Successive Convexification & High Performance Feedback Control for Agile Flight

Michael Szmuk

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

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Behçet Açıkmeşe, Chair

Mehran Mesbahi

Lillian Ratliff

Program Authorized to Offer Degree:
William E. Boeing Department of Aeronautics & Astronautics

University of Washington

Abstract

Successive Convexification & High Performance
Feedback Control for Agile Flight

Michael Szmuk

Chair of the Supervisory Committee:
Professor Behçet Açıkmeşe
William E. Boeing Department of Aeronautics & Astronautics

The topic of this dissertation is successive convexification and high performance feedback control for agile flight. This document is divided into three primary branches. The first branch (Chapter 2) is focused on successive convexification – a sequential convex programming approach that lends itself well to real-time applications requiring advanced feed-forward guidance. This branch uses a complex rocket landing application to showcase the versatility and computational capabilities of successive convexification.

The second branch (Chapters 3-5) applies convexification techniques to quad-rotor motion planning problems. Using flight test results, the work detailed in these chapters shows that convexification techniques like successive convexification can generate trajectories that are trackable by quad-rotors in real-time.

The third branch (Chapter 6) is dedicated to developing a high performance feedback control architecture for multi-rotor vehicles. This architecture is designed to complement a variety of feed-forward guidance methodologies by enhancing the system's robustness to model uncertainties and external disturbances. The core of this architecture relies on a fusion between advanced classical control techniques and convex optimization in order to systematically handle actuator saturation and feedback maximization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

A PhD is a journey of awakening. It certainly has been for me. And while I am not sure that I am able to properly thank everyone who has helped and inspired me throughout this journey, I will do my best.

I have to begin by thanking my advisor, Professor Behçet Açıkmeşe. Simply put, I cannot have asked for a better advisor. To say that I have benefited from his broad theoretical and practical knowledge, his clarity, and his energy would be an understatement. Thanks to him, I was given professional opportunities I could have only dreamt of, and met prominent names in our field that I would likely have never met. Behçet, I thank you for the faith you placed in me, and for all of the responsibility you entrusted me with during my PhD. But most of all, I am thankful for the potential you saw in me when I did not see it in myself. You have changed my life. For that I will be forever grateful.

I would like to thank Professors Mehran Mesbahi, Lillian Ratliff, Eli Livne, Chris Lum, and Sam Burden for serving on my General Exam and Final Exam committees. I thank you all for your time and for your thoughtful comments and suggestions. I would especially like to thank Mehran and Lillian for serving on my reading committee.

I would like to thank Professor Maruthi Akella from the University of Texas at Austin for giving me my first opportunity in graduate school, and for allowing me to explore the world of quad-rotor autopilot design. That coupled with his love of linear systems theory gave me a strong foundation upon which to build my doctoral work.

I would also like to thank my undergraduate mentor, Professor Armand J. Chaput, whom I met during my sophomore year in college. The stories of his experience in industry broadened my understanding of the field of aerospace in ways courses and textbooks never

could. I am grateful for all of the teaching and research opportunities he offered me. His reassuring voice of experience was particularly helpful in the midst of all of the aircraft design projects I undertook during my undergraduate studies.

I thank Professor Hans Mark for inspiring me on the first day I walked into Woolrich Laboratories during his opening Introduction to Aerospace Engineering lecture, and for the inspiration he provided on many occasions thereafter. It was a rare privilege to talk and listen to a man who had his hand in so many historic decisions. I thank him for his kindness and for the impact he had on me as a young engineering student.

I would like to thank Mark Maughmer II for his help, guidance, and friendship during my undergraduate and early graduate years. Without him, my undergraduate experience would have been devoid of some of its greatest practical lessons. His help and calm demeanor was instrumental in the success of the flight experiments carried out as part of my Masters thesis. The lessons I learned from him on running a lab proved invaluable in the later portions of my graduate school career. I consider myself incredibly lucky to have been in the presence of such a talented individual.

I would like to thank all the people I worked with at Blue Origin for all of their help and support throughout my internships. I want to thank Heather Nelson and Sarah Knights for overseeing the best internship problem in which I have partaken, and for tolerating my long presentations. I want to thank Carter Allen for his friendship throughout all of my internships at Blue, and for patiently entertaining the plethora of questions I always had about propulsion. I would especially like to thank my mentor Geoff Huntington, without whom I certainly would not have had the opportunity to participate in the new space race in the manner that I did. Geoff, thanks to you, my internships at Blue Origin proved to be invaluable experiences that helped me grow technically, and motivated me further in my academic research. I thank for your continued friendship, and will always aspire (perhaps in vain) to match your youthful and energetic spirit.

I would like to thank my team members at Amazon Prime Air for their help, support, and friendship. In particular, I want to thank Topher McFarland for making my internship at Amazon Prime Air possible, and for opening my eyes to a whole new world of exciting challenges. I look forward to joining your team shortly.

I would like to thank our sponsors from the Office of Naval Research. In particular, I want to thank Luke Steelman, Amy Dykstra, and Paul Conolly for their support, and for giving us the opportunity to develop and demonstrate our methodology.

I would like to thank the University of Washington Aeronautics & Astronautics staff, without whom my PhD work would have been infinitely more difficult. In particular, I would like to thank Nancy-Lou Polk for helping me purchase *virtually every item* that now sits in the Autonomous Controls Laboratory; Steve Pearson for helping me with countless administrative issues; Ed Connery for all of his support and help during the course of my degree and teaching assistantships; Patrick Gibbs for all of his help during the early stages of our time at the university; Michael Domar for all of his recent help in purchasing a myriad of items; Winnie Lin for making the process of travel reimbursement as easy as possible; Danyel Hacker for all of her help in arranging my final exam; and Steve Scheier for all of his recent help in re-configuring our lab's IT infrastructure.

I would like to thank the staff at Solstice Cafe for their great service, friendliness, and delicious cappuccinos. Without them, I do not know where I would have worked when I could no longer stand being in the lab.

I want to thank all of the students I have taught over the years. Teaching has been one the most rewarding experiences of my graduate school career. This was only made possible by the countless students that made teaching such a joy.

I want to thank all of my lab mates for their friendship, help, and interesting discussions over the years. In particular, I would like to thank Tim Lowery for all of his help during the early years of my PhD; AJ Berning for all the good times we had working on trajectories early

on; Carlo Pascucci for his crucial role in setting up our lab space, and for his fastidious nature that vastly improved the appearance of the lab; Yue Yu, Uktu Eren, and Nazli Demirer for their friendship and help during our transition to UW; Yuanqi Mao for his help in developing successive convexification; Sean Rice for his help in testing various elements of the autopilot software; and Mo Zhao for his help in a multitude of projects in our lab, for the many interesting discussions, and for never missing a chance to remind me of my age.

To my close research colleagues, Taylor Reynolds and Danylo Malyuta, meeting you guys is one of the best things that happened to me during my academic career. Our interactions reinvigorated me, and gave me the energy I needed to get to the finish line. I consider myself lucky for having had the opportunity to work with such talented and hard working individuals.

I want to also thank Skye Mceowen for all of her effort and dedication in the process of handing over the lab. Skye, without your help, I would not have been able to complete my PhD. I am impressed and inspired by your hard work, determination, and courage. Having watched the progress you have already made, I have every confidence that the lab is being left in good hands.

I would like to thank my fourth grade ESL teacher Ms. Julia Love who believed in me and helped me over the language barrier, and my sixth and eighth grade teacher Mr. Patrick Cone for believing in me during a particularly formative part of my childhood. I would like to thank Professor Viorel Florescu for his role in shaping my mathematical and abstract thinking at a young age, and for his continued friendship. To my soccer coach and friend John Elder, I thank you for your guidance, support, and friendship after all these years, and for always inspiring me to look at the bright side of life. To my friend Coitt Kessler, thank you for your friendship and for inspiring me to be a better human being. I stand in awe of your energy and positive spirit. I cannot wait to get french toast next time I am in town. Lastly, I would like to thank my rowing coach Bob Krentler, whose intensity and structure

taught me that success is not an accident, but rather a deliberate slow process that requires sacrifice, dedication, and perseverance. Without knowing it at the time, the lessons I learned from his coaching structured the mental fortitude I needed to overcome the challenges I faced in my academic and professional careers.

To Jonathan Tamir, thank you for your friendship and for all of the enlightening conversations we have had over the years. The days of the TAC, MOC, and the Johnson Criteria are long behind us, but I always look forward to the next time we hang out.

I would like to thank my long time friends Jahshan Bhatti and Daniel Dueri for their friendship throughout our years in undergrad and graduate school. I am lucky to have met you guys early in my academic career. I have learned so much from both of you, and I cannot imagine having gone through this journey without you guys. I count myself lucky to have been in the presence of such bright and talented individuals.

To my brothers, Soody and Amir, I am grateful to be a part of your lives, and to know your beautiful families. You guys inspired me to be a better version of myself at every step of the way. I am incredibly lucky to have friends like you, whom celebrate with me during the good times, and guide me the light during the bad times. I hope that we always make time to get our families together, and I cannot wait to stop hearing you guys ask me "when are you going to graduate?" I love you guys.

To my Morgan, I love you and I cannot wait to begin our life together. Although you will no longer be able to get me student discounts when we go to museums, I suspect that finishing my PhD will come with some other advantages. Thank you for being there for me during the good days and the bad. Thank you for proof reading all of my papers. And thank you for all of your encouragement and support, especially during the final months of my PhD.

Most of all, I am eternally grateful to my parents, who provided me with every opportunity I could have ever asked for, and more. My gratitude is that of a child who knows not

what he did to deserve all he was given. Mom and Dad, I have done my best to earn what you have given me, and my best to live up to all of the sacrifices and hardships you and our family have endured. My success is your success. I hope you can take this work as a token of my gratitude.

DEDICATION

This dissertation is dedicated to my loving family: Mom, Dad, & Lisa.

Chapter 1

INTRODUCTION

This dissertation focuses on the development of feed-forward guidance and feedback control methodologies for agile flight. Feed-forward and feedback strategies address two philosophically different aspects that are central to controlling the motion of a vehicle.

Feed-forward guidance understands the effect current actions have on future states, the limitations of the system, and the constraints imposed on the vehicle by the mission or the surrounding environment. Feed-forward control is model-based in the sense that it uses an idealized model of the vehicle to decide what system inputs are required to guide the vehicle through its constrained environment. This reliance on a system model makes feed-forward strategies sensitive and leaves them vulnerable to uncertainties in the model and in the surrounding environment.

Feedback control, on the other hand, is myopic in that it is typically blind to the future implications of current actions. Moreover, feedback control is often ignorant to system and environmental constraints. Instead, feedback control is focused on sensing deviations from a desired state and applying robust corrective actions to eliminate said deviations. This property of feedback control gives the system robustness to model uncertainties and external disturbances.

When employed together, the predictive powers of feed-forward guidance are complemented by the robustness afforded by feedback control. This division of labor between feed-forward and feedback has long been understood and practiced, and is the overarching theme of this work. This dissertation can be divided into three branches.

The first branch is the topic of Chapter 2, and involves the development of successive convexification, a sequential convex programming approach to solving non-convex optimal

control problems. This development was carried out in the context of a generalized rocket landing problem. One significant contribution resulting from this work was the formulation of state-triggered constraints (STCs), a concise constraint formulation that allows complex discrete decisions to be embedded in continuous optimization frameworks (like successive convexification). While the resulting formulation is not convex, we have found that in practice it works very well with successive convexification, and in many cases is a viable real-time alternative to mixed-integer programming approaches. While the concept behind STCs is not entirely new, to the best of our knowledge, STC-like constraints have not been presented in the optimal control literature to date. We hope that the work in this (and latter) chapters sufficiently motivates the further study of STCs and their properties. In this branch of work, STCs and successive convexification have been shown to handle a wide variety of rocket landing scenarios with promising real-time capabilities. This work is closely associated with the following publications: [60, 75, 76, 83–85, 89, 90].

The second branch is the topic of Chapters 3-5, and involves the application of convexification techniques to agile quad-rotor guidance. Chapter 3 details the implementation of real-time lossless convexification [5] and successive convexification for agile quad-rotor maneuvering and static obstacle avoidance. This work demonstrated that three-degree-of-freedom models can be used to effectively and aggressively command quad-rotors. Moreover, this work specifically addressed non-convex thrust lower-bound and tilt angle constraints that were not captured by existing methodologies [68, 71]. Chapter 4 details the application of real-time successive convexification to mobile obstacle avoidance applications. This work highlights that successive convexification can be used to model non-convex obstacle keep-out zones at sufficiently fast rates to react to environmental (i.e. extrinsic) uncertainties. Chapter 5 employs STCs – originally developed for rocket landing applications – to solve more advanced quad-rotor applications such as flying through hoops and collaboratively negotiating an obstacle course. All three chapters present algorithm timing results, while only Chapters 3 and 4 present flight test results. The novelty in this branch stems from the application of convex optimization techniques to real-time quad-rotor motion planning. The

key advantage of the proposed methodology is its ability to explicitly incorporate a variety of important practical vehicle and mission constraints with real-time to near-real-time performance. While the complexity of these constraints makes the convergence properties of the algorithm more difficult to analyze, empirical results have shown that this methodology is scalable, accommodating free-final-time formulations, a variety of nonlinear dynamics, and discrete decision constraints. This branch of work is associated with the following publications: [86–88]. Videos of the associated flight demonstrations can be found online on the Autonomous Control Laboratory's YouTube channel: Agile Non-Convex Flip Maneuver, Static Obstacle Avoidance (1), Static Obstacle Avoidance (2), Mobile Obstacle Avoidance.

The third branch is the topic of Chapter 6, and involves the development of a comprehensive high performance feedback control architecture for multi-rotor vehicles. This architecture is designed to interface easily with guidance strategies of varying fidelity, ranging from the six degree-of-freedom minimum-snap guidance strategy presented in [68] to manually specified low fidelity trajectories. The novelty in this branch stems from the use of advanced classical feedback control techniques in conjunction with convex optimization to obtain a high performance full-envelope control architecture that interfaces well with existing guidance strategies.

Chapter 2

SUCCESSIVE CONVEXIFICATION FOR 6-DOF POWERED DESCENT GUIDANCE

2.1 Introduction

This chapter presents a real-time guidance algorithm that solves a generalized free-final-time 6-degree-of-freedom (DoF) powered descent guidance problem with free ignition time, aerodynamic effects, and conditionally enforced constraints. Real-time optimal guidance algorithms are an enabling technology for future manned and unmanned planetary missions that require autonomous precision landing capabilities. Such algorithms allow for the explicit inclusion of operational and mission constraints, and are able to compute trajectories that are (locally) optimal with respect to key metrics, such as propellant consumption, burn time, or miss distance. Consequently, these algorithms enhance a lander's ability to recover from a wider range of dispersions encountered during the entry, descent, and landing phase, and to react to obstructions on the surface that become apparent only as the vehicle approaches the landing site.

The constraint satisfaction afforded by these algorithms allows designers to select more ambitious and scientifically interesting landing sites, enhances the lander's ability to handle uncertainties, and ultimately increases the likelihood of mission success. Moreover, the optimality may be leveraged to improve the scientific return of planetary missions by reducing the propellant mass fraction. The relevance of real-time optimal guidance algorithms been demonstrated in the (now routine) landings of orbital-class vertical-takeoff-vertical-landing reusable launch vehicles [13].

Solving the 6-DoF powered descent guidance problem in real-time is challenging for several reasons. First, the problem consists of nonlinear dynamics and non-convex state and

control constraints, and does not yet have an analytical solution. Second, since the problem must be solved using numerical methods, the validity of the solution depends heavily on the discretization scheme. Third, the non-convex nature of the problem makes it difficult to select a suitable reference trajectory to initialize an iterative solution process. The successive convexification methodology used in this chapter addresses these issues, and is able to solve the problem quickly and reliably over a wide range of conditions.

2.1.1 Related Research

The powered descent guidance literature can be separated into works that consider 3-DoF translation dynamics, and ones that consider more general 6-DoF rigid body dynamics. Work on 3-DoF powered descent guidance began many years ago during the Apollo program, with several authors approaching the problem using optimal control theory [51,66] and the calculus of variations [64]. These early works noted that the solution to the fuel-optimal powered descent guidance problem exhibited bang-bang behavior, where the thrust assumed either the minimum or maximum allowable value everywhere along the optimal trajectory. While these results offered important insights into the powered descent guidance problem, they were not incorporated into the Apollo flight code, since the polynomial-based guidance methods used to perform the landings were deemed sufficiently close to optimal, and were far simpler to design [48]. In the years following Apollo, researchers continued searching for analytical solutions to the 3-DoF landing problem [8, 20, 30, 49] that would be conducive to on-board implementation for future missions.

Unmanned missions to the Martian surface in the early the 2000s renewed interest in using direct methods to compute solutions to the powered descent guidance problem. In 2005, Topcu, Casoliva, and Mease presented results for the 3-DoF problem that resembled a modern take on [51], adding numerical simulations to reinforce and demonstrate theoretical results [92, 93]. Around the same time, Açıkmeşe and Ploen published work on a convex programming approach to the problem [4, 5, 74]. Using Pontryagin's maximum principle, they showed that the non-convex thrust magnitude lower bound constraint could

be losslessly convexified by introducing a relaxation that rendered the optimal solution of the (now convex) relaxed problem identical to that of the original non-convex problem. Hence, the difficult task of solving the non-convex 3-DoF powered descent guidance problem was converted into the far simpler, but equivalent, task of solving a convex second-order cone programming problem. Subsequently, lossless convexification was extended to include non-convex pointing constraints, and to encompass more general optimal control problems [2, 2, 3, 14, 15, 41, 42]. This methodology was demonstrated in a sequence of flight experiments in the early 2010s [32, 79, 80].

More recently, researchers have considered the 6-DoF powered descent guidance problem. In 2012, Lee and Mesbahi formulated the powered descent guidance problem using dual quaternions [52]. This work revealed that certain coupled rotational-translational constraints are convex when using this parameterization [53]. In [54], a piecewise-affine approximation of the dynamics was used to design a model predictive controller that generated thrust and torque commands. However, since the accuracy of the equations of motion relied heavily on the temporal resolution of the approximation, the approach resulted in either short prediction horizons or prohibitively large optimization problems.

Work on powered descent guidance and atmospheric entry problems turned to Sequential Convex Programming (SCP) methods in order to handle more general non-convexities [23, 56, 58, 94]. These methods solve non-convex problems by iteratively solving a sequence of local convex approximations obtained via linearization. The first order approach used in SCP methods guarantees that the approximations are convex, and lies in stark contrast to the second order approach used in Sequential Quadratic Programming (SQP) methods [17], which may expend significant computational effort to ensure the convexity of each approximation.

In this chapter, we solve a generalized free-final-time powered descent guidance problem using successive convexification. Successive convexification can be classified as an SCP method that uses virtual control and trust region modifications to facilitate convergence. The algorithms detailed in [62] use an exact penalty method in conjunction with hard trust regions, whereas those used in this chapter and in [83–85] employ soft trust regions.

2.1.2 Statement of Contribution

This chapter presents three primary contributions: (i) a free-ignition-time modification that allows the algorithm to determine the optimal engine ignition time, (ii) a tractable aerodynamics formulation that models both lift and drag, and (iii) a continuous state-triggered constraint formulation that emulates conditionally enforced constraints. In particular, contribution (iii) effectively allows constraints to be enabled or disabled by *if*-statements in a continuous optimization framework.

To the best of our knowledge, state-triggered constraints bear the most resemblance to two existing approaches: mixed-integer programming, and complementarity constraints. The former approach implements discrete decisions explicitly using integer variables, whereas the latter formulates such decisions implicitly using continuous variables. Despite the existence of efficient branch and bound techniques, mixed-integer formulations can suffer from poor complexity [77], and are not conducive to solving the generalized powered descent guidance problem in real-time. Like state-triggered constraints, complementarity constraints [25, 43] permit completely continuous formulations that can be more efficient than mixed-integer approaches (see Section 2.3 in [12]). However, complementarity constraints represent bi-directional *if-and-only-if*-statements, whereas state-triggered constraints represent more general uni-directional *if*-statements. Therefore, we argue that the proposed continuous state-triggered constraints are a key building block that enable the formulation of a broader set of constraints.

The secondary contributions of this chapter are an improved description of the successive convexification methodology used in [83], and timing results that demonstrate the real-time capabilities of the algorithm. This chapter regards the powered descent guidance problem as a feed-forward trajectory generation problem, and does not address the topic of feedback control or issues arising from trajectory re-computation.

2.1.3 Outline

In §2.2, we present the primary contributions of this chapter in the context of a generalized powered descent guidance problem. In §2.3, we detail the successive convexification procedure and algorithm. In §2.4, we present simulation results that highlight the contributions, and timing results that demonstrate the real-time capabilities of the proposed algorithm. Lastly, §2.5 provides concluding remarks.

2.2 Problem Statement

In this section, we present a 6-DoF formulation for a generalized powered descent guidance problem in the presence of atmospheric effects. This section is organized as follows. In §2.2.1 and §2.2.2, we introduce the assumptions and notation used in our formulation. In §2.2.3, we present a baseline problem formulation. In §2.2.4, §2.2.5, and §2.2.6, we discuss the primary contributions, namely the free-ignition-time modification, the aerodynamic models, and state-triggered constraints. Lastly, §2.2.7 provides a statement of the non-convex generalized powered descent guidance problem.

2.2.1 Assumptions

Most powered descent maneuvers commence at speeds substantially below orbital velocities and within only a few kilometers of the landing site. Hence, we neglect the effects of planetary rotation and assume a uniform gravitational field. However, we note that the inclusion of planetary rotation and higher-fidelity gravitational models is straightforward using the proposed framework. We assume that the vehicle is equipped with a single rocket engine that can be gimbaled symmetrically about two axes up to a maximum gimbal angle, but stress that other actuator configurations can be readily accommodated. Further, we assume that the engine can be throttled between fixed minimum and maximum thrusts, and that once the engine is ignited it remains on until the terminal condition is reached. To tailor our treatment to applications with non-negligible atmospheric effects, we assume that the

ambient atmospheric density and pressure are constant, that the aerodynamic forces are governed by the simplified models detailed in §2.2.5, and that the center-of-pressure is fixed with respect to a body-fixed reference frame. Further, we neglect the effects of winds, noting that constant uniform wind profiles can be readily incorporated into our formulation, and account for thrust reduction induced by atmospheric back-pressure by assuming the affine mass depletion dynamics in [83]. Lastly, to make the problem tractable, we neglect higher order phenomena such as elastic structural modes and fuel slosh, and model the vehicle as a rigid body with a constant body-fixed center-of-mass and inertia.

2.2.2 Notation

We begin by denoting time as $t \in \mathbb{R}$, and define the *initial time* t_{in} as the time at which the optimal control problem begins, the *ignition time* t_{ig} as the time at which the engine turns on, and the *final time* t_f as the time at which the vehicle reaches the terminal condition. These epochs are defined such that $t_{in} \leq t_{ig} < t_f$, and their subscripts are used to denote problem parameters associated with the respective time epochs. Further, we define *coast time* as $t_c \coloneqq t_{ig} - t_{in}$ and *burn time* as $t_b \coloneqq t_f - t_{ig}$. During the coast phase, the vehicle's states passively evolve according to its engine-off dynamics, whereas during the burn phase, the vehicle actively maneuvers in order to achieve its landing objective. Without loss of generality, we define the ignition time epoch as the time at which $t = 0$, where it follows that $t_c = -t_{in}$ and that $t_b = t_f$. This timeline is illustrated in Figure 2.1.

The subscripts $\mathcal{I}$ and $\mathcal{B}$ are used to denote problem parameters expressed in the inertial and body-fixed reference frames $\mathcal{F}_\mathcal{I}$ and $\mathcal{F}_\mathcal{B}$, respectively. We define $\mathcal{F}_\mathcal{I}$ as a surface-fixed Up-East-North coordinate frame whose origin coincides with the landing site. Likewise, we define $\mathcal{F}_\mathcal{B}$ such that its origin coincides with the vehicle's center-of-mass, its x-axis points along the vertical axis of the vehicle, its y-axis points out the side of the vehicle, and its z-axis completes the right-handed system. We use $m(t) \in \mathbb{R}_{++}$, $r(t) \in \mathbb{R}^3$, and $v(t) \in \mathbb{R}^3$ to respectively denote the mass, position, and velocity states, and $T(t) \in \mathbb{R}^3$ and $A(t) \in \mathbb{R}^3$ to respectively denote the thrust vector and aerodynamic force. We denote the unit

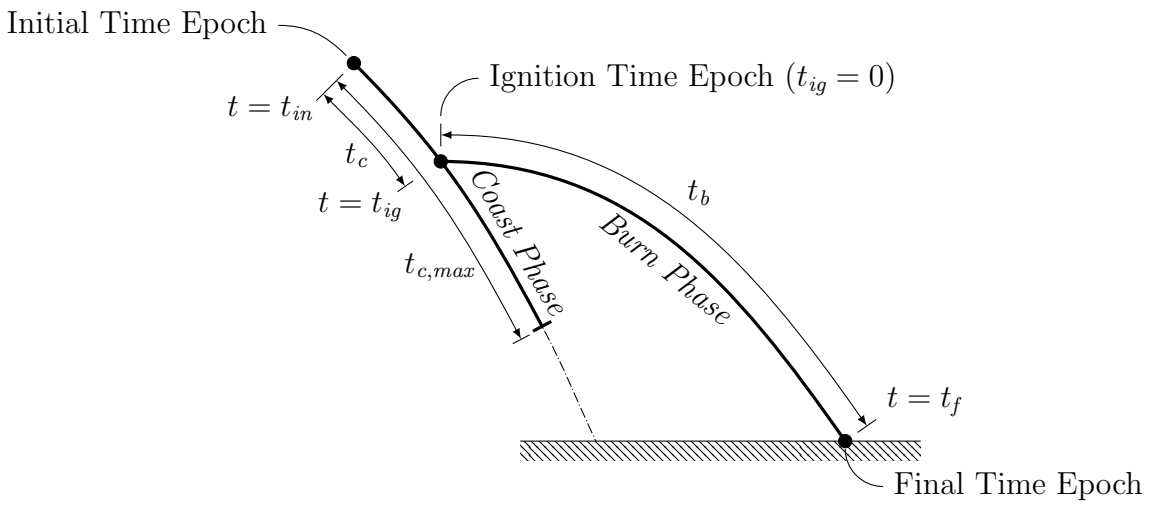


Figure 2.1: Timeline of the generalized powered descent guidance problem. The problem begins at the initial time epoch t_{in} at a prescribed state and with the engine off. The vehicle is allowed to coast a maximum duration of $t_{c,max}$ before engine ignition occurs at the ignition time epoch t_{ig}. The problem ends at the final time epoch t_f, when the vehicle lands at the prescribed landing site.

quaternion that parameterizes the transformation from $\mathcal{F}_\mathcal{I}$ to $\mathcal{F}_\mathcal{B}$ by $q_{\mathcal{B}\leftarrow\mathcal{I}}(t) \in \mathcal{S}^3 \subset \mathbb{R}^4$, and its corresponding direction cosine matrix by $C_{\mathcal{B}\leftarrow\mathcal{I}}(t) \coloneqq C_{\mathcal{B}\leftarrow\mathcal{I}}\big(q_{\mathcal{B}\leftarrow\mathcal{I}}(t)\big) \in \mathrm{SO}(3)$ [40]. The conjugate of $q_{\mathcal{B}\leftarrow\mathcal{I}}(t)$ is denoted by $q^*_{\mathcal{B}\leftarrow\mathcal{I}}(t)$, where it follows that $C_{\mathcal{I}\leftarrow\mathcal{B}}(t) \coloneqq C^T_{\mathcal{B}\leftarrow\mathcal{I}}(t) = C_{\mathcal{B}\leftarrow\mathcal{I}}\big(q^*_{\mathcal{B}\leftarrow\mathcal{I}}(t)\big)$. Quaternion multiplication and the identity quaternion are denoted by $\otimes$ and q_{id}, respectively. The angular velocity of $\mathcal{F}_\mathcal{B}$ relative to $\mathcal{F}_\mathcal{I}$ is denoted by $\omega_\mathcal{B}(t) \in \mathbb{R}^3$, and is expressed in body-fixed coordinates. Lastly, we use e_i to denote the i^{th} basis vector of $\mathbb{R}^n$, $\times$ to denote the vector cross product, and $\cdot$ to denote the vector dot product.

2.2.3 Baseline Problem

Dynamics: As stated in §2.2.1, the mass-depletion dynamics are assumed to be an affine function of the thrust magnitude, and are given by

$$\dot{m}(t) = -\alpha_{\dot{m}} \left\| T_\mathcal{B}(t) \right\|_2 - \beta_{\dot{m}} \,, \tag{2.1}$$

where $\alpha_{\dot{m}} \coloneqq 1/I_{sp}g_0$ and $\beta_{\dot{m}} \coloneqq \alpha_{\dot{m}} P_{amb}\, A_{noz}$, I_{sp} is the vacuum specific impulse of the engine, g_0 is standard Earth gravity, A_{noz} is the nozzle exit area of the engine, and P_{amb} is ambient atmospheric pressure. The second term in (2.1) represents the specific impulse reduction incurred by atmospheric back-pressure, and assumes that the nozzle remains choked over the allowable throttle range.

The translational states are governed by the following dynamics

$$\dot{r}_{\mathcal{I}}(t) = v_{\mathcal{I}}(t)\,, \tag{2.2a}$$

$$\dot{v}_{\mathcal{I}}(t) = \frac{1}{m(t)} F_{\mathcal{I}}(t) + g_{\mathcal{I}}, \tag{2.2b}$$

where $g_{\mathcal{I}} \in \mathbb{R}^3$ and $F_{\mathcal{I}}(t) \coloneqq C_{\mathcal{I}\leftarrow\mathcal{B}}(t) T_{\mathcal{B}}(t) + A_{\mathcal{I}}(t) \in \mathbb{R}^3$ respectively denote the constant gravitational acceleration and net propulsive and aerodynamic force acting on the vehicle. The thrust vector is expressed in $\mathcal{F}_{\mathcal{B}}$ coordinates to simplify the attitude dynamics and control constraints that follow. The aerodynamic term $A_{\mathcal{I}}(t)$ is defined §2.2.5.

The attitude states are governed by the following rigid-body attitude dynamics

$$\dot{q}_{\mathcal{B}\leftarrow\mathcal{I}}(t) = \frac{1}{2}\Omega\big(\omega_{\mathcal{B}}(t)\big) q_{\mathcal{B}\leftarrow\mathcal{I}}(t)\,, \tag{2.3a}$$

$$J_{\mathcal{B}}\dot{\omega}_{\mathcal{B}}(t) = M_{\mathcal{B}}(t) - \omega_{\mathcal{B}}(t) \times J_{\mathcal{B}}\omega_{\mathcal{B}}(t)\,, \tag{2.3b}$$

where $\Omega(\cdot)$ is a skew-symmetric matrix defined such that the quaternion kinematics in (2.3a) hold [40], $J_{\mathcal{B}} \in \mathbb{S}^3_{++}$ denotes the body-fixed constant inertia tensor of the vehicle about its center of mass, and $M_{\mathcal{B}}(t) \coloneqq r_{T,\mathcal{B}} \times T_{\mathcal{B}}(t) + r_{cp,\mathcal{B}} \times A_{\mathcal{B}}(t) \in \mathbb{R}^3$ denotes the net propulsive and aerodynamic torque acting on the vehicle. The vectors $r_{T,\mathcal{B}} \in \mathbb{R}^3$ and $r_{cp,\mathcal{B}} \in \mathbb{R}^3$ give the constant positions of the engine gimbal pivot point and the aerodynamic center of pressure, respectively. The aerodynamic term $A_{\mathcal{B}}(t)$ is defined in §2.2.5.

State Constraints: We impose four state constraints in our baseline formulation. First, we constrain the mass of the vehicle to values greater than a minimum dry mass $m_{dry} \in \mathbb{R}_{++}$ by enforcing

$$m(t) \geq m_{dry}. \tag{2.4}$$

Second, we constrain the inertial position to lie inside of a glide-slope cone with half-angle $\gamma_{gs} \in [0^\circ\,, 90^\circ)$ and vertex at the origin of $\mathcal{F}_\mathcal{I}$ by enforcing

$$e_1 \cdot r_\mathcal{I}(t) \geq \tan\gamma_{gs} \left\| H_\gamma r_\mathcal{I}(t) \right\|_2 , \tag{2.5}$$

where $H_\gamma \coloneqq [e_2 \;\; e_3]^T \in \mathbb{R}^{2\times 3}$. Third, we define the tilt angle of the vehicle as the angle between the x-axes of $\mathcal{F}_\mathcal{I}$ and $\mathcal{F}_\mathcal{B}$, and constrain it to be less than a maximum tilt angle $\theta_{max} \in (0^\circ\,, 90^\circ]$ by enforcing

$$\cos\theta_{max} \leq 1 - 2\left\| H_\theta q_{\mathcal{B}\leftarrow\mathcal{I}}(t) \right\|_2 , \tag{2.6}$$

where $H_\theta \coloneqq [e_3 \;\; e_4]^T \in \mathbb{R}^{2\times 4}$ if a scalar-first quaternion convention is used. Fourth, we limit the angular velocity to a maximum value of $\omega_{max} \in \mathbb{R}_{++}$ by enforcing

$$\left\| \omega_\mathcal{B}(t) \right\|_2 \leq \omega_{max}. \tag{2.7}$$

Control Constraints: We impose two control constraints in our baseline formulation. First, we impose lower and upper bounds on the thrust magnitude such that

$$0 < T_{min} \leq \left\| T_\mathcal{B}(t) \right\|_2 \leq T_{max} , \tag{2.8}$$

where T_{min} and T_{max} are the minimum and maximum allowable thrust magnitudes, respectively. Second, we constrain the thrust vector to lie within a prescribed maximum gimbal angle $\delta_{max} \in (0^\circ\,, 90^\circ)$ relative to the x-axis of $\mathcal{F}_\mathcal{B}$ by enforcing

$$\cos\delta_{max} \left\| T_\mathcal{B}(t) \right\|_2 \leq e_1 \cdot T_\mathcal{B}(t). \tag{2.9}$$

Boundary Conditions: We now present a notional set of boundary conditions, with the understanding that they may be modified to accommodate different scenarios. The conditions at the ignition time epoch are given by

$$m(t_{ig}) = m_{ig} , \quad r_\mathcal{I}(t_{ig}) = r_{\mathcal{I},ig} , \quad v_\mathcal{I}(t_{ig}) = v_{\mathcal{I},ig} , \quad \omega_\mathcal{B}(t_{ig}) = 0 , \tag{2.10}$$

where $m_{ig} \in \mathbb{R}_{++}$, $r_{\mathcal{I},ig} \in \mathbb{R}^3$, and $v_{\mathcal{I},ig} \in \mathbb{R}^3$ are the prescribed mass, position, and velocity at ignition time t_{ig}, respectively. We assume that $m_{ig} > m_{dry}$. The conditions at the final

time epoch are given by

$$r_{\mathcal{I}}(t_f) = 0\,, \quad v_{\mathcal{I}}(t_f) = -v_d e_1\,, \quad q_{\mathcal{B}\leftarrow\mathcal{I}}(t_f) = q_{id}\,, \quad \omega_{\mathcal{B}}(t_f) = 0\,, \tag{2.11}$$

where $v_d \in \mathbb{R}_+$ is the prescribed terminal vertical descent speed.

2.2.4 Free Ignition Time

In the baseline problem formulation presented in §2.2.3, $r_{\mathcal{I}}(t_{ig})$ and $v_{\mathcal{I}}(t_{ig})$ are restricted to prescribed *points* at the time of ignition. We now introduce *the free-ignition-time modification* to (2.10), such that $r_{\mathcal{I}}(t_{ig})$ and $v_{\mathcal{I}}(t_{ig})$ are constrained to a prescribed *curve* as follows

$$m(t_{ig}) = m_{ig}\,, \quad r_{\mathcal{I}}(t_{ig}) = p_{r,ig}(t_c)\,, \quad v_{\mathcal{I}}(t_{ig}) = p_{v,ig}(t_c)\,, \quad \omega_{\mathcal{B}}(t_{ig}) = 0\,, \tag{2.12}$$

where the coast time $t_c \in [0\,, t_{c,max}]$ is included as a solution variable, and $p_{r,ig} : \mathbb{R} \to \mathbb{R}^3$ and $p_{v,ig} : \mathbb{R} \to \mathbb{R}^3$ are vector valued polynomial functions of t_c that describe an engine-off trajectory. We choose these polynomials to represent an aerodynamics-free trajectory using

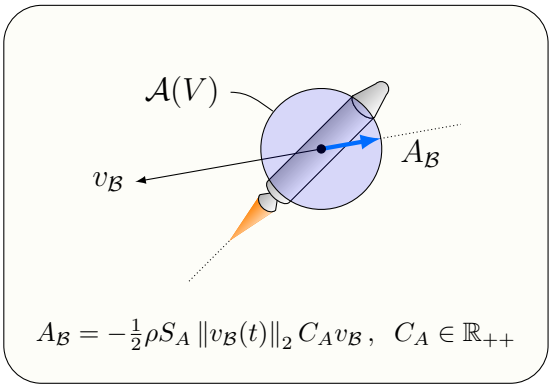
$$p_{r,ig}(\xi) \coloneqq r_{\mathcal{I},in} + v_{\mathcal{I},in}\,\xi + \frac{1}{2} g_{\mathcal{I}} \xi^2, \quad p_{v,ig}(\xi) \coloneqq v_{\mathcal{I},in} + g_{\mathcal{I}} \xi, \tag{2.13}$$

where $r_{\mathcal{I},in}$ and $v_{\mathcal{I},in}$ are prescribed position and velocity vectors at the initial time epoch (see Figure 2.1). Higher order effects (e.g., aerodynamics) can be embedded in $p_{r,ig}(\cdot)$ and $p_{v,ig}(\cdot)$ by using higher fidelity models to propagate the vehicle state over a prediction horizon of length $t_{c,max}$, and fitting polynomials to the resulting path.

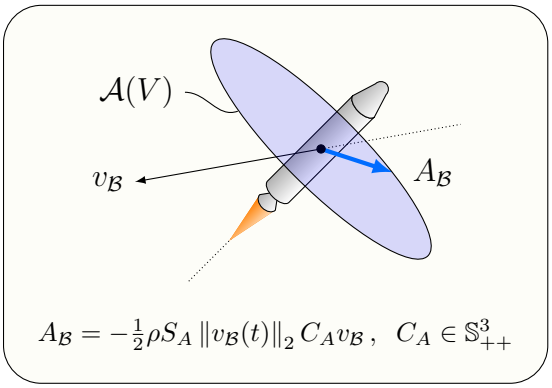
2.2.5 Aerodynamic Model

We now introduce a tractable aerodynamic model that approximates the relationship between the aerodynamic force and the velocity vector. The model expresses the aerodynamic force in $\mathcal{F}_{\mathcal{B}}$ coordinates as follows

$$A_{\mathcal{B}}(t) = -\frac{1}{2} \rho V(t) S_A C_A C_{\mathcal{B}\leftarrow\mathcal{I}}(t) v_{\mathcal{I}}(t)\,, \quad C_A \in \mathbb{S}^3_{++}\,, \tag{2.14}$$



(a) Spherical Aerodynamic Model



(b) Ellipsoidal Aerodynamic Model

Figure 2.2: Depictions of the aerodynamic models introduced in §2.2.5. The spherical model generates drag forces only, whereas the ellipsoidal model can generate both lift and drag forces. The ellipsoidal model introduced in this chapter is a generalization of the spherical model in [84].

where ρ is the ambient atmospheric density, $V(t) \coloneqq \|v_{\mathcal{I}}(t)\|_2$, and $S_A \in \mathbb{R}_{++}$ is a constant reference area. We refer to C_A as the aerodynamic coefficient matrix, and emphasize that it is a symmetric-positive-definite matrix that generalizes the standard scalar definition. Our definition of C_A ensures that for any $V \in \mathbb{R}_+$ the set $\mathcal{A}(V) \coloneqq \left\{ A_{\mathcal{B}} : A_{\mathcal{B}} = -\frac{1}{2}\rho V^2 S_A C_A \hat{v}_{\mathcal{B}}\,,\ \hat{v}_{\mathcal{B}} \in \mathcal{S}^2 \right\}$ defines a fixed ellipsoid in $\mathcal{F}_{\mathcal{B}}$ coordinates. Since most rocket-powered vehicles are approximately axisymmetric, we assume $C_A = \mathbf{diag}\left([c_{a,x},\ c_{a,yz},\ c_{a,yz}]\right)$, where $c_{a,x}$ and $c_{a,yz}$ are positive scalars. This assumption aligns the principal axes of $\mathcal{A}(V)$ with the axes of $\mathcal{F}_{\mathcal{B}}$.

If $c_{a,x} = c_{a,yz}$, then $A_{\mathcal{B}}(t)$ is always anti-parallel to $v_{\mathcal{B}}(t)$. In this case, $A_{\mathcal{B}}(t)$ may be interpreted as a pure drag force, and the model recovers the aerodynamic drag model used in [84]. Since the set $\mathcal{A}(V)$ corresponding to this choice of C_A defines a sphere, we refer to the corresponding model as the *spherical aerodynamic model*, illustrated in Figure 2.2a. Under the assumptions of the spherical model, the product $C_{\mathcal{I}\leftarrow\mathcal{B}}(t) C_A C_{\mathcal{B}\leftarrow\mathcal{I}}(t)$ simplifies to $c_{a,x} I_{3\times 3}$, thus rendering $A_{\mathcal{I}}(t) = C_{\mathcal{I}\leftarrow\mathcal{B}}(t) A_{\mathcal{B}}(t)$ independent of attitude.

Alternatively, if $c_{a_x} \neq c_{a,yz}$, then $A_{\mathcal{B}}(t)$ can also have components orthogonal to $v_{\mathcal{B}}(t)$. In this case, $A_{\mathcal{B}}(t)$ may be interpreted as the vector sum of a drag and lift force. Furthermore, if we assume that $c_{a,x} < c_{a,yz}$, we ensure that the vehicle experiences minimum drag when $v_{\mathcal{B}}$ is aligned with the x-axis of $\mathcal{F}_{\mathcal{B}}$, and that the lift component of $A_{\mathcal{B}}$ points in the correct direction. Since the set $\mathcal{A}(V)$ corresponding to this choice of C_A defines an ellipsoid, we refer to the corresponding model as the *ellipsoidal aerodynamic model*, illustrated in Figure 2.2b.

2.2.6 State-Triggered Constraints

In this section, we introduce the most important contribution of this chapter: a continuous formulation of *state-triggered constraints* (STCs). The most common type of constraints seen in the optimal control literature are enforced over predetermined time intervals; we refer to such constraints as *temporally-scheduled constraints*. In contrast, an STC is enforced only when a state-dependent condition is satisfied, and emulates a constraint gated by an *if*-statement conditioned on the solution variables. Thus, an optimal control problem containing an STC determines its solution variables with a simultaneous understanding of how the constraint affects the optimization, and of how the optimization enables or disables the constraint. While the resulting continuous formulation is still non-convex, we have found that it is effectively handled by the successive convexification framework [62,83–85], as shown in §2.4.

Formal Definition of State-Triggered Constraints:

Formally, we define a *state-triggered constraint* as an equality constraint that is enforced conditionally according to the following logical statement

$$g(z) < 0 \;\Rightarrow\; c(z) = 0\,, \tag{2.15}$$

where $z \in \mathbb{R}^{n_z}$ represents the optimization variable of the parent problem, $g(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}$ is a piecewise continuously differentiable function called the *trigger function*, and $c(\cdot) : \mathbb{R}^{n_z} \to \mathbb{R}$ is a piecewise continuously differentiable function called the *constraint function*. Accordingly, we refer to $g(z) < 0$ as the *trigger condition*, and $c(z) = 0$ as the *constraint condition*.

The logical implication in (2.15) states that if the trigger condition is satisfied, then the constraint condition is enforced. By De Morgan's Law, (2.15) also implies satisfaction of the contrapositive. However, we emphasize that the satisfaction of the constraint condition does not imply satisfaction of the trigger condition (see §2.4).

Remark 1 *The constraint condition in* (2.15) *can be used to represent an inequality constraint by augmenting z with a non-negative slack variable and modifying $c(z)$ accordingly [19].*

Continuous State-Triggered Constraints:

The STC expressed in (2.15) represents a binary decision that is not readily incorporated into a continuous optimization framework. We address this issue by introducing *continuous state-triggered constraints* (cSTCs), which represent the logical implication in (2.15) using the auxiliary variable $\sigma \in \mathbb{R}_+$ and the system of equations

$$g(z) + \sigma \geq 0\,, \tag{2.16a}$$

$$\sigma \geq 0\,, \tag{2.16b}$$

$$\sigma \cdot c(z) = 0. \tag{2.16c}$$

The feasible set of triples $(g(z),\, c(z),\, \sigma) \in \mathbb{R}^3$ defined by (2.16) is shown on the left side of Figure 2.3, where the lower-left axes show the feasible set of the STC in (2.15). The formulation in (2.16) ensures that σ is strictly positive if the trigger condition is satisfied. Since $\sigma > 0$ implies that (2.16c) holds if and only if $c(z) = 0$, (2.15) and (2.16) are logically equivalent. Note that (2.16b) is not necessary when an equality constraint condition is used (i.e., when (2.16c) is an equality). However, we retain it in our formulation in order to accommodate inequality constraint conditions, which can be used in lieu of (2.16c) due to Remark 1.

Projected Formulation Using Linear Complementarity:

As illustrated in Figure 2.3, (2.16) does not admit a unique σ given z. To resolve this ambiguity, we augment (2.16a) and (2.16b) to form a complementarity constraint and obtain

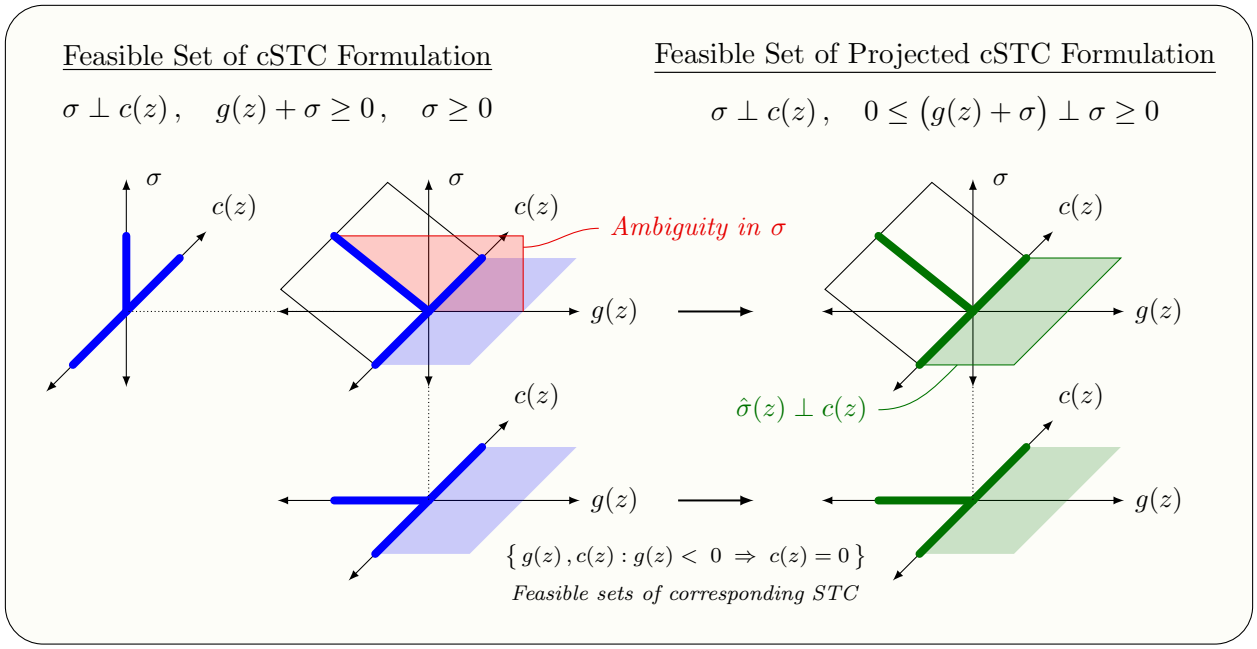


Figure 2.3: A geometrical interpretation of cSTCs. The red and blue sets on the left represent the feasible set of formulation (2.16), whereas the green sets on the right represent the feasible set of the projected formulation (2.17). The bottom two axes show that the feasible sets of both cSTC formulations comply with that of the STC in (2.15), despite the removal of the σ-ambiguity in the projected formulation.

the following projected cSTC formulation

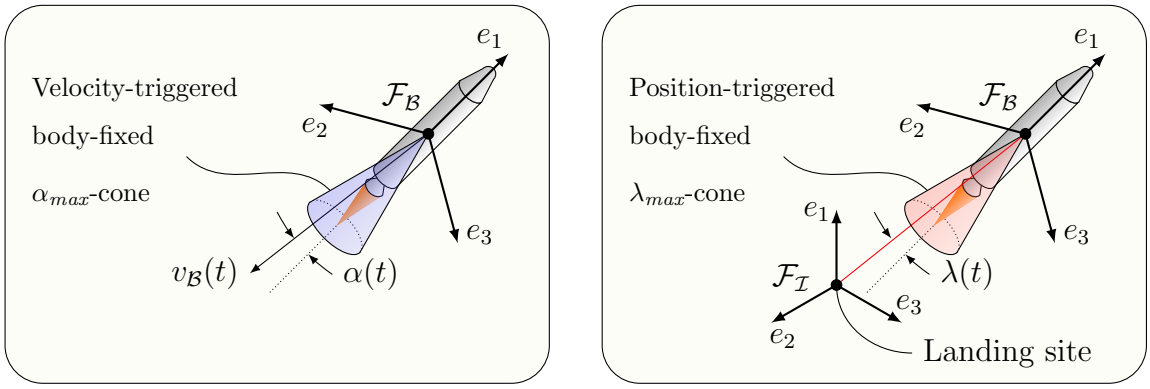
$$0 \leq \sigma \perp \big(g(z)+\sigma\big) \geq 0\,, \tag{2.17a}$$

$$\sigma \perp c(z)\,, \tag{2.17b}$$

where $a \perp b$ is used to denote $a \cdot b = 0$. For a given z, (2.17a) forms a linear complementarity problem (LCP) in σ [25]. This problem has a unique solution $\hat{\sigma}$ that varies continuously in $g(z)$ [25], and therefore in z. The analytical solution to the LCP is given by

$$\hat{\sigma}(z) \coloneqq -\min\big(g(z)\,, 0\big). \tag{2.18}$$

Substituting (2.18) into (2.17) guarantees satisfaction of (2.17a), resulting in the following



(a) Angle of Attack State-Triggered Constraint (b) Field of View State-Triggered Constraint

Figure 2.4: Example applications where STCs are used to (a) limit the angle of attack $\alpha(t)$ at large dynamic pressures, and (b) to impose a field of view constraint that limits the line of sight angle $\lambda(t)$ to the landing site.

equality constraint

$$h(z) \coloneqq -\min\left(g(z)\,,0\right)\cdot c(z) = 0\,, \tag{2.19}$$

where the negative sign is retained to allow (2.19) to be formulated as an inequality (see Remark 1). Thus, the ambiguity in σ is resolved by replacing the constraints in (2.16) with the logically equivalent constraint given in (2.19), which complies with the feasible set of the corresponding STC defined in (2.15). This can be seen in the two rightmost axes of Figure 2.3. Subsequent sections use the projected cSTC formulation in lieu of the original one.

Example Application:

We now present an example powered descent constraint whose formulation within a continuous optimization framework is enabled by the state-triggered constraints introduced in §2.2.6.Consider the problem of limiting the aerodynamic loads on a vehicle during a powered descent maneuver. On ascent, aerodynamic loads are often limited by imposing what are known as q-α limits, where q refers to dynamic pressure, and α refers to angle of attack.

