Successive Convexification of Non-convex Optimal Control Problems: Theory and Applications

Yuanqi Mao

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

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Behçet Açıkmeşe, Chair

Mehran Mesbahi

Marco Salviato

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

University of Washington

Abstract

Successive Convexification of Non-convex Optimal Control Problems: Theory and Applications

Yuanqi Mao

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

The topic of this dissertation centers around *Successive Convexification*, a family of iterative algorithms designed to solve non-convex constrained optimal control problems. This document begins with an introduction to optimal control and finite-dimensional optimization in Chapter 2. It then presents the main algorithm within the *Successive Convexification* framework, `SCvx`, in Chapter 3. `SCvx` is a general-purpose solver that can handle problems with nonlinear system dynamics and non-convex state and control constraints. Analytical and numerical results are presented to demonstrate its convergence properties, including global convergence, strong convergence and superlinear convergence rate. `SCvx-fast`, a specialized version of `SCvx` is introduced next in Chapter 4 to handle systems with simpler dynamics and convex keep-out zones type of constraints commonly seen in quadrotor obstacle avoidance problems. It has new features such as a *project-and-convexify* step, removes the smoothness assumption, and does not rely on the trust-region updating mechanism. As a result, more aggressive steps can be taken and thus convergence occurs in much fewer iterations. With `SCvx` or `SCvx-fast` as the central pillar for on-board trajectory planning, we can build a fully autonomous system by further integrating i) a computer-vision-based perception unit and ii) Signal-Temporal-Logic (STL)-based mission specifications. Chapter 5 explores these directions as aerospace applications of *Successive Convexification*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

SCvx: Successive Convexification

K-L: Kurdyka-Łojasiewicz

SLP: Successive Linear Programming

SQP: Sequential Quadratic Programming

IPM: Interior Point Method

STL: Signal Temporal Logic

MPC: Model Predictive Control

DOF: Degree of Freedom

SOCP: Second-Order Cone Programming

MILP: Mixed-Integer Linear Programming

PMP: Pontryagin's Maximum Principle

LP: Linear Programming

QP: Quadratic Programming

SDP: Semidefinite Programming

BFGS: Broyden-Fletcher-Goldfarb-Shanno

KKT: Karush-Kuhn-Tucker

CNN: Convolutional Neural Network

UAV: Unmanned Aerial Vehicles

ACKNOWLEDGMENTS

This Ph.D has been a turbulent but rewarding journey for me. It certainly has ups and downs, but all will prove to be invaluable experience in the end. I could not make it to the finish line without the help, support and inspiration from all the people along the way.

I have to begin by thanking my advisor, Behçet Açıkmeşe. He led me into the world of optimal control and convex optimization, and I benefited immensely from his broad knowledge base, mathematical insight and thought-provoking research ideas. It is the research and professional access provided by Behçet made me who I am today, and it is also his support and patience got me through the hard times, and for that I shall be eternally grateful.

I would like to thank Professors Mehran Mesbahi, Marco Salviato, Krithika Manohar from UW and Pierre-Loic Garoche from ENAC, France for serving on my General Exam and Final Exam committees. Your valuable input and continuous guidance are what make this dissertation comes into fruition. A special thanks goes to Pierre-Loic Garoche for being like my second advisor for the past year, not to mention his generous support for my time here in France. I'm also grateful for the cooperation and friendship of my fellow students from ACL, Michael Szmuk, Daniel Dueri, Yue Yu, Utku Eren, Danylo Malyuta and Sarah Li, who have all inspired and motivated me in my research. Michael in particular, thank you for your tremendous help with the development of *Successive Convexification*.

I would like to thank University of Washington and particularly the staff of the Aeronautics and Astronautics Department. You helped me get through unexpected situations and provided me with the comfort of knowing someone will always have my back, which is extremely important to me in difficult times. I would also like to thank grants from the Air Force Research Laboratory, Office of Naval Research, and National Science Foundation for

funding and facilitating my research.

I would like to thank my manager Andrew Sparks and my teammates Brigid Blakeslee, Jerry Ding, Edward Tunstel, and Julian Ryde at United Technologies Research Center for all of their support and patience throughout my internship there. The project we worked on is indeed challenging, but with all the helpful discussions we had, I feel like in the end we did acheive something meaningful, and that makes me proud. It also motivated my academic research later on, for which I am grateful as well.

