max\left(-\overline{\psi}_k\, \omega_{\max},\, \omega_{\text{STC}_{\max}}\right)$ $(\mathbb{D})$ $k = 2\!:\!N-1$

ball $\left\|{}^{\xi}\boldsymbol{\omega}_k^{[4:6]}\right\|_2 \le \max\left(-\overline{\psi}_k\, v_{\max},\, v_{\text{STC}_{\max}}\right)$ $(\mathbb{D})$ $k = 2\!:\!N-1$

halfspace $\overline{\boldsymbol{q}}_k^{[1:2]\top}\,{}^{\xi}\boldsymbol{q}_k^{[1:2]} \le \left\|\overline{\boldsymbol{q}}_k^{[1:2]}\right\|_2 \sin\frac{\theta_{\text{STC}_{\max}}}{2}$ $(\mathbb{D})$ $k \ni \overline{\psi}_k \ge 0$

halfspace Linearize Equation (6.26d) (maximum line-of-sight angle) $(\mathbb{D})$ $k \ni \overline{\psi}_k \ge 0$

halfspace $\overline{\boldsymbol{q}}_k^{[1:2]\top}\,{}^{\xi}\boldsymbol{q}_k^{[1:2]} \le \left\|\overline{\boldsymbol{q}}_k^{[1:2]}\right\|_2 \sin\frac{\theta_{\max}}{2}$ $(\mathbb{D})$ $k \ni \overline{\psi}_k < 0$

halfspace Linearize Equation (6.25d) (minimum altitude) $(\mathbb{D})$ $k \ni \overline{\psi}_k < 0$

Boundary conditions

singleton ${}^{\xi}m_1 = m_i$ $(\mathbb{D})$

singleton $${}^{\xi}\boldsymbol{q}_1 = \begin{pmatrix} q_i \\ \frac{1}{2}\begin{pmatrix} r_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix}$$ $(\mathbb{D})$

singleton $${}^{\xi}\boldsymbol{\omega}_1 = \begin{pmatrix} \omega_{\mathcal{B}_i} \\ q_i^* \otimes \begin{pmatrix} v_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix}$$ $(\mathbb{D})$

halfspace ${}^{\xi}m_N \ge m_f$ $(\mathbb{D})$

singleton ${}^{\xi}\boldsymbol{q}_N^{[1:2]} = 0$ $(\mathbb{D})$

subspace $$\begin{bmatrix} \left(\frac{1}{2}[r_{\mathcal{I}_f}]_\otimes \begin{pmatrix} \mathbf{0}_{2\times 2} \\ I_2 \end{pmatrix}\right) & -I_4 \end{bmatrix} {}^{\xi}\boldsymbol{q}_N^{[3:8]} = 0$$ $(\mathbb{D})$

singleton ${}^{\xi}\boldsymbol{\omega}_N = \begin{pmatrix} \mathbf{0}_{5\times 1} \\ v_{z_{\mathcal{I}_f}} \end{pmatrix}$ $(\mathbb{D})$

6.2.8 Results

6.2.9 Offline Benchmarking

We benchmark PIPG$_{\text{custom}}$ against three state-of-the-art convex optimization solvers: ECOS, MOSEK, and GUROBI [59, 152, 86], by means of a lunar approach-phase test case, with a fixed final attitude quaternion, q_f. We use the absolute-variable version of the solver described in [102], which does not include row normalization. The solver parameter, λ, is manually tuned. The PIPG$_{\text{custom}}$ solver is implemented via C code, generated using the MATLAB Coder [139, 138]. The YALMIP convex optimization modeling tool in MATLAB is used to parse the problem and interface with the off-the-shelf solvers [124]. All trials are run on a 2018 MacBook Pro with a 2.6 GHz 6-core Intel Core i7 processor and 16 GB of RAM.

For consistency, the DQG problem instance is set up such that each benchmarked solver solves the problem to a predetermined open-loop accuracy in exactly 5 SeCO iterations. The entire DQG problem is solved 100 times and the mean total (across all SeCO iterations) discretization-, parse-, and solve-times are reported, as shown in Figures 6.6. The same procedure is carried out across 4 different problem sizes, representative of onboard guidance: $N \in \{10, 15, 20, 25\}$, where N is the number of discrete temporal nodes. The terminal position and velocity error tolerances (between the computed solution and the open-loop single-shot integrated trajectory) are set to 10 m and 0.25 m/s, respectively—similar to the tolerances chosen in [176]. Note that it is possible to significantly reduce parsing time for the other solvers for online execution [174], and the desktop parsing times are only reported for completeness; as such, the true performance comparison is between solve-times. The DQG parameter values chosen for the benchmark test are given in Table 6.1. The 3-dimensional landing trajectory and the line-of-sight angle as a function of time (corresponding to $N = 15$), obtained via PIPG$_{\text{custom}}$, are shown in Figures 6.3 and 6.4, respectively.

We observe that the solution framework (SeCO) itself leads to a speedup, regardless of the solver chosen, when compared with previously used SCP methods and solve-times reported in the literature [174, 176, 202]. Further, PIPG is significantly faster than the solvers it is benchmarked against, as shown in Figure 6.5, and over an order of magnitude faster than the previously reported mean solve-time for DQG [176] for the same problem size.

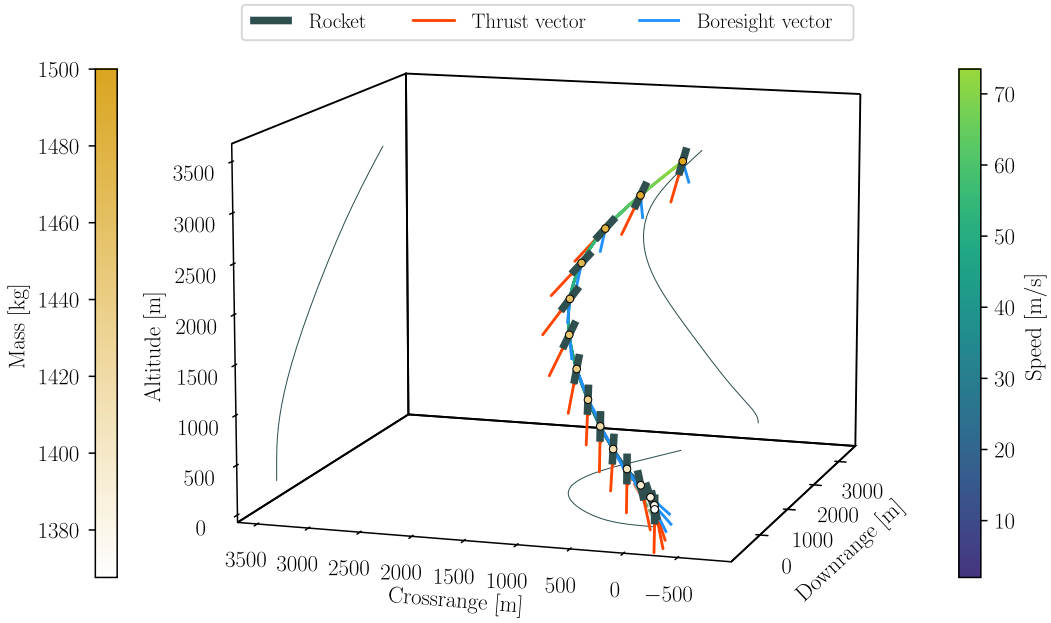

Figure 6.3: The 3D landing trajectory obtained via SeCO in real-time ($N = 15$).

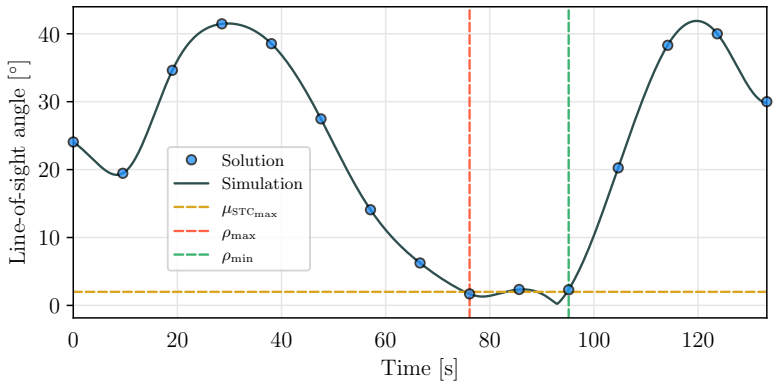

Figure 6.4: The line-of-sight angle, which is constrained to be within 2° in the trigger window ($N = 15$).

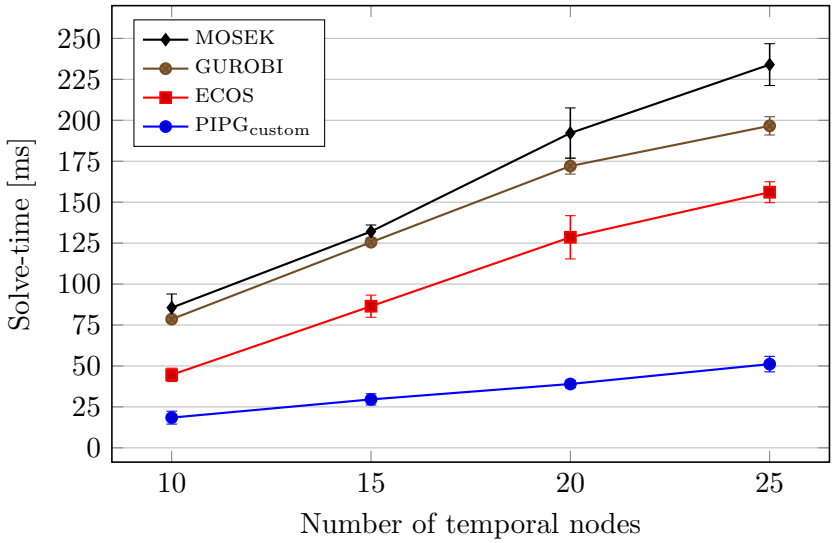

Figure 6.5: Solve-time comparison between the DQG-customized version of PIPG and three state-of-the-art convex optimization solvers. The error bars indicate three standard deviations ($\pm 3\sigma$).

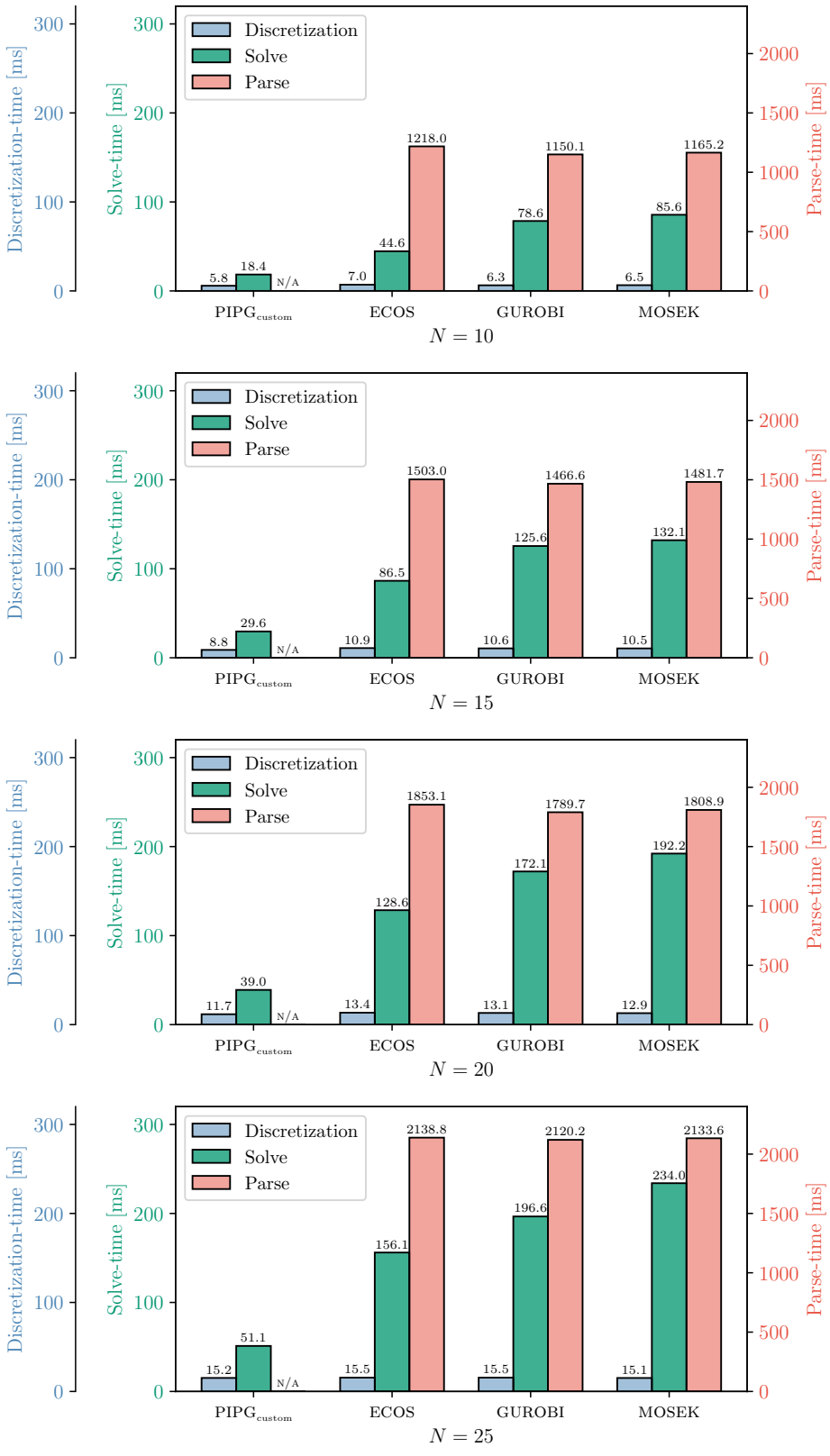

Figure 6.6: DQG benchmark test results.

Parameter	Value
g	$1.625\ \mathrm{m\,s^{-2}}$
g_0	$9.81\ \mathrm{m\,s^{-2}}$
$I_{\mathrm{sp}_{\mathrm{ME}}}$	$300\ \mathrm{s}$
α_{ME}	$\frac{1}{I_{\mathrm{sp}_{\mathrm{ME}}} g_0}\ \mathrm{s\,m^{-1}}$
$I_{\mathrm{sp}_{\mathrm{RCS}}}$	$200\ \mathrm{s}$
α_{RCS}	$\frac{1}{I_{\mathrm{sp}_{\mathrm{RCS}}} g_0}\ \mathrm{s\,m^{-1}}$
$\tau_{\max}$	$50\ \mathrm{kg\,m^2\,s^{-2}}$
$T_{\max}$	$3000\ \mathrm{kg\,m\,s^{-2}}$
$T_{\min}$	$600\ \mathrm{kg\,m\,s^{-2}}$
$\dot{T}_{\max}$	$0.75 \cdot (T_{\max} - T_{\min})\ \mathrm{kg\,m\,s^{-3}}$
$\delta_{\max}$	5°
$\dot{\delta}_{\max}$, $\dot{\phi}_{\max}$	$5^\circ\,\mathrm{s^{-1}}$
l_{CM}	$1\ \mathrm{m}$
$p_{\mathcal{B}}$	$\left[0.5,\ 0,\ -\frac{\sqrt{3}}{2}\right]^\top$
m_i	$1500\ \mathrm{kg}$
m_f	$750\ \mathrm{kg}$
J	$\mathrm{diag}\{4.2,\ 4.2,\ 0.6\}\ \mathrm{m^2}$

Parameter	Value
$\theta_{\max}$	90°
$\omega_{\max}$	$5^\circ\,\mathrm{s^{-1}}$
$v_{\max}$	$90\ \mathrm{m\,s^{-1}}$
$h_{\min}$	$100\ \mathrm{m}$
$\rho_{\max}$	$1250\ \mathrm{m}$
$\rho_{\min}$	$500\ \mathrm{m}$
$\theta_{\mathrm{STC}_{\max}}$	20°
$\omega_{\mathrm{STC}_{\max}}$	$1^\circ\,\mathrm{s^{-1}}$
$v_{\mathrm{STC}_{\max}}$	$30\ \mathrm{m\,s^{-1}}$
$\mu_{\mathrm{STC}_{\max}}$	2°
$r_{\mathcal{I}_i}$	$[3000,\ 600,\ 3000]^\top\ \mathrm{m}$
$r_{\mathcal{I}_f}$	$[0,\ 0,\ 100]^\top\ \mathrm{m}$
$v_{\mathcal{I}_i}$	$[-60,\ 30,\ -30]^\top\ \mathrm{m\,s^{-1}}$
$v_{z_{\mathcal{I}_f}}$	$-2\ \mathrm{m\,s^{-1}}$
q_i	$[-0.15,\ 0.3,\ -1,\ 1]^\top$ (normalized)
q_f	$[0,\ 0,\ -1.25,\ 1]^\top$ (normalized)
$\omega_{\mathcal{B}_i}$	$[0,\ 0,\ 0]^\top\ {}^\circ\,\mathrm{s^{-1}}$

Table 6.1: The DQG parameter values chosen for the solver benchmark test.

6.2.10 Onboard (Hardware-in-the-Loop) Testing

We consider a terrestrial rocket landing mission scenario for an upcoming closed-loop (DQG-in-the-loop) rocket landing flight test campaign [150, 149], and solve the problem, for 100 divert sites on a uniform grid, as shown in Figure 6.7. The custom solver used for this application, based on deviation variables, is detailed in Chapter 2. We perform row-normalization, and manually tune the solver parameter, λ. The problem formulation is identical to Problem 6.2.4, with a few modifications, such as an independent thrust and torque model (without gimbaling of the rocket engine in guidance and with control allocation handled outside of guidance), the inclusion of aerodynamic forces, independent component-wise torque bounds, a state-triggered glideslope constraint to replace the state-triggered tilt constraint, and imposition of the initial condition constraint on the true state (as opposed to the virtual state). See [58] for more details on the problem formulation. The $\mathrm{PIPG}_{\mathrm{custom}}$ solver is implemented via C code, which, again, is generated using the MATLAB Coder [139, 138]. The solver is

executed onboard the NASA SPLICE Descent and Landing Computer (DLC), which consists of a cluster of 4 ARM Cortex A53 processors, on which the flight software runs [184].

For the terrestrial landing scenario considered in [75] for the suborbital flight tests of the Blue Origin New Shepard reusable launch vehicle (with DQG executed in an open-loop, onboard the DLC), with the older SCP algorithm and a customized version of the IPM-based subproblem solver, BSOCP, the 10-node version took an average (over 3 runs) of 2.65 seconds, with all 3 runs taking 4 SCP iterations each.

For the lunar landing scenario considered in [202], with the older SCP algorithm and the IPM-based subproblem solver, BSOCP [63, 62], the 10-node version took an average (over 100 runs) of 5.85 seconds, with 97 runs taking 4 SCP iterations each, and the remaining 3 runs taking 5 SCP iterations each. The 20-node version, on the other hand, took an average (again, over 100 runs) of 11.36 seconds, with 27 runs taking 4 SCP iterations each, and the remaining 73 runs taking 3 SCP iterations each. Neither of these versions met the SPLICE goal of a guidance update-rate of 1 second, or even the SPLICE requirement of a guidance update-rate of 3 seconds.

In contrast, the custom solver proposed in this work, applied to the terrestrial landing scenario considered in [58], with 15 nodes, took an average (over 100 divert scenarios) of 0.5887 seconds, with all 100 runs taking 5 SCP (SeCO) iterations each. Note that this not only meets, but exceeds both the SPLICE requirement of a guidance update-rate of 3 seconds and the SPLICE goal of a guidance update-rate of 1 second, by a significant margin. Further, we note that this marks the first time in the duration of the NASA SPLICE program that the 1-second goal has been achieved onboard the DLC.

These results are presented in Figure 6.8. We note that this does not represent a direct comparison, owing to the differing problem formulations and parameters considered in the preceding tests between the different solvers. That said, given the similarity in mission complexity and the fact that all solvers were executed on the same computing platform (the DLC), we conclude that our proposed solver is roughly 5 to 10 times faster than the old solver.

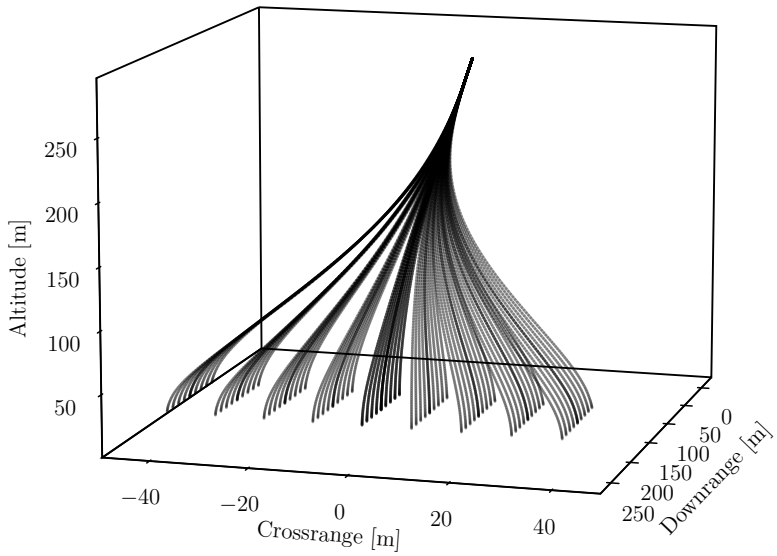

Figure 6.7: Hazard-avoidance divert trajectories computed onboard the NASA SPLICE Descent and Landing Computer (DLC) in a hardware-in-the-loop setting.

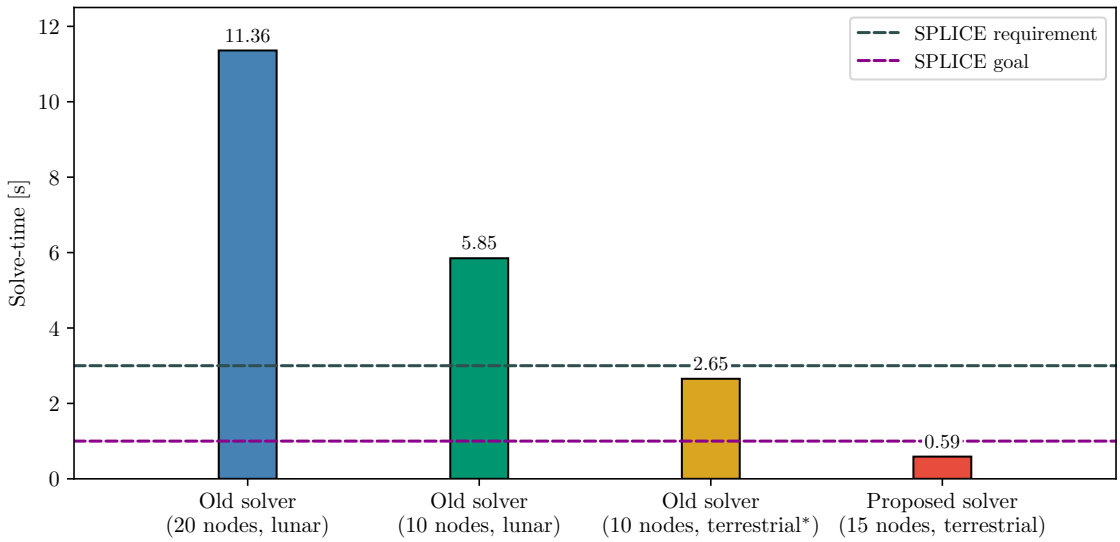

Figure 6.8: Average solve-times onboard the NASA SPLICE Descent and Landing Computer (DLC). The "old solver" refers to the previously-used SCP algorithm [176] with the BSOCP convex subproblem solver [62]. From the left, the first two bars (averaged over 100 runs) are from [202] with generic BSOCP, the third bar (averaged over 3 runs) is from [75] with customized BSOCP (*in-flight), and the rightmost bar (averaged over 100 runs) corresponds to the proposed solver with customized PIPG (this work), which meets both the SPLICE requirement and the SPLICE goal for the guidance update-rate, i.e., solve-time.

6.3 Set-based Control for Autonomous Precision Landing

To demonstrate the controllable tube-based optimal, robust, and resilient control framework that we propose in Chapter 4, we consider an autonomous precision landing case study. We formulate the optimal control problem in §6.3.1, describe the closed-loop simulation setup in §6.3.2, and demonstrate free-final-time optimal control in §6.3.3, robust control in §6.3.4, and resilient control in §6.3.5. We use `pycvxset` [224] for all the set-based computations.

6.3.1 Formulation

The original nonconvex continuous-time 3-DoF minimum-fuel precision landing guidance optimal control problem [2] is given by Problem P10. In Problem P10, t_f is the final time,

Problem P10: Continuous-Time Precision Landing Problem (Nonconvex)

$$
\begin{aligned}
\underset{t_\mathrm{f},\, T}{\text{minimize}} \quad & -m(t_\mathrm{f}) && \text{(P10a)}\\
\text{subject to} \quad & \forall\, t \in [0, t_\mathrm{f}] \\
& \dot{r}(t) = v(t) && \text{(P10b)}\\
& \dot{v}(t) = \frac{1}{m(t)}\, T(t) - g\, e_z && \text{(P10c)}\\
& \dot{m}(t) = -\alpha \left\| T(t) \right\|_2 && \text{(P10d)}\\
& T_\mathrm{min} \leq \left\| T(t) \right\|_2 \leq T_\mathrm{max} && \text{(P10e)}\\
& e_z^\top\, T(t) \geq \left\| T(t) \right\|_2 \cos\theta_\mathrm{max} && \text{(P10f)}\\
& H_\mathrm{GS}\, r(t) \leq h_\mathrm{GS} && \text{(P10g)}\\
& \left\| r(t) \right\|_\infty \leq r_\mathrm{max} && \text{(P10h)}\\
& \left\| v(t) \right\|_\infty \leq v_\mathrm{max} && \text{(P10i)}\\
& m_\mathrm{dry} \leq m(t) \leq m_\mathrm{wet} && \text{(P10j)}\\
& r(0) = r_\mathrm{i},\ \ v(0) = v_\mathrm{i},\ \ m(0) = m_\mathrm{wet} && \text{(P10k)}\\
& r(t_\mathrm{f}) = r_\mathrm{f},\ v(t_\mathrm{f}) = v_\mathrm{f},\ m(t_\mathrm{f}) \geq m_\mathrm{dry} && \text{(P10l)}
\end{aligned}
$$

$r(t) \in \mathbb{R}^3$ is the position, $v(t) \in \mathbb{R}^3$ is the velocity, $m(t) \in \mathbb{R}$ is the vehicle mass, $T(t) \in \mathbb{R}^3$

is the thrust vector in Cartesian coordinates, $g \in \mathbb{R}_{++}$ is the acceleration due to gravity, α is the thrust-specific fuel consumption, $T_{\min}$, $T_{\max} \in \mathbb{R}_{++}$ are the minimum and maximum thrust magnitude bounds, respectively, $e_z^\top := (0, 0, 1)$, $\theta_{\max}$ is the maximum tilt angle from the vertical, $H_{\text{GS}} \in \mathbb{R}^{n_h \times 3}$ and $h_{\text{GS}} \in \mathbb{R}^{n_h}$ are the parameters defining the glideslope constraint (as the intersection of n_h halfspaces; see [134, Equations S10 and S11]), $r_{\max}$, $v_{\max} \in \mathbb{R}_{++}$ are the component-wise bounds on position and velocity, respectively, m_{wet}, $m_{\text{dry}} \in \mathbb{R}_{++}$ are the wet-mass and dry-mass of the vehicle, respectively, r_{i}, $r_{\text{f}} \in \mathbb{R}^3$ are the initial and final conditions for the position, respectively, and v_{i}, $v_{\text{f}} \in \mathbb{R}^3$ are the initial and final conditions for the velocity, respectively. Since this is a three-degree-of-freedom formulation, all quantities are defined relative to a reference frame fixed to the celestial body under consideration.

The autonomous precision landing problem has been extensively studied, and it has been shown that the nonconvex thrust lower bound constraint in Equation (P10e) can be losslessly convexified by means of a convex relaxation, such that the solution to the relaxed problem is a globally optimal solution to the original problem [2, 5, 134]. The thrust pointing constraint given by Equation (P10f) would be nonconvex for $\theta_{\max} > 90°$, but we only require $\theta_{\max} \leq 90°$. Note that the state constraints in Equations (P10h) and (P10i) are only included to ensure that the state is bounded (as required for controllable tube generation). Typically, $r_{\max}$ and $v_{\max}$ are set to large values, however, and are not expected to be active.