These constraints are typically valid only for small angles of attack, where aerodynamic loads are relatively easy to model. Unlike an ascent trajectory, a powered descent maneuver can exhibit a wider range of angles of attack, possibly exceeding 90° in cases where "hopping" maneuvers are permitted. Measures must therefore be taken to ensure that the vehicle does not operate in flight regimes with large model uncertainty. Specifically, we propose a simplified q-α limit that enforces an angle of attack constraint *only* at high dynamic pressures. Formally, we express this constraint using the following STC

$$\|v_{\mathcal{B}}(t)\|_2 > V_\alpha \;\Rightarrow\; -e_1 \cdot v_{\mathcal{B}}(t) \geq \cos\alpha_{max} \|v_{\mathcal{B}}(t)\|_2 \,, \tag{2.20}$$

where $V_\alpha \in \mathbb{R}_{++}$ is a speed above which the angle of attack is limited to $\alpha(t) \in [0\,, \alpha_{max}]$. This STC is illustrated in Figure 2.4a. Noting Remark 1, we convert the STC in (2.20) into the following cSTC

$$h_\alpha\big(v_{\mathcal{I}}(t), q_{\mathcal{B}\leftarrow\mathcal{I}}(t)\big) \coloneqq -\min\Big(g_\alpha\big(v_{\mathcal{I}}(t)\big)\,, 0\Big) \cdot c_\alpha\big(v_{\mathcal{I}}(t), q_{\mathcal{B}\leftarrow\mathcal{I}}(t)\big) \leq 0\,, \tag{2.21a}$$

$$g_\alpha\big(v_{\mathcal{I}}(t)\big) \coloneqq V_\alpha - \|v_{\mathcal{I}}(t)\|_2\,, \tag{2.21b}$$

$$c_\alpha\big(v_{\mathcal{I}}(t), q_{\mathcal{B}\leftarrow\mathcal{I}}(t)\big) \coloneqq \cos\alpha_{max} \|v_{\mathcal{I}}(t)\|_2 + e_1 \cdot C_{\mathcal{B}\leftarrow\mathcal{I}}\big(q_{\mathcal{B}\leftarrow\mathcal{I}}(t)\big) v_{\mathcal{I}}(t). \tag{2.21c}$$

Remark 2 *A range-triggered field of view constraint imposed between a body-fixed downward-looking camera and the landing site may be obtained by replacing $v_{\mathcal{I}}(t)$, V_α, and α_{max} in* (2.21) *with $r_{\mathcal{I}}(t)$, R_{fov}, and λ_{max}, respectively (see Figure 2.4b). Such a constraint limits the line of sight angle to $\lambda(t) \in [0\,, \lambda_{max}]$ when the vehicle is at distances greater than R_{fov} away from the landing site.*

To conclude this section, we briefly discuss the shortcomings of three non-combinatorial alternatives to the cSTC proposed in (2.21):

Alternative 1: Consider a naive temporally-scheduled implementation, in which the problem is first solved without (2.20), and whose solution is used to determine the set of times $\mathcal{T}$ over which the trigger condition in (2.20) is satisfied. The problem is then solved a second

time with the constraint condition enforced over $t \in \mathcal{T}$. This new solution now satisfies the constraint condition over $t \in \mathcal{T}$, but does not necessarily satisfy the trigger condition for all $t \in \mathcal{T}$. In fact, this solution may satisfy the trigger condition for times where $t \notin \mathcal{T}$, thus necessitating a redefinition of $\mathcal{T}$. Unlike cSTCs, this approach does not convey the relationship between the trigger and constraint conditions to the optimization, thus allowing this situation to persist.

Alternative 2: Next, consider an implementation that replaces the STC in (2.20) with $-e_1 \cdot v_{\mathcal{B}}(t) / \left\| v_{\mathcal{B}}(t) \right\|_2 \geq \cos f_\alpha(t)$, where $f_\alpha(t) \coloneqq f_\alpha\big(\left\| v_{\mathcal{B}}(t) \right\|_2 \big)$ is a nonlinear scalar valued function that relates the maximum allowable angle of attack to the speed of the vehicle. This approach has two disadvantages: (i) it is less intuitive since the relationship between α and q is embedded in $f_\alpha(t)$, and (ii) obtaining a satisfactory $f_\alpha(t)$ with proper numerical scaling may be difficult.

Alternative 3: Lastly, consider an implementation using two phases: the first with an angle of attack constraint and no velocity constraint, and the second with a velocity constraint and no angle of attack constraint. Further, assume that the terminal condition of the first phase is equated to the initial condition of the second, and that both phases are solved simultaneously. Such a multi-phase optimization approach ensures satisfaction of (2.20), and is well suited for applications where the temporal ordering and quantity of phases are known a priori. However, since the formulation presupposes a specific phase structure, this approach is not capable of introducing additional phases. In contrast, cSTCs can do so in order to achieve feasibility or improve optimality.

2.2.7 Non-Convex Problem Statement

We now summarize the problem developed throughout this section. We assume a minimum-fuel objective function, but note that other objective functions can be readily used (e.g., a minimum-time problem would minimize t_b). The non-convex generalized powered descent guidance problem is summarized in Problem 1.

2.3 Convex Formulation

The successive convexification algorithm described in this section is designed to solve Problem 1 such that the converged continuous-time solution (i) *exactly* satisfies the original nonlinear dynamics, (ii) *approximates* the state and control constraints by enforcing them only at a finite number of temporal nodes[1], and (iii) *conservatively approximates* the optimality and feasibility of the problem by using a finite-dimensional representation of the infinite-dimensional control signal. The proposed algorithm works by iteratively solving a sequence of subproblems until a converged solution is attained. Each iteration consists of two steps: a *propagation step* responsible for obtaining a subproblem, followed by a *solve step* responsible for solving said subproblem to full optimality. Each subproblem is a convex approximation of Problem 1, and each solve step results in a state and control trajectory, or *iterate*. The propagation step in the first iteration is computed using a user-defined initialization trajectory, whereas subsequent iterations perform said approximation about the trajectory obtained by the previous iteration's solve step. Since the solve step is executed using well understood algorithms (e.g., IPMs [73]), this section primarily focuses on the propagation step.

This section is organized as follows. In §2.3.1, we outline a procedure to convert a *free-final-time nonlinear continuous-time* optimal control problem into a general implementable *fixed-final-time linear-time-varying discrete-time* convex parameter optimization subproblem [83]. In §2.3.2, we specialize the general subproblem to Problem 1. Lastly, in §2.3.3, we describe two initialization strategies, and conclude by summarizing the proposed algorithm.

[1]Common temporal discretization methods ensure that convex control constraints are satisfied between temporal nodes when the discrete-time control inputs are feasible. However, the same cannot be said for convex state constraints or non-convex constraints. There are methods that ensure inter-node feasibility using conservative constraints (e.g., [6]), but we deem these matters to be beyond the scope of this chapter.

2.3.1 Formulation of General Convex Subproblem

This section explains the successive convexification procedure using the following general optimal control problem

$$\underset{t_c,\, t_b,\, u(t)}{\text{minimize}} \;\; J\big(t_b\,, x(t_f)\big), \tag{2.22a}$$

$$\text{s.t.} \quad \dot{x}(t) = f\big(x(t)\,, u(t)\big)\,, \forall\, t \in [t_{ig}\,, t_f]\,, \tag{2.22b}$$

$$h_i\big(z(t)\big) = 0\,, \qquad \forall\, t \in [t_{ig}\,, t_f]\,,\ \forall\, i \in \mathcal{I}_{cvx} \cup \mathcal{I}_{ncvx} \cup \mathcal{I}_{stc}, \tag{2.22c}$$

where $x(t) \in \mathbb{R}^{n_x}$ and $u(t) \in \mathbb{R}^{n_u}$ denote respectively the state and control vectors, $z(t) \coloneqq \big[t_c \;\; t_b \;\; x^T(t) \;\; u^T(t)\big]^T \in \mathbb{R}^{n_z}$ (see §2.2.2 for definitions of t_c and t_b), $J : \mathbb{R}_{++} \times \mathbb{R}^{n_x} \to \mathbb{R}$ is a Mayer objective function [10] that is convex in its arguments, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ represents the continuous-time nonlinear dynamics, and $h_i : \mathbb{R}^{n_z} \to \mathbb{R}$ represent equality constraints imposed on the trajectory. The equality constraints in (2.22c) are assumed to be convex for $i \in \mathcal{I}_{cvx} \coloneqq \{1, \ldots, n_{cvx}\}$, and non-convex for $i \in \mathcal{I}_{ncvx} \coloneqq \{n_{cvx}+1, \ldots, n_{cvx}+n_{ncvx}\}$ and $i \in \mathcal{I}_{stc} \coloneqq \{n_{cvx}+n_{ncvx}+1, \ldots, n_{cvx}+n_{ncvx}+n_{stc}\}$. The functions J and f are assumed to be continuously differentiable, whereas each h_i is assumed to be at least once differentiable almost everywhere.

Normalization:

The normalization step converts the *free-final-time* nonlinear continuous-time optimal control problem in (2.22) into an equivalent *fixed-final-time* nonlinear continuous-time problem. This is achieved by temporally normalizing the burn phase from $t \in [t_{ig}\,, t_f]$ to $\tau \in [0\,, 1]$, where τ is the normalized burn phase time. Using the chain rule, the nonlinear dynamics in (2.22b) can be rewritten as

$$x'(t) \coloneqq \frac{d}{d\tau}x(t) = \frac{dt}{d\tau}\frac{d}{dt}x(t) = \left(\frac{dt}{d\tau}\right)\dot{x}(t). \tag{2.23}$$

Defining the *dilation factor* $s \coloneqq dt/d\tau \in \mathbb{R}_{++}$ and replacing $\dot{x}(t)$ with the right-hand side of (2.22b), the temporally-normalized dynamics are expressed as

$$x'(\tau) = F\big(x(\tau)\,, u(\tau)\,, s\big) \coloneqq s \cdot f\big(x(\tau)\,, u(\tau)\big). \tag{2.24}$$

Since $\tau \in [0\,,1]$, it follows that $s = t_b$. Thus, the temporal normalization of (2.22a) and (2.22c) is achieved by replacing the first argument of the cost function with s, and all subsequent t_f and t arguments with 1 and τ, respectively.

Linearization:

The linearization step converts the fixed-final-time *nonlinear* continuous-time problem obtained in the previous step into a fixed-final-time *linear-time-varying* continuous-time problem. By approximating non-convexities to first-order, the linearization step guarantees convexity of the subproblem.

Dynamics: The right-hand side of (2.24) is approximated by a first-order Taylor series, evaluated about a reference trajectory denoted by $\bar{z}(\tau) \coloneqq \left[\bar{t}_c \;\; \bar{s} \;\; \bar{x}^T(\tau) \;\; \bar{u}^T(\tau)\right]^T$ for all $\tau \in [0\,,1]$. The resulting linear-time-varying dynamics are given by

$$x'(\tau) \approx A(\tau)x(\tau) + B(\tau)u(\tau) + S(\tau)s + w(\tau), \tag{2.25a}$$

$$A(\tau) \coloneqq \left.\frac{\partial F}{\partial x}\right|_{\bar{z}(\tau)}, \tag{2.25b}$$

$$B(\tau) \coloneqq \left.\frac{\partial F}{\partial u}\right|_{\bar{z}(\tau)}, \tag{2.25c}$$

$$S(\tau) \coloneqq \left.\frac{\partial F}{\partial s}\right|_{\bar{z}(\tau)}, \tag{2.25d}$$

$$w(\tau) \coloneqq -A(\tau)\bar{x}(\tau) - B(\tau)\bar{u}(\tau). \tag{2.25e}$$

Non-Convex State and Control Constraints: Using the assumption that the functions $h_i(\cdot)$ are at least once differentiable almost everywhere, we define $\delta z(\tau) \coloneqq z(\tau) - \bar{z}(\tau)$ and approximate the constraints for each $i \in \mathcal{I}_{ncvx}$ in (2.22c) using a first-order Taylor series:

$$h_i\big(z(\tau)\big) \approx h_i\big(\bar{z}(\tau)\big) + \left.\frac{\partial h_i}{\partial z}\right|_{\bar{z}(\tau)} \delta z(\tau)\,. \tag{2.26}$$

Continuous State-Triggered Constraints: Similarly, since the trigger and constraint functions of each cSTC are assumed to be piecewise continuously differentiable, it follows that each h_i is at least once differentiable almost everywhere for all $i \in \mathcal{I}_{stc}$. However, due to the $\min(\cdot)$ function in (2.19), the partial $\partial h_i/\partial z$ is not well defined when $g_i(\cdot) = 0$. To ensure that the

constraint condition is not enforced when $g_i\big(\bar{z}(\tau)\big) = 0$, we define $\partial h_i/\partial z$ to hold the same value as when $g_i\big(\bar{z}(\tau)\big) > 0$. Thus, the approximation is given as follows:

$$h_i\big(z(\tau)\big) \approx \begin{cases} h_i\big(\bar{z}(\tau)\big) + \dfrac{\partial h_i}{\partial z}\Big|_{\bar{z}(\tau)} \delta z(\tau)\,, & \text{if } g_i\big(\bar{z}(\tau)\big) < 0\,, \\ 0\,, & \text{otherwise.} \end{cases} \tag{2.27}$$

Discretization:

The discretization step converts the fixed-final-time linear-time-varying *continuous-time* problem obtained in the previous section into a fixed-final-time linear-time-varying *discrete-time* parameter optimization problem. This step is critical in ensuring that the converged solution *exactly* adheres to the prescribed *nonlinear* dynamics. We begin by introducing K evenly spaced temporal nodes that divide the burn phase into $K-1$ subintervals, and define the sets $\mathcal{K} \coloneqq \{1,\, 2,\, \ldots\,,\, K\}$ and $\bar{\mathcal{K}} \coloneqq \{1,\, 2,\, \ldots\,,\, K-1\}$. Each temporal node is associated with an index $k \in \mathcal{K}$, and corresponding normalized time $\tau_k = (k-1)/(K-1)$.

To proceed, the control signal must be projected from the infinite-dimensional space it inhabits to a finite-dimensional space suitable for numerical optimization. This can be done in numerous ways, including zero-order-hold (ZOH) and first-order-hold (FOH) interpolation. Alternatively, the state and control can be projected to a finite-dimensional space directly using pseudospectral methods [11, 35]. We have found that, compared to pseudospectral methods, ZOH and FOH interpolation yield sparsity patterns that noticeably decrease solve time. Our approach utilizes FOH interpolation because it (i) provides a noticeable increase in optimality when compared ZOH interpolation, and (ii) ensures that when the discrete-time control variables satisfy convex control constraints, the interpolated values follow suit. Formally, FOH interpolation represents the control signal over each subinterval $k \in \bar{\mathcal{K}}$ as

$$u(\tau) = \lambda_k^-(\tau) u_k + \lambda_k^+(\tau) u_{k+1}\,, \quad \forall \tau \in [\tau_k\,, \tau_{k+1}]\,, \tag{2.28a}$$

$$\lambda_k^-(\tau) \coloneqq \frac{\tau_{k+1} - \tau}{\tau_{k+1} - \tau_k}\,, \quad \lambda_k^+(\tau) \coloneqq \frac{\tau - \tau_k}{\tau_{k+1} - \tau_k}, \tag{2.28b}$$

where $u_k \coloneqq u(\tau_k)$. Substituting (2.28) into (2.25), we obtain the following for each subin-

terval $k \in \bar{\mathcal{K}}$:

$$x'(\tau) = A(\tau)x(\tau) + \lambda_k^-(\tau)B(\tau)u_k + \lambda_k^+(\tau)B(\tau)u_{k+1} + S(\tau)s + w(\tau)\,, \quad \forall\, \tau \in [\tau_k\,, \tau_{k+1}]. \tag{2.29}$$

The state transition matrix $\Phi_A\big(\xi\,, \tau_k\big)$ for $\xi \in [\tau_k\,, \tau_{k+1}]$ associated with (2.29) is given by

$$\Phi_A\big(\xi\,, \tau_k\big) = I_{n_x \times n_x} + \int_{\tau_k}^{\xi} A(\zeta)\, \Phi_A\big(\zeta\,, \tau_k\big) d\zeta\,. \tag{2.30}$$

Denoting the discrete-time state vectors by $x_k \coloneqq x(\tau_k)$, the inverse and transitive properties of $\Phi_A\big(\cdot\,, \cdot\big)$ [7] are used to obtain the following discrete-time solution to (2.29) for each $k \in \bar{\mathcal{K}}$:

$$x_{k+1} = A_k x_k + B_k^- u_k + B_k^+ u_{k+1} + S_k s + w_k, \tag{2.31a}$$

$$A_k \coloneqq \Phi_A\big(\tau_{k+1}\,, \tau_k\big), \tag{2.31b}$$

$$B_k^- \coloneqq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi_A^{-1}\big(\xi\,, \tau_k\big) B(\xi) \lambda_k^-(\xi) d\xi, \tag{2.31c}$$

$$B_k^+ \coloneqq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi_A^{-1}\big(\xi\,, \tau_k\big) B(\xi) \lambda_k^+(\xi) d\xi, \tag{2.31d}$$

$$S_k \coloneqq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi_A^{-1}\big(\xi\,, \tau_k\big) S(\xi) d\xi, \tag{2.31e}$$

$$w_k \coloneqq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi_A^{-1}\big(\xi\,, \tau_k\big) w(\xi) d\xi. \tag{2.31f}$$

In implementation, the previous iteration's solve step generates $\bar{t}_c$, $\bar{s}$, $\bar{x}_k \coloneqq \bar{x}(\tau_k)$, and $\bar{u}_k \coloneqq \bar{u}(\tau_k)$ for all $k \in \mathcal{K}$. The $\bar{u}_k$'s and (2.28) are used to obtain $\bar{u}(\tau)$ over $\tau \in [0\,, 1]$, and (2.24), (2.31b), and the integrands in (2.31c)-(2.31f) are computed simultaneously for each $k \in \bar{\mathcal{K}}$ using the intermediate quantity $\Phi_A\big(\xi\,, \tau_k\big)$. When the propagation reaches τ_{k+1}, the quantity in (2.31b) can be left multiplied against the integrands of (2.31c)-(2.31f) to obtain the final values of A_k, B_k^-, B_k^+, S_k, and w_k.

The integration of (2.24) is initialized with $\bar{x}_k$, and is analogous to a multiple shooting method. We have observed that this multiple shooting strategy improves the convergence

behavior of the algorithm by keeping $\bar{z}(\tau)$ closer to the path obtained by the constrained optimization problem. In contrast, since the dynamics are nonlinear and may be unstable, a single shooting method is more susceptible to poor initializations, since the dynamics have more time to evolve away from feasibility. Thus, the propagation and solve steps are designed to play complementary roles, whereby the former relates the discrete-time optimization problem to the continuous-time physics in the absence of constraints, and the latter helps reset the shooting method at more frequent temporal intervals while taking into account the constraints.

Remark 3 *The initial condition of the k^{th} subinterval depends only on $\bar{z}_k$ obtained by the solve step of the previous iteration. It does not depend on the propagation of the $(k-1)^{th}$ subinterval. Therefore, the propagation step can be parallelized such that all subintervals $k \in \bar{\mathcal{K}}$ are computed simultaneously, significantly reducing the computation time of the propagation step.*

The state and control constraints are enforced at the temporal nodes τ_k for all $k \in \mathcal{K}$. This discretization choice is adopted for simplicity, and does not guarantee the absence of inter-node state constraint and non-convex control constraint violations.

We conclude by noting that the number of temporal nodes and interpolation scheme do not affect the *accuracy* of the converged solution with respect to the nonlinear dynamics. Instead, these choices affect the *optimality* and, in extreme cases, the *feasibility* of the solution. Simply put, a coarser or less expressive interpolation results in a finite-dimensional control signal with fewer degrees of freedom. This yields a problem that is implicitly more constrained, and thus a solution that is generally less optimal.

Virtual Control & Trust Region Modifications:

The process outlined in the previous three sections results in an implementable convex parameter optimization subproblem that may suffer from *artificial infeasibility* and *artificial unboundedness*. These issues are addressed by the virtual control and trust region modifications, respectively [62, 83–85].

Artificial Infeasibility & Virtual Control: Consider linearizing and discretizing Problem 1 about a burn time $\bar{s} = 0$ (or some small value). From (2.25), (2.30), and (2.31), it follows that (2.31a) does not admit a solution for arbitrary $x_{k+1} \neq x_k$. This remains true even if Problem 1 admits a feasible solution, resulting in a condition we term *artificial infeasibility.* To mitigate artificial infeasibility, we augment (2.31a) with a virtual control term $\nu_k \in \mathbb{R}^{n_x}$ for all $k \in \bar{\mathcal{K}}$,

$$x_{k+1} = A_k x_k + B_k^- u_k + B_k^+ u_{k+1} + S_k s + w_k + \nu_k \,, \tag{2.32}$$

and add the following penalty term to (2.22a),

$$J_{vc}(\nu) \coloneqq w_\nu \sum_{k \in \bar{\mathcal{K}}} \|\nu_k\|_1 \,, \tag{2.33}$$

where $w_\nu \in \mathbb{R}_{++}$ is a large weight. The virtual control modification guarantees that each subproblem has a non-empty feasible set, and thus ensures that the convergence process is not obstructed. A 1-norm minimization is employed in (2.33) to encourage sparsity in the vectors ν_k. Upon successful convergence, the virtual control terms are zero, and (2.32) is equivalent to (2.31).

Artificial Unboundedness & Trust Region: The second issue that may arise is that of *artificial unboundedness*, which occurs when the *linearized* constraints permit the cost of a subproblem to be minimized indefinitely. This issue is mitigated by adding the following soft quadratic trust region to the cost function,

$$J_{tr}(\bar{z}\,, z) \coloneqq \sum_{k \in \mathcal{K}} \delta z_k^T \, W_{tr} \, \delta z_k \,, \tag{2.34}$$

where $\delta z_k \coloneqq \delta z(\tau_k)$ for all $k \in \mathcal{K}$, and $W_{tr} \in \mathbb{S}_{++}^{n_z}$ is a symmetric positive definite weighting matrix. The trust region modification also serves to ensure that the solve step does not venture excessively far from the reference trajectory used in the propagation step.

2.3.2 Specialized Implementable Convex Subproblem for 6-DoF Powered Descent Guidance

To specialize the subproblem developed in §2.3.1 to Problem 1, we define the objective function (2.22a) to be the minimum-fuel objective $J\big(s\,, x_K\big) \coloneqq -m_K$. We define $x(t) \coloneqq$

$\left[m(t)\ \ r_{\mathcal{I}}^T(t)\ \ v_{\mathcal{I}}^T(t)\ \ q_{\mathcal{B}\leftarrow\mathcal{I}}^T(t)\ \ \omega_{\mathcal{B}}^T(t)\right]^T$ and $u(t) \coloneqq T_{\mathcal{B}}(t)$, and concatenate (2.1)-(2.3) to form the corresponding dynamics. In accordance with (2.26), the thrust lower bound constraint in (2.8) is approximated using the following form for each $k \in \mathcal{K}$

$$h_{tlb,k} + H_{tlb,k}\,\delta z_k \leq 0. \tag{2.35}$$

Similarly, in accordance with (2.27), the cSTC in (2.21) is approximated using the following form for each $k \in \mathcal{K}$

$$h_{\alpha,k} + H_{\alpha,k}\,\delta z_k \leq 0. \tag{2.36}$$

Problem 2 presents a summary of the specialized convex parameter optimization subproblem used in our algorithm. This problem is primarily concerned with solving for the states x_k, controls u_k, and time dilation s associated with the *burn phase* of the trajectory. The only aspect of the *coast phase* optimized by Problem 2 is the coast time t_c, which in turn determines the ignition time position $r_{\mathcal{I},1}$ and velocity $v_{\mathcal{I},1}$ via (2.12) and (2.13).

2.3.3 Successive Convexification Algorithm

Initialization:

We consider two initialization approaches: *straight-line initialization* and *3-DoF initialization*. Both approaches assume $\bar{t}_c = 0$ and a user-specified initial guess for $\bar{s}$. The former is equivalent to assuming $t_{in} = t_{ig}$, whereas the latter is equivalent to guessing the burn time t_b (see Figure 2.1). The nature of the powered descent guidance problem is such that $\bar{s}$ can typically be guessed accurately as a function of distance to the landing site, initial velocity, and available thrust. In our experience, the proposed algorithm is able to handle a wide range of initialization values for $\bar{s}$, although poor guesses may lead to increased convergence time.

The straight-line initialization approach constructs the discrete-time state trajectory $\bar{x}_k$ by linearly interpolating the state at each temporal node between the ignition and final states. The control trajectory $\bar{u}_k$ is assumed to oppose the gravitational force at each temporal node.

The initialization assumes an initial attitude q_{id} and final mass m_{dry}, since these quantities are not known a priori. Formally, the state and control for each $k \in \mathcal{K}$ is computed as follows

$$\bar{x}_k = \left(\frac{K-k}{K-1}\right)\bar{x}_{ig} + \left(\frac{k-1}{K-1}\right)\bar{x}_f, \tag{2.37a}$$

$$\bar{u}_k = -\left(\frac{K-k}{K-1}\right)m_{ig}g_{\mathcal{I}} - \left(\frac{k-1}{K-1}\right)m_{dry}g_{\mathcal{I}}, \tag{2.37b}$$

where $\bar{x}_{ig} \coloneqq \begin{bmatrix} m_{ig} & r_{\mathcal{I},ig}^T & v_{\mathcal{I},ig}^T & q_{id}^T & 0^T \end{bmatrix}^T$, and $\bar{x}_f \coloneqq \begin{bmatrix} m_{dry} & 0^T - v_d e_1^T & q_{id}^T & 0^T \end{bmatrix}^T$. If a value other than q_{id} is assumed for the initial attitude, this approach can be improved by interpolating the quaternion states using Spherical Linear Interpolation (SLERP) [40].

The 3-DoF initialization approach constructs the state and control trajectories using a solution obtained from a convex 3-DoF guidance problem [5]. The 3-DoF problem is solved using the same number of temporal nodes, K, as in the 6-DoF problem. This problem may be solved once using a user-defined burn time, or in conjunction with a line-search that optimizes the burn time, and thus generates $\bar{s}$. The mass, position, and velocity components of $\bar{x}_k$ and the controls $\bar{u}_k$ are obtained directly from the 3-DoF solution. The attitude is computed such that the vertical axis of the vehicle is aligned with $\bar{u}_k$, and the angular velocity is obtained by inverting (2.3a).

Unsurprisingly, the initialization approach can have a significant impact on the converged solution attained by the algorithm. In our work, we have found that neither approach offers a clear and consistent advantage over the other, (see §2.4.4). We ultimately regard the initialization approach as a design choice.

Algorithm:

The algorithm is initialized using one of the two approaches discussed in the previous section. For each iteration, the algorithm performs a propagation step to compute A_k, B_k^+, B_k^-, S_k, and w_k for all $k \in \bar{\mathcal{K}}$, followed by a solve step that solves the convex second-order cone programming subproblem summarized in Problem 2. The process terminates when $J_{vc}(\nu) < \epsilon_{vc}$ and $J_{tr}(z\,, \bar{z}) < \epsilon_{tr}$, where ϵ_{vc}, $\epsilon_{tr} \in \mathbb{R}_{++}$ are user-specified convergence

tolerances. Satisfaction of the convergence criteria ensures that the attained solution eliminates the first order terms δz_k generated by the approximation without using virtual control. If an iterate satisfies the convergence criteria, then feasibility of the corresponding subproblem implies that the iterate exactly satisfies the nonlinear dynamics of Problem 1 for all $t \in [t_{ig}, t_f]$, and satisfies the state and control constraints of Problem 1 at each temporal node. However, failure to converge does not necessarily imply that Problem 1 is infeasible. A summary of the algorithm is provided in Algorithm 1.

We note that prior to convergence, the iterates may not be feasible with respect to Problem 1 due to the linearization used in the propagation step. This statement holds true even though each convex subproblem is designed to be feasible through the use of virtual control.

Finally, we highlight that no convergence guarantees are presented in this work. However, we have found that Algorithm 1 works well in practice, and note its similarity to [62], which does guarantee convergence to a local optima of the original problem *when* the converged solution requires no virtual control.

2.4 Numerical Results

In this section, we present simulation results that demonstrate the proposed successive convexification algorithm, while highlighting the principal contributions of this work. In §2.4.1, §2.4.2, and §2.4.3 we present case studies that respectively illustrate the effects of the aerodynamic models introduced in §2.2.5, the state-triggered constraints introduced in §2.2.6, and the free-ignition-time modification introduced in §2.2.4. In §2.4.4, we provide performance and timing results. To present the results, we introduce the problem feature labels given in Table 2.1.

The simulations are designed around a notional non-dimensionalized scenario with time, length, and mass units U_T, U_L, and U_M. Each scenario is defined by the problem parameters in Tables 2.2 and the initial position and velocity vectors defined in each subsequent subsection. For the sake of illustration, the initial conditions used in the case studies in §2.4.1- §2.4.3

define in-plane maneuvers, whereas those used to generate the timing results in §2.4.4 define more computationally intensive out-of-plane maneuvers.

Table 2.1: Generalized Powered Descent Guidance Problem Features

Feature	Section	Description
(B)	§2.2.3	Baseline problem setup with no aerodynamics and straight-line initialization.
(FI)	§2.2.4	Includes the free-ignition-time modification shown in Figure 2.1, where initial position and velocity are restricted to a free fall trajectory prior to engine ignition.
(SA)	§2.2.5	Includes the spherical aerodynamic model from Figure 2.2a.
(EA)	§2.2.5	Includes the ellipsoidal aerodynamic model from Figure 2.2b.
(ST)	§2.2.6	Includes the state-triggered constraint from Figure 2.4a, and introduced in (2.20).
(3I)	§2.3.3	Uses the 3-DoF initialization in lieu of straight-line initialization. Straight-line initialization is implied in the absence of this feature.

2.4.1 Aerodynamic Models Case Study

In this case study, we solve three otherwise identical powered descent guidance problems, assuming no aerodynamic effects, a spherical aerodynamic model, and an ellipsoidal aerody-

Table 2.2: Problem Parameters

Param.	Value	Units	Param.	Value	Units
$g_{\mathcal{I}}$	$-e_1$	U_L/U_T^2	V_α	2.0	U_L/U_T
ρ	1.0	U_M/U_L^3	α_{max}	3.0	$^\circ$
$J_{\mathcal{B}}$	$0.168\,\mathbf{diag}\left(\left[0.002\ 1\ 1\right]\right)$	$U_M \cdot U_L^2$	m_{dry}	1.0	U_M
P_{amb}	0.0	$U_M/U_T^2/U_L$	m_{ig}	2.0	U_M
A_{noz}	0.0	U_L^2	$r_{\mathcal{I},K}$	$0_{3\times1}$	U_L
$r_{cp,\mathcal{B}}$	$0.05 \cdot e_1$	U_L	$v_{\mathcal{I},K}$	$-0.1 \cdot e_1$	U_L/U_T
$r_{T,\mathcal{B}}$	$-0.25 \cdot e_1$	U_L	$\omega_{\mathcal{B},1}$, $\omega_{\mathcal{B},K}$	$0_{3\times1}$	$^\circ/U_T$
I_{sp}	30.0	U_T	$q_{\mathcal{B}\leftarrow\mathcal{I},K}$	q_{id}	-
θ_{max}	90.0	$^\circ$	K	20	-
ω_{max}	28.6	$^\circ/U_T$	w_ν	1e+4	-
γ_{gs}	75.0	$^\circ$	W_{tr}	0.5	-
δ_{max}	20.0	$^\circ$	ϵ_{vc}	1e-8	-
T_{min}	1.5	$U_M \cdot U_L/U_T^2$	ϵ_{tr}	5e-4	-
T_{max}	6.5	$U_M \cdot U_L/U_T^2$	$\bar{s}$	5.0	-

namic model. These problems are labeled using (B), (B)+(SA) and (B)+(EA), respectively. In each problem, the vehicle begins above and east of the landing pad, traveling west at a shallow flight path angle. These initial conditions are given by $r_{\mathcal{I}}(t_{in}) = [4.33\ 3.5\ 0.0]^T\ U_L$ and $v_{\mathcal{I}}(t_{in}) = [-0.5\ -2.5\ 0.0]^T\ U_L/U_T$. To land successfully, the vehicle must shed significant horizontal momentum while ensuring an upright final attitude.

Figure 2.5 shows the converged trajectory for each of the three cases. To reduce clutter, only 11 of the 20 temporal nodes are shown. The insets (i) and (ii) represent free body diagrams of the forces acting on the vehicle at the same two temporal nodes for each case.

Figure 2.5a shows the trajectory for the case without aerodynamic effects. From the insets, it is clear that the maneuver resembles a reverse gravity turn, with the thrust pointed nearly anti-parallel to the velocity vector.

Figure 2.5b shows the trajectory for the case with the spherical aerodynamic model. The corresponding insets show aerodynamic forces that are anti-parallel to the velocity vector, and may therefore be interpreted as drag forces. The thrust directions, however, remain consistent with the no-aerodynamic case. Thus, the trajectory in Figure 2.5b may be interpreted as the atmospheric counterpart to the reverse gravity turn observed in case (B).

Figure 2.5c shows the trajectory for the case with the ellipsoidal aerodynamic model. In this case, the aerodynamic force vectors are seen to have a component orthogonal to the velocity vector, and may therefore be interpreted as a composition of a drag and lift force (recall Figure 2.2). The vehicle is observed to exploit the lift force to bend the trajectory downwards by adjusting its attitude to control the angle of attack. As a consequence, the thrust is no longer aligned with the velocity as it was in the previous two cases, and is instead gimbaled such that the vehicle remains trimmed at an angle of attack that applies the lift force in a desirable direction.

2.4.2 State-Triggered Constraints Case Study

This case study highlights the effects of the q-α state-triggered constraint introduced in (2.20) on a landing scenario with non-negligible atmospheric effects. We consider the cases (B)+(EA) and (B)+(EA)+(ST), with initial position and velocity vectors $r_{\mathcal{I}}(t_{in}) = [5.33\ 4.5\ 0.0]^T\ U_L$ and $v_{\mathcal{I}}(t_{in}) = [-0.5\ \ -2.5\ 0.0]^T\ U_L/U_T$, respectively.

The resulting trajectories are shown in Figures 2.6a-2.6b, with the corresponding speed and angle of attack profiles provided in Figures 2.6c-2.6d. In Figures 2.6a-2.6b, inset (i) shows the initial condition where the speed is greater than V_α, while inset (ii) shows the last temporal node at which the STC is active in case (B)+(EA)+(ST). The lift-to-drag ratios displayed in inset (i) of Figures 2.6a-2.6b indicate that the aerodynamic loading in case (B)+(EA)+(ST) is reduced when the compared to case (B)+(EA) due to the inclusion

of the q-α constraint. The same effect is apparent to a lesser extent in inset (ii), where the STCs trigger condition is still satisfied for case (B)+(EA)+(ST). In Figure 2.6b, the temporal nodes after inset (ii) clearly exhibit larger angles of attack, indicating that the vehicle's speed has dropped below the trigger limit and that the q-α constraint has been disabled.

Figures 2.6c-2.6d show the corresponding speed and angle of attack time histories. Since case (B)+(EA) does not implement the q-α STC, the angle of attack is seen to violate the 3.0° limit when the speed is greater than 2.0 U_L/U_T. In contrast, the angle of attack in case (B)+(EA)+(ST) remains below the prescribed 3.0° limit until $t \approx 2.0\ U_T$, where the speed drops below the prescribed 2.0 U_L/U_T speed limit.

Lastly, notice that Figure 2.6d shows that case (B)+(EA)+(ST) rides the speed constraint for $t \in [2.0\,, 3.0]$. Although the trajectory may gain optimality by traveling faster over this time interval, the angle of attack is above the specified 3.0° limit over this time interval. Consequently, we observe the enforcement of the *contrapositive* of (2.20) (i.e., an angle of attack greater than 3.0° implies that the speed must be less than 2.0 U_L/U_T). Further, certain scenarios may allow the angle of attack to drop below the prescribed limit towards the end of the trajectory, thereby allowing the speed to increase above the STC trigger limit. Such an eventuality would cause the optimization algorithm to add a new constrained phase *without a priori input*, per the discussion in Alternative 3 in §2.2.6.

2.4.3 Free-Ignition-Time Case Study

The third case study considers how the free-ignition time modification affects the trajectory by comparing the cases (B)+(EA) and (B)+(EA)+(FI). The initial position and velocity are identical to those used in §2.4.2. Figure 2.7 depicts the converged trajectories of both cases. Inset (i) on each figure corresponds to the ignition time epoch t_{ig}, while inset (ii) corresponds to a temporal node in the middle of the maneuver. Case (B)+(EA)+(FI) selects a coast time of $t_c = 0.96\ U_T$ and as a result observes a reduction in both burn time and fuel cost (see §2.4.4).

2.4.4 Trajectory and Computational Performance

The trajectory and computational performance data presented in this section were generated for the cases listed in the leftmost column of Table 2.3. The results were obtained by executing a batch of 10 runs for each case using the initial position and velocity vectors $r_{\mathcal{I}}(t_{in}) = [5.33 \ 4.5 \ 0.0]^T \ U_L$ and $v_{\mathcal{I}}(t_{in}) = [-0.5 \ \ -2.5 \ 0.25]^T \ U_L/U_T$. These initial conditions generate three-dimensional out-of-plane trajectories that render the problem less sparse and thus more computationally intensive than their their planar counterparts presented in §2.4.1-§2.4.3.

Trajectory performance metrics are given in Table 2.3. For each case, the entries represent the median values of the metrics generated in the batch. Due to the determinism of the proposed algorithm, the standard deviations of the trajectory performance metrics was zero.

To validate each solution against the nonlinear dynamics, we integrate the nonlinear equations of motion using a piecewise linear interpolation of the controls, given in (2.28). The errors between the integrated trajectory and the discrete states generated by the optimization process are then used as a measure of the solution's feasibility. The position and attitude errors are defined as $e_{pos} \coloneqq \max_{k \in K} \ \|r_{\mathcal{I}}(t_k) - r_{\mathcal{I},k}\|_2$ and $e_{att} \coloneqq \max_{k \in K} \ 2\cos^{-1}\left[q^*_{\mathcal{B}\leftarrow\mathcal{I}}(t_k) \otimes q_{\mathcal{B}\leftarrow\mathcal{I},k}\right]$, where $q_{\mathcal{B}\leftarrow\mathcal{I},k}$ and $r_{\mathcal{I},k}$ are the discrete solution values, while $q_{\mathcal{B}\leftarrow\mathcal{I}}(t_k)$ and $r_{\mathcal{I}}(t_k)$ are the corresponding integrated values.

The computational performance results for the solve step is given in Table 2.4, and were generated on a 2014 MacBook Pro with a 2.2 GHz Intel Core i7 processor and 16 GB of RAM. The propagation step was implemented in C++ using the Eigen matrix library [38], and was omitted from Table 2.4 since the maximum propagation time per run was on the order of 10 milliseconds. The solve time results were generated in MATLAB using ECOS [27] and CVX [37]. For each case, the solve times were obtained by totaling the fifth argument reported by the `cvx_toc` function over all ECOS calls in a run, and computing the statistics over the entire batch.

We conclude with three final observations. First, in the cases presented, the inclusion

Table 2.3: Trajectory Performance Results for Combinations of Problem Features

Case	Burn Time (U_T)	Fuel Used (%)	Succ. Iters.	e_{pos} (U_L)	e_{att} (°)
(B)	4.42	3.17	4	1.1e-3	8.0e-2
(B) + (SA)	4.27	3.72	4	2.2e-2	1.3e-1
(B) + (EA)	4.67	3.55	4	7.8e-2	1.0e-0
(B) + (EA) + (ST)	4.06	3.40	7	3.3e-3	3.7e-1
(B) + (ST) + (FI)	3.10	3.03	7	3.2e-5	1.1e-2
(B) + (EA) + (FI)	2.56	3.13	5	1.4e-2	$\leq$ 1.0e-6
(B) + (EA) + (3I)	4.89	3.61	4	2.7e-2	3.2e-1
(B) + (EA) + (3I) + (FI)	4.41	3.58	4	2.2e-2	1.1e-1
(B) + (EA) + (ST) + (FI)	2.69	3.21	7	2.1e-3	$\leq$ 1.0e-6
(B) + (EA) + (ST) + (FI) + (3I)	3.21	3.27	7	8.0e-3	$\leq$ 1.0e-6

of feature (ST) yielded a decrease in both burn time and fuel consumption. This result is counter intuitive since one would expect the optimal cost of a problem to remain the same or increase when additional constraints are added. Since Problem 1 may have multiple local minima, we posit that the inclusion of the q-α state-triggered constraint forced the iterative solution process towards a more optimal local minima. On the other hand, the inclusion of feature (FI) reduced the fuel cost. This result agrees with intuition since the free-ignition-time modification effectively adds a degree of freedom to the problem.

Second, we note that the 3-DoF initialization approach (i.e., the inclusion of feature (3I)) did not yield a clear improvement in optimality or computational performance. In fact, cases (B)+(EA)+(3I), (B)+(EA)+(3I)+(FI), and (B)+(EA)+(ST)+(FI)+(3I) all resulted in less optimal trajectories, while cases (B)+(EA)+(3I) and (B)+(EA)+(ST) +(FI)+(3I) also increased the solve time. We conclude that the straight-line initialization approach

Table 2.4: Computational Performance Results Averaged Over 10 Runs

Case	Solve Time [s]			
	Min	Max	Median	Std
(B)	0.164	0.287	0.164	0.007
(B) + (SA)	0.170	0.210	0.174	0.012
(B) + (EA)	0.271	0.312	0.276	0.012
(B) + (EA) + (ST)	0.496	0.565	0.504	0.020
(B) + (ST) + (FI)	0.643	0.682	0.654	0.011
(B) + (EA) + (FI)	0.233	0.263	0.241	0.008
(B) + (EA) + (3I)	0.299	0.329	0.312	0.011
(B) + (EA) + (3I) + (FI)	0.224	0.253	0.229	0.008
(B) + (EA) + (ST) + (FI)	0.494	0.537	0.505	0.012
(B) + (EA) + (ST) + (FI) + (3I)	0.602	0.632	0.618	0.009

initializes the algorithm in a more favorable region of attraction, but stress that this may not be the case for different scenarios.

Lastly, the timing results presented in Table 2.4 show a maximum solve time of 0.7 seconds and a standard deviation on the order of milliseconds and were all obtained using the same algorithm parameters (e.g., w_ν, W_{tr}, K). We argue that these results are an important step in demonstrating the efficacy of the successive convexification methodology for real time autonomous applications. Ultimately, results obtained on representative flight hardware will be crucial in accurately assessing the viability of the proposed methodology for on-board computation in real-world applications.

2.5 Concluding Remarks

This chapter presents a real-time successive convexification algorithm for a generalized free-final-time 6-DoF powered descent guidance problem, and introduces three primary contributions: (i) a free-ignition-time modification that allows the algorithm to determine when to begin the burn phase, (ii) an ellipsoidal aerodynamics model that provides a computationally tractable way to model lift and drag forces, and (iii) a continuous formulation for state-triggered constraints. Contribution (iii) allows continuous optimization problems to be formulated using conditionally enforced constraints, and was motivated by landing scenarios that necessitate velocity-triggered angle of attack and range-triggered line of sight constraints.

Three simulation case studies are presented, each illustrating one of the primary contributions of this work. The corresponding trajectory and computational performance results show that the proposed algorithm can successfully compute trajectories in under 0.7 seconds for the problem features considered. While additional work is required to provide convergence guarantees and to quantify the optimality of the computed trajectories, we argue that our results demonstrate the efficacy of the successive convexification approach for real-time powered descent guidance applications.

Problem 1 ***Non-Convex Optimal Control Problem***

Cost Function:

$$\underset{t_c,\, t_b,\, T_{\mathcal{B}}(t)}{\text{minimize}} \quad -m(t_f)$$
$$\text{s.t.} \quad t_c \in [0\,, t_{c,max}]$$

Boundary Conditions:

$$m(t_{ig}) = m_{ig} \qquad q_{\mathcal{B}\leftarrow\mathcal{I}}(t_f) = q_{id}$$
$$r_{\mathcal{I}}(t_{ig}) = p_{r,ig}(t_c) \qquad r_{\mathcal{I}}(t_f) = 0$$
$$v_{\mathcal{I}}(t_{ig}) = p_{v,ig}(t_c) \qquad v_{\mathcal{I}}(t_f) = -v_d e_1$$
$$\omega_{\mathcal{B}}(t_{ig}) = 0 \qquad \omega_{\mathcal{B}}(t_f) = 0$$

Dynamics:

$$\dot{m}(t) = -\alpha_{\dot{m}} \left\| T_{\mathcal{B}}(t) \right\|_2 - \beta_{\dot{m}}$$
$$\dot{r}_{\mathcal{I}}(t) = v_{\mathcal{I}}(t)$$
$$\dot{v}_{\mathcal{I}}(t) = \frac{1}{m(t)} C_{\mathcal{I}\leftarrow\mathcal{B}}(t)\big(T_{\mathcal{B}}(t) + A_{\mathcal{B}}(t)\big) + g_{\mathcal{I}}$$
$$\dot{q}_{\mathcal{B}\leftarrow\mathcal{I}}(t) = \frac{1}{2}\Omega\big(\omega_{\mathcal{B}}(t)\big) q_{\mathcal{B}\leftarrow\mathcal{I}}(t)$$
$$J_{\mathcal{B}}\dot{\omega}_{\mathcal{B}}(t) = r_{T,\mathcal{B}} \times T_{\mathcal{B}}(t) + r_{cp,\mathcal{B}} \times A_{\mathcal{B}}(t) - \omega_{\mathcal{B}}(t) \times J_{\mathcal{B}}\omega_{\mathcal{B}}(t)$$

State Constraints:

$$m_{dry} \leq m(t)$$
$$\tan\gamma_{gs} \left\| H_\gamma r_{\mathcal{I}}(t) \right\|_2 \leq e_1 \cdot r_{\mathcal{I}}(t)$$
$$\cos\theta_{max} \leq 1 - 2\left\| H_\theta q_{\mathcal{B}\leftarrow\mathcal{I}}(t) \right\|_2$$
$$\left\| \omega_{\mathcal{B}}(t) \right\|_2 \leq \omega_{max}$$

Control Constraints:

$$0 < T_{min} \leq \left\| T_{\mathcal{B}}(t) \right\|_2 \leq T_{max}$$
$$\cos\delta_{max} \left\| T_{\mathcal{B}}(t) \right\|_2 \leq e_3 \cdot T_{\mathcal{B}}(t)$$

State-Triggered Constraints:

$$h_\alpha\big(v_{\mathcal{I}}(t), q_{\mathcal{B}\leftarrow\mathcal{I}}(t)\big) \leq 0$$

Problem 2 ***Convex Parameter Optimization Subproblem***

Cost Function:

$$\underset{t_c,\, s,\, u_k,\, \nu_k}{\text{minimize}} \; -m_k + J_{vc}(\nu) + J_{tr}(\bar{z}\,,\, z)$$

$$\text{s.t.} \quad t_c \in [0\,,\, t_{c,max}]$$

Boundary Conditions:

$$\begin{aligned} m_1 &= m_{ig} & q_{\mathcal{B}\leftarrow\mathcal{I},K} &= q_{id} \\ r_{\mathcal{I},1} &= p_{r,ig}(t_c) & r_{\mathcal{I},K} &= 0 \\ v_{\mathcal{I},1} &= p_{v,ig}(t_c) & v_{\mathcal{I},K} &= -v_d e_1 \\ \omega_{\mathcal{B},1} &= 0 & \omega_{\mathcal{B},K} &= 0 \end{aligned}$$

Dynamics:

$$x_{k+1} = A_k x_k + B_k^- u_k + B_k^+ u_{k+1} + S_k s + w_k + \nu_k$$

State Constraints:

$$\begin{aligned} m_{dry} &\leq m_k \\ \tan\gamma_{gs} \left\| H_\gamma r_{\mathcal{I},k} \right\|_2 &\leq e_1 \cdot r_{\mathcal{I},k} \\ \cos\theta_{max} &\leq 1 - 2\left\| H_\theta q_{\mathcal{B}\leftarrow\mathcal{I},k} \right\|_2 \\ \left\| \omega_{\mathcal{B},k} \right\|_2 &\leq \omega_{max} \end{aligned}$$

Control Constraints:

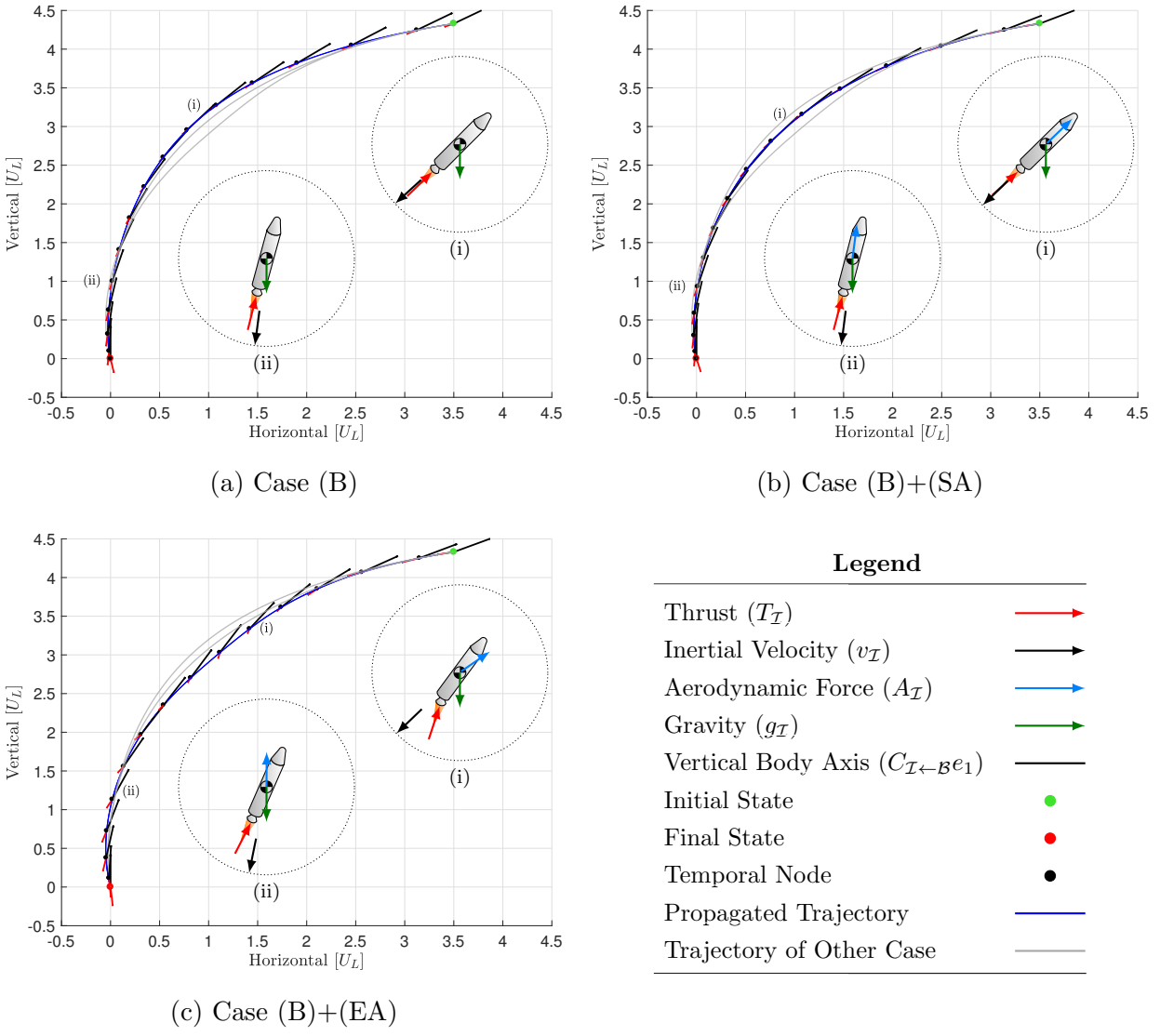
$$\begin{aligned} \left\| u_k \right\|_2 &\leq T_{max} \\ \cos\delta_{max} \left\| u_k \right\|_2 &\leq e_3 \cdot u_k \\ h_{tlb,k} + H_{tlb,k}\, \delta z_k &\leq 0 \end{aligned}$$

State-Triggered Constraints:

$$h_{\alpha,k} + H_{\alpha,k}\, \delta z_k \leq 0$$

Algorithm 1: ***6-DoF Powered Descent Guidance Algorithm***

Initialize $\{\bar{t}_c, \bar{s}, \bar{x}, \bar{u}\}$
while not converged **do**
    Compute $\{A_k, B_k^+, B_k^-, S_k, w_k\} \ \forall\, k \in \bar{\mathcal{K}}$ ▷ Propagation Step
    Solve Problem 2 to obtain $\{t_c, s, x, u, \nu\}$ ▷ Solve Step
    if $J_{vc}(\nu) < \epsilon_{vc}$ and $J_{tr}(z\,, \bar{z}) < \epsilon_{tr}$ **then** ▷ Exit Criteria
        converged
    end if
    $\{\bar{t}_c, \bar{s}, \bar{x}, \bar{u}\} \leftarrow \{t_c, s, x, u\}$
end while
return $\{t_c, s, x, u\}$

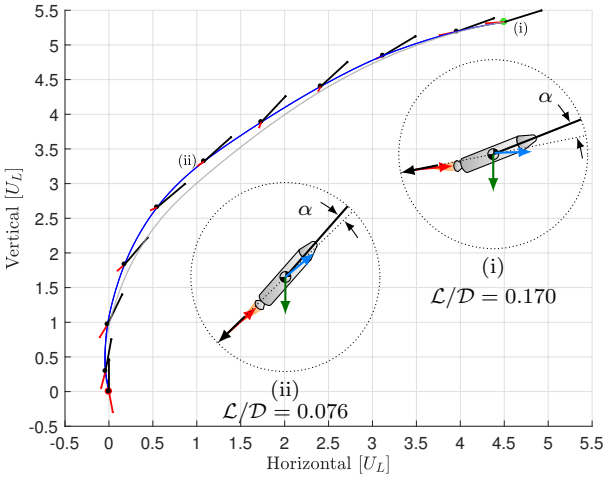


(a) Case (B)

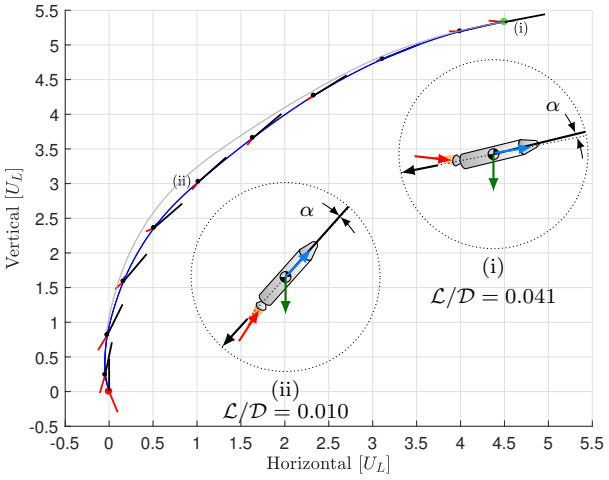
(b) Case (B)+(SA)

(c) Case (B)+(EA)

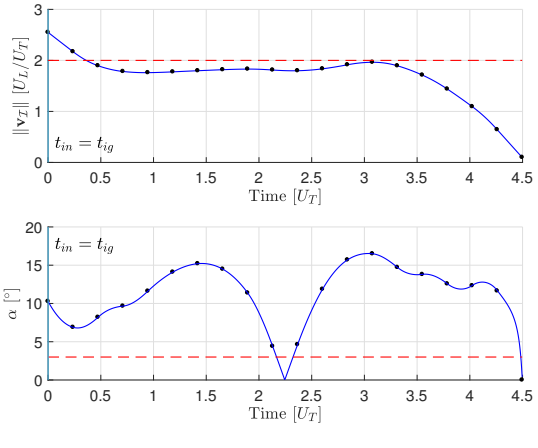
Figure 2.5: Aerodynamic Models Case Study: Trajectories for the baseline (B), spherical aerodynamic model (B)+(SA) and ellipsoidal aerodynamic model (B)+(EA) cases with $t_{in} = t_{ig}$, $r_\mathcal{I}(t_{in}) = [4.33\ 3.5\ 0.0]^T\ U_L$ and $v_\mathcal{I}(t_{in}) = [-0.5\ \ -2.5\ 0.0]^T\ U_L/U_T$. All force vectors are normalized to show direction only.