I would like to thank Professors Efstathios Bakolas and Maruthi Akella from the University of Texas at Austin for introducing me to the world of optimal control and linear systems, and preparing me for the Qualifying Exam. Almost my entire doctoral research is based on these fundamental subjects, which in hindsight might just be the most useful things I learned during the early days.

I would like to thank my undergraduate advisor, Professor XU Jinsong from Shanghai Jiao Tong University for piquing my interest in control and optimization for dynamical systems. Subsequently, his meticulous guidance and high standard in my undergraduate thesis directly led me into the realm of serious scientific research. I would also like to thanks him and all the other colleagues in Seastel for the one year we spent together. Not only I learned a lot industry-wise, but also it was just a fun and memorable experience. It was still small at the time, but to me it feels like a family.

To my undergraduate roommates ZHANG Chenliang, WANG Jianqiang and ZHANG Hanqin, I consider myself lucky to have met you early in my adulthood. It seems like only yesterday when we queued up in the cafeteria, when we went on our graduation trip, and when we had our farewell barbecue. Not just fond memories though, your friendship truly made me a better person and is still motivating me today even when we are oceans apart. I also want to express my deepest appreciation to other long-time friends from undergraduate/high school years, HE Jiayi, QIN Hao, QIN Mian, XIE Yi, CHEN Jie, ZHANG Hongsheng, JIANG

Hanchen, LI Yang, ZHAO Yanjun, CHEN Yangyang, XU Yu, LIU Junyang etc. as well.

I want to thank my parents for all the sacrifices they made to raise me as a child, educate me as a teenager, accommodate me as a young adult, and still support me from thousands of miles away. During the unintended time I was in China, it was their words that kept me together and kept me going. I have not done nearly enough for them compared with what they have done for me, and probably never will, but I would definite try. Mom and dad, I would definitely try.

Finally, to my wife Yirou, it is impossible for me to be writing this paragraph in my dissertation today without your love and support. From high school to college, from Shanghai to Seattle, we were there together. I cannot imagine a different twenties with you absent, let alone finishing this Ph.D. I want to thank you for being there when I needed the most, thank you for sharing the moments I cherish the most, and thank you for just being my best friend. Even though we are apart now, but soon we will walk together along an unnamed river bank, side by side again.

DEDICATION

to my dear wife, Yirou

Chapter 1
INTRODUCTION

1.1 Background

With the recent release of *NASA Space Technology Roadmaps and Priorities*, next-generation Guidance, Navigation, and Control (GN&C) technologies will be of paramount importance for future planetary explorations [67]. Spacecraft GN&C technologies involves complex sciences and have been evolving since the launch of the first rocket. Simply put, *guidance* is the onboard generation of desired trajectories, which is often synonymous with *trajectory optimization*. *Navigation* is to determine the current states (position, velocity, and attitude) of the vehicles and the environment. *Control* is to track guidance commands while maintaining stability, by the manipulation of vehicle steering. The main focus of this dissertation is on optimization-based self-guidance technologies, but in order to build a fully autonomous system, we will also touch upon vision-based navigation and mission level specifications, and how they can interact with guidance.

Guidance and control technologies have progressed throughout aerospace history. To meet the performance standards of future applications, recent years have seen the advent of new technologies to enable and enhance a wide range of capabilities. For example, autonomous precision rocket landing is one such capability that is key to enabling the reuse of rockets[11]. Convex optimization and convexification based technologies have been developed for the 3-Degree of Freedom (DoF) cases [5, 4], while 6-DoF formulations with attitude control have been studied in recent years [41, 74, 73]. Lately we have also seen commercial usages of these technologies on, for instance, SpaceX's Falcon 9 reusable rockets [10].

Autonomous rendezvous and docking is another key enabling technology that presents many challenges. Guidance and control technologies have been developed, for instance,

via conic optimization [47], and via Model Predictive Control (MPC) [81]. Furthermore, the complexity and safety of such operations can be enhanced by considering the attitude control problem. Using a quaternion-based formulation, non-convex quadratic constraints can be convexified, and the problem can be solved via semidefinite programming [38]. Other techniques may also be applicable, such as the special MPC design proposed in [37].

As a final example, consider quadrotors, systems that are becoming increasingly ubiquitous in today's world. Obstacle avoidance is one key challenge faced by these systems [62]. To address this issue, a sampling based real-time guidance framework was proposed recently in [7], within which simplified quadrotor dynamics were used. For robust real-time embedded applications, a convexification-based real-time guidance for quadrotors was proposed in [72], which includes non-convex obstacle avoidance constraints.