Problem P10 can be equivalently written in terms of the mass-normalized thrust by means of a log-mass transformation [2], with $u(t) := \frac{1}{m(t)} T(t)$ and $z(t) := \ln m(t)$. Consequently, the control magnitude bounds become:

$$T_{\min} e^{-z(t)} \leq \|u(t)\|_2 \leq T_{\max} e^{-z(t)} \tag{6.33}$$

and the log-mass dynamics are:

$$\dot{z}(t) = -\alpha \|u(t)\|_2 \tag{6.34}$$

both of which are nonconvex. First, we replace the time-varying bounds in Equation (6.33) with the following time-invariant bounds, such that the resulting constraint is a conservative approximation of the original constraint:

$$\frac{T_{\min}}{m_{\text{dry}}} \leq \|u(t)\|_2 \leq \frac{T_{\max}}{m_{\text{wet}}} \tag{6.35}$$

Note that the lower bound constraint in Equation (6.35) is still nonconvex. For this, we can employ one of the following two remedies:

(i) The standard convex relaxation to convexify the nonconvex lower bound [2, 5, 134], i.e.,

$$\sigma(t) \geq \frac{T_{\min}}{m_{\text{dry}}} \tag{6.36}$$

Note that this does not exactly fit the template of Problem P1, since this serves as a convex relaxation to the original nonconvex constraint. However, when implemented, we observe that the relaxation is tight in practice, i.e., lossless convexification holds [2, 5, 134]. Although we consider a polytopic *inner* approximation to the conic frustum defined by the control magnitude bounds, satisfaction of the polytopic constraint does not necessarily guarantee satisfaction of the original constraint—the lower bound constraint can be violated, since the polytopic approximation we consider is of a convex relaxation to a nonconvex constraint set. This has the adverse consequence of expanding the feasible control space—beyond that of the original control constraint set—in some regions. That said, the degree to which this constraint is violated can be arbitrarily controlled by choosing the number of points to be spread on the unit sphere. Further, this issue can be eliminated entirely by ensuring that the projection of the polytopic approximation onto the non-lifted space is an inner approximation to the original nonconvex set.

(ii) Replacing the lower bound constraint with the following conservative convex constraint:

$$e_z^\top u(t) \geq \frac{T_{\min}}{m_{\text{dry}}} \tag{6.37}$$

With the conservative constraint given by Equation (6.37), the control magnitude, and in turn, the slack variable, are implicitly lower-bounded by $\frac{T_{\min}}{m_{\text{dry}}}$. With Equation (6.37), the problem fits the template of Problem P1, in which case the only nonconvexity stems from the log-mass dynamics (Equation (6.34)). For the discretized version of this template, there exists a blanket lossless convexification guarantee [225]. We consider the conservative convex constraint given by Equation (6.37) in the remainder of this work.

Further, we replace $\|u\|_2$ by σ in the corresponding pointing constraint, $e_z^\top u(t) \geq \|u(t)\|_2 \cos\theta_{\max}$ (even though it is convex), hence converting the second-order cone into an equivalent halfspace, which mitigates the need for an additional polytopic approximation.

The (Mayer) cost function, after the log-mass transformation, can be given by:

$$J := -z(t_{\text{f}}) \tag{6.38}$$

which can be equivalently expressed as a running cost:

$$J = -\int_0^{t_{\text{f}}} \dot{z}(\tau)\, d\tau = -\int_0^{t_{\text{f}}} -\alpha\, \sigma(\tau)\, d\tau = \int_0^{t_{\text{f}}} \alpha\, \sigma(\tau)\, d\tau \tag{6.39}$$

Now, the cost-to-go can be defined as follows:

$$c(t) := \int_t^{t_{\text{f}}} \alpha\, \sigma(\tau)\, d\tau \tag{6.40}$$

Taking the time-derivative of Equation (6.40), we get:

$$\dot{c}(t) = \frac{d}{dt}\int_{t}^{t_\mathrm{f}} \alpha\,\sigma(\tau)\,d\tau = \frac{d}{dt}\int_{t_\mathrm{f}}^{t} -\alpha\,\sigma(\tau)\,d\tau = -\alpha\,\sigma(t) \tag{6.41}$$

From Equation (6.40), we can derive the boundary conditions for this auxiliary dynamical system:

$$c(0) = \int_{0}^{t_\mathrm{f}} \alpha\,\sigma(\tau)\,d\tau \tag{6.42a}$$

$$c(t_\mathrm{f}) = 0 \tag{6.42b}$$

Now, consider the following:

$$z(t_\mathrm{f}) = z(0) - \int_{0}^{t_\mathrm{f}} \alpha\,\sigma(\tau)\,d\tau \tag{6.43a}$$

$$\implies \int_{0}^{t} \alpha\,\sigma(\tau)\,d\tau = z(0) - z(t_\mathrm{f}) \tag{6.43b}$$

Substituting Equation (6.43b) in Equation (4.3a), we get:

$$c(0) = z(0) - z(t_\mathrm{f}) = \ln m_\mathrm{wet} - z(t_\mathrm{f}) \tag{6.44}$$

Collecting Equations (6.41), (6.42b), and (6.44), we have the following auxiliary dynamical system:

$$\dot{c}(t) = -\alpha\,\sigma(t) \tag{6.45a}$$

$$c(0) = \ln m_\mathrm{wet} - z(t_\mathrm{f}) \tag{6.45b}$$

$$c(t_\mathrm{f}) = 0 \tag{6.45c}$$

with the bounds on the cost-to-go state being:

$$0 \leq c(t) \leq \ln m_{\text{wet}} - \ln m_{\text{dry}} = \ln \frac{m_{\text{wet}}}{m_{\text{dry}}} \tag{6.46}$$

Although the right-hand-side (RHS) of Equation (6.45) is identical to the RHS of the log-mass dynamics given by Equation (6.34), Equations (6.45) represent a different system due to the different boundary conditions. In this work, we employ the standard procedure of augmenting the physical system with the auxiliary cost-to-go system, making the resulting system 8-dimensional, which, in turn, leads to 8-dimensional polytopes (CZs) in the controllable tube.

Remark 7. *The similarity between the log-mass dynamics (Equation* (6.34)*) and the cost-to-go dynamics (Equation* (6.45)*) can be exploited to eliminate the log-mass variable and dynamics entirely, and to equivalently rewrite the problem in terms of the cost-to-go variable and dynamics, in which case the polytopes in the controllable tube would be* 7*-dimensional, instead of* 8*-dimensional. Particularly, we observe (and can leverage the fact) that the log-mass and the cost-to-go are related by:*

$$c(t) = z(t) - z(t_{\text{f}}) \tag{6.47}$$

with the bounds on the cost-to-go state given by Equation (6.46)*. Using this formulation would require additional considerations to account for the initial condition of the log-mass at every step; particularly, this would require either (i) taking the intersection of the controllable set with the halfspaces defined by the log-mass bounds (prior to solving the one-step optimal control problem), in which case potential infeasibility can be detected prior to solving the one-step OCP, or (ii) imposing the halfspaces directly in the one-step OCP, in which case potential infeasibility would be detected via solving the one-step OCP itself.*

Given how well CZs scale with problem dimension, we do not observe a marked difference in performance with the formulation resulting from Remark 7, and hence retain the

8-dimensional formulation for simplicity. Note that the cost-to-go state, $c(t)$, serves as a proxy for the (instantaneous) "required fuel" to reach the target.

Finally, we discretize the log-mass-transformed and cost-to-go-augmented problem using a ZOH control parameterization, as described in §4.1.1—the discretized conic optimal control problem is given by Problem P11. The state constraints in Problem P11 are polytopic to

Problem P11: Discrete-Time Precision Landing Problem (Conic)

$$
\begin{aligned}
&\underset{\substack{N,\, c_1,\\ u_1,\ldots,u_{N-1},\\ \sigma_1,\ldots,\sigma_{N-1}}}{\text{minimize}} && c_1 && && \text{(P11a)}\\
&\text{subject to} && (r_{k+1}, v_{k+1}, z_{k+1}, c_{k+1}) = A\cdot(r_k, v_k, z_k, c_k) + B\cdot(u_k, \sigma_k) + d, && k = 1,\ldots,N-1 && \text{(P11b)}\\
& && \|u_k\|_2 \leq \sigma_k && k = 1,\ldots,N-1 && \text{(P11c)}\\
& && \sigma_k \leq \frac{T_{\max}}{m_{\text{wet}}} && k = 1,\ldots,N-1 && \text{(P11d)}\\
& && e_z^\top u_k \geq \frac{T_{\min}}{m_{\text{dry}}} && k = 1,\ldots,N-1 && \text{(P11e)}\\
& && e_z^\top u_k \geq \sigma_k\,\cos\theta_{\max} && k = 1,\ldots,N-1 && \text{(P11f)}\\
& && H_{\text{GS}}\, r_k \leq h_{\text{GS}} && k = 2,\ldots,N-1 && \text{(P11g)}\\
& && \|r_k\|_\infty \leq r_{\max} && k = 2,\ldots,N-1 && \text{(P11h)}\\
& && \|v_k\|_\infty \leq v_{\max} && k = 2,\ldots,N-1 && \text{(P11i)}\\
& && \ln m_{\text{dry}} \leq z_k \leq \ln m_{\text{wet}} && k = 2,\ldots,N-1 && \text{(P11j)}\\
& && 0 \leq c_k \leq \ln\frac{m_{\text{wet}}}{m_{\text{dry}}} && k = 2,\ldots,N-1 && \text{(P11k)}\\
& && r_1 = r_{\text{i}},\ \ v_1 = v_{\text{i}},\ \ z_1 = \ln m_{\text{wet}},\ \ c_1 \leq \ln\frac{m_{\text{wet}}}{m_{\text{dry}}} && && \text{(P11l)}\\
& && r_N = r_{\text{f}},\ v_N = v_{\text{f}},\ z_N \geq \ln m_{\text{dry}},\ c_N = 0 && && \text{(P11m)}
\end{aligned}
$$

begin with. The control constraint set, however, is conic, due to Equation (P11c)—the control input slack constraint that arises as a result of lossless convexification is a 4-dimensional quadratic cone (note that the following applies in both continuous time and in discrete time):

$$\mathcal{U}_{\text{slack}} := \left\{(u,\,\sigma) \in \mathbb{R}^4 \;\middle|\; \|u\|_2 \leq \sigma,\ \sigma \geq 0\right\} \tag{6.48}$$

where $u \in \mathbb{R}^3$ and $\sigma \in \mathbb{R}$. The set is closed and convex, but not compact, and a linear approximation of this set would yield an unbounded set. However, since the controllable

tube generation method using constrained zonotopes requires the control input constraint set to be a bounded polytope, we also consider the upper bound constraint on the control magnitude, which is a halfspace:

$$\mathcal{U}_{\text{UB}} := \left\{ (u,\, \sigma) \in \mathbb{R}^4 \;\middle|\; \sigma \leq \frac{T_{\max}}{m_{\text{wet}}} \right\} \tag{6.49}$$

Now, consider the intersection of the sets given by Equations (6.48) and (6.49), $\mathcal{U}_{\text{slack}}$ and $\mathcal{U}_{\text{UB}}$, respectively:

$$\mathcal{U}_{\text{CQC}} := \mathcal{U}_{\text{slack}} \cap \mathcal{U}_{\text{UB}} = \left\{ (u,\, \sigma) \in \mathbb{R}^4 \;\middle|\; \|u\|_2 \leq \sigma,\; 0 \leq \sigma \leq \frac{T_{\max}}{m_{\text{wet}}} \right\} \tag{6.50}$$

The set given by Equation (6.50), $\mathcal{U}_{\text{CQC}}$, is a compact quadratic cone. We compute a polytopic inner approximation of $\mathcal{U}_{\text{CQC}}$ by following the procedure described in §4.1.2.

With this approximation applied to the discretized precision landing guidance problem, we get the discrete-time polytopic precision landing guidance problem that fits the template of Problem P5, which is shown in Problem P12. Note that Equations (P11c) and (P11d) in Problem P11 are replaced by Equation (P12c) in Problem P12.

6.3.2 Closed-Loop Simulation Setup

In this subsection, we describe the closed-loop optimal control architecture in §6.3.2 and the problem and simulation parameters in §6.3.2, and discuss the sources of approximation and conservatism in §6.3.2.

Architecture

The closed-loop optimal control architecture for autonomous precision landing is shown in Figure 6.9. Specifically, the figure depicts the architecture adopted for the deterministic simulations in this section.

The framework involves both offline and online components. Offline, we first obtain a

Problem P12: Discrete-Time Precision Landing Problem (Polytopic)

$$
\begin{aligned}
\underset{\substack{N,\, c_1,\\ u_1,\ldots,u_{N-1},\\ \sigma_1,\ldots,\sigma_{N-1}}}{\text{minimize}} \quad & c_1 && && \text{(P12a)}\\
\text{subject to} \quad & (r_{k+1}, v_{k+1}, z_{k+1}, c_{k+1}) = A\cdot(r_k, v_k, z_k, c_k) + B\cdot(u_k, \sigma_k) + d, && k = 1,\ldots,N-1 && \text{(P12b)}\\
& (u_k, \sigma_k) \in \mathcal{U}_{\text{CQC}} && k = 1,\ldots,N-1 && \text{(P12c)}\\
& e_z^\top u_k \geq \frac{T_{\min}}{m_{\text{dry}}} && k = 1,\ldots,N-1 && \text{(P12d)}\\
& e_z^\top u_k \geq \sigma_k \cos\theta_{\max} && k = 1,\ldots,N-1 && \text{(P12e)}\\
& H_{\text{GS}}\, r_k \leq h_{\text{GS}} && k = 2,\ldots,N-1 && \text{(P12f)}\\
& \|r_k\|_\infty \leq r_{\max} && k = 2,\ldots,N-1 && \text{(P12g)}\\
& \|v_k\|_\infty \leq v_{\max} && k = 2,\ldots,N-1 && \text{(P12h)}\\
& \ln m_{\text{dry}} \leq z_k \leq \ln m_{\text{wet}} && k = 2,\ldots,N-1 && \text{(P12i)}\\
& 0 \leq c_k \leq \ln\frac{m_{\text{wet}}}{m_{\text{dry}}} && k = 2,\ldots,N-1 && \text{(P12j)}\\
& r_1 = r_{\text{i}},\ \ v_1 = v_{\text{i}},\ \ z_1 = \ln m_{\text{wet}},\ \ c_1 \leq \ln\frac{m_{\text{wet}}}{m_{\text{dry}}} && && \text{(P12k)}\\
& r_N = r_{\text{f}},\ v_N = v_{\text{f}},\ z_N \geq \ln m_{\text{dry}},\ c_N = 0 && && \text{(P12l)}
\end{aligned}
$$

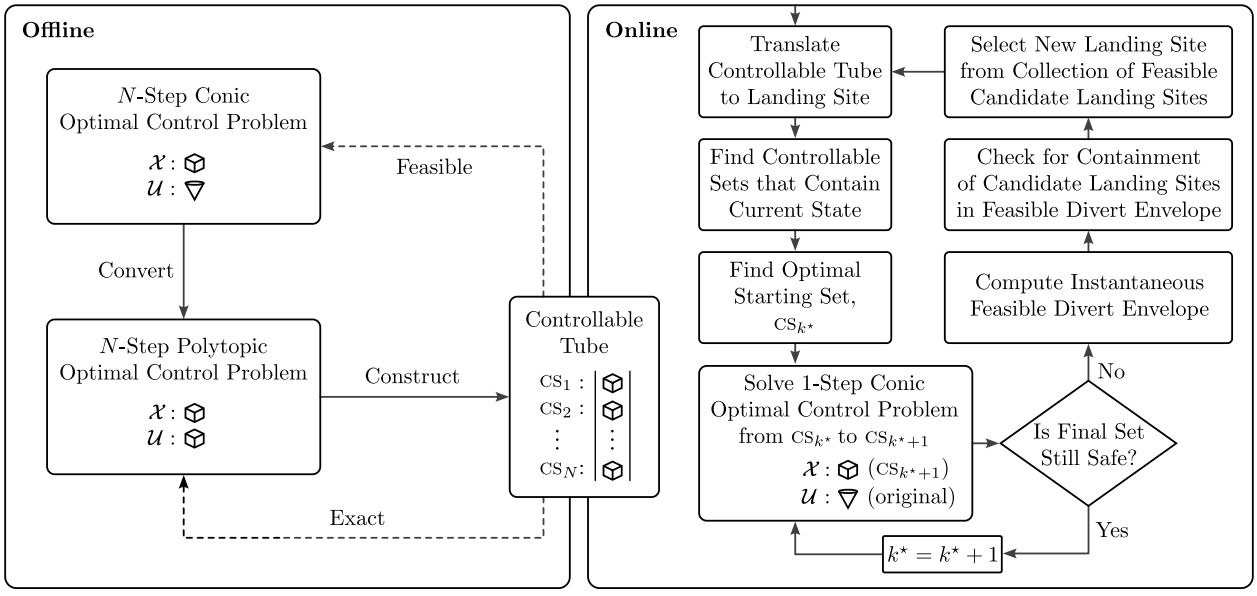

Figure 6.9: A closed-loop optimal control architecture for autonomous precision landing with integrated divert capabilities.

polytopic approximation (Problem P5) of the conic optimal control problem, Problem P4. Next, we generate the controllable tube based on the polytopic optimal control problem, using Algorithm 11 (or Algorithm 12 in the robust case), the recursion for which is guaranteed to terminate for the autonomous precision landing problem owing to the limited fuel. The controllable tube is then stored onboard the system. What follows is the description of a notional closed-loop control framework for autonomous precision landing with integrated divert capabilities, as shown in Figure 6.9.

Given the translation-invariance of the position subspace, rather than generating a controllable tube for each target landing site, we only construct one controllable tube corresponding to the origin, and either (i) translate the tube to the landing site in question via a Minkowski sum operation, or (ii) shift the reference frame such that the corresponding landing site becomes the origin. Next, we perform containment checks (see Table 4.2) to determine which of the controllable sets in the controllable tube contain the current state. Once these controllable sets have been determined, Algorithm 15 is executed to determine the optimal starting controllable set, i.e., the one that corresponds to the lowest cost-to-go value. Then, the forward rollout (Algorithm 16) is commenced from that controllable set, which involves solving a sequence of one-step optimal control problems (of the form of Problem P7).

The forward rollout is executed in the nominal mode of operation, i.e., as described in Algorithm 16, for as long as the target landing site is safe. In case it is deemed unsafe at any point in time, computation of the instantaneous feasible divert envelope, i.e., the instantaneous reachable set in the horizontal position subspace, is triggered. Given a collection of candidate landing sites (assumed to be priority-ordered), the next best landing site can be determined by performed containment checks against the feasible divert envelope: if the landing site is contained within the divert envelope, it is guaranteed to be reachable (by construction); if it is not contained in the divert envelope, it is not reachable. It could also be used to (i) determine a feasible site in a safe region (by taking the intersection of the divert envelope with a safety map, for instance), and (ii) achieve maximal diverts or "flyaway"s to

ensure safety of the delivered payload [9], i.e., a point on the boundary of the divert envelope can be chosen in any desired direction, and a closed-loop solution to it is guaranteed to exist by construction (based on Problem P12). Once a new landing site is chosen, the controllable tube is then translated to that site, and the process is repeated until touchdown.

Problem Parameters

The problem parameter values chosen for the optimal, resilient, and robust control simulations are listed in Table 6.2. In the table, Δt refers to the discretization sampling time, and n_{points} refers to the number of points to be spread on the 3D unit sphere for the polytopic approximation of the conic control constraint set; $r_{\text{f}}^{\text{backup}}$ is the backup landing site used in the resilient control simulations.

The closed-loop robust control (Monte Carlo) simulation is run for a fixed final time, i.e., a fixed horizon length, N, in order to stay consistent with the adopted probabilistic framework. The 3-sigma control uncertainty is denoted by $\delta_{3\sigma}^{u}$—note that $R_u = \frac{\delta_{3\sigma}^{u}}{3}$ and $\Sigma^u = R_u^2\, I_3 \in \mathbb{R}^3$. The time-varying 3-sigma position and velocity uncertainties are denoted by $\delta_{3\sigma}^{r}(k)$ and $\delta_{3\sigma}^{v}(k)$, respectively, where $k = 1, \ldots, N$. Similarly, note that $\Sigma_k^x = \text{blkdiag}\left\{ \left(\frac{\delta_{3\sigma}^{r}(k)}{3}\right)^2 I_3,\, \left(\frac{\delta_{3\sigma}^{v}(k)}{3}\right)^2 I_3 \right\} \in \mathbb{R}^6$. Finally, we note that $\delta_{3\sigma}^{r}(1)$ corresponds to a 30 m initial 3-sigma navigation uncertainty in position, and $\delta_{3\sigma}^{v}(1)$ corresponds to a 0.6 $\text{m}\,\text{s}^{-1}$ initial 3-sigma navigation uncertainty in velocity; the value for $\delta_{3\sigma}^{u}$ corresponds to roughly 0.5% the maximum control magnitude.

Sources of Approximation and Conservatism

Here, we catalog the sources of approximation and conservatism in the proposed closed-loop control framework. As shown in Equation (6.33), the original bounds on the control magnitude are (time-varying) nonlinear functions of the log-mass; instead, we only consider conservatively chosen time-invariant bounds, as shown in Equation (6.35). Further, we introduce conservatism in the control magnitude lower bound, as shown in Equation (6.37).

Parameter	Value
g	$1.625\ \mathrm{m\,s^{-2}}$
T_{full}	$10500\ \mathrm{kg\,m\,s^{-2}}$
$T_{\max}$	$0.8\,T_{\mathrm{full}}$
$T_{\min}$	$0.2\,T_{\mathrm{full}}$
m_{wet}	$1905\ \mathrm{kg}$
m_{dry}	$1505\ \mathrm{kg}$
$\theta_{\max}$	50°
$r_{\max}$	$4000\ \mathrm{m}$
$v_{\max}$	$100\ \mathrm{m\,s^{-1}}$
r_{f}	$(0,0,0)\ \mathrm{m}$
v_{f}	$(0,0,0)\ \mathrm{m\,s^{-1}}$
$\gamma_{\max}$	80°
H_{GS}	$\begin{bmatrix} \cos\gamma_{\max} & 0 & -\sin\gamma_{\max} \\ 0 & \cos\gamma_{\max} & -\sin\gamma_{\max} \\ -\cos\gamma_{\max} & 0 & -\sin\gamma_{\max} \\ 0 & -\cos\gamma_{\max} & -\sin\gamma_{\max} \end{bmatrix}$
h_{GS}	$(0,0,0,0)$

Parameter	Value
N	free
Δt	$3\ \mathrm{s}$
α	$0.00115\ \mathrm{s\,m^{-1}}$
r_{i}	$(875,0,635)\ \mathrm{m}$
$r_{\mathrm{f}}^{\mathrm{backup}}$	$(1700,0,0)\ \mathrm{m}$
v_{i}	$(40,0,-30)\ \mathrm{m\,s^{-1}}$
n_{points}	302
N	20
Δt	$15\ \mathrm{s}$
α	$0.0002875\ \mathrm{s\,m^{-1}}$
r_{i}	$(4000,4000,4000)\ \mathrm{m}$
v_{i}	$(-10,-10,-10)\ \mathrm{m\,s^{-1}}$
n_{points}	14
λ	0.95
$\delta^u_{3\sigma}$	$0.023\ \mathrm{m\,s^{-2}}$
$\delta^r_{3\sigma}(k)$	$1.5\,(N-k+1)\ \mathrm{m}$
$\delta^v_{3\sigma}(k)$	$0.03\,(N-k+1)\ \mathrm{m\,s^{-1}}$

Table 6.2: Problem parameters. Left: common; right top: closed-loop optimal and resilient control simulations; right bottom: closed-loop robust control (Monte Carlo) simulations, where $k = 1, \ldots, N$.

Next, we consider a polytopic inner approximation to the control constraint set for controllable tube generation via set recursion, which effectively shrinks the controllable space—this approximation can be made arbitrarily accurate by choosing more points to spread on the unit sphere (see §4.1.2).

In the robust case, specifically, there are three additional sources of conservatism: (i) the independence assumption on the effective disturbance for robustification, in which case the controllable tube is robust to the worst-case disturbance at each time-step without accounting for coupling of the state disturbances in time, (ii) invocation of the triangle and reverse triangle inequalities to tighten the control constraints in §6.3.4, and (iii) the Pontryagin difference algorithm [229, Algorithm 3] in the set recursion for robust controllable tube generation, which returns an inner approximation of the true Pontryagin difference that worsens as the latent dimension of the constrained zonotope(s) increases, i.e., as the number of columns, n_g, of G in Equation (4.6) increases. Choosing more points on the unit sphere to obtain the polytopic approximation of the conic control set, for example, will lead to constrained zonotopes that are larger in latent dimension, thus making the Pontryagin difference more conservative. Note, however, that the conservatism introduced by (i), (ii), and (iii) is *in the right direction*, i.e., they lead to the robust controllable tube being *more* robust than necessary, thus preserving all the claimed probability guarantees.

6.3.3 Optimal Control: The Optimal Guidance Algorithm

We solve the deterministic, free-final-time (minimum-fuel) autonomous precision landing guidance problem using Algorithm 11 for controllable tube generation and Algorithm 16 for the forward rollout. Within the forward rollout, we solve Problem P6 as is and do not invoke Remark 1 here to demonstrate global optimality. For comparison, we also solve Problem P12 directly in an open-loop setting, and show the control magnitude profiles in Figure 6.10.

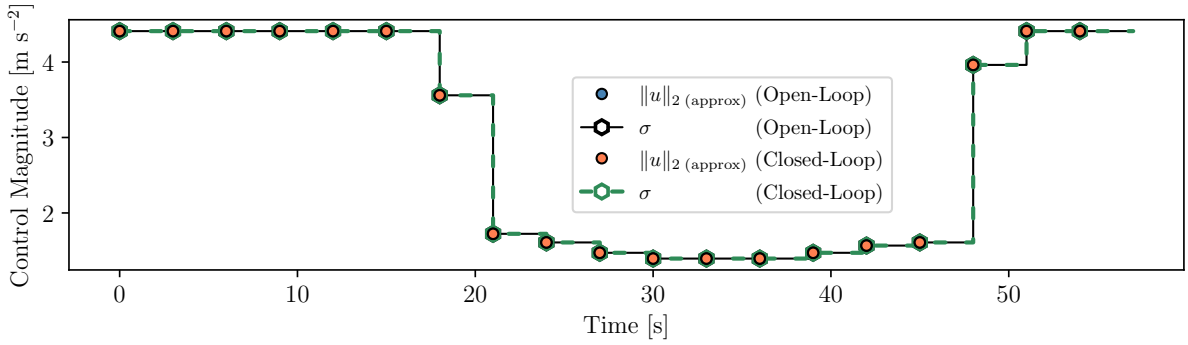

Figure 6.10: Algorithm 16—with the one-step optimal control problem in accordance with Problem P6—recovers a globally optimal solution to the polytopic open-loop *free-final-time* (minimum-fuel) problem (Problem P12), which is solved in concert with a golden section search.

6.3.4 Robust Control: Robustness to Navigation and Actuation Uncertainties

In this subsection, we describe robust control in the context of the autonomous precision landing problem. Specifically, we model navigation and actuation uncertainty in §6.3.4, describe practical aspects of robust controllable tube generation in §6.3.4, and describe robustification of the control and state constraints in §6.3.4 and §6.3.4, respectively. Finally, we present Monte Carlo simulation results in §6.3.4.

Navigation and Actuation Uncertainty Models

We assume that the estimated position and velocity are subject to additive navigation (state) uncertainty and that the control input is subject to additive actuation (control) uncertainty, both of which are modeled as Gaussian disturbances, in accordance with §4.2.2. Note that we treat the augmented control slack variable, σ, such that it is not directly perturbed; instead, we robustify the original control constraints first and then convexify them (i.e., introduce σ). We also assume that the log-mass state and the cost-to-go state can be estimated perfectly (since they are functions of the *applied* control input and not the commanded control input).

The bounded disturbance sets for the uncertainties, as required by Algorithm 12, are constructed in accordance with §4.2.3.