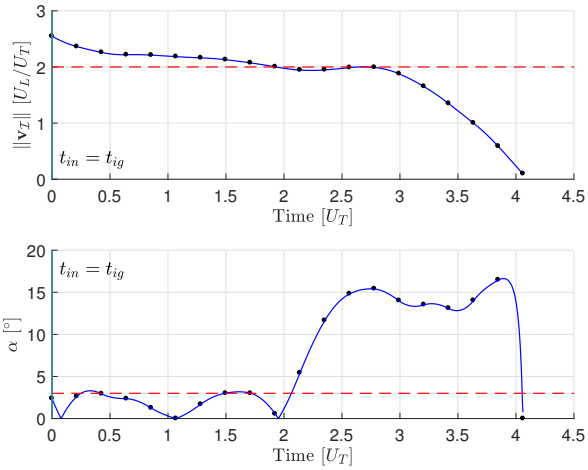
(a) Case (B)+(EA): Trajectory



(b) Case (B)+(EA)+(ST): Trajectory

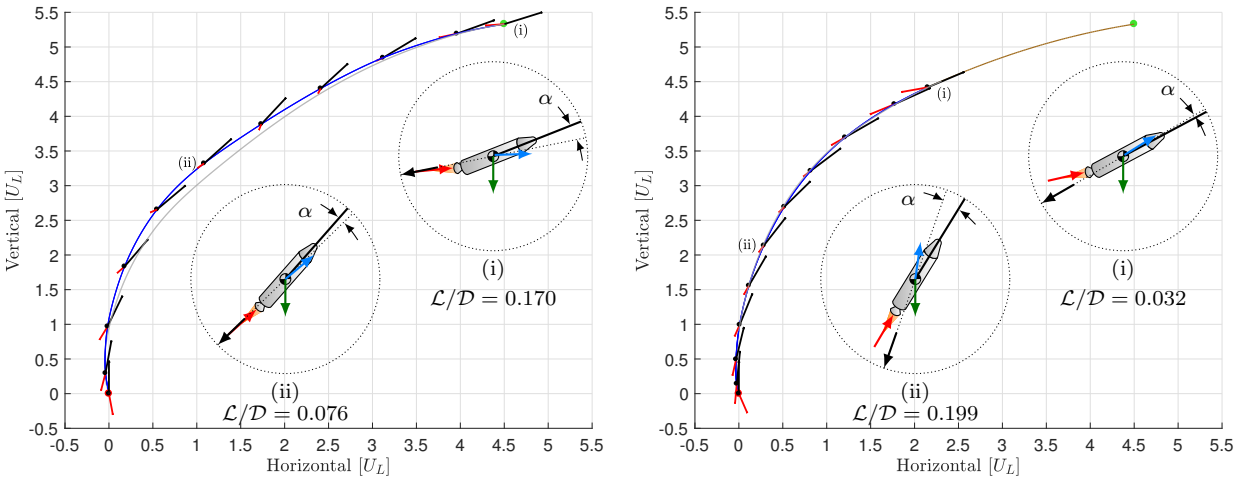


(c) Case (B)+(EA): STC Parameters



(d) Case (B)+(EA)+(ST): STC Parameters

Figure 2.6: State-Triggered Constraint Case Study: Trajectories and speed-angle of attack histories for cases (B)+(EA) and (B)+(EA)+(ST). This case study uses $r_\mathcal{I}(t_{in}) = [5.33\ 4.5\ 0.0]^T\ U_L$ and $v_\mathcal{I}(t_{in}) = [-0.5\ -2.5\ 0.0]^T\ U_L/U_T$. Case (B)+(EA)+(ST) implements the q-α state-triggered constraint introduced in (2.20), which limits the angle of attack to 3.0° when the speed is greater than 2.0 U_L/U_T. The horizontal dashed red lines in (c) and (d) represent the velocity limit V_α in the top set of axes, and the angle of attack limit α_{max} in the bottom set of axes. Note that that these limits are not enforced in (c), and are shown only for reference. Refer to the legend in Figure 2.5 for additional definitions.



(a) Case (B)+(EA)

(b) Case (B)+(EA)+(FI)

Figure 2.7: Free-Ignition-Time Case Study: Trajectories for case (B)+(EA) and (B)+(EA)+(FI), with $r_{\mathcal{I}}(t_{in}) = [5.33\ 4.5\ 0.0]^T\ U_L$ and $v_{\mathcal{I}}(t_{in}) = [-0.5\ \ -2.5\ 0.0]^T\ U_L/U_T$. Case (B)+(EA)+(FI) implements the free-ignition-time modification introduced in §2.2.4. Refer to the legend in Figure 2.5 for additional definitions.

Chapter 3

CONVEXIFICATION FOR QUAD-ROTOR PATH PLANNING

3.1 Introduction

In this chapter, our main contribution is to present preliminary experimental results on the application of lossless and successive convexification techniques to real-time on-board trajectory generation for quad-rotor systems. To demonstrate these techniques, we present two scenarios. In the first, we utilize *lossless* convexification to perform an agile flip maneuver. In doing so, we take advantage of a non-convex region of the control space, thus enabling the vehicle to exploit a portion of its feasible flight envelope that could not be used with existing convex optimization techniques. In the second, we employ both *lossless* and *successive* convexification to compute a feasible trajectory through a non-convex flight space populated with cylindrical no-fly-zones. These problems present some of the key challenges inherent in autonomous aerial robotic applications.

As quad-rotor and aerial robotics systems have come into prominence, their missions have increased in complexity. In many such applications, aerial platforms would be required to operate in cluttered environments, often in close proximity to humans. Thus, the ability to move in a constrained flight space while complying with environmental constraints is key to ensuring that aerial robotics systems can interact with humans in a safe and efficient manner. Specifically, these missions depend on the ability of the vehicles to transition from one state to another in a dynamically feasible way. Ideally, this can be done without sacrificing the performance that is inherent to the quad-rotor system. This topic has been addressed in a variety of ways, with implementations ranging from off-board pre-computed trajectories to on-board real-time solutions.

Historically, several path planning approaches have been proposed in the literature. Clas-

sical methods based on potential fields [39] make use of repulsive and attractive forces based on objects' assigned polarity. Although intuitive and easy to implement, these methods tend to generate conservative trajectories, and often produce undesirable equilibrium points away from the prescribed destination [24]. In [78], Mixed Integer Quadratic Programming (MIQP) is used to generate feasible trajectories and avoid obstacles for a small spacecraft in the context of rendezvous, docking, and proximity operation with the International Space Station. However, MIQPs are generally NP-hard and quickly become computationally intractable for on-board and real-time implementations. In [9], a hierarchical approach combines Model Predictive Control (MPC) and Hybrid MPC for quad-rotor stabilization and path planning with obstacle avoidance, and in [71], MPC is used to generate optimal trajectories to steer a quad-rotor from its initial state to a desired one, but the computations were done off-board on a standard desktop computer. Moreover, agile maneuvering often requires controls from the extremal points of the (possibly non-convex) feasible control space. Further, high-performance maneuvers typically traverse enough of the state-space that linearization about a single condition results in poor performance, or even instability. To overcome these issues, nonlinear MPC solvers [44] and methods like [47, 72] have been shown to be effective on multi-rotor vehicles equipped with high performance processors. Other approaches address the problem by exploiting the differential flatness property of the system to generate minimum snap trajectories [69]. This technique can also be embedded on-board and in real-time in a receding horizon scheme [95] or in a trajectory segmentation framework in which a different controller is designed for each flight phase [55].

Our approach focuses on leveraging the speed and reliability of convex optimization techniques to generate quad-rotor trajectories in a computationally tractable manner. This is increasingly true due to recent advancements in powerful customized convex optimization solvers [27, 28, 33, 65]. While other successful applications of convex optimization have used lower-order methods to similar ends [45, 65], our use of second-order methods allows us to handle a broader set of constraints. Although quad-rotor trajectory optimization problems are inherently non-convex, their dynamics and control constraints bear a strong resemblance

to those encountered in the planetary powered rocket landing problem. Consequently, the lossless convexification technique developed for the latter application [5, 16], and that was later generalized to a class of linear systems [2, 42], can be applied to the former. Lossless convexification of non-convex *control* constraints is important because it captures two key characteristics of quad-rotor motion in a convex optimization framework. First, it enables the application of a lower bound on the norm of the commanded thrust vector. This primarily ensures that the rotors continue to spin, and that the quad-rotor maintains attitude control authority throughout the trajectory. Second, it enables the use of the full tilt envelope, thus allowing the vehicle's thrust vector to point beyond 90° off-vertical.

Trajectory optimization for motion planning problems with non-convex *state* constraints require additional convexification in order to be solved in a convex optimization framework. For such scenarios, successive convexification can be employed to solve the problem as a sequence of convex optimization problems. This convexification strategy has been used for non-convex rocket landing problems [84,85] and hypersonic re-entry scenarios [58], and recent theoretical results on its convergence properties guarantee that if the process converges to a feasible solution, then the solution is a local optimum of the original non-convex problem [62,63]. In the case of quad-rotors, successive convexification is important because it enables us to solve a non-convex path planning problem using convex optimization methods well suited for on-board real-time use.

Using these convexification techniques, the problems outlined at the beginning of this section are cast into a second-order cone programming (SOCP) convex optimization framework. The resulting SOCPs are solved using fast and reliable Interior Point Method (IPM) algorithms. We emphasize that all computations presented in this chapter were performed in real-time and on-board the vehicle, whereby the trajectories were computed once at the onset of each maneuver, and thereafter tracked using closed-loop control.

The remainder of this chapter is organized as follows. In Section 3.2, we give a brief primer on lossless and successive convexification. In Section 3.3, we describe the hardware used in our experiments, and we present a high-level system architecture to illustrate how

the methods described herein interact with the quad-rotor system. In Section 3.4, we outline the problem formulations of the two aforementioned scenarios. Experimental results are presented in Section 3.5, and concluding remarks are given in Section 3.6.

3.2 Convexification

3.2.1 Lossless Convexification

Lossless convexification is a technique used to handle non-convex *control* constraints in an otherwise convex optimal control problem. To illustrate the technique, consider a fixed-time optimal control problem with state variable $x(t) \in \mathbb{R}^n$ and control variable $u(t) \in \mathbb{R}^3$. Suppose that our objective is to minimize the integral of $\|u(t)\|_2$ subject to the following constraints:

$$0 < u_{min} \leq \|u(t)\|_2 \leq u_{max} \tag{3.1a}$$

$$\|u(t)\|_2 \cos(\theta_{max}) \leq \hat{n}^T u(t) \tag{3.1b}$$

where u_{min} and u_{max} are, respectively, the minimum and maximum allowable magnitudes of $u(t)$, and $\theta_{max} \in (0^\circ, 180^\circ]$ is the maximum angle $u(t)$ is allowed to deviate from $\hat{n}$. Furthermore, suppose that $x(t) \in \mathbb{X}$, and that $\mathbb{X}$ is a convex set. Clearly, (3.1) define a non-convex control domain, thus rendering the optimal control problem non-convex.

Now, consider the relaxed problem whereby we introduce a slack variable, $\Gamma(t)$, and seek to minimize the integral of $\Gamma(t)$ subject to $x(t) \in \mathbb{X}$ and the following constraints:

$$0 < u_{min} \leq \Gamma(t) \leq u_{max} \tag{3.2a}$$

$$\|u(t)\|_2 \leq \Gamma(t) \tag{3.2b}$$

$$\Gamma(t) \cos(\theta_{max}) \leq \hat{n}^T u(t) \tag{3.2c}$$

Note that due to (3.2), the relaxed optimal control problem is now convex, although it is not obvious that the inequality in (3.2b) remains tight. A geometric interpretation of this relaxation is shown in Figure 3.1.

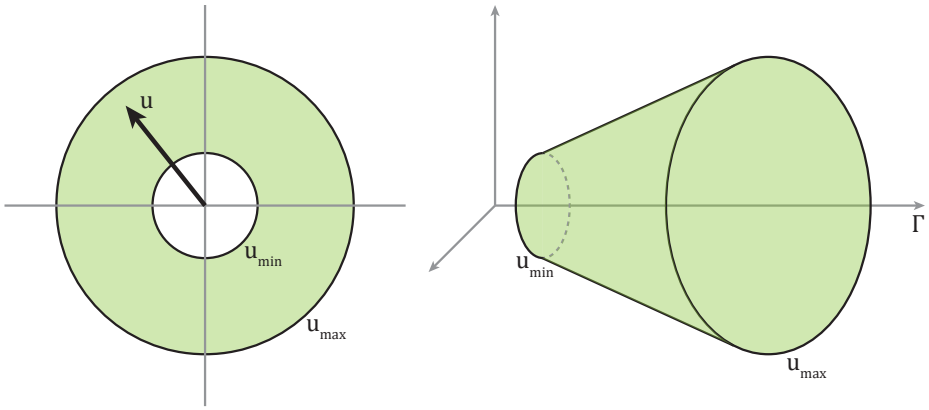


Figure 3.1: A geometric interpretation of lossless convexification for non-convex control constraints. (Left): Non-convex set of feasible controls; (Right): Convexified set of feasible controls

Simply stated, lossless convexification guarantees that if a feasible solution exists, the optimal solution to the (convex) relaxed problem recovers the optimal solution of the original non-convex problem. That is, we are guaranteed that the inequality in (3.2b) remains tight, and consequently, we are able to apply powerful convex optimization algorithms to solve the original non-convex problem.

The reader is referred to references [5, 42] for more details on lossless convexification of the thrust lower bound and [22] for lossless convexification of the pointing constraint when $\theta_{\max} \in (90°, 180°)$.

3.2.2 Successive Convexification

Successive convexification is a technique by which a non-convex optimal control problem is solved by solving a sequence of convex sub-problems. The original problem may have non-convex dynamics, and may contain non-convexities in both state equality and inequality constraints. Each sub-problem is derived from the original problem by linearizing the former

about the solution from the previous iteration. The process is relatively easy to initialize (e.g. see [84, 85]). Additionally, the problem is augmented with trust regions and virtual controls in order to avoid artificial unboundedness and infeasibility, respectively.

Recent theoretical results guarantee that if the solution process converges to a feasible solution, then the solution is a feasible local optimum of the original non-convex problem. However, in general, if no feasible solution is found, we are not guaranteed that the original problem is infeasible, and conversely, if the original problem is feasible, we are not guaranteed to find a feasible solution. Nevertheless, under mild assumptions, successive convexification performs well in practice.

The reader is referred to references [62, 63] for more details on successive convexification.

3.3 System Description

3.3.1 Hardware

The experimental results described in this chapter were generated using the Autonomous Controls Laboratory's (ACL's) custom-built quad-rotor platform, shown in Figure 3.2. The platform consists of a rugged custom-designed carbon fiber frame, a 2000 mAh LiPo battery, and four brushless DC motors. The on-board electronics were custom designed to house a Gumstix Overo Computer-On-Module that runs an ARM Cortex-A7 clocked at 750 MHz, has 512 MB of RAM, and uses an IEEE 802.11g compliant WiFi chip for network connectivity. All of the on-board and ground control software was written in C++, and was custom made for ACL's quad-rotor platform.

For indoor positioning, the ACL is equipped with an OptiTrack motion capture (MoCap) system that provides position data at millimeter-level accuracy using the Motive tracking software application. The system is comprised of Prime 17W and Prime 41 cameras, and can provide position and attitude data at up to 180 Hz.

Figure 3.2: ACL's custom quad-rotor platform. The vehicle is approximately 240 mm long from motor to motor and can hover for over 20 minutes.

3.3.2 SOCP Solution Algorithm and Software

The objective of our SOCP software is to quickly and reliably solve optimization problems related to our path planning approach. To this end, we utilize a custom C++ IPM solver that is capable of efficiently solving SOCP optimization problems [32, 33]. IPMs have polynomial time complexity and are guaranteed to find an optimal solution if a feasible solution exists. Conversely, if a problem is infeasible, IPMs provide a certificate of infeasibility in polynomial time complexity. Further, we make use of a highly efficient in-house C++ parser that easily transforms human-readable optimization problems (such as Problem 3) into a format that the IPM is expecting. Together, these tools enable us to solve convex optimization problems in real-time on-board the quad-rotors.

3.3.3 System Architecture

Figure 3.3 provides a high-level illustration of our system architecture. As shown, the aforementioned SOCP software was used to generate trajectories on-board the vehicle. The

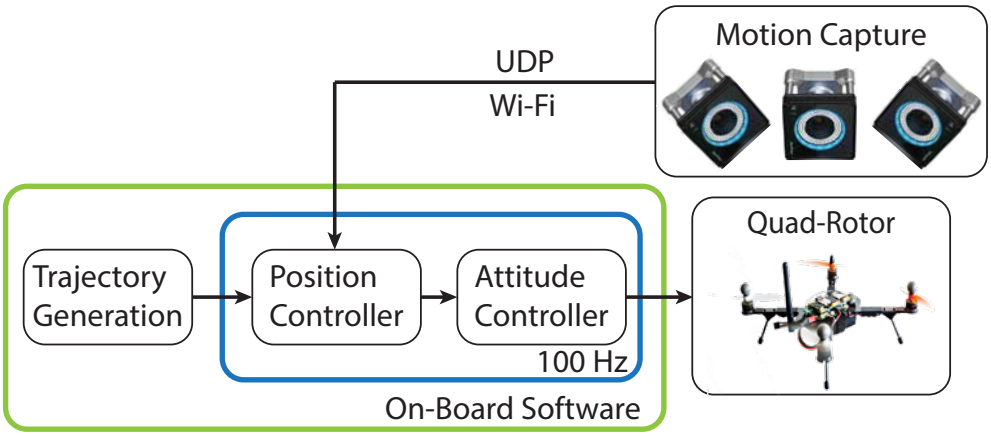


Figure 3.3: A high-level illustration of the system architecture. The content inside the green rectangle resides on-board the vehicle, and the content in the blue rectangle executes continuously at a high frequency.

optimization routine was executed once, after which its output was used as a reference for a feedback controller that we refer to as the *position* controller. This controller is charged with tracking the commanded positions, velocities, and accelerations of the generated trajectory. The position controller generates commands for the *attitude* controller, which actuates the motors in order to track the commanded attitude. The control loop was executed at 100 Hz, and position and velocity feedback were uplinked to the vehicle over the lab's WiFi network using UDP.

3.4 Problem Formulation

3.4.1 Agile Flip Maneuver

To demonstrate the benefits afforded by lossless convexification, we present a fixed-final-time optimal control problem that performs an agile flip maneuver. In what follows, we use t to denote time, and the subscripts i, c, and f to denote the initial, intermediate, and final temporal points, respectively. We model the vehicle using 3-degree-of-freedom (3-

DoF) translational dynamics, and denote the vehicle's position, velocity, and commanded acceleration as $r \in \mathbb{R}^3$, $v \in \mathbb{R}^3$, and $u \in \mathbb{R}^3$, respectively.

This formulation is used to keep to optimization problem tractable, while closed-loop control is used to address the higher-order dynamics present in the physical system. Consequently, some conservatism must be used in selecting the 3-DoF problem parameters to ensure the closed-loop control retains sufficient control authority.

We use e_j to denote the unit vector along the j^{th}-axis, and g to denote gravitational acceleration. Unless otherwise stated, all remaining parameters are constants that will be numerically specified in Section 3.5.1. The optimal control problem is stated in Problem 3. Note that unlike the minimum fuel objective used in Section 3.2.1, here we use a minimum-energy objective function. This objective still allows us to use lossless convexification, and ensures that the trajectory is smoother, and thus easier to follow. To implement this problem using a numerical solver, the problem is discretized assuming a first-order-hold on the control variable $u(t)$. As with our choice of the objective function, we assume a first-order-hold instead of a zero-order-hold in order to make the trajectory (and the control input) smoother and easier to track.

3.4.2 Obstacle Avoidance

To demonstrate the capabilities of successive convexification, we present a fixed-final-time optimal control problem where the objective is to fly from one point to another while avoiding known obstacles (i.e., obstacle locations, shapes, and sizes are known a-priori). First, we define our flight space using a set of half-spaces (i.e. floor, ceiling, etc.). Next, we populate our flight space with three-dimensional ellipsoidal keep-out zones, described by the following non-convex constraint

$$\|H_j(r(t) - p_j)\|_2 \geq R_j \quad \forall j \in \mathbb{J} \tag{3.4}$$

where p_j defines the center of the j^{th} ellipsoid, and where H_j and R_j define the shape and size of the the j^{th} ellipsoid. Note that H_j is assumed to be symmetric positive semi-

Problem 3

$$\underset{u(t),\Gamma(t)}{\textit{minimize}} \quad \int_0^{t_f} \Gamma^2(t)dt$$

subject to:

$$\begin{aligned}
r(0) &= r_i & v(0) &= v_0 & u(0) &= ge_3 \\
r(t_c) &= r_c & e_1^T v(t_c) &= 0 & & \\
& & e_3^T v(t_c) &= 0 & & \\
r(t_f) &= r_f & v(t_f) &= v_f & u(t_f) &= ge_3
\end{aligned}$$

$$\begin{aligned}
\dot{r}(t) &= v(t) \\
\dot{v}(t) &= u(t) - ge_3
\end{aligned}$$

$$\begin{aligned}
\|u(t)\|_2 &\leq \Gamma(t) \\
0 < u_{min} &\leq \Gamma(t) \leq u_{max} \\
\Gamma(t)\cos(\theta_{max}) &\leq e_3^T u(t)
\end{aligned}$$

definite, and that the number of keep-out zones is given by **card**($\mathbb{J}$). We assume that none of the keep-out zones intersect, and that each keep-out zone is either entirely contained within the flight space, or its center, p_j, is outside the flight zone. Next, we introduce a relaxation term, $\nu_j \geq 0$, to the linearized version of (3.4) (see (3.3)). This relaxation allows for violations of the keep-out zones during iterations where the solution has not yet converged, thus allowing the sub-problem to remain feasible and the convergence process to continue. The dynamics and control constraints of the quad-rotor are assumed to be the same as in

Section 3.4.1 (see Problem (4)), and thus, we retain the slack variable Γ introduced by the lossless convexification.

Before we present the algorithm, we note that a double integrator subject to acceleration constraints can follow any twice-continuously-differentiable path provided that it is given a *long enough* time to do so. Thus, it follows that if the problem stated here admits a feasible path, then there exists a time t_f^* such that for $t_f \geq t_f^*$ a feasible solution can be found.

The remainder of our solution strategy consists of solving Problem 4 in accordance to the steps outlined in Algorithm 2. Values for parameters that have not been specified will be provided in Section 3.5.2.

3.5 Experimental Results

In this section we present experimental flight results to (1) show that the proposed problem formulations capture enough of the dynamics to be practically useful, and (2) that the solutions to these optimal control problems can be computed quickly and reliably on embedded processors on-board the quad-rotors.

3.5.1 Agile Flip Maneuver

The parameters used to implement Problem 3 are presented in Table 3.1. They were selected such that the vehicle starts at the origin, passes through a waypoint that is up and to the right while moving with an unspecified velocity in the Y-direction, and then returns to a point 3 meters away from the origin along the Y-axis. The final time was chosen such that the maximum vehicle tilt angle exceeds 90°. 23 time discretization points were used in order to retain sufficient time sampling while keeping the runtime sufficiently low.

The commanded and actual trajectories are shown in Figure 3.4. The thrust tilt and magnitude profiles are shown in Figure 3.5. The commanded tilt angle can be seen to reach the 100° limit between 0.6 and 0.8 seconds.

As can be seen, the tracking error grows as the trajectory proceeds. We believe this is largely due to Problem 3's assumption that translational and attitude control efforts can be

Table 3.1: Parameters for Agile Flip Maneuver

Parameter	Value	Units	Parameter	Value	Units
t_c	0.7	s	r_i	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$	m
t_f	1.4	s	r_c	$\begin{bmatrix}1 & 1.5 & 1\end{bmatrix}^T$	m
g	9.81	m/s^2	r_f	$\begin{bmatrix}0 & 3 & 0\end{bmatrix}^T$	m
u_{min}	0.6	m/s^2	v_i	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$	m/s
u_{max}	23.2	m/s^2	v_f	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$	m/s
θ_{max}	100	$^\circ$			

Table 3.2: Timing Statistics for Agile Flip Maneuver (in [ms])

Processor	Min	Max	Med.	Mean	Std. Dev.
Overo	610.32	618.35	612.17	612.22	1.25
BBB	330.06	410.83	331.80	335.05	11.02
Edison	218.97	229.57	219.21	219.39	1.05
Joule	36.14	41.06	36.45	36.48	0.24

decoupled. This assumption relies heavily on the quick response time of the attitude loop and on the effectiveness of the feedback controller. Additionally, it is possible that a more conservative choice for u_{min} and u_{max} would result in better closed-loop performance, as the controller would have more thrust to correct for deviations. Further work is required to improve closed-loop three-dimensional trajectory tracking.

The timing statistics of Problem 3 are given in Table 3.2. All times are given in milliseconds. The statistics are given for four systems: (1) the Gumstix Overo with an ARM Cortex-A7 clocked at 750 MHz (flown on-board the vehicle), (2) the BeagleBone Black (BBB) with an ARM Cortex-A8 clocked at 1 GHz, (3) the Intel Edison with a dual-core Atom clocked at 500 MHz, and (4) the Intel Joule with a quad-core Atom clocked at 1.7 GHz. All computations were performed using double precision arithmetic.

3.5.2 Obstacle Avoidance

The parameters used to implement Algorithm 2 and Problem 4 are provided in Table 3.3. Vehicle parameters were provided in Table 3.1. In this scenario, we present the algorithm with two obstacle configurations consisting of the same two obstacles placed in different locations within the flight space. Flight area boundaries were not activated in this scenario and are thus omitted for brevity. The vehicle is required to traverse 6 meters by 2.5 meter space in 2 seconds while maintaining a constant altitude, starting and ending in a hover condition. This problem was discretized with 22 time discretization points for the same

Table 3.3: Parameters for Obstacle Avoidance

Parameter	Value	Units
t_f	2.0	s
α	1.5	-
i_{max}	5	-
r_{tol}	0.05	m
H_j	$\mathbf{diag}\left(\begin{bmatrix}1 & 1 & 0\end{bmatrix}\right)$	-
w	$1e^{-5}$	-

reasons mentioned in Section 3.5.1.

The computed and executed trajectories are shown in relation to the obstacles in Figure 3.6. As seen in the figure, the algorithm guides the vehicle to the right of the first obstacle and to the left of the second obstacle for the first configuration, and does the opposite for the second configuration. Despite being commanded to tilt angles in excess of 46 degrees, the closed-loop controller maintained the tracking error below 5 centimeters throughout the entirety of both trajectories.

Note that the discretization of the problem results in clipping of the keep-out-zones. This can be mitigated through a combination of enlarging the radius of the keep-out zones, increasing the number of discretization points, and imposing a sufficiently low maximum velocity constraint on the vehicle. Lastly, timing statics for Algorithm 2 are given in Table 3.4 for the same systems referenced in Table 3.2. It is notable that the Intel processors largely outperformed the ARM processors without a significant increase in power consumption. Moreover, the compact size of the Intel processors makes them ideal for aerial robotic applications, and the performance of the Joule makes real-time successive convexification attainable.

Table 3.4: Timing Statistics for Obstacle Avoidance Maneuver (in [ms])

Processor	Min	Max	Med.	Mean	Std. Dev.
Overo	2065	2089	2069	2069	2.93
BBB	1209	1259	1214	1224	18.31
Edison	753.7	763.7	754.1	754.7	2.09
Joule	120.4	131.2	122.5	122.2	1.12

3.6 Conclusion

In this chapter, we have demonstrated that numerical optimization techniques based on convex optimization are well-suited for applications that require real-time on-board motion planning in the presence of non-convex constraints. First, an agile flip maneuver that required the use of non-convex control constraints was planned on-board the vehicle in real-time via lossless convexification. Then, the quad-rotor computed trajectories that avoided cylindrical obstacles by utilizing successive convexification techniques. Trajectory generation runtimes in Table 3.4 suggest that obstacle avoidance trajectories can be obtained at approximately 10 Hz, and thus could be linked with a computer vision toolkit that detects (potentially moving) obstacles in real-time. Future work will focus on further improving the computational efficiency of successive convexification algorithms and acquiring theoretical bounds on its convergence rate.

Problem 4

$$\underset{u^k(t),\Gamma^k(t)}{minimize} \quad w\int_0^{t_f}\left(\Gamma^k(t)\right)^2 dt + \sum_{j\in\mathbb{J}}\nu_j$$

subject to:

$$r^k(0) = r_i \qquad v^k(0) = v_0 \qquad u^k(0) = ge_3$$
$$r^k(t_f) = r_f \qquad v^k(t_f) = v_f \qquad u^k(t_f) = ge_3$$

$$\dot{r}^k(t) = v^k(t)$$
$$\dot{v}^k(t) = u^k(t) - ge_3$$
$$\|u^k(t)\|_2 \leq \Gamma^k(t)$$
$$0 < u_{min} \leq \Gamma^k(t) \leq u_{max}$$
$$\Gamma^k(t)\cos(\theta_{max}) \leq e_3^T u^k(t)$$

$$x_{min} \leq e_1^T x^k(t) \leq x_{max}$$
$$y_{min} \leq e_2^T x^k(t) \leq y_{max}$$
$$z_{min} \leq e_3^T x^k(t) \leq z_{max}$$

For all $j \in \mathbb{J}$ *and for* $t \in [0, t_f]$:

$$\nu_j \geq 0, \quad H_j \succeq 0$$
$$\Delta r^{k,j}(t) \triangleq (r^{k-1}(t) - p_j), \quad \delta r^k(t) \triangleq r^k(t) - r^{k-1}(t)$$
$$\xi^{k,j}(t) \triangleq \|H_j \Delta r^{k,j}(t)\|_2, \quad \zeta^{k,j}(t) \triangleq \frac{H_j^T H_j \Delta r^{k,j}(t)}{\|H_j \Delta r^{k,j}(t)\|_2}$$

$$\xi^{k,j} + \left[\zeta^{k,j}(t)\right]^T \delta r^k(t) \geq R_j - \nu_j \tag{3.3}$$

Algorithm 2

Step 1: *Select a final time,* $t_f > 0$.

Step 2: *Solve Problem 4 using* t_f *without* (3.3).

Step 2.1: *If feasible, apply successive convexification (as in [62]) to solve Problem 4 using* t_f.

Step 2.1.1: *If* $\sup_{t \in 0, t_f} \|r^{k+1}(t) - r^k(t)\|_\infty \leq r_{tol}$ *for iterations* k *and* $(k+1)$, *then* **Exit**.

Step 2.1.2: *Else if number of iterations exceeds* i_{max} *reached, return to* **Step 1**, *and set* t_f *to* αt_f, $\alpha > 1$.

Step 2.2: *Else, return to* **Step 1**, *and set* t_f *to* αt_f, $\alpha > 1$.

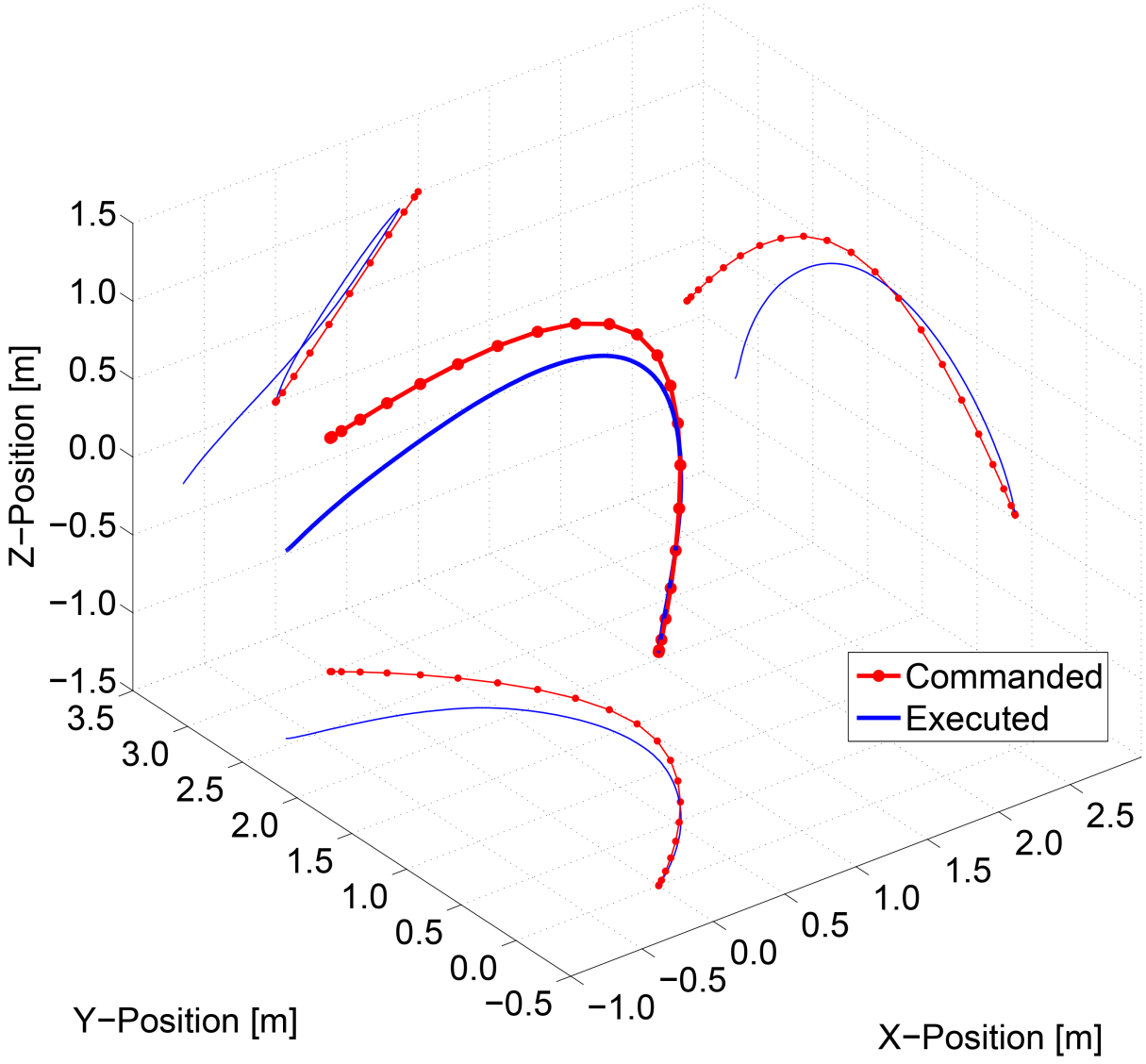


Figure 3.4: The agile flip maneuver trajectory. The red line indicates the commanded trajectory, and the blue represents the flown trajectory. The solid points along the red line indicate the discretization points of the optimization problem, and the thin lines represent projections of the three-dimensional profiles. The motion is from right to left.

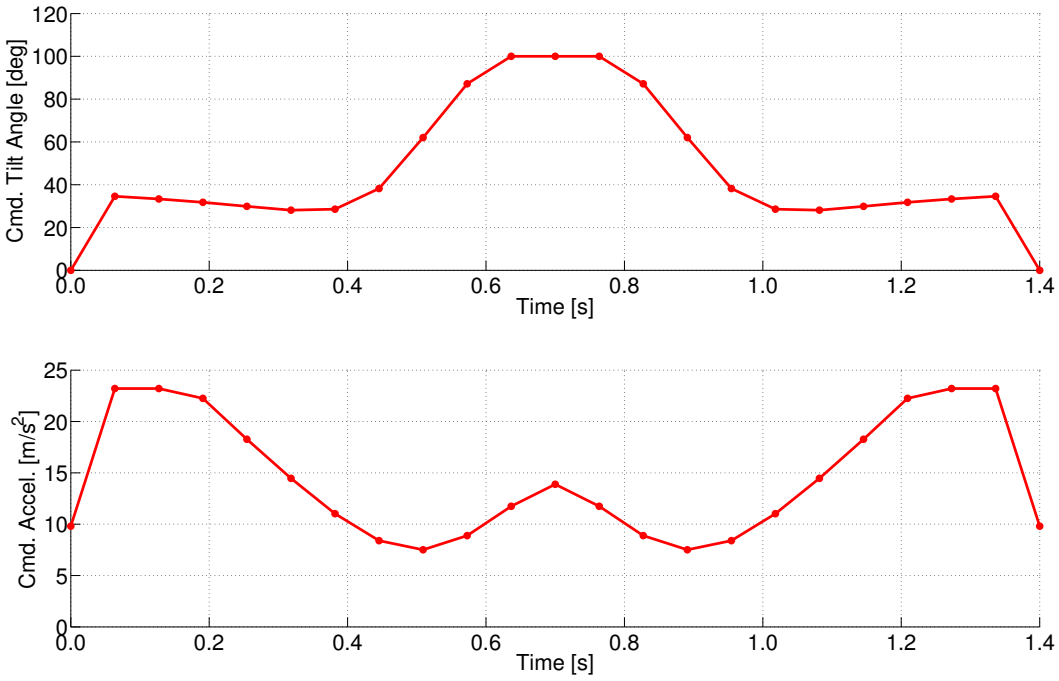


Figure 3.5: The top and bottom plots show the commanded tilt angles and accelerations as a function of time, respectively. The solid points represent the discrete time instances solved by the on-board optimizer.

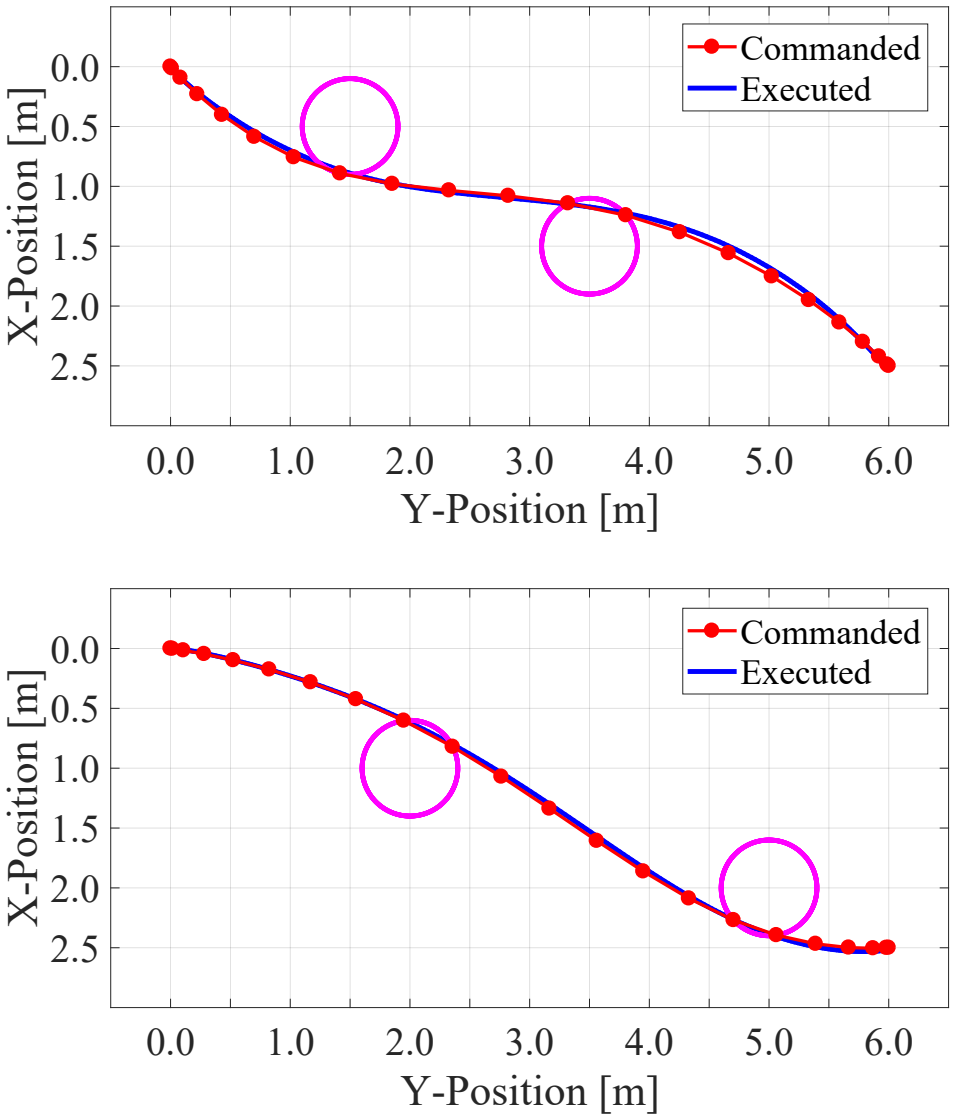


Figure 3.6: The top figure shows the commanded and executed trajectories in red and blue, respectively. The keep-out zones are shown as circles, and the motion is from left to right. The bottom figure shows the same information for the second obstacle configuration.

Chapter 4

QUAD-ROTOR PATH PLANNING FOR MOBILE OBSTACLE AVOIDANCE

4.1 Introduction

In this chapter, our primary contribution is demonstrating the real-time feasibility of on-board convex-optimization-based path planning in the presence of mobile obstacles. As such, our intent is to present a path planning scenario that requires real-time re-planning in order to adapt to a changing and uncertain environment. This represents a continuation of our previous work [88] with the following differences: (1) we assume that the obstacles follow constant-acceleration paths, and (2) we formulate the problem assuming linear interpolation of the control variable to improve tracking performance. Additionally, since a new trajectory is generated during each re-planning event, we employ a simple strategy to integrate the new trajectory with the previously computed ones.

The idea of path planning in the presence of obstacles is far from new, and a broad spectrum of algorithms have been proposed to find feasible solutions. Classical potential fields-based methods [39] make use of repulsive and attractive forces to avoid regions of space. Although intuitive and easy to implement, these methods tend to generate conservative trajectories, and often produce undesirable equilibrium points away from the prescribed destination [24]. In [78], Mixed Integer Quadratic Programming (MIQP) is used to generate feasible trajectories that avoid obstacles for a small spacecraft performing proximity operations with the International Space Station. However, MIQPs are generally NP-hard and quickly become computationally intractable for on-board and real-time implementations.

In [9], a hierarchical approach combines Model Predictive Control (MPC) and Hybrid MPC for quad-rotor stabilization and path planning with obstacle avoidance. MPC is used

to generate optimal trajectories to steer a quad-rotor from its initial state to a desired one in [71], but the computations were done off-board on a standard desktop computer. Agile maneuvering often requires controls from the extremal points of the (possibly non-convex) feasible control space. Further, high-performance maneuvers typically traverse enough of the state-space that linearization about a single state results in poor performance, or even instability. To overcome these issues, nonlinear MPC solvers [44] and methods like [47, 72] have been shown to be effective on multi-rotor vehicles equipped with high performance processors. Other approaches address the problem by exploiting the differential flatness property of the system to generate minimum-snap trajectories [69].

Our strategy is twofold. First, we leverage recent advancements in convexification techniques, specifically lossless [2,5,16,42] and successive [61,62,84,85] convexification, to convert the non-convex optimization problem into a sequence of Second-Order Cone Programming (SOCP) subproblems. Non-convex control constraints are handled by lossless convexification, whereas non-convex collision avoidance constraints are incorporated via successive convexification. Second, we solve each SOCP subproblem by leveraging recent advancements in Interior-Point-Method (IPM) algorithms [27, 28, 33, 45, 65]. As a result, we are able to compute feasible trajectories rapidly, thus enabling the real-time re-planning necessary to traverse an uncertain flight space inhabited by mobile obstacles.

The remainder of this chapter is organized as follows. In Section 4.2, we describe the original non-convex formulation, the convexified formulation, and implementation details. In Section 4.3, we give a brief description of the hardware and software used to conduct the experiments. In Section 4.4, we present flight testing and timing results for an example scenario. Lastly, in Section 4.5, we make concluding remarks.

4.2 Problem Formulation

4.2.1 Problem Description

In this section, we present a fixed-final-time non-convex path planning problem with mobile obstacles. We assume that the quad-rotor is equipped with fixed-pitch propellers.

For tractability, we begin by approximating the quad-rotor as a point mass subject to 3-degree-of-freedom (DoF) translational dynamics. This approximation is generally acceptable because the bandwidth of the attitude dynamics is significantly greater than the bandwidth of the translational dynamics.

We denote the vehicle's mass, position, and velocity as $m \in \mathbb{R}_{++}$, $r(t) \in \mathbb{R}^3$, and $v(t) \in \mathbb{R}^3$, respectively. The initial conditions are specified by the vectors r_i and v_i, and the final conditions are specified by the vectors r_f and v_f. The final time is denoted by t_f.

We denote the sum of the thrust vectors produced by the motors as, $T(t) \in \mathbb{R}^3$. The total thrust magnitude is limited to a minimum and maximum thrust of T_{min} and T_{max}, respectively. Due to the omission of the attitude dynamics, the values chosen for T_{min} and T_{max} should allow for sufficient attitude control authority when the total thrust activates either boundary. Furthermore, the thrust tilt angle is constrained to θ_{max} off-vertical. The initial and final thrust vectors are specified by the vectors T_i and T_f, respectively.

We assume the flight space is populated with n_{obs} mobile obstacles, and model the j^{th} obstacle as an ellipsoidal keep-out-zone centered at $r_j(t)$. The ellipsoidal keep-out-zone is defined by the positive semi-definite matrix $H_j(t) \in \mathbb{S}^3_{++}$. We denote the set of obstacles as $\mathbb{J} \coloneqq \{1, \ldots, n_{obs}\}$.

Lastly, we elect to use a minimum-fuel cost function, whereby we seek to minimize the integral of $\|T(t)\|_2$ over the trajectory.

A summary of the problem formulation is given in Problem 5, where g is used to denote gravity, and e_i is used to denote the unit vector along the i^{th} axis. The problem is formulated in an Up-East-North reference frame.

4.2.2 Implementable Convex Formulation

To implement Problem 5 as a numerical convex optimization problem, we must convexify and discretize the problem. The former step is achieved by following the procedure outlined in our previous work [88]. In what follows, we reference equations given in the summary provided in Problem 6, and simplify our notation by defining the commanded acceleration

Problem 5 ***Non-Convex Formulation***

$$\underset{T(t)}{\textit{minimize}} \quad \int_0^{t_f} \|T(t)\|_2 dt$$

subject to:

$$r(0) = r_i \qquad v(0) = v_i \qquad T(0) = T_i$$
$$r(t_f) = r_f \qquad v(t_f) = v_f \qquad T(t_f) = T_f$$

$$\dot{r}(t) = v(t)$$
$$\dot{v}(t) = \frac{1}{m}T(t) - ge_1$$

$$0 < T_{min} \leq \|T(t)\|_2 \leq T_{max} \tag{4.1}$$
$$\|T(t)\|_2 \cos(\theta_{max}) \leq e_1^T T(t) \tag{4.2}$$

$$\|H_j(t)\big(r(t) - r_j(t)\big)\|_2 \geq 1 \quad \forall j \in \mathbb{J}$$

term $a(t) \coloneqq T(t)/m$, and defining the terms a_{min}, a_{max}, a_i, and a_f accordingly.

We begin by applying lossless convexification to remove the non-convexity associated with the minimum thrust bound and the tilt angle constraint (for $\theta_{max} > 90°$). We do so by introducing the slack variable $\sigma(t)$, and replacing (4.1)-4.2 with (4.4). Lossless convexification ensures that the first inequality in (4.4) remains tight [2,5,16,42], and hence guarantees that the constraints of (4.1)-4.2 are satisfied. Thus, the solution to the relaxed convex problem recovers the solution to the original non-convex problem.

Next, we linearize the non-convex ellipsoidal keep-out-zone constraints to obtain (4.5),

Problem 6 ***Convex Formulation***

$$\underset{a_k^i,\sigma_k^i}{\textit{minimize}} \quad \sum_{k\in\mathbb{K}} \sigma_k + \sum_{j\in\mathbb{J}} \nu_j \,, \quad \nu_j \in \mathbb{R}_+$$

subject to:

$$\begin{aligned} r_1^i &= r_i & v_1^i &= v_i & a_1^i &= a_i \\ r_K^i &= r_f & v_K^i &= v_f & a_K^i &= a_f \end{aligned}$$

$$\begin{aligned} r_{k+1}^i &= r_k^i + v_k^i \Delta t + \frac{\Delta t^2}{6}\Big[2a_k^i + a_{k+1}^i - 3ge_1\Big] \\ v_{k+1}^i &= v_k^i + \frac{\Delta t}{2}\Big[a_k^i + a_{k+1}^i - 2ge_1\Big] \quad \forall k \in \bar{\mathbb{K}} \end{aligned} \tag{4.3}$$

$$\left.\begin{aligned} \|a_k^i\|_2 &\le \sigma_k^i \\ 0 < a_{min} \le \sigma_k^i &\le a_{max} \\ \sigma_k^i \cos(\theta_{max}) &\le e_1^T a_k^i \end{aligned}\right\} \forall k \in \mathbb{K} \tag{4.4}$$

$$\left.\begin{aligned} f_{j,k}^i &\coloneqq \|H_{j,k}(r_k^i - r_{j,k})\|_2 \\ 1 - f_{j,k}^{i-1} - \tfrac{\partial f_{j,k}^{i-1}}{\partial r_k^{i-1}}(r_k^i - r_k^{i-1}) &\le \nu_j \end{aligned}\right\} \begin{aligned} &\forall j \in \mathbb{J} \\ &\forall k \in \mathbb{K} \end{aligned} \tag{4.5}$$

$$\|r_k^i - r_k^{i-1}\|_1 \le \Delta^i \quad \forall k \in \mathbb{K} \tag{4.6}$$

and apply successive convexification to generate a sequence of convex SOCP subproblems using the `SCvx` algorithm developed in [62]. Consequently, we solve Problem 6 repeatedly, denoting the current iterate using the superscript i, and linearizing about the $(i-1)^{\text{th}}$ iterate. Note that the `SCvx` algorithm introduces a trust region constraint on r^i (see (4.6)).