These examples represent a clear shift in paradigm when dealing with optimal control problems. Thanks to the advancement of digital computing, we have seen guidance technologies evolve from offline computation of analytical trajectories in the Apollo era to now real-time onboard computation driven by optimization algorithms. Similar paradigm shift is seen in control technologies as well [78]. However, onboard computing resources are always limited and thus precious, which poses a major challenge on how to design a guidance algorithm as efficient as possible with complex system dynamics and constraints.

Efficiency is where traditional nonlinear programming methods fall short. When applied to complex optimal control problems, optimization algorithms like the Sequential Quadratic Programming (SQP) [48, 15, 28] and the general purpose Interior Point Method (IPM) [83, 80] often suffer from unpredictable convergence behaviors that are highly dependent on user-supplied initial guesses. Additionally, they offer few bounds, if any, on the computational effort required to achieve convergence. As a result, these techniques are typically not applicable for real-time autonomous operations. In the next section, we will see some related works on how convex optimization and convexification of non-convex problems can tackle both issues, and thus play a major role in designing these efficient algorithms.

1.2 Prior Work

In this section we will focus on convex optimization [14] based guidance technologies. Thanks to recent advances, the solutions to a large class of convex optimization problems can be computed in polynomial time [57] using either generic Second-Order Cone Programming (SOCP) solvers [20], or customized solvers that take advantage of problem-specific structures [56, 24]. Moreover, if a feasible solution exists, these algorithms are guaranteed to find the global optimal solution in a bounded number of iterations. Conversely, they provide a certificate of infeasibility if no feasible solution exists.

However, most real-world guidance problems are inherently non-convex. The performance and guarantees of convex optimization algorithms are the primary reasons researchers are motivated to formulate non-convex guidance problems in a convex framework. This practice is referred to as *convexification*. In addition to the applications we mentioned above, convexification based approaches have also been applied to, for example, swarm formation flying [6] and planetary entry [44]. While the results of these numerical experiments look promising, few theoretical proofs are provided. Inspired by the powered descent guidance for planetary landing problem [5], recent results have introduced a procedure known as *lossless convexification* to handle a large class of non-convex control constraints [3, 35]. It proves that these non-convex optimal control problems can be posed as equivalent convex optimization problems without sacrificing feasibility or optimality.

1.3 The Author's Work

More recently, an iterative procedure known as *successive convexification* [51, 52, 54] solves problems with nonlinear dynamics and non-convex state and control constraints, and provides proofs of global (weak and strong) convergence and superlinear rate of convergence. Along with *lossless convexification*, these two convexification techniques are among the first rigorous attempts to solve complex non-convex optimal control problems in real-time. The latter in partucular, is the main work of the author's Ph.D research, and thus the main

topic of this dissertation (in Chapter 3 and 4). The author's work in these area also includes inter-sample obstacle avoidance [22], stochastic motion planning with probabilistic occupancy functions [79], and `SCvx`-based MPC with aerospace applications [55, 53].

To build a fully autonomous system centered around *Successive Convexification*, first one needs to further complete its theoretical framework. A strong convergence result of the `SCvx` algorithm for continuous-time systems is being prepared, as an extension to [54]. Several enhancements have been made to the `SCvx-fast` algorithm as well, hence extending [52]. They include a novel initialization procedure that gets rid of the requirement of feasible start, revised theory that can now deal with non-smooth obstacles, and additional proofs of superlinear convergence rate. These new results will be presented in Chapter 4 as well.

In terms of practical applications, one would need a navigation module that takes account of the unknown environment, since the sole focus of *Successive Convexification* is on the guidance aspect of the GN&C problem. So far, we have assumed that the problem setup is given to us. For example, for quadrotors where the obstacles are and how they look like are known to us beforehand. That however, may not be realistic for vehicles in an unknown environment. In that case, we have to rely on data provided by onboard sensors, then the problem becomes how to bridge the gap between raw sensory data and the information required by the guidance algorithm (in this case, `SCvx` or `SCvx-fast`). The author has been working on a computer-vision-based solution that uses machine learning to produce convex-shaped obstacles from the occupancy grid, which can then be directly used by the `SCvx-fast` algorithm. Preliminary results are presented in Chapter 5.