Robust Controllable Tube Generation

We schedule the state covariances such that the corresponding disturbance ellipses decrease in size over time (the standard deviations decrease linearly)—this accounts for the improvement in the accuracy of the navigation estimates as the system approaches the landing site (as more information is acquired by the sensors). In the robust controllable tube set recursion algorithm (Algorithm 12), the terminal set constrained zonotope needs to be full-dimensional and have a MinRow representation for the Pontryagin difference step to be computationally tractable [229, Definition 3 and Proposition 1]. To generate this full-dimensional terminal set, we first perform a deterministic set recursion (Algorithm 11) for one Δt-second time interval, and then proceed with robust set recursion. This has the added benefit of guaranteeing the existence of a solution to the original terminal set for any true state lying in the thus-generated full-dimensional terminal set, and can thus account for reserve fuel, for example, in an exact manner, without requiring conservative estimates. In practice, we observe that the conditions required for Pontryagin differencing to be tractable [229, Definition 3 and Proposition 1] do not hold when the set is generated via a one-step recursion. To mitigate this, we perform a two-step recursion, but with a sampling time of $\frac{\Delta t}{2}$ instead (such that the effective interval is still Δt).

Robustification of Control Constraints

Here, we present one approach to robustifying the control constraint set with respect to the worst-case control disturbance.

Control Magnitude Bounds To robustify the control magnitude bound constraint, we first consider the original constraint given by Equation (6.35), and tighten the bounds with respect to the worst-case disturbance from the control disturbance set. Once the tightened bounds are obtained, the lower bound constraint is replaced with the conservative convex counterpart of Equation (6.37).

The control magnitude constraint, Equation (6.35), with actuation uncertainty, is given by:

$$\frac{T_{\min}}{m_{\text{dry}}} \leq \|u_k + w_k^u\|_2 \leq \frac{T_{\max}}{m_{\text{wet}}}, \quad k = 1, \ldots, N-1 \tag{6.51}$$

To be able to account for the slack variable that will be introduced for lossless convexification, we seek a representation of Equation (6.51) that is solely in terms of $\|u_k\|_2$. As such, we approximate the thrust magnitude constraint given by Equation (6.51) as follows.

Invoking the triangle and reverse triangle inequalities, we get:

$$|\|u_k\|_2 - \|w_k^u\|_2| \leq \|u_k + w_k^u\|_2 \leq \|u_k\|_2 + \|w_k^u\|_2 \tag{6.52}$$

We make the assumption that $\|w_k^u\|_2 \leq \|u_k\|_2$, i.e., the magnitude of the bounded disturbance will be less than or equal to the control input magnitude, to ensure that any constraint tightening we perform will not make the control magnitude constraint set infeasible. With this assumption, Equation (6.52) can be written as:

$$\|u_k\|_2 - \|w_k^u\|_2 \leq \|u_k + w_k^u\|_2 \leq \|u_k\|_2 + \|w_k^u\|_2 \tag{6.53}$$

Now, our goal is to represent Equation (6.53) such that it is solely in terms of u_k. For this, we consider the Equation (6.53) for all possible disturbances in the disturbance set:

$$\|u_k\|_2 - \|w_k^u\|_2 \leq \|u_k + w_k^u\|_2 \leq \|u_k\|_2 + \|w_k^u\|_2 \quad \forall\, w_k^u \in \mathcal{W}_u \tag{6.54a}$$

$$\iff \inf_{w_k^u \in \mathcal{W}_u} (\|u_k\|_2 - \|w_k^u\|_2) \leq \|u_k + w_k^u\|_2 \leq \sup_{w_k^u \in \mathcal{W}_u} (\|u_k\|_2 + \|w_k^u\|_2) \tag{6.54b}$$

$$\iff \|u_k\|_2 - \sup_{w_k^u \in \mathcal{W}_u} \|w_k^u\|_2 \leq \|u_k + w_k^u\|_2 \leq \|u_k\|_2 + \sup_{w_k^u \in \mathcal{W}_u} \|w_k^u\|_2 \tag{6.54c}$$

Therefore,

$$\|u_k\|_2 - \psi_u \leq \|u_k + w_k^u\|_2 \leq \|u_k\|_2 + \psi_u \tag{6.55}$$

where,

$$\psi_u := \sup_{w_k^u \in \mathcal{W}_u} \|w_k^u\|_2 \tag{6.56}$$

Now, the control bound constraint can be rewritten (approximated) in terms of u_k as follows:

$$\frac{T_{\min}}{m_{\text{dry}}} \leq \|u_k\|_2 - \psi_u \leq \|u_k + w_k^u\|_2 \leq \|u_k\|_2 + \psi_u \leq \frac{T_{\max}}{m_{\text{wet}}} \tag{6.57a}$$

$$\implies \frac{T_{\min}}{m_{\text{dry}}} + \psi_u \leq \|u_k\|_2 \leq \frac{T_{\max}}{m_{\text{wet}}} - \psi_u \tag{6.57b}$$

Equation (6.57b) represents a conservative tightening of the thrust magnitude bound constraint set, in that it is guaranteed to be smaller than or equal to the tightened constraint set that would have resulted had the approximation not been introduced.

Further, we derive a closed-form expression for ψ_u by making the assumption that the control disturbance is isotropic, in that the projection of the effective disturbance ellipsoid (given by Equation (4.25b)) onto the control dimensions is a 3-dimensional ball, which in turn implies that Σ_u in Equation (4.10) is of the form $R_u^2\, I_3$, where $R_u > 0$ is the radius of the 3D ball. Therefore:

$$\psi_u = R_u \tag{6.58}$$

$$\implies \frac{T_{\min}}{m_{\text{dry}}} + R_u \leq \|u_k\|_2 \leq \frac{T_{\max}}{m_{\text{wet}}} - R_u \tag{6.59}$$

Note that the control disturbance can be ellipsoidal as well if desired, but we make the isotropic assumption for simplicity. Finally, we (i) retain the upper bound constraint in

Equation (6.59), but replace the 2-norm term with the slack variable, and (ii) replace the lower bound constraint with the conservative convex constraint (see Equation (6.37)), to yield the following tightened constraints:

$$\sigma_k \leq \frac{T_{\max}}{m_{\text{wet}}} - R_u \tag{6.60a}$$

$$e_z^\top u_k \geq \frac{T_{\min}}{m_{\text{dry}}} + R_u \tag{6.60b}$$

Thrust Pointing As with the control magnitude bound constraints, the thrust pointing constraint with actuation uncertainty, for $k = 1, \ldots, N-1$, is robustified and reformulated as follows:

$$e_z^\top (u_k + w_k^u) \geq \|u_k + w_k^u\|_2 \cos\theta_{\max}, \quad \forall\, w_k^u \in \mathcal{W}_u \tag{6.61a}$$

$$\iff \inf_{w_k^u \in \mathcal{W}_u} e_z^\top (u_k + w_k^u) \geq \sup_{w_k^u \in \mathcal{W}_u} \|u_k + w_k^u\|_2 \cos\theta_{\max} \tag{6.61b}$$

$$\impliedby \inf_{w_k^u \in \mathcal{W}_u} (e_z^\top u_k + e_z^\top w_k^u) \geq \sup_{w_k^u \in \mathcal{W}_u} (\|u_k\|_2 + \|w_k^u\|_2) \cos\theta_{\max} \tag{6.61c}$$

$$\iff e_z^\top u_k + \inf_{w_k^u \in \mathcal{W}_u} e_z^\top w_k^u \geq (\|u_k\|_2 + \sup_{w_k^u \in \mathcal{W}_u} \|w_k^u\|_2) \cos\theta_{\max} \tag{6.61d}$$

$$\iff e_z^\top u_k - \psi_u \geq (\|u_k\|_2 + \psi_u) \cos\theta_{\max} \tag{6.61e}$$

$$\iff e_z^\top u_k - R_u \geq (\|u_k\|_2 + R_u) \cos\theta_{\max} \tag{6.61f}$$

$$\iff e_z^\top u_k - R_u \geq (\sigma_k + R_u) \cos\theta_{\max} \tag{6.61g}$$

Robustification of State Constraints

The position and velocity states are assumed to be subject to time-varying uncertainty. While the log-mass and cost-to-go states are assumed to be estimated perfectly, and while we do not explicitly subject the augmented control slack variable, σ, to uncertainty, the influence of actuation uncertainty on them calls for additional consideration.

Specifically, since there are no explicit sources of disturbance affecting the log-mass and cost-to-go states, the Pontryagin difference step in the robust set recursion algorithm (Algo-

rithm 12) is agnostic to the influence of control disturbances on them. As such, we need to "manually" ensure that the bounds on these states are appropriately tightened.

The log-mass dynamics with actuation uncertainty are given as follows, for $k = 1, \ldots, N-1$:

$$z_{k+1} = z_k - \alpha \left\| u_k + w_k^u \right\|_2 \tag{6.62}$$

We consider the worst-case mass depletion and for this, we consider the most adversarial disturbance. From the RHS of Equation (6.55), since $\alpha > 0$, we get:

$$z_{k+1} = z_k - \alpha \left(\|u_k\|_2 + \psi_u \right) = z_k - \alpha \left(\|u_k\|_2 + R_u \right) \tag{6.63}$$

Similarly, the worst-case cost-to-go "depletion", for $k = 1, \ldots, N-1$, would be:

$$c_{k+1} = c_k - \alpha \left(\|u_k\|_2 + R_u \right) \tag{6.64}$$

Rather than explicitly tightening the bounds for these states—which would result in time-varying state constraint sets—we directly consider the worst-case depletion dynamics given by Equations (6.63) and (6.64) in the robust set recursion (with the original bounds)—this is an equivalent approach that implicitly achieves the effect of tightening the bounds, since both z and c are strictly monotonically decreasing quantities.

The Pontryagin difference step in Algorithm 12 accounts for robustification of the constraints on the remaining states that are influenced by control and/or state disturbances.

Monte Carlo Simulation

To demonstrate the closed-loop robust control framework and highlight the probabilistic guarantees, we perform a Monte Carlo simulation with the problem parameters given in Table 6.2. For the same problem parameters (and boundary conditions), we run 100 cases with different random disturbance sequences sampled from Gaussian distributions, in accordance

with §4.2. We set the probability that the terminal state estimate, $\tilde{y}_N$, lies in the robustified terminal set, $\tilde{\mathcal{X}}_\mathrm{f} \ominus \mathcal{W}_N$—which corresponds to the probability that the true terminal state, y_N, lies in the true terminal set, $\tilde{\mathcal{X}}_\mathrm{f}$—to be $\lambda = 95\%$. In other words, the true terminal state is guaranteed to lie in the true terminal set with an 95% probability. We plot the terminal values of the true position in Figure 6.11 and the true velocity in Figure 6.12; all 100 trials are successful, in that all the true terminal state values lie inside the true terminal set in all 100 trials. Note that the success-rate is 100% despite the requested probability only being 95%, owing to the conservatism in the robustification (in the right direction).

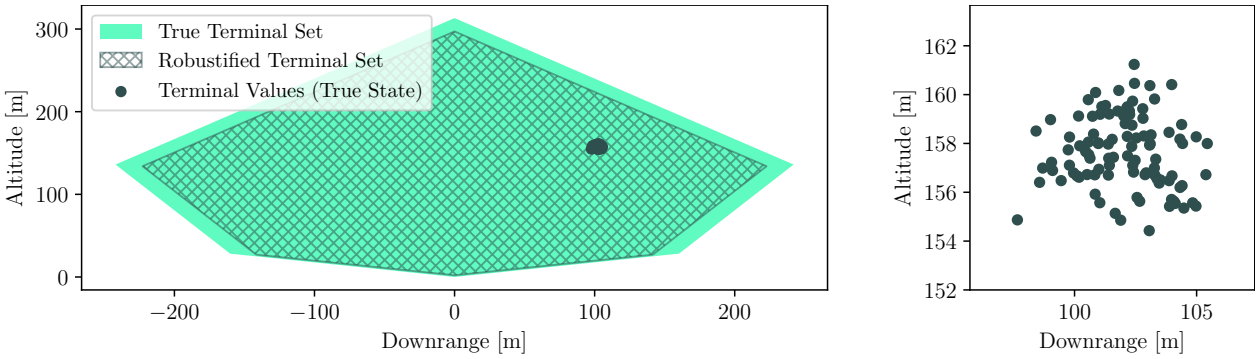

Figure 6.11: Terminal position values from the Monte Carlo simulation, projected onto the x-z plane.

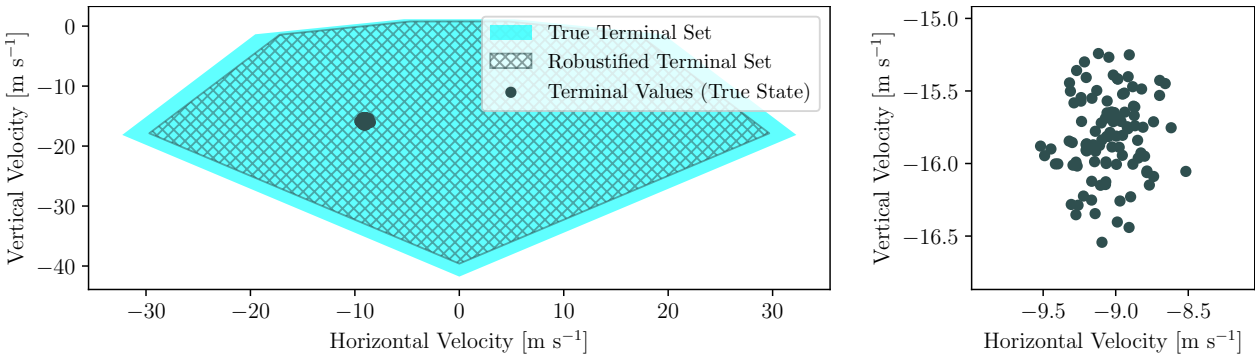

Figure 6.12: Terminal velocity values from the Monte Carlo simulation, projected onto the x-z plane.

6.3.5 Resilient Control: Instantaneous Divert Envelope and Decision-Deferral

We demonstrate two modes of closed-loop resilient control for autonomous precision landing, the first based on the computation of the instantaneous feasible divert envelope, i.e., the instantaneous reachable set in the horizontal position subspace (see §4.3.1), and the second based on the set-based deferred-decision trajectory optimization framework (see §4.3.2).

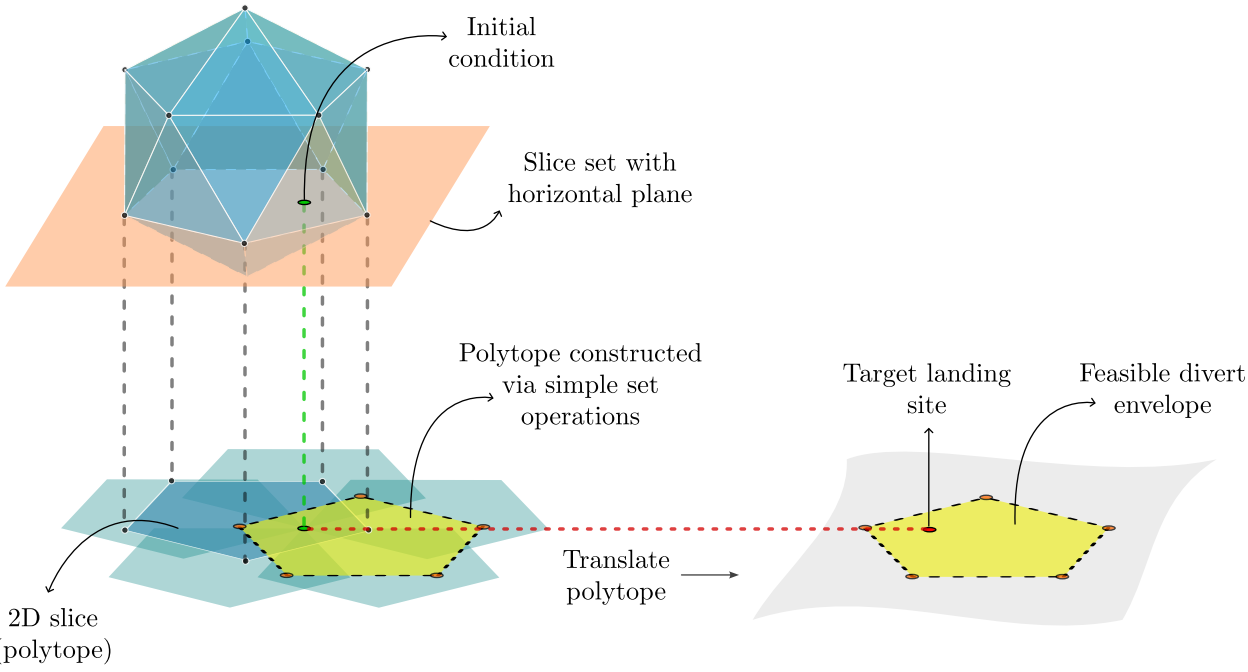

Figure 6.13: Construction of the instantaneous feasible divert envelope for Problem P12 using Algorithm 13. The 3D set depicted is the projection of the 8D controllable set onto the position coordinates.

The first mode of resilient control relies on computation of the instantaneous feasible divert envelope (based on Algorithm 13), and is depicted in Figure 6.13. We note that divert envelope thus generated is *exact* with respect to Problem P12. Here, we consider two landing sites, a nominal site and a divert (backup) landing site—the numerical values for parameters chosen for these simulations are provided in Table 6.2. In Figure 6.14, we show the (independent) optimal free-final-time trajectories corresponding to the two landing

sites, the corresponding control magnitude profiles (in which lossless convexification holds), the instantaneous reachable set (i.e., the instantaneous divert envelope) at the 1st time-step (corresponding to time $t = 0$ s), and the instantaneous reachable set at the 8th time-step (corresponding to time $t = 21$ s). Note that at $t = 0$ s, both the target landing sites are reachable, but at $t = 21$ s, neither of the reachable sets contains the other, i.e., by committing to a landing site too early (at $t = 0$ s) and performing the forward rollout (Algorithm 16) accordingly, the other landing site is no longer reachable at $t = 21$ s.

The second mode of resilient control involves the set-based counterpart of the *deferred-decision trajectory optimization* (DDTO) framework [68]. Here, we consider the same scenario as the previous case, but instead, use the forward rollout algorithm with decision-deferral (Algorithm 14). In this mode, the system is required to remain in the intersection of the controllable tube (corresponding to the nominal site) and the translated controllable tube (corresponding to the divert site) for as long as possible. The two trajectories, the corresponding control magnitude profiles (in which lossless convexification holds), and the instantaneous reachable sets at $t = 0$ s and $t = 21$ s, are shown in Figure 6.15. Note that both the target landing sites are reachable at (and until) $t = 21$ s in this case, unlike in the previous divert case, thus demonstrating the benefit of deferring decision. In the event one of the target landing sites becomes unsafe during the first 21 seconds, with decision-deferral, we preserve the ability to divert to the other site, whereas this would not be possible in the feasible divert case.

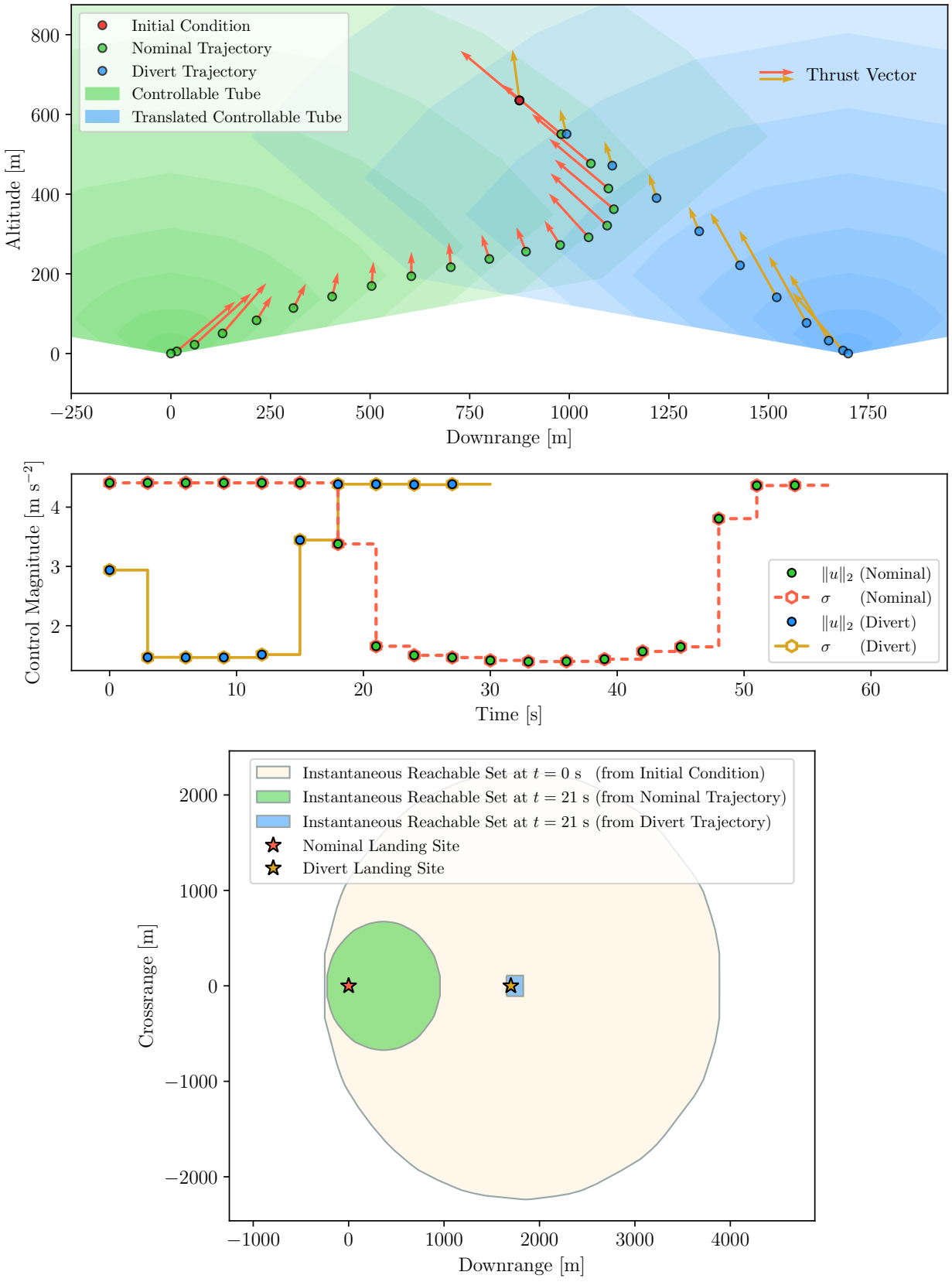

Figure 6.14: **Feasible divert mode: Free-final-time trajectory optimization using the controllable tube and its translation, with Algorithm 16: at $t = 21$ s, the two divert envelopes are mutually exclusive, and thus the system en route to one landing site is unable to reach the other.**

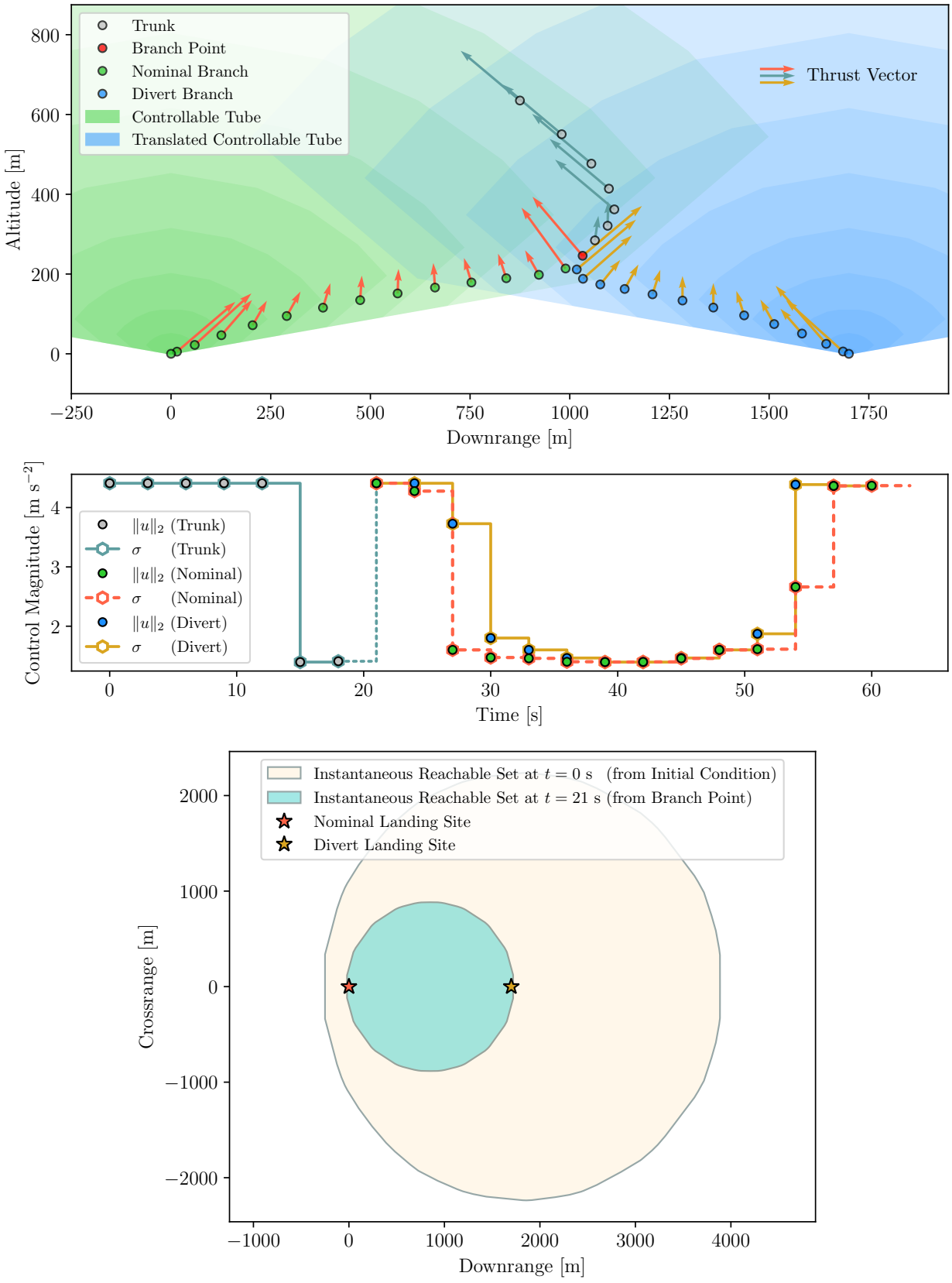

Figure 6.15: **Maximal decision-deferral mode: Free-final-time *deferred-decision* trajectory optimization (DDTO) using the controllable tube and its translation, with Algorithm 14: at $t = 21$ s, the divert envelope(s) contain both the landing sites as a result of deferring decision for as long as possible, and hence, both the sites are reachable until then, in contrast to the feasible divert mode (see Figure 6.14).**

Chapter 7

SCvxGEN: SUCCESSIVE CONVEXIFICATION WITH AUTOMATIC CODE GENERATION

SCvxGEN is a software-based framework for general-purpose and real-time trajectory optimization. By general-purpose, we mean that SCvxGEN can natively handle a general class of nonconvex trajectory optimization problems, including problems with multiple phases and logical constraints, for which we provide equivalent smooth formulations via a modeling library. In general, SCvxGEN only requires that the nonconvex functions in question are once continuously differentiable ($\mathcal{C}^1$). This is in contrast to many existing trajectory optimization methods, which rely on sequential quadratic programming (SQP), and, consequently, require the nonconvexities to be twice continuously differentiable ($\mathcal{C}^2$). By real-time, we mean that SCvxGEN can solve challenging real-world problems fast enough to be amenable to online implementation for embedded applications. This is achieved through the automatic generation of high-performance custom code for the problem at hand.

7.1 Formulation

In this section, we set up the general template optimal control problem that can be solved using SCvxGEN. We first discuss the integral constraint reformulation (§7.1.1) that helps us achieve continuous-time constraint satisfaction via a state augmentation, and then discuss time-dilation (§7.1.2), which helps make free-final-time problems tractable, via a control input augmentation.

We consider dynamical systems of the form:

$$\dot{x}(t) = f(x(t), u(t)), \quad \text{a.e. } t \in [0, t_\text{f}] \tag{7.1}$$

where $t \in [0, t_\mathrm{f}]$ is the wall-clock time, $t_\mathrm{f} > 0$ is the final time, $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $u(t) \in \mathbb{R}^{n_u}$ is the control input vector, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is a once continuously differentiable ($\mathcal{C}^1$) vector-valued function, $\dot{\square}$ is the derivative with respect to t, and a.e. $t \in [0, t_\mathrm{f}]$ stands for "holds almost everywhere on $[0, t_\mathrm{f}]$", which simply means "holds for all $t \in [0, t_\mathrm{f}]$, except for a finite set of points, i.e., a set of measure zero" [196]. We set the initial time to 0 to simplify notation, but it can be any nonzero value $< t_\mathrm{f}$. Further, f can be time-varying, i.e., a function of t, as well, but we omit the explicit temporal argument without loss of generality, since t can be be treated as a state (with $\dot{t} = 1$) and augmented to the original state vector.