Algorithm 3 ***High-Level Algorithm***
Step 1: *Select a final time,* $t_f > 0$.
Step 2: *Solve Problem 6 using* t_f *without* (4.5).
Step 2.1: *If not feasible, increase* t_f,
recompute Δt, *and return to* **Step 2**.
Step 2.2: *Else, if obstacle violation is too great,*
use t_f *to solve Problem 6 using* `SCvx`.
Step 2.2.1: *If obstacle violation is still too great,*
increase t_f, *recompute* Δt, *and*
return to **Step 2.2**.
Step 2.2.2: *Else, proceed to* **Step 2.4**
Step 2.3: *Else, if sufficiently close to* r_f,
complete trajectory, and **Exit**
Step 2.4: *Else, accept trajectory*
Step 2.4.1: *Fly trajectory for one time step.*
Step 2.4.2: *Set* $(r_2, v_2, a_2) \rightarrow (r_i, v_i, a_i)$,
and $(t_f - \Delta t) \rightarrow t_f$
Step 2.4.3: *Return to* **Step 2**.

Lastly, we discretize the problem at K evenly spaced time points, and denote the discrete time index using the subscript $k \in \mathbb{K}$, where $\mathbb{K} \coloneqq \{1, \ldots, K\}$. For convenience, we also define $\bar{\mathbb{K}} \coloneqq \{1, \ldots, K-1\}$. The time interval Δt between adjacent time points is given by $\Delta t = t_f/(K-1)$. To attain smoother trajectories, we perform a first-order-hold on $a(t)$ over $t \in (t_k, t_{k+1})$ for all $k \in \bar{\mathbb{K}}$. This measure aids in tracking the 3-DoF trajectory, while causing only a slight detriment to computational performance. The discretized dynamics are given in (4.3).

4.2.3 Uncertain Obstacle Motion

Thus far, we have assumed that the obstacle motion is perfectly known. However, in realistic scenarios, mobile obstacles have uncertain motion. To deal with this uncertainty, we must recompute the trajectory when new obstacle motion information becomes available.

In this chapter, we assume that obstacle motion is estimated externally to our algorithm, and that the motion information assumes constant acceleration. Thus, if the obstacle continues to accelerate as assumed, then the obstacle's position is known as a function of time, and the computed trajectory can be followed safely. However, if the obstacle deviates from its assumed trajectory, then the motion information must be reevaluated, and a new trajectory must be computed.

At every time instance, t_k, an `SCvx` solution is computed using the most up-to-date obstacle motion information. Once the solution is obtained, the trajectory is tracked via closed-loop control for a period of Δt. Simultaneously, the next trajectory computation begins using the updated obstacle motion information, and starting from r_{k+1}, v_{k+1}, and a_{k+1} from the previously computed trajectory. With each computation, t_f is shortened by Δt. The process terminates when the vehicle is sufficiently close to r_f.

If infeasibility or unacceptable obstacle violation is encountered, t_f is increased and the trajectory is recomputed. Assuming that the obstacle cannot out-maneuver the vehicle, this procedure will reduce the maximum acceleration in the trajectory, eventually producing a feasible trajectory with finite t_f. The process is summarized in Algorithm 3.

4.3 System Description

For completeness, we include this section to provide a few updates to the system description presented in [88].

4.3.1 Hardware

The flight experiments conducted in the Autonomous Controls Laboratory (ACL) employ a custom-designed quad-rotor platform. The vehicles have a motor-to-motor distance of

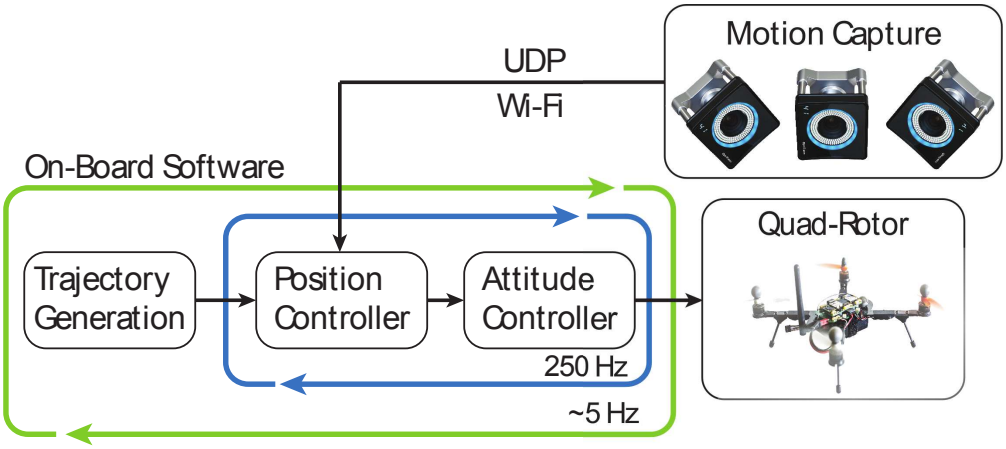


Figure 4.1: A high-level illustration of the system architecture. The components inside the green loop reside on-board the vehicle. The components inside the blue loop comprise the feedback architecture.

240 mm, use variable-speed brushless DC motors in conjunction with fixed-pitch propellers, and are each powered by a 2200 mAh LiPo battery that provides a flight time of 15-30 minutes. The on-board electronics include a 6-DoF inertial measurement unit, a 500 MHz dual-core Intel Edison, a 1.7 GHz quad-core Intel Joule, and an IEEE 802.11n compliant WiFi communication link. The flight critical software is executed on the Edison, whereas the optimization software is executed on the Joule. Furthermore, the ACL is equipped with an OptiTrack motion capture system that is used for both position and attitude feedback.

4.3.2 SOCP Software

The trajectories generated for the experiments presented herein were computed using ACL's custom SOCP parser and solver software. The solver is based on an IPM algorithm, and was implemented in C++. It is capable of solving SOCP problems quickly and reliably [32, 33], and enjoys polynomial time complexity. Additionally, IPMs are easy to initialize, and, if a solution exists, are guaranteed to find the optimal solution in a deterministic number of

iterations. In the event of infeasibility, IPMs can provide a certificate of infeasibility stating that no solution exists. We remind the reader that the guarantees associated with IPMs pertain to the SOCP subproblems. Thus, they can be leveraged in applications that use lossless convexification, but do not hold for the successive convexification process as a whole.

4.3.3 System Architecture

The algorithm detailed in Section 4.2 was used in conjunction with the aforementioned SOCP software to generate trajectories at a rate of 5 Hz. This update rate was based on the timing results obtained during testing (see Section 4.4.2), and was selected to provide sufficient margin to account for observed system delays. The trajectory was tracked using a 250 Hz feedback control loop. Figure 4.1 provides a high-level illustration of the system architecture.

4.4 Experimental Results

4.4.1 Experimental Setup

In this section we present results from flight experiments conducted in the ACL indoor flight space. The experiment was performed using four vehicles: a primary vehicle that executed the path planning algorithm, and three secondary vehicles that served as mobile obstacles. For the sake of clarity, we elected to constrain the scenario to the horizontal plane. However, we stress that the results presented herein were obtained using an algorithm that performed the full three-dimensional computations.

In our example flight demonstration, the obstacles are set to circle about a fixed point at a constant radius, R_{obs}, at a constant speed, V_{obs}, and at equal angular spacing. As mentioned in Section 4.2.3, the algorithm assumes that the obstacles move with constant acceleration. As such, if the obstacles maintain their constant speed, the motion estimate of the obstacles retains its validity for the entire trajectory. Thus, in order to introduce uncertainty, we command the obstacles to either stop, or reverse direction. Consequently, every time the obstacles change speed, the algorithm recomputes the trajectory in order to

Table 4.1: Nominal Scenario & Algorithm Parameters

Parameter	Value	Units	Parameter	Value	Units
K	20		r_i	$-3e_3$	m
t_f	2.75	s	r_f	$3e_3$	m
m	0.33	kg	v_i	0	m/s
g	9.81	m/s^2	v_f	0	m/s
T_{min}	1.0	N	R_{obs}	1.25	m
T_{max}	4.0	N	V_{obs}	$\{1, 0, -1\}$	m/s
θ_{max}	40	$^\circ$	$H_{j,k}$	$I_{3\times3}$	m^{-1}

avoid a collision.

The obstacle keep-out-zones were sized through trial and error, and were designed to account for the size of the primary and secondary vehicles, the maximum expected trajectory tracking error, and the expected uncertainty in the obstacle motion. As such, scenarios with less predictable fast moving obstacles should use larger keep-out-zones than a scenario with slow moving predictable obstacles. Note, however, that while larger keep-out-zones may provide increased vehicle spacing, they can also cause the trajectory to be unnecessarily aggressive if t_f is not appropriately increased. Additionally, $H_{j,k}$ can be selected to have an eccentricity in directions of larger motion uncertainty. For instance, if the obstacle is known to follow unicycle dynamics, its uncertainty can be assumed larger in the along-track direction than in the cross-track direction. For simplicity, we assume a constant keep-out radius, R, of one meter for all obstacles.

The scenario parameters are provided in Table 4.1. Care must be taken to select t_f and K so as to accommodate subsequent computations and re-computations.

4.4.2 Timing Results

For the scenario described in the previous section, five sets of 1000 `SCvx` computations were executed, with K varying from 20 to 40 in intervals of 5. To capture more variability, the runs were conducted from starting points along the entire trajectory in order to simulate the run times that would be encountered when traversing the flight space. The computations were executed on an on-board Intel Joule, and their timing statistics were collected. These statistics represent a measure of the combined IPM and `SCvx` computation timings. The results are tabulated in Table 4.2. For the case where $K = 20$, the maximum computation time encountered is 81.4 [ms], enabling update rates in excess of 10 Hz. For this reason, and due to ease of implementation, the re-planning rate was limited to approximately 7 Hz (Δt^{-1}), and was maintained even if t_f was increased (e.g. Step 2.2.1 in Algorithm 3). This was done to ensure that the on-board processor could recompute the trajectory in the event obstacle motion uncertainty resulted in infeasibility. Note that these results were obtained using a generic SOCP solver, and could potentially be improved via solver customization [33, 65].

4.4.3 Flight Demonstrations

Figures 4.2a-4.2j and 4.3a-4.3j show the results obtained during flight testing. Readers are encouraged to view the associated online video. The primary vehicle is shown as a red circle, whereas the obstacle keep-out-zones are shown as black circles. The dots along the path

Table 4.2: Timing Statistics for Scenario (in [ms])

K	Min	Max	Med.	Mean	Std. Dev.
20	**8.7**	**81.4**	**33.0**	**37.9**	**22.2**
25	17.7	177.4	44.4	48.7	38.3
30	24.6	166.1	62.7	69.7	33.6
35	41.0	164.5	79.9	87.5	31.4
40	85.6	367.3	100.5	125.2	71.2

represent the K temporal discretization points, and the thin line emanating at each point represents the commanded acceleration vector, a_k. The alternating colors of the thick lines represent the portion of the trajectory implemented during the corresponding time step. Together, these thick red and blue lines represent the executed trajectory. Due to changes in obstacle motion, the executed trajectory differs from the trajectories computed at each time instance.

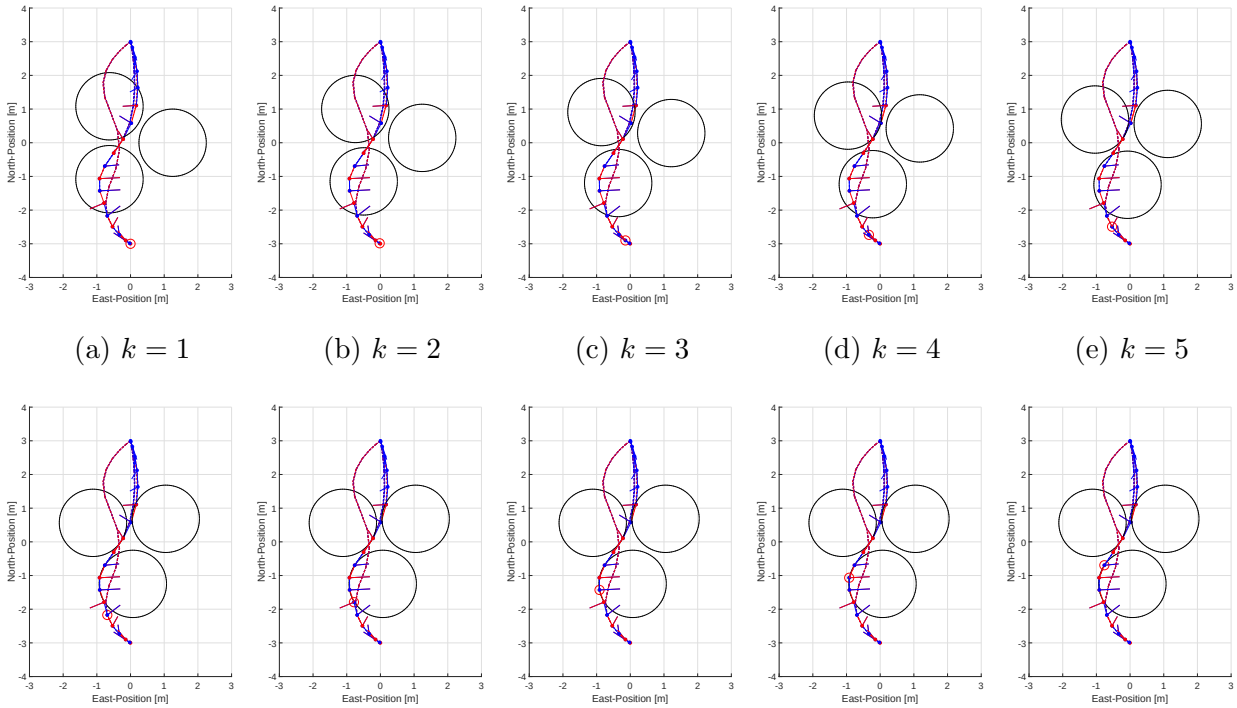
The scenario starts at $k = 1$ in the configuration shown in Figure 4.2a. The initial motion of the obstacles is in the *counter-clockwise* direction. The initially computed trajectories are seen bundled together under the red dashed line traversing the flight space. At $k = 6$ (Figure 4.2f), the obstacles stop suddenly, and the newly computed trajectory changes to avoid the bottom obstacle that was anticipated to continue its leftward motion. The thrust vector implemented at $k = 7$ (Figure 4.2f) indicates that the vehicle is aggressively trying avoid the obstacle, and in $k = 8$ subsequently resumes its right turn.

The obstacles remain stationary until $k = 11$ (Figure 4.3a), when they assume a constant *clockwise* motion. The bundle of trajectories from $6 \leq k \leq 10$ is seen as the dashed blue line at the top middle of the figures. The change at $k = 11$ indicates that the primary vehicle modified its trajectory to avoid the now clockwise-moving top obstacle. The solid black line in Figure 4.3j represents the trajectory flown in the flight demonstration.

Note that if the obstacles initiated motion towards the primary vehicle while the vehicle was riding an obstacle boundary, the subsequent trajectory computation would necessarily violate one of the keep-out-zones. While the current implementation would retain feasibility and attempt to exit the keep-out-zone as soon as possible, robust MPC techniques could be used to enlarge the keep-out-zones and avoid such situations altogether.

4.5 Conclusion

In this chapter, we have presented results from flight experiments that employed real-time on-board convex-optimization-based path planning for mobile obstacle avoidance. The key contribution of this chapter is the demonstration of the feasibility of real-time on-board

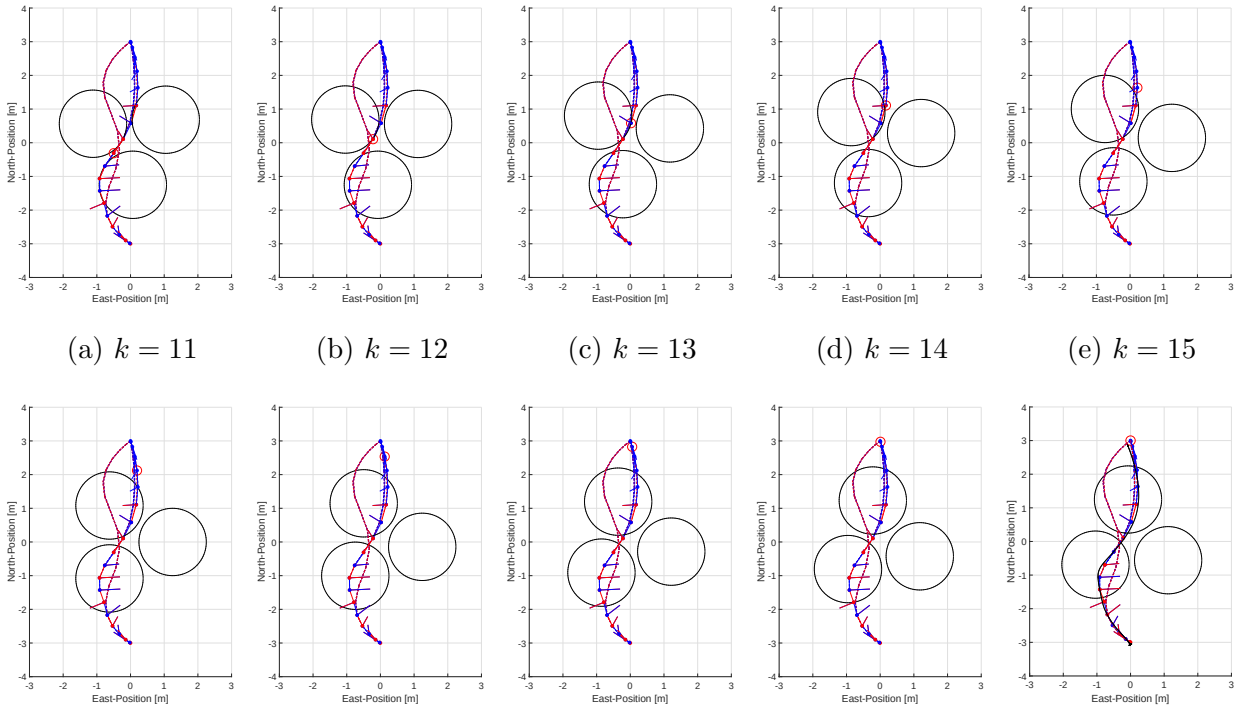


(a) $k = 1$ (b) $k = 2$ (c) $k = 3$ (d) $k = 4$ (e) $k = 5$

(f) $k = 6$ (g) $k = 7$ (h) $k = 8$ (i) $k = 9$ (j) $k = 10$

Figure 4.2: (Part 1 of 2) A time-lapse of the motion of the primary and secondary vehicles. The primary vehicle is shown as the red circle, whereas the keep-out-zones around the secondary vehicles are shown as black circles. The solid points along the path represent each of the K discretization points, and the thin line emanating from each point represents the thrust vector at the respective time index. The scenario begins with plot (a) and terminates at plot (t). The black line seen in plot (t) represents the flown trajectory. The obstacles orbit the $(0, 0)$ point in a counter-clockwise direction for $k = 1, \ldots, 5$, and in a clockwise direction for $k = 11, \ldots, 20$. The obstacles are stationary for $k = 6, \ldots, 10$.

convex-optimization-based path planning, and was made possible by the application of lossless and successive convexification techniques. This point was emphasized by selecting a scenario that necessitated re-planning of the trajectory due to the uncertainty in the motion of the obstacles. The execution time of the algorithm on an Intel Joule allowed path planning to be executed at a rate in excess of 10 Hz with sufficient margin for various system delays.



(a) $k = 11$ (b) $k = 12$ (c) $k = 13$ (d) $k = 14$ (e) $k = 15$

(f) $k = 16$ (g) $k = 17$ (h) $k = 18$ (i) $k = 19$ (j) $k = 20$

Figure 4.3: (Part 2 of 2) A time-lapse of the motion of the primary and secondary vehicles. The primary vehicle is shown as the red circle, whereas the keep-out-zones around the secondary vehicles are shown as black circles. The solid points along the path represent each of the K discretization points, and the thin line emanating from each point represents the thrust vector at the respective time index. The scenario begins with plot (a) and terminates at plot (t). The black line seen in plot (t) represents the flown trajectory. The obstacles orbit the $(0, 0)$ point in a counter-clockwise direction for $k = 1, \ldots, 5$, and in a clockwise direction for $k = 11, \ldots, 20$. The obstacles are stationary for $k = 6, \ldots, 10$.

While we believe that our results are an important step towards realizing the vision of real-time on-board optimization-based path planning for robotic platforms, our future work will focus on improving the fidelity of the algorithm with quad-rotor dynamics and constraints, as well as improving the means by which vehicle parameters and limitations are translated to motion planning constraints.

Chapter 5

QUAD-ROTOR PATH PLANNING USING STATE-TRIGGERED CONSTRAINTS

5.1 Introduction

The main contribution of this chapter is the application of compound state-triggered constraints (STCs) to quad-rotor path planning applications. STCs, and their generalized counterparts, compound STCs, were recently introduced to solve powered-descent guidance rocket landing problems that contained discrete decisions [76, 89, 90]. Simply stated, this class of constraints enables user-defined constraint conditions to be enforced if other user-defined trigger conditions are satisfied. To the best of our knowledge, STCs are novel since they capture this discrete logical implication without relying on discrete decision variables. Instead, STCs are formulated using continuous variables, and in practice work well within existing continuous optimization frameworks (e.g. successive convexification). As a result, STCs can be interpreted as *if*-statements that are embedded inside a continuous optimization problem.

Over the past two decades, direct methods for solving optimal control problems have seen a rise in popularity due to the ease of use, performance, and convergence properties offered by modern optimization algorithms [11, 57]. Direct methods are typically used to solve problems with continuous variables, and cannot readily enforce constraints involving discrete decisions. The most common technique used to address this shortcoming is through the use of mixed-integer programming techniques. Despite the existence of efficient branch-and-bound methods, mixed-integer programming techniques suffer from poor computational complexity [12, 77]. The real-time capabilities of such techniques are further hampered when evaluating each set of discrete decisions is expensive.

Mixed-integer programming problems appear in quad-rotor applications quite frequently.

In [70], a centralized mixed-integer quadratic programming (MIQP) approach was used to perform collision avoidance among a team of four heterogeneous quad-rotors. The results showed that feasible solutions could be found in tenths of a second, but that optimality required significantly more computational effort. As discussed in the paper, the methodology was not easily scalable to larger teams of quad-rotors. In [91], the authors formulated a different MIQP problem to handle the hybrid dynamics of a quad-rotor flying with a mass suspended by a non-rigid string. The paper illustrated that the hybrid nature of the dynamics could be exploited to allow the vehicle to perform otherwise infeasible maneuvers. However, the paper reported computation times upwards of 100 s. In [50], a Mixed-Integer Semi-Definite Programming (MISDP) approach was proposed to perform aggressive obstacle avoidance in highly cluttered environments (5-26 obstacles). The approach was able to impressively avoid very small obstacles, but reported average computation times of approximately 10 minutes.

In this chapter, we propose an STC-based approach that prioritizes computational speed while settling for locally optimal solutions. Two scenarios are used to demonstrate the proposed methodology: (1) a quad-rotor flying through a hoop, and (2) a pair of quad-rotors carrying a beam-like payload through an obstacle course. The combinatorial elements of these scenarios are formulated into a continuous framework using compound STCs. The successive convexification framework [62, 63, 83] is used to cast the original non-convex problem into a sequence of convex Second-Order Cone Programs (SOCPs). Our results show that the optimal control problems associated with the first and second scenarios can be solved reliably at average update rates of 15 and 4 Hz, and no slower than 3 and 1.5 Hz, respectively.

In this chapter, we adopt the following notation and conventions: an Up-East-North reference frame is used throughout; $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{R}_{++}$ are used to denote the set of reals, non-negative reals, and positive reals; $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$, and $\mathbb{S}^n_{++}$ are used to denote the space of n-dimensional vectors, $m\times n$-dimensional matrices, and $n\times n$-dimensional symmetric positive semi-definite matrices; $\mathcal{S}^n \subset \mathbb{R}^{n+1}$ is the unit n-sphere; for vectors quantities, the symbol $\hat{}$ is used to signify unity norm; $\hat{e}_j$ is used to denote a unit vector with a unity j^{th} element;

$z \in \mathbb{R}^{n_z}$ is used to denote a generic solution variable of an optimization problem.

This chapter is organized as follows: in §5.2, we give a brief overview of STCs and compound STCs; in §5.3, we detail our modeling assumptions and the two motivating scenarios; in §5.4, we outline the successive convexification algorithm used in the subsequent section; in §5.5, we present our Monte Carlo simulation results for both scenarios; and in §5.6, we provide concluding remarks.

5.2 State-Triggered Constraints

In this section we provide a concise introduction to STCs. We refer the reader to [76, 89, 90] for more details.

5.2.1 Logical Statement

An STC is composed of two parts: a *trigger condition* given by the strict inequality $g(z) < 0$, and a *constraint condition* given by the inequality $c(z) \leq 0$. We call $g(z) : \mathbb{R}^{n_z} \to \mathbb{R}$ the *trigger function*, and $c(z) : \mathbb{R}^{n_z} \to \mathbb{R}$ the *constraint function*. Both $g(\cdot)$ and $c(\cdot)$ are assumed to be differentiable. Formally, an STC enforces the following logical relationship:

$$g(z) < 0 \;\Rightarrow\; c(z) \leq 0. \tag{5.1}$$

The practical value of an STC is most evident from the contrapositive of (5.1), namely, that the constraint condition is not satisfied *only if* the trigger condition is not satisfied.

5.2.2 Continuous Formulation

Mixed-integer programming is the most common framework used to implement discrete decisions such as (5.1). However, this approach suffers from poor computational complexity due to the combinatorial nature of integer variables [12, 77]. Moreover, even in the absence of STC-like constraints, practical (non-convex) path planning problems often require the use of sequential solution methods (e.g. Sequential Quadratic Programming, Successive Convexification). For these reasons we seek a continuous formulation of (5.1) that is amenable to

a sequential (continuous) implementation without incurring the added computational complexity of mixed-integer approaches.

An equivalent continuous formulation of (5.1) was introduced in [89], and is given by

$$h(z) \coloneqq \hat{\sigma}(z) \cdot c(z) \leq 0, \tag{5.2}$$

where $\hat{\sigma}(z) \coloneqq -\min\big(0, g(z)\big)$. By inspection, we see that if $g(z) < 0$, then $\hat{\sigma}(z) > 0$, and (5.2) reduces to $c(z) \leq 0$. In contrast, if $g(z) \geq 0$, then $\hat{\sigma}(z) = 0$, and (5.2) is trivially satisfied for any value of $c(z)$ (i.e. the constraint condition is not enforced). Thus, we conclude that (5.1) and (5.2) are logically equivalent, and emphasize that the latter can be implemented in a continuous optimization framework.

5.2.3 Compound State-Triggered Constraints

Compound STCs were introduced in [90], and are a generalization of the scalar STC formulation given in (5.1) and (5.2). Compound STCs have trigger and constraint conditions that are composed using Boolean "and" or "or" operations. Here, we present compound STCs with "and"- and "or"-trigger conditions, and "or"-constraint conditions. The logical representations of these STCs are given by

$$\bigwedge_{j=1}^{n_g} \big(g_j(z) < 0\big) \;\Rightarrow\; \bigvee_{j=1}^{n_c} \big(c_j(z) \leq 0\big), \tag{5.3a}$$

$$\bigvee_{j=1}^{n_g} \big(g_j(z) < 0\big) \;\Rightarrow\; \bigvee_{j=1}^{n_c} \big(c_j(z) \leq 0\big), \tag{5.3b}$$

where there are n_g trigger conditions, n_c constraint conditions, and each $g_j(\cdot)$ and $c_j(\cdot)$ is defined as in the scalar case. The corresponding continuous formulations are given by

$$h_{\wedge}(z) \coloneqq \left[\prod_{j=1}^{n_g} \hat{\sigma}_j(z)\right] \cdot \left[\prod_{j=1}^{n_c} \big(c_j(z) + \alpha_j\big)\right] = 0, \tag{5.4a}$$

$$h_{\vee}(z) \coloneqq \left[\sum_{j=1}^{n_g} \hat{\sigma}_j(z)\right] \cdot \left[\prod_{j=1}^{n_c} \big(c_j(z) + \alpha_j\big)\right] = 0, \tag{5.4b}$$

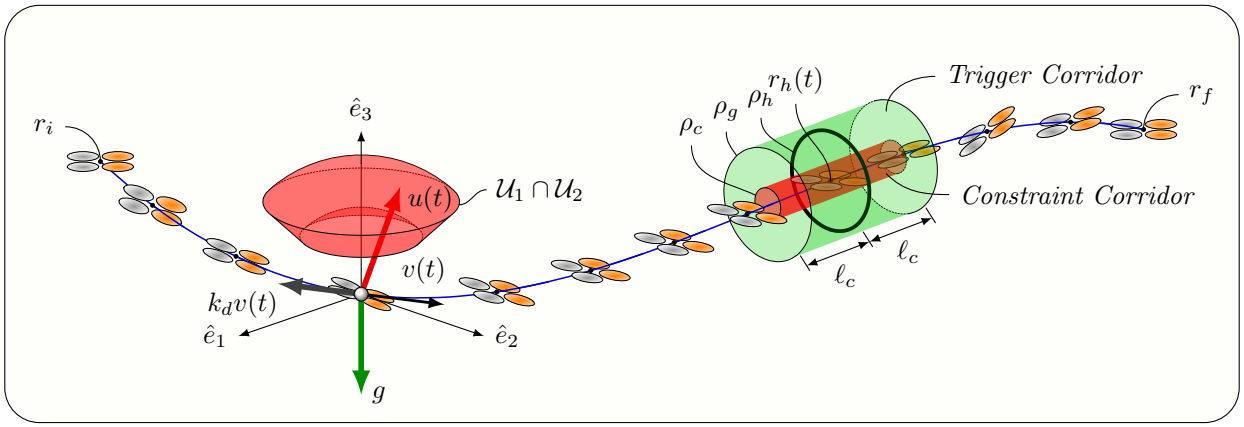


Figure 5.1: Illustration of Scenario 1: The feasible thrust set is shown as the red volume on the left. The velocity, thrust, drag, and gravity vectors are shown as the black, red, gray, and green vectors, respectively. The initial and final positions are indicated on the left and right, respectively. The hoop, trigger corridor, and constraint corridor are shown on the right. The vector $\hat{n}_h$ is orthogonal to the plane of the hoop, and is not shown. The compound STC makes the volume that is inside the trigger corridor but outside of the constraint corridor infeasible.

where $\alpha_j \in \mathbb{R}_+$ are non-negative slack variables, and each $\hat{\sigma}_j(\cdot)$ is defined as in the scalar case.

We conclude this section with two comments. First, formulations with equality constraint conditions can be obtained by substituting equalities in place of the (non-strict) inequalities in (5.1)-(5.3), and omitting the slack variables in (5.4). Second, a compound STC with an "and"-constraint condition is emulated by enforcing n_c separate compound STCs, each with the original compound trigger condition and one of the (scalar) constraint conditions. We are now ready to apply STCs to the scenarios detailed in the next section.

5.3 Problem Formulation

In this section, we outline two quad-rotor path planning scenarios using compound STCs. Before presenting the two scenarios, we briefly discuss our assumed guidance and control (G&C) architecture and simplified quad-rotor model.

5.3.1 G&C Architecture

The G&C architecture assumed in this chapter separates guidance and control into two distinct tasks. The *guidance task* involves generating an open-loop trajectory at a low frequency, whereas the *control task* involves computing high frequency closed-loop control actions to stay on the guidance trajectory. The objective of the guidance task is to ensure feasibility (e.g. respecting vehicle dynamics and control limits, and avoiding obstacles), while the objective of the control task is to provide robustness to plant uncertainties and external disturbances (e.g. wind gusts, and battery voltage variability). The control task is typically subdivided into hierarchically arranged thrust, attitude, and translation controllers.

5.3.2 Simplified Quad-Rotor Dynamics

We assume a 3-DoF quad-rotor dynamics model, similar to the one used in [87, 88]. This model is given by

$$\dot{x}(t) = Ax(t) + Bu(t) + Ew,$$

$$A \coloneqq \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & -k_d I_{3\times3} \end{bmatrix},\ B \coloneqq \frac{1}{m}\begin{bmatrix} 0_{3\times3} \\ I_{3\times3} \end{bmatrix},$$

$$x(t) \coloneqq \begin{bmatrix} r^\intercal(t) & v^\intercal(t) \end{bmatrix}^\intercal,\ E \coloneqq -\hat{e}_4,\ w \coloneqq g,$$

where $r(t) \in \mathbb{R}^3$ is the position state, $v(t) \in \mathbb{R}^3$ is the velocity state, $u(t) \in \mathcal{U} \subset \mathbb{R}^3$ is the thrust (control) vector, $m \in \mathbb{R}_{++}$ is the mass of the vehicle, $k_d \in \mathbb{R}_{+}$ is the drag coefficient, and $g \in \mathbb{R}_{++}$ is the local gravitational acceleration. Note that the drag model is simplified due to its linear dependence on $v(t)$.

To define the control set $\mathcal{U}$, we first define three sets. The first set represents the allowable thrust magnitudes of the vehicle, and is given by

$$\mathcal{U}_1 \coloneqq \big\{u \in \mathbb{R}^3 : 0 < T_{min} \leq \|u\|_2 \leq T_{max}\big\},$$

where T_{min} and T_{max} are the minimum and maximum allowable thrust magnitudes. In practice, these bounds are selected conservatively to ensure that the underlying controllers can command thrust and torques independently. Note that $\mathcal{U}_1$ is non-convex. The second set represents the allowable tilt angles of the vehicle, and is given by

$$\mathcal{U}_2 \coloneqq \big\{u \in \mathbb{R}^3 : \cos\theta_{max}\|u\|_2 \leq \hat{e}_1^{\mathsf{T}} u\big\},$$

where $\theta_{max} \in (0°, 180°)$ is the maximum allowable tilt angle. Note that $\mathcal{U}_2$ is non-convex for $\theta_{max} > 90°$. The third set represents the thrust vectors with a vertical component equal and opposite to the weight of the vehicle (i.e. control inputs that maintain a constant altitude). This set is given by

$$\mathcal{U}_3 \coloneqq \big\{u \in \mathbb{R}^3 : \hat{e}_1^{\mathsf{T}} u = mg\big\},$$

and has a non-empty interior when $T_{min} \leq mg < T_{max}$.

For three-dimensional applications, $\mathcal{U} = \mathcal{U}_1 \cap \mathcal{U}_2$ is non-convex, and the optimal control problem can be convexified using the lossless convexification technique introduced in [5] (also see [88]). For two-dimensional applications requiring only horizontal motion, $\mathcal{U} = \mathcal{U}_1 \cap \mathcal{U}_2 \cap \mathcal{U}_3$ is convex, and the vertical dimension of the problem can be omitted from the formulation of the guidance problem.

5.3.3 Scenario 1: Quad-Rotor Flying Through a Hoop

The first scenario involves flying a quad-rotor through a hoop, and is three dimensional in nature (see Figure 5.1). We formulate this scenario as a fixed-final-time optimal control problem of duration $t_f \in \mathbb{R}_{++}$.

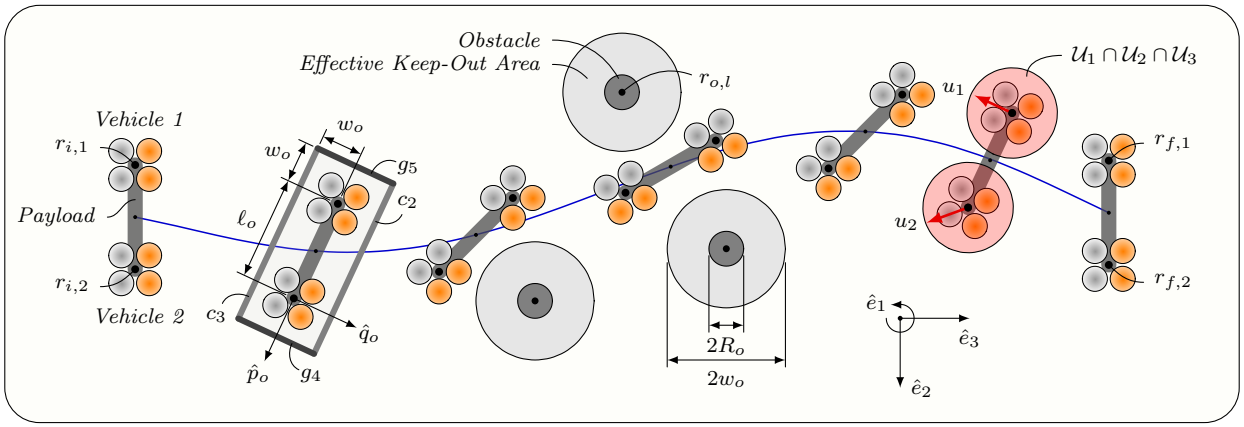


Figure 5.2: Illustration of Scenario 2: The two vehicles and the payload are shown in their initial state on the left. The keep-out region defined by (5.8) and (5.9) is shown at the second time instance. This region effectively keeps the vehicles and the payload outside of the *effective keep-out areas* shown around the obstacles. Lastly, the convex control sets are represented by the red circles in the second to last time instance.

The boundary conditions of this problem are prescribed through the initial and final position vectors r_i and r_f. The initial and final velocities are assumed to equal zero, and the initial and final controls are assumed to be equal and opposite the weight of the vehicle.

The position of the center of the hoop is denoted by $r_h(t) \in \mathbb{R}^3$, and can vary with time. We assume that $r_h(t)$ is known for all $t \in [0,\, t_f]$. We represent the hoop's orientation using an orthogonal unit vector $\hat{n}_h \in \mathcal{S}^2$, which we assume is constant for simplicity. We note that our formulation readily handles small variations in hoop orientation, and emphasize that $r_h(t)$ and $\hat{n}_h$ *are not* decision variables of the optimal control problem.

The constraint imposed by the hoop is formulated using a compound STC that restricts the path of the quad-rotor to a *constraint corridor* whenever the vehicle is inside a *trigger corridor*. The constraint corridor functions to guide the vehicle through the hoop without collision. The geometry of this constraint is illustrated in Figure 5.1, where ρ_h denotes the radius of the hoop. Omitting the time arguments of $r(t)$ and $r_h(t)$, the logical implication of

this compound STC is given by

$$\bigwedge_{j=1}^{3} \big(g_j(r, r_h) < 0\big) \;\Rightarrow\; \big(c_1(r, r_h) \leq 0\big), \tag{5.5a}$$

$$g_1(r, r_h) \coloneqq \hat{n}_h^{\mathsf{T}}(r_h - r) - \ell_c, \tag{5.5b}$$

$$g_2(r, r_h) \coloneqq \hat{n}_h^{\mathsf{T}}(r - r_h) - \ell_c, \tag{5.5c}$$

$$g_3(r, r_h) \coloneqq (r - r_h)^{\mathsf{T}} \hat{N}_h^{\mathsf{T}} \hat{N}_h (r - r_h) - \rho_g^2, \tag{5.5d}$$

$$c_1(r, r_h) \coloneqq (r - r_h)^{\mathsf{T}} \hat{N}_h^{\mathsf{T}} \hat{N}_h (r - r_h) - \rho_c^2, \tag{5.5e}$$

where $\ell_c \in \mathbb{R}_{++}$ is the half-length of the corridor, ρ_g is the radius of the trigger corridor, ρ_c is the radius of the constraint corridor, and $\hat{N}_h \coloneqq I_{3\times 3} - \hat{n}_h \hat{n}_h^{\mathsf{T}}$. In practice, these parameters are selected to satisfy $0 \leq \rho_c \ll \rho_h \ll \rho_g$. From (5.4a) and (5.5), we obtain the following continuous formulation:

$$h_1(r, r_h) \coloneqq \left[\prod_{j=1}^{3} \hat{\sigma}_j(r, r_h)\right] \cdot c_1(r, r_h) \leq 0, \tag{5.6}$$

Problem 7 ***Non-Convex Formulation of Scenario 1***

$$\underset{u}{\textit{minimize}} \int_0^{t_f} \|u(t)\|_2 dt$$

subject to:

$$r(0) = r_i,\; r(t_f) = r_f,$$

$$v(0) = v(t_f) = 0_{3\times 1},$$

$$u(0) = u(t_f) = mg\hat{e}_1,$$

$$\dot{x}(t) = Ax(t) + Bu(t) + Ew,\; u(t) \in \mathcal{U}_1 \cap \mathcal{U}_2,$$

$$\|v(t)\|_2 \leq v_{max},\; h_1\big(r(t), r_h(t)\big) \leq 0.$$

where $\hat{\sigma}_j(r, r_h) \coloneqq -\min\big(0, g_j(r, r_h)\big)$. Since $n_c = 1$ in this case, we omit the slack variable from (5.4a), and replace the equality with an inequality. The associated fuel-optimal non-convex optimal control problem is summarized in Problem 7. This problem has two sources of non-convexity: the control set, and the compound STC given in (5.6). The former is addressed using lossless convexification (see §5.3.2), whereas the latter is convexified using successive convexification (see §5.4).

We conclude this section with a few remarks. First, note that Problem 7 includes a constraint that limits the velocity to a maximum of $v_{max} \in \mathbb{R}_{++}$. This constraint is added in order to mitigate constraint clipping introduced by temporal discretization (see §5.5). Second, the above scenario is similar to the *Agile Flip Maneuver* presented in [88], which required the quad-rotor to maneuver through a waypoint defined midway along the trajectory. However, the key difference between [88] and the scenario described in this section is that the compound STC enables the optimization to choose *if* and *when* the trajectory will pass through the hoop. Third, the above formulation can be modified such that the vehicle is *required* to pass through the hoop. This can be done either by selecting ρ_g sufficiently large, or by omitting $g_3(\cdot)$ from the formulation. Lastly, the direction and speed of the trajectory at the hoop can be specified by enforcing an additional compound STC with an appropriate velocity-dependent constraint condition.

5.3.4 Scenario 2: Cooperative Obstacle Avoidance

The second scenario consists of two identical quad-rotors cooperatively negotiating an obstacle course (see Figure 5.2). We restrict the motion of the vehicles to the horizontal plane, hence making this a two-dimensional scenario. As in the first scenario, we treat this problem as a fixed-final-time problem, and use the subscripts 1 and 2 to distinguish between quantities associated with the two vehicles.

The position boundary conditions are given by the initial position vectors $r_{i,1}$ and $r_{i,2}$, and the final position vectors $r_{f,1}$ and $r_{f,2}$. For each vehicle, the velocity and control boundary conditions are identical to those used in the first scenario.

The quad-rotors are linked together by a beam-like payload of length $\ell_o \in \mathbb{R}_{++}$, modeled by the following non-convex equality constraint:

$$\|r_1(t) - r_2(t)\|_2 = \ell_o. \tag{5.7}$$

We assume that the vehicles maintain their ability to control their attitudes independently of one another (i.e. each vehicle can control its attitude as in §5.3.3), and that the boundary conditions are feasible with respect to (5.7).

The flight space contains N_o stationary cylindrical obstacles of identical radius $R_o \in \mathbb{R}_{++}$. The position of each obstacle $l \in \mathcal{N}_o \coloneqq \{1, \ldots, N_o\}$ is denoted by $r_{o,l} \in \mathbb{R}^3$. Each obstacle is assumed to vertically span the available space.

This scenario is challenging since the absence of vehicle-obstacle collisions does not guarantee the absence of payload-obstacle collisions. To address this issue, a compound STC is used to define a keep-out rectangle around the two vehicles. Omitting the time arguments of $r_1(t)$ and $r_2(t)$, the logical implication of this compound STC is given by

$$\bigwedge_{j=1}^{2} \big(g_{j+3}(\cdot,\cdot,\cdot) < 0\big) \;\Rightarrow\; \bigvee_{j=1}^{2} \big(c_{j+1}(\cdot,\cdot,\cdot) \leq 0\big), \tag{5.8a}$$

$$g_4(r_1, r_2, r_{o,l}) \coloneqq \hat{p}_o^{\mathsf{T}}(r_1 - r_{o,l}) - w_o, \tag{5.8b}$$

$$g_5(r_1, r_2, r_{o,l}) \coloneqq \hat{p}_o^{\mathsf{T}}(r_{o,l} - r_2) - w_o, \tag{5.8c}$$

$$c_2(r_1, r_2, r_{o,l}) \coloneqq \hat{q}_o^{\mathsf{T}}(r_{o,l} - r_2) + w_o, \tag{5.8d}$$

$$c_3(r_1, r_2, r_{o,l}) \coloneqq \hat{q}_o^{\mathsf{T}}(r_1 - r_{o,l}) + w_o, \tag{5.8e}$$

where $\hat{p}_o \coloneqq (r_2 - r_1)/\|r_2 - r_1\|_2$, $\hat{q}_o$ is orthogonal to $\hat{p}_o$, and $w_o \in \mathbb{R}_{++}$ is the minimum spacing enforced around each vehicle. In practice, w_o is selected such that $w_o \geq R_o$. These quantities are illustrated in Figure 5.2. From (5.4a) and (5.8), we obtain the following continuous formulation:

$$\begin{aligned} h_2(r_1, r_2, r_{o,l}) \coloneqq &\left[\prod_{j=1}^{2} \hat{\sigma}_{j+3}(r_1, r_2, r_{o,l})\right] \\ &\cdot \left[\prod_{j=1}^{2} \big(c_{j+1}(r_1, r_2, r_{o,l}) + \alpha_j\big)\right] = 0, \end{aligned} \tag{5.9}$$

where $\hat{\sigma}_j(\cdot,\cdot,\cdot) \coloneqq -\min\big(0, g_j(\cdot,\cdot,\cdot)\big)$, and $\alpha_1, \alpha_2 \in \mathbb{R}_+$ are non-negative slack variables as in (5.4). Defining the following quantities

$$\tilde{A} \coloneqq \mathbf{blkdiag}\left(A, A\right), \; \tilde{B} \coloneqq \mathbf{blkdiag}\left(B, B\right),$$
$$\tilde{E} \coloneqq \mathbf{blkdiag}\left(E, E\right), \; \tilde{w} \coloneqq [w \;\; w]^\mathsf{T},$$
$$\tilde{x} \coloneqq [x_1^\mathsf{T} \;\; x_2^\mathsf{T}]^\mathsf{T}, \; \tilde{u} \coloneqq [u_1^\mathsf{T} \;\; u_2^\mathsf{T}]^\mathsf{T},$$

the associated fuel-optimal non-convex optimal control problem is summarized in Problem 8. This problem has two sources of non-convexity: the equality constraint given in (5.7), and the compound STC given in (5.9). Both of these non-convexities are handled using successive convexification (see §5.4).

We conclude this section with three remarks. First, the control set is convex due to the intersection of $\mathcal{U}_1 \cap \mathcal{U}_2$ with $\mathcal{U}_3$. Second, implementations of Problem 8 can omit the vertical dimension of the problem due to the two-dimensional nature of this scenario. Third, the compound STC in (5.8) and (5.9) can be extended to more complicated geometries (e.g. multiple vehicles vehicles carrying an L-shaped payload).

5.4 Successive Convexification

In this section, we provide an overview of the successive convexification algorithm used to solve Problems 7 and 8. We refer the reader to [89] for more details.

Successive convexification is a framework that solves non-convex continuous-time optimal control problems by solving a sequence of convex discrete-time parameter optimization *subproblems*. Each subproblem is an SOCP that approximates the original problem by linearizing non-convexities about the previous iteration, and is obtained using two steps: discretization and linearization.

5.4.1 Discretization & Linearization

The (temporal) *discretization step* divides the time horizon of the optimal control problem into $K-1$ temporal intervals of length $\Delta t \coloneqq t_f/(K-1)$. For each node $k \in \mathcal{K} \coloneqq$

Problem 8 ***Non-Convex Formulation of Scenario 2***

$$\underset{u}{minimize} \int_0^{t_f} \big(\|u_1(t)\|_2 + \|u_2(t)\|_2\big)dt$$

subject to:

$$r_1(0) = r_{i,1},\ r_1(t_f) = r_{f,1},$$

$$r_2(0) = r_{i,2},\ r_2(t_f) = r_{f,2},$$

$$v_1(0) = v_2(0) = v_1(t_f) = v_2(t_f) = 0_{3\times 1},$$

$$u_1(0) = u_2(0) = u_1(t_f) = u_2(t_f) = mg\hat{e}_1,$$

$$\dot{\tilde{x}}(t) = \tilde{A}\tilde{x}(t) + \tilde{B}\tilde{u}(t) + \tilde{E}\tilde{w},$$

$$u_1(t), u_2(t) \in \mathcal{U}_1 \cap \mathcal{U}_2 \cap \mathcal{U}_3,$$

$$h_2\big(r_1(t), r_2(t), r_{o,l}\big) = 0,\ \ \forall j \in \mathcal{N}_o,$$

$$\|v_1(t)\|_2 \leq v_{max},\ \|v_2(t)\|_2 \leq v_{max},$$

$$\|r_1(t) - r_2(t)\|_2 = \ell_o.$$

$\{1, 2, \ldots, K\}$, the time is given by $t_k \coloneqq (k-1)\Delta t$.

Since the dynamics of Problems 7 and 8 are linear time-invariant, the discrete-time dynamics can be expressed analytically as a function of t_f. Assuming a first-order-hold on the control, we represent these discrete-time dynamics by the quantities $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, $A_d \in \mathbb{R}^{n_x \times n_x}$, $B_d^-, B_d^+ \in \mathbb{R}^{n_x \times n_u}$, $E_d \in \mathbb{R}^{n_x \times n_w}$, and $w_d \in \mathbb{R}^{n_w}$. The discrete-time state and control constraints are obtained by enforcing said constraints at each temporal node.

The *linearization step* linearizes the non-convexities that cannot be convexified using lossless convexification (i.e. (5.6), (5.7), and (5.9)). Since this approximation is only made to first order, the subproblem is *guaranteed* to be convex. However, the linearization also

introduces two issues: *artificial infeasibility* and *artificial unboundedness*.

To aide in the ensuing explanation, we define $\bar{\mathcal{K}} \coloneqq \mathcal{K} \setminus K$, $\bar{u} \coloneqq [u_1^\mathsf{T}, \ldots, u_K^\mathsf{T}]^\mathsf{T}$, $z_k \coloneqq [x_k^\mathsf{T}, u_k^\mathsf{T}, \alpha_k]^\mathsf{T}$, and $\bar{z} \coloneqq [z_1^\mathsf{T}, \ldots, z_K^\mathsf{T}]^\mathsf{T}$, where $\alpha_k \in \mathbb{R}_+^{n_\alpha}$ is a vector of non-negative slack variables and $n_z = n_x + n_u + n_\alpha$. We concatenate the non-convex state constraints into the vector-valued equality constraint $h(z_k) = 0$.

Artificial infeasibility is resolved by adding *virtual control* terms $\bar{\nu} \in \mathbb{R}^{n_x}$ to the discrete-time dynamics, and augmenting the cost with $J_{vc}(\bar{\nu}) \coloneqq \sum_{k \in \bar{\mathcal{K}}} \|W_{vc}\nu_k\|_1$, where $W_{vc} \in \mathbb{S}_{++}^{n_x}$ is a user-specified weight matrix, and $\bar{\nu} \coloneqq [\nu_1^\mathsf{T}, \ldots, \nu_{K-1}^\mathsf{T}]^\mathsf{T}$. The addition of $J_{vc}(\cdot)$ penalizes violations of the dynamics, and allows dynamic infeasibility to occur (if necessary) during the convergence process.