Last but not least, guidance as part of an entire mission, it can incorporate some of the mission specifications as its objectives or constraints. Since *Successive Convexification* deals with continuous-time dynamic systems, Signal Temporal Logic (STL) specifications fit perfectly given its continuous-time nature. Robustness measures of STL specifications is introduced in [21] to facilitate its usage in satisfaction or falsification of cyber-phisical systems. STL as part of the MPC design has been studied in [61], in which the combinatorial nature of such specifications is preserved, and thus Mixed-Integer Linear Programming (MILP) has

to be employed. More recently, [58] relaxes the robustness measures using smooth approximations so that the resulting problem is suitable for a smooth optimization solver. After examining the possibility of convexification of such specifications and the usage of `SCvx` as the solution method, the author proposed a four-steps `SCvx-STL` procedure that uses a novel STL subdynamics formulation. Numerical results are presented in Chapter 5 for spacecraft rendezvous and quadrotor motion planning problems.

With a solid theoretical foundation and promising real-world use-cases, The author firmly believes that we can and will make a step further toward a *Successive Convexification* based fully autonomous system.

Chapter 2

OPTIMAL CONTROL AND FINITE-DIMENSIONAL OPTIMIZATION

The purpose of this chapter is primarily to familiarize the reader with the nomenclature and mathematical preliminaries of the underlying theory, namely optimal control and finite-dimensional optimization. The author also wants to make some connections between these theories and what we have used in our own convergence proofs.

2.1 Optimal Control Theory

This section primarily serves as an introduction to the optimal control theory, with a slight shift of focus on the geometric aspect of the said theory because the geometric insight leads to a clearer picture of how optimal control really works, how it is closely related to convex analysis, and how the classical results (calculus of variations and Pontryagin's Maximum Principle [59]) can be easily extended to including a more diverse set of functions, terminal constraints and state constraints. In fact, the interplay among control, geometry, and physics goes back to ancient times. However, the pre-Maximum Principle relationship is primarily one-sided, with optimal control, or more precisely the calculus of variations, exercising a decisive and revolutionary influence on geometry, such as the birth of differential geometry. In the much shorter post-Maximum Principle era though, the influence goes mainly in the other direction, with control theory being heavily influenced by geometric ideas.

2.1.1 The "Brachystochrone Problem" and Calculus of Variations

In 1696, Johann Bernoulli challenged the mathematicians of his time to solve the "brachystochrone problem": among all curves ξ going from a point A to a point B in a vertical $x-y$

plane, determine the one for which a particle falling freely along ξ – subject only to the action of the gravitational force plus whatever "virtual force" is needed to keep the particle on ξ – will reach B from A in minimum time. Mathematically, one can formulate this by observing that, if our curve ξ is given—in terms of a parameter s which is an increasing function of time t—by $s \to (x(s), y(s))$, $0 \le s \le 1$, $A = (a, \alpha)$, $B = (b, \beta)$, and t is time, initialized so that $t(0) = 0$, then the total energy

$$E = \frac{1}{2}\left(\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2\right) + gy$$

is constant (taking the mass to be 1). So

$$dt = \sqrt{\frac{dx^2 + dy^2}{2E - 2gy}}\,.$$

If we assume that $a < b$, and that our minimizing curve is the graph of a function $x \to y(x)$, then $dt = \sqrt{\frac{1+y'(x)^2}{2E-2gy}}\,dx$, so our problem becomes that of minimizing the integral $\int_a^b L(y(x), y'(x))\,dx$, subject to the constraints $y(a) = \alpha$, $y(b) = \beta$, where

$$L(y, u) \coloneqq \sqrt{\frac{1 + u^2}{2E - 2gy}}\,. \tag{2.1}$$

On the other hand, without assuming that $a < b$ and without making any assumption about y being a function of x, we can formulate our problem as that of minimizing the integral

$$I = \int_0^1 \Lambda(x(s), y(s), \dot{x}(s), \dot{y}(s))\,ds\,, \tag{2.2}$$

subject to $x(0) = a$, $y(0) = \alpha$, $x(1) = b$, $y(1) = \beta$, where

$$\Lambda(x, y, u, v) \coloneqq \sqrt{\frac{u^2 + v^2}{2E - 2gy}}\,. \tag{2.3}$$

(The curves under consideration are, of course, restricted to lie in the closed half-plane $H_E = \{(x, y) : y \leq \frac{E}{g}\}$.) Finally, we can also reformulate it as a *minimum-time optimal control* problem, by writing the equations of motion (after choosing units so that $2g = 1$, and changing y to $2E - y$) as

$$\dot{x} = u\sqrt{y}\,, \;\; \dot{y} = v\sqrt{y}\,, \;\; u^2 + v^2 = 1\,, \tag{2.4}$$

where x and y are the state variables, u and v are the controls, and y is restricted to satisfy $y \geq 0$. This third formulation is clearly more natural than the first two, since it is a direct mathematical description of the problem. This is the main reasons that people consider Johann Bernoulli's question properly belongs to optimal control.