7.1.1 Continuous-Time Constraint Reformulation

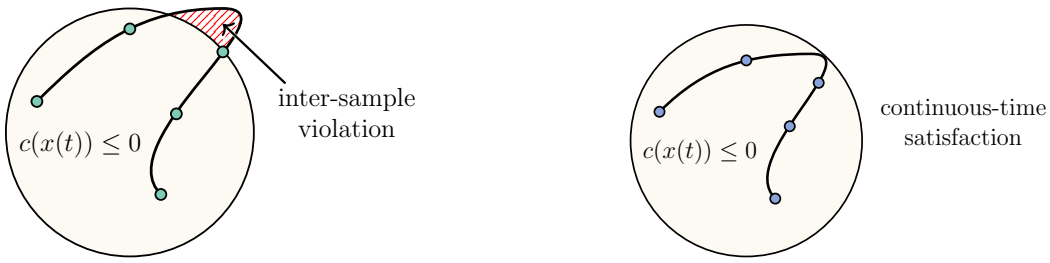

Figure 7.1: Direct methods for trajectory optimization impose path constraints (such as $c(x(t)) \leq 0$) at finitely-many discrete temporal nodes (shown on the left, with green nodes), invariably causing inter-sample violation (region shaded in red) in the state trajectory x obtained by simulating the dynamical system in an open-loop by applying the optimized control input signal. The CT-SCvx framework mitigates this phenomenon (shown on the right, blue nodes) [66].

Direct optimal control methods are able to ensure continuous-time satisfaction of nonlinear dynamics at solution convergence using direct shooting-type discretization schemes [35, 168, 134]. So the key idea in ensuring continuous-time satisfaction of path constraints as well is a reformulation that embeds the constraints in the dynamics, i.e., the integral constraint reformulation.

Consider the following inequality and equality path constraints, respectively:

$$q(x(t), u(t)) \leq 0_{n_q} \tag{7.2}$$

$$r(x(t), u(t)) = 0_{n_r} \tag{7.3}$$

where $q : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_q}$ and $r : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_r}$ are once continuously differentiable ($\mathcal{C}^1$) vector-valued functions, and the inequalities and equalities are element-wise. Now, we consider the following $\mathcal{C}^1$ scalar-valued continuous exterior penalty functions:

$$p_q(z) \begin{cases} = 0 & \text{if } z \leq 0 \\ > 0 & \text{otherwise} \end{cases} \tag{7.4}$$

$$p_r(z) \begin{cases} = 0 & \text{if } z \leq 0 \\ > 0 & \text{otherwise} \end{cases} \tag{7.5}$$

and the scalar-valued penalty function:

$$f_y(x(t), u(t)) = \sum_{i=1}^{n_q} p_q(w_{q_i}\, q_i(x(t), u(t))) + \sum_{j=1}^{n_r} p_r(w_{r_i}\, r_j(x(t), u(t))) \tag{7.6}$$

where $w_{q_i} > 0$, $i = 1, \ldots, n_q$, and $w_{r_j} > 0$, $j = 1, \ldots, n_r$, are scalar weights, and q_i and r_j are scalar-valued elements of q and r, respectively. In practice, w_{q_i} and w_{r_i} are chosen so as to *precondition* (see §7.2.2) the corresponding q_i and r_i functions, respectively. Consider the following rectified linear unit (ReLU) function:

$$\text{ReLU}(z) := \max\{0, z\} \tag{7.7}$$

One choice for the exterior penalty function is [66]:

$$p_q(z) := \mathrm{ReLU}(z)^2 \tag{7.8}$$

$$p_r(z) := z^2 \tag{7.9}$$

Another choice (with better numerical conditioning) is [218]:

$$p_q(z) := \sqrt{\mathrm{ReLU}(z)^2 + a^2} - a \tag{7.10}$$

$$p_r(z) := \sqrt{z^2 + a^2} - a \tag{7.11}$$

where $a > 0$ is *any* (typically small) positive number. Note that both of the aforementioned choices are smooth and also exact, in that they do not introduce any approximations.

Further, note that $x(t)$ and $u(t)$ satisfy path constraints given by Equations (7.2) and (7.3), at all times (a.e.) $t \in [0, t_\mathrm{f}]$, *if and only if* the following integral condition holds [66, Lemma 2]:

$$\int_0^{t_\mathrm{f}} f_y(x(t), u(t))\, dt = 0 \tag{7.12}$$

Next, we define an auxiliary state, $y(t)$. Now, Equation (7.12) can be represented via the following boundary value problem:

$$\dot{y}(t) = f_y(x(t), u(t)) \tag{7.13}$$

$$y(0) = y(t_\mathrm{f}) \tag{7.14}$$

which is satisfied if and only if Equation (7.12) holds [66, Corollary 3]. We augment the original dynamical system given by Equation (7.1) with Equation (7.13), as follows:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} f(x(t), u(t)) \\ f_y(x(t), u(t)) \end{bmatrix} \in \mathbb{R}^{n_x+1}, \quad \text{a.e. } t \in [0, t_\mathrm{f}] \tag{7.15}$$

7.1.2 Generalized Time-Dilation

Free-final-time optimal control problems are typically intractable with dynamics in the standard form (Equation (7.1)), since t_f is not an explicit variable. Time-dilation is a transformation that exposes t_f as an explicit variable and converts the free-final-time problem to an equivalent fixed-final-time problem in normalized time [203, 208, Section III.A.1]. The dilation factor can either be treated as a single scalar parameter variable [203, 208, 100] or multiple discrete scalar parameter variables [103, 49, 42, 109]; it can also be generalized as a continuous-time control input variable [51, 66].

Consider dynamics in a sub-interval of $[0, t_\mathrm{f}]$, $[t_k, t_{k+1}]$, where $0 = t_1 \leq t_k < t_{k+1} \leq t_N = t_\mathrm{f}$, $k = 1, \ldots, N-1$, N being the number of discrete temporal nodes chosen. Consider the following invertible mapping $\tau : [t_k, t_{k+1}] \to [0, 1]$:

$$\tau(t) := \frac{t}{t_\mathrm{f}}, \quad t \in [t_k, t_{k+1}] \tag{7.16}$$

which normalizes (dilates) each wall-clock time-interval, $[t_k, t_{k+1}]$, to a fixed interval, $[0, 1]$. Let the derivative with respect to τ be denoted by $\mathring{\square}$. Taking the derivative of the left hand side (LHS) of Equation (7.15) with respect to τ and invoking the chain rule, we get a.e. $t \in [t_k, t_{k+1}]$, i.e., a.e. $\tau \in [0, 1]$:

$$\begin{bmatrix} \mathring{x}(t) \\ \mathring{y}(t) \end{bmatrix} = \frac{d}{d\tau} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \frac{dt}{d\tau} \frac{d}{dt} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \underbrace{\frac{dt}{d\tau}}_{:= s(\tau)} \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} \tag{7.17}$$

Note that t can be expressed as a function of τ, i.e., $t(\tau)$, $t : [0, 1] \to [t_k, t_{k+1}]$. Henceforth, we treat τ as the independent variable, and write $\square(t(\tau))$ as $\square(\tau)$. Next, we augment the control input vector with $s(\tau)$, and obtain the following equivalent transformed system:

$$\begin{bmatrix} \mathring{x}(\tau) \\ \mathring{y}(\tau) \end{bmatrix} = s(\tau) \begin{bmatrix} f(x(\tau), u(\tau)) \\ f_y(x(\tau), u(\tau)) \end{bmatrix} =: F\left(\begin{bmatrix} x(\tau) \\ y(\tau) \end{bmatrix}, \begin{bmatrix} u(\tau) \\ s(\tau) \end{bmatrix} \right), \quad \text{a.e. } \tau \in [0, 1] \tag{7.18}$$

7.1.3 Control Parameterization and Discretization

In order to obtain continuous-time solutions to optimal control problems, we first adopt a control parameterization, i.e., a parameterization for the augmented control input that completely describes the continuous-time control input signal with a finite number of discrete parameters. Specifically, we consider three control parameterization techniques: zero-order hold (ZOH) parameterization, in which the control input is assumed to be piecewise-constant (and discontinuous), first-order hold (FOH) parameterization, in which the control input is assumed to be piecewise-affine (and continuous), and impulse-input (IMP) parameterization, in which the control input is assumed to be a discrete train of impulses. In what follows, we present a unified description of control parameterization and discretization.

Consider the following control parameterization in wall-clock time-interval $[t_k, t_{k+1}]$, $k = 1, \ldots, N-1$, where $\tau \in [0, 1]$:

$$u(\tau) = \sigma^-(\tau)\, u_k + \sigma^+(\tau)\, u_{k+1} \tag{7.19}$$

where $u(\tau)$, and $\sigma^-(\tau)$ and $\sigma^+(\tau)$ take the following parameterization-dependent values:

	ZOH	FOH	IMP
$\sigma^-(\tau)$	1	$1-\tau$	$\delta(\tau)$
$\sigma^+(\tau)$	0	τ	0

Table 7.1: Control input interpolation parameters.

where $\delta(\cdot)$ is the Dirac-delta function. The dilation factor control input, $s(\tau)$, is only allowed to assume either ZOH or FOH. The node values of the augmented state and augmented control, (x_k, y_k) and (u_k, s_k), respectively, $k = 1, \ldots, N$, are treated as the decision

variables, yielding the following discrete-time nonlinear system:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \int_0^1 F\left(\begin{bmatrix} x(\tau) \\ y(\tau) \end{bmatrix}, \begin{bmatrix} u(\tau) \\ s(\tau) \end{bmatrix} \right) d\tau \tag{7.20a}$$

$$=: F_k\left(\begin{bmatrix} x_k \\ y_k \end{bmatrix}, \begin{bmatrix} u_k \\ s_k \end{bmatrix}, \begin{bmatrix} u_{k+1} \\ s_{k+1} \end{bmatrix} \right) \tag{7.20b}$$

7.1.4 Optimal Control Problem

After performing the integral constraint reformulation (§7.1.1), generalized time-dilation (§7.1.2), and control parameterization and discretization (§7.1.3), we are now ready to present the general discrete-time optimal control problem template:

Problem 7.21: General Optimal Control Problem

$\underset{\substack{u_1,\ldots,u_N,\\ s_1,\ldots,s_N}}{\text{minimize}}$	*Cost Function*:	$\sum_{k=1}^{N} L(x_k, u_k)$ or $e_c^\top x_N$ or $\sum_{k=1}^{N-1} L_s(s_k, s_{k+1})$		(7.21a)
subject to	*Dynamics (Augmented)*:	$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = F_k\left(\begin{bmatrix} x_k \\ y_k \end{bmatrix}, \begin{bmatrix} u_k \\ s_k \end{bmatrix}, \begin{bmatrix} u_{k+1} \\ s_{k+1} \end{bmatrix} \right)$,	$k = 1, \ldots, N-1$	(7.21b)
	Inequality Constraints:	$g(x_k, u_k) \leq 0$,	$k = 1, \ldots, N$	(7.21c)
	Equality Constraints:	$h(x_k, u_k) = 0$,	$k = 1, \ldots, N$	(7.21d)
	Convex Conic Constraints:	$(x_k, u_k) \in \mathcal{Z}_k$,	$k = 1, \ldots, N$	(7.21e)
	Control Bounds:	$u_{\min} \leq u_k \leq u_{\max}$,	$k = 1, \ldots, N$	(7.21f)
	Dilation Bounds:	$s_{\min} \leq s_k \leq s_{\max}$,	$k = 1, \ldots, N$	(7.21g)
	Control Rate Bounds (FOH):	$\lvert u_{k+1} - u_k \rvert \leq \dot{u}_{\max} \min\{s_k, s_{k+1}\}$,	$k = 1, \ldots, N-1$	(7.21h)
	Auxiliary Constraints:	$y_{k+1} - y_k \leq \epsilon_{\text{LICQ}}$,	$k = 1, \ldots, N-1$	(7.21i)
	Initial Condition:	$E_{\text{i}}\, x_1 = \tilde{x}_{\text{i}}$		(7.21j)
	Final Condition:	$E_{\text{f}}\, x_N = \tilde{x}_{\text{f}}$		(7.21k)

where, in the case of ZOH, $(u_N, s_N) = (u_{N-1}, s_{N-1})$, and in the case of IMP, $(u_N, s_N) = (0_{n_u}, s_{N-1})$, with ZOH assumed only on the dilation factor. We have already described the

augmented dynamics in §7.1.3 (Equation (7.20b)), which includes constraints—for which continuous-time satisfaction is desired—in integral form. Next, we describe the remaining components of the template optimal control problem.

Remark 8. *Problem* (7.21) *is convex when the cost function* (7.21a) *is convex and the dynamics* (7.21b)*, inequality constraints* (7.21c)*, and equality constraints* (7.21d)*, are affine. Note that for the dynamics to be affine: (i) the problem must be fixed-final-time, in which case* s_k*,* $k = 1, \ldots, N$*, are constants, and (ii) the problem must not have continuous-time constraints (i.e., must not have* y_k*,* $k = 1, \ldots, N$*, and the associated auxiliary dynamics).*

Cost Function The cost function in (7.21a) can either define a running cost, a terminal cost, or a minimum-time cost. The running cost is $\sum_{k=1}^{N} L(x_k, u_k)$, where $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is either convex or a $\mathcal{C}^1$ nonconvex function. The terminal cost is of the form $e_c^\top x_N$, where $e_c \in \mathbb{R}^{n_x}$ is a basis vector with only one element, which is ± 1. The minimum-time cost function is $\sum_{k=1}^{N-1} L_s(s_k, s_{k+1})$, where $L_s(s_k, s_{k+1}) := s_k$ if ZOH is used and $L_s(s_k, s_{k+1}) := \frac{s_k + s_{k+1}}{2}$ if FOH is used.

Inequality and Equality Constraints The inequality and equality node constraints, i.e., the inequality and equality constraints only imposed at the nodes, are given by Equations (7.21c) and (7.21d), respectively. These constraints can either be affine or $\mathcal{C}^1$ nonconvex constraints, which are treated accordingly.

Convex Conic Constraints The convex conic constraints on the state and control are given by Equation (7.21e), where each $\mathcal{Z}_k$, $k = 1, \ldots, N$, is a Cartesian product of convex conic sets.

Control, Dilation, and Control Rate Bounds We explicitly bound the control input variables and the dilation factors element-wise, as shown in Equations (7.21f) and (7.21g),

respectively. The dilation bounds, $s_{\min}$ and $s_{\max}$, are assumed to be strictly positive. In the case of FOH, control rate constraints can be imposed element-wise, as shown in Equation (7.21h), which is derived as follows—consider the control rate, where the the control input is FOH-parameterized:

$$\dot{u}(\tau) = \mathring{u}(\tau)\,\frac{d\tau}{dt} = \frac{u_{k+1} - u_k}{s(\tau)} = \frac{u_{k+1} - u_k}{(1-\tau)\,s_k + \tau\,s_{k+1}}, \quad \tau \in [0,1] \tag{7.22}$$

From Equation (7.22), the control rate constraint can now be given as:

$$|\dot{u}(\tau)| \leq \dot{u}_{\max}, \qquad \forall\,\tau \in [0,1] \tag{7.23a}$$

$$\iff |u_{k+1} - u_k| \leq \dot{u}_{\max}\,((1-\tau)\,s_k + \tau\,s_{k+1}), \quad \forall\,\tau \in [0,1] \tag{7.23b}$$

$$\iff |u_{k+1} - u_k| \leq \dot{u}_{\max}\,\min\{s_k, s_{k+1}\} \tag{7.23c}$$

which can be equivalently written as the following convex constraints:

$$|u_{k+1} - u_k| \leq \dot{u}_{\max}\,s_k \tag{7.24a}$$

$$|u_{k+1} - u_k| \leq \dot{u}_{\max}\,s_{k+1} \tag{7.24b}$$

where $\dot{u}_{\max} \in \mathbb{R}^{n_u}$ is element-wise strictly positive. However, control rate constraints can be imposed in general (with both ZOH and FOH) by treating the control input in question as a state, its derivative as the new control input, and bounding this new control input.

Auxiliary Constraints While setting the boundary conditions of the auxiliary dynamical system for the continuous-time constraints to be equal to each other—as shown in Equation (7.14)—suffices for Equation (7.12) to hold, and in turn, for the continuous-time constraints to be satisfied at all (a.e.) times, the reformulated constraints (i) violate the linear independence constraint qualification (LICQ) at all feasible points and (ii) cause inexactness of the penalty term, the exactness of which is crucial to the solution method we describe in §7.2 [66, Subsection 3.1]. To mitigate these pathologies, we employ the relaxation shown in

Equation (7.21i), where ϵ_{CTCS} is a small positive number that is numerically significant (i.e., larger than machine precision) yet physically insignificant. See [66, Figure 4] for the effect of choice of ϵ_{CTCS} on continuous-time constraint violation for a real-world application, wherein $\epsilon_{\text{CTCS}} \approx 10^{-4}$ ensures that the continuous-time constraint violation is less than 1%.

Boundary Conditions We assume fixed boundary conditions on some or all of the states, as shown in Equation (7.21j) and (7.21k), where $E_{\text{i}} \in \mathbb{R}^{n_{\text{i}} \times n_x}$ and $E_{\text{f}} \in \mathbb{R}^{n_{\text{f}} \times n_x}$ are wide (or identity or zero) matrices that select components of the state, and $\tilde{x}_{\text{i}}$ and $\tilde{x}_{\text{f}}$ are the fixed boundary conditions, where $n_{\text{i}}, n_{\text{f}} \leq n_x$.

State-Triggered Constraints State-triggered constraints (STCs) [177, 208, 176] are conditional constraints of the form *trigger condition implies constraint condition*, i.e.:

$$g_{\text{STC}}(x(t)) \leq 0 \implies c_{\text{STC}}(x(t)) \leq 0 \tag{7.25}$$

where the trigger function, g_{STC}, and the constraint function, c_{STC}, are vector-valued functions that are not necessarily convex. The constraints themselves are referred to as the trigger condition and the constraint condition, respectively.

State-triggered constraints provide a continuous encoding of `if`-type constraints, which are mixed-integer constraints in their native form [132]. Another approach to handling discrete logical constraints in a continuous optimization framework is via homotopy [130]. Multiple STCs can be imposed together, either with the `AND` logic or the `OR` logic (or with a combination of them) [207, 204], and are referred to as compound STCs. Compound STCs can also be satisfied in continuous-time using the integral constraint reformulation [216].

Multi-Phase Trajectory Optimization For a class of state-triggered constraints, wherein (i) the trigger condition is only activated once and (ii) the trigger function is monotonic in a neighborhood around the trigger condition activation point, compound conditional constraints can also be formulated as *single-crossing* compound STCs, enabling multi-phase tra-

jectory optimization [103, 42, 109, 110]. Note that all state-triggered constraint formulations, except for the single-crossing compound STC formulation, after the continuous encoding, are inherently nonconvex, regardless of whether the trigger and constraint conditions are individually convex or not. However, single-crossing compound STCs are entirely convex in the special case where the trigger conditions are linear and the constraint conditions are convex, but do require time-interval dilation, which adds nonlinearity to the dynamics [103].

7.2 Successive Convexification

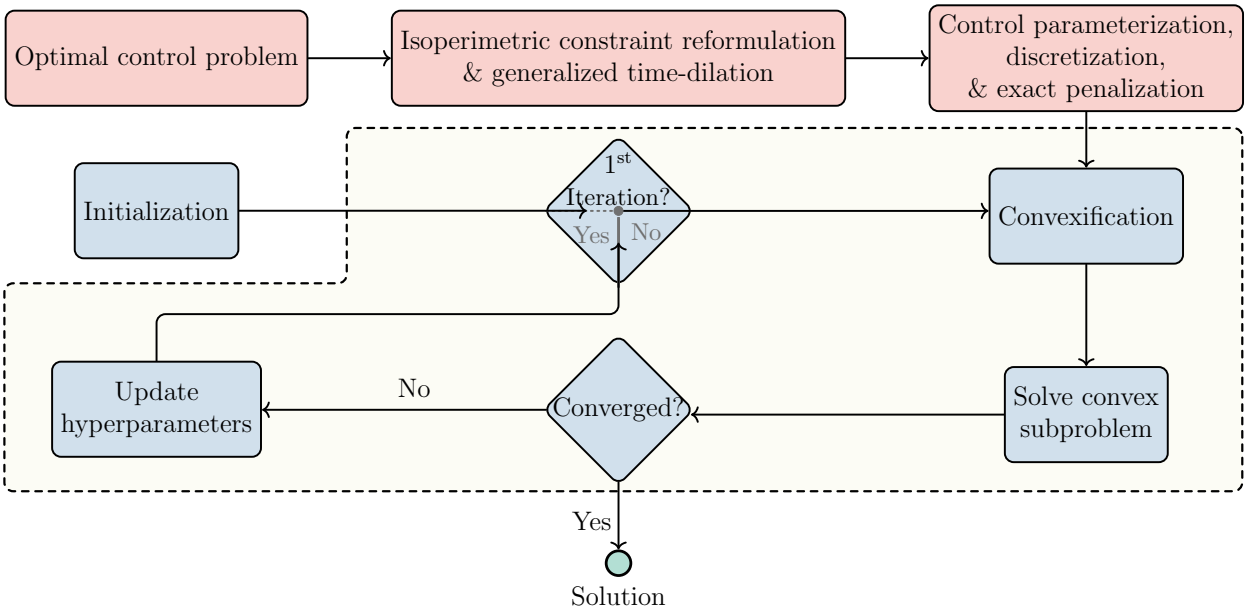

Figure 7.2: The successive convexification (SCvx) framework [66]. The ▭ blocks show the construction of optimal control problem to be solved, which is performed *offline*. The ▭ blocks show the components of the prox-linear method—with the ⬚ block demarcating the iterative components—which is executed *online*.

7.2.1 Convexification

Our solution method relies on solving a sequence of convex approximations to the original discrete-time nonconvex problem, Problem 7.21. To this end, we successively convexify the

problem by (i) linearizing the nonconvexity stemming from Equations (7.21b), (7.21c), and (7.21d), and (ii) preserving the remaining constraints that are convex to begin with.

Linearization of Nonlinear Dynamics

We linearize the discrete-time nonlinear dynamics in Equation (7.20b) in a matrix-inverse-free manner using the variational approach to multiple shooting [168, 85, 103, 66]. Consider the Jacobians of the right hand side (RHS) of Equation (7.18) with respect to the augmented state and augmented control, respectively, in a given wall-clock time-interval, $t \in [t_k, t_{k+1}]$, where $\tau \in [0, 1]$:

$$A(\tau) := \nabla_{(x,y)} F \left(\begin{bmatrix} x(\tau) \\ y(\tau) \end{bmatrix}, \begin{bmatrix} u(\tau) \\ s(\tau) \end{bmatrix} \right) \tag{7.26}$$

$$B(\tau) := \nabla_{(u,s)} F \left(\begin{bmatrix} x(\tau) \\ y(\tau) \end{bmatrix}, \begin{bmatrix} u(\tau) \\ s(\tau) \end{bmatrix} \right) \tag{7.27}$$

Consider the following reference trajectory:

$$\begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix}, \quad \begin{bmatrix} \bar{u}_k \\ \bar{s}_k \end{bmatrix}, \quad k = 1, \ldots, N \tag{7.28}$$

Linearizing the RHS of Equation (7.20b) around the reference trajectory (7.28) yields the following first-order approximation:

$$\begin{bmatrix} \Delta x_{k+1} \\ \Delta y_{k+1} \end{bmatrix} = A_k \begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix} + B_k^- \begin{bmatrix} \Delta u_k \\ \Delta s_k \end{bmatrix} + B_k^+ \begin{bmatrix} \Delta u_{k+1} \\ \Delta s_{k+1} \end{bmatrix} + \begin{bmatrix} \Delta x_{k+1}^{\text{prop}} \\ \Delta y_{k+1}^{\text{prop}} \end{bmatrix}, \quad k = 1, \ldots, N-1 \tag{7.29}$$

where, for $k = 1, \ldots, N$:

$$\begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix} := \begin{bmatrix} u_k - \bar{u}_k \\ s_k - \bar{s}_k \end{bmatrix} \tag{7.30}$$

$$\begin{bmatrix} \Delta u_k \\ \Delta s_k \end{bmatrix} := \begin{bmatrix} u_k - \bar{u}_k \\ s_k - \bar{s}_k \end{bmatrix} \tag{7.31}$$

and for $k = 1, \ldots, N-1$:

$$A_k := \Phi_A(1) \quad = I_{n_x} + \int_0^1 A(\tau)\, \Phi_A(\tau)\, d\tau \tag{7.32}$$

$$B_k^- := \Phi_{B^-}(1) = \int_0^1 \big(A(\tau)\, \Phi_{B^-}(\tau) + B(\tau)\, \sigma^-(\tau)\big)\, d\tau \tag{7.33}$$

$$B_k^+ := \Phi_{B^+}(1) = \int_0^1 \big(A(\tau)\, \Phi_{B^+}(\tau) + B(\tau)\, \sigma^+(\tau)\big)\, d\tau \tag{7.34}$$

$$\begin{bmatrix} \Delta x_{k+1}^{\text{prop}} \\ \Delta y_{k+1}^{\text{prop}} \end{bmatrix} := \begin{bmatrix} x_{k+1}^{\text{prop}} - \bar{x}_{k+1} \\ y_{k+1}^{\text{prop}} - \bar{y}_{k+1} \end{bmatrix} \tag{7.35}$$

where:

$$\mathring{\Phi}_A(\tau) := A(\tau)\, \Phi_A(\tau) \tag{7.36}$$

$$\mathring{\Phi}_{B^-}(\tau) := A(\tau)\, \Phi_{B^-}(\tau) + B(\tau)\, \sigma^-(\tau) \tag{7.37}$$

$$\mathring{\Phi}_{B^+}(\tau) := A(\tau)\, \Phi_{B^+}(\tau) + B(\tau)\, \sigma^+(\tau) \tag{7.38}$$

$$\begin{bmatrix} x_{k+1}^{\text{prop}} \\ y_{k+1}^{\text{prop}} \end{bmatrix} := \begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix} + F_k \left(\begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix}, \begin{bmatrix} \bar{u}_k \\ \bar{s}_k \end{bmatrix}, \begin{bmatrix} \bar{u}_{k+1} \\ \bar{s}_{k+1} \end{bmatrix} \right) \tag{7.39}$$

Linearization of Nonconvex Constraints

The nonconvex constraint functions in Equations (7.21c) and (7.21d), respectively, are linearized about the reference trajectory (7.28) to yield the following first-order approximations,

$k = 1, \ldots, N$:

$$g(x_k, u_k) = G_k^x \, \Delta x_k + G_k^u \, \Delta u_k + g_k \tag{7.40}$$

$$h(x_k, u_k) = H_k^x \, \Delta x_k + H_k^u \, \Delta u_k + h_k \tag{7.41}$$

where:

$$G_k^x := \nabla_x \, g(x_k, u_k) \Big|_{(\bar{x}_k, \, \bar{u}_k)} \tag{7.42}$$

$$G_k^u := \nabla_u \, g(x_k, u_k) \Big|_{(\bar{x}_k, \, \bar{u}_k)} \tag{7.43}$$

$$g_k := g(\bar{x}_k, \bar{u}_k) \tag{7.44}$$

$$H_k^x := \nabla_x \, h(x_k, u_k) \Big|_{(\bar{x}_k, \, \bar{u}_k)} \tag{7.45}$$

$$H_k^u := \nabla_u \, h(x_k, u_k) \Big|_{(\bar{x}_k, \, \bar{u}_k)} \tag{7.46}$$

$$h_k := h(\bar{x}_k, \bar{u}_k) \tag{7.47}$$

7.2.2 Pre-Solve

Scaling

Scaling of decision variables plays an important role in numerical optimization and numerical optimal control algorithms [155, 28, 134]. Although, in theory, convexification of the original optimal control problem is sufficient to be able to solve it, practice suggests otherwise [176]. Specifically, sequential convex programming (SCP)-based algorithms are typically not scale-invariant, and benefit significantly from good scaling, both numerically and with respect to optimality [134]. Finite precision arithmetic suffers when variables have largely varying relative magnitudes. Further, proximal optimization methods like successive convexification that rely on a proximal term ("a soft/penalized trust region" [174]), or on a trust region constraint ("a hard trust region" [134]), are sensitive to relative magnitude differences between the decision variables, especially in terms of meeting termination tolerances [132, 134].