Artificial unboundedness is resolved by augmenting the cost with $J_{tr}(\bar{z}) \coloneqq \sum_{k \in \mathcal{K}} \delta z_k^\mathsf{T} W_{tr} \delta z_k$, where $W_{tr} \in \mathbb{S}_{++}^{n_z}$ is a user-specified weight matrix, $\delta z_k \coloneqq z_k - z_k^*$, and z_k^* denotes the solution obtained during the previous iteration. The addition of $J_{tr}(\cdot)$ ensures that Problem 9 remains bounded, and keeps the solution close to the linearization point.

5.4.2 Subproblem

Problem 9 summarizes the subproblem used in the proposed lossless convexification algorithm. This problem consists of (i) an objective function made up of the original objective $J(\bar{z})$ and the two augmented cost terms discussed in §5.4.1; (ii) the boundary conditions denoted by $x_{d,i}$, $u_{d,i}$, $x_{d,f}$, and $u_{d,f}$; (iii) the discrete-time dynamics with virtual control; $(\ iv\ )$ the non-convex state constraints linearized about the previous solution; and (v) the control set $\mathcal{U}$ detailed in §5.3.2. As stated previously, we assume that non-convexity in $\mathcal{U}$ is handled by lossless convexification [88].

5.4.3 Algorithm

The successive convexification algorithm is outlined in Algorithm 4, and is a simplified version of the soft-trust-region algorithm presented in [89]. The algorithm is initialized by user-specified problem data (e.g. t_f, boundary conditions) and an initialization trajectory

Problem 9 ***Convex Subproblem (SOCP)***

$$\underset{\bar{u},\bar{\nu}}{minimize}\ J(\bar{z}) + J_{tr}(\bar{z}) + J_{vc}(\bar{\nu})$$

subject to:

$$x_1 = x_{d,i},\ x_K = x_{d,f},\ u_1 = u_{d,i},\ u_K = u_{d,f},$$

$$x_{k+1} = A_d x_k + B_d^- u_k + B_d^+ u_{k+1} + E_d w_d + \nu_k,\ \ \forall k \in \bar{\mathcal{K}},$$

$$h(z_k^*) + \left.\frac{\partial h}{\partial z_k}\right|_{z_k^*} \delta z_k = 0,\ u_k \in \mathcal{U},\ \ \forall k \in \mathcal{K}.$$

generated by linearly interpolating between the specified boundary conditions. The subsequent discretization step computes the discrete-time dynamics. The algorithm then enters a loop that successively linearizes and solves Problem 9 until the trust region and virtual control costs $J_{tr}(\cdot)$ and $J_{vc}(\cdot)$ are less than their respective specified thresholds $\epsilon_{tr}, \epsilon_{vc} \in \mathbb{R}_{++}$. Upon convergence, the algorithm returns the converged solution $\bar{z}$. If the algorithm does not converge within a set number of iterations, then t_f is increased and the problem is resolved.

5.5 Results

This section presents Monte-Carlo simulation results for the two scenarios presented in §5.3. We focus on measuring the runtime of Algorithm 4 and quantifying two primary failure modes of Algorithm 4: failure to converge in less than 20 iterations, and inter-sample constraint violation. All results were obtained on a desktop PC running a Ubuntu 18.04.1 operating system with a 3.60 GHz Intel Cote i7-6850K processor and 64 GB of RAM. MATLAB was used to run the ECOS [27] solver through the CVX [37] parsing interface, and timing data were obtained from the *solve time* parameter returned by the `cvx_toc` function.

Algorithm 4 ***Successive Convexification (Soft-TR)***

initialize - *provide problem data, and* z_k^* *for all* $k \in \mathcal{K}$

discretize - *compute* A_d, B_d^-, B_d^+, E_d, w_d

set `converged` $= 0$

while (`converged` $= 0$) **do**

    linearize - *compute* $h(z_k^*)$ *and* $\partial h / \partial z_k|_{z_k^*}$

    solve SOCP - *compute solution for Problem 9*

    if $(J_{tr}(\bar{z}) < \epsilon_{tr}) \wedge (J_{vc}(\bar{\nu}) < \epsilon_{vc})$ **then**

        `converged` $= 1$

    else

        $z_k^* \leftarrow z_k$ *for all* $k \in \mathcal{K}$

    end if

end while

return - *computed solution* z_k *for all* $k \in \mathcal{K}$

5.5.1 Scenario 1

For this scenario, the performance of Algorithm 4 was evaluated by running multiple cases with identical boundary conditions and final time, but with different hoop locations and orientations. Example simulations are presented in Figure 5.3, which shows 13 trajectories obtained using a temporal resolution of $K = 30$, randomized hoop orientations, and a manually selected grid of hoop positions. The hoop positions, tilt angles, and heading angles were sampled from uniform random distributions given by $r_h \in [-1, 1] \times [-2, 2] \times [2, 4]$ m, $\phi \in [-25°, 25°]$, and $\psi \in [-35°, 35°]$, respectively. Other problem parameters were set to the

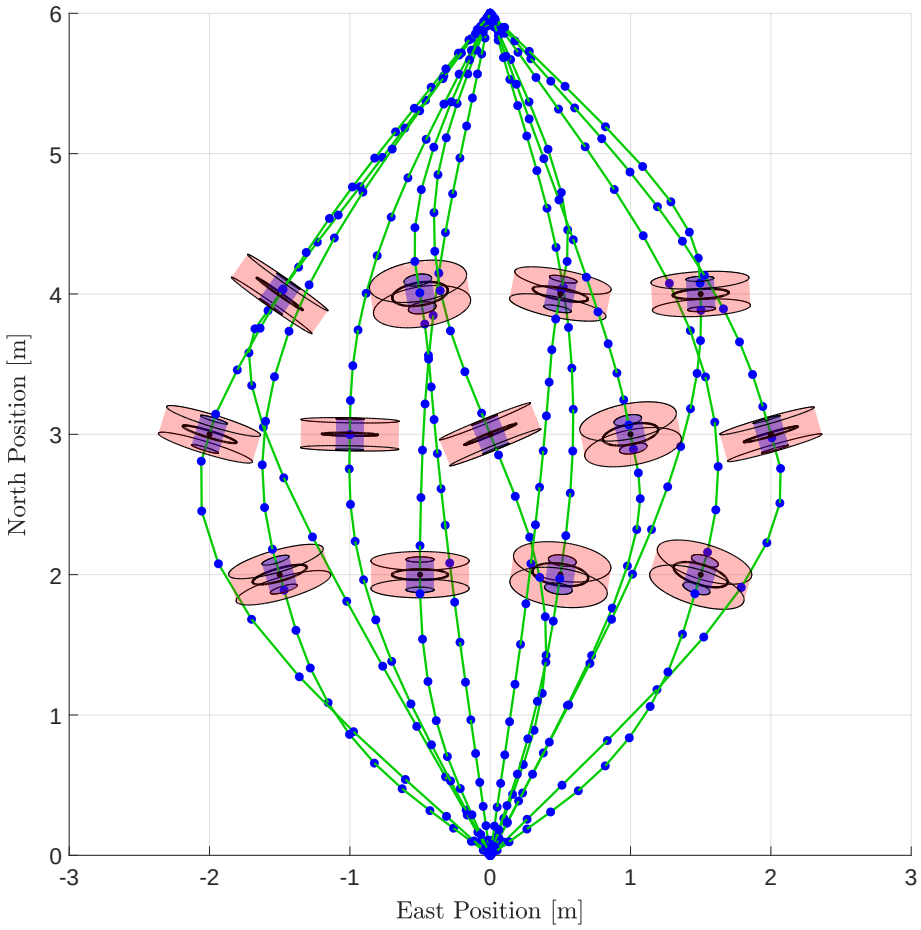


Figure 5.3: Scenario 1 sample trajectories for $K = 30$ using a grid of hoop positions and randomized hoop orientations. Vehicle motion is from top to bottom. The blue dots represent temporal nodes, and the trigger and constraint corridors are not drawn to scale. Each trajectory is constrained to pass through one of the hoops by omitting $g_3(\cdot)$ from (5.5)-(5.6).

following constant values:

$$t_f = 4 \text{ s}, \quad , \quad r_i = (0, 0, 0) \text{ m},$$

$$r_f = (0, 0, 6) \text{ m}, \quad \ell_c = 0.5 \text{ m}, \quad \rho_c = 0 \text{ m},$$

$$\rho_g = \infty \text{ m}, \quad v_{max} = 2\ell_c/\Delta t \text{ m/s}, \quad T_{min} = 2 \text{ N},$$

$$T_{max} = 5 \text{ N}, \quad \theta_{max} = 45^\circ, \quad m = 0.35 \text{ kg},$$

$$g = 9.81 \text{ m/s}^2, \quad k_d = 0 \text{ s}^{-1}, \quad W_{vc} = 10^5 \cdot I_{6\times6},$$

$$W_{tr} = 0.1 \cdot \textbf{blkdiag}\left(I_{3\times3}, 0_{6\times6}\right).$$

Monte Carlo batch runs were conducted for temporal resolutions $K = 15, 20, 25, 30$. Each batch consisted of 100 cases. Cases that resulted in infeasibility were noted, and new cases were sampled in their place. Tables 5.1 and 5.2 provide statistics for Algorithm 4 runtime and inter-sample constraint violation. As expected, the runtime increased with K since the number of decision variables in Problem 9 increased. Nevertheless, the algorithm demonstrated the ability to run at interactive rates even for $K = 30$, with a worst-case observed execution rate of approximately 3 Hz.

Additionally, it was observed that inter-sample constraint violation decreased as K increased. This was expected, since a smaller Δt reduces the possibility of the quad-rotor "hopping" over the hoop in one discrete time step.

Lastly, our simulations resulted in 400 converged cases, and 19 failures to converge. In each case, the cause of infeasibility was an excessively short final time, and feasibility was recovered by selecting a larger t_f.

Table 5.1: Algorithm 4 runtime for Scenario 1 [ms].

K	Mean	Median	Std. Dev.	Min	Max
15	34	31	25	15	151
20	43	34	32	19	219
25	55	42	47	24	306
30	67	52	53	29	337

5.5.2 Scenario 2

For Scenario 2, the performance of Algorithm 4 was evaluated by running multiple cases with identical terminal conditions and final times, but with different initial positions, initial formation angles, and obstacle configurations. Example simulations are presented in Figure 5.4, which shows three sets of trajectories obtained with a temporal resolution of $K = 30$ and a manually positioned configuration of $N_o = 5$ obstacles. In what follows, recall that Scenario 2 is two-dimensional in nature, and is thus implemented using only two spatial dimensions.

Monte Carlo batch runs were conducted as in §5.5.1. The initial positions and angles of the formation were sampled from uniformly random distributions given by $(r_{i,1} + r_{i,2})/2 \in [-2, 2] \times [0, 1]$ and $\psi \in [-70°, 70°]$. Each case had $N_o = 4$ obstacles, whose positions $r_{o,\{1,2,3,4\}}$ were sampled from a uniform random distributions of positions $r_{o,l} \in [-2, 2] \times [2, 4.5]$ m. In generating the obstacle configuration, an inter-obstacle 1-norm distance of at least $\max\{0.8(\ell_o + 2w_o), 2w_o\}$ m was enforced. This heuristic encouraged the obstacle configuration to be feasibly navigable by the quad-rotor pair. To make sure that the obstacles obstruct a straight path from the initial to the final payload position, one of the obstacles

Table 5.2: Constraint clipping for Scenario 1 [cm].

K	Mean	Median	Std. Dev.	Min	Max
15	37	32	37	2	224
20	15	10	14	1	100
25	6	6	3	1	16
30	4	4	2	0	10

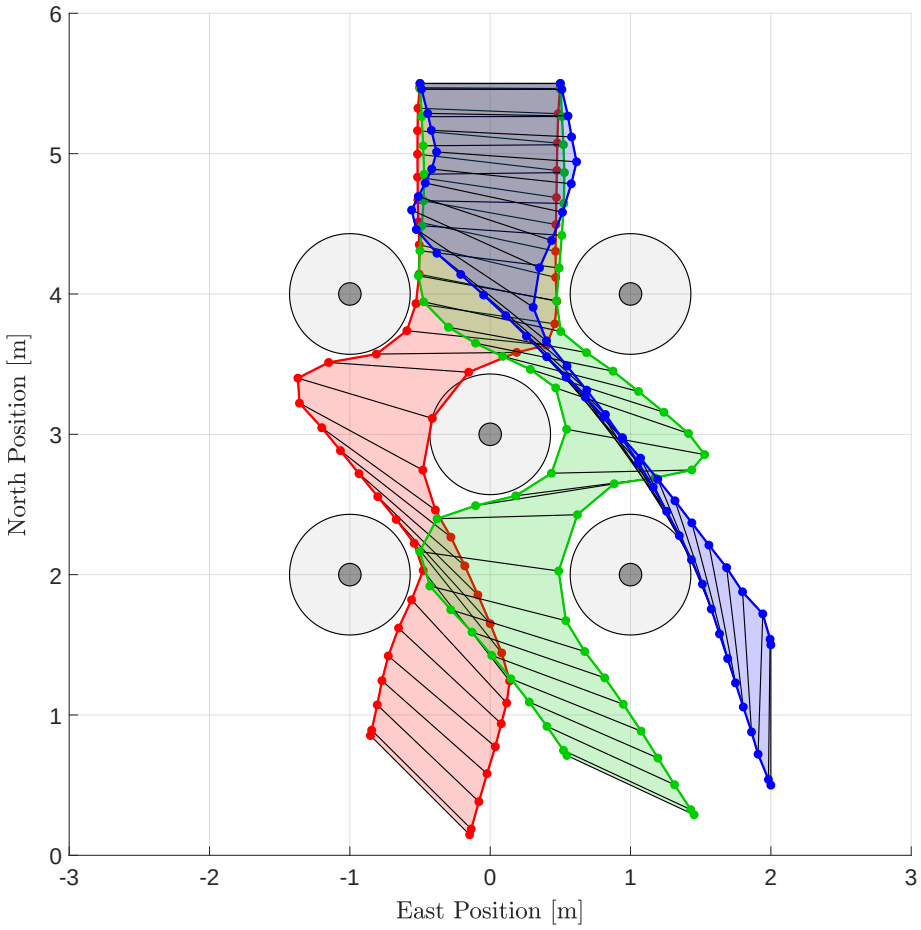


Figure 5.4: Scenario 2 sample trajectories for $K = 30$ and a set of $N_o = 5$ obstacles. Vehicle motion is from top to bottom. Each color represents a separate run, and the dots and black lines represent the vehicles and payload, respectively. The dark gray circles represent the obstacles, while the light gray circle represent the effective keep-out areas. A minor inter-sample constraint violation is observed between the red case and the center obstacle.

was always placed along the segment from $(r_{i,1} + r_{i,2})/2$ to $(r_{f,1} + r_{f,2})/2$. Other problem

Table 5.3: Algorithm 4 runtime for Scenario 2 [ms].

K	Mean	Median	Std. Dev.	Min	Max
15	59	53	22	36	157
20	138	127	50	67	294
25	204	192	76	112	571
30	243	212	100	132	693

parameters were set to the following constant values:

$$t_f = 4 \text{ s}, \quad r_{f,1} = (-0.5, 5.5) \text{ m},$$
$$r_{f,2} = (0.5, 5.5) \text{ m}, \quad \ell_o = 1 \text{ m}, \quad m = 0.35 \text{ kg},$$
$$v_{max} = 3 \text{ m/s}, \quad T_{max} = 5 \text{ N}, \quad R_o = 0.08 \text{ m},$$
$$w_o = 0.43 \text{ m}, \quad g = 9.81 \text{ m/s}^2, \quad W_{vc} = 10^5 \cdot I_{8\times 8},$$
$$W_{tr} = 50 \cdot \mathbf{blkdiag}\left(I_{8\times 8}, 0_{4\times 4}\right).$$

Tables 5.3 and 5.4 provide runtime and inter-sample constraint violation statistics. For the same reasons given in Scenario 1, runtime increased and constraint clipping decreased for larger K. However, because Scenario 2 is a larger and more nonlinear problem, Algorithm 4 is less capable of running at interactive rates, with a worst-case observed solve rate of approximately 1.5 Hz for $K = 30$. Furthermore, we note that inter-sample constraint violation occurred almost persistently for $K = 15$, where the discretized quad-rotor pair was able to "jump" through an obstacle in a single discrete time step without violating (5.9). Therefore, in this scenario it appears necessary to either have $K \geq 25$ or to decrease v_{max} in

Table 5.4: Constraint clipping for Scenario 2 [cm].

K	Mean	Median	Std. Dev.	Min	Max
15	42	42	4	6	43
20	5	4	6	0	42
25	2	< 0.5	2	0	10
30	< 0.5	0	1	0	5

order to avoid constraint clipping. Finally, 31 infeasible cases were encountered during the Monte-Carlo simulation, all of which can be remedied by increasing t_f.

5.6 Conclusion & Future Work

The main contribution of this chapter is the application of the recently introduced compound state-triggered constraints to quad-rotor path planning. Two quad-rotor scenarios are outlined, one involving a quad-rotor flying through a hoop, and the second involving two quad-rotors cooperatively negotiating an obstacle course while carrying a beam-like payload. Our simulation results indicate that solutions to both scenarios can be obtained in real-time. Future work will focus on the implementation the proposed methodology on the Autonomous Control Lab's in-house quad-rotor platforms.

Chapter 6

A HIGH PERFORMANCE MULTI-ROTOR FEEDBACK CONTROL ARCHITECTURE

6.1 Introduction

The objective of this chapter is to outline a high performance under-actuated multi-rotor feedback control architecture that complements a variety of existing feed-forward guidance methodologies. By high performance, we mean that the feedback control architecture makes the vehicle robust to extrinsic disturbances and intrinsic plant uncertainties, while also providing good transient behavior *over the entire maneuvering envelope*. By under-actuated, we imply that the multi-rotors we consider are capable of generating only one force and three torques, and thus cannot simultaneously control the six degrees-of-freedom along which they evolve. By saying a variety of feed-forward guidance methodologies, we include methodologies that produce high fidelity 6-DoF trajectories [68], medium fidelity 3-DoF trajectories [88], and low fidelity manually specified position trajectories.

The proposed architecture is comprised of a hierarchy of control layers. The outermost layer, called the *high-level controller*, is commanded by a specified feed-forward trajectory, and determines the desired pointing direction and thrust required to remain on the trajectory. The middle layer, called the *mid-level controller*, is commanded by the high-level controller, and determines the angular velocity required to achieve the desired attitude. The innermost layer, called the *low-level controller*, is commanded by the mid-level controller, and determines the collective and torque commands required to achieve the desired force and angular velocity. The output of the low-level controller is then passed to an allocator and converter, which together convert the force and torque commands into individual motor commands that are communicated to and implemented by the hardware.

The structure of this hierarchy is nothing new, and elements of it can readily be found in the in the existing literature (e.g. [36, 67]). Instead, the key novelty – and the bulk of the performance – of the proposed architecture can be attributed to the design and interplay between the low-level controller and the allocator, which together address the issues of stability, disturbance rejection, robustness to model uncertainty, and transient performance in the presence of coupled control channels and actuator saturation. These issues are largely disregarded in the existing literature, and thus lead to control strategies that cannot fully and safely exploit the capabilities of the vehicle.

6.2 Overview

Figure 6.1 provides a high-level layout of the proposed feedback control architecture and its interaction with the system hardware. The feedback control architecture is outlined in blue and is comprised of the high-level controller, mid-level controller, low-level controller, allocator, and converter blocks. All of these blocks are implemented digitally in software.

The system hardware is outlined in red and is comprised of the actuator and vehicle dynamics system blocks, and the accelerometer, gyroscope, and motion capture sensor blocks. Notably, the actuator block represents the n_m independently controlled actuators of the multi-rotor, where we assume $n_m \geq 4$.

The software and hardware interact through the actuator command vector u, and the sensed position, attitude, angular velocity, and acceleration, denoted by $r_{\mathcal{I}}^{(s)}$, $q_{\mathcal{B}/\mathcal{I}}^{(s)}$, $\omega_{\mathcal{B}}^{(s)}$, and $a_{\mathcal{B}}^{(s)}$, respectively. External disturbances enter the system through the disturbance wrench vector d_{w}, and are typically a result of aerodynamic effects. Sensor noise corrupts the measured quantities through n_a, n_ω, n_q, and n_r. Uncertainties in the system are typically dominated by unmodeled aerodynamic interactions and dynamics of the actuators.

The control architecture is commanded by a trajectory specified through five inputs entering the high-level controller. These inputs are an inertial position, a specific force, a reference attitude, an angular velocity, and a torque. These quantities are denoted by $r_{\mathcal{I}}^{(h)}$, $f_s^{(h)}$, $q_{\mathcal{H}/\mathcal{I}}^{(h)}$, $\omega_{\mathcal{H}}^{(h)}$, and $\tau_{\mathcal{H}}^{(h)}$, respectively.

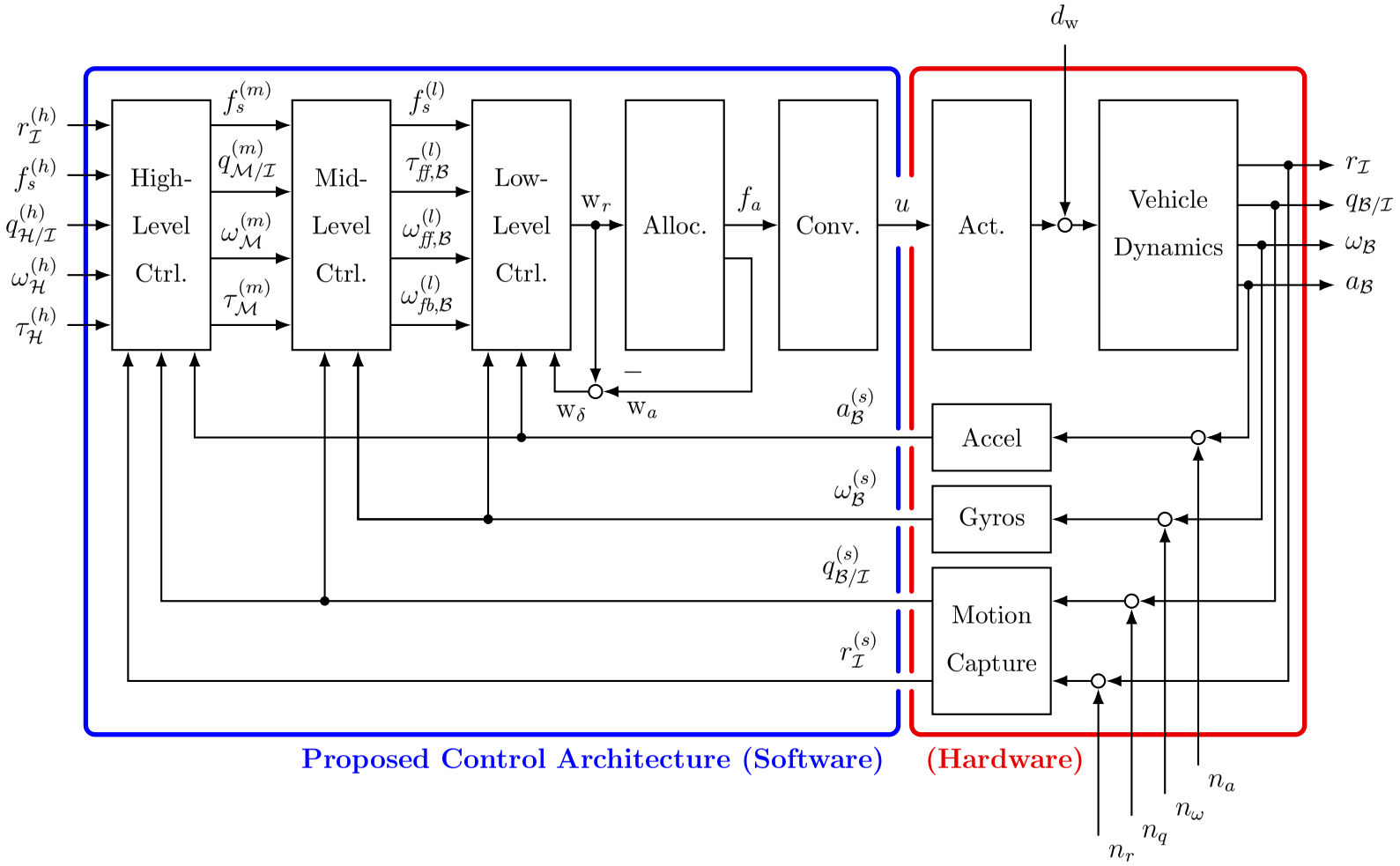


Figure 6.1: This figure shows a high-level view of the proposed feedback control architecture. The feedback control architecture is implemented in software, and is shown in blue. The system hardware interacts with this control architecture through sensors and actuators, and is shown in red.

6.2.1 Conventions & Nomenclature

The superscript labels (h), (m), and (l) are used to denote quantities associated with inputs to the high-level, mid-level, and low-level control blocks, respectively. The superscript label (s) is used to denote sensed quantities output by the sensor blocks.

Four primary coordinate frames are defined: the *inertial frame*, the *high-level frame*, the *mid-level frame*, and the *body frame*. Quantities expressed in these frames are indicated by the subscripts $\mathcal{I}$, $\mathcal{H}$, $\mathcal{M}$, and $\mathcal{B}$, respectively. The $\mathcal{I}$-frame is a ground-fixed coordinate frame, whereas the $\mathcal{B}$-frame is body-fixed and centered at the vehicle's center of mass. The $\mathcal{H}$- and $\mathcal{M}$-frames are used to express quantities in intermediate frames used at the inputs of the high-level and mid-level control blocks, respectively. Without loss of generality, we adopt a North-East-Down coordinate convention so that rolling right, pitching up, and yawing right all result in positive body rates. One undesirable consequence of this convention is that the positive thrust direction of the vehicle is aligned with the negative z-axis of the $\mathcal{B}$-frame.

The attitude of a frame $\mathcal{F}_a$ with respect to frame $\mathcal{F}_b$ is parameterized using a unit quaternion $q_{\mathcal{F}_a/\mathcal{F}_b} \in \mathcal{S}^3$, where $\mathcal{S}^3 \subset \mathbb{R}^4$ is the unit 3-sphere. We use a scalar-first convention such that a quaternion q is expressed as

$$q = \begin{bmatrix} q_0 & q_v^\mathsf{T} \end{bmatrix}^\mathsf{T} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^\mathsf{T}.$$

By this convention, the frames $\mathcal{F}_a$ and $\mathcal{F}_b$ are aligned when $q_{\mathcal{F}_a/\mathcal{F}_b} = [1 \;\; 0 \;\; 0 \;\; 0]^\mathsf{T}$. We define the quaternion conjugate $\bar{q}$ as

$$\bar{q} \coloneqq \begin{bmatrix} q_0 & -q_v^\mathsf{T} \end{bmatrix}^\mathsf{T} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}^\mathsf{T},$$

and quaternion multiplication between the quaternions p and q as

$$p \otimes q \coloneqq \begin{bmatrix} p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 \\ p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2 \\ p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1 \\ p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0 \end{bmatrix}.$$

In §6.6, we assume that the a product of quaternion multiplication is negated if its scalar element is negative. We refer to this operation as a reflection into the positive quaternion hemisphere.

The angular velocity of frame $\mathcal{F}_a$ with respect to frame $\mathcal{F}_b$ expressed in frame $\mathcal{F}_a$ coordinates is denoted by $\omega_{\mathcal{F}_a} \in \mathbb{R}^3$. Sensed-acceleration[1] and torque vectors are denoted by $a \in \mathbb{R}^3$ and $\tau \in \mathbb{R}^3$, respectively. The position and velocity of the vehicle are denoted by $r \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$, and are typically expressed in $\mathcal{I}$-frame coordinates.

The scalar quantity f_s is used to represent the specific-force commanded along the positive thrust direction of the vehicle (i.e. the acceleration due to thrust projected along the negative z-axis). For multi-rotors quipped with fixed-pitch variable-RPM propellers, the quantity f_s is non-negative. However, vehicles equipped with variable-pitch propellers are able to realize negative values of f_s as well [26].

We use $\mathrm{w} \in \mathbb{R}^4$ to denote a four-dimensional wrench vector formed by concatenating a scalar force element with a torque 3-vector, and the subscripts r and a to denote *requested* and *allocated* quantities. Both w_r and w_a are always expressed in the body-frame, and thus are used without the subscript $\mathcal{B}$. We use $u \in \mathbb{R}^{n_m}$ to represent an n_m-dimensional actuator command vector.

The variable $d_{\mathrm{w}} \in \mathbb{R}^6$ represents a six-dimensional disturbance wrench formed by concatenating disturbance force and torque 3-vectors. Both of these disturbance vectors are represented in $\mathcal{B}$-frame coordinates. The quantities $n_a \in \mathbb{R}^3$, $n_\omega \in \mathbb{R}^3$, $n_q \in \mathcal{S}^3$, and $n_r \in \mathbb{R}^3$ represent noise quantities that corrupt the accelerometer, gyroscope, and motion capture measurements.

We use the term *tilt angle* to refer to the angle between the z-axes of $\mathcal{I}$ and $\mathcal{B}$. We use the term *clock angle* to refer to the orientation of the vehicle around the z-axis of $\mathcal{B}$.[2]

Summation nodes in figures are represented by small circles, where a minus sign signifies

[1] In this context, the word *sensed* is meant to distinguish between sensed and inertial acceleration. For example, a sensor sitting on a table would measure a sensed acceleration of 1 g despite being subjected to zero inertial acceleration.

[2] The clock angle is equivalent to the yaw angle when the tilt angle is zero.

subtraction of an input, and addition is assumed otherwise. In Figure 6.1, summation nodes with quaternion inputs and outputs are understood as quaternion multiplication. We use 1 to denote the vector of ones, and leave its size to be interpreted from context. Lastly, we use denote the boundary set of a set $\mathcal{P}$ by $\partial\mathcal{P}$.

6.2.2 Vehicle Dynamics Block

We regard the vehicle as a rigid body subject to both translational and rotational motion. The vehicle's attitude dynamics are governed by

$$\dot{q}_{\mathcal{B}/\mathcal{I}}(t) = \frac{1}{2}\Omega(\omega_{\mathcal{B}})q_{\mathcal{B}/\mathcal{I}}, \tag{6.1}$$

$$J_{\mathcal{B}}\dot{\omega}_{\mathcal{B}}(t) = d_\tau + \tau_{m,\mathcal{B}} - [\omega_{\mathcal{B}}\times]\, J_{\mathcal{B}}\, \omega_{\mathcal{B}}, \tag{6.2}$$

where $\Omega(\cdot)$ is a 4×4 skew-symmetric matrix of $\omega_{\mathcal{B}}$, $J_{\mathcal{B}} \in \mathbb{S}^3_{++}$ is the vehicle's inertia tensor, $d_\tau \in \mathbb{R}^3$ is the torque component of the disturbance wrench d_{w}, and $\tau_{m,\mathcal{B}} \in \mathbb{R}^3$ is the vector of propulsive torques. The vehicle's translational dynamics are given by

$$\dot{r}_{\mathcal{I}}(t) = v_{\mathcal{I}}, \tag{6.3}$$

$$\dot{v}_{\mathcal{I}}(t) = \frac{1}{m}C_{\mathcal{B}/\mathcal{I}}\big(d_f + u_{m,\mathcal{B}}\big) + g, \tag{6.4}$$

where $C_{\mathcal{B}/\mathcal{I}}$ is the direction-cosine-matrix associated with $q_{\mathcal{B}/\mathcal{I}}$, $m \in \mathbb{R}_{++}$ is the vehicle's mass, $g \in \mathbb{R}^3$ is the gravitational acceleration vector, $d_f \in \mathbb{R}^3$ is the force component of d_{w}, and $f_{m,\mathcal{B}} \in \mathbb{R}^3$ is the propulsive force vector. In our treatment, we regard aerodynamic forces and torques as disturbances that enter the system through d_{w}.

6.2.3 Actuators Block

The actuator block represents the n_m actuators of the vehicle. Each actuator constitutes a spinning motor with a propeller. Our treatment allows for both variable-pitch fixed-RPM actuators and fixed-pitch variable-RPM actuators. Each actuator produces a force, or thrust, produced by the lifting force of the propeller blades, and a torque, produced by the drag force of the propeller blades.

We assume that the actuator's force responds like an over-damped second-order system, with one slow pole and one fast pole. The former is associated with the inertia of the rotor and propeller, while the latter is associated with the inductance in the circuitry. We assume that the actuator's torque response is comprised of one zero and two poles. This difference in the torque model stems from the fact that a commanded torque can be sensed by the vehicle virtually instantaneously, since the torque sensed by the vehicle are an equal and opposite reaction to the electromagnetic forces applied by the motor. On the other hand, the force requires that the rotor speed up before a noticeable change in force is realized.

6.2.4 Sensor Blocks

The proposed feedback control architecture utilizes two sensors on-board the vehicle, and accelerometers, and gyroscopes. These measurements can be typically be obtained at rates between 100 and 1000 Hz. The vehicle is also provided attitude and position feedback from a motion capture system. These measurements tend to be very accurate, and are typically available at rates between 50-150 Hz.

The dominant errors in the inertial measurements are caused by accelerometer and gyro biases. For short duration flights, these biases are stable enough of the purpose of control, and can be estimated offline. Assuming the motion capture system is well calibrated, the dominant errors in the motion capture system stem from frame alignment issues. Unsurprisingly, ensuring that the measured frame is well aligned with the body frame improves performance of the system. However, with a properly designed feedback architecture, small variations in these errors can be handled in a robust manner.

We note that other sensors such as magnetometers and GPS can be integrated into the system using various estimation frameworks. However, we consider such matters beyond the scope of this work, and do not address them further.

6.3 Converter

The converter block converts the allocated motor forces computed by the allocator to actuator commands that can be communicated to the hardware. These quantities are denoted by $f_a \in \mathbb{R}^{n_m}$ and $u \in \mathbb{R}^{n_m}$ in Figure 6.1. The vast majority of commercially available multi-rotors use BLDC motors, and most commercially available BLDC electronic speed controllers are commanded via pulse-width-modulated (PWM) signals. Hence, we assume that the elements of the vector $u \in \mathbb{R}^{n_m}$ are PWM commands. We express the elements of f_a in units of Newtons, and the elements of u in units of microseconds.

The conversion between force units and actuator commands has been widely addressed in the literature (e.g. [34, 36, 67]). The relationship between the motor command and the motor thrust is typically assumed to be quadratic. Formally, this relationship is expressed as

$$f_{a,i} = \alpha_2 u_i^2 + \alpha_1 u_i + \alpha_0,$$

for each actuator $i \in \{1, \ldots, n_m\}$. The inverse relationship is obtained by applying the quadratic formula and discarding the negative root:

$$u_i = -\frac{\alpha_1}{2\alpha_2} + \frac{1}{2\alpha_2}\sqrt{\alpha_1^2 - 4\alpha_2(\alpha_0 - f_{a,i})}. \tag{6.5}$$

The constant coefficients α_0, α_1, and α_2 are obtained experimentally, typically through static system identification tests on the motor-battery-propeller combination used on the vehicle.

The practical value of the conversion in (6.5) is the removal of nonlinearity from the control loop. Due to variability in battery voltage, motor heating, motor dynamics, and aerodynamics, this simplistic conversion is not constant over the entire duration of a flight. A significant amount of effort can be devoted to the development of more sophisticated models, which if done properly, can improve the transient behavior of the system. However, such efforts can become intractable quickly, and are inherently less robust due to their open-loop nature. Therefore, herein we use the simple conversion model given in (6.5), and rely on feedback control to compensate for variations in a robust manner.

6.4 *Allocator*

Low-level control of an under-actuated multi-rotor is performed using wrench commands composed of a force and three torques. The allocator is responsible for determining a physically realizable *allocated wrench* $\mathrm{w}_a \in \mathbb{R}^4$ that most closely matches a *requested wrench* $\mathrm{w}_r \in \mathbb{R}^4$ generated by the low-level control block in Figure 6.1. The vector w_a is related to the vehicle's n_m motor forces through the linear relationship

$$\mathrm{w}_a = H_{\mathrm{w}} f_a, \tag{6.6}$$

where $f_a \in \mathbb{R}^{n_m}$ is the vector containing the corresponding allocated motor forces, and $H_{\mathrm{w}} \coloneqq \begin{bmatrix} h_c & h_x & h_y & h_z \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{4\times n_m}$ is a constant coupling matrix. The vectors h_c, h_x, h_y, $h_z \in \mathbb{R}^{n_m}$ are derived from vehicle geometry and propeller characteristics. For convenience we define the following matrices:

$$\begin{aligned} H_{cz} &\coloneqq [\, h_c \;\; h_z \,]^{\mathsf{T}} \in \mathbb{R}^{2\times n_m}, \\ H_{xy} &\coloneqq [\, h_x \;\; h_y \,]^{\mathsf{T}} \in \mathbb{R}^{2\times n_m}, \\ H_{cxy} &\coloneqq [\, h_c \;\; h_x \;\; h_y \,]^{\mathsf{T}} \in \mathbb{R}^{3\times n_m}, \\ H_{xyz} &\coloneqq [\, h_x \;\; h_y \;\; h_z \,]^{\mathsf{T}} \in \mathbb{R}^{3\times n_m}. \end{aligned}$$

The elements of w_a and w_r constitute the vehicle's *control degrees-of-freedom* (DoFs), while the elements of f_a constitute the vehicle's *actuator degrees-of-freedom*. The under-actuated vehicles considered herein nominally have four control DoFs, and therefore must have at least four actuator DoFs (i.e. $n_m \geq 4$). Moreover, the matrix H_{w} is surjective only if the vectors h_c, h_x, h_y, and h_z are linearly independent, and can be bijective only if $n_m = 4$.[3]

The first element of the wrench vector is called the *collective*, and represents the total thrust produced by the vehicle. The remaining three elements of the wrench vector represent the torques acting on the body of the vehicle. We denote these four control DoFs by

[3] H_{w} is not necessarily bijective if $n_m = 4$. For instance, consider a hexa-rotor with a failed pair of opposing motors. The vehicle remains with four operational motors, but only three control DoFs.

c, x, y, $z \in \mathbb{R}$, and define the allocated and requested wrench vectors as

$$\mathrm{w}_r \coloneqq [\, c_r \;\; x_r \;\; y_r \;\; z_r \,]^\mathsf{T},$$
$$\mathrm{w}_a \coloneqq [\, c_a \;\; x_a \;\; y_a \;\; z_a \,]^\mathsf{T}.$$

We say that a vector of motor forces $f \in \mathbb{R}^{n_m}$ is feasible if

$$f \in \mathcal{F}_m \coloneqq \left\{ x \in \mathbb{R}^{n_m} : \; x_i \in [\, f_{min}, f_{max} \,], \;\; \forall\, i \in \{1, \ldots, n_m\} \right\},$$

where $f_{min} < f_{max}$. Since f_{min} is allowed to be negative, our algorithm applies to both vehicles with variable-pitch fixed-RPM actuators and vehicles with fixed-pitch variable-RPM actuators. For convenience, we introduce the complete allocation feasibility sets

$$\bar{\mathcal{F}}_{xy} \coloneqq \left\{ (x, y) : \; \begin{bmatrix} x & y \end{bmatrix}^\mathsf{T} = H_{xy}\, f, \;\; f \in \mathcal{F}_m \right\},$$
$$\bar{\mathcal{F}}_{cz} \coloneqq \left\{ (c, z) : \; \begin{bmatrix} c & z \end{bmatrix}^\mathsf{T} = H_{cz}\, f, \;\; f \in \mathcal{F}_m \right\}.$$

and the partial allocation feasibility sets

$$\mathcal{F}_{xy}(z) \coloneqq \left\{ (x, y) : \; \begin{bmatrix} x & y & z \end{bmatrix}^\mathsf{T} = H_{xyz}\, f, \;\; f \in \mathcal{F}_m \right\},$$
$$\mathcal{F}_{cz}(x, y) \coloneqq \left\{ (c, z) : \; \begin{bmatrix} c & x & y & z \end{bmatrix}^\mathsf{T} = H_{\mathrm{w}}\, f, \;\; f \in \mathcal{F}_m \right\},$$
$$\mathcal{F}_{c}(x, y, z) \coloneqq \left\{ c : \; \begin{bmatrix} c & x & y & z \end{bmatrix}^\mathsf{T} = H_{\mathrm{w}}\, f, \;\; f \in \mathcal{F}_m \right\},$$
$$\mathcal{F}_{z}(c, x, y) \coloneqq \left\{ z : \; \begin{bmatrix} c & x & y & z \end{bmatrix}^\mathsf{T} = H_{\mathrm{w}}\, f, \;\; f \in \mathcal{F}_m \right\}.$$

In the following, we highlight the assumptions and properties of the algorithm, and describe the algorithm in the context of a generic under-actuated multi-rotor.

6.4.1 Assumptions

Assumption 1: We assume that each element of h_z has the same magnitude, and that $1^\mathsf{T} h_z = 0$. Most multi-rotors satisfy this assumption, which implies that the vehicle has symmetric yaw-torque capabilities. This assumption implies that n_m is even.

Assumption 2: For tractability, we neglect motor dynamics in the development of this algorithm. Consequently, the algorithm is memoryless, and represents a static mapping between w_r and w_a.

Assumption 3: We assume that $3 \leq \mathrm{rank}\,(H_{\mathrm{w}}) \leq 4$. Further, if $\mathrm{rank}\,(H_{\mathrm{w}}) = 3$, then we assume that h_z is in the orthogonal complement of the null set of H_{xy}.

6.4.2 Properties

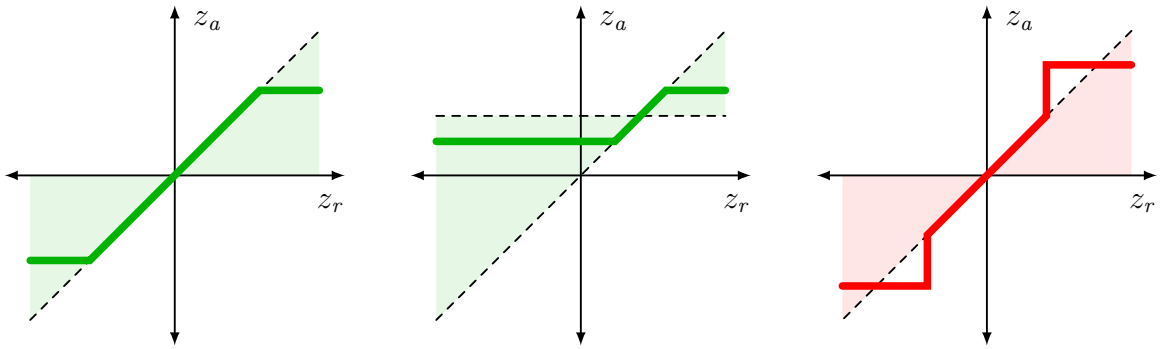
Property 1: In general, the allocator regards x and y as the highest priority, c as the intermediate priority, and z as the lowest priority. The low prioritization of z follows from the fact that translational control of a multi-rotor is not affected by attitude errors resulting from rotations about the desired thrust direction. The prioritization of c follows from the fact that the direction of the thrust is more important than the magnitude of the thrust over short time-scales. This ordering is accepted as the preferable prioritization of control DoFs for an under-actuated multi-rotor [21, 34].

Property 2: Like the mixer presented in [34], the proposed allocation algorithm is capable of guaranteeing values of z within a specified band of yaw torques. We refer to this feature as *yaw prioritization*. This feature is useful in applications where on-board sensors impose pointing constraints. The width of this band is modulated by a user-specified parameter $z_{rsv} \in \mathbb{R}$, and is upper-bounded by the maximum achievable yaw torque magnitude, denoted by $z_{max} \in \mathbb{R}_{++}$. Yaw prioritization is enabled when $z_{rsv} \geq 0$, and is disabled otherwise. The yaw prioritization logic is captured by the following implications

$$|z_r| \leq \beta_z \implies z_a = z_r, \tag{6.8a}$$

$$|z_r| > \beta_z \implies \begin{cases} \alpha_z |z_a| \geq \beta_z, \\ \mathrm{sign}\,(\alpha_z z_a) = \mathrm{sign}\,(\alpha_z z_r)\,, \end{cases} \tag{6.8b}$$

where $\alpha_z \coloneqq \max\,(z_{rsv}, 0)$ and $\beta_z \coloneqq \min\,(z_{rsv}, z_{max})$. We see that when yaw prioritization is enabled, the allocated yaw torque is set to the requested yaw torque up to a guaranteed magnitude of $\min\,(z_{rsv}\,, z_{max})$. If the requested yaw torque exceeds this threshold, larger yaw



(a) (b) (c)

Figure 6.2: Examples of allocator's saturation property. Case (a) and (b) show acceptable saturation-like relationship between z_r and z_a for fixed c_r, x_r, and y_r. Case (c) shows an unacceptable non-saturation-like response.

torques may be allocated, as dictated by the prioritization of Property 1. When yaw prioritization is disabled, the condition on the left of (6.8a) is never satisfied, and the implication in (6.8b) is satisfied trivially. In this case, the algorithm performs its z-allocation according the prioritization outlined in Property 1.

Property 3: The allocator preserves the direction of the requested xy-torque vector in order to better interface with the low-level controller. This property the low-level control response more predictable, and reduces adverse cross-channel interactions in the face of actuator saturation.

Property 4: The allocator preserves a saturation-like response in each of the four control DoFs. Specifically, if any three of four control DoFs in w_r are fixed, then the relationship between the requested and allocated values of the fourth control DoF resembles a saturation function. For example, assuming that c_r, x_r, and y_r have been specified, the allocator ensures that the relation between z_a and z_r resembles the function shown in Figure 6.2a or 6.2b, but not the function shown in Figure 6.2c. This property ensures that there exists a set point where the nonlinearity in the loop satisfies property (6.37) assumed by the low-level controllers. Thus, this property plays a key role in ensuring that the low-level controllers

remain stable in the face of actuator saturation.

Property 5: The allocator can function in a degraded mode of operation where a linear dependence between h_z and the vectors h_x and h_y results in $\text{rank}\,(H_{\text{w}}) = 3$. This case results is experienced by hexa-rotors with a failed pair of opposing motors [31]. In this degraded mode, one of the original control DoFs becomes dependent on the remaining three (independent) control DoFs. For reasons discussed above, typically yaw torque is designated as the dependent control DoF.

6.4.3 Algorithm

A summary of the allocator is presented in Algorithm 5 for a generic under-actuated multi-rotor vehicle. Its inputs consist of the requested wrench vector w_r, and the constants H_{w}, z_{max}, and z_{rsv}. Its outputs consist of the allocated motor force vector f_a and the corresponding allocated wrench vector w_a. At the core of the algorithm is a sequence of constrained optimization problems, specifically Linear Programs (LPs), that are solved with a lexicographic ordering [1]. The lexicographic ordering is responsible for embedding the prioritization outlined in Property 1 of §6.4.2. The details of each step of the algorithm are given in §6.4.4-§6.4.10.

6.4.4 Pre-Allocation (Step 1)

This step computes the set of xy-allocations that admits a feasible z-allocation that satisfies (6.8) (and thus Property 2 of §6.4.2). In light of the fact that the xy-allocation precedes the z-allocation (per Property 1 of §6.4.2), the purpose of this step is to ensure that the former does not compromise the latter.

The xy-allocations which admit feasible z-allocations that satisfy (6.8) are given by $(x_a, y_a) \in \mathcal{F}_{xy}(\bar{z})$, where

$$\bar{z} := \text{sign}\,(z_r)\,\min\left(|z_r|, \max\left(\beta_z, 0\right)\right). \tag{6.9}$$

Algorithm 5: ***Under-Actuated Multi-Rotor Allocator***

given: $\mathrm{w}_r \coloneqq [\, c_r \;\; x_r \;\; y_r \;\; z_r \,]$, $H_{\mathrm{w}} \in \mathbb{R}^{4 \times n_m}$, $z_{max} \in \mathbb{R}_{++}$, $z_{rsv} \in \mathbb{R}$

step 1: compute $\mathcal{F}_{xy}(\bar{z})$ ▷ (6.9)

step 2: solve $P_{xy}([\, x_r \;\; y_r \,]^{\mathsf{T}}) \to [\, x_a \;\; y_a \,]^{\mathsf{T}}$ ▷ (6.13)

if $(\mathrm{rank}\,(H_{\mathrm{w}}) = 4)$ **then** ▷ Nominal Operation

  if $(z_{rsv} < 0)$ **then**

    step 3.1: solve $P_{c,xy}(c_r, [\, x_a \;\; y_a \,]^{\mathsf{T}}) \to c_a$ ▷ (6.14)-(6.15)

    step 3.2: solve $P_{z,cxy}(z_r, [\, c_a \;\; x_a \;\; y_a \,]^{\mathsf{T}}) \to z_a$ ▷ (6.16)-(6.17)

  else

    if $(|z_r| \leq \beta_z)$ **then**

      step 4.1: set $z_r \to z_a$ ▷ (6.19)

      step 4.2: solve $P_{c,xyz}(c_r, [\, x_a \;\; y_a \;\; z_a \,]^{\mathsf{T}}) \to c_a$ ▷ (6.20)-(6.21)

    else

      step 5.1: solve $\bar{P}_{c,xy}(c_r, [\, x_a \;\; y_a \,]^{\mathsf{T}}, z_r, \bar{z}) \to c_a$ ▷ (6.22)-(6.24)

      step 5.2: solve $\bar{P}_{z,cxy}(z_r, [\, c_a \;\; x_a \;\; y_a \,]^{\mathsf{T}}, \bar{z}) \to z_a$ ▷ (6.25)-(6.26)

    end if

  end if

  step 7.1: set $H_{\mathrm{w}}^{\mathsf{T}}(H_{\mathrm{w}}H_{\mathrm{w}}^{\mathsf{T}})^{-1}\,\mathrm{w}_a \to f_a$ ▷ (6.28)

else if $(\mathrm{rank}\,(H_{\mathrm{w}}) = 3)$ **then** ▷ Degraded Operation

  step 6.1: solve $P_{c,xy}(c_r, [\, x_a \;\; y_a \,]^{\mathsf{T}}) \to c_a$ ▷ (6.14)-(6.15)

  step 6.2: set $h_c^{\mathsf{T}} H_{cxy}^{\mathsf{T}}(H_{cxy}H_{cxy}^{\mathsf{T}})^{-1}\,\mathrm{w}_{a,cxy} \to z_a$ ▷ (6.27)

  step 7.2: set $H_{cxy}^{\mathsf{T}}(H_{cxy}H_{cxy}^{\mathsf{T}})^{-1}\,\mathrm{w}_{a,cxy} \to f_a$ ▷ (6.29)

end if

return $f_a \in \mathbb{R}^{n_m}$ and $\mathrm{w}_a \coloneqq [\, c_a \;\; x_a \;\; y_a \;\; z_a \,]^{\mathsf{T}}$

Further, if H_{w} is such that the following property holds

$$\mathcal{F}_{xy}(z_1) \supseteq \mathcal{F}_{xy}(z_2) \quad \forall\, 0 \leq z_1 \leq z_2 \leq z_{max}, \tag{6.10}$$

then $\mathcal{F}_{xy}(\bar{z})$ does not introduce conservatism into the xy-allocation. If yaw prioritization is enabled and (6.10) is not satisfied, then the xy-allocation may be conservative in the sense that w_a may not be the best achievable xy-allocation. However, if yaw prioritization is disabled, no conservatism is introduced.