What makes the solution of the "brachystochrone" problem important for people hundreds years later, is that it marked the beginning of the systematic study – which would be given the name "calculus of variations" by Euler – of minimization problems in the form

Problem 2.1.1. Calculus of Variations Problem

$$\min \quad \int_a^b L(\xi(t), \dot{\xi}(t), t)dt$$

subject to:

$$\xi(a) = \bar{q}\,, \qquad \xi(b) = \hat{q}\,.$$

The "brachystochrone" problem happens to have a very significant differential-geometric aspect. In fact, (2.4) describes the curves parameterized by arc-length for a Riemannian metric on the upper half-plane, given by

$$ds^2 = \frac{dx^2 + dy^2}{y}\,. \tag{2.6}$$

The supposed cycloids that solve the problem are the geodesics of this metric. Johann Bernoulli studied the same problem with other functions instead of $\sqrt{y}$ in (2.4), and in

particular for the function y, which in modern terms corresponds to the metric

$$ds^2 = \frac{dx^2 + dy^2}{y^2}\,. \tag{2.7}$$

In this case, he found that the minimum-time paths were half-circles rather than cycloids. So Johann Bernoulli had almost found what we would call today the *Poincaré half-plane*, i.e., the simplest model of a non-Euclidean (hyperbolic) geometry. All it takes to make the connection is to call the travel time "length", i.e. of thinking that an expression such as (2.6) or (2.7) could serve as just another way of measuring length.

2.1.2 The Euler-Lagrange equation and the Hamiltonian

Nevertheless, Euler and Lagrange further develops the necessary condition for a curve $[a, b] \ni t \to \xi_*(t) \in \mathbb{R}^n$ to be a solution of Problem 2.1.1 is that it satisfy the *Euler-Lagrange equation*

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = \frac{\partial L}{\partial q}\,, \tag{2.8}$$

i.e., if we use u instead of $\dot{q}$:

$$\frac{d}{dt}\frac{\partial L}{\partial u}\Big(\xi_*(t), \dot{\xi}_*(t), t\Big) = \frac{\partial L}{\partial q}\Big(\xi_*(t), \dot{\xi}_*(t), t\Big)\,, \tag{2.9}$$

or

$$\frac{d}{dt}\frac{\partial L}{\partial u^i}\Big(\xi_*(t), \dot{\xi}_*(t), t\Big) = \frac{\partial L}{\partial q^i}\Big(\xi_*(t), \dot{\xi}_*(t), t\Big)\,, \quad i = 1, \ldots, n\,. \tag{2.10}$$

In 1830s, W. R. Hamilton published his results on dynamics showing how to rewrite the Euler-Lagrange equations in what we would now call "Hamiltonian form": for a curve

$t \mapsto \xi_*(t)$, (2.8) is exactly equivalent to the system of equations

$$\begin{aligned} \tfrac{d\xi_*}{dt}(t) &= \tfrac{\partial H}{\partial p}\Big(\xi_*(t), \dot{\xi}_*(t), \psi(t), t\Big)\,, \\ \tfrac{d\psi}{dt}(t) &= -\tfrac{\partial H}{\partial q}\Big(\xi_*(t), \dot{\xi}_*(t), \psi(t), t\Big)\,, \\ 0 &= \tfrac{\partial H}{\partial u}\Big(\xi_*(t), \dot{\xi}_*(t), \psi(t), t\Big)\,, \end{aligned} \tag{2.11}$$

where the *Hamiltonian* H and the *momentum* ψ are defined by

$$H(q,u,p,t) \coloneqq \langle p, u\rangle - L(q,u,t)\,, \tag{2.12}$$

$$\psi(t) \coloneqq \frac{\partial L}{\partial u}\Big(\xi_*(t), \dot{\xi}_*(t), t\Big)\,. \tag{2.13}$$