The goal of variable scaling (scaling of variables) is to ensure all decision variables that the numerical solver sees are roughly on the same order of magnitude to mitigate the aforementioned undesirable effects. A popular variable scaling technique that has been widely used in the trajectory optimization literature is that of linear/affine scaling [174, 134].

Example. *As a representative example, consider two decision variables, thrust T and pointing angle δ. Let $T \leq T_{\max}$, where $T_{\max} = 2200$ kilo-Newtons, and $\delta \leq \delta_{\max}$, where $\delta_{\max} = 1^\circ \approx 0.0175$ radians be constraints imposed [103]. A change of 10% the maximum magnitude would be 220000 N for T and 0.00175 radians for δ, the values for which are on completely different orders of magnitude, and, if provided to a numerical solver directly, would result in numerical errors, skewed trust regions, and inconsistent convergence checks (given a termination tolerance). Now consider the scaled variables $\hat{T}$ and $\hat{\delta}$, and the corresponding linear scaling factors $P_T := T_{\max}$ and $P_\delta := \delta_{\max}$, respectively, such that $T = P_T \hat{T}$ and $\delta = P_\delta \hat{\delta}$. Treating $\hat{T}$ and $\hat{\delta}$ as the decision variables instead would ensure that both decision variables have a maximum magnitude of roughly unity, and would mitigate the aforementioned issues.*

We aim to equivalently transform the problem to one in which the decision variables have, roughly, a magnitude less than unity. Consider the following, $k = 1, \ldots, N$:

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = P_x \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix} \tag{7.48}$$

$$\begin{bmatrix} u_k \\ s_k \end{bmatrix} = P_u \begin{bmatrix} \hat{u}_k \\ \hat{s}_k \end{bmatrix} \tag{7.49}$$

where $P_x \succ 0 \in \mathbb{R}^{n_x+1}$ and $P_u \succ 0 \in \mathbb{R}^{n_u+1}$ are positive-definite diagonal scaling matrices, and $\hat{x}_k$, $\hat{y}_k$, $\hat{u}_k$, and $\hat{s}_k$ are the scaled absolute variables. Subtracting the reference trajectory

from both sides of Equations (7.48) and (7.49), we get:

$$\begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix} = \begin{bmatrix} x_k - \bar{x}_k \\ y_k - \bar{y}_k \end{bmatrix} = P_x \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix} - \begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix} = P_x \underbrace{\left(\begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix} - P_x^{-1} \begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix} \right)}_{:= \Delta \hat{x}_k} \tag{7.50}$$

$$\begin{bmatrix} \Delta u_k \\ \Delta s_k \end{bmatrix} = \begin{bmatrix} u_k - \bar{u}_k \\ s_k - \bar{s}_k \end{bmatrix} = P_u \begin{bmatrix} \hat{u}_k \\ \hat{s}_k \end{bmatrix} - \begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix} = P_u \underbrace{\left(\begin{bmatrix} \hat{u}_k \\ \hat{s}_k \end{bmatrix} - P_u^{-1} \begin{bmatrix} \bar{x}_k \\ \bar{y}_k \end{bmatrix} \right)}_{:= \Delta \hat{u}_k} \tag{7.51}$$

We treat the scaled deviation variables, $\Delta \hat{x}_k$ and $\Delta \hat{u}_k$, $k = 1, \ldots, N$, as the decision variables.

Equation (7.29) can be equivalently written as in terms of the scaled variables as:

$$P_x \begin{bmatrix} \Delta \hat{x}_{k+1} \\ \Delta \hat{y}_{k+1} \end{bmatrix} = A_k \, P_x \begin{bmatrix} \Delta \hat{x}_k \\ \Delta \hat{y}_k \end{bmatrix} + B_k^- \, P_u \begin{bmatrix} \Delta \hat{u}_k \\ \Delta \hat{s}_k \end{bmatrix} + B_k^+ \, P_u \begin{bmatrix} \Delta \hat{u}_{k+1} \\ \Delta \hat{s}_{k+1} \end{bmatrix} + \begin{bmatrix} \Delta x_{k+1}^{\text{prop}} \\ \Delta y_{k+1}^{\text{prop}} \end{bmatrix}, \quad k = 1, \ldots, N-1 \tag{7.52}$$

Similarly, the constraints can be equivalently written in terms of the scaled variables.

Automatic Scaling Typically, optimal control problems contain enough information for the determination of good scaling factors for the decision variables [132]. For the control input and dilation factor variables, the (absolute value of) user-chosen element-wise bounds (Equations (7.21f) and (7.21g)) serve as good scaling factors. For the augmented-state variables, the scaling matrix can either be predetermined based on problem data, or can be automatically computed on-the-fly, without the need for user intervention, as follows:

$$P_x = \begin{bmatrix} \max\{\|^1 x^{\text{prop}}\|_\infty, 1\} & & & \\ & \ddots & & \\ & & \max\{\|^{n_x} x^{\text{prop}}\|_\infty, 1\} & \\ & & & \max\{\|y^{\text{prop}}\|_\infty, 1\} \end{bmatrix} \in \mathbb{R}^{n_x+1} \tag{7.53}$$

where ${}^{i}x$, $i = 1, \ldots, n_x$, and y are the scalar components of the augmented state, and ${}^{i}x^{\text{prop}}$, $i = 1, \ldots, n_x$, and y^{prop} are the respective propagated multiple shooting trajectories, which are accessible at every nonconvex solver iteration.

Preconditioning

While scaling of variables helps ensure they are on the same order of magnitude, it can adversely affect the numerical conditioning of the resulting problem data. Moreover, the constraints and objective function can be ill-conditioned to begin with. Consider the scaled variables from the example in 7.2.2. Assembling the constraints in the example into a canonical affine inequality constraint that a numerical solver would see, we get:

$$\begin{bmatrix} P_T & 0 \\ 0 & P_\delta \end{bmatrix} \begin{bmatrix} \hat{T} \\ \hat{\delta} \end{bmatrix} \leq \begin{bmatrix} T_{\max} \\ \delta_{\max} \end{bmatrix} \tag{7.54}$$

where the constraint matrix has a condition number of $\frac{P_T}{P_\delta} \approx 1.26 \cdot 10^8$, making the problem ill-conditioned. The affine inequality constraint in Equation (7.54) can be *preconditioned* by normalizing the rows of the constraint matrix, which would yield:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{T} \\ \hat{\delta} \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{7.55}$$

which is equivalent to Equation (7.54), but now, with a condition number of unity.

Preconditioning is a heuristic that seeks to compute an invertible transformation that minimizes the condition number of the matrix in question, often employed to improve the performance of iterative numerical methods [232, 22]. In numerical optimization, diagonal preconditioning and matrix equilibration are popular preconditioning techniques [162, 80, 82]. Specialized preconditioners have also been developed for quadratic cone programs with strongly convex objective functions [52, 101]. Perhaps the most commonly used preconditioning technique in off-the-shelf convex solvers, however, is an iterative method called

Ruiz equilibration [183], modified with a cost-scaling step [201, 59, 50]. While most modern convex solvers are inherently robust to ill-conditioning and/or possess internal preconditioners at the solver level, we implement the proposed preconditioning and objective function scaling techniques to mitigate ill-conditioning in the problem even before it is passed to the convex solver, which we have observed to be useful, especially when first-order convex solvers are used.

For the prescaled dynamics in Equation (7.52), we first complete the similarity transformation by multiplying throughout by P_x^{-1}:

$$\begin{bmatrix} \Delta\hat{x}_{k+1} \\ \Delta\hat{y}_{k+1} \end{bmatrix} = P_x^{-1}\, A_k\, P_x \begin{bmatrix} \Delta\hat{x}_k \\ \Delta\hat{y}_k \end{bmatrix} + P_x^{-1}\, B_k^-\, P_u \begin{bmatrix} \Delta\hat{u}_k \\ \Delta\hat{s}_k \end{bmatrix} + P_x^{-1}\, B_k^+\, P_u \begin{bmatrix} \Delta\hat{u}_{k+1} \\ \Delta\hat{s}_{k+1} \end{bmatrix} + P_x^{-1} \begin{bmatrix} \Delta x_{k+1}^{\text{prop}} \\ \Delta y_{k+1}^{\text{prop}} \end{bmatrix}$$
$$k = 1, \ldots, N-1 \tag{7.56}$$

Then, for all constraints in the convexified problem, including the similarity-transformed dynamics in Equation (7.56), we apply block row-normalization from [101, Section II.B] in the ℓ_∞-norm sense, which normalizes the rows of the affine constraints, and scales each scalar-valued convex conic constraint by a positive scalar. Further, we divide the objective function by a function of the largest magnitude coefficient of its terms. The combined effect of the preconditioning procedure we employ is similar to that of modified Ruiz equilibration [201]. However, unlike modified Ruiz equilibration, the method we use: (i) only normalizes the rows of the constraint matrix rather than fully equilibrating it, i.e., rather than ensuring all rows have the same norm and all columns have the same norm, and (ii) is not iterative.

7.2.3 Prox-Linear Method

Within the successive convexification framework, we adopt the prox-linear method [60, 61], a convergence-guaranteed sequential convex programming (SCP) method. The prox-linear method is closely related to the penalized trust region (PTR) algorithm popularly used in trajectory optimization [203, 177, 132, 208, 176, 135], the connection between which is

described in [65, Remark 26].

A notable characteristic of this method is that it only requires the nonlinear dynamics and nonconvex constraints to be once continuously differentiable ($\mathcal{C}^1$), as opposed to twice continuously differentiable ($\mathcal{C}^2$), as is the case with sequential quadratic programming (SQP)-based methods [236, 87, 164].

Exact Penalty

Let $z = ((\hat{x}_1, \hat{y}_1), \ldots, (\hat{x}_N, \hat{y}_N), (\hat{u}_1, \hat{s}_1), \ldots, (\hat{u}_N, \hat{s}_N)) \in \mathbb{R}^{((nx+1)+(nu+1))N}$ be the stacked scaled absolute variable vector. Consider the following stacked version of Problem 7.21:

$$\begin{aligned} \underset{z \in \Omega}{\text{minimize}} \quad & C(z) && \text{(7.57a)} \\ \text{subject to} \quad & D_{\text{eq}}(z) = 0 && \text{(7.57b)} \\ & D_{\text{ineq}}(z) \leq 0 && \text{(7.57c)} \end{aligned}$$

where Ω is a convex set corresponding to all the convex constraints, $C(z)$ is the convex cost, $D_{\text{eq}}(z)$ includes all the nonconvex equality constraints, and $D_{\text{ineq}}(z)$ includes all the nonconvex inequality constraints in Problem 7.21.

Now, the penalized problem is defined as follows:

$$\underset{z \in \Omega}{\text{minimize}} \quad C(z) + \underbrace{\lambda_{\text{eq}}^\top |D_{\text{eq}}(z)| + \lambda_{\text{ineq}}^\top \text{ReLU}(D_{\text{ineq}}(z))}_{\text{exact penalty term}} \tag{7.58}$$

where λ_{eq} and λ_{ineq} are element-wise positive vectors of appropriate dimensions. For large enough, finite values of λ_{eq} and λ_{ineq} (element-wise), it can be shown that any solution to Problem 7.58, in which the exact penalty term evaluates to zero, is also a solution to Problem 7.57 [57].

Penalized Trust Region

The prox-linear method computes a stationary point of Problem 7.58 by solving a sequence of convex approximations to Problem 7.58, which is originally nonconvex due to terms $D_{\text{eq}}(z)$ and $D_{\text{ineq}}(z)$. At each iteration in the sequence, the convex subproblem is given by:

$$\begin{aligned}\underset{z\in\Omega}{\text{minimize}}\quad & C(z) + \underbrace{\lambda_{\text{eq}}^{\top}\left|D_{\text{eq}}(\bar{z}) + \nabla D_{\text{eq}}(\bar{z})\,\Delta z\right| + \lambda_{\text{ineq}}^{\top}\,\text{ReLU}(D_{\text{ineq}}(\bar{z}) + \nabla D_{\text{ineq}}(\bar{z})\,\Delta z)}_{\text{exact penalty term (convexified)}} \\ & + \underbrace{w_{\text{trust}}\,\frac{1}{2}\,\|\Delta z\|^2}_{\text{penalized trust region}}\end{aligned} \tag{7.59}$$

where $w_{\text{trust}} > 0$ is the penalized trust region weight, $\Delta z := z - \bar{z}$, and $\bar{z}$ is the scaled reference solution, i.e., the solution to the convex subproblem at the previous iteration in the sequence. While the prox-linear method is not scale-invariant (choice of scaling affects the penalized trust region term in Problem 7.59), it is preconditioning-invariant.

Convex Subproblem

After convexification of Problem 7.21 (§7.2.1), carrying out the pre-solve steps (§7.2.2), and with the exact penalty and penalized trust region modifications (§7.2.3), the stacked convex problem, Problem 7.59, can be written explicitly as follows:

Problem 7.60: Convex Subproblem

$$
\begin{aligned}
& w_{\text{cost}} \left(\sum_{k=1}^{N} \hat{L}(\Delta\hat{x}_k + \hat{\bar{x}}_k, \Delta\hat{u}_k + \hat{\bar{u}}_k) \;\text{ or }\; \hat{e}_c^{\top}(\Delta\hat{x}_N + \hat{\bar{x}}_N) \;\text{ or }\; \sum_{k=1}^{N-1} \frac{(\Delta\hat{s}_k + \hat{\bar{s}}_k) + (\Delta\hat{s}_{k+1} + \hat{\bar{s}}_{k+1})}{2} \right) \\
\underset{\substack{\Delta\hat{u}_1,\ldots,\Delta\hat{u}_N,\\ \Delta\hat{s}_1,\ldots,\Delta\hat{s}_N}}{\text{minimize}} \quad & + \sum_{k=1}^{N-1} \lambda_k^{\top}\left(\nu_k^{+} + \nu_k^{-}\right) + \sum_{k=1}^{N} \left(\lambda_{g_k}^{\top}\nu_{g_k} + \lambda_{h_k}^{\top}\left(\nu_{h_k}^{+} + \nu_{h_k}^{-}\right)\right) && \text{(7.60a)} \\
& + w_{\text{trust}} \frac{1}{2} \sum_{k=1}^{N} \left(\|\Delta\hat{x}_k\|_2^2 + (\Delta\hat{y}_k)^2 + \|\Delta\hat{u}_k\|_2^2 + (\Delta\hat{s}_k)^2 \right) \\
\text{subject to} \quad & \hat{A}_k^{-}\begin{bmatrix}\Delta\hat{x}_k \\ \Delta\hat{y}_k\end{bmatrix} + \hat{A}_k^{+}\begin{bmatrix}\Delta\hat{x}_{k+1} \\ \Delta\hat{y}_{k+1}\end{bmatrix} + \hat{B}_k^{-}\begin{bmatrix}\Delta\hat{u}_k \\ \Delta\hat{s}_k\end{bmatrix} + \hat{B}_k^{+}\begin{bmatrix}\Delta\hat{u}_{k+1} \\ \Delta\hat{s}_{k+1}\end{bmatrix} + E_k\begin{bmatrix}\Delta x_{k+1}^{\text{prop}} \\ \Delta y_{k+1}^{\text{prop}}\end{bmatrix} = E_k\left(\nu_k^{+} - \nu_k^{-}\right), \quad k = 1,\ldots,N-1 \\
& && \text{(7.60b)} \\
& \left(\nu_k^{+}, \nu_k^{-}\right) \geq 0, \qquad k = 1,\ldots,N-1 \\
& && \text{(7.60c)} \\
& \hat{G}_k^{x}\Delta\hat{x}_k + \hat{G}_k^{u}\Delta\hat{u}_k + E_{g_k} g_k \leq E_{g_k}\nu_{g_k}, \qquad k = 1,\ldots,N && \text{(7.60d)} \\
& \nu_{g_k} \geq 0, \qquad k = 1,\ldots,N && \text{(7.60e)} \\
& \hat{H}_k^{x}\Delta\hat{x}_k + \hat{H}_k^{u}\Delta\hat{u}_k + E_{h_k} h_k = E_{h_k}\left(\nu_{h_k}^{+} - \nu_{h_k}^{-}\right), \qquad k = 1,\ldots,N && \text{(7.60f)} \\
& \left(\nu_{h_k}^{+}, \nu_{h_k}^{-}\right) \geq 0, \qquad k = 1,\ldots,N && \text{(7.60g)} \\
& (\Delta\hat{x}_k, \Delta\hat{u}_k) \in \hat{\mathcal{Z}}_k, \qquad k = 1,\ldots,N && \text{(7.60h)} \\
& \hat{u}_{\min} \leq \Delta\hat{u}_k + \hat{\bar{u}}_k \leq \hat{u}_{\max}, \qquad k = 1,\ldots,N && \text{(7.60i)} \\
& \hat{s}_{\min} \leq \Delta\hat{s}_k + \hat{\bar{s}}_k \leq \hat{s}_{\max}, \qquad k = 1,\ldots,N && \text{(7.60j)} \\
& \left|(\Delta\hat{u}_{k+1} + \hat{\bar{u}}_{k+1}) - (\Delta\hat{u}_k + \hat{\bar{u}}_k)\right| \leq \hat{u}_{\max} \min\{\Delta\hat{s}_k + \hat{\bar{s}}_k,\, \Delta\hat{s}_{k+1} + \hat{\bar{s}}_{k+1}\}, \qquad k = 1,\ldots,N-1 \\
& && \text{(7.60k)} \\
& (\Delta\hat{y}_{k+1} + \hat{\bar{y}}_{k+1}) - (\Delta\hat{y}_k + \hat{\bar{y}}_k) \leq \hat{\epsilon}_{\text{LICQ}}, \qquad k = 1,\ldots,N-1 && \text{(7.60l)} \\
& \hat{E}_{\text{i}}(\Delta\hat{x}_1 + \hat{\bar{x}}_1) = \hat{\bar{x}}_{\text{i}} && \text{(7.60m)} \\
& \hat{E}_{\text{f}}(\Delta\hat{x}_N + \hat{\bar{x}}_N) = \hat{\bar{x}}_{\text{f}} && \text{(7.60n)}
\end{aligned}
$$

where $\hat{\square}$ is used to denote scaled variables and preconditioned problem data, ν_k^{+}, ν_k^{-}, $k = 1,\ldots,N-1$ and $\nu_{h_k}^{+}$, $\nu_{h_k}^{-}$, $k = 1,\ldots,N$, are the slack variables associated with the ℓ_1-norm penalty to linear cost reformulation for the linearized equality constraints [65], and ν_{g_k}, $k = 1,\ldots,N$, are the slack variables for the linearized inequality constraints. The diagonal positive definite matrices, $E_k \in \mathbb{R}^{n_x+1}$, $k = 1,\ldots,N-1$, are constructed by multiplying the row-normalization coefficient matrix for the augmented dynamics by P_x^{-1}, to remain consistent with the similarity-transformation. The diagonal positive definite matrices, $E_{g_k} \in \mathbb{R}^{n_g}$ and $E_{h_k} \in \mathbb{R}^{n_h}$, $k = 1,\ldots,N$, contain the row-normalization coefficients for the linearized inequality and equality constraints, respectively. The vector exact penalty weights, λ_k, $k = 1,\ldots,N-1$, and λ_{h_k}, $k = 1,\ldots,N$, are elements of λ_{eq} in Problem 7.59, and λ_{g_k}, $k = 1,\ldots,N$, are elements of λ_{ineq} in Problem 7.59.

7.2.4 Hyperparameter Auto-Tuning

Exact Penalty Auto-Tuning

SCP algorithms used in trajectory optimization have typically made use of scalar weights for the penalty term, which are computed by performing a line search (by trial-and-error) [134, 66]. However, recently, primal-dual SCP algorithms have been introduced [144, 145, 126], which consider *auto-tuned* vector penalty weights (dual variables), the benefits of which are two-fold: (i) the penalty weight being a vector provides additional degrees-of-freedom over a single scalar weight, and (ii) these vector weights are auto-tuned based on the amount of constraint violation. The associated weight updates are similar are akin to the weight updates in augmented Lagrangian methods [91, 163]. Specifically, at each iteration of the prox-linear method, we consider the following weight update rules based on the integration of the amount of violation in the nonconvex constraints:

$$\lambda_k = \lambda_k + \alpha \left|\Delta x_{k+1}^{\text{prop}}\right|, \qquad k = 1, \ldots, N-1 \tag{7.61}$$

$$\lambda_{g_k} = \lambda_{g_k} + \alpha \, \text{ReLU}(g_k), \qquad k = 1, \ldots, N \tag{7.62}$$

$$\lambda_{h_k} = \lambda_{h_k} + \alpha \, |h_k|, \qquad k = 1, \ldots, N \tag{7.63}$$

$$\alpha = \alpha \, \beta \tag{7.64}$$

where $\alpha \geq 0$ is the integration step-size, and $\beta \geq 0$ is the step-size multiplier [55].

Adaptive Trust Region Weighting

To ensure convergence of the prox-linear algorithm, the trust region penalty must be scaled in proportion to the Lipschitz constant of the penalized cost function [60, Lemma 5.1]. In practice, however, estimating this constant in advance is prohibitively expensive, and even when available, such a choice of penalty weight often leads to extremely slow convergence [215]. To overcome these difficulties, we employ an adaptive trust region weighting strategy [45, 215], as shown in Algorithm 17. In this approach, the trust region weight is adjusted

dynamically after each prox-linear iteration: it is increased when the convexification becomes inaccurate, reflecting a locally larger Lipschitz constant, and decreased when the convexification accuracy improves. The method can also reject candidate solutions if the nonconvex cost fails to decrease, thereby balancing progress with stability. Figure 7.3 depicts the convexification accuracy metric, the terms in which are defined in the equations that follow.

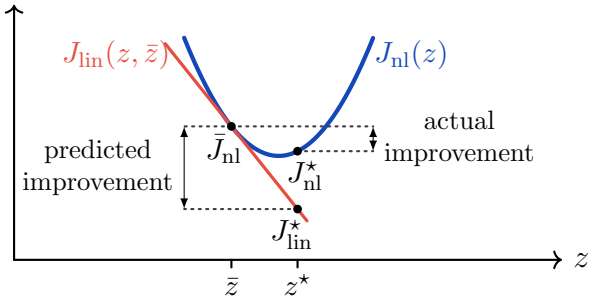

Figure 7.3: Depiction of the convexification accuracy metric, $\rho_{\text{cvx}} := \frac{\textbf{actual improvement}}{\textbf{predicted improvement}} = \frac{\bar{J}_{\text{nl}} - J^\star_{\text{nl}}}{\bar{J}_{\text{nl}} - J^\star_{\text{lin}}}$.

$$\bar{J}_{\text{nl}} := w_{\text{cost}}\, \bar{J} + \sum_{k=1}^{N-1} \lambda_k^\top \begin{bmatrix} |\Delta \bar{x}_{k+1}^{\text{prop}}| \\ |\Delta \bar{y}_{k+1}^{\text{prop}}| \end{bmatrix} + \sum_{k=1}^{N} \left(\lambda_{g_k}^\top \text{ReLU}(\bar{g}_k) + \lambda_{h_k}^\top |\bar{h}_k| \right) \tag{7.65}$$

$$J^\star_{\text{nl}} := w_{\text{cost}}\, J + \sum_{k=1}^{N-1} \lambda_k^\top \begin{bmatrix} |\Delta x_{k+1}^{\text{prop}}| \\ |\Delta y_{k+1}^{\text{prop}}| \end{bmatrix} + \sum_{k=1}^{N} \left(\lambda_{g_k}^\top \text{ReLU}(g_k) + \lambda_{h_k}^\top |h_k| \right) \tag{7.66}$$

$$\begin{aligned} J^\star_{\text{lin}} := w_{\text{cost}}\, J &+ \sum_{k=1}^{N-1} \lambda_k^\top (\nu_k^+ + \nu_k^-) + \sum_{k=1}^{N} \left(\lambda_{g_k}^\top \nu_{g_k} + \lambda_{h_k}^\top (\nu_{h_k}^+ + \nu_{h_k}^-) \right) \\ &+ w_{\text{trust}} \frac{1}{2} \sum_{k=1}^{N} \left(\|\Delta \hat{x}_k\|_2^2 + (\Delta \hat{y}_k)^2 + \|\Delta \hat{u}_k\|_2^2 + (\Delta \hat{s}_k)^2 \right) \end{aligned} \tag{7.67}$$

$$\rho_{\text{cvx}} := \frac{\text{actual improvement}}{\text{predicted improvement}} = \frac{\bar{J}_{\text{nl}} - J^\star_{\text{nl}}}{\bar{J}_{\text{nl}} - J^\star_{\text{lin}}} \tag{7.68}$$

Algorithm 17 Adaptive Trust Region Weighting

Inputs: w_{trust}, $\Delta z^\star$,

$\bar{J}_{\text{nl}}$, $J^\star_{\text{nl}}$, $J^\star_{\text{lin}}$, $0 \leq \rho_{\text{cvx}_0} < \rho_{\text{cvx}_1} < \rho_{\text{cvx}_2}$, $0 < \beta_{\text{decrease}} < 1 < \beta_{\text{increase}}$, $w_{\text{trust}_{\min}}$, $w_{\text{trust}_{\max}}$

`/* Determine Convexification Accuracy */`

predicted improvement $\leftarrow \bar{J}_{\text{nl}} - J^\star_{\text{lin}}$

if predicted improvement is 0 **then**

  Return: w_{trust}, Δz

else

  actual improvement $\leftarrow \bar{J}_{\text{nl}} - J^\star_{\text{nl}}$

  $\rho_{\text{cvx}} \leftarrow \frac{\text{actual improvement}}{\text{predicted improvement}}$

end if

`/* Solution Acceptance */`

if $\rho_{\text{cvx}} < \rho_{\text{cvx}_0}$ **then**

  $\Delta z \leftarrow 0$ ▷ reject solution

else

  $\Delta z \leftarrow \Delta z^\star$ ▷ accept solution

end if

`/* Update Trust Region Weight */`

if $\rho_{\text{cvx}} < \rho_{\text{cvx}_1}$ **then**

  $w_{\text{trust}} \leftarrow \min(\beta_{\text{increase}}\, w_{\text{trust}},\, w_{\text{trust}_{\max}})$ ▷ increase trust region weight

else if $\rho_{\text{cvx}} \in [\rho_{\text{cvx}_1}, \rho_{\text{cvx}_2})$ **then**

  $w_{\text{trust}} \leftarrow w_{\text{trust}}$

else if $\rho_{\text{cvx}} \geq \rho_{\text{cvx}_2}$ **then**

  $w_{\text{trust}} \leftarrow \max(\beta_{\text{decrease}}\, w_{\text{trust}},\, w_{\text{trust}_{\min}})$ ▷ decrease trust region weight

end if

Return: w_{trust}, Δz

7.3 Software

In this section, we describe the SCvxGEN software. Specifically, we describe the high-level user interface in §7.3.1 and automatic custom code generation in §7.3.2.

7.3.1 Modeling Interface

The modeling interface is written in the Python programming language [220], and consists of a user-friendly custom domain specific language (DSL) that relies on SymPy [151]—a popular computer algebra system (CAS) in Python—for symbolic manipulation and automatic generation of Jacobians.

A notable feature of this interface is that the user only needs to provide the true optimal control problem definition in continuous-time. All the required problem reformulations, including the integral constraint reformulation (§7.1.1), time-dilation (§7.1.2), and control parameterization and discretization (§7.1.3), are automatically handled by the software. An example optimal control problem definition is provided in Listing 7.1.

7.3.2 Automatic Custom Code Generation

Once the optimal control problem is defined via the high-level user interface and is parsed and reformulated by the software, as described in §7.3.1, the software automatically generates custom code for the entire successive convexification stack, from default initial-guess generation to solution. This includes code generation for: (i) the Jacobians of the augmented system via SymPy, (ii) the convex subproblem solver either directly (PIPG) or via CVXPYGEN [189] (for off-the-shelf solvers, including the default solver, QOCO), and (iii) the rest of the stack, including custom linear algebra routines, preconditioning, multiple-shooting, and fixed-step RK4 integration routines, via custom templates. We provide the option to interface with BLAS routines [244] as well.

A key feature of our software is that the C code it generates is highly readable—it uses the same symbols that the user defines, for example—and can be modified post-code-generation

if needed. Further, it is possible to generate library-free C code that uses static memory allocations only (see Remark 9), which is desirable for safety-critical applications, especially in the field of aerospace [107, 92, 73].