To illustrate property (6.10), consider two notional examples with $f_{min} = 0$ and $f_{max} = 1$, one with the quad-rotor mixing matrix

$$H_{quad} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ \frac{3}{2} & -\frac{3}{2} & \frac{3}{2} & -\frac{3}{2} \end{bmatrix}, \tag{6.11}$$

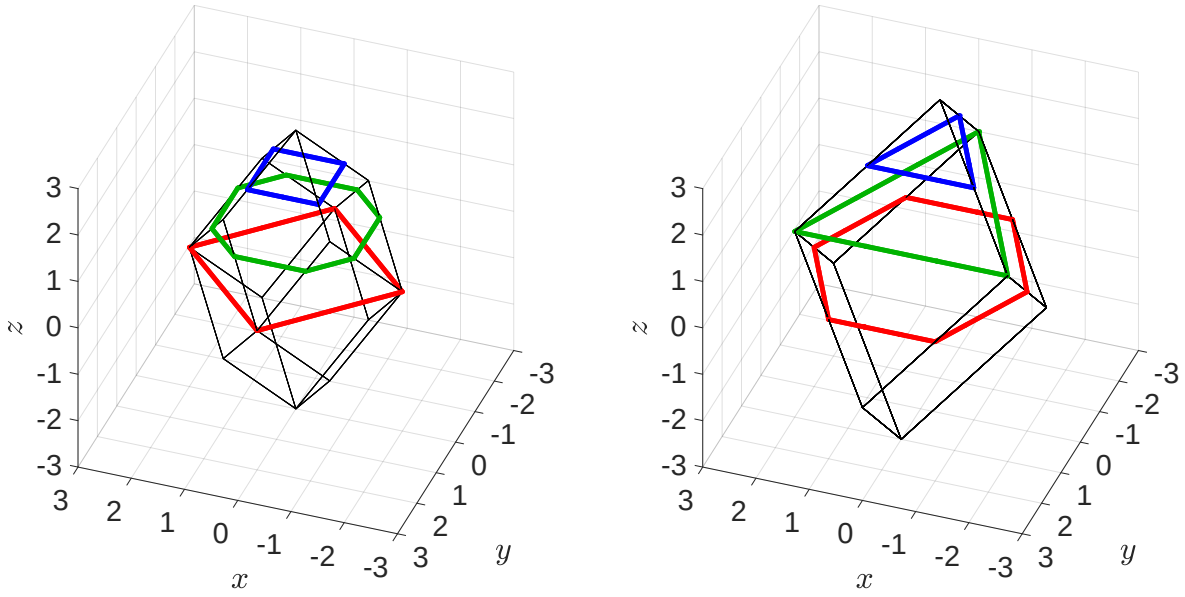
and the other with the hexa-rotor mixing matrix

$$H_{hexa} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & 0 & 1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}. \tag{6.12}$$

Figure 6.3 shows the feasible torque polytopes associated with the two mixing matrices. Figure 6.4 shows an xy-view of the same polytopes. Both figures show the set $\mathcal{F}_{xy}(z)$ for $z = \{0, 1, 2\}$. Evidently, property (6.10) is satisfied for the quad-rotor case, but not for the hexa-rotor case.

6.4.5 XY-Allocation (Step 2)

This step performs the xy-allocation, and preserves the direction of the xy-torque as per Property 3 of §6.4.2. Formally, this allocation takes the form of an LP that operates on a requested xy-torque $\mathrm{w}_{r,xy} \coloneqq [\, x_r \;\; y_r \,]^{\mathsf{T}}$. We represent this LP as $\mathrm{w}_{a,xy} = P_{xy}(\mathrm{w}_{r,xy})$, where



(a) Quad-Rotor (b) Hexa-Rotor

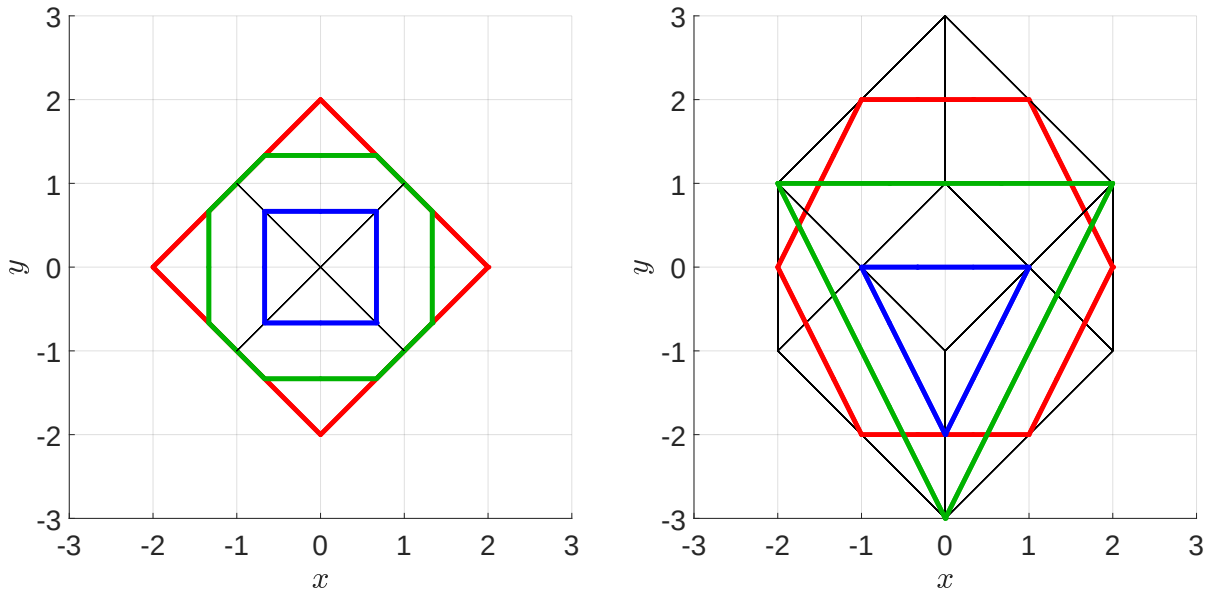
Figure 6.3: Feasible torque polytopes for a notional quad-rotor and hexa-rotor. The boundaries of $\mathcal{F}_{xy}(z)$ for $z = \{0,\ 1,\ 2\}$ are shown in red, green, and blue, respectively.

$\mathrm{w}_{a,xy} \coloneqq [\,x_a \;\; y_a\,]^{\mathsf{T}}$ is the resulting xy-torque allocation, and the function $P_{xy} : \mathbb{R}^2 \to \mathbb{R}^2$ is defined as

$$P_{xy}(\mathrm{w}_{r,xy}) \coloneqq \begin{cases} \underset{\mathrm{w}_{a,xy}}{\text{minimize}} \;\; |\xi|, \\ \text{subject to:} \;\; \mathrm{w}_{a,xy} = H_{xy} f, \\ \mathrm{w}_{a,xy} \in \mathcal{F}_{xy}(\bar{z}),\;\; (1-\xi)\mathrm{w}_{r,xy} = \mathrm{w}_{a,xy}, \end{cases} \tag{6.13}$$

where $\xi \in \mathbb{R}$. The resulting allocation minimizes the quantity $\|\mathrm{w}_{r,xy} - \mathrm{w}_{a,xy}\|_2$ subject to the constraints, thus giving the best possible allocation. Moreover, since

$$\mathrm{w}_{a,xy} \in \mathcal{F}_{xy}(\bar{z}) \subset \bar{\mathcal{F}}_{xy} \quad \forall\, |z| \leq z_{max},$$



(a) Quad-Rotor (b) Hexa-Rotor

Figure 6.4: Illustration of allocator property (6.10) showing an xy-plane view of the polytopes in Figure 6.3. The boundaries of $\mathcal{F}_{xy}(z)$ for $z = \{0,\ 1,\ 2\}$ are shown in red, green, and blue, respectively. The quad-rotor case on the left satisfies property (6.10), whereas the hexa-rotor case on the right does not.

there always exists at least one feasible motor force vector that generates the allocated xy-torque. The solution to (6.13) is given by

$$\mathrm{w}_{a,xy} = \begin{cases} \mathrm{w}_{r,xy}, & \text{if } \mathrm{w}_{r,xy} \in \mathcal{F}_{xy}(\bar{z}), \\ \alpha_{xy}\mathrm{w}_{r,xy}, & \text{otherwise.} \end{cases}$$

where $\alpha_{xy} \in [0, 1]$ is a scalar chosen such that $\alpha_{xy}\mathrm{w}_{r,xy} \in \partial\mathcal{F}_{xy}(\bar{z})$.

6.4.6 Yaw Prioritization Disabled (Step 3)

When yaw prioritization is disabled, the allocator follows the prioritization outlined in Property 1 of §6.4.2. Given the xy-allocation performed in Step 2, we must now perform the c- and z-allocations.

The c-allocation is posed as an LP that operates on a requested collective c_r and a prescribed xy-torque allocation $\mathrm{w}_{a,xy}$. We represent this LP as $c_a = P_{c,xy}(c_r, \mathrm{w}_{a,xy})$, where the function $P_{c,xy} : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$ is given by

$$P_{c,xy}(c_r, \mathrm{w}_{a,xy}) \coloneqq \begin{cases} \underset{c_a}{\text{minimize}} \;\; |c_a - c_r|, \\ \text{subject to:} \;\; c_a = h_c^{\mathsf{T}} f, \\ f \in \mathcal{F}_m, \; \mathrm{w}_{a,xy} = H_{xy} f. \end{cases} \tag{6.14}$$

This LP minimizes the collective allocation error subject to the motor feasibility constraints imposed by $\mathcal{F}_m$ and the prescribed xy-allocation obtained through (6.13). Given (x_a, y_a), the solution to (6.14) can be obtained as follows:

$$c_a = \underset{c}{\operatorname{argmin}} \big\{ |\, c - c_r \,| : (c, z) \in \mathcal{F}_{cz}(x_a, y_a) \big\}. \tag{6.15}$$

The z-allocation is posed as an LP that operates on a requested yaw-torque z_r and a prescribed cxy-wrench allocation $\mathrm{w}_{a,cxy} \coloneqq \begin{bmatrix} c_a & x_a & y_a \end{bmatrix}^{\mathsf{T}}$. We represent this LP as $z_a = P_{z,cxy}(z_r, \mathrm{w}_{a,cxy})$, where the function $P_{z,cxy} : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}$ is given by

$$P_{z,cxy}(z_r, \mathrm{w}_{a,cxy}) \coloneqq \begin{cases} \underset{z_a}{\text{minimize}} \;\; |z_a - z_r|, \\ \text{subject to:} \;\; z_a = h_z^{\mathsf{T}} f, \\ f \in \mathcal{F}_m, \; \mathrm{w}_{a,cxy} = H_{cxy} f. \end{cases} \tag{6.16}$$

This LP minimizes the yaw torque allocation error subject to the motor feasibility constraints imposed by $\mathcal{F}_m$, the prescribed xy-allocation obtained through (6.13), and the c-allocation obtained through (6.14). Given (c_a, x_a, y_a), the solution to (6.16) can be obtained as follows:

$$z_a = \underset{z}{\operatorname{argmin}} \big\{ |\, z - z_r \,| : z \in \mathcal{F}_z(c_a, x_a, y_a) \big\}. \tag{6.17}$$

An illustration of this allocation process is shown in Figure 6.5 for a notional quadrotor with fixed-pitch variable-RPM actuators. The illustration shows three different cases, each assuming the same xy-allocation and different (c_r, z_r) pairs. The black diamond represents $\bar{\mathcal{F}}_{cz}$, the set of feasible cz-allocations. The green region represents $\mathcal{F}_{cz}(x_a, y_a)$ – the

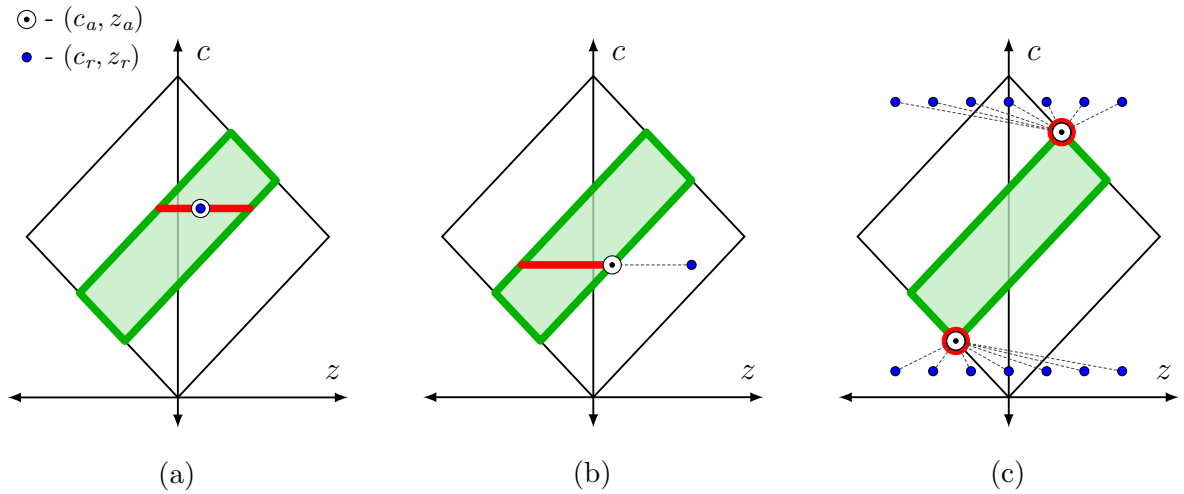


Figure 6.5: Illustration of the cz-allocation with yaw prioritization disabled. Three cases are shown with different (c_r, z_r) requests and the same prescribed (x_a, y_a). The black diamond represents $\bar{\mathcal{F}}_{cz}$, the green region represents $\mathcal{F}_{cz}(x_a, y_a)$, and the red region represents $\mathcal{F}_z(c_a, x_a, y_a)$.

set of feasible cz-allocations that admit the assumed xy-allocation. The red region represents $\mathcal{F}_z(c_a, x_a, y_a)$, the set of feasible z-allocations that admit the assumed cxy-allocation.

Clearly, $\mathcal{F}_z(c_a, x_a, y_a) \subseteq \mathcal{F}_{cz}(x_a, y_a) \subseteq \bar{\mathcal{F}}_{cz}$. Moreover, when (x_a, y_a) is selected from the interior of $\bar{\mathcal{F}}_{xy}$, then the interior of $\mathcal{F}_{cz}(x_a, y_a)$ is non-empty, as seen in Figure 6.5. However, if $(x_a, y_a) \in \partial\bar{\mathcal{F}}_{xy}$, then a motor is saturated, $\mathcal{F}_{cz}(x_a, y_a)$ has an empty interior, and the c- and z-allocations become coupled.

Figure 6.5a shows a case where $(c_r, z_r) \in \mathcal{F}_{cz}(x_a, y_a)$ and the cz-allocation error is zero. Figure 6.5b shows a case where $(c_r, z_r) \notin \mathcal{F}_{cz}(x_a, y_a)$, and the yaw torque is sacrificed to meet the collective request. Figure 6.5b shows a case where $(c_r, z_r) \notin \bar{\mathcal{F}}_{cz}$, the collective allocation error is minimized, and the yaw torque is implicitly determined by (c_a, x_a, y_a).

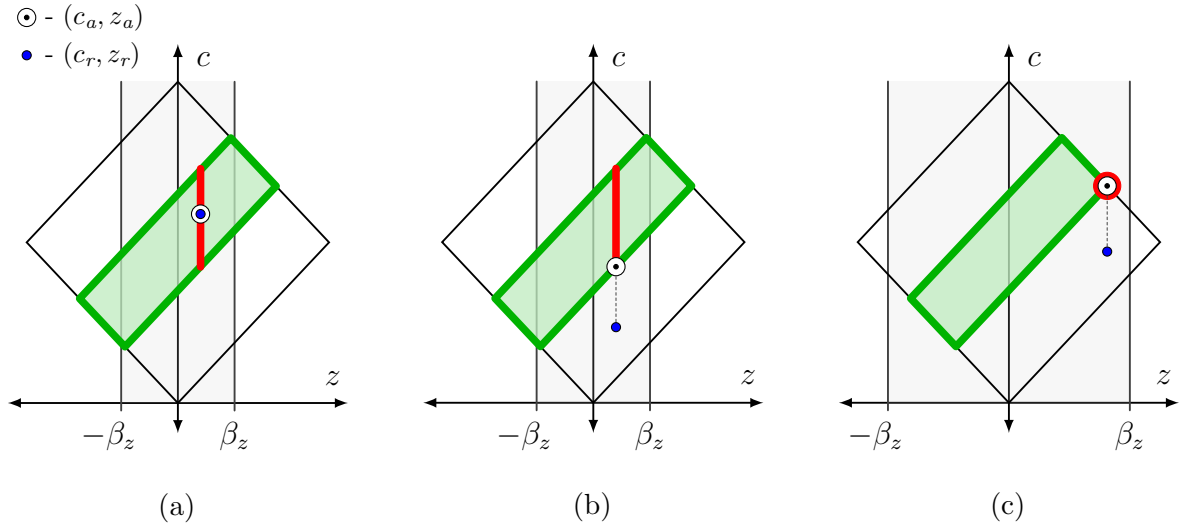


Figure 6.6: Illustration of the cz-allocation inside of enabled yaw prioritization band. Three example (c_r, z_r) are shown for the same prescribed (x_a, y_a). The black diamond and green region are defined as in Figure 6.5. The red region represents $\mathcal{F}_c(x_a, y_a, z_a)$, and the gray region represents the yaw prioritization band with $z_{rsv} \in (0, z_{max})$.

6.4.7 Inside Enabled Yaw Prioritization Band (Step 4)

When yaw prioritization is enabled and the requested yaw torque is within the prioritization band given by

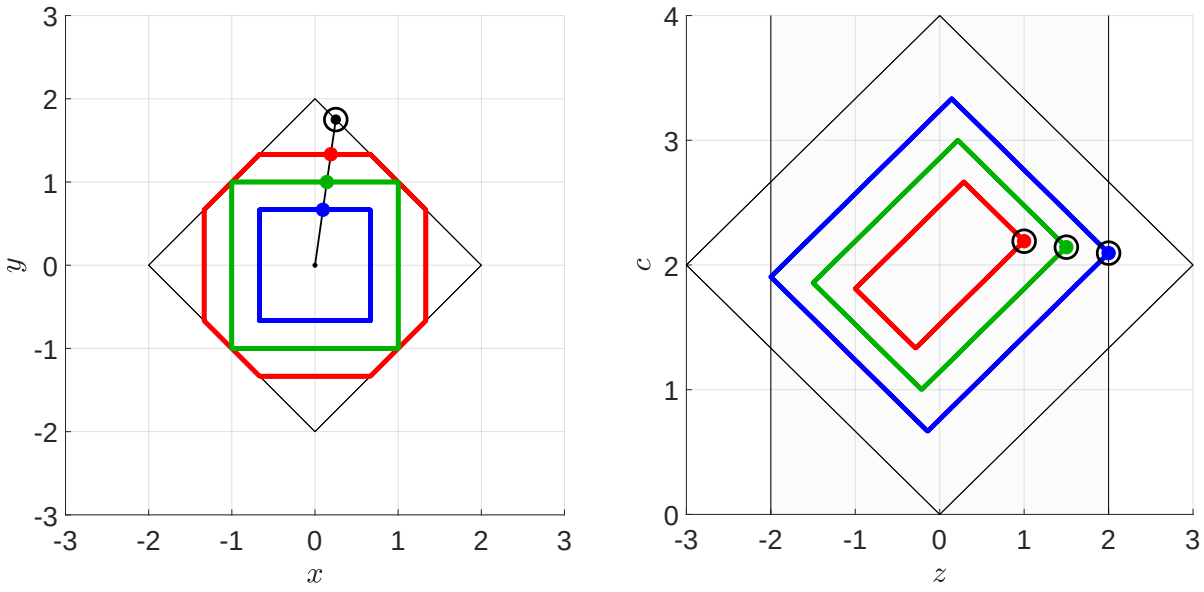
$$|z_r| \leq \beta_z, \tag{6.18}$$

the algorithm allocates the requested yaw torque using

$$z_a = z_r, \tag{6.19}$$

satisfying the implication (6.8a). From the definition of z_{max}, it follows that the allocation made in (6.19) admits a feasible vector of motor forces when (6.18) is satisfied.

With the z-allocation completed, the algorithm proceeds to the c-allocation, which is posed as an LP that operates on a requested collective c_r and a prescribed xyz-torque allocation $\mathrm{w}_{a,xyz} \coloneqq [\, x_a \;\; y_a \;\; z_a \,]^\mathsf{T}$. We represent this LP as $c_a = P_{c,xyz}(c_r, \mathrm{w}_{a,xyz})$, where the



(a) XY-Allocation (b) CZ-Allocation

Figure 6.7: Interaction between the xy- and cz-allocation with yaw prioritization enabled. The black diamonds represent the boundaries of $\bar{\mathcal{F}}_{xy}$ and $\bar{\mathcal{F}}_{cz}$.The yaw prioritization band is shown as the shaded gray region in (b). The red, green, and blue polytopes represent $\mathcal{F}_{xy}(\bar{z})$ and $\mathcal{F}_{cz}(x_a, y_a)$ for $(x_r, y_r) = (\frac{1}{4}, \frac{7}{4})$, $z_r = \{1, \frac{3}{2}, 2\}$, and $z_{rsv} = 2$. The colored solid dots represent requested quantities, whereas the black circles represent allocated quantities. Evidently, $(c_a, z_a) \in \partial\mathcal{F}_{cz}(x_a, y_a)$ for all three cases.

function $P_{c,xyz} : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}$ is given by

$$P_{c,xyz}(c_r, \mathrm{w}_{a,xyz}) := \begin{cases} \underset{c_a}{\text{minimize}} \;\; |c_a - c_r|, \\ \text{subject to:} \;\; c_a = h_c^{\mathsf{T}} f, \\ f \in \mathcal{F}_m, \; \mathrm{w}_{a,xyz} = H_{xyz} f. \end{cases} \tag{6.20}$$

Given (x_a, y_a, z_a), the solution to (6.20) can be obtained as follows:

$$c_a = \underset{c}{\operatorname{argmin}} \big\{ |\, c - c_r \,| : c \in \mathcal{F}_c(x_a, y_a, z_a) \big\}. \tag{6.21}$$

An illustration of this allocation process is shown in Figure 6.6 for the vehicle used in

Figure 6.5. The illustration shows three different cases, each assuming the same xy-allocation and different (c_r, z_r) pairs. The first two cases assume a different value of $z_{rsv} \in (0, z_{max})$ than does the third. The black diamond and green region are defined as in Figure 6.5. The red region represents $\mathcal{F}_c(x_a, y_a, z_a)$, the set of feasible c-allocations that admit the assumed xyz-allocation.

Figure 6.6a shows a case where $(c_r, z_r) \in \mathcal{F}_{cz}(x_a, y_a)$ and the cz-allocation error is zero. Figure 6.6b shows a similar case, where the c-allocation is sacrificed in order to attain the requested (and guaranteed) yaw torque. Figure 6.6c shows a case where

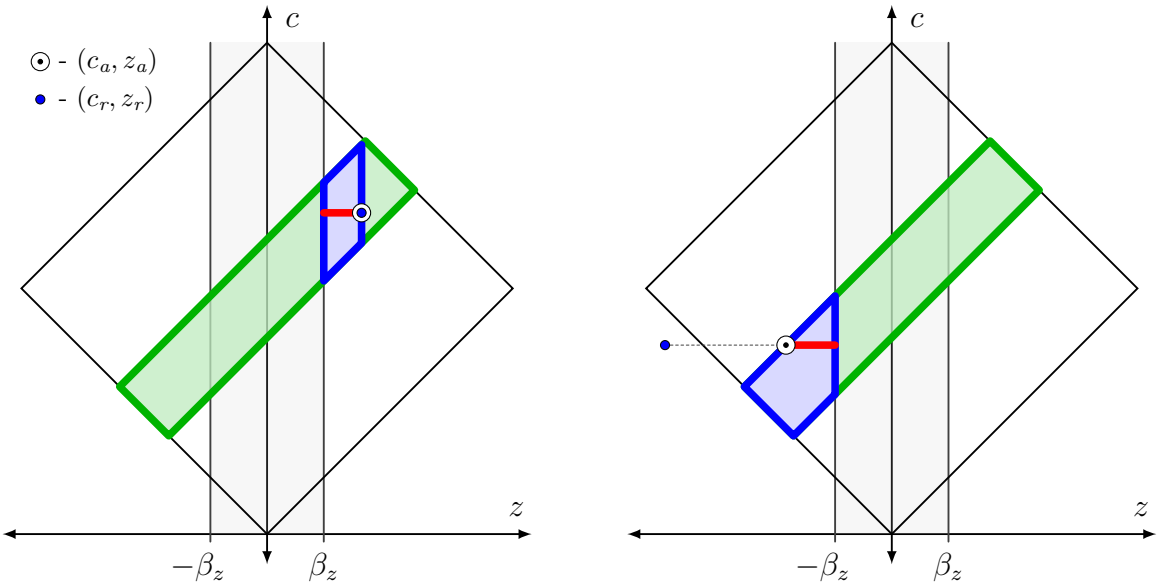
$$z_r = \max\big\{z : (c, z) \in \mathcal{F}_{cz}(x_a, y_a)\big\} \leq \beta_z.$$

This case suggests that increasing z_r while remaining within the band may lead to infeasibility. However, increasing z_r would increase $\bar{z}$, thus shrinking the set $\mathcal{F}_{xy}(\bar{z})$ computed in Step 1 and altering the xy-allocation accordingly. As a result, the set $\mathcal{F}_{cz}(x_a, y_a)$ would expand exactly enough so that $(c_a, z_a) \in \partial\mathcal{F}_{cz}(x_a, y_a)$, ensuring that $z_a = z_r$ remains a feasible allocation for all $|z_r| \leq \beta_z$. This interaction between the xy- and cz-allocation is shown in Figure 6.7 for the quad-rotor mixing matrix given in (6.11), with $(x_r, y_r) = (\frac{1}{4}, \frac{7}{4})$ and $z_{rsv} = 2$. In this case, the c-allocation is completely dependent on the xyz-allocation due to the xy-request and the yaw prioritization. Therefore, the value of c_r is chosen to match the value of c_a to simplify the illustration.

6.4.8 Outside Enabled Yaw Prioritization Band (Step 5)

When yaw prioritization is enabled and the requested yaw torque is outside of the prioritization band $|z_r| > \beta_z$, the algorithm reverts back to the prioritization outlined in Property 1 of §6.4.2. As in Step 3, this prioritization involves solving a c-allocation problem followed by a z-allocation problem. However, in order to satisfy the implication in (6.8b) and address the saturation-like behavior discussed in Property 4 of §6.4.2, these two allocation problems differ from their counterparts (6.14) and (6.16) used in Step 3.

The c-allocation is posed as an LP that operates on a requested collective c_r, a prescribed



(a) (b)

Figure 6.8: Illustration of the cz-allocation outside of enabled yaw prioritization band. The sets $\hat{\mathcal{F}}_{cz}(x_a, y_a, z_r) \cap \check{\mathcal{F}}_{cz}(x_a, y_a, z_r)$ and $\hat{\mathcal{F}}_{z}(c_a, x_a, y_a, z_r) \cap \check{\mathcal{F}}_{z}(c_a, x_a, y_a, z_r)$ are shown in blue and red, respectively, for two different (c_r, z_r) requests and the same prescribed (x_a, y_a).

xy-torque allocation $\mathrm{w}_{a,xy}$, a requested yaw torque z_r, and the quantity $\bar{z}$. We represent this LP as $c_a = \bar{P}_{c,xy}(c_r, \mathrm{w}_{a,xy}, z_r, \bar{z})$, where the function $\bar{P}_{c,xy} : \mathbb{R} \times \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is given by

$$\bar{P}_{c,xy}(c_r, \mathrm{w}_{a,xy}, z_r, \bar{z}) \coloneqq \begin{cases} \underset{c_a}{\text{minimize}} \;\; |c_a - c_r|, \\ \text{subject to:} \;\; c_a = h_c^{\mathsf{T}} f, \\ f \in \mathcal{F}_m, \; \mathrm{w}_{a,xy} = H_{xy} f, \\ |\bar{z}| \leq \operatorname{sign}(z_r)\, h_z^{\mathsf{T}} f \leq |z_r|. \end{cases} \tag{6.22}$$

The key difference between (6.14) and (6.22) is the addition of two inequality constraints in the latter. To explain the purpose of these two constraints, we introduce the sets

$$\hat{\mathcal{F}}_{cz}(x, y, \tilde{z}) \coloneqq \mathcal{F}_{cz}(x, y) \cap \big\{(c, z) : \operatorname{sign}(\tilde{z})\, z \geq \beta_z\big\},$$
$$\check{\mathcal{F}}_{cz}(x, y, \tilde{z}) \coloneqq \mathcal{F}_{cz}(x, y) \cap \big\{(c, z) : \operatorname{sign}(\tilde{z})\, z \leq |\tilde{z}|\big\},$$

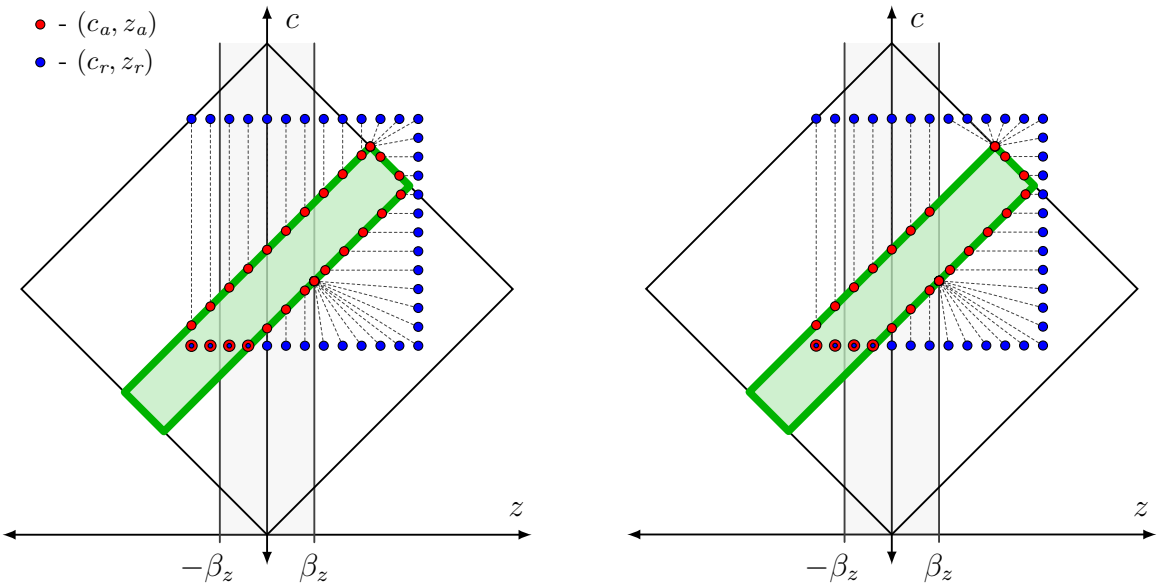


Figure 6.9: Behavior of the cz-allocation with yaw prioritization enabled. The two cases show the effect of including (a) or excluding (b) the inequality sign $(z_r)\, h_z^\intercal f \leq |z_r|$ from (6.22) and (6.25).

$$\hat{\mathcal{F}}_z(c, x, y, \tilde{z}) \coloneqq \big\{z : (c, z) \in \hat{\mathcal{F}}_{cz}(x, y, \tilde{z})\big\},$$
$$\check{\mathcal{F}}_z(c, x, y, \tilde{z}) \coloneqq \big\{z : (c, z) \in \check{\mathcal{F}}_{cz}(x, y, \tilde{z})\big\}.$$

The inequality $|\bar{z}| \leq \text{sign}\,(z_r)\, h_z^\intercal f$ ensures that the c-allocation satisfies

$$\hat{\mathcal{F}}_z(c_a, x_a, y_a, z_r) \neq \varnothing,$$

and thus, that the subsequent z-allocation admits a solution satisfying implication (6.8b). The inequality $\text{sign}\,(z_r)\, h_z^\intercal f \leq |z_r|$ ensures that the subsequent z-allocation admits a solution satisfying the saturation-like behavior discussed in Property 5 in §6.4.2. Together, the two inequalities in (6.22) ensure that there exists a z such that the c-allocation satisfies

$$(c_a, z) \in \hat{\mathcal{F}}_{cz}(x_a, y_a, z_r) \cap \check{\mathcal{F}}_{cz}(x_a, y_a, z_r).$$

Given (x_a, y_a), the solution to (6.22) can be obtained as follows:

$$c_a = \underset{c}{\operatorname{argmin}} \big\{ |\, c - c_r \,| : (c, z) \in \hat{\mathcal{F}}_{cz}(x_a, y_a, z_r) \cap \check{\mathcal{F}}_{cz}(x_a, y_a, z_r) \big\}. \tag{6.24}$$

The z-allocation is posed as an LP that operates on a requested yaw torque z_r, a prescribed cxy-wrench allocation $\mathrm{w}_{a,cxy}$, and the quantity $\bar{z}$. We represent this LP as $z_a = \bar{P}_{z,cxy}(z_r, \mathrm{w}_{a,cxy}, \bar{z})$, where the function $\bar{P}_{z,cxy} : \mathbb{R} \times \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ is given by

$$\bar{P}_{z,cxy}(z_r, \mathrm{w}_{a,cxy}, \bar{z}) \coloneqq \begin{cases} \underset{z_a}{\text{minimize}} \;\; |z_a - z_r|, \\ \text{subject to:} \;\; z_a = h_z^{\mathsf{T}} f, \\ f \in \mathcal{F}_m, \; \mathrm{w}_{a,cxy} = H_{cxy} f, \\ |\bar{z}| \leq \operatorname{sign}(z_r)\, h_z^{\mathsf{T}} f \leq |z_r|. \end{cases} \tag{6.25}$$

Given (c_a, x_a, y_a), the solution to (6.25) can be obtained as follows:

$$z_a = \underset{z}{\operatorname{argmin}} \big\{ |\, z - z_r \,| : z \in \hat{\mathcal{F}}_z(c_a, x_a, y_a, z_r) \cap \check{\mathcal{F}}_z(c_a, x_a, y_a, z_r) \big\}. \tag{6.26}$$

Figure 6.8 shows an illustration of the cz-allocation for the case when $|z_r| > \beta_z$ and yaw prioritization is enabled. Notably, the figure shows the geometry of the sets $\hat{\mathcal{F}}_{cz}(\cdot)$, $\check{\mathcal{F}}_{cz}(\cdot)$, $\hat{\mathcal{F}}_z(\cdot)$, and $\check{\mathcal{F}}_z(\cdot)$ in relation to $\mathcal{F}_{cz}(x_a, y_a)$ and $\bar{\mathcal{F}}_{xy}$.

Figure 6.9 shows the behavior of the cz-allocation for a sweep of (c_r, z_r) pairs. Figure 6.9a shows the result of applying (6.22)-(6.26) as described above. By inspection, we see that both the c- and z-allocations demonstrate saturation-like responses similar to the one shown in Figures 6.2a and (6.2b). Also notable is the set of (c_r, z_r) at the lower right that map to the same (c_a, z_a) point at the intersection of the yaw prioritization band and $\mathcal{F}_{cz}(x_a, y_a)$. This feature is a consequence of the collective allocation being sacrificed in order to satisfy implication (6.8b) of Property 2 in §6.4.2.

Figure 6.9b shows the outcome for the same pairs of (c_r, z_r) if the inequality

$$\operatorname{sign}(z_r)\, h_z^{\mathsf{T}} f \leq |z_r|,$$

is removed from (6.22) and (6.25). As evident at the top right of the figure, as z_r transitions from left to right, the z-allocation jumps from tracking to top surface of the green polytope

to the top right corner of the polytope. The allocated value persists as z_r passes the top right corner of the green polytope until c_r drops below said corner. Although the response of c_r is saturation-like, that of z_r is not and instead resembles the response shown in Figure 6.2c. Hence, we conclude that the inclusion of $\operatorname{sign}(z_r)\, h_z^\intercal f \leq |z_r|$ in the cz-allocation is key to satisfying Property 4 of §6.4.2, and therefore, to ensuring that the allocator does not adversely affect the low-level controller.

6.4.9 Degraded Operation (Step 6)

This step addresses Property 5 outlined in §6.4.2. In Assumption 3 of §6.4.1, we assumed that $\operatorname{rank}(H_{\mathrm{w}}) = 3$ results from a linear dependence between the rows of H_{xy} and $h_z^\intercal$. In this circumstance, the vehicle loses a control DoF. In light of the prioritization outlined in Property 1 of §6.4.2, the remaining three control DoFs are assigned to the collective and xy-torques, and yaw-prioritization is disabled.

In this mode of operation, the algorithm obtains its xy-torque and collective allocations through (6.13) and (6.14). Since the linear dependence is between the rows of H_{xy} and $h_z^\intercal$, it follows that $\operatorname{rank}(H_{cxy}) = 3$. Consequently, the z-allocation can be obtained using the pseudo-inverse of H_{cxy}:

$$z_a = h_z^\intercal H_{cxy}^\intercal \left(H_{cxy}\, H_{cxy}^\intercal\right)^{-1} \mathrm{w}_{a,cxy}. \tag{6.27}$$

6.4.10 Compute Allocated Motor Forces (Step 7)

If $\operatorname{rank}(H_{\mathrm{w}}) = 4$, the vector of allocated motor forces is related to the allocated wrench w_a through the pseudo-inverse of H_{w}:

$$f_a = H_{\mathrm{w}}^\intercal \left(H_{\mathrm{w}}\, H_{\mathrm{w}}^\intercal\right)^{-1} \mathrm{w}_a. \tag{6.28}$$

Note that if $n_m = 4$, the inverse of H_{w} can be used in lieu of (6.28). If $\operatorname{rank}(H_{\mathrm{w}}) = 3$, then $\operatorname{rank}(H_{cxy}) = 3$, as discussed in §6.4.9. Hence, f_a is obtained in terms of the allocated wrench $\mathrm{w}_{a,cxy}$ through the pseudo-inverse of H_{cxy}:

$$f_a = H_{cxy}^\intercal \left(H_{cxy}\, H_{cxy}^\intercal\right)^{-1} \mathrm{w}_{a,cxy}. \tag{6.29}$$

The minimum-norm property of the pseudo-inverses applied in (6.28) and (6.29) ensures that f_a is orthogonal to the null space of H_{w}, and that the motor forces are feasible even when H_{w} is not bijective.

6.5 Low-Level Controller

In this section, we detail the low-level controller block shown in Figure 6.1. The low-level controller is the part of the proposed architecture most directly responsible for making the system robust to variations in actuator dynamics, variations in plant dynamics, and external force and torque disturbances. The design of the low-level controller is based on the nonlinear dynamic compensator detailed in Chapter 10 of [59], a loop-shaping based classical control architecture that systematically addresses three critical aspects of practical feedback control: (1) feedback maximization for disturbance rejection and sensitivity reduction, (2) feed-forward control for improved transient response, and (3) nonlinear dynamic compensa-

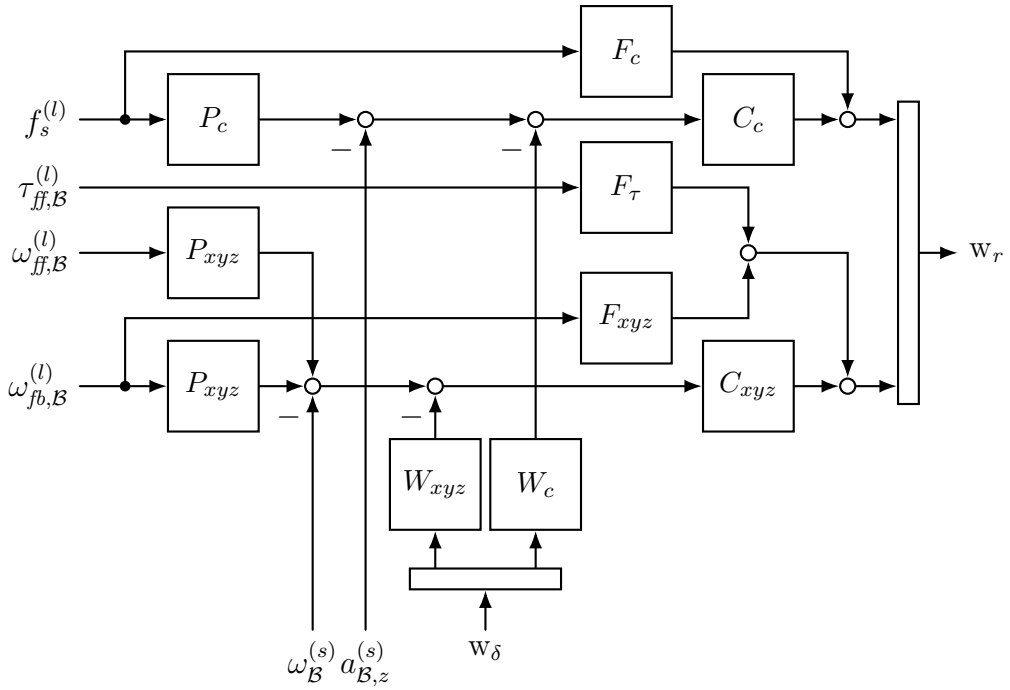


Figure 6.10: Layout of the low-level controller.

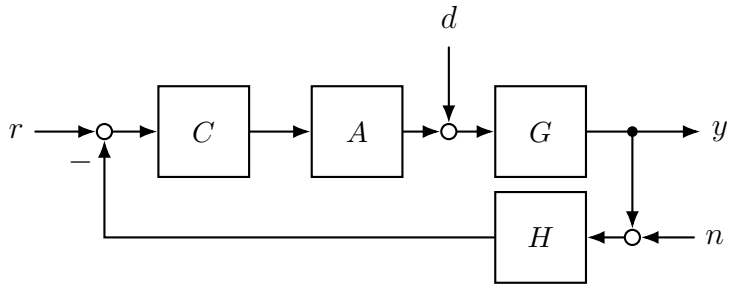


Figure 6.11: An idealized single-input single-output closed-loop system.

tion for stability guarantees and transient performance in the face of actuator saturation.

The layout of the low-level controller is shown in Figure 6.10. The low-level controller regulates the vehicle's vertical acceleration and three angular velocity components, and is commanded by the mid-level controller through the collective term $f_s^{(l)} \in \mathbb{R}$, the feed-forward torque and angular velocity $\tau_{ff,\mathcal{B}}^{(l)}, \omega_{ff,\mathcal{B}}^{(l)} \in \mathbb{R}^3$, and the feedback angular velocity $\omega_{fb,\mathcal{B}}^{(l)} \in \mathbb{R}^3$. The low-level controller uses the measured z-axis acceleration $a_{\mathcal{B},z}^{(s)} \in \mathbb{R}$ and angular velocity $\omega_{\mathcal{B}}^{(s)} \in \mathbb{R}^3$ as feedback. The output of the low-level controller is the requested wrench vector $\mathrm{w}_r \in \mathbb{R}^4$, which specifies the desired values of the control DoF to the allocator. The allocator computes the allocated wrench vector $\mathrm{w}_a \in \mathbb{R}^4$ that is physically realizable according to its prioritization logic. The difference between the requested and allocated wrenches

$$\mathrm{w}_\delta = \mathrm{w}_r - \mathrm{w}_a,$$

is returned to the controller (see Figure 6.1).

In what follows, we outline a three step NDC design procedure. These three steps constitute a three degree-of-freedom design, where feedback, feed-forward, and anti-windup elements are designed independently in succession. These three steps are discussed in §6.5.1-§6.5.3 in the context of an abstract single-input single-output system.

6.5.1 Feedback Design

Consider an idealized single-input single-output closed-loop system like the one shown in Figure 6.11. This system is comprised of four linear time-invariant (LTI) dynamic elements:

a *controller*, an *actuator*, a *plant*, and a *sensor*. We represent these blocks using the transfer functions $C(s)$, $A(s)$, $G(s)$, and $H(s)$, respectively. Disturbances enter the system additively between the actuator output and plant input, and noise enters the system additively between the plant output and sensor input. We define the *loop transfer function* as

$$L(s) \coloneqq G(s)A(s)C(s)H(s), \tag{6.30}$$

the *sensitivity function* as

$$S(s) \coloneqq \frac{1}{1 + L(s)}, \tag{6.31}$$

the *disturbance-to-output transfer function* as

$$G_d(s) \coloneqq S(s)G(s), \tag{6.32}$$

and the *reference-to-output transfer function* as

$$G_{r,1}(s) \coloneqq S(s)G(s)A(s)C(s). \tag{6.33}$$

The sensitivity function owes its name to its derivation as the ratio between the relative change in $G_{r,1}(s)$ induced by a relative change in $G(s)A(s)$ [29]. Consequently, frequencies $\omega \geq 0$ for which $|S(j\omega)|$ is small are frequencies at which the closed-loop system response is effectively unaffected by variations in $G(s)A(s)$, which we collectively refer to as *plant uncertainties*. Moreover, from (6.32), it follows that the effects of disturbances on the output of the system are small when $|S(s)|$ is significantly smaller than $|G(s)|$. Therefore, robustness to plant uncertainties and good disturbance rejection require that $|S(j\omega)|$ be small over the desired frequency range.

Adopting the terminology introduced in Chapter 1 of [59], we say that the system exhibits *negative feedback* at frequencies where $|S(j\omega)| < 1$, and *positive feedback* at frequencies where $|S(j\omega)| > 1$. In these terms, the primary objective of feedback control is to maximize negative feedback – the quantity $|1 + L(j\omega)|$ – over the range of frequencies where low sensitivity and good disturbance rejection are desired. We will refer to this range of frequencies as

the *functional bandwidth*, and note that it typically inhabits the portion of the frequency spectrum below the gain-crossover frequency of $L(j\omega)$.

From the Bode sensitivity integral relation, we know that the integral of the log-sensitivity of an LTI closed-loop system is conserved. For a system whose loop transfer function contains n_{RHP} right-half-plane poles located at $p_i \in \mathbb{C}$ for $i = \{1, \ldots, n_{RHP}\}$, and no right half-plane zeros, this conservation law is given by [81, 82]:

$$\int_0^\infty \ln |S(j\omega)| d\omega = \pi \sum_{i=1}^{n} \mathrm{Re}(p_i) - \frac{\pi}{2} \lim_{s\to\infty} sL(s). \tag{6.34}$$

The most important implication of (6.34) is that a reduction in the sensitivity over one interval of frequencies necessitates an increase in the sensitivity elsewhere on the frequency spectrum. This behavior is often referred to as the *waterbed effect*.

From (6.34) it follows that maximizing negative feedback over the functional bandwidth is equivalent to maximizing positive feedback elsewhere on the spectrum (see Corollary 3.1 in [59]). From (6.31), it is evident that positive feedback occurs only over frequencies where $L(j\omega)$ passes inside the unit circle centered at the Nyquist critical point $-1 + j0$ – that is, for frequencies where $|L(j\omega) + 1| < 1$. Moreover, positive feedback increases the closer $L(j\omega)$ passes to the critical point. Therefore, the original design objective of feedback maximization can be restated in terms of designing $C(s)$ such that $L(j\omega)$ wraps around the critical point as tightly as possible. This process is often referred to as *loop-shaping*.

The loop-shaping procedure must satisfy the Nyquist Stability Criterion with sufficient gain and phase margin in order to ensure that the stability of the closed-loop system is robust to plant uncertainties.[4] For a given gain-crossover frequency, gain and phase margin are the primary limitations on the amount of positive (and therefore negative) feedback that can be attained. Reassuringly, the process of feedback maximization exhibits the quintessential quality of an optimization problem, wherein an optimal quantity – the largest integral of

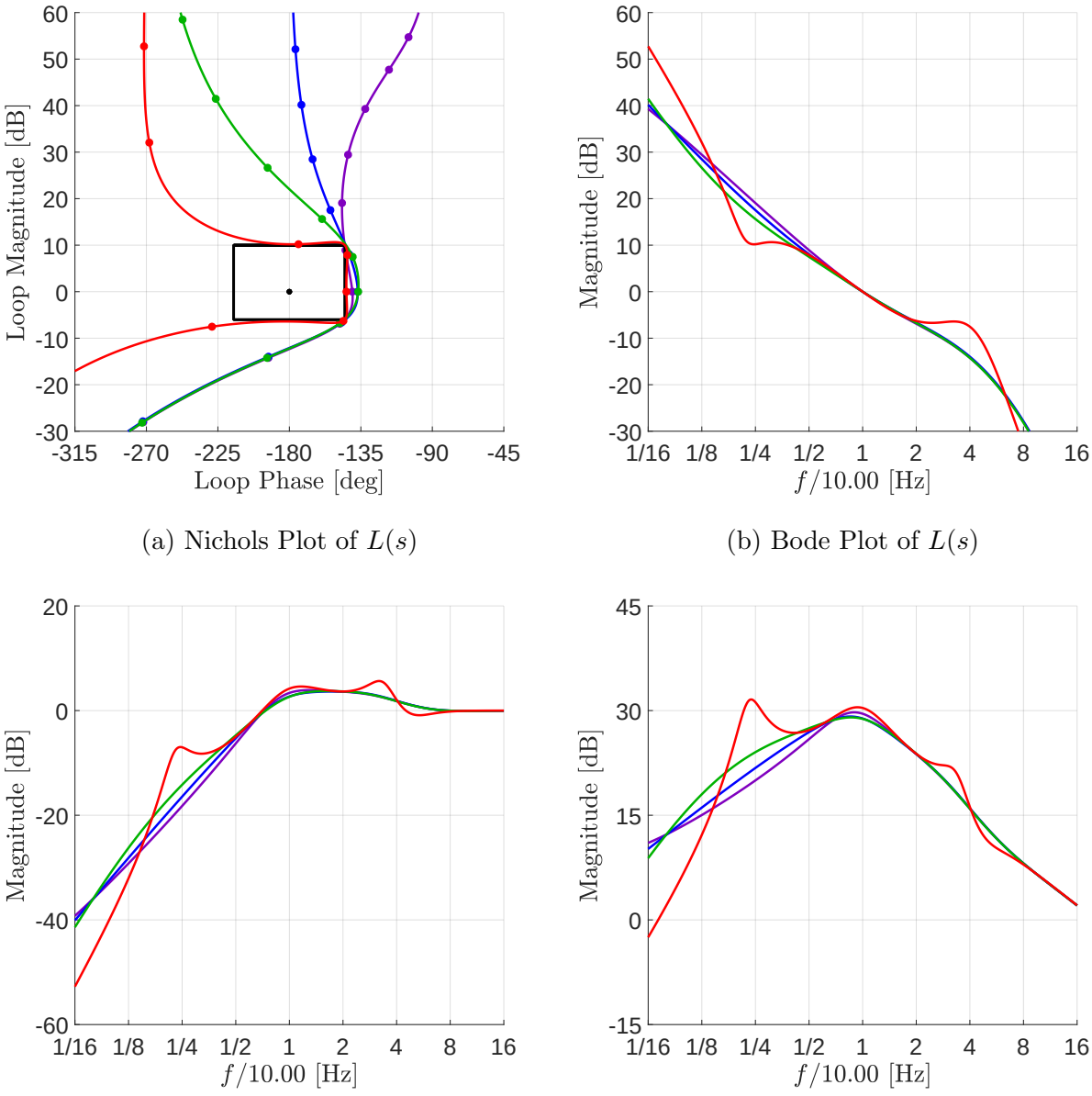
[4]Note that making the closed-loop stability robust to plant uncertainties is distinctly different from lowering the sensitivity of the system. The former ensures that variations in $G(s)A(s)$ do not destabilize the closed-loop system, whereas the latter ensures that variations in $G(s)A(s)$ do not alter the behavior of the closed-loop system significantly.

negative feedback over the functional bandwidth – is attained at the boundary of the active constraints – the gain and phase margin boundaries.

This interplay presents an intuitive trade-off between stability robustness to plant uncertainty, and disturbance rejection and sensitivity reduction. For example, if large uncertainties exist in $G(s)A(s)$, large gain and phase margins must be used to ensure robust closed-loop stability. To satisfy the larger stability margins, $L(j\omega)$ must be shaped to be farther away from the Nyquist critical point, thus reducing the available positive (and negative) feedback. Consequently, the closed-loop system exhibits reduced disturbance rejection capabilities and increased sensitivity. On the other hand, if $G(s)$ and $A(s)$ are known precisely, smaller gain and phase margins may be used, and thus more negative feedback can be realized for the sake of improved disturbance rejection and sensitivity reduction.

The process of feedback maximization presents a second, but less apparent, trade-off. In order to tightly follow the rectangular gain and phase margin boundaries, the loop must be shaped using sharp corners, which are realized by introducing lightly damped complex poles and zeros into $C(s)$. These sharp corners typically translate to lightly damped oscillatory time domain responses at the frequencies where the corners were placed.