The system (2.11)-(2.13) is not exactly what is commonly known as "Hamilton's equations," although in the author's view it is how Hamilton's equations should be written. The more commonly seen Hamilton equations are

$$\frac{dq}{dt} = \frac{\partial \mathcal{H}}{\partial p}\,, \quad \frac{dp}{dt} = -\frac{\partial \mathcal{H}}{\partial q}\,, \tag{2.14}$$

where $(q,p,t) \to \mathcal{H}(q,p,t)$ is a function of p, q and t alone, defined by the formula

$$\mathcal{H}(q,p,t) \coloneqq \langle p, \dot{q}\rangle - L(q,\dot{q},t)\,. \tag{2.15}$$

This is certainly similar to (2.11)-(2.13), but with a crucial difference: in (2.14) and (2.15) $\dot{q}$ is supposed to be treated *not* as an independent variable, but as a function of q, p, and t, defined implicitly by the equation

$$p = \frac{\partial L}{\partial \dot{q}}(q,\dot{q},t)\,, \tag{2.16}$$

whereas in (2.11)-(2.13) q, u, p, and t are treated as independent variables.

2.1.3 Nonlinear control and Lie Brackets

For an autonomous control system $\Sigma : \dot{q} = f^{\Sigma}(q, u)$, $u \in U$, on a smooth manifold Q^{Σ}, define Λ^{Σ} to be the free Lie algebra generated by a family $\{F_u : u \in U\}$ of indeterminates indexed by the set U. Let L^{Σ} be the Lie algebra of vector fields generated by the f_u^{Σ}, $u \in U$, where we let $f_u^{\Sigma}(q) = f^{\Sigma}(q, u)$. Then there is a unique Lie-algebra homomorphism $P^{\Sigma} : \Lambda^{\Sigma} \to L^{\Sigma}$ that sends F_u to f_u^{Σ} for each u. Given a point $q \in Q^{\Sigma}$, by setting $EV_q^{\Sigma}(V) = P^{\Sigma}(V)(q)$, we can define a linear map $EV_q^{\Sigma} : \Lambda^{\Sigma} \to T_q Q^{\Sigma}$, where $T_q Q^{\Sigma}$ is the tangent space of Q^{Σ} at q.

Definition 2.1.1. The kernel

$$REL_q^{\Sigma} \coloneqq \ker\left(EV_q^{\Sigma}\right)$$

is the *set of Lie-bracket relations at q* of the vector fields of Σ.

As an example, suppose u, v, and w are three members of U. Then the formal expression $[F_u, [F_v, F_u]] + 4[F_u, F_w] + F_v$ belongs to REL_q^{Σ} if and only if $[f_u, [f_v, f_u]](q) + 4[f_u, f_w](q) + f_v(q) = 0$. The following theorem can be proved by a simple application of E. Cartan's graph method (see [68, Theorem 7] for details).

Theorem 2.1.1. *For a fixed control set U, if we consider pairs (Σ, q) such that Σ is a real analytic system with control set U, q belongs to the state space Q^{Σ} of Σ, and Σ has the accessibility property at q, then the space REL_q^{Σ} is a complete set of local invariants for the germ of Σ at q under diffeomorphisms. That is, given two pairs (Σ_i, q_i), $i = 1, 2$, the following two conditions are equivalent:*

(i) There exists a diffeomorphism Φ from a neighborhood U_1 of q_1 in Q^{Σ_1} onto a neighborhood U_2 of q_2 in Q^{Σ_2} that maps each vector field $f_u^{\Sigma_1}$ to the corresponding vector field $f_u^{\Sigma_2}$ and is such that $\Phi(q_1) = q_2$,

(ii) $REL_{q_1}^{\Sigma_1} = REL_{q_2}^{\Sigma_2}$.

Note that diffeomorphism is just an isomorphism of smooth manifolds, meaning it invertibly maps one differentiable manifold to another such that both the function and its inverse are smooth. Theorem 2.1.1 tells us that–for real-analytic systems–all properties of interest that are invariant under nonlinear coordinate changes are determined by the Lie bracket relations. For systems where U is a linear space and the control enters the equations in an affine way, it can be shown the class of *linear systems* is characterized by the vanishing of certain Lie brackets.