Automatic Convexity Detection

The general optimal control problem template we consider is convex when the conditions detailed in Remark 8 hold. Specifically, for a given free-final-time optimal control problem devoid of continuous-time constraints to be a convex optimal control problem, it suffices to check for linearity (affineness) in the dynamics (Equation (7.21b)), the inequality constraints (Equation (7.21c)), and the equality constraints (Equation (7.21d)). We assume that the nonlinear convex constraints are explicitly imposed via Equation (7.21e). To determine whether a function is affine or not in practice, we leverage the fact that a $\mathcal{C}^1$ function, $l(z)$, is affine in z if and only if its Jacobian, $\nabla_z l(z)$, is constant. If linearity is detected in a function, it is imposed as a convex constraint, and is not exact-penalized. Further, if a problem is deemed to be entirely convex, custom code is generated accordingly without exact penalization or a penalized trust region.

Customized Convex Solvers

SCvxGEN provides the ability to interface with off-the-shelf convex solvers, such as QOCO [50], that are supported by CVXPYGEN, a code generator for convex optimization solvers [189]. SCvxGEN also provides the ability to generate $\text{PIPG}_{\text{custom}}$, a customized first-order solver based on the proportional-integral projected gradient (PIPG) method [237, 239, 238], generated directly for the problem at hand. QOCO, the quadratic objective conic optimizer, is the default convex optimization solver in SCvxGEN. Altogether, QOCO, $\text{QOCO}_{\text{custom}}$, and $\text{PIPG}_{\text{custom}}$, are our in-house-developed convex optimization solvers.

QOCO and QOCOGEN QOCO is a state-of-the-art convex optimization solver written in C. It is based on an interior-point method (IPM), and is designed to solve feasible

second-order cone programs (SOCPs) with quadratic objectives, making it ideal for the subproblems that arise in successive convexification. QOCOGEN is a solver generator that produces a customized C-based solver (called $\text{QOCO}_{\text{custom}}$), based on the same algorithm as QOCO, but tailored to a specific problem family. By exploiting the fixed sparsity structure of the problem, $\text{QOCO}_{\text{custom}}$ achieves significantly faster solves solve-times than both QOCO and other SOCP solvers in general [50]. This is especially advantageous in successive convexification, wherein the subproblems share the same sparsity pattern across all iterations. While $\text{QOCO}_{\text{custom}}$ can be significantly faster than QOCO, it comes at the cost of often requiring longer compile-times, owing to code-customization. That said, the fact that they share the same algorithm makes QOCO ideal for prototyping and $\text{QOCO}_{\text{custom}}$ ideal for deployment in high-performance applications, wherein the solver only needs to be compiled once. Further, being based on an IPM, QOCO and $\text{QOCO}_{\text{custom}}$ are numerically robust and are suitable for high-accuracy solves. Both QOCO and QOCOGEN are available via the CVXPYGEN interface.

PIPG PIPG is a first-order method for convex optimization that has recently emerged as a fast solver for specialized trajectory optimization applications [64, 240, 103, 136]. The algorithm in PIPG itself can be directly customized to the sparsity structure of trajectory optimization problems, and the problem-specific solver thus generated, $\text{PIPG}_{\text{custom}}$, often leads to the simplest source-code and fastest compilation times when compared with off-the-shelf solvers, making it well-suited for safety-critical guidance and control applications requiring strict verification and validation (V&V) procedures [125, 58, 100]. Further, this simplicity has also aided in the demonstration of SCvxGEN on a GPU, via the generation of CUDA code [51]. See [64, 51, 100] for customized implementations of PIPG, and [51], in particular, for the customized implementation used within SCvxGEN. While PIPG has been shown to achieve fast solve-times for specialized applications that only require moderate-accuracy solves, it often requires extensive manual tuning, which has led to research on preconditioning for quadratic cone programs to improve its performance [52, 101]. As such, we provide $\text{PIPG}_{\text{custom}}$ as an experimental solver within SCvxGEN.

Remark 9. *When* QOCO$_{custom}$ *or* PIPG$_{custom}$ *and the provided custom linear algebra routines are used, the generated C codebase is entirely library-free and involves static memory allocations only. Further, with* PIPG$_{custom}$*, the entire successive convexification stack is matrix-inversion- and factorization-free.*

The Main Algorithm

The main algorithm that the C code generated by SCvxGEN implements is given by Algorithm 18. For consistency, we refer to the main nonconvex solution method/framework, with the following attributes:

1. The applied integral constraint reformulation (§7.1.1)
2. Generalized time-dilation (§7.1.2)
3. Control parameterization and multiple-shooting discretization/linearization (§7.1.3 & §7.2.1)
4. The prox-linear method with exact penalization (§7.2.3)
5. Exact penalty auto-tuning (§7.2.4) and adaptive trust region weighting (§7.2.4)

as, simply, *successive convexification* (SCvx). Being an SCP-based method, SCvx requires an initial guess, which can either be constructed automatically as a straight-line between user-chosen endpoints for each element in the state and control input vectors, or be provided manually. The solution is deemed to have converged when the following conditions are met:

$$\|\Delta z^\star\|_\infty \leq \delta_{\text{trust}} \tag{7.69}$$

$$\sum_{k=1}^{N-1} \left(\nu_k^+ + \nu_k^-\right) + \sum_{k=1}^{N} \nu_{g_k} + \sum_{k=1}^{N} \left(\nu_{h_k}^+ + \nu_{h_k}^-\right) \leq \delta_{\text{penalty}} \tag{7.70}$$

where $\Delta z = \Delta z^\star$ is the candidate solution (see §7.2.3), and $\delta_{\text{trust}} > 0$ and $\delta_{\text{penalty}} > 0$ are the chosen trust region and penalty term convergence tolerances, respectively.

Algorithm 18 Successive Convexification (SCvx) via SCvxGEN

`/* Initial Guess */`
Generate initial guess trajectory
for $k = 1, \ldots,$ `max_iters` **do**
`/* Convexification */` ▷ §7.2.1
Linearize augmented dynamics ▷ §7.2.1
Linearize nonconvex node constraints ▷ §7.2.1
`/* Prescaling */` ▷ §7.2.2
Prescale trajectory
Prescale augmented dynamics
Prescale node constraints
`/* Preconditioning */` ▷ §7.2.2
Precondition augmented dynamics
Precondition node constraints
`/* Exact-Penalty Auto-Tuning */` ▷ §7.2.4
Update dual variables
`/* Solution */`
Scale objective function
Solve convex subproblem ▷ solve Problem 7.60
Unscale objective function
`/* Adaptive Trust Region Weighting */` ▷ §7.2.4
Update trust region weight
`/* Post-Processing */`
Unscale trajectory
Evaluate termination criteria
end for

```
import scvxgen as scvx

# Parameters
parameters = scvx.Parameters('g')
g, = parameters

# States
states = scvx.States('x y v')
x, y, v = states

# Controls
controls = scvx.Controls('theta')
theta, = controls

# Symbolic entities
sin = scvx.sympy.sin
cos = scvx.sympy.cos
pi = float(scvx.sympy.pi)

# Dynamics
dynamics = scvx.Dynamics(
     v * sin(theta),
    -v * cos(theta),
     g * cos(theta)
)

# Default numerical values
time = scvx.Time(guess=2, t_min_factor=0.25, t_max_factor=5) # temporal attributes
parameters.set_value(9.81)                                   # acceleration due to gravity (Earth)
states.set_initial(0, 10, 0)                                 # initial condition
states.set_final(10, 5, scvx.Free())                         # final condition
controls.set_initial(5 * pi / 180)                           # initial-guess start point
controls.set_final(100.5 * pi / 180)                         # initial-guess end point
controls.set_minimum(0)                                      # lower bound
controls.set_maximum(100.5 * pi / 180)                       # upper bound

# Settings
settings = scvx.Settings(N_disc=2)

# Set up problem
problem = scvx.Problem(
    parameters=parameters, states=states, controls=controls,
    time=time, dynamics=dynamics, settings=settings
)

# Generate C code and compile it (once)
problem.generate_code()
problem.compile_code()

# Solve problem (Earth)
output_earth = problem.solve()

# Plot solution and simulation (Earth)
states.plot(show=True)
theta.plot(show=True)

# Solve problem (Mars)
# (Modify parameters and re-solve the problem without re-compiling the generated code)
output_mars = problem.solve(modified_parameters={'g' : 3.71})

# Plot solution and simulation (Mars)
states.plot(show=True)
theta.plot(show=True)
```

Listing 7.1: The SCvxGEN modeling interface in Python, defining the Brachistochrone optimal control example (See §7.4.1). The user can generate C code for the entire successive convexification stack, solve the problem, modify parameters on-the-fly, and access/plot the solution and simulation, all directly via Python. The generated C code forms a standalone codebase that can be used for embedded applications.

7.4 Numerical Examples

In this section, we present numerical results for a wide-range of examples arising in classical optimal control, robotics, and aerospace trajectory optimization, including many real-world examples with logical constraints.

7.4.1 Brachistochrone

We start with the Brachistochrone problem, which is a classical optimal control problem that aims to compute the path that minimizes the time it takes to reach a target position from an initial position, in the presence of gravity. We take the problem formulation and parameters from [72], with the problem definition given in Listing 7.1. For this problem, we only use two nodes along with the FOH control parameterization, since the optimal control profile is known to be a straight line. SCvxGEN finds the globally optimal path traversal time of 1.81 seconds.

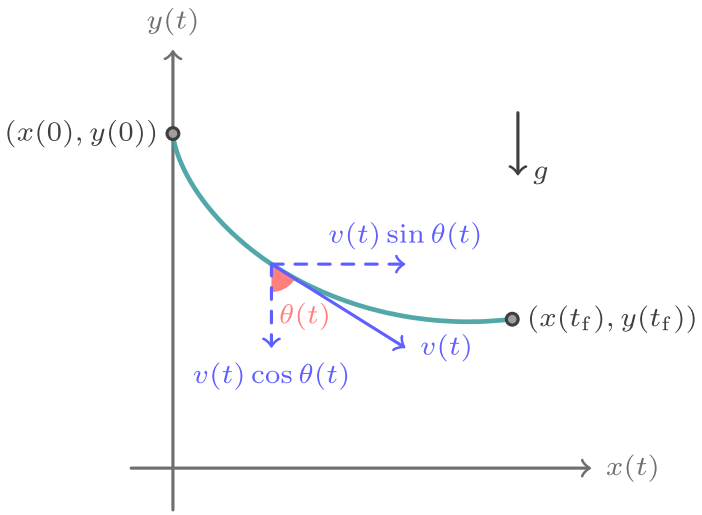

Figure 7.4: The Brachistochrone problem.

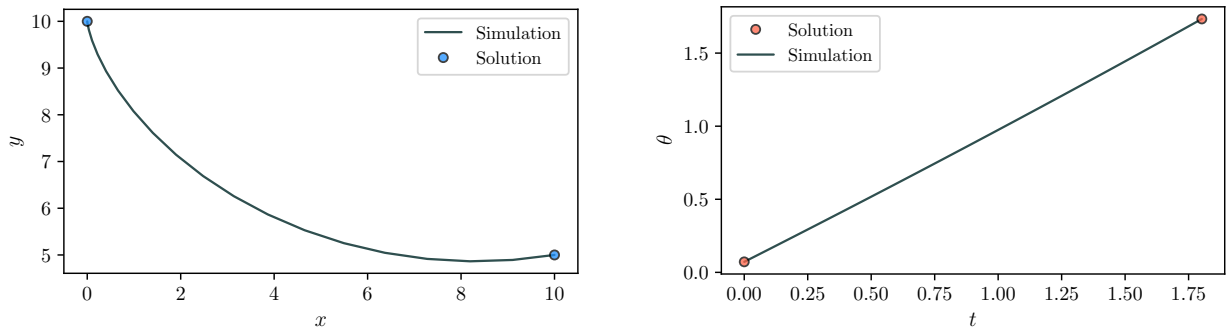

Figure 7.5: The optimal Brachistochrone trajectory (left) and optimal control profile (right). For this problem, SCvxGEN **found the globally optimal solution in under** 70 **microseconds on a MacBook Pro (M2 Pro chip; 16 GB of RAM).**

7.4.2 Static & Dynamic Obstacle Avoidance

The static and dynamic obstacle avoidance examples are from [66], where SCvxGEN was used to solve the problem. The corresponding trajectories are shown in Figures 7.6 and 7.7, respectively. This example clearly demonstrates the benefit of the integral constraint reformulation—even with a sparse discretization grid, the trajectory (in black) does not clip any of the obstacles. The same goes for the control input profile—an interesting characteristic of the solution is that the control lower bound constraint is active *in between* nodes.

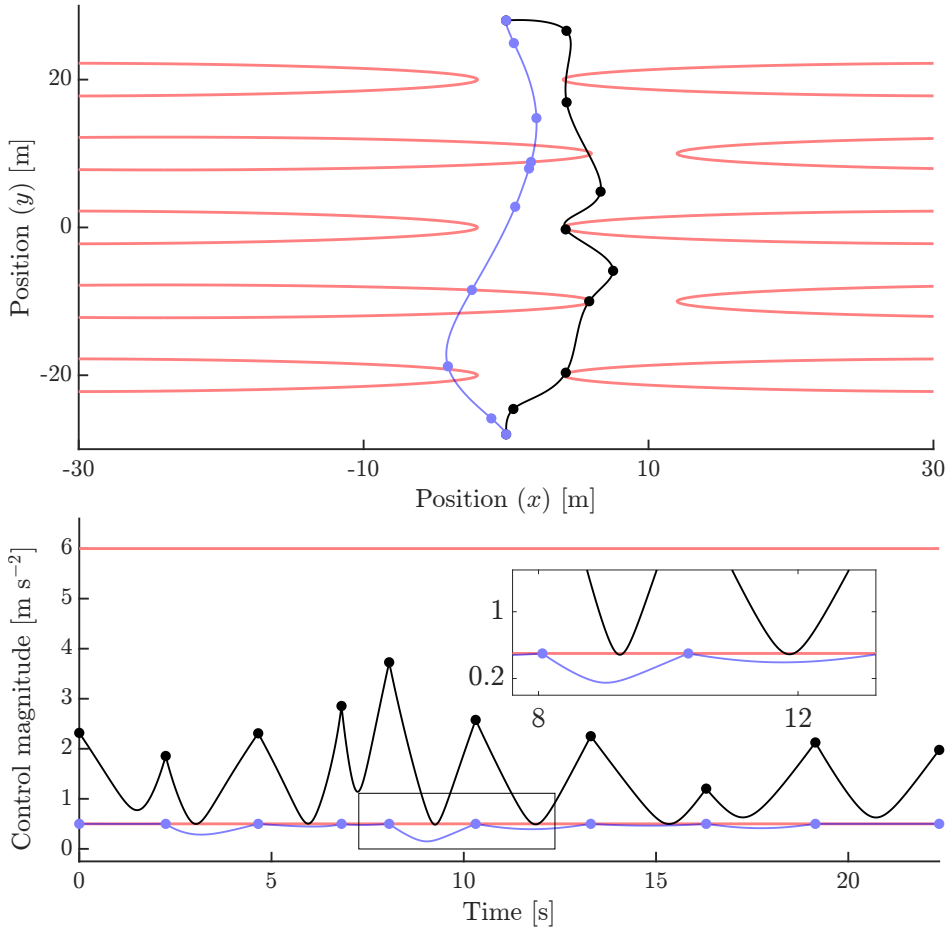

Figure 7.6: An obstacle avoidance trajectory obtained using SCvxGEN **(in black) compared against a node-only trajectory optimization method that exhibits intersample obstacle clipping (in blue). The problem was solved in** 59.4 **milliseconds (averaged over** 1000 **solves).**

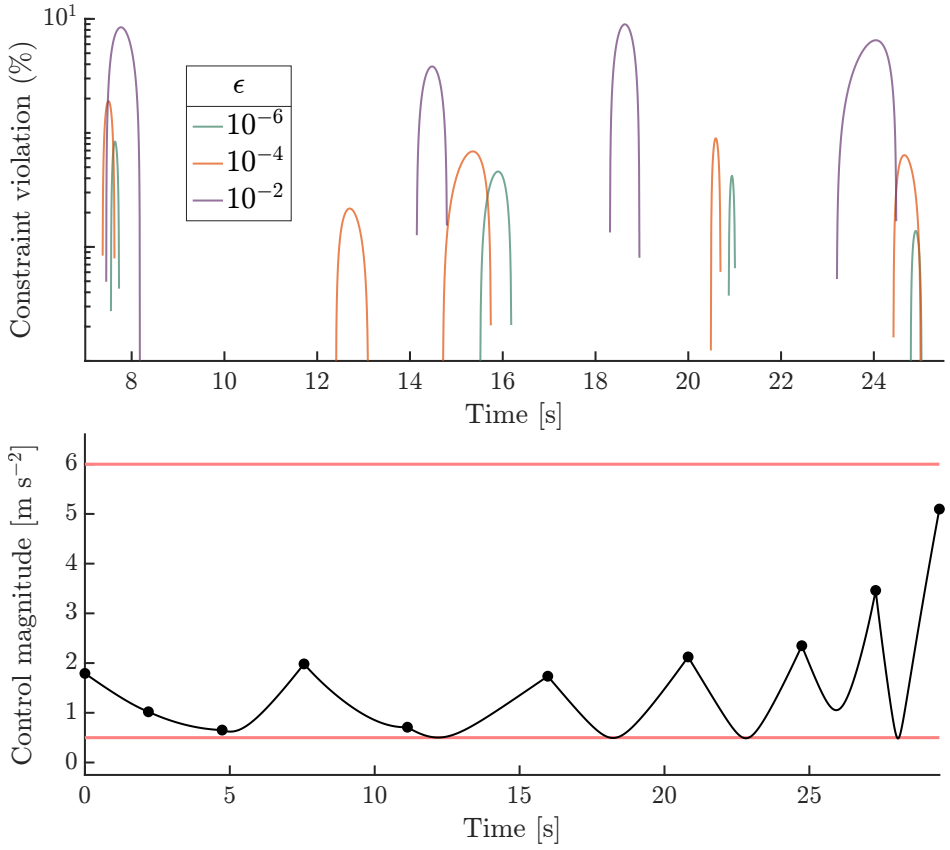

Figure 7.7: A dynamic obstacle avoidance motion planning solution obtained using SCvxGEN **(in black).**

7.4.3 3-DoF Rocket Landing (Convex)

We solve the 3-DoF rocket landing problem with lossless convexification [2]. SCvxGEN automatically detects convexity in the problem, and switches to a pure convex solve mode. The generated solver is called within a one-dimensional time-of-flight search algorithm via the Python interface. The resulting control input profile, with lossless convexification holding, is shown in Figure 7.8.

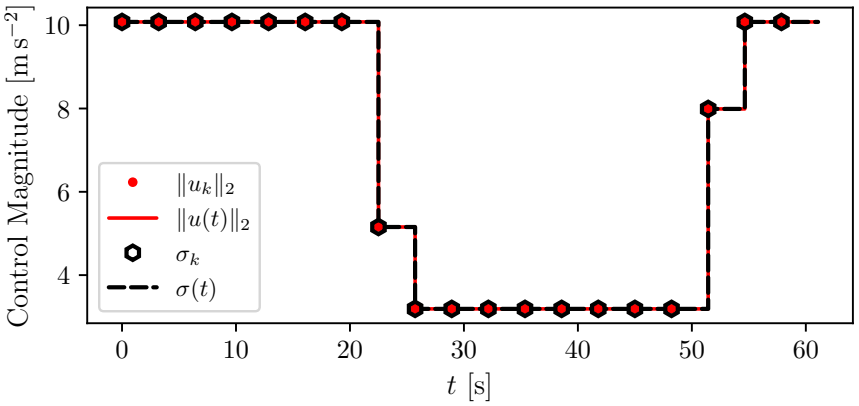

Figure 7.8: Lossless convexification holds for the 3-DoF rocket landing problem.

7.4.4 6-DoF Rocket Landing

The 6-DoF rocket landing example is from [66], where SCvxGEN was used to solve the problem. The solutions are shown in Figure 7.10. Similar to the obstacle avoidance example (§7.4.2), the SCvxGEN solution demonstrates continuous-time constraint satisfaction.

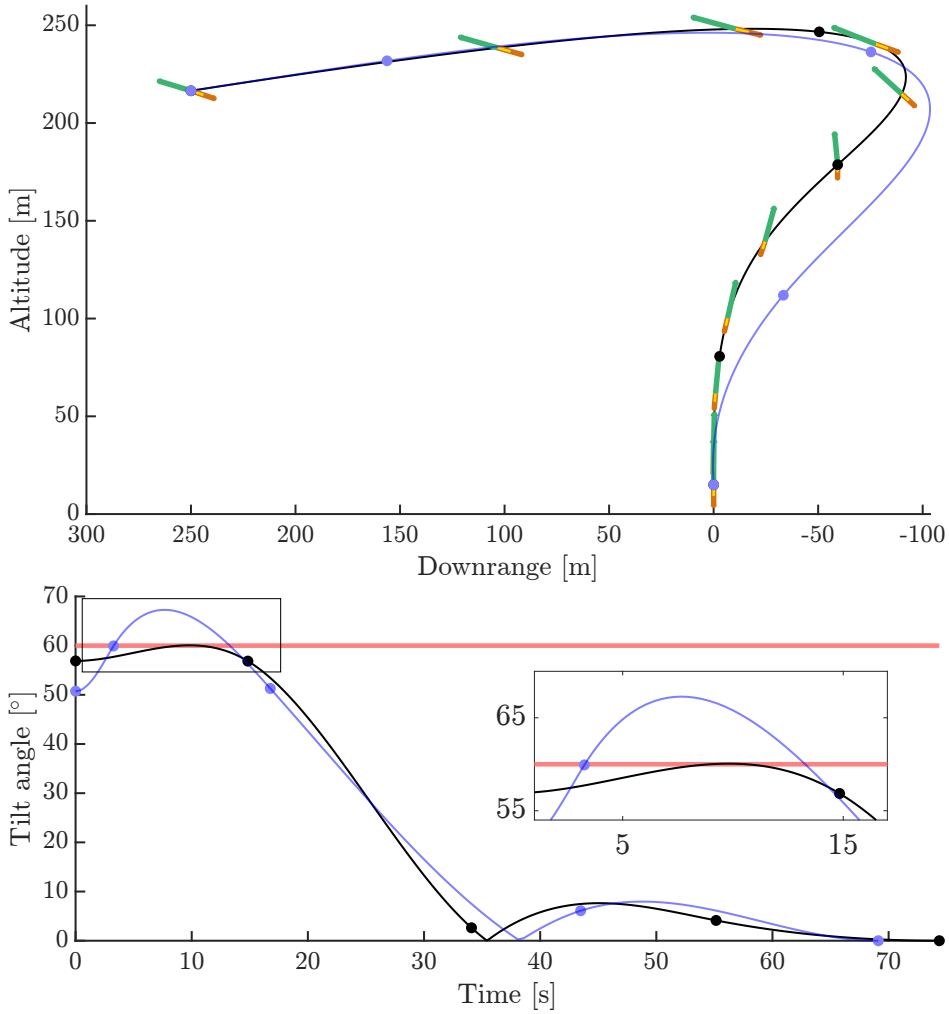

Figure 7.9: The 6-DoF rocket landing position and tilt trajectories. The tilt trajectory from the SCvxGEN solution (in black) satisfies the maximum tilt constraint in continuous-time. The guidance problem was solved in 42.5 milliseconds (averaged over 1000 solves).

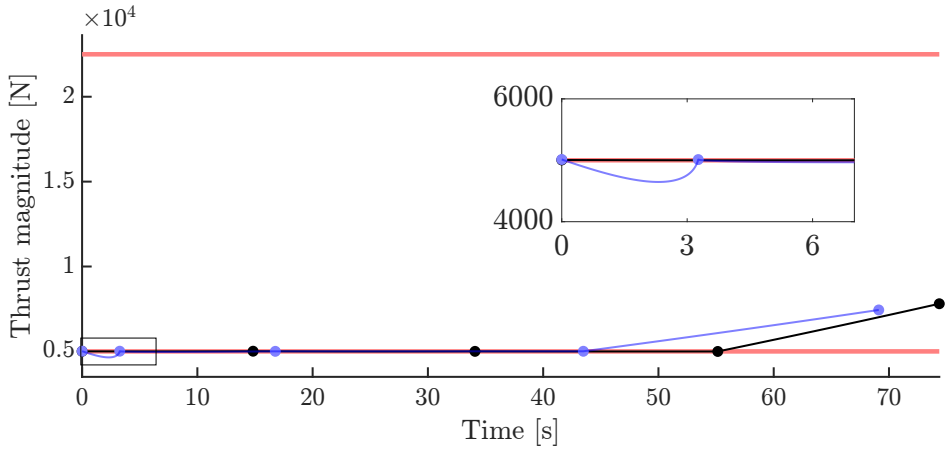

Figure 7.10: The 6-DoF rocket landing guidance control input profile obtained using SCvxGEN **(in black) compared against a node-only trajectory optimization method that exhibits intersample constraint violation (in blue).**

7.4.5 Spacecraft Rendezvous with Impulsive Controls

We consider the spacecraft rendezvous guidance problem from [49] with impulse controls, and solve the minimum-time version of the problem. Figure 7.11 shows the position trajectory of the spacecraft, and Figure 7.12 shows the speed profile of the spacecraft, both of which demonstrate continuous-time constraint satisfaction. The problem definition in SCvxGEN is given in Listing 7.2, to demonstrate the imposition of constraints.

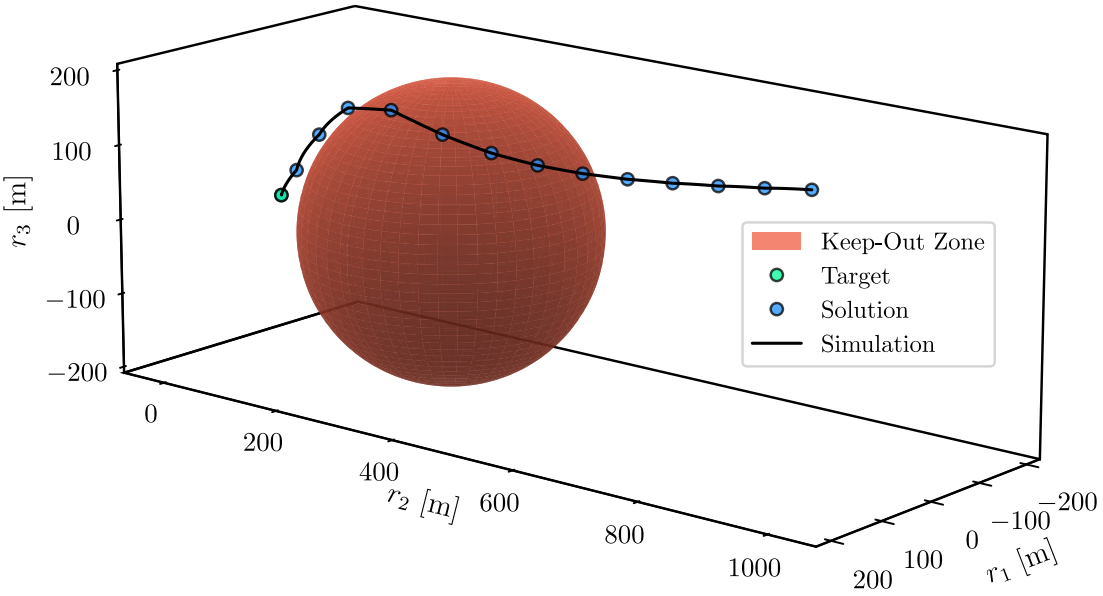

Figure 7.11: Spacecraft rendezvous with a keep-out zone constraint, satisfied in continuous-time.