To illustrate the concepts discussed above, consider the four example loop shapes shown in Figure 6.12. All four loop shapes are designed with a gain-crossover frequency of 10 Hz. Each has the same $A(s)$, $G(s)$, and $H(s)$, and owes its unique shape to $C(s)$. Figure 6.12a shows the Nichols plot for all four cases, with frequency increasing from top to bottom. The DC phase shift indicates that the systems have different steady-state characteristics, with the purple loop shape representing a type-1 system, the blue loop shape representing a type-2 system, and the red and green loop shapes representing type-3 systems. Notably, the red loop shape follows the gain and phase margin box virtually as tightly as possible, and thus maximizes feedback. The corresponding Bode plots of $L(s)$ are shown in Figure 6.12b, where the red loop shape exhibits the upper and lower Bode steps described in Chapter 4 of [59].



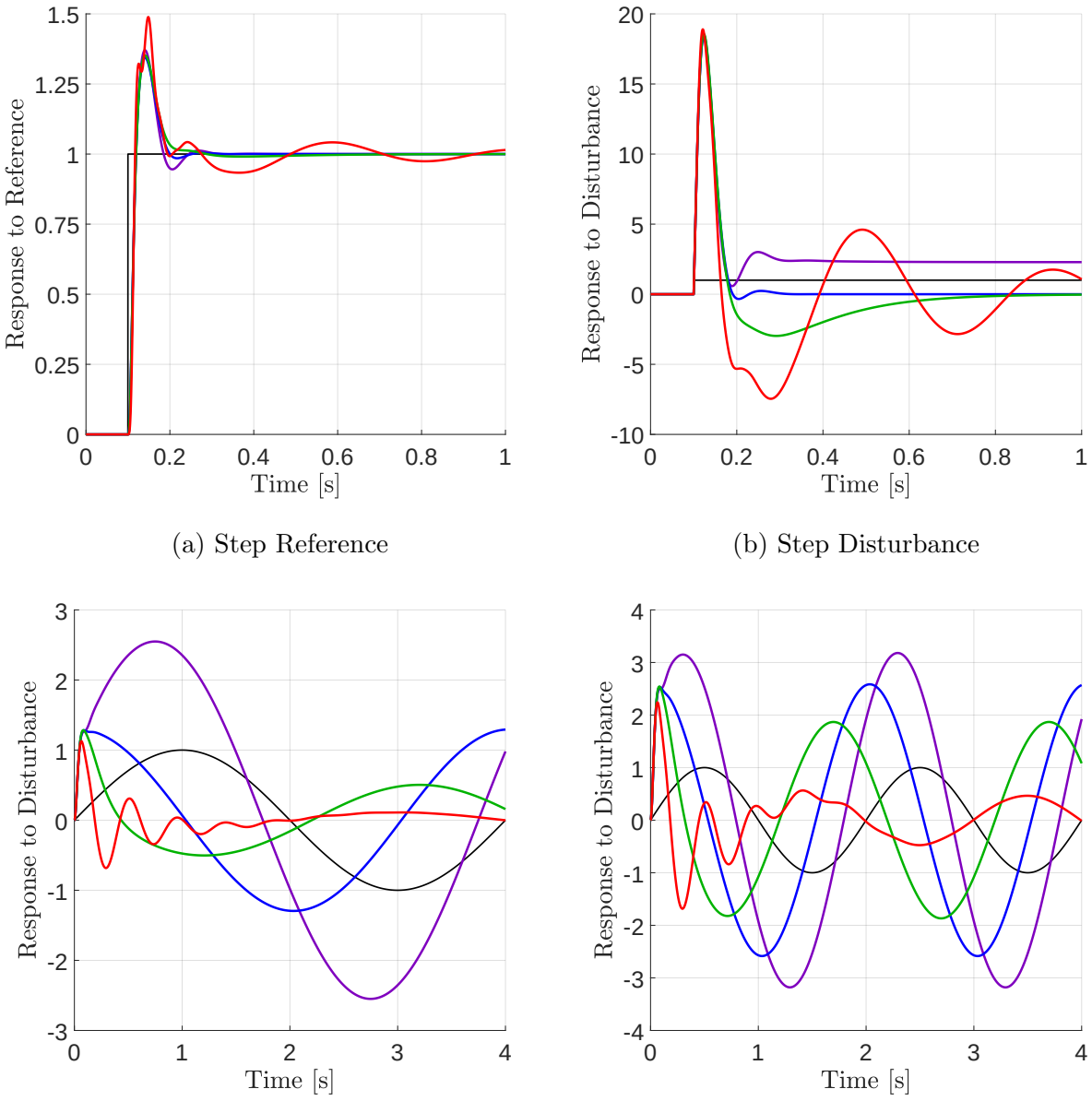
(a) Nichols Plot of $L(s)$

(b) Bode Plot of $L(s)$

(c) Bode Plot of $S(s)$

(d) Bode Plot of $G_d(s)$

Figure 6.12: Nichols and Bode plots for four example loop shapes. The dots in (a) represent the octave increments indicated on the Bode plots in (b)-(d).



(a) Step Reference

(b) Step Disturbance

(c) 0.25 Hz Disturbance

(d) 0.50 Hz Disturbance

Figure 6.13: Time domain responses of the four loop shapes shown in Figure 6.12. (a) and (b) show responses to a step reference and disturbance, respectively. (c) and (d) show responses to low frequency sinusoidal disturbances. The black lines represent the reference and disturbance signals.

Figures 6.12c and 6.12d show Bode plots of the corresponding sensitivity and disturbance-to-output transfer functions. Since the red loop shape maximizes feedback, it is no surprise that it achieves the lowest sensitivity and best disturbance rejection over the functional bandwidth located at lower frequencies. However, due to the waterbed effect and the sharp corners introduced into $L(s)$, the red system exhibits higher sensitivity and poorer disturbance rejection in the vicinity of the crossover frequency. The extent to which this matters depends on the anticipated plant uncertainties and disturbances.

Figure 6.13 shows various time domain responses of the four loop shapes introduced above. Figure 6.13a shows the response of the systems to a step in the reference signal. All four systems are seen to exhibit considerable overshoot, an issue that will be addressed in §6.5.2. However, while the smoother purple, blue, and green loop shapes settle to their final values rapidly, the sharper red loop shape exhibits a lightly damped oscillation around the commanded reference. The frequency of this oscillation can be attributed to the lower Bode step seen in the red loop shape in Figure 6.12b.

Figures 6.13b-6.13d show the response of the systems to step and sinusoidal disturbances. The step disturbance highlights the different steady-state characteristics of the four systems. Specifically, we see that the type-1 purple system cannot eliminate the steady-state error induced by a step disturbance. Moreover, since the spectral energy of a step is distributed over all frequencies, the red system once more exhibits an undesirable lightly damped oscillation. However, unlike with the response to a reference change, the response to a disturbance typically cannot be altered using feed-forward design. While the red system seems to perform poorly in the results presented thus far, its feedback maximization if evident in its response to the two low frequency sinusoidal disturbances shown in Figures 6.13c and 6.13d. Here, the red system is seen to greatly out perform the other systems in attenuating the disturbance signal.

In summary, the objective of this step is to design the feedback controller $C(s)$ so as to maximize disturbance rejection and minimize sensitivity. The resulting loop shape must balance the objective of feedback maximization with the need to have satisfactory time

domain response. Since the time domain response to a reference can be significantly improved by the methodology outlined in §6.5.2, it is the disturbance response that most affects this trade-off. The loop shape must also account for the desired steady-state characteristics of the closed-loop system, and must distribute the achieved negative feedback across the functional bandwidth appropriately.

6.5.2 Step 2 - Feed-Forward Design

In this step, we introduce feed-forward control in order to improve the reference-to-output response of the feedback controller designed in the previous step. This is a well-known strategy for improving the transient response of the system, and is often described as a two degree-of-freedom design [81]. To do this, we modify the control architecture shown in Figure 6.11 by adding a prefilter $P(s)$ and a command feed-forward element $F(s)$, as shown in Figure 6.14. The reference-to-output transfer function of this new system is given by

$$G_{r,2}(s) \coloneqq S(s)\Big[G(s)A(s)\big[F(s)+C(s)P(s)\big]\Big], \tag{6.35}$$

which recovers $G_{r,1}(s)$ when $F(s)=0$ and $P(s)=1$. We emphasize that the sensitivity and disturbance-to-output transfer functions of the system in Figure 6.14 are identical to those of the system in Figure 6.11. This allows us to proceed with the feed-forward design without compromising the feedback performance attained in the feedback design step.

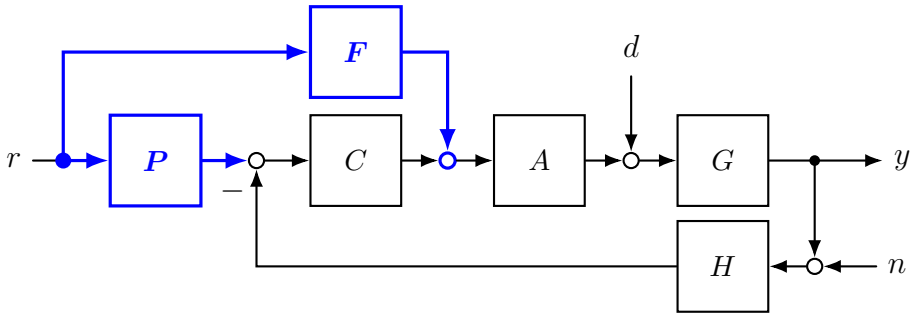
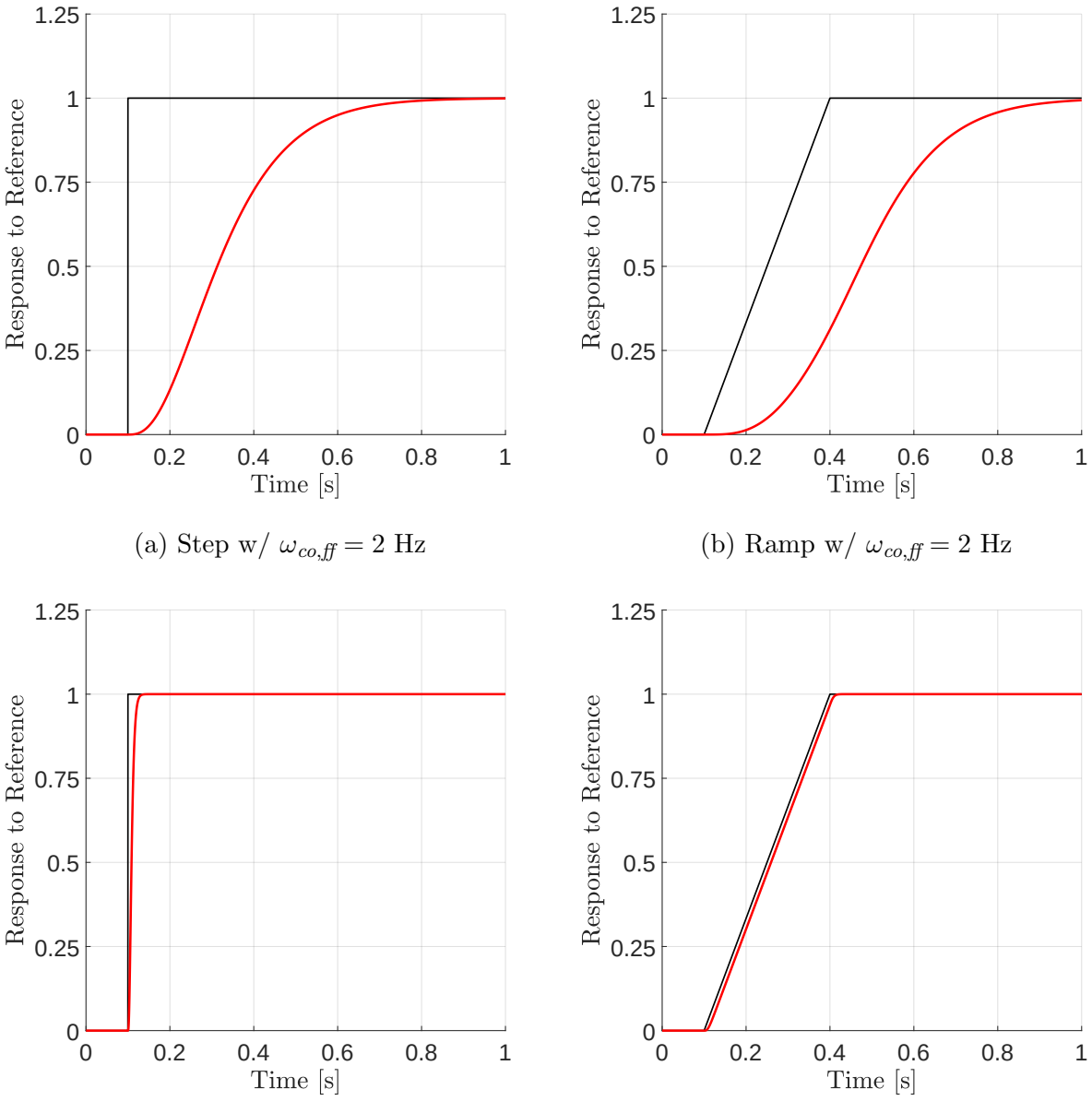


Figure 6.14: The closed-loop system shown in Figure 6.11 with feed-forward elements.



(a) Step w/ $\omega_{co,ff} = 2$ Hz

(b) Ramp w/ $\omega_{co,ff} = 2$ Hz

(c) Step w/ $\omega_{co,ff} = 50$ Hz

(d) Ramp w/ $\omega_{co,ff} = 50$ Hz

Figure 6.15: Effect of feed-forward design on the reference-to-output response of the red system presented in Figures 6.12 and 6.13.

However, since the feed-forward elements are not part of the feedback loop, the sensitivity of (6.35) with respect to $F(s)$ and $P(s)$ is unity. This high sensitivity is a well-known property of feed-forward control, which relies on a nominal system model to anticipate the behavior of

the actual physical system. Since feed-forward control is not robust to plant uncertainties, it should be applied only at frequencies where the system is known precisely.

To proceed, we propose the following feed-forward elements

$$F(s) \coloneqq \big[G(s)A(s)\big]^{-1}Q(s), \tag{6.36a}$$

$$P(s) \coloneqq H(s)Q(s), \tag{6.36b}$$

where $Q(s)$ is a transfer function specified by the designer. By inspection, we see that $Q(s) = 1$ results in $G_{r,2}(s) = 1$, which implies perfect reference-to-output tracking. While this choice of $Q(s)$ gives an ideal response, it is typically not realizable. Specifically, if $G(s)A(s)$ has more poles than zeros – as is the case in virtually all practical systems – then the inversion in (6.36a) results in a non-realizable improper transfer function that violates causality. To mitigate this issue, $Q(s)$ must be selected to have a pole-zero excess greater than that of $G(s)A(s)$. Doing so guarantees that both $F(s)$ and $P(s)$ are realizable proper transfer functions.

To simplify the feed-forward design process, we assume that $Q(s)$ is a low-pass filter with a cutoff frequency of $\omega_{co,ff}$. Since the order of the filter is determined by the required pole-zero excess mentioned above, $\omega_{co,ff}$ becomes the primary design variable of the feed-forward design procedure.

Unsurprisingly, the selection of $\omega_{co,ff}$ results in a trade-off. On one hand, as $\omega_{co,ff}$ is increased, $Q(s) \to 1$ and $G_{r,2}(s) \to 1$, and the nominal reference-to-output tracking is improved. On the other hand, increasing $\omega_{co,ff}$ exposes the uncertain, higher frequency portions of $G(s)$ and $A(s)$ that are amplified by the inversion in (6.36a). This can result in imprecise, if not dangerous, system response.

Figure 6.15 illustrates the concepts discussed above for the red system in Figures 6.12 and 6.13. Figure 6.15a shows the reference-to-output response of the system to a step input with $\omega_{co,ff} = 2$ Hz. Figure 6.15c shows the corresponding response with $\omega_{co,ff} = 50$ Hz. Figures 6.15b and 6.15d show the corresponding responses to a ramp input. The effect of $\omega_{co,ff}$ on the swiftness of the response is evident in the plots. Also notable is the marked improvement over the red system's reference-to-output response seen in Figure 6.13a.

6.5.3 Step 3: Anti-Windup Design

In this section, we address the issue of actuator saturation. The Nyquist Stability Criterion is *necessary and sufficient* for closed-loop stability of the idealized LTI systems shown in Figure 6.11 and 6.14. However, if a saturation nonlinearity is present in the loop – as it is in virtually any practical system – then the Nyquist Stability Criterion is reduced to a *necessary* condition, as discussed in Chapter 10 of [59]. This circumstance is illustrated by the black system seen in Figure 6.16, where a static saturation element limits the range of inputs commanded to the actuator. For high-performance control systems, this saturation heavily influence the transient behavior and stability of the closed-loop system, and thus cannot be neglected.

Reference [59] distinguishes between two kinds of systems: *Nyquist stable* (NS) systems, and *absolutely stable* (AS) systems. To illustrate these concepts, consider the feedback system shown in Figure 6.17a – a simplified version of the black and red system shown in Figure 6.16.

A system is NS if it satisfies the Nyquist Stability Criterion. All four loop shapes shown in Figure 6.12a are NS. A system is AS if it is asymptotically globally stable for any nonlin-

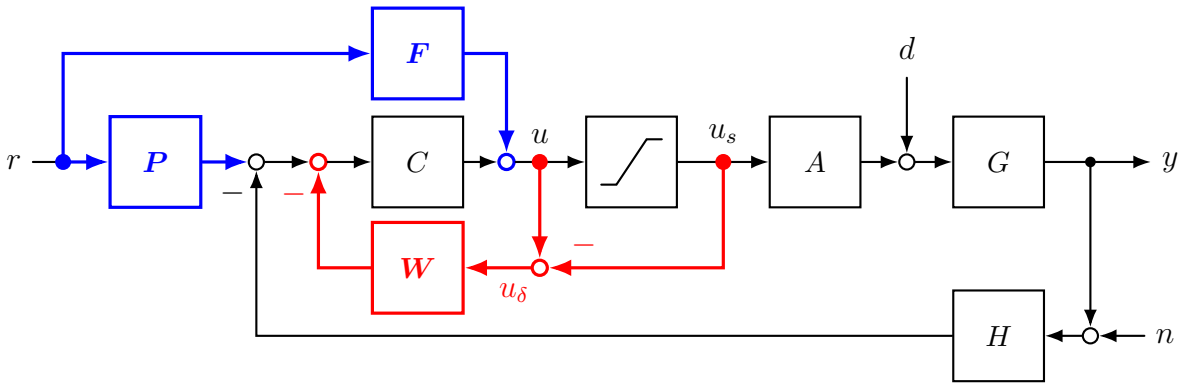


Figure 6.16: A practical single-input single-output closed-loop system. The components shown in black constitute a typical closed-loop control system with a static nonlinear saturation. The components shown in red render the (otherwise standard) controller a nonlinear dynamic compensator [59].

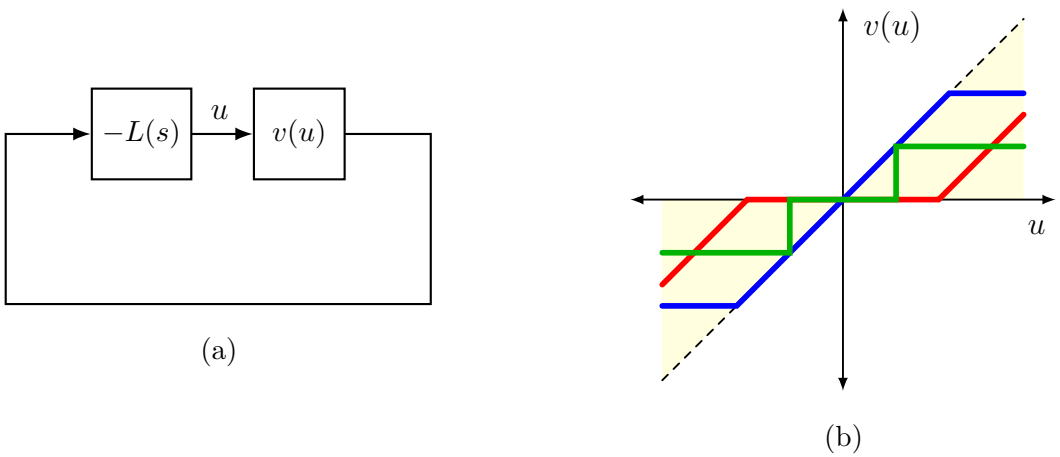


Figure 6.17: Feedback systems like the one shown in (a) are absolutely stable if the nonlinear element $v(u)$ satisfies property (6.37) illustrated in (b). For reference, (b) presents a few common nonlinearities that satisfy this condition.

earity $v(u)$ satisfying

$$0 \leq v(u)/u \leq 1. \tag{6.37}$$

Saturation, dead zone, and three-level relay are common nonlinearities that satisfy (6.37), as seen in Figure 6.17b (see Section 10.4 of [59]). From the necessity of the Nyquist Stability Criterion, it follows that all AS systems are also NS.

In contrast to the Nyquist Stability Criterion, the Popov Stability Criterion is a *sufficient* condition for the absolute stability of the system in Figure 6.17a (see Section 10.5 in [59]). The Popov Stability Criterion can be stated in several ways. Arguably, its most intuitive form is captured by the condition

$$\omega \operatorname{Im} L(j\omega) < \alpha_{psc}\big[\operatorname{Re} L(j\omega) + 1\big], \quad \exists\, \alpha_{psc} > 0. \tag{6.38}$$

where L is the loop transfer function defined in (6.30). Condition (6.38) written with an equality forms what is referred to as the *Popov line*. This condition can be interpreted graphically using the modified Nyquist plane, which is simply the well-known Nyquist plane with its imaginary axis scaled by ω [59] (see Figure 6.18).

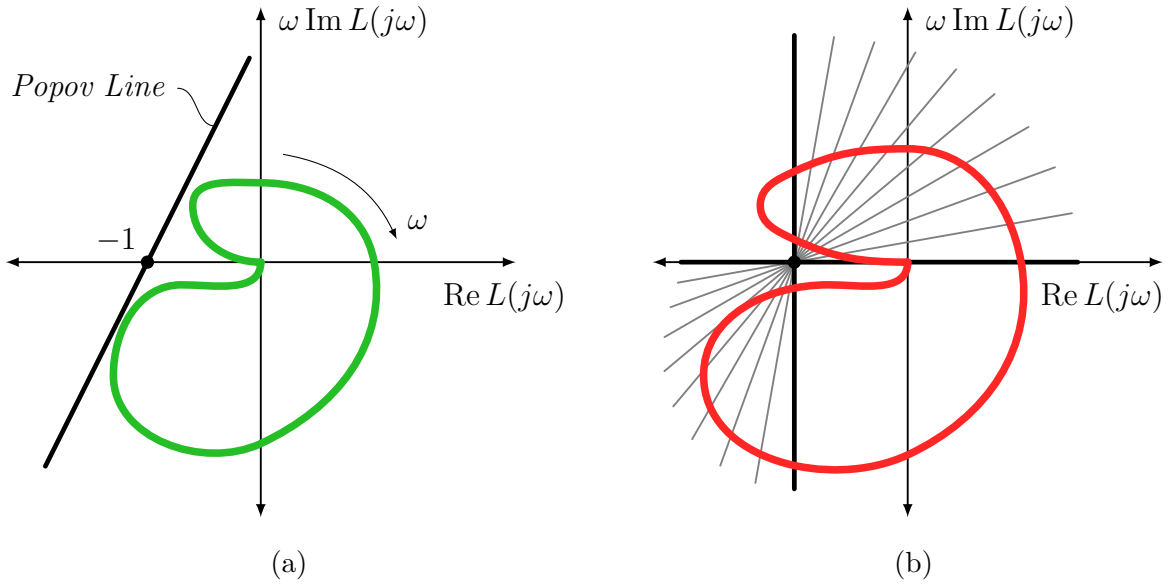


Figure 6.18: The modified Nyquist plane used to graphically interpret the Popov Stability Criterion given in (6.38). The system in (a) satisfies the Popov Stability Criterion, whereas the one in (b) does not.

In this context, a system satisfies the Popov Stability Criterion – and is therefore AS – if there exists a Popov line with positive slope α_{psc} such that the entirety of $L(j\omega)$ can be drawn to its right. Figure 6.18a shows an AS system that satisfies (6.38). If no Popov line can be drawn to satisfy (6.38), then no conclusion can be drawn regarding the overall stability of the system.[5] To simplify the following discussion, we henceforth refer to systems that satisfy both the Nyquist Stability Criterion and the Popov Stability Criterion as AS systems, and systems that satisfy only the former as NS systems.

Systems whose Nichols plot passes over the Nyquist critical point are NS, whereas those whose Nichols plot does not pass over the critical point are AS [59]. For example, the red and green systems shown in Figure 6.12a are NS, whereas the blue and purple systems are AS. This observation implies that an AS system cannot violate the Nyquist Stability Criterion

[5]In such a case, the plot can only rule out certain periodic oscillations of frequency ω if all integer multiples of ω satisfy (6.38) [59].

when subjected to a sudden loss in loop gain, and provides some intuition as to why the stability of AS systems is resilient to saturation. However, as was seen in §6.5.1, limiting the phase lag of the loop at lower frequencies reduced the amount of feedback available at lower frequencies. Hence, while AS systems are resilient to saturation, they do not maximize feedback over the functional bandwidth to the same extent as NS systems.

To circumvent this issue, we introduce the red anti-windup block shown in Figure 6.16, which we denote by $W(s)$. The resulting controller is called a *nonlinear dynamic compensator* (NDC). NDCs introduce a third degree of freedom into the design of the controller. Specifically, they allow the unsaturated loop to satisfy the less restrictive – but less resilient – Nyquist Stability Criterion, while ensuring that the saturated loop satisfies the more restrictive – but absolutely stable – Popov Stability Criterion (see Section 10.7 in [59]). In other words, the system enjoys the feedback performance improvements afforded to NS systems when the actuator is not saturated, and maintains the global stability guarantees afforded to AS systems when the actuator is saturated.

When the actuator is not saturated, the signals u and u_s shown in Figure 6.16 are equal. Hence, the input into $W(s)$ is zero, and the system operates as it does in Figure 6.14. When the actuator becomes saturated, the loop effectively breaks at the saturation element. It can be shown that the transfer function from the output of the saturation to the input of the saturation is given by

$$T(s) = \frac{L(s) - W(s)C(s)}{1 + W(s)C(s)},$$

where $T(s)$ is the transfer function found in Figure 6.17a. The transfer function $T(s)$ dictates the anti-windup response of the system when saturation occurs. Hence, the design procedure involves specifying a desired shape for $T(s)$, and computing the resulting $W(s)$. Since $C(s)$, and therefore $L(s)$, were determined in the feedback design step, we can solve for $W(s)$ in terms of $C(s)$, $L(s)$, and $T(s)$ to obtain

$$W(s) = \frac{L(s) - T(s)}{\big[1 + T(s)\big]C(s)}. \tag{6.39}$$

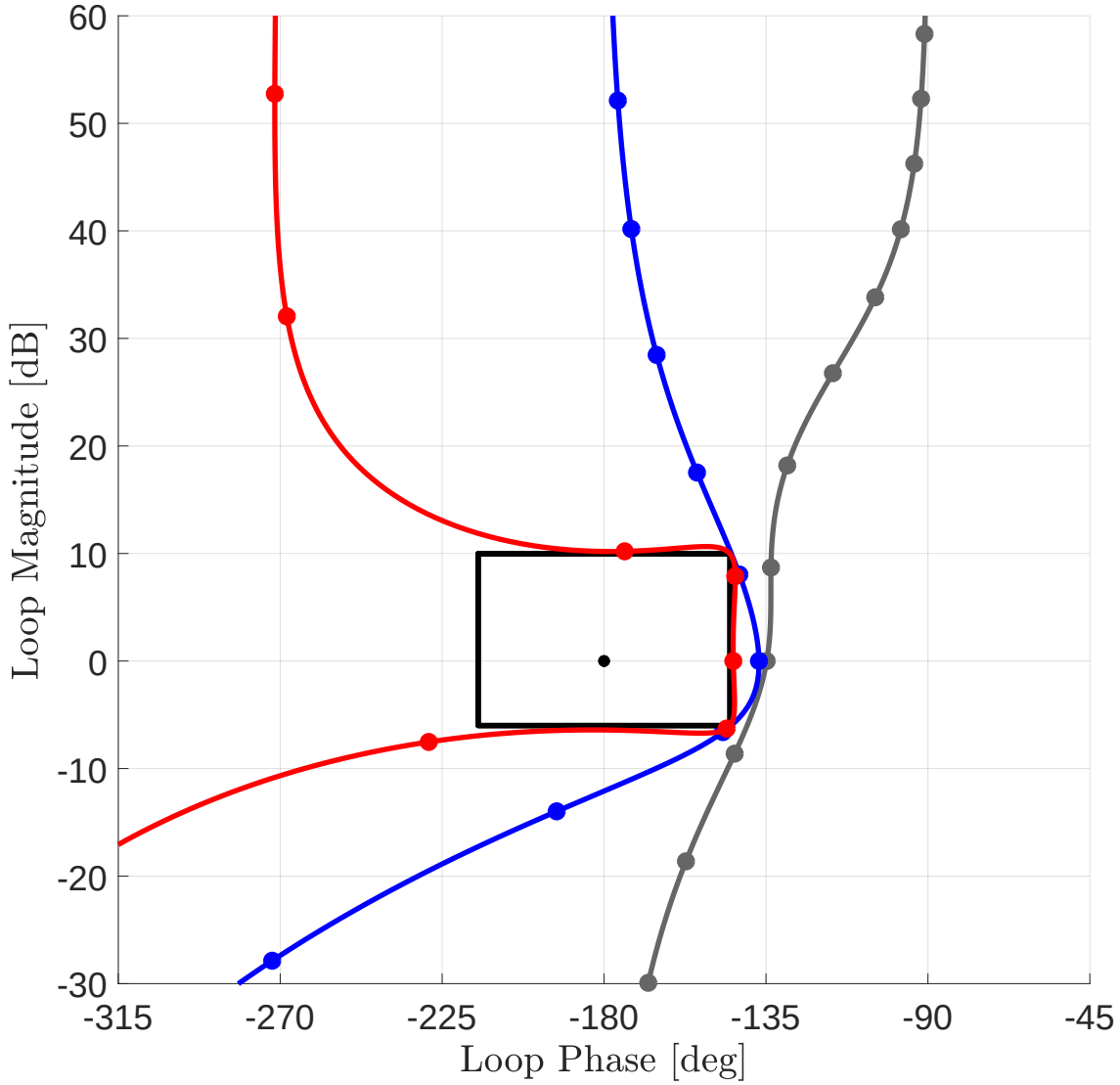
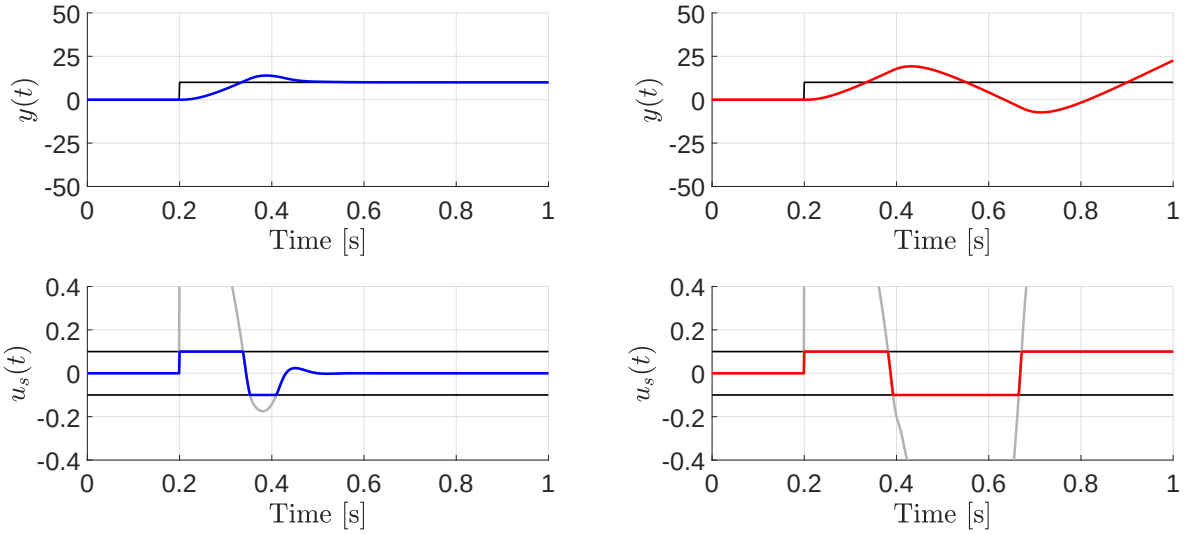


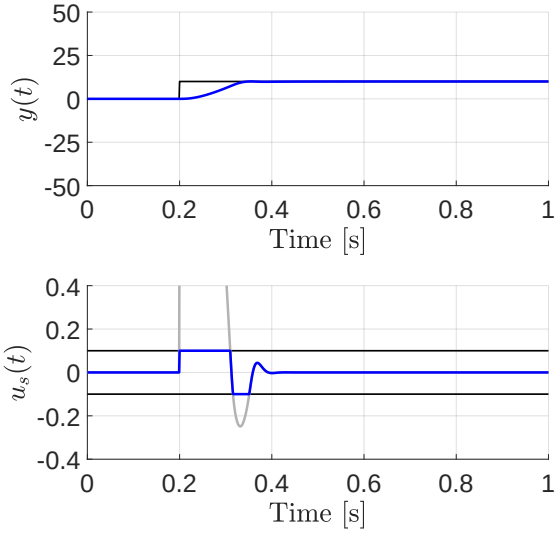
Figure 6.19: Nichols plot of $L(s)$ and $T(s)$ used for design of an NDC. The red and blue curves represent the loop shapes introduced in Figure 6.12a. The gray loop shape represents the desired $T(s)$ used to design $W(s)$.

Figure 6.19 shows the Nichols plots of the red and blue loop shapes introduced in 6.12a. The gray loop shape represents the specified loop shape of $T(s)$, from which we obtain $W(s)$.

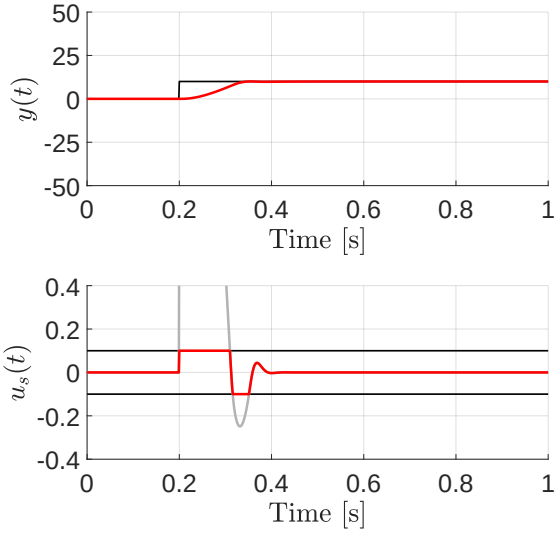


(a) Blue System w/o Anti-Windup

(b) Red System w/o Anti-Windup

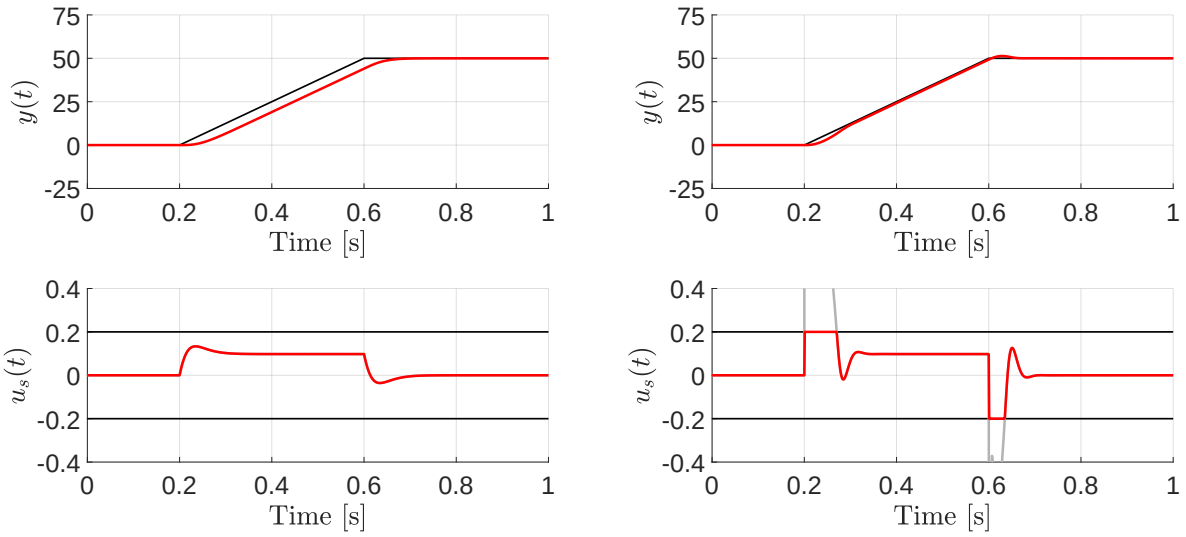


(c) Blue System w/ Anti-Windup



(d) Red System w/ Anti-Windup

Figure 6.20: Response of an NDC to a step in reference. (a) and (b) show the AS blue system and the NS red system from Figure 6.19 with $W(s) = 0$. (c) and (d) show the same systems but with $W(s)$ obtained from (6.39) using the $T(s)$ shown in Figure 6.19. In all cases, the feed-forward cutoff frequency is set to $\omega_{co,ff} = 10$ Hz.



(a) Ramp w/ $\omega_{co,ff} = 10$ Hz

(b) Ramp w/ $\omega_{co,ff} = 80$ Hz

Figure 6.21: Response of an NDC to a ramp in reference for the NS red system from Figure 6.19 with $W(s)$ as in (6.39). The feed-forward elements in (a) use $\omega_{co,ff} = 10$ Hz, whereas those in (b) use $\omega_{co,ff} = 80$ Hz.

While section 10.7 of [59] recommends that $T(s)$ be designed with a phase margin of approximately 90°, the given loop shape was found to perform better for the application at hand. The gain-crossover frequency of this loop was set to 20 Hz, twice that of the blue and red loop shapes. Since no exact formula exists for designing $T(s)$, this portion of the design was determined through trial and error.

Figures 6.20a-6.20d demonstrate the performance improvement realized by using nonlinear dynamic compensation. The top plots in Figures 6.20a and 6.20b show the blue and red systems' response to a step in the reference, with $W(s) = 0$, and with the feed-forward cutoff frequency $\omega_{co,ff} = 10$ Hz. Evidently, the AS blue system overshoots and stabilizes shortly thereafter, while the NS red system enters an unstable limit cycle. The bottom plots show the saturated and unsaturated actuator commands. Both systems are seen to hit and ride the saturation limits, with the blue system recovering about halfway through the simulation, and the red system entering a perpetual saturation cycle.

Figures 6.20c and 6.20d show the same two systems but with $W(s)$ as given in (6.39). The response of the two systems become indistinguishable. More importantly, the overshoot of the blue system and the unstable limit cycle exhibited by the red system have both vanished. This example highlights the ability of NDCs to stabilize NS systems. Moreover, it highlights the ability of NDCs to markedly improve the transient behavior of AS systems, despite the fact that such systems do not require NDCs to retain stability in the presence of saturation.

Figures 6.21a and 6.21b show the response of the red system to a ramp with $W(s)$ as in (6.39). Figure 6.21a shows the response with $\omega_{co,ff} = 10$ Hz, while Figure 6.21b shows the response with $\omega_{co,ff} = 80$ Hz. Clearly, the increased bandwidth of the feed-forward elements allows for better tracking in the latter case. In particular, the deviations of the red line from the reference in the top plot of Figure 6.21b show that the controller is following the reference virtually to the best of the system's ability.

In conclusion, the simulation results presented in Figures 6.20 and 6.21 include the elements from all three design steps presented in this section. The feedback controller $C(s)$ is designed for high performance disturbance rejection and sensitivity reduction, the feed-forward blocks $F(s)$ and $P(s)$ serve to improve the transient response of the system in nominal operation, and the anti wind-up element $W(s)$ provides the system with improved transient response and global stability guarantees in the presence of actuator saturation.

6.6 Mid-Level Controller

This section details the mid-level controller block depicted in Figure 6.1. The mid-level controller is comprised of an attitude prefilter, an attitude feedback controller, an attitude correction block, and a collective correction block. The details of each block are discussed its corresponding subsection below. The layout of the mid-level controller is shown in Figure 6.22, where circular nodes with the "$-$" and "$\times$" symbols represent quaternion conjugation and multiplication, respectively. In this section, we also assume that the quaternion resulting from each quaternion multiplication is reflected into the positive hemisphere such that its scalar element is non-negative.

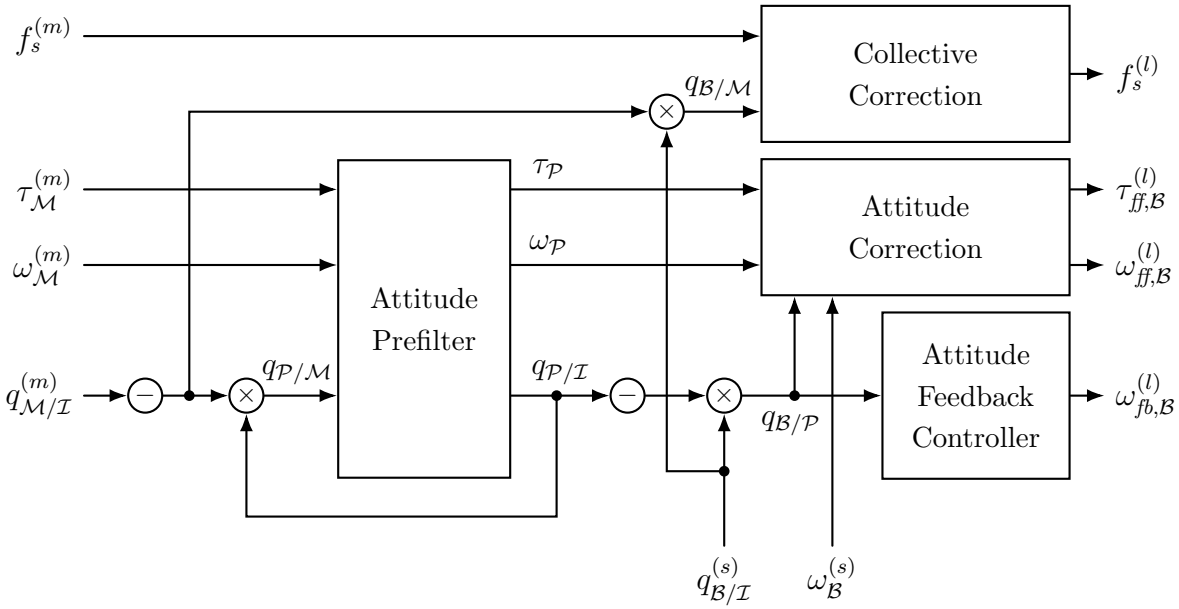


Figure 6.22: Layout of the mid-level controller block depicted in Figure 6.1. The circular nodes with the "−" symbol represent quaternion conjugation, whereas those with the "×" symbol represent quaternion multiplication. The quaternion resulting from each quaternion multiplication is reflected into the positive quaternion hemisphere such that its scalar element is non-negative.

6.6.1 Attitude Prefilter

The purpose of the prefilter is to smooth attitude commands so that they are more readily trackable by the low-level controller. The smoothed attitude trajectory it produces is used as a feed-forward command to the low-level controller, as well as a reference attitude for the attitude feedback controller.

More specifically, the input to the prefilter is an attitude trajectory specifying the orientation of the $\mathcal{M}$-frame relative to the $\mathcal{I}$-frame, and is comprised of the quantities $q_{\mathcal{P}/\mathcal{M}}$, $\omega_{\mathcal{M}}^{(m)}$, and $\tau_{\mathcal{M}}^{(m)}$.[6] The output of the prefilter is a smoothed attitude trajectory specifying the

[6]Note that the first input is the error quaternion quantifying the orientation of the $\mathcal{P}$-frame relative to the $\mathcal{M}$-frame.

orientation of the $\mathcal{P}$-frame relative to the $\mathcal{I}$-frame, and is comprised of the quantities $q_{\mathcal{P}/\mathcal{I}}$, $\omega_{\mathcal{P}}$, and $\tau_{\mathcal{P}}$.

The prefilter is designed with three properties in mind. First, the prefilter ensures that the output attitude trajectory is feasible with respect to the vehicle's attitude dynamics, prescribed feed-forward angular velocity limits $\omega_{ff,max} \in \mathbb{R}^3_{++}$, and prescribed feed-forward torque limits $\tau_{ff,max} \in \mathbb{R}^3_{++}$. Second, if the inputs are not feasible, or if all of the inputs are not specified, then the prefilter smooths the outputs according to a set of tunable parameters $G_p, G_d \in \mathbb{S}^3_{++}$, and thus generates a feasible output. Third, if the inputs are feasible, then the outputs are identical to the inputs.

The input-output behavior of the attitude prefilter is dictated by the following system of differential equations:

$$\dot{q}_{\mathcal{P}/\mathcal{I}} = \frac{1}{2}\Omega(\omega_{\mathcal{P}})\, q_{\mathcal{P}/\mathcal{I}}, \tag{6.40a}$$

$$J_{\mathcal{B}}\, \dot{\omega}_{\mathcal{P}} = u_{\tau,sat}\left(q_{\mathcal{P}/\mathcal{M}}, \omega_{\mathcal{P}}, \omega^{(m)}_{\mathcal{M}}, \tau^{(m)}_{\mathcal{M}}\right) - [\,\omega_{\mathcal{P}}\times]\, J_{\mathcal{B}}\, \omega_{\mathcal{P}}. \tag{6.40b}$$

where $J_{\mathcal{B}}$ is the assumed (constant) inertia tensor of the vehicle, and the $[\,\cdot\times]$ and $\Omega(\,\cdot\,)$ operators are defined as

$$[\,v\times] \coloneqq \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \quad \Omega(v) \coloneqq \begin{bmatrix} 0 & -v_0 & -v_1 & -v_2 \\ v_0 & 0 & v_2 & -v_1 \\ v_1 & -v_2 & 0 & v_0 \\ v_2 & v_1 & -v_0 & 0 \end{bmatrix}. \tag{6.41}$$

The term $u_\tau(\cdot)$ in (6.40b) is a control law given by

$$u_{\tau,sat}(q_{\mathcal{P}/\mathcal{M}}, \omega_{\mathcal{P}}, \omega^{(m)}_{\mathcal{M}}, \tau^{(m)}_{\mathcal{M}}) = u_\tau / \max\left(\|H_{\tau,ff,sat}\, u_\tau\|, 1\right), \tag{6.42a}$$

$$u_\tau \coloneqq u_{\tau,p} + u_{\tau,d} + u_{\tau,ff}, \tag{6.42b}$$

$$u_{\tau,p} \coloneqq -G_p\, q_{v,sat}\,, \tag{6.42c}$$

$$u_{\tau,d} \coloneqq -G_d \left(\omega_{\mathcal{P}} - \omega_{\mathcal{M}}^{(m)}\right), \tag{6.42d}$$

$$u_{\tau,ff} \coloneqq \tau_{\mathcal{M}}^{(m)} + \left[\omega_{\mathcal{M}}^{(m)}\times\right] J_{\mathcal{B}}\, \omega_{\mathcal{M}}^{(m)}, \tag{6.42e}$$

$$q_{v,sat} \coloneqq q_{\mathcal{P}/\mathcal{M},v} / \max\left(\|G_p G_d^{-1} H_{\omega,ff,sat}\, q_{\mathcal{P}/\mathcal{M},v}\|, 1\right), \tag{6.42f}$$

where $H_{\omega,ff,sat} \coloneqq \mathbf{diag}\,(\omega_{ff,max})^{-1}$ and $H_{\tau,ff,sat} \coloneqq \mathbf{diag}\,(\tau_{ff,max})^{-1}$. The expressions in (6.42a) and (6.42f) bound the magnitudes of the vectors u_τ and $q_{\mathcal{P}/\mathcal{M},v}$ according to the norm used in each equation (e.g. $\|\cdot\|_1$, $\|\cdot\|_2$, $\|\cdot\|_\infty$), while preserving their original directions. Since this control law drives the output trajectory's evolution, we set the prefilter's torque output as

$$\tau_{\mathcal{P}} = u_{\tau,sat}(q_{\mathcal{P}/\mathcal{M}}, \omega_{\mathcal{P}}, \omega_{\mathcal{M}}^{(m)}, \tau_{\mathcal{M}}^{(m)}).$$

The core of the control law is given in (6.42b). Conceptually, this expression can be separated into feedback and feed-forward components. The first two terms in (6.42b) constitute the feedback component, which can be understood as the quaternion equivalent of a proportional-derivative (PD) controller with gains G_p and G_d. The feedback component was inspired by the control law given in [46], but has two key differences. One difference is that it is non-smooth due to the projection of $q_{\mathcal{P}/\mathcal{M}}$ into the positive hemisphere (see Figure 6.22). This reflection is performed to ensure that the controller always corrects errors the "short way."[7] Another difference is that the vector term used in (6.42c) and defined in (6.42f) is saturated to ensure the response does not exceed the angular velocities specified in $\omega_{ff,max}$. Taking into account the saturation in (6.42a), we see that (6.42a) satisfies the first and second prefilter properties outlined above.

The third term in (6.42b) constitutes the feed-forward component of the control law. Given a feasible input, the feedback component will drive $q_{v,sat}$ and $\omega_{\mathcal{P}} - \omega_{\mathcal{M}}^{(m)}$ to zero, resulting

[7] Owing to the fact that the quaternions q and $-q$ both represent the same physical attitude, smooth quaternion controllers can correct attitude errors by going the "long way." For example, to correct a 1° error the "long way," the vehicle would perform a 359° turn in the opposite direction.

in an output trajectory that matches the input trajectory. If the prefilter is initialized on a feasible input trajectory, the error terms remain identically zero. Thus the feed-forward component satisfies the third prefilter property outlined above.

Practically speaking, the response of the prefilter is governed by four parameters: the angular velocity limits $\omega_{ff,max}$, the torque limits $\tau_{ff,max}$, and the symmetric-positive-definite gain matrices G_p and G_d. The behavior of the prefilter can be further manipulated by the choice of norm used in the saturations in (6.42a) and (6.42f). The gain matrices G_p and G_d should be selected such that prefilter produces a trackable attitude trajectory for the low-level controller. These matrices are typically diagonal, with a z-axis gain that is less than or equal to the x- and y-axis gains. The angular velocity limits should be selected to allow sufficient body rates in the feed-forward trajectory. These limits should also be selected such that any sum of feasible feed-forward and feedback angular rates do not violate the saturation limits of the gyroscope block in Figure 6.1. The torque limits should be selected conservatively to account for coupling with collective commands, and to provide sufficient torque margins for the low-level controller.[8] For the saturations in (6.42a) and (6.42f), we have found that using a 1-norm in (6.42a) more accurately accounts for the torque limitations of the vehicle, whereas using a 2-norm in (6.42f) generates a more isotropic (i.e. direction-agnostic) response. In practice, once the low-level and attitude feedback controllers are tuned, adequate prefilter parameters can be determined relatively easily through simulations.

6.6.2 Attitude Feedback Controller

The objective of the attitude feedback controller is to compensate for attitude errors between the prefilter's feed-forward $\mathcal{P}$-frame and the vehicle's body frame $\mathcal{B}$. The input into the attitude feedback controller is the quaternion error between the two frames, and is expressed as $q_{\mathcal{B}/\mathcal{P}}$. Due to the reflection applied after each quaternion multiplication in Figure 6.22, the scalar element of $q_{\mathcal{B}/\mathcal{P}}$ is always non-negative. The output of the attitude feedback controller

[8]For tractability, the torque limits used by the prefilter are static, and are not coupled to the collective command.

is a feedback angular velocity command, which we denote as $\omega_{fb,\mathcal{B}}^{(l)}$.

The attitude feedback controller is designed with two properties in mind. First, the controller prioritizes correcting tilt errors over correcting clock angle errors. This is the same property found in the *mixed full and reduced attitude controller* presented in [21,36]. Second, the controller limits the commanded angular velocities by applying a magnitude saturation similar to the ones used in (6.42a) and (6.42f). The angular velocity limits are specified using the vector $\omega_{fb,max} \in \mathbb{R}^3_{++}$, and allow tilt-rate and clock-rate limits to be specified independently.