Since the Lie bracket relations give a complete set of invariants under diffeomorphism, it is not unreasonable to expect that the structural, coordinate invariant properties of a nonlinear system should be expressible in terms of Lie brackets. For example, the structure of the time-optimal trajectories (e. g., whether the time-optimal controls are bang-bang) should be determined by Lie bracket conditions. Since the main tool for analyzing this structure is the necessary condition for optimality given by the Pontryagin's Maximum Principle, it is obviously useful to have a statement of the Maximum Principle where the Lie brackets appear explicitly. We will describe such a formulation next and then show by means of a simple example how this formulation can be used to derive qualitative properties of optimal trajectories.

2.1.4 Pontryagin's Maximum Principle

The Pontryagin's Maximum Principle [9, 59] extends the necessary conditions for optimality of the calculus of variations to the much more general setting of fixed-endpoint optimal control problems with a state space $Q \subset \mathbb{R}^n$, and a set U for control variables:

Problem 2.1.2. Fixed-endpoint Optimal Control Problems (FEOCP)

$$\min \quad \int_a^b L(\xi(t), \dot{\xi}(t), t)dt$$

subject to:

$$\begin{aligned} &\dot{\xi}(t) = f(\xi(t), \eta(t), t), && \text{for a.e. } t \\ &(\xi(t), \eta(t)) \in Q \times U, && \text{for all } t \\ &\xi(a) = \bar{q}, \quad \xi(b) = \hat{q}, \end{aligned}$$

or to the following problem that has a more general endpoint condition:

Problem 2.1.3. General-endpoint Optimal Control Problems (GEOCP)

$$\min \quad \int_a^b L(\xi(t), \dot{\xi}(t), t)dt$$

subject to:

$$\begin{aligned} &\dot{\xi}(t) = f(\xi(t), \eta(t), t), && \text{for a.e. } t \\ &(\xi(t), \eta(t)) \in Q \times U, && \text{for all } t \\ &(\xi(b), \xi(a)) \in S. \end{aligned}$$

Pontryagin's result is summarized as follows.

Theorem 2.1.2 (Pontryagin's Maximum Principle (PMP)). *For a trajectory-control pair* $[a, b] \ni t \to (\xi_*(t), \eta_*(t))$ *to be a minimizer, it is necessary that there exist an absolutely continuous curve* $[a, b] \ni t \to \psi(t) \in \mathbb{R}_n$ *and a constant* $\psi_0 \geq 0$ *such that the adjoint equation*

$$-\dot{\psi}(t) = \frac{\partial H}{\partial q}(\xi_*(t), \eta_*(t), \psi(t), \psi_0, t)$$

and the Hamiltonian maximization condition

$$H(\xi_*(t), \eta_*(t), \psi(t), \psi_0, t) = \max\{H(\xi_*(t), u, \psi(t), \psi_0, t) : u \in U\}$$

hold for a.e. $t \in [a, b]$*, as well as the nontriviality condition*

$$(\psi(t), \psi_0) \neq (0, 0) \quad \text{for some (and hence every)} \quad t\,,$$

and the transversality condition

$$(-\psi(b), \psi(a)) \in C^\dagger\,.$$

Here

- H is the *Hamiltonian*, defined by

$$H(q, u, p, p_0, t) = p \cdot f(q, u, t) - p_0 L(q, u, t).$$

- C is a *tangent cone* at the point $(\xi_*(b), \xi_*(a))$ to the set S.

- If C is any subset of a real linear space X, then $C^\dagger$ is the *polar cone* of C, i.e. the set

$$C^\dagger \;:=\; \{v \in X^* : \langle v, w\rangle \leq 0 \;\text{ for all }\; w \in C\}\,,$$

  where X^* is the dual space of X.

- The symbol $\mathbb{R}_n$ denotes the set of n-dimensional real *row vectors*, whereas the members of $\mathbb{R}^n$ are *column vectors*.

2.1.5 The Maximum Principle and Set Separation

It turns out that the maximum principle, as formulated, is really a "geometric" result, giving a necessary condition for a separation property between a reachable set of a control system and some other set.