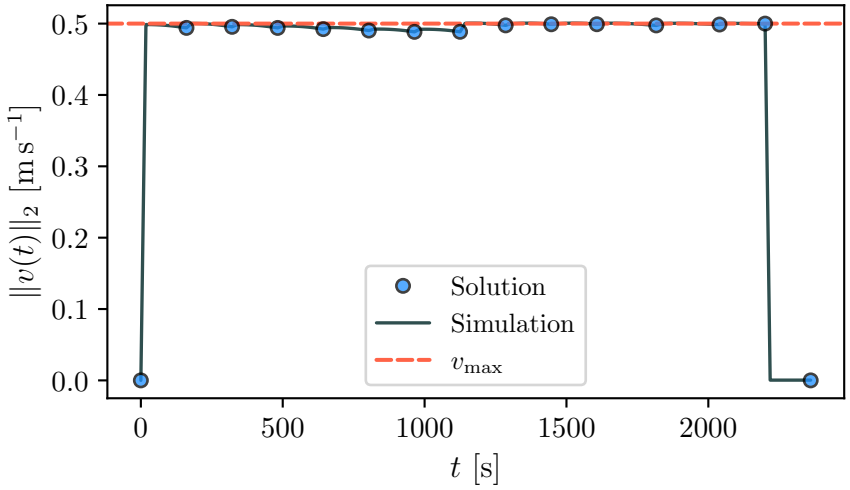

Figure 7.12: The maximum speed constraint in the spacecraft rendezvous problem is satisfied in continuous-time.

```
import scvxgen as scvx

# Parameters
parameters = scvx.Parameters('n rc1:4 rho_c v_max u_max')
n, rc1, rc2, rc3, rho_c, v_max, u_max = parameters
rc = scvx.Vector(rc1, rc2, rc3)

# States
states = scvx.States('r1:4 v1:4')
r1, r2, r3, v1, v2, v3 = states
r = scvx.Vector(r1, r2, r3)
v = scvx.Vector(v1, v2, v3)

# Controls
controls = scvx.Controls('u1:4')
u1, u2, u3 = controls
u = scvx.Vector(u1, u2, u3)

# Dynamics
dynamics = scvx.Dynamics(
    v,
    3 * n**2 * r1 + 2 * n * v2 + u1,
    -2 * n * v1 + u2,
    -n**2 * r3 + u3
)

# Inequality constraints (continuous-time)
inequality_constraints = scvx.InequalityConstraints(
    rho_c**2 - (r - rc).norm()**2, # squared to ensure differentiability
    v.norm()**2 - v_max**2         # squared to ensure differentiability
)

# Node constraints (discrete-time)
node_constraints = scvx.NodeConstraints(
    scvx.NormCone(Au_cone=scvx.sympy.matrices.eye(3), d_cone=u_max), # required (CT constraints will not capture impulsive
      controls)
    scvx.AutoInequality(rho_c - (r - rc).norm()), # optional
    scvx.NormCone(Ax_cone=scvx.sympy.diag(0, 0, 0, 1, 1, 1), d_cone=v_max) # optional
)


# Numerical values
time = scvx.Time(guess=3000, t_max_factor=1.25, t_min_factor=0.75) # temporal attributes
parameters.set_value(0.00113, 0, 300, 0, 200, 0.5, 0.1)            # system/constraint parameters
states.set_initial(150, 1000, 200, 0, 0, 0)                        # initial condition
states.set_final(0, 0, 0, 0, 0, 0)                                 # final condition
states.set_scale(150, 1000, 200, 0.5, 0.5, 0.5)                    # scaling factors (optional)
controls.set_initial(0.0, 0.0, 0.0)                                # initial-guess start point
controls.set_final(0.0, 0.0, 0.0)                                  # initial-guess end point
controls.set_minimum(-u_max.value, -u_max.value, -u_max.value)     # lower bounds
controls.set_maximum(u_max.value, u_max.value, u_max.value)        # upper bounds

# Settings
settings = scvx.Settings(
    N_disc=15,
    disc_order=-1 # impulse-train (IMP)-based discretization
)

# Set up problem
problem = scvx.Problem(
    time=time,
    parameters=parameters,
    states=states,
    controls=controls,
    dynamics=dynamics,
    inequality_constraints=inequality_constraints,
    node_constraints=node_constraints,
    settings=settings
)

# Generate code
problem.generate_code()

# Solve problem
output = problem.solve() # code automatically compiles the first time if not compiled previously

# Plot results
states.plot(show=True, save=True)
controls.plot(show=True, save=True)
```

Listing 7.2: Definition of the spacecraft rendezvous guidance problem (see §7.4.5) in SCvxGEN. This listing showcases the description of continuous-time and discrete-time constraints within the SCvxGEN modeling interface.

Chapter 8
CONCLUSIONS

This chapter provides chapter-wise conclusions to the research directions pursued in this dissertation, including potential avenues for future research.

8.1 Summary of Contributions & Future Work

8.1.1 **Chapter 2**: *Sequential Conic Optimization for Trajectory Generation*

Chapter 2 provides a comprehensive trajectory optimization framework called sequential conic optimization (SeCO). For nonconvex optimal control problems that can be modeled to fit the template conic subproblem structure, it is shown that this framework can be used to obtain high-performance guidance algorithms (as detailed in Chapter 6).

An avenue for future work is to extend the sequential optimization framework to natively handle a class of nonconvex path constraints, i.e., to handle them directly without linearization in a manner that is *exact* at every solver iteration. Consider the subproblem template given by Equations 2.17, with set $\mathbb{D}$ replaced by $\mathbb{D}_{\text{ncvx}}$ (a nonconvex set):

$$\underset{z}{\text{minimize}} \quad \frac{1}{2} z^\top Q z + \langle q, z \rangle \tag{8.1a}$$

$$\text{subject to} \quad Hz - h = 0 \tag{8.1b}$$

$$z \in \mathbb{D}_{\text{ncvx}}$$

In Chapter 2, the set $\mathbb{D}$ is assumed to be a closed convex set. In common trajectory optimization examples, the path constraints imposed through this set are temporally separable and possess closed-form projection operations [237, 64, 240]. However, there exist several commonly imposed path constraints in trajectory optimization that are nonconvex and yet

possess closed-form projection operations; for example, the thrust magnitude constraint (in Cartesian coordinates), that is best represented as an annulus [160]. Such a template would lead to a *sequential nonconvex optimization* framework, wherein only the constraints that do not possess closed-form projections (such as the dynamics) are linearized at each iteration.

Analyzing the convergence properties of PIPG with the closed convex set $\mathbb{D}$ (as in Equation 2.17), replaced by a nonconvex set $\mathbb{D}_{\text{ncvx}}$ (as in Equation 8.1), is a potential research direction. Despite this modification, the complexity of the algorithm would remain the same as that of the original PIPG algorithm, and all the numerical methods proposed to improve the performance of PIPG would likely apply as presented.

8.1.2 **Chapter 3**: *Optimal Preconditioning for Conic Optimization*

In Chapter 3, we propose a three-step preconditioning procedure, the hypersphere preconditioner, to improve the performance of conic optimization solvers for online applications that involve solving quadratic cone programs (QCPs) with strongly convex objective functions. This preconditioner is amenable to a mostly analytical, fully factorization-free, and customizable implementation, while significantly improving the performance of first-order conic optimization solvers, making it particularly beneficial for online applications that involve solving a sequence of QCPs with dynamically changing problem data. We derive an analytical expression for the optimal objective function scaling factor in the sense of minimizing the condition number of the KKT matrix of the preconditioned problem, and obtain a lower bound on it. We demonstrate the efficacy of the hypersphere preconditioner by means of numerical experiments in both convex optimization and sequential convex programming settings.

Future work involves expanding the class of problems to which the proposed preconditioning procedure can apply. Generalizing the preconditioning procedure to problems with non-strongly convex objective functions would allow for its use within most NMPC and SCP-based trajectory optimization frameworks [135, 134, 146, 51, 49, 219, 109].

8.1.3 **Chapter 4**: *Set-based Dynamic Programming for Predictive Control*

We have developed a unified computational framework for closed-loop optimal, robust, and resilient predictive control for autonomous systems. Motivated by the autonomous precision landing problem, we have proposed a set-based framework for robust control with respect to both state and control uncertainty, instantaneous reachable set computation, and maximal decision-deferral. The framework is based on dynamic programming over controllable tubes, achieving computational tractability using constrained zonotopes, a parameterization of compact convex polytopes that enables efficient high-dimensional set operations.

This work aims at bridging the gap between traditional open-loop guidance approaches and closed-loop control methods for applications such as precision landing.

Avenues for future research include generalizing the class of problems considered, reducing the number of sources of approximation, and implementing the computational framework via high-performance code to assess onboard viability for autonomous systems in terms of memory requirements and solve-times.

8.1.4 **Chapter 5**: *Temporal Techniques for Guidance & Control*

Chapter 5 deals with temporal techniques for trajectory optimization. The main theme of this chapter is to leverage the notion of time to develop more capable optimal control problem formulations without sacrificing performance, i.e., to treat time itself as a degree-of-freedom to achieve certain trajectory generation capabilities within a continuous optimization framework (without having to resort to mixed-integer programming).

To that end, in Chapter 5, an extension to the idea of time-dilation [203] is proposed, which we refer to as *time-interval* dilation, that treats the time intervals between consecutive discrete temporal nodes themselves as decision variables (as opposed to only the final time). This extension allows for the formulation of a class of optimal control problems that can be solved using existing sequential convex programming frameworks, but with the added capability of achieving certain trajectory generation capabilities that were not possible before,

such as accounting for discrete switches in the dynamics and enabling multi-phase trajectory optimization with free-phase-transition-times.

Further, we use time-interval dilation to formulate a specialized class of compound state-triggered constraints (STCs), wherein the trigger function is strictly monotonic in a neighborhood around the trigger. In this special case, we show that the compound STC possesses the following properties (which are not enjoyed by the original STC formulation): (i) triggering is exact in time, i.e., it is not subject to intersample violations, and (ii) the entire compound STC is convex if the trigger function and the constraint conditions are individually convex.

Finally, we have proposed a set-based approach to closed-loop free-final-time optimal control.

8.1.5 **Chapter 6**: *Rocket Landing Guidance & Control*

In Chapter 6, we present two guidance algorithms for rocket landing using the SeCO framework: one based on a terrestrial multi-phase landing scenario for a Starship-like vehicle [193] and the other, using a dual quaternion-based description of the 6-DoF equations of motion [117, 118, 119, 177, 176], applied to a lunar landing scenario. Further, we present a set-based autonomous precision landing case-study.

In §6.1, we formulate and solve a multi-phase, free-transition-time rocket landing guidance problem posed as a single unified optimal control problem using a framework that handles discrete temporal events and state-triggered constraints in a continuous optimization setting rather than mixed-integer programming, by means of *time-interval* dilation, which allows for phase-transition-times to be treated an decision variables. Further, we propose a specialized formulation for compound state-triggered constraints (STCs), which we refer to as *single-crossing* compound STCs. We solve the resulting optimal control problem using the SeCO framework, and observe that PIPG performs 2.7 times faster than ECOS, a state-of-the-art convex optimization solver.

In §6.2, we formulate the nonconvex DQG problem—with mission-critical constraints—in compliance with the SeCO framework, and solve it using $\text{PIPG}_{\text{custom}}$, a custom first-order conic

optimization solver developed for this application. We observe that this solver is able to solve the entire nonconvex problem in a matter of milliseconds, and is much faster than other state-of-the-art convex optimization solvers across varying problem sizes, and under one second onboard the SPLICE flight computer, thus meeting NASA's real-time requirement for powered-descent guidance.

In §6.3, the set-based control architecture proposed in Chapter 4 has been validated through a comprehensive autonomous precision landing case study.

One avenue for future work is to explore bilevel convex optimization techniques for 6-DoF powered-descent guidance. This approach would likely involve solving two purely convex problems in succession, potentially with an intermediate *lifting* step. A similar idea was presented in [42] to generate an initial guess trajectory for sequential convex programming, which, however, was not dynamically feasible. We propose to extend this idea to generate *dynamically feasible* 6-DoF powered-descent guidance trajectories. Such an approach would be able to *guarantee feasibility*—a feature that SCP algorithms do not yet possess—albeit in a restricted setting.

Another avenue for future work is extending the proposed set-based control methods to handle the (nonconvex) 6-DoF rocket landing guidance problem. At present, generalization of these techniques to nonconvex optimal control problems—beyond the problem class considered in [225]—remains an open challenge.

8.1.6 **Chapter 7**: SCvxGEN*: Successive Convexification with Automatic Code Generation*

In Chapter 7, we present a software tool to automatically generate code generation for trajectory optimization, SCvxGEN. The tool is capable of automatically generating embeddable C code for full-stack SCP for trajectory optimization.

The underlying framework, continuous-time successive convexification (CT-SCvx) [65], is explained in detail, along with its ability to generate trajectories with continuous-time constraint satisfaction. Finally, the capabilities of the software are demonstrated by means of relevant optimal control problems motivated by real-world applications.

BIBLIOGRAPHY

[1] Behçet Açıkmeşe, M Aung, Jordi Casoliva, Swati Mohan, Andrew Johnson, Daniel Scharf, David Masten, Joel Scotkin, Aron Wolf, and Martin W Regehr. Flight testing of trajectories computed by G-FOLD: Fuel optimal large divert guidance algorithm for planetary landing. In *AAS/AIAA spaceflight mechanics meeting*, pages 863–870, 2013.

[2] Behçet Açıkmeşe and Scott R Ploen. Convex programming approach to powered descent guidance for Mars landing. *Journal of Guidance, Control, and Dynamics*, 30(5):1353–1366, September 2007.

[3] Behçet Açıkmeşe and Lars Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 47(2):341–347, 2011.

[4] Behçet Açıkmeşe, John M Carson III, and David S Bayard. A robust model predictive control algorithm for incrementally conic uncertain/nonlinear systems. *International Journal of Robust and Nonlinear Control*, 21(5):563–590, 2011.

[5] 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. *IEEE Transactions on Control Systems Technology*, 21(6):2104–2113, 2013.

[6] Behçet Açıkmeşe, J Casoliva, John M Carson III, and L Blackmore. G-fold: A real-time implementable fuel optimal large divert guidance algorithm for planetary pinpoint landing. *Concepts and Approaches for Mars Exploration*, 1679:4193, 2012.

[7] Behçet Açıkmeşe and Scott Ploen. A powered descent guidance algorithm for mars pinpoint landing. In *AIAA guidance, navigation, and control conference and exhibit*,

page 6288, 2005.

[8] Behçet Açıkmeşe, Daniel Scharf, Lars Blackmore, and Aron Wolf. Enhancements on the convex programming based powered descent guidance algorithm for Mars landing. In *AIAA/AAS astrodynamics specialist conference and exhibit*, page 6426, 2008.

[9] Behçet Açıkmeşe, Steven W Sell, A Miguel San Martin, and Jeffrey J Biesiadecki. Mars science laboratory flyaway guidance, navigation, and control system design. *Journal of Spacecraft and Rockets*, 51(4):1227–1236, 2014.

[10] Yunus M Agamawi and Anil V Rao. Cgpops: A c++ software for solving multiple-phase optimal control problems using adaptive gaussian quadrature collocation and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 46(3):1–38, 2020.

[11] Panos J Antsaklis and Anthony N Michel. *Linear systems*. Basel, Switzerland: Birkhauser, 2006.

[12] David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy. Pdlp: A practical first-order method for large-scale linear programming. *arXiv preprint arXiv:2501.07018*, 2025.

[13] Nancy Atkinson. SpaceX tests its Starship-catching launch tower. www.universetoday.com/spacex-tests-its-starship-catching-launch-tower, 2022. Accessed: 11.18.2022.

[14] Goran Banjac, Bartolomeo Stellato, Nicholas Moehle, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Embedded code generation using the osqp solver. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1906–1911. IEEE, 2017.

[15] Heinz H. Bauschke, Minh N. Bui, and Xianfu Wang. Projecting onto the intersection of a cone and a sphere. *SIAM Journal on Optimization*, 2018.

[16] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer International, 2017.

[17] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[18] Amir Beck. *First-Order Methods In Optimization*. MOS-SIAM Series on Optimization. Society for Industrial & Applied Mathematics, 2019.

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

[20] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[21] Aharon Ben-Tal and Arkadi Nemirovski. On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26(2):193–205, 2001.

[22] Michele Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 2002.

[23] Leonard David Berkovitz. *Optimal control theory*, volume 12. Springer Science & Business Media, 2013.

[24] Danil Berrah, Alexandre Chapoutot, and Pierre-Loïc Garoche. Scvxpygen: Autocoding scvx algorithm. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 5086–5093. IEEE, 2024.

[25] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

[26] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[27] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[28] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Advances in Design and Control. SIAM, 2nd edition, January 2010.

[29] Neal S Bhasin, WR Whittaker, and CG Atkeson. Fuel-optimal spacecraft guidance for landing in planetary pits. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, 2016.

[30] Patrick Billingsley. *Probability and Measure*. Wiley Series in Probability & Mathematical Statistics: Probability & Mathematical Statistics. John Wiley & Sons, Nashville, TN, May 1995.

[31] L Susan Blackford, Antoine Petitet, Roldan Pozo, Karin Remington, R Clint Whaley, James Demmel, Jack Dongarra, Iain Duff, Sven Hammarling, Greg Henry, et al. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.

[32] Lars Blackmore. Autonomous precision landing of space rockets. In *in Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*, volume 46, pages 15–20, 2016.

[33] Lars Blackmore, Behçet Açıkmeşe, and John M Carson III. Lossless convexification of control constraints for a class of nonlinear optimal control problems. *Systems & Control Letters*, 61(8):863–870, 2012.

[34] Franco Blanchini and Stefano Miani. *Set-theoretic methods in control*. Springer, 2015.

[35] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[36] Riccardo Bonalli, Abhishek Cauligi, Andrew Bylard, and Marco Pavone. Gusto: Guaranteed sequential trajectory optimization via sequential convex programming. In *2019 International conference on robotics and automation (ICRA)*, pages 6741–6747. IEEE, 2019.

[37] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[38] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, January 1994.

[39] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[40] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[41] Bobbi Jo Broxson. The Kronecker product. *Master's thesis, University of North Florida*, 2006.

[42] Samuel C Buckner, Joshua Shaffer, John M Carson III, Breanna J Johnson, Ronald R Sostaric, and Behçet Açıkmeşe. Constrained visibility guidance for 6-DOF powered descent maneuvers with terrain scanning using sequential convex programming. In *AIAA SciTech 2024 Forum*, page 1759, 2024.

[43] Francesco Capolupo and Antonio Rinalducci. Descent & landing trajectory and guidance algorithms with divert capabilities for moon landing. In *AIAA SciTech 2024 Forum*, page 0086, 2024.

[44] John M Carson III, Michelle M Munk, Ronald R Sostaric, Jay N Estes, Farzin Amzajerdian, James B Blair, David K Rutishauser, Carolina I Restrepo, Alicia M Dwyer-Cianciolo, George Chen, et al. The splice project: Continuing nasa development of gn&c technologies for safe and precise landing. In *AIAA SciTech 2019 Forum*, page 0660, 2019.

[45] Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. On the evaluation com-

plexity of composite function minimization with applications to nonconvex nonlinear programming. *SIAM Journal of Optimization*, 21(4):1721–1739, October 2011.

[46] Jordi Casoliva, Gurkirpal Singh, Paul Brugarolas, and David W Way. Reconstructed flight performance of the powered descent guidance and control system for the Mars 2020 perseverance mission. *AAS*, 2021.

[47] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, December 2010.

[48] Luiz FO Chamon, Alexandre Amice, Santiago Paternain, and Alejandro Ribeiro. Resilient control: Compromising to adapt. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5703–5710. IEEE, 2020.

[49] Govind M Chari and Behçet Açıkmeşe. Spacecraft rendezvous guidance via factorization-free sequential convex programming using a first-order method. *arXiv preprint arXiv:2402.04561*, 2024.

[50] Govind M Chari and Behçet Açikmeşe. Qoco: A quadratic objective conic optimizer with custom solver generation. *arXiv preprint arXiv:2503.12658*, 2025.

[51] Govind M Chari, Abhinav G Kamath, Purnanand Elango, and Behçet Açıkmeşe. Fast Monte Carlo analysis for 6-DoF powered-descent guidance via GPU-accelerated sequential convex programming. In *AIAA SciTech*, 2024.

[52] Govind M Chari, Yue Yu, and Behçet Açıkmeşe. Constraint preconditioning and parameter selection for a first-order primal-dual method applied to model predictive control. In *IEEE Conference on Decision and Control*, 2024.

[53] Greg Chavers, N Suzuki, M Smith, Lisa Watson-Morgan, Steven W Clarke, Walter C Engelund, L Aitchison, S McEniry, L Means, M DeKlotz, et al. NASA's human lunar landing strategy. In *70th International Astronautical Congress*, pages 1–6, 2019.

[54] Greg Chavers, Lisa Watson-Morgan, Marshall Smith, Nantel Suzuki, and Tara Polsgrove. Nasa's human landing system: The strategy for the 2024 mission and future sustainability. In *2020 IEEE Aerospace Conference*, pages 1–9, 2020.

[55] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Lancelot: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[56] Austin DeSisto. Starship and its belly flop maneuver. https://everydayastronaut.com/starships-belly-flop-maneuver/, 2021. Accessed: October 14, 2022.

[57] G. Di Pillo and L. Grippo. Exact penalty functions in constrained optimization. *SIAM Journal on Control and Optimization*, 27(6):1333–1360, November 1989.

[58] Javier A Doll, Abhinav G Kamath, Kyle W Smith, Jeanette M Harper, Isaac Rowe, Behçet Açıkmeşe, Samuel M Pedrotty, and Gavin F Mendeck. Hardware in the loop performance of terrestrial powered descent dual quaternion guidance with a custom first-order solver. In *AIAA SciTech*, 2025.

[59] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

[60] Dmitriy Drusvyatskiy and Adrian Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Mathematics of Operations Research*, 43(3):919–948, 2018.

[61] Dmitriy Drusvyatskiy and Courtney Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, 178:503–558, 2019.

[62] Daniel Dueri, Behçet Açıkmeşe, Daniel P Scharf, and Matthew W Harris. Customized real-time interior-point methods for onboard powered-descent guidance. *Journal of Guidance, Control, and Dynamics*, 40(2):197–212, 2017.

[63] Daniel Dueri, Jing Zhang, and Behçet Açıkmeşe. Automated custom code generation for embedded, real-time second order cone programming. *IFAC Proceedings Volumes*, 47(3):1605–1612, 2014.

[64] Purnanand Elango, Abhinav G Kamath, Yue Yu, John M Carson III, Mehran Mesbahi, and Behçet Açıkmeşe. A customized first-order solver for real-time powered-descent guidance. In *AIAA SciTech*, 2022.

[65] Purnanand Elango, Dayou Luo, Abhinav G Kamath, Samet Uzun, Taewan Kim, and Behçet Açıkmeşe. Successive convexification for trajectory optimization with continuous-time constraint satisfaction. *arXiv preprint arXiv:2404.16826*, 2024.

[66] Purnanand Elango, Dayou Luo, Abhinav G Kamath, Samet Uzun, Taewan Kim, and Behçet Açıkmeşe. Continuous-time successive convexification for constrained trajectory optimization. *Automatica*, 180:112464, 2025.

[67] Purnanand Elango, Selahattin Burak Sarsilmaz, and Behçet Açıkmeşe. Deferring decision in multi-target trajectory optimization. In *AIAA SciTech 2022 Forum*, page 1583, 2022.

[68] Purnanand Elango, Selahattin Burak Sarsilmaz, and Behçet Açıkmeşe. Deferred-decision trajectory optimization. *arXiv preprint arXiv:2502.06623*, 2025.

[69] Purnanand Elango, Abraham P Vinod, Stefano Di Cairano, and Avishai Weiss. Continuous-time successive convexification for passively-safe six-degree-of-freedom powered-descent guidance. In *AIAA SciTech 2025 Forum*, page 1894, 2025.

[70] Purnanand Elango, Abraham P Vinod, Kenji Kitamura, Behçet Açıkmeşe, Stefano Di Cairano, and Avishai Weiss. Successive convexification for passively-safe spacecraft rendezvous on near rectilinear halo orbit. *arXiv preprint arXiv:2505.17251*, 2025.

[71] Utku Eren, Daniel Dueri, and Behçet Açıkmeşe. Constrained reachability and controllability sets for planetary precision landing via convex optimization. *Journal of*

Guidance, Control, and Dynamics, 38(11):2067–2083, 2015.

[72] Robert Falck, Justin S. Gray, Kaushik Ponnapalli, and Ted Wright. dymos: A python package for optimal control of multidisciplinary systems. *Journal of Open Source Software*, 6(59):2809, 2021.

[73] Federal Aviation Administration. Advisory circular 20-193: Use of multi-core processors in aircraft systems, August 2024.

[74] Gianluca Frison and Moritz Diehl. Hpipm: a high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine*, 53(2):6563–6569, 2020.

[75] Matthew Fritz, Javier Doll, Kari C Ward, Gavin Mendeck, Ronald R Sostaric, Sam Pedrotty, Chris Kuhl, Behçet Açıkmeşe, Stefan R Bieniawski, Lloyd Strohl, et al. Post-flight performance analysis of navigation and advanced guidance algorithms on a terrestrial suborbital rocket flight. In *AIAA SciTech 2022 Forum*, page 0765, 2022.

[76] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[77] José V Garrido and Marco Sagliano. Ascent and descent guidance of multistage rockets via pseudospectral methods. In *AIAA SciTech 2021 Forum*, page 0859, 2021.

[78] Jared Lee Gearhart, Kristin Lynn Adair, Justin David Durfee, Katherine A Jones, Nathaniel Martin, and Richard Joseph Detry. Comparison of open-source linear programming solvers, 2013.

[79] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[80] Pontus Giselsson and Stephen Boyd. Diagonal scaling in Douglas-Rachford splitting and ADMM. In *IEEE Conference on Decision and Control*, 2014.

[81] Pontus Giselsson and Stephen Boyd. Preconditioning in fast dual gradient methods.

In *IEEE Conference on Decision and Control*, 2014.

[82] Pontus Giselsson and Stephen Boyd. Metric selection in fast dual forward–backward splitting. *Automatica*, 62:1–10, December 2015.

[83] Joseph D Gleason, Abraham P Vinod, and Meeko MK Oishi. Lagrangian approximations for stochastic reachability of a target tube. *Automatica*, 128:109546, 2021.

[84] Paul J Goulart and Yuwen Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives. *arXiv preprint arXiv:2405.12762*, 2024.

[85] Sébastien Gros and Moritz Diehl. Numerical optimal control (draft), 2020.

[86] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021.

[87] S. P. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22(3):297–309, July 1977.

[88] Matthew W Harris and Behçet Açıkmeşe. Maximum divert for planetary landing using convex optimization. *Journal of Optimization Theory and Applications*, 162:975–995, 2014.

[89] Richard F Hartl, Suresh P Sethi, and Raymond G Vickson. A survey of the maximum principles for optimal control problems with state constraints. *SIAM review*, 37(2):181–218, 1995.

[90] Christopher R Hayner, Samuel C Buckner, Daniel Broyles, Evelyn Madewell, Karen Leung, and Behçet Açikmeşe. Halo: Hazard-aware landing optimization for autonomous systems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3261–3267. IEEE, 2023.

[91] Magnus R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, November 1969.

[92] Gerard J Holzmann. The power of 10: Rules for developing safety-critical code. *Com-*

puter, 39(6):95–99, 2006.

[93] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge University Press, 2012.

[94] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal control applications and methods*, 32(3):298–312, 2011.

[95] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.

[96] Taylor A Howell, Kevin Tracy, Simon Le Cleac'h, and Zachary Manchester. Calipso: A differentiable solver for trajectory optimization with conic and complementarity constraints. In *The International Symposium of Robotics Research*, pages 504–521. Springer, 2022.

[97] Qi Huangfu and JA Julian Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.

[98] Junsub Hwang and Jaemyung Ahn. Integrated optimal guidance for reentry and landing of a rocket using multi-phase pseudo-spectral convex optimization. *International Journal of Aeronautical and Space Sciences*, pages 1–9, 2022.

[99] John M Ingram and M.M Marsh. Projections onto convex cones in Hilbert space. *Journal of Approximation Theory*, 64:343–350, 1991.

[100] Abhinav G Kamath, Javier A Doll, Purnanand Elango, Taewan Kim, Skye Mceowen, Yue Yu, Taylor P Reynolds, Gavin F Mendeck, John M Carson III, Mehran Mesbahi, et al. Onboard dual quaternion guidance for rocket landing. *arXiv preprint arXiv:2508.10439*, 2025.

[101] Abhinav G Kamath, Purnanand Elango, and Behçet Açıkmeşe. Optimal preconditioning for online quadratic cone programming. *IEEE Control Systems Letters*, 9(10.1109/LCSYS.2025.3563219), 2025.

[102] Abhinav G Kamath, Purnanand Elango, Skye Mceowen, Yue Yu, John M Carson III, Mehran Mesbahi, and Behçet Açıkmeşe. Customized real-time first-order methods for onboard dual quaternion-based 6-DoF powered-descent guidance. In *AIAA SciTech*, 2023.

[103] Abhinav G Kamath, Purnanand Elango, Yue Yu, Skye Mceowen, Govind M Chari, John M Carson III, and Behçet Açıkmeşe. Real-time sequential conic optimization for multi-phase rocket landing guidance. *IFAC-PapersOnLine*, 2023.