The rationale for the first property follows from the fact that tilt errors strongly influence translational control, whereas clock angle errors do not. This property allows the clock angle response, and thus clock angle disturbances, to be decoupled from the tilt angle response. This property is particularly important during aggressive maneuvers where the z-axis torque is sacrificed to improve the collective and tilt torque allocation (see §6.4).

The motivation for the second property is twofold. First, as was the case with feedforward angular velocity generated by the prefilter, the feedback angular velocity needs to be saturated to ensure that the combined angular velocity command does not exceed the saturation limits of the gyroscope. Second, since no physical saturation exists between the mid-level and low-level controller, this artificial saturation allows the controller to be tuned aggressively for small errors while keeping the feedback signal within reasonable bounds for large errors.

Figure 6.23 provides a three-dimensional illustration of the attitude feedback problem in terms of the error quaternion $q_{\mathcal{B}/\mathcal{P}}$. Since the identity quaternion q_{id} represents a perfect alignment between two frames, the point $q_{\mathcal{B}/\mathcal{P}} = q_{id}$ corresponds to the desired attitude $\mathcal{P}$. All attitudes that deviate from $\mathcal{P}$ by only a clock angle error map to the zero-tilt-error manifold given by

$$\mathcal{Q}_{\mathcal{P}'/\mathcal{P}} \coloneqq \left\{ q : q = \begin{bmatrix} \cos\dfrac{\theta}{2} & 0 & 0 & \sin\dfrac{\theta}{2} \end{bmatrix}^{\mathsf{T}},\ \forall\theta \in [0,\ 4\pi] \right\}. \tag{6.43}$$

To develop the prioritized attitude feedback controller, we introduce an intermediate atti-

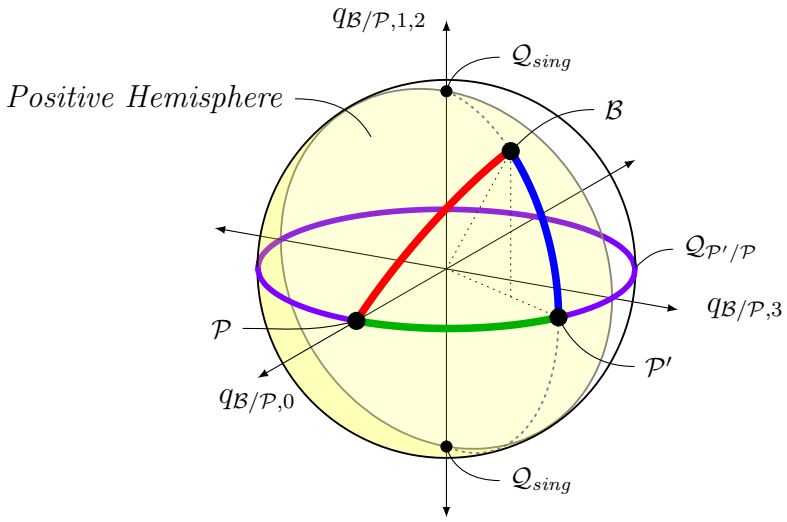


Figure 6.23: A three-dimensional illustration of the prioritized attitude feedback controller, and the relation between the current attitude $\mathcal{B}$, the desired attitude $\mathcal{P}$, and the closest zero-tilt-error attitude $\mathcal{P}'$. The positive hemisphere of the error quaternion $q_{\mathcal{B}/\mathcal{P}}$ is highlighted in yellow, while the set of all zero-tilt-error attitudes $\mathcal{Q}_{\mathcal{P}'/\mathcal{P}}$ is represented by the purple manifold. The set of singular attitudes is labeled as $\mathcal{Q}_{sing}$. The blue and green arcs represent the tilt and clock angle errors used by the prioritized attitude controller, whereas the red arc represents the total attitude error that a non-prioritized attitude controller would use.

tude $\mathcal{P}'$ by projecting $q_{\mathcal{B}/\mathcal{P}}$ into the horizontal plane of the $q_{\mathcal{B}/\mathcal{P}}$-space shown in Figure 6.23 and re-normalizing. This yields the error quaternion

$$q_{\mathcal{P}'/\mathcal{P}} \coloneqq \frac{\begin{bmatrix} q_{\mathcal{B}/\mathcal{P},0} & 0 & 0 & q_{\mathcal{B}/\mathcal{P},3} \end{bmatrix}^{\mathsf{T}}}{\left\| \begin{bmatrix} q_{\mathcal{B}/\mathcal{P},0} & 0 & 0 & q_{\mathcal{B}/\mathcal{P},3} \end{bmatrix}^{\mathsf{T}} \right\|_2}. \tag{6.44}$$

The definition in (6.44) ensures that $q_{\mathcal{P}'/\mathcal{P}} \in \mathcal{Q}_{\mathcal{P}'/\mathcal{P}}$, and minimizes the tilt error between $\mathcal{B}$ and $\mathcal{P}'$ (i.e. the length of the blue arc in Figure 6.23). However, the normalization used in (6.44) is singular for any $q_{\mathcal{P}'/\mathcal{P}}$ in the set

$$\mathcal{Q}_{sing} = \left\{ q : q = \begin{bmatrix} 0 & \cos\dfrac{\theta}{2} & \sin\dfrac{\theta}{2} & 0 \end{bmatrix}^{\mathsf{T}}, \ \forall \theta \in [0,\, 4\pi] \right\},$$

which is represented by the points at the top and bottom of the sphere in Figure 6.23.[9] That is, the normalization in (6.44) introduces a set of singular attitudes that have a combined roll and pitch error of 180° from the desired attitude $\mathcal{P}$. For example, the attitudes obtained by rolling 180° or pitching 180° away from $\mathcal{P}$ generate error quaternions that belong to $\mathcal{Q}_{sing}$, whereas the attitude obtained by yawing 180° away from $\mathcal{P}$ does not. The existence of these singular attitudes is simply an indication that the shortest path between $q_{\mathcal{B}/\mathcal{P}}$ and the zero-tilt-error set $\mathcal{Q}_{\mathcal{P}'/\mathcal{P}}$ is not unique for attitudes where $q_{\mathcal{B}/\mathcal{P}} \in \mathcal{Q}_{sing}$. Since any $q_{\mathcal{B}/\mathcal{P}} \in \mathcal{Q}_{sing}$ is an unstable equilibrium for the feedback law presented here, singular attitudes can addressed by introducing a perturbation to $q_{\mathcal{B}/\mathcal{P}}$ when it is sufficiently close to $\mathcal{Q}_{sing}$.

In minimizing the tilt error between $\mathcal{B}$ and $\mathcal{P}'$, we have consolidated the clock angle error into $q_{\mathcal{P}'/\mathcal{P}}$, and the remaining tilt error into the error quaternion

$$q_{\mathcal{B}/\mathcal{P}'} = \bar{q}_{\mathcal{P}'/\mathcal{P}} \otimes q_{\mathcal{B}/\mathcal{P}},$$

where the result is assumed to have been reflected into the positive hemisphere. Thus, the x and y vector components of $q_{\mathcal{P}'/\mathcal{P}}$ and the z vector component of $q_{\mathcal{B}/\mathcal{P}'}$ are zero. Consequently, the unsaturated feedback signal is obtained from the vector parts of $q_{\mathcal{B}/\mathcal{P}'}$ and $q_{\mathcal{P}'/\mathcal{P}}$ using

$$\omega_{fb,\mathcal{B}} = -\,\mathbf{diag}\left(k_{att}\right)\left(q_{\mathcal{B}/\mathcal{P}',v} + \, q_{\mathcal{P}'/\mathcal{P},v}\right),$$

where $k_{att} \in \mathbb{R}^3_+$ is a tunable feedback gain vector. The first two elements of k_{att} tune the tilt error response, whereas its third element tunes the clock angle error response. The saturated output of the attitude feedback controller is given by

$$\omega^{(l)}_{fb,\mathcal{B}} = \omega_{fb,\mathcal{B}} / \max\left(\|H_{\omega,fb,sat}\,\omega_{fb,\mathcal{B}}\|, 1\right), \tag{6.45}$$

where $H_{\omega,fb,sat} \coloneqq \mathbf{diag}\left(\omega_{fb,max}\right)^{-1}$.

In summary, the feedback law in (6.45) is tuned through two parameters: k_{att} and $\omega_{fb,max}$. The gain vector k_{att} is typically selected as

$$k_{att} = \left[\, k_{tilt} \;\; k_{tilt} \;\; k_{clock} \,\right]^{\mathsf{T}},$$

[9]Note that the vertical axis in Figure 6.23 represents two dimensions of $q_{\mathcal{B}/\mathcal{P}}$.

where $k_{tilt} \in \mathbb{R}_{++}$ and $k_{clock} \in \mathbb{R}_{+}$ determine the responsiveness of the feedback to tilt and clock angle errors, respectively. Moreover, since most multi-rotors are far more responsive along their x- and y-axes than they are along their z-axis, $k_{tilt} \gg k_{clock}$. In situations where clock angle error regulation is not desired, or is not possible (e.g. due to a failure), k_{clock} can be set to zero. The vector $\omega_{fb,max}$ is used to limit the maximum value commanded by the feedback controller, and can be used in conjunction with large feedback gains to ensure the vehicle responds well to both small and large attitude errors. As with the feedback gains, the z-component of $\omega_{fb,max}$ is typically selected such that it is much less than the other two components. The response can be further manipulated by selecting different norms in (6.45). We have found that the standard 2-norm provides good isotropic response.

Lastly, we note that the prioritized attitude controller presented in this section will always reduce tilt errors more rapidly than a non-prioritized quaternion attitude controller, provided that equivalent gains are used. However, this improvement in tilt response will come at the expense of the settling time of the entire response (i.e. clock angle errors will persist longer for the prioritized controller). This property follows from the fact that a non-prioritized controller is guided by the shorter (overall) red path in Figure 6.23, rather than the longer blue and green paths.

6.6.3 Attitude Correction

Much of the existing literature on multi-rotor control overlooks the adverse affects feed-forward control has when being applied to a vehicle with an off-nominal attitude. For example, applying feed-forward attitude commands to a vehicle with a clock angle error of 180° will cause the vehicle to respond in the opposite direction than intended. While this is arguably the clearest case where misaligned feed-forward control adversely affects tracking performance, this issue arises any time the vehicle's attitude deviates from the nominal attitude assumed by the feed-forward commands. Hence, to truly ensure full-envelope performance, feed-forward commands must compensate for discrepancies in the vehicle's attitude.

The attitude correction is applied to the feed-forward signals in the attitude correc-

tion block shown in Figure 6.22. This block takes as inputs the angular velocity $\omega_{\mathcal{P}}$ and torque $\tau_{\mathcal{P}}$ signals generated by the prefilter, and outputs the corrected feed-forward signals $\omega^{(l)}_{ff,\mathcal{B}}$ and $\tau^{(l)}_{ff,\mathcal{B}}$. The relationship between $\omega_{\mathcal{P}}$ and $\omega^{(l)}_{ff,\mathcal{B}}$ is given by

$$\omega^{(l)}_{ff,\mathcal{B}} = C_{\mathcal{B}/\mathcal{P}}\, \omega_{\mathcal{P}},$$

where $C_{\mathcal{B}/\mathcal{P}} \in \mathrm{SO}(3)$ is the direction-cosine-matrix associated with the error quaternion $q_{\mathcal{B}/\mathcal{P}}$. We would like to apply the same correction to the angular acceleration vector generated by the prefilter, which is obtained from $\omega_{\mathcal{P}}$ and $\tau_{\mathcal{P}}$ through

$$\alpha_{\mathcal{P}} = J_{\mathcal{B}}^{-1}\big(\, \tau_{\mathcal{P}} - [\, \omega_{\mathcal{P}} \times]\, J_{\mathcal{B}}\, \omega_{\mathcal{P}}\,\big).$$

Rotating this angular acceleration into the $\mathcal{B}$-frame, we obtain the feed-forward torque required to realize the inertial angular acceleration vector specified by the prefilter:

$$\tau^{(l)}_{ff,\mathcal{B}} = J_{\mathcal{B}}\, C_{\mathcal{B}/\mathcal{P}}\, \alpha_{\mathcal{P}} + \Big[\, \omega^{(s)}_{\mathcal{B}} \times\Big]\, J_{\mathcal{B}}\, \omega^{(s)}_{\mathcal{B}}. \tag{6.46}$$

Since the gyroscope measurements $\omega^{(s)}_{\mathcal{B}}$ are typically noisy, and since the second term in (6.46) is typically negligible, the torque correction can be simplified to

$$\tau^{(l)}_{ff,\mathcal{B}} = J_{\mathcal{B}}\, C_{\mathcal{B}/\mathcal{P}}\, J_{\mathcal{B}}^{-1}\big(\, \tau_{\mathcal{P}} - [\, \omega_{\mathcal{P}} \times]\, J_{\mathcal{B}}\, \omega_{\mathcal{P}}\,\big). \tag{6.47}$$

This simplification allows us to eliminate the $\omega^{(s)}_{\mathcal{B}}$ signal entering the attitude correction block in Figure 6.22 and the mid-level controller block in Figure 6.1.

6.6.4 Collective Correction

The collective correction is the final piece of the mid-level controller, and is responsible for adjusting the collective command to any attitude error between the $\mathcal{B}$-frame and $\mathcal{P}$-frame. A multi-rotor's collective channel is typically far more responsive than the vehicle's attitude, and in this section may be regarded as having an instantaneous response. Nevertheless, if an under-actuated multi-rotor vehicle (e.g. a quad-rotor), deviates from its desired attitude, then no instantaneous command can generate the desired acceleration vector.

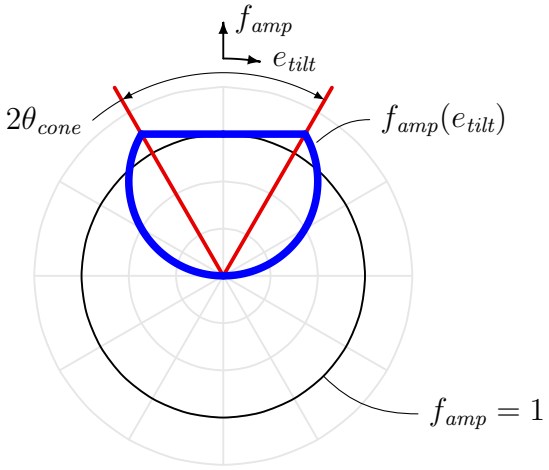


Figure 6.24: A polar plot showing the collective correction applied as a function of tilt error when $\theta_{cone} = 30°$. The correction amplifies the commanded collective for $|e_{tilt}| \leq \theta_{cone}$, attenuates the command when $\theta_{cone} < |e_{tilt}| \leq 90°$, and sets the command to zero when $90° < |e_{tilt}| \leq 180°$.

The designer is faced with two options. One option is to increase the collective as the tilt angle increases so that the nominal acceleration is produced in the desired direction. This option inevitably introduces even larger lateral errors, and is only viable for tilt angles that are less than 90°. The second option is to reduce the collective with increasing tilt error, thus reducing the lateral errors introduced by the tilt error. While this option can be used over the entire range of tilt errors, it introduces errors in both the nominal vertical and lateral directions.

The proposed collective correction is a blend of these two options, and is formulated as an amplification that is a function of the tilt error e_{tilt}. The tilt error is given as a function of the error quaternion $q_{\mathcal{B}/\mathcal{M}}$ seen in Figure 6.22 as follows:

$$e_{tilt} = \arccos\left(z_{\mathcal{B}}^{\mathsf{T}}\, C_{\mathcal{B}/\mathcal{M}}\, z_{\mathcal{M}}\right),$$

where $C_{\mathcal{B}/\mathcal{M}} \in \mathrm{SO}(3)$ is the direction-cosine-matrix associated with $q_{\mathcal{B}/\mathcal{M}}$, and $z_{\mathcal{B}}$ and $z_{\mathcal{M}}$ are the unit vectors aligned with the z-axes of the $\mathcal{B}$-frame and $\mathcal{M}$-frame, respectively. We

define the amplification function by

$$f_{amp}\left(e_{tilt}\right) = \max\left(\frac{\cos e_{tilt}}{\cos^2\theta_{sat}}, 0\right), \tag{6.48}$$

where

$$\theta_{sat} \coloneqq \max\left(\min\left(e_{tilt}, \theta_{cone}\right), -e_{tilt}\right).$$

The angle $\theta_{cone} \in [\,0°, 90°\,)$ is a tunable parameter that determines the tilt error at which $f_{amp}(\cdot)$ switches from the first option to the second. Notably, (6.48) is non-singular over the entire tilt error envelope. Figure 6.24 shows a polar plot of the amplification function as a function of tilt error for $\theta_{cone} = 30°$. The output of the collective correction block is given by

$$f_s^{(l)} = f_{amp}(e_{tilt}) f_s^{(m)}.$$

Chapter 7

CONCLUSIONS

7.1 Summary

The objective of this dissertation was to develop novel feed-forward guidance and feedback control methodologies.

In Chapter 2, a comprehensive layout of successive convexification was provided in the context of a generalized rocket landing problem. Specifically, a general framework for converting a free-final-time optimal control framework into a solvable sequence of convex optimization problems was given; a novel class of constraints – state-triggered constraints – that facilitate the formulation of discrete decisions was introduced; and real-time computation results were presented. This chapter had two key takeaways. First, state-triggered constraints are a reliable and practical method for formulating discrete decisions into a fast continuous optimization framework. Second, successive convexification is capable of consistently handling complicated nonlinear dynamics, free-final-time and free-ignition-time formulations, and discrete decision making all while generating results in near-real-time to real-time.

In Chapters 3-5, successive convexification and state-triggered constraints were applied to a number of quad-rotor motion planning problems involving aggressive maneuvers and obstacle avoidance. The key advantage of the proposed approach was that pertinent system and environmental constraints could be directly transcribed into the problem formulation. This allowed a richer set of motion planning problems, some involving mixed-integer-like constraints, to be solved consistently in near real-time.

Lastly, in Chapter 6, a comprehensive high performance multi-rotor feedback control architecture was proposed. This architecture was specifically designed to provide robustness to plant uncertainties and external disturbances for a variety of feed-forward guidance

methodologies. The core of this architecture relied on a blend of advanced classical control techniques and convex optimization to properly prioritize and cope with multi-rotor actuator saturations.

7.2 Future Work

The successive convexification framework has been shown to be highly adaptable to a range of complex motion planning problems. However, like virtually any other nonlinear programming methodology, the statements that can be made about its convergence properties are somewhat limited when compared to existing convex optimization algorithms (e.g. interior-point-method algorithms).

Works like [18, 61–63] have attempted to rigorously characterize the convergence properties of sequential convex programming techniques like successive convexification. However, these works perform their analysis in a manner that is largely agnostic to the specific optimal control problem at hand. Consequently, and understandably, the general theoretical statements that can then be made about such algorithms are somewhat limited. Moreover, they do not directly address the important questions that future practitioners want answered – e.g., Given a vehicle, a fixed problem size, and a set of possible boundary conditions, how many iterations are required to reach feasibility or local optimality? Which set of boundary conditions have guaranteed convergence within a set number of iterations? What does a known offline solution afford an online solution process?

One possible avenue to resolving questions of this nature may involve a mixture of numerical and theoretical techniques. For example, it is conceivable that a complex guidance problem could be solved offline over a coarse grid covering an area of interest. Then, theoretical statements may provide guarantees that neighboring trajectories can be computed online within a bounded number of iterations and to within a prescribed level of feasibility or optimality. For real-time applications, a result of this type would be substantially more informative than, for example, traditional computational complexity bounds used to characterize existing optimization algorithms. Such bounds typically loosely upper bound the

computational complexity of algorithms in terms of the size of the problem – a quantity that remains largely static beyond the initial phases of guidance algorithm design.

Another aspect that the work presented in this dissertation has not explored is the marriage of fast dynamically feasible guidance methodologies (such as successive convexification) with algorithms like Rapidly-Exploring Random Tree (RRT). The former is good at finding local physically feasible solutions, and has been shown to be quite tolerant to different initializations. However, it is nonetheless susceptible to local regions of attraction. These regions can be benign – as they often are in rocket landing problems – or quite pathological – as they can be in problems with a large number of obstacles. The latter is not as good at finding complex dynamically feasible trajectories, but is remarkably good at finding candidate paths through complex environments. Thus, it stands to reason that such algorithms can be employed in unison in order to complement each other.

Lastly, virtually anyone working in the field of multi-rotor control would agree that the nested architecture presented in the previous chapter largely makes sense. When such architectures are proposed, very little – if any – rigor is presented to justify the architecture. While rigor is applied within each level of the architecture, the architectural decisions themselves are made largely based on intuition. Clearly, in the case of multi-rotors, the frequency separation between the inner loops and the outer loops offer some justification for the hierarchical nature of the architecture. However, many of the details presented herein (e.g. the force and attitude corrections, the interplay between the low-level controller and allocator) were designed by intuition. This raises an interesting question: Given a cyber-physical system, how does one go about constructing a reasonable architecture in a systematic manner? While the answer to this question is not likely to provide a surprising answer in the realm of multi-rotors, it will more than likely be very useful to the next generation of robotic systems which may be severely under-actuated and may be comprised of a set of heterogeneous actuators and manipulators.

BIBLIOGRAPHY

[1] B. Açıkmeşe. Application of lexicographic goal programming with convex optimization in control systems. In *AIAA Guidance, Navigation, and Control Conference*, page 5103, Boston, MA, USA, August 2013.

[2] B. Açıkmeşe and L. Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 47(2):341–347, 2011.

[3] B. Açıkmeşe, J. M. Carson III, and L. Blackmore. Lossless convexification of non-convex control bound and pointing constraints of the soft landing optimal control problem. *IEEE Transactions on Control Systems Technology*, 21(6):2104–2113, 2013.

[4] B. Açıkmeşe and S. R. Ploen. A Powered Descent Guidance Algorithm for Mars Pinpoint Landing. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, USA, 2005.

[5] B. Açıkmeşe and S. R. Ploen. Convex programming approach to powered descent guidance for Mars landing. *AIAA Journal of Guidance, Control and Dynamics*, 30(5):1353–1366, 2007.

[6] B. Açıkmeşe, D. P. Scharf, E. A. Murray, and F. Y. Hadaegh. A convex guidance algorithm for formation reconfiguration. In *AIAA Guidance, Navigation, and Control Conference*, Keystone, CO, USA, 2006.

[7] P. J. Antsaklis and A. N. Michel. *A Linear Systems Primer*. Birkhauser, Boston, MA, USA, 2007.

[8] D. M. Azimov. Analytic solutions for intermediate-thrust arcs of rocket trajectories in a newtonian field. *Journal of Applied Mathematics and Mechanics*, 60(3):421–427, 1996.

[9] A. Bemporad, C. A. Pascucci, and C. Rocchi. Hierarchical and hybrid model predictive control of quadcopter air vehicles. *IFAC Proceedings Volumes*, 42(17):14 – 19, 2009. 3rd IFAC Conference on Analysis and Design of Hybrid Systems.

[10] L. D. Berkovitz. *Optimal Control Theory*. Springer-Verlag, 1975.

[11] J. T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[12] L. T. Biegler. Recent advances in chemical process optimization. *Chemie-Ingenieur-Technik*, 86(7):943–952, 2014.

[13] L. Blackmore. Autonomous precision landing of space rockets. *NAE Winter Bridge on Frontiers of Engineering*, 4(46), 2016.

[14] L. Blackmore, B. Açıkmeşe, and J. M. Carson III. Lossless convexfication of control constraints for a class of nonlinear optimal control problems. *System and Control Letters*, 61(4):863–871, 2012.

[15] L. Blackmore, B. Açıkmeşe, and J. M. Carson III. Lossless convexication of control constraints for a class of nonlinear optimal control problems. *Systems and Control Letters*, 61:863–871, 2012.

[16] L. Blackmore, B. Açıkmeşe, and D. P. Scharf. Minimum-landing-error powered-descent guidance for mars landing using convex optimization. *Journal of Guidance, Control, and Dynamics*, 33(4):1161–1171, July 2010.

[17] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4(4), 1995.

[18] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone. Gusto: Guaranteed sequential trajectory optimization via sequential convex programming. *arXiv e-prints*, March 2019. arXiv:1903.00155.

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

[20] J. V. Breakwell and J. F. Dixon. Minimum-Fuel Rocket Trajectories Involving Intermediate-Thrust Arcs. *Journal of Optimization Theory and Applications*, 17(5):465–479, 1975.

[21] D. Brescianini, M. Hehn, and R. D'Andrea. Nonlinear quadrocopter attitude control. *Institute for Dynamic Systems and Control Technical Report*, October 2013.

[22] J. M. Carson III, B. Açıkmeşe, L. Blackmore, and A. A. Wolf. Capabilities of convex powered-descent guidance algorithms for pinpoint and precision landing. In *Aerospace Conference, 2011 IEEE*, pages 1–8. IEEE, 2011.

[23] J. Casoliva. *Spacecraft Trajectory Generation by Successive Approximation for Powered Descent and Cyclers*. PhD thesis, University of California, Irvine, 2013.

[24] J.-H. Chuang. Potential-based modeling of three-dimensional workspace for obstacle avoidance. *IEEE Transactions on Robotics and Automation*, 14(5):778–785, October 1998.

[25] R. W. Cottle, J.-S. Pang, and R. E. Stone. *Linear Complementarity Problem*. Academic Press Limited, San Diego, CA, USA, 1992.

[26] M. Cutler, N. Kemal Ure, B. Michini, and J. P. How. Comparison of fixed and variable pitch actuators for agile quadrotors. In *AIAA Guidance, Navigation, and Control Conference*, Portland, OR, USA, August 2011.

[27] A. Domahidi, E. Chu, and S. Boyd. Ecos: An socp solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, Zurich, Switzerland, July 2013.

[28] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones. Efficient interior point methods for multistage problems arising in receding horizon control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 668–674. IEEE, 2012.

[29] R. C. Dorf and R. H. Bishop. *Modern Control Systems, 11th Edition*. Pearson Prentice Hall, 2008.

[30] C. D'Souza. An optimal guidance law for planetary landing. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pages 1376–1381, New Orleans, LA, USA, 1997.

[31] G.-X. Du, Q. Quan, and K.-Y. Cai. Controllability analysis and degraded control for a class of hexacopters subject to rotor failures. *Journal of Intelligent and Robotic Systems*, September 2014.

[32] D. Dueri, B. Açıkmeşe, D. P. Scharf, and M. W. Harris. Customized real-time interior-point methods for onboard powered-descent guidance. *Journal of Guidance, Control, and Dynamics*, pages 1–16, 2016.

[33] D. Dueri, J. Zhang, and B. Açıkmeşe. Automated custom code generation for embedded, real-time second order cone programming. In *19th IFAC World Congress*, pages 1605–1612, 2014.

[34] M. Faessler, D. Falanga, and D. Scaramuzza. Thrust mixing, saturation, and body-rate control for accurate aggressive quadrotor flight. *IEEE Robotics and Automation Letters*, 2(2), April 2017.

[35] F. Fahroo and I. M. Ross. Direct trajectory optimization by a Chebyshev pseudospectral method. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):160–166, 2002.

[36] M. Fassler. *Quadrotor Control for Accurate Agile Flight*. PhD thesis, University of Zurich, Business, Economics and Informatics, April 2018.

[37] M. J. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. *http://cvxr.com/cvx*, March 2014.

[38] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[39] S. Haddadin, R. Belder, and A. Albu-Schäffer. Dynamic motion planning for robots in partially unknown environments. *IFAC Proceedings Volumes*, 44(1):6842–6850, 2011.

[40] A. J. Hanson. *Visualizing Quaternions*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[41] M. W. Harris and B. Açıkmeşe. Lossless convexification for a class of optimal control problems with first order state constraints. In *American Control Conference*, Washington, D.C., USA, June 2013.

[42] M. W. Harris and B. Açıkmeşe. Lossless convexification of non-convex optimal control problems for state constrained linear systems. *Automatica*, 50(9):2304–2311, 2014.

[43] W. P. M. H. Heemels, J. M. Schumacher, and S. Weiland. Linear Complementarity Systems. *SIAM Journal of Applied Math*, 60(4):1234–1269, 2000.

[44] B. Houska, H. J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.

[45] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari. Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12):3238–3251, 2014.

[46] S. M. Joshi, A. G. Kelkar, and J. T.-Y. Wen. Robust attitude stabilization of spacecraft using nonlinear quaternion feedback. *IEEE Transactions on Automatic Control*, 40(10), October 1995.

[47] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart. Fast nonlinear model predictive control for multicopter attitude tracking on so(3). In *2015 IEEE Conference on Control Applications (CCA)*, pages 1160–1166, September 2015.

[48] A. R. Klumpp. Apollo lunar descent guidance. *Automatica*, 10:133–146, 1974.

[49] A. L. Kornhauser and P. M. Lion. Optimal Deterministic Guidance for Bounded-Thrust Spacecrafts. *Celestial Mechanics*, 5(3):261–281, 1972.

[50] B. Landry, R. Deits, P. R. Florence, and R. Tedrake. Aggressive quadrotor flight throgh cluttered environments using mixed integer programming. In *IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, May 2016.

[51] D. F. Lawden. *Optimal Trajectories for Space Navigation*. Butterworths, London, England, 1963.

[52] U. Lee and M. Mesbahi. Dual quaternions, rigid body mechanics, and powered-descent guidance. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3386–3391, 2012.

[53] U. Lee and M. Mesbahi. Optimal powered descent guidance with 6-dof line of sight constraints via unit dual quaternions. In *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, FL, January 2015.

[54] U. Lee and M. Mesbahi. Constrained Autonomous Precision Landing via Dual Quaternions and Model Predictive Control. *AIAA Journal of Guidance, Control, and Dynamics*, 40(2):292–308, 2017.

[55] S. Liu, M. Watterson, S. Tang, and V. Kumar. High speed navigation for quadrotors with limited onboard sensing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1484–1491, May 2016.

[56] X. Liu and P. Lu. Solving nonconvex optimal control problems by convex optimization. *AIAA Journal of Guidance, Control, and Dynamics*, 37(3):750–765, 2014.

[57] X. Liu, P. Lu, and B. Pan. Survey of Convex Optimization for Aerospace Applications. *Astrodynamics*, 1(1):1–23, 2017.

[58] X. Liu, Z. Shen, and P. Lu. Entry trajectory optimization by second-order cone programming. *Journal of Guidance, Control, and Dynamics*, 39(2):227–241, 2015.

[59] B. J. Lurie and P. J. Enright. *Classical Feedback Control with MATLAB and Simulink, 2nd Edition*. CRC Press, 2012.

[60] D. Malyuta, T. P. Reynolds, M. Szmuk, M. Mesbahi, B. Açıkmeşe, and J. M. Carson III. Discretization performance and accuracy analysis for the rocket powered descent guidance problem. In *AIAA SciTech Forum*, San Diego, CA, USA, 2019.

[61] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe. Successive convexification of non-convex optimal control problems with state constraints. *ArXiv e-prints*, January 2017. arXiv:1701.00558.

[62] Y. Mao, M. Szmuk, and B. Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In *IEEE 55th Conference on Decision and Control*, pages 3636–3641, Las Vegas, NV, USA, December 2016.

[63] Y. Mao, M. Szmuk, and B. Açıkmeşe. Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems. *ArXiv e-prints*, April 2018. arXiv:1804.06539.

[64] J.-P. Marec. *Optimal Space Trajectories*. Elsevier Scientific Publishing Company, Amsterdam, Netherlands, 1979.

[65] J. Mattingley and S. Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[66] J. S. Meditch. On the Problem of Optimal Thrust Programming For a Lunar Soft Landing. *IEEE Transactions on Automatic Control*, 9(4):477–484, 1964.

[67] D. Mellinger. *Trajectory Generation and Control for Quadrotors*. PhD thesis, University of Pennsylvania, January 2012.

[68] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. *IEEE International Conference on Robotics and Automation*, May 2011.

[69] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, May 2011.

[70] D. Mellinger, A. Kushleyev, and V. Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *IEEE International Conference on Robotics and Automation*, Saint Paul, MN, USA, May 2012.

[71] M. W. Mueller and R. D'Andrea. A model predictive controller for quadrocopter state interception. In *2013 European Control Conference (ECC)*, pages 1383–1389, July 2013.

[72] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli. Fast nonlinear model predictive control for unified trajectory optimization and tracking. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1398–1404, May 2016.

[73] J. Nocedal and S. J. Wright. *Numerical Optimization 2nd Edition*. Springer, 2006.

[74] S. R. Ploen, B. Açıkmeşe, and A. Wolf. A comparison of powered descent guidance laws for Mars pinpoint landing. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.

[75] T. P. Reynolds, M. Szmuk, Malyuta D., M. Mesbahi, B. Açıkmeşe, and J. M. Carson III. Dual quaternion based powered descent guidance with state-triggered constraints. *arXiv e-prints*, April 2019. arXiv:1904.09248.

[76] T. P. Reynolds, M. Szmuk, D. Malyuta, M. Mesbahi, B. Açıkmeşe, and J. M. Carson III. A State-Triggered Line of Sight Constraint for 6-DoF Powered Descent Guidance Problems. In *AIAA SciTech Forum*, San Diego, CA, USA, 2019.

[77] A. Richards and J. P. How. Mixed-integer Programming for Control. In *Proceedings of the American Control Conference*, pages 2676–2683, Portland, OR, USA, 2005.

[78] A. Richards, T. Schouwenaars, J. P. How, and E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764, 2002.

[79] D. P. Scharf, B. Açıkmeşe, D. Dueri, J. Casoliva, and J. Benito. Implementation and experimental demonstration of onboard powered descent guidance. *AIAA Journal of Guidance, Control, and Dynamics*, 40(2):213–229, 2016.

[80] D. P. Scharf, M. W. Regehr, D. Dueri, B. Açıkmeşe, G. M. Vaughan, J. Benito, H. Ansari, M. Aung, A. Johnson, D. Masten, S. Nietfeld, J. Casoliva, and S. Mohan. Adapt demonstrations of onboard large-divert guidance with a reusable launch vehicle. In *IEEE Aerospace Conference*, March 2014.

[81] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design, 2nd Edition*. John Wiley & Sons Ltd., 2005.

[82] G. Stein. Respect the unstable. *IEEE Control Systems Magazine*, August 2003.

[83] M. Szmuk and B. Açıkmeşe. Successive convexification for 6-dof mars rocket powered landing with free-final-time. In *AIAA Guidance, Navigation, and Control Conference*, page 0617, Kissimmee, FL, USA, 2018.

[84] M. Szmuk, B. Açıkmeşe, and A. W. Berning Jr. Successive convexification for fuel-optimal powered landing with aerodynamic drag and non-convex constraints. In *AIAA Guidance, Navigation, and Control Conference*, page 0378, San Deigo, CA, USA, 2016.

[85] M. Szmuk, U. Eren, and B. Açıkmeşe. Successive convexification for mars 6-dof powered descent landing guidance. In *AIAA Guidance, Navigation, and Control Conference*, page 1500, Grapevine, TX, USA, 2017.

[86] M. Szmuk, D. Malyuta, T. P. Reynolds, M. S. Mceowen, and B. Açıkmeşe. Real-time quad-rotor path planning using convex optimization and compound state-triggered constraints. *arXiv e-prints*, February 2019. arXiv:1902.09149.

[87] M. Szmuk, C. A. Pascucci, and B. Açıkmeşe. Real-time quad-rotor path planning for mobile obstacle avoidance using convex optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, October 2018.

[88] M. Szmuk, C. A. Pascucci, D. Dueri, and B. Açıkmeşe. Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, BC, Canada, September 2017.

[89] M. Szmuk, T. P. Reynolds, and B. Açıkmeşe. Successive convexification for real-time 6-dof powered descent guidance with state-triggered constraints. *arXiv e-prints*, November 2018. arXiv:1811.10803.

[90] M. Szmuk, T. P. Reynolds, B. Açıkmeşe, M. Mesbahi, and J. 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 Deigo, CA, USA, 2019.

[91] S. Tang and V. Kumar. Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload. In *IEEE International Conference on Robotics and Automation*, Seattle, WA, USA, May 2015.

[92] U. Topcu, J. Casoliva, and K. D. Mease. Fuel efficient powered descent guidance for mars landing. In *Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, USA, August 2005.

[93] U. Topcu, J. Casoliva, and K. D. Mease. Minimum-fuel powered descent for mars pinpoint landing. *Journal of Spacecraft and Rockets*, 44(2):324–331, 2007.

[94] Z. Wang and M. J. Grant. Constrained trajectory optimization for planetary entry via sequential convex programming. *AIAA Journal of Guidance, Control, and Dynamics*, 40(10):2603–2615, 2017.

[95] M. Watterson and V. Kumar. Safe receding horizon control for aggressive mav flight with limited range sensing. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3235–3240, September 2015.

Appendix A
TRANSFER FUNCTIONS

A.1 Linear Time-Varying (LTV) Systems

Consider a general continuous-time linear time-varying (LTV) dynamical system given by the state-space realization

$$\dot{x}(t) = A(t)x(t) + B(t)u(t), \tag{A.1a}$$

$$y(t) = C(t)x(t) + D(t)u(t), \tag{A.1b}$$

where $A(t) \in \mathbb{R}^{n_x \times n_x}$, $B(t) \in \mathbb{R}^{n_x \times n_u}$, $C(t) \in \mathbb{R}^{n_y \times n_x}$, and $D(t) \in \mathbb{R}^{n_y \times n_u}$ are the time-varying system matrices, $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $u(t) \in \mathbb{R}^{n_u}$ is a scalar control input, and $y(t) \in \mathbb{R}^{n_y}$ is a scalar output. It can be shown that the solution to (A.1a) is given by

$$x(t_2) = \Phi_A(t_2, t_1)x(t_1) + \int_{t_1}^{t_2} \Phi_A(t_2, \xi)B(\xi)u(\xi)d\xi,$$

where $\Phi_A(\cdot, \cdot) \in \mathbb{R}^{n_x \times n_x}$ is the state-transition-matrix. This matrix represents the homogeneous (i.e. zero-input) solution of (A.1a), and evolves according to the following initial-value problem

$$\dot{\Phi}_A(t, t_0) = A(t)\Phi_A(t, t_0), \quad \Phi_A(t_0, t_0) = I, \tag{A.2}$$

where $I \in \mathbb{R}^{n_x \times n_x}$ denotes the identity matrix. From (A.1b) it follows that

$$y(t_2) = C(t_2)\left[\Phi_A(t_2, t_1)x(t_1) + \int_{t_1}^{t_2} \Phi_A(t_2, \xi)B(\xi)u(\xi)d\xi\right] + D(t_2)u(t_2). \tag{A.3}$$

A.2 Linear Time-Invariant (LTI) Systems

A linear time-invariant (LTI) system is one whose system matrices $A(t) = A$, $B(t) = B$, $C(t) = C$, and $D(t) = D$ are constant with time. It is important to note that although the

system is called time-invariant, $x(t)$ and $y(t)$ *are not* necessarily constant with time. The *time-invariant* modifier strictly pertains to the system matrices A, B, C, and D.

An LTI system can be represented in the time-domain using the following state-space realization:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{A.4a}$$

$$y(t) = Cx(t) + Du(t). \tag{A.4b}$$

In the LTI special case, the state-transition-matrix can be represented by the matrix exponential:

$$e^{A(t_2-t_1)} = \Phi_A(t_2, t_1), \tag{A.5}$$

where, from (A.2), it follows that $\dot{e}^{A(t_2-t_1)} = Ae^{A(t_2-t_1)}$. Although the system in (A.4) is both linear and time-invariant, the terms inside of the state-transition-matrix in (A.5) can be both time-varying and nonlinear.

A.3 What Are Transfer Functions?

A transfer function is a *unique* frequency-domain representation of the input-output relationship of a single-input single-output (SISO) LTI system, for whom $n_u = n_y = 1$. Any time one talks of a transfer function, it is implied that the underlying system is LTI. Although frequency-domain representations of multiple-input multiple-output (MIMO) LTI systems exist and are widely used [81], they are beyond the scope of this discussion.

The transfer function of (A.4) is obtained by taking the Laplace transform on both sides:

$$sx(s) - x(0) = Ax(s) + Bu(s), \tag{A.6a}$$

$$y(s) = Cx(s) + Du(s), \tag{A.6b}$$

where $s \in \mathbb{C}$ is the complex Laplace variable, $x(s)$, $u(s)$, and $y(s)$ are the Laplace transforms of their respective time-domain signals, and $x(0) \in \mathbb{R}^{n_x}$ is the initial condition of the system.

Solving (A.6a) for $x(s)$ and substituting into (A.6b), we obtain

$$y(s) = C(sI - A)^{-1}\big[Bu(s) + x(0)\big] + Du(s). \tag{A.7}$$

The quantity $(sI - A)^{-1}$ is called the *resolvent* of A, and represents the Laplace transform of the state-transition-matrix given in (A.5). Like the state-transition-matrix, the resolvent is *always* invertible, regardless of whether A is invertible.

Setting the initial condition in (A.7) to zero, we obtain the transfer function from the scalar input $u(s)$ to the scalar output $y(s)$:

$$T(s) \coloneqq \frac{y(s)}{u(s)} = C(sI - A)^{-1}B + D. \tag{A.8}$$

We emphasize that the right hand side of (A.8) is in fact a fraction of two polynomials in s.

A transfer function whose numerator has a lesser order than the denominator is called *strictly proper*, whereas one whose numerator and denominator have equal order is termed *bi-proper* or *semi-proper* [81]. A transfer function that is either strictly proper or semi-proper is said to be *proper*. The feed-through matrix D *is zero* when the transfer function is strictly proper, and is non-zero when the transfer function is semi-proper. A state-space realization of a system exists only if its transfer function is proper. That is, no x, A, B, C, and D can be found for a system whose transfer function is not proper. Physical systems are always proper, and are practically always strictly proper.

We define the *impulse response* of the system by

$$h(t) \coloneqq Ce^{At}B + D\delta(t), \tag{A.9}$$

where $\delta(t)$ is the Dirac-delta function. The expression in (A.9) represents the time-domain response of the system to a unit impulse, and is obtained by setting $x(0) = 0$ and $u(t) = \delta(t)$ in (A.3). Notably, the transfer function given in (A.8) is nothing but the Laplace transform of (A.9). *That is, a system's transfer function is simply the Laplace transform of the system's impulse response.*

Lastly, we note an important interpretation of the transfer function for Bounded-Input Bounded-Output (BIBO) stable system[1]. If a BIBO stable system is subjected to a sinusoidal input

$$u(t) = A \sin \omega t,$$

then the *steady-state* response of the system is given by

$$y_\infty(t) = |T(j\omega)| A \sin\big(\omega t + \angle T(j\omega)\big),$$

where $|\cdot|$ and $\angle\cdot$ denote the magnitude and phase operators. That is, when a BIBO stable LTI system is subjected to a persistent sinusoidal input with a given amplitude and frequency, the *steady-state* output is a sinusoidal with the same frequency that is amplified and phase shifted according to the magnitude and phase of $T(s)$ evaluated at $s = j\omega$ (see Figure A.1). For a BIBO stable LIT system, this interpretation is concisely illustrated using the well-known Bode magnitude and phase plots.

A.4 Why Are Transfer Functions Useful?

Linear systems satisfy two important properties:

Homogeneity: This property states that if an input produces an output, then a scaled version of that input produces an equally scaled version of the corresponding output. This is formally stated as:

$$u(t) \to y(t) \;\Rightarrow\; \alpha u(t) \to \alpha y(t).$$

Additivity: This property states that if two distinct inputs produce two distinct outputs, then the sum of these two inputs produces the sum of the corresponding outputs. This is formally stated as:

$$u_1(t) \to y_1(t) \;\;\&\;\; u_2(t) \to y_2(t) \;\Rightarrow\; u_1(t) + u_2(t) \to y_1(t) + y_2(t).$$

[1] A transfer function is BIBO stable if and only if the real components of each of its poles is strictly negative. A system with poles on the $j\omega$-axis is not BIBO stable. BIBO stability is not influenced by the location of the zeros.

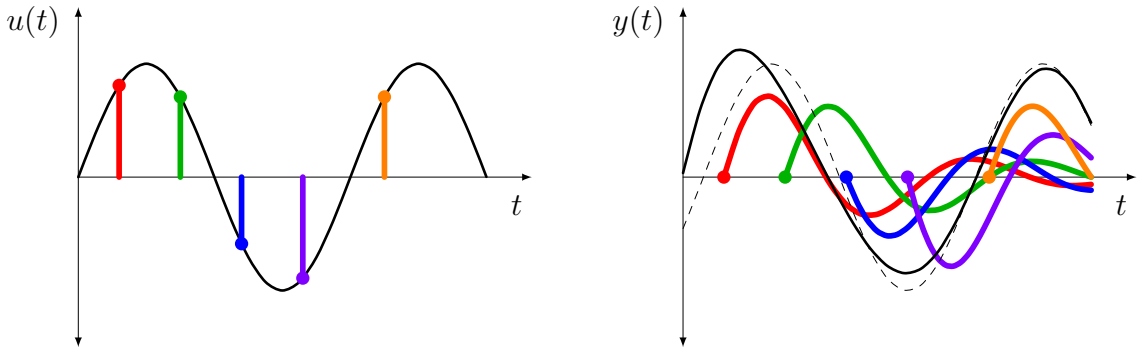


Figure A.1: Superpositioning and convolution. The input signal is shown as the black sinusoid on the left. This signal can be understood as a sequence of impulses of varying magnitudes. The impulse response of the system due to each of the impulses is shown on the right. The resulting output is obtained by summing up all of the scaled and time-shifted impulse responses, and is shown as the solid black sinusoid on the right. The dashed black sinusoid represents the steady-state response of the BIBO stable system.

Together, these two properties form what is known as the *superpositioning principle*. Superpositioning coupled with the time-invariance afforded by LTI systems allows us to characterize the output of the system by considering the effects of the system's initial condition and inputs separately. Moreover, these properties allow us to characterize the effect of a generic input by summing scaled and time-shifted impulse responses of the system.

An example of this is shown in Figure A.1, where a generic input is interpreted as a sequence of scaled impulses initiating accordingly scaled impulse responses, each starting at the appropriate time. This example highlights that the impulse response is, in fact, the "finger print" of an LTI system, and is solely responsible for characterizing the response of the system. The operation shown in Figure A.1 is formally expressed using the well-known *convolution integral*:

$$y(t) = [h * u](t) \coloneqq \int_0^t h(\tau)u(t-\tau)d\tau. \tag{A.10}$$

Convolution plays a central role in classical control theory, since it relates the input, out-

put, and impulse response of every system. However, the convolution integral is prohibitively difficult to work with for all but the simplest of systems. Fortunately, convolution in time-domain (i.e. (A.10)) is equivalent to multiplication in the frequency-domain. Formally, we state this as

$$\mathcal{L}\big\{[x * y](t)\big\} = \mathcal{L}\big\{x(t)\big\} * \mathcal{L}\big\{y(t)\big\} = x(s)\, y(s), \tag{A.11}$$

where $\mathcal{L}\{\cdot\}$ denotes the Laplace transform. *Property* (A.11) *is the key reason for why we use transfer functions and frequency-domain techniques in classical controls.* This single property allows us to replace countless nested convolution integrals with simple multiplication. That is, (A.11) allows us to understand control theory through the lens of algebra (e.g. multiplication of transfer functions) instead of calculus (i.e. convolution integrals). This property underpins the block diagram manipulation and control design techniques taught in introductory control systems course.

A.5 Uniqueness of Transfer Functions

For a given LTI system, the transfer function in (A.8) *is unique*, whereas the states-space representation in (A.4) *is not.* For example, suppose we define a new (different) state variable

$$\hat{x} \coloneqq Px, \tag{A.12}$$

where $P \in \mathbb{R}^{n_x \times n_x}$ is some invertible matrix. Then, we can rewrite (A.1) in terms of $\hat{x}$ as follows:

$$\begin{aligned}\dot{\hat{x}}(t) &= PAP^{-1}\hat{x}(t) + PBu(t),\\ y(t) &= CP^{-1}x(t) + Du(t).\end{aligned}$$

Defining $\hat{A} \coloneqq PAP^{-1}$, $\hat{B} \coloneqq PB$, $\hat{C} \coloneqq CP^{-1}$, and $\hat{D} \coloneqq D$, we obtain a new state-space representation:

$$\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{B}u(t), \tag{A.14a}$$

$$y(t) = \hat{C}\hat{x}(t) + \hat{D}u(t). \tag{A.14b}$$

By applying (A.8) to the state-space representation in (A.14), we obtain:

$$\begin{aligned}
\hat{T}(s) &= \hat{C}(sI - \hat{A})^{-1}\hat{B} + \hat{D} \\
&= CP^{-1}(sI - PAP^{-1})^{-1}PB + D \\
&= CP^{-1}\left[P(sI - A)P^{-1}\right]^{-1}PB + D \\
&= CP^{-1}P(sI - A)^{-1}P^{-1}PB + D \\
&= C(sI - A)^{-1}B + D \\
&= T(s).
\end{aligned}$$

This result demonstrates that the transfer function is invariant to the state vector transformations defined in (A.12)[2]. *Moreover, it highlights that the transfer function of an LTI system is unique, while its state-space representation is not.* It is for this reason that both the transfer function and the impulse response (which are intimately related) can be regarded as the "finger prints" of an LTI system.

Lastly, we observe that the feed-through matrix D did not change between (A.4) and (A.14). This is because, like the transfer function, the D matrix of a system *is always unique.*

A.6 Polar Form

Transfer functions are typically expressed as fractions of polynomials. Any proper transfer function can be expressed in the following form:

$$T(s) = \frac{\alpha_m s^m + \alpha_{m-1}s^{m-1} + \cdots + \alpha_1 s + \alpha_0}{\beta_n s^n + \beta_{n-1}s^{n-1} + \cdots + \beta_1 s + \beta_0}, \tag{A.15}$$

where $\alpha_i, \beta_i \in \mathbb{R}$ are real scalar coefficients, and $n \geq m$. We can rewrite (A.15) in factored form as:

$$T(s) = K\,\frac{\prod^{m}(s - z_i)}{\prod^{n}(s - p_i)}, \tag{A.16}$$

[2]The transfer function can be shown to be invariant to more general state-space transformations than the one assumed in (A.12).

where $K \coloneqq \alpha_m/\beta_n$, and $z_i, p_i \in \mathbb{C}$ are the roots of the numerator and denominator polynomials in (A.15), respectively. The complex numbers z_i and p_i are called the *zeros* and *poles* of the transfer function, respectively. Since each polynomial has real coefficients, complex zeros or poles always appear in complex-conjugate pairs (e.g. a complex zero will always have an accompanying complex zero with a negated imaginary part).

In general, passing a complex number $s \in \mathbb{C}$ through the transfer function results in another complex number $T(s) \in \mathbb{C}$. However, the Cartesian form used in the products of (A.16) is not conducive to understanding the relationship between the input s, the output $T(s)$, the zeros z_i, and the poles p_i. To elucidate this relationship, we leverage the fact that a complex vector $(s - \bar{s}) \in \mathbb{C}$ expressed in Cartesian form can be rewritten in polar form as follows:

$$(s - \bar{s}) = |s - \bar{s}| \exp\left[j\angle(s - \bar{s})\right],$$

where $\exp[j\angle\cdot]$ denotes Euler's Identity. Applying this to (A.16), we obtain:

$$T(s) = K \frac{\prod^{m} |s - z_i| e^{j\angle(s - z_i)}}{\prod^{n} |s - p_i| e^{j\angle(s - p_i)}}.$$

Rearranging, we obtain the polar form of the transfer function in (A.15):

$$T(s) = K \frac{\prod^{m} |s - z_i|}{\prod^{n} |s - p_i|} \cdot \exp\left[\sum^{m} j\angle(s - z_i) - \sum^{n} j\angle(s - p_i)\right]. \tag{A.17}$$

Since the magnitude of Euler's identity is unity for any argument, the magnitude of (A.17) is entirely determined by the product of the magnitudes on the left. Furthermore, since the phase of a positive real number is always zero, the phase of (A.17) is entirely determined by the phase arguments inside of $\exp[\cdot]$ on the right. Hence, the expression in (A.17) provides a very intuitive geometric interpretation of the relation between s, $T(s)$, z_i, and p_i. Given, s, z_i, and p_i, the magnitude of $T(s)$ is obtained by multiplying K times the length of each $(s - z_i)$, and dividing by the length of each $(s - p_i)$. Likewise, the phase of $T(s)$ can be determined simply by adding the phase angle of each vector $(s - z_i)$ and subtracting the phase

angle of each vector $(s - p_i)$. *This geometric interpretation is instrumental in understanding how to draw Bode, Nyquist, and Nichols plots.*