To see this, we first observe that the Maximum Principle for problems of the form (GEOCP) can easily be reduced to the special case of a problem of the same form but with a more restrictive endpoint condition:

$$\xi(a) = \bar{q}\,, \qquad \xi(b) \in S\,, \tag{2.19}$$

where S is a given subset of Q. Indeed, suppose we are given a problem of type Problem 2.1.3 (GEOCP), and an optimal trajectory-control pair (ξ_*, η_*), consider a new problem

Problem 2.1.4. Terminal Set Optimal Control Problems (TSOCP)

$$\min \quad \int_{a-1}^{b} L_{\text{new}}(\xi_{\text{new}}(t), \dot{\xi}_{\text{new}}(t), t)dt$$

subject to:

$$\begin{aligned}
&\dot{\xi}_{\text{new}}(t) = f_{\text{new}}(\xi_{\text{new}}(t), \eta_{\text{new}}(t), t), && \text{for a.e. } t\\
&(\xi_{\text{new}}(t), \eta_{\text{new}}(t)) \in Q_{\text{new}} \times U_{\text{new}}, && \text{for all } t\\
&\xi_{\text{new}}(a-1) = \bar{q}_{\text{new}} \quad \xi_{\text{new}}(b) \in S_{\text{new}},
\end{aligned}$$

where $Q_{\text{new}} = Q \times Q$, $U_{\text{new}} = U \times \mathbb{R}^n$, $S_{\text{new}} = S$, $\bar{q}_{\text{new}} = (\xi_*(a), \xi_*(a))$, and, for $q^1 \in Q$, $q^2 \in Q$, $u \in U$, $v \in \mathbb{R}^n$, $a - 1 \leq t \leq b$, we let

$$\begin{aligned}
L_{\text{new}}(q^1, q^2, u, v, t) &= \chi_{[a,b]}(t) L(q^1, u, t)\,,\\
f_{\text{new}}(q^1, q^2, u, v, t) &= \chi_{[a,b]}(t).\Big(f(q^1, u, t), 0\Big) + (1 - \chi_{[a,b]}(t)).(v, v)\,,
\end{aligned}$$

where, if E is a set, then $\chi_{_E}$ denotes the indicator function of E, i. e., $\chi_{_E}(x) = 1$ if $x \in E$ and $\chi_{_E}(x) = 0$ if $x \notin E$.

It is then readily shown that the new reference trajectory-control pair, consisting of the curve $[a, b] \ni t \to \xi_{*,\text{new}}(t) \coloneqq (\xi_*(t), \xi_*(a))$, and the corresponding control $[a, b] \ni t \to \eta_{*,\text{new}}(t) \coloneqq (\eta_*(t), 0)$ (extended to the interval $[a - 1, b]$ by letting $\xi_{*,\text{new}}(t) = (\xi_*(a), \xi_*(a))$

and $\eta_{*,\text{new}}(t) = (0,0)$ for $a-1 \leq t < a$) is optimal for Problem 2.1.4 (TSOCP). Applying to TSOCP the conclusion of the Maximum Principle for problems of this special form, we get precisely the conclusion of for the general one (GEOCP).

The quest to find the necessary conditions for optimality of Problem 2.1.4 (TSOCP) can be readily reduced as a geometric problem about separation of sets.

Definition 2.1.2. We say that two subsets $\mathcal{R}$ and S of a topological space Q are *separated* at a point $\hat{q} \in Q$ if $\mathcal{R} \cap S = \{\hat{q}\}$, and that $\mathcal{R}$ and S are *locally separated* at $\hat{q}$ if there exists a neighborhood V of $\hat{q}$ such that $\mathcal{R} \cap S \cap V = \{\hat{q}\}$.

For a control system Σ with dynamical law

$$\dot{q} = f(q,u,t)\,,\ \ q \in Q\,,\ \ u \in U\,,\ \ t \in I\,, \tag{2.21}$$

where I is a subinterval of $\mathbb{R}$ (and for the time being Q is an open subset of $\mathbb{R}^n$). Define $\mathcal{R}^{\Sigma}_{[a,b]}(\bar{q})$ – the Σ-*reachable set from* $\bar{q}$ *over* $[a,b]$ – to be the set of all points $\xi(b)$, for all possible trajectory-control pairs $\gamma = (\xi,\eta)$ of Σ such that ξ is defined on $[a,b]$ and $\xi(a) = \bar{q}$. Also, define H^{Σ} – the *Hamiltonian* of Σ – to be the function

$$Q \times U \times \mathbb{R}^n \times I \ni (q,u,p,t) \to H^{\Sigma}(q,u,p,t) = \langle p, f(q,u,t)\rangle\,. \tag{2.22}$$

Then it is not hard to show that the Maximum Principle follows from a necessary condition for a reachable set $\mathcal{R}^{\Sigma}_{[a,b]}(\bar{q})$ to be separated from the terminal set S.

Theorem 2.1.3 (**Set Separation Maximum Principle (SSMP)**). *Let $\Sigma