[104] Abhinav G Kamath, Samet Uzun, Govind M Chari, Purnanand Elango, Michael Szmuk, and Behçet Açıkmeşe. SCvxGEN: Successive convexification with automatic custom code generation for fast and embedded general-purpose trajectory optimization. `http://scvxgen.com`, 2026.

[105] Abhinav G Kamath, Abraham P Vinod, Purnanand Elango, Stefano Di Cairano, and Avishai Weiss. Robust optimal control for autonomous precision landing via set-based dynamic programming. In *AIAA SciTech 2026 Forum*, 2026.

[106] Abhinav G Kamath, Abraham P Vinod, Purnanand Elango, Stefano Di Cairano, and Avishai Weiss. Set-based optimal, robust, and resilient control with applications to autonomous precision landing. *arXiv preprint arXiv:2512.07043*, 2025.

[107] MFR Keuning. Software partitioning for safety-critical airborne systems in practice, 2000.

[108] Taewan Kim, Purnanand Elango, Danylo Malyuta, and Behçet Açıkmeşe. Guided policy search using sequential convex programming for initialization of trajectory optimization algorithms. In *2022 American Control Conference (ACC)*, pages 3572–3578.

IEEE, 2022.

[109] Taewan Kim, Abhinav G Kamath, Niyousha Rahimi, Jasper Corleis, Behçet Açıkmeşe, and Mehran Mesbahi. Six-degree-of-freedom aircraft landing trajectory planning with runway alignment. *Journal of Guidance, Control, and Dynamics*, 2025.

[110] Taewan Kim, Niyousha Rahimi, Abhinav G Kamath, Behçet Açıkmeşe, and Mehran Mesbahi. Generation of approach and landing trajectories with operational constraints for aircraft with multiple degrees of freedom, November 13 2025. US Patent App. 18/662,831.

[111] Allan R Klumpp. Apollo lunar descent guidance. *Automatica*, 10(2):133–146, 1974.

[112] Mykel J Kochenderfer and Tim A Wheeler. *Algorithms for optimization*. Mit Press, 2019.

[113] Alex A Kurzhanskiy and Pravin Varaiya. Ellipsoidal toolbox (et). In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1498–1503. IEEE, 2006.

[114] Lev Davidovich Landau and Evgenii Mikhailovich Lifshitz. *Mechanics*. Pergamon, Amsterdam, Netherlands, 1969.

[115] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[116] Sang-Don Lee and Chang-Hun Lee. Multi-phase and dual aero/propulsive rocket landing guidance using successive convex programming. *Proceedings of the Institution of Mechanical Engineers, Journal of Aerospace Engineering*, 2022.

[117] Unsik Lee and Mehran Mesbahi. Dual quaternions, rigid body mechanics, and powered-descent guidance. In *2012 ieee 51st ieee conference on decision and control (cdc)*, pages 3386–3391. IEEE, 2012.

[118] Unsik Lee and Mehran Mesbahi. Optimal power descent guidance with 6-DoF line of sight constraints via unit dual quaternions. In *AIAA Guidance, Navigation, and*

Control Conference, page 0319, 2015.

[119] Unsik Lee and Mehran Mesbahi. Constrained autonomous precision landing via dual quaternions and model predictive control. *Journal of Guidance, Control, and Dynamics*, 40(2):292–308, 2017.

[120] Thomas Lipp and Stephen Boyd. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17:263–287, 2016.

[121] Yana Lishkova, Abraham P Vinod, Stefano Di Cairano, and Avishai Weiss. Divert-feasible lunar landing under navigational uncertainty. In *2024 IEEE Conference on Decision and Control (CDC)*, pages 1–8. IEEE, 2024.

[122] Changliu Liu and Masayoshi Tomizuka. Real time trajectory optimization for nonlinear robotic systems: Relaxation and convexification. *Systems & Control Letters*, 108:56–63, October 2017.

[123] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1-3):193–228, 1998.

[124] Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

[125] Pedro Lourenço, Hugo Costa, João Branco, Pierre-Loïc Garoche, Arash Sadeghzadeh, Jonathan Frey, Gianluca Frison, Anthea Comellini, Massimo Barbero, and Valentin Preda. Verification & validation of optimisation-based control systems: methods and outcomes of vv4rtos. In *ESA GNC-ICATT*, 2023.

[126] Dayou Luo. *Modeling and Algorithms for Nonconvex Trajectory Generation Problems: From Constraint Reformulations and First-Order Proximal Methods to Structure-Exploiting Convex Solvers*. PhD thesis, University of Washington, 2025.

[127] Dayou Luo, Purnanand Elango, and Behçet Açıkmeşe. Remarks on "successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems". *arXiv preprint arXiv:2403.00733*, 2024.

[128] Lin Ma, Kexin Wang, Zhijiang Shao, Zhengyu Song, and Lorenz T Biegler. Direct trajectory optimization framework for vertical takeoff and vertical landing reusable rockets: case study of two-stage rockets. *Engineering Optimization*, 51(4):627–645, 2019.

[129] Danylo Malyuta. *Convex Optimization in a Nonconvex World: Applications for Aerospace Systems*. PhD thesis, University of Washington, 2021.

[130] Danylo Malyuta and Behçet Açıkmeşe. Fast homotopy for spacecraft rendezvous trajectory optimization with discrete logic. *Journal of Guidance, Control, and Dynamics*, 46(7):1262–1279, 2023.

[131] Danylo Malyuta, Behçet Açıkmeşe, and Martin Cacan. Robust model predictive control for linear systems with state and input dependent uncertainties. In *2019 American Control Conference (ACC)*, pages 1145–1151. IEEE, 2019.

[132] Danylo Malyuta, Taylor Reynolds, Michael Szmuk, Behçet Açıkmeşe, and Mehran Mesbahi. Fast trajectory optimization via successive convexification for spacecraft rendezvous with integer constraints. In *AIAA SciTech 2020 Forum*, page 0616, 2020.

[133] Danylo Malyuta, Taylor Reynolds, Michael Szmuk, Mehran Mesbahi, Behçet Açıkmeşe, and John M Carson III. Discretization performance and accuracy analysis for the rocket powered descent guidance problem. In *AIAA SciTech 2019 Forum*, page 0925, 2019.

[134] Danylo Malyuta, Taylor P Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behçet Açıkmeşe. Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently. *IEEE Control Systems*, 42(5):40–113, October 2022.

[135] Danylo Malyuta, Yue Yu, Purnanand Elango, and Behçet Açıkmeşe. Advances in trajectory optimization for space vehicle control. *Annual Reviews in Control*, 2021.

[136] Ashish Rao Mangalore, Gabriel Andreas Fonseca Guerra, Sumedh R Risbud, Philipp Stratmann, and Andreas Wild. Neuromorphic quadratic programming for efficient and scalable model predictive control. *arXiv preprint arXiv:2401.14885*, 2024.

[137] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3636–3641. IEEE, 2016.

[138] The Mathworks, Inc., Natick, Massachusetts. *MATLAB Coder version 5.3*, 2021.

[139] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 9.11.0.1769968 (R2021a)*, 2021.

[140] Jacob Mattingley and Stephen Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13:1–27, 2012.

[141] Jacob Mattingley and Stephen P Boyd. Automatic code generation for real-time convex optimization., 2010.

[142] David Q Mayne, Erric C Kerrigan, EJ Van Wyk, and Paola Falugi. Tube-based robust nonlinear model predictive control. *International journal of robust and nonlinear control*, 21(11):1341–1353, 2011.

[143] Skye Mceowen and Behçet Açıkmeşe. Hypersonic entry trajectory optimization via successive convexification with abstracted control. In *AIAA SciTech 2022 Forum*, page 0950, 2022.

[144] Skye Mceowen, Daniel J Calderone, Aman Tiwary, Jason S Zhou, Taewan Kim, Purnanand Elango, and Behçet Açıkmeşe. Auto-tuned primal-dual successive convexification for hypersonic reentry guidance. In *AIAA SciTech 2025 Forum*, page 1317,

2025.

[145] Skye Mceowen, Daniel J Calderone, Aman Tiwary, Jason SK Zhou, Taewan Kim, Purnanand Elango, and Behçet Açıkmeşe. Autotuned primal–dual successive convexification for reentry guidance. *Journal of Guidance, Control, and Dynamics*, 48(9):2020–2040, 2025.

[146] Skye Mceowen, Abhinav G Kamath, Purnanand Elango, Taewan Kim, Samuel C. Buckner, and Behçet Açıkmeşe. High-Accuracy 3-DoF Hypersonic Reentry Guidance via Sequential Convex Programming. In *AIAA SciTech*, 2023.

[147] Robert McGill. Optimum control, inequality state constraints, and the generalized newton-raphson algorithm. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 3(2):291–298, 1965.

[148] Bernhard Meindl and Matthias Templ. Analysis of commercial and free and open source solvers for linear optimization problems. *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*, 20:64, 2012.

[149] Gavin Mendeck. Space Technology Mission Directorate - Game Changing Development Program - FY24 SPLICE Annual Review Presentation. In *NASA Game Changing Development Annual Program Review*, pages 1–13, San Antonio, TX, August 2024. National Aeronautics and Space Administration. Presentation, Document ID: 20240010956, Accessed via NASA Technical Reports Server (NTRS).

[150] Gavin Mendeck and Wade May. Space Technology Mission Directorate - Game Changing Development Program - FY23 SPLICE Annual Review Presentation. In *NASA Game Changing Development Annual Program Review*, pages 1–17, San Antonio, TX, September 2023. National Aeronautics and Space Administration. Presentation, Document ID: 20230011737, Accessed via NASA Technical Reports Server (NTRS).

[151] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

[152] MOSEK ApS. *The MOSEK Optimization Toolbox for MATLAB Manual. Version 9.0.*, 2019.

[153] Brian K. Muirhead, Austin K. Nicholas, Jeffrey Umland, Orson Sutherland, and Sanjay Vijendran. Mars sample return campaign concept status. *Acta Astronautica*, 176:131–138, November 2020.

[154] Elon Musk. Making humans a multi-planetary species. *New Space*, 5(2):46–61, 2017.

[155] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[156] Brendan O'Donoghue. Operator splitting for a homogeneous embedding of the linear complementarity problem. *SIAM Journal on Optimization*, 31(3):1999–2023, 2021.

[157] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, February 2016.

[158] Rushen B. Patel and Paul J. Goulart. Trajectory generation for aircraft avoidance maneuvers using online optimization. *Journal of Guidance, Control, and Dynamics*, 34(1):218–230, January 2011.

[159] Michael A. Patterson and Anil V. Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software*, 41(1):1:1–1:37, October 2014.

[160] Bas Peters and Felix J Herrmann. Algorithms and software for projections onto intersections of convex and non-convex sets with applications to inverse problems. *arXiv*

preprint arXiv:1902.09699, 2019.

[161] D Petersen, J Charvat, J Somers, J Pattarini, MB Stenger, M Van Baalen, and SMC Lee. Apollo to artemis: Mining 50-year old records to inform future human lunar landing systems. *LSAH Newsletter*, 25(1):6–7, 2020.

[162] Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision*. IEEE, November 2011.

[163] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, New York, NY, 1969.

[164] M. J. D. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14(1):224–248, December 1978.

[165] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[166] Warren B Powell. What you should know about approximate dynamic programming. *Naval Research Logistics (NRL)*, 56(3):239–249, 2009.

[167] WL Quaide and VR Oberbeck. Geology of the apollo landing sites. *Earth-Science Reviews*, 5(4):255–278, 1969.

[168] Rien Quirynen, Milan Vukov, and Moritz Diehl. Multiple shooting in a microsecond. In *Multiple Shooting and Time Domain Decomposition Methods: MuS-TDD, Heidelberg, May 6-8, 2013*, pages 183–201. Springer, 2015.

[169] Vignesh Raghuraman and Justin P Koeln. Set operations and order reductions for constrained zonotopes. *Automatica*, 139:110204, 2022.

[170] Davide Martino Raimondo, Daniel Limon, Mircea Lazar, Lalo Magni, and Ed-

uardo Ferná ndez Camacho. Min-max model predictive control of nonlinear systems: A unifying overview on stability. *European Journal of Control*, 15(1):5–21, 2009.

[171] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the astronautical Sciences*, 135(1):497–528, 2009.

[172] Carolina Restrepo, Ronney Lovelace, Ronald Sostaric, and John M Carson III. NASA SPLICE project: Developing the next generation hazard detection system. Technical report, NASA, 2019.

[173] Carolina I Restrepo, Po-Ting Chen, Ronald R Sostaric, and John M Carson III. Next-generation NASA hazard detection system development. In *AIAA SciTech 2020 Forum*, page 0368, 2020.

[174] Taylor Reynolds, Danylo Malyuta, Mehran Mesbahi, Behçet Açıkmeşe, and John M Carson III. A real-time algorithm for non-convex powered descent guidance. In *AIAA SciTech 2020 Forum*, page 0844, 2020.

[175] Taylor P Reynolds and Mehran Mesbahi. Coupled 6-DOF control for distributed aerospace systems. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 5294–5299. IEEE, 2018.

[176] Taylor P Reynolds, Michael Szmuk, Danylo Malyuta, Mehran Mesbahi, Behçet Açıkmeşe, and John M Carson III. Dual quaternion-based powered descent guidance with state-triggered constraints. *Journal of Guidance, Control, and Dynamics*, 43(9):1584–1599, September 2020.

[177] Taylor P Reynolds, Michael Szmuk, Danylo Malyuta, Mehran Mesbahi, Behçet Açıkmeşe, and John M Carson III. A state-triggered line of sight constraint for 6-DoF powered descent guidance problems. In *AIAA SciTech 2019 Forum*, page 0924, 2019.

[178] Taylor Patrick Reynolds. *Computational Guidance and Control for Aerospace Systems.*

University of Washington, 2020.

[179] Jack Ridderhof and Panagiotis Tsiotras. Uncertainty quantication and control during Mars powered descent and landing using covariance steering. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 0611, 2018.

[180] Jack Ridderhof and Panagiotis Tsiotras. Minimum-fuel powered descent in the presence of random disturbances. In *AIAA SciTech 2019 forum*, page 0646, 2019.

[181] Jack Ridderhof and Panagiotis Tsiotras. Minimum-fuel closed-loop powered descent guidance with stochastically derived throttle margins. *Journal of Guidance, Control, and Dynamics*, 44(3):537–547, 2021.

[182] Craig G Rieger, David I Gertman, and Miles A McQueen. Resilient control systems: Next generation design research. In *2009 2nd Conference on Human System Interactions*, pages 632–636. IEEE, 2009.

[183] Daniel Ruiz. A scaling algorithm to equilibrate both rows and columns norms in matrices. Technical report, CM-P00040415, 2001.

[184] David Rutishauser, John Prothro, and Jordan Fail. A system to provide deterministic flight software operation and maximize multicore processing performance: The safe and precise landing–integrated capabilities evolution (splice) datapath. In *2023 IEEE Space Computing Conference (SCC)*, pages 51–56. IEEE, 2023.

[185] David Rutishauser, Ray Ramadorai, John Prothro, Thadeus Fleming, and Peter Fidelman. NASA and Blue Origin collaborative assessment of precision landing algorithms and computing. In *AIAA SciTech 2021 Forum*, page 0377, 2021.

[186] Marco Sagliano, David Seelbinder, and Stephan Theil. Spartan: Rapid trajectory analysis via pseudospectral methods. In *International Conference on Astrodynamics Tools and Techniques*, 2021.

[187] A Miguel San Martin, Steven W Lee, and Edward C Wong. The development of the msl guidance, navigation, and control system for entry, descent, and landing. *AAS*, 2013.

[188] Kathrin Schacke. On the Kronecker product. *Master's thesis, University of Waterloo*, 2004.

[189] Maximilian Schaller, Goran Banjac, Steven Diamond, Akshay Agrawal, Bartolomeo Stellato, and Stephen Boyd. Embedded code generation with cvxpy. *IEEE Control Systems Letters*, 6:2653–2658, 2022.

[190] Daniel P Scharf, Behçet Açıkmeşe, Daniel Dueri, Joel Benito, and Jordi Casoliva. Implementation and experimental demonstration of onboard powered-descent guidance. *Journal of Guidance, Control, and Dynamics*, 40(2):213–229, 2017.

[191] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[192] Joseph K Scott, Davide M Raimondo, Giuseppe Roberto Marseglia, and Richard D Braatz. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69:126–136, 2016.

[193] Gwynne E Shotwell and Lars Blackmore. Space launch in 50 years: Abundance at last? *National Academy of Engineering*, 2021.

[194] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 1967.

[195] Marshall Smith, Douglas Craig, Nicole Herrmann, Erin Mahoney, Jonathan Krezel, Nate McIntyre, and Kandyce Goodliff. The Artemis program: An overview of NASA's activities to return humans to the Moon. In *2020 IEEE Aerospace Conference*, pages

1–10, 2020.

[196] Eduardo D Sontag. *Mathematical control theory: deterministic finite dimensional systems*, volume 6. Springer Science & Business Media, 2013.

[197] Pantelis Sopasakis, Emil Fresk, and Panagiotis Patrinos. Open: Code generation for embedded nonconvex optimization. *IFAC-PapersOnLine*, 53(2):6548–6554, 2020.

[198] Fabio Spada, Purnanand Elango, and Behçet Açıkmeşe. Impulsive relative motion control with continuous-time constraint satisfaction for cislunar space missions. In *2025 American Control Conference (ACC)*, pages 1719–1724. IEEE, 2025.

[199] Neeraj Srinivas, Abraham P Vinod, Stefano Di Cairano, and Avishai Weiss. Lunar landing with feasible divert using controllable sets. In *AIAA SciTech 2024 Forum*, page 0324, 2024.

[200] Joseph A Starek, Behçet Açıkmeşe, Issa A Nesnas, and Marco Pavone. Spacecraft autonomy challenges for next-generation space missions. In *Advances in control system technology for aerospace applications*, pages 1–48. Springer, 2015.

[201] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, February 2020.

[202] Lloyd Strohl, Javier Doll, Matthew Fritz, Andrew W Berning, Stephanie White, Stefan R Bieniawski, John M Carson III, and Behçet Açıkmeşe. Implementation of a six degree of freedom precision lunar landing algorithm using dual quaternion representation. In *AIAA SciTech 2022 Forum*, page 1831, 2022.

[203] Michael Szmuk and Behçet Açıkmeşe. Successive convexification for 6-DoF Mars rocket powered landing with free-final-time. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 0617, 2018.

[204] Michael Szmuk, Danylo Malyuta, Taylor P Reynolds, Margaret Skye Mceowen, and Behçet Açıkmeşe. 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)*, pages 7666–7673. IEEE, 2019.

[205] Michael Szmuk, Carlo Alberto Pascucci, and Behçet Açıkmeşe. Real-time quad-rotor path planning for mobile obstacle avoidance using convex optimization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[206] Michael Szmuk, Carlo Alberto Pascucci, Daniel Dueri, and Behçet Açıkmeşe. 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)*, pages 4862–4868. IEEE, 2017.

[207] Michael Szmuk, Taylor P Reynolds, Behçet A Açıkmeşe, Mesbahi Mehran, and John M Carson III. Successive convexification for 6-DoF powered descent guidance with compound state-triggered constraints. In *AIAA Guidance, Navigation, and Control Conference*, page 0926, San Diego, 2019.

[208] Michael Szmuk, Taylor P Reynolds, and Behçet Açıkmeşe. Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints. *Journal of Guidance, Control, and Dynamics*, 43(8):1399–1413, August 2020.

[209] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[210] Hans Raj Tiwary. On the hardness of computing intersection, union and minkowski sum of polytopes. *Discrete & Computational Geometry*, 40:469–479, 2008.

[211] Kento Tomita and Koki Ho. Reachability-steering approach for autonomous safe planetary landing. In *AIAA SciTech 2025 Forum*, page 1898, 2025.

[212] Kento Tomita, Yuri Shimane, and Koki Ho. Optimal predictive guidance for autonomous hazard detection and avoidance. In *AIAA SciTech 2024 Forum*, page 1585, 2024.

[213] Yung Liang Tong. *The multivariate normal distribution*. Springer Science & Business Media, 2012.

[214] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[215] Samet Uzun and Behçet Açıkmeşe. A proximal method for composite optimization with smooth and convex components, 2025.

[216] Samet Uzun, Behçet Açıkmeşe, and John M Carson III. Sequential convex programming for 6-DoF powered descent guidance with continuous-time compound state-triggered constraints. In *AIAA SciTech 2025 Forum*, page 1895, 2025.

[217] Samet Uzun, Behçet Açıkmeşe, and Stefano Di Cairano. Motion planning for information acquisition via continuous-time successive convexification. *IEEE Control Systems Letters*, 2025.

[218] Samet Uzun, Purnanand Elango, Pierre-Loic Garoche, and Behçet Açıkmeşe. Optimization with temporal and logical specifications via generalized mean-based smooth robustness measures. *arXiv preprint arXiv:2405.10996*, 2024.

[219] Samet Uzun, Purnanand Elango, Abhinav G Kamath, Taewan Kim, and Behçet Açıkmeşe. Successive convexification for nonlinear model predictive control with continuous-time constraint satisfaction. *IFAC-PapersOnLine*, 2024.

[220] Guido Van Rossum et al. Python programming language. In *USENIX annual technical*

conference, volume 41, pages 1–36. Santa Clara, CA, 2007.

[221] Lieven Vandenberghe. The CVXOPT linear and quadratic cone program solvers. *Online: http://cvxopt. org/documentation/coneprog. pdf*, 2010.

[222] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, 2022.

[223] Alexander Vinel and Pavlo A Krokhmal. Polyhedral approximations in p-order cone programming. *Optimization Methods and Software*, 29(6):1210–1237, 2014.

[224] Abraham P Vinod. pycvxset: A Python package for convex set manipulation. In *Proceedings of the American Control Conference (ACC), Denver, USA*, 2025.

[225] Abraham P Vinod, Abhinav G Kamath, Avishai Weiss, and Stefano Di Cairano. Set-based lossless convexification for a class of robust nonlinear optimal control problems. In *IEEE Conference on Decision and Control*, 2025. `https://www.merl.com/publications/docs/TR2025-160.pdf` (Accepted).

[226] Abraham P Vinod and Meeko MK Oishi. Stochastic reachability of a target tube: Theory and computation. *Automatica*, 125:109458, 2021.

[227] Abraham P Vinod, Avishai Weiss, and Stefano Di Cairano. Abort-safe spacecraft rendezvous under stochastic actuation and navigation uncertainty. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6620–6625. IEEE, 2021.

[228] Abraham P Vinod, Avishai Weiss, and Stefano Di Cairano. Inscribing and separating an ellipsoid and a constrained zonotope: Applications in stochastic control and centering. In *Proc. Conf. Dec. & Ctrl*, pages 1–8, 2024.

[229] Abraham P Vinod, Avishai Weiss, and Stefano Di Cairano. Projection-free computa-

tion of robust controllable sets with constrained zonotopes. *Automatica*, 175:112211, 2025.

[230] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[231] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 2010.

[232] A. J. Wathen. Preconditioning. *Acta Numerica*, 2015.

[233] Avishai Weiss, Morgan Baldwin, Richard Scott Erwin, and Ilya Kolmanovsky. Model predictive control for spacecraft rendezvous and docking: Strategies for handling constraints and case studies. *IEEE Transactions on Control Systems Technology*, 23(4):1638–1647, 2015.

[234] James Hardy Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, Inc., 1988.

[235] Samuel Williams, Leonid Oliker, Richard Vuduc, John Shalf, Katherine Yelick, and James Demmel. Optimization of sparse matrix-vector multiplication on emerging multicore platforms. In *SC'07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, pages 1–12. IEEE, 2007.

[236] Robert B. Wilson. *A simplicial algorithm for concave programming*. Doctoral dissertation, Harvard Business School, 1963.

[237] Yue Yu, Purnanand Elango, and Behçet Açıkmeşe. Proportional-integral projected gradient method for model predictive control. *IEEE Control Systems Letters*, 5(6):2174–2179, December 2021.

[238] Yue Yu, Purnanand Elango, Behçet Açıkmeşe, and Ufuk Topcu. Extrapolated

proportional-integral projected gradient method for conic optimization. *IEEE Control Systems Letters*, 7:73–78, 2023.

[239] Yue Yu, Purnanand Elango, Ufuk Topcu, and Behçet Açıkmeşe. Proportional–integral projected gradient method for conic optimization. *Automatica*, 2022.

[240] Yue Yu, Kartik Nagpal, Skye Mceowen, Behçet Açıkmeşe, and Ufuk Topcu. Real-time quadrotor trajectory optimization with time-triggered corridor constraints. *Journal of Guidance, Control, and Dynamics*, 2023.

[241] Yue Yu and Ufuk Topcu. Proportional-integral projected gradient method for infeasibility detection in conic optimization. *arXiv preprint arXiv:2109.02756*, 2021.

[242] Minsen Yuan and Yue Yu. Sequential convex programming with filtering-based warm-starting for continuous-time multiagent quadrotor trajectory optimization. *arXiv preprint arXiv:2508.14299*, 2025.

[243] Daxi Zhang and Yulin Zhang. PySCP: A multiple-phase optimal control software using sequential convex programming. *International Journal of Aerospace Engineering*, 2022, 2022.

[244] Xianyi Zhang, Wang Qian, Werner Saar, et al. OpenBLAS: An optimized BLAS library. `https://www.openblas.net/`, 2025.

Appendix A

QUATERNION & DUAL QUATERNION ALGEBRA

A.1 Quaternion Algebra

A.1.1 Unit quaternions

$$a = \left[a_v^\top,\, a_4\right]^\top,\ b = \left[b_v^\top,\, b_4\right]^\top \in \mathbb{R}_u^4 := \left\{ q \in \mathbb{R}^4 \;\middle|\; q^\top q = 1 \right\}$$

where

$$a_v = [a_1,\, a_2,\, a_3]^\top,\ b_v = [b_1,\, b_2,\, b_3]^\top \in \mathbb{R}^3 \text{ and } a_4, b_4 \in \mathbb{R}$$

Note: All quaternions are in accordance with the scalar-last convention.

A.1.2 Conjugation

$$a^* := \left[-a_v^\top,\, a_4\right]^\top$$

A.1.3 Skew-symmetric matrix operator

$$a_v^\times := \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

A.1.4 $SO(4)$ matrix operators

$$[a]_\otimes := \begin{pmatrix} a_4 I_3 + a_v^\times & a_v \\ -a_v^\top & a_4 \end{pmatrix}$$

$$[b]_\otimes^* := \begin{pmatrix} b_4 I_3 - b_v^\times & b_v \\ -b_v^\top & b_4 \end{pmatrix}$$

A.1.5 Multiplication

$$\begin{aligned} a \otimes b &:= \left[a_4 b_v + b_4 a_v + a_v \times b_v,\, a_4 b_4 - a_v^\top b_v\right]^\top \\ &= [a]_\otimes b \\ &= [b]_\otimes^* a \end{aligned}$$

A.1.6 Cross product

$$a \oslash b := \left[a_4 b_v + b_4 a_v + a_v \times b_v,\, 0\right]^\top$$

A.2 Dual Quaternion Algebra

A.2.1 Unit dual quaternions

$$\boldsymbol{a} = \left[a_1^\top,\, a_2^\top\right]^\top,\, \boldsymbol{b} = \left[b_1^\top,\, b_2^\top\right]^\top \in \mathbb{R}_u^8 := \left\{ \boldsymbol{q} = \left[q_1^\top,\, q_2^\top\right]^\top \in \mathbb{R}^8 \;\middle|\; q_1^\top q_1 = 1 \text{ and } q_1^\top q_2 = 0,\; q_1, q_2 \in \mathbb{R}^4 \right\}$$

A.2.2 Conjugation

$$\boldsymbol{a}^* := \begin{pmatrix} a_1^* \\ a_2^* \end{pmatrix}$$

A.2.3 Multiplication matrix operators

$$[\boldsymbol{a}]_\otimes := \begin{pmatrix} [a_1]_\otimes & 0_{4\times 4} \\ [a_2]_\otimes & [a_1]_\otimes \end{pmatrix}$$

$$[\boldsymbol{b}]_\otimes^* := \begin{pmatrix} [b_1]_\otimes^* & 0_{4\times 4} \\ [b_2]_\otimes^* & [b_1]_\otimes^* \end{pmatrix}$$

A.2.4 Multiplication

$$\boldsymbol{a} \otimes \boldsymbol{b} := [\boldsymbol{a}]_\otimes \boldsymbol{b} = [\boldsymbol{b}]_\otimes^* \boldsymbol{a}$$

A.2.5 Cross product

$$\boldsymbol{a} \oslash \boldsymbol{b} := \left[a_1 \oslash b_1,\, a_1 \oslash b_2 + a_2 \oslash b_1\right]^\top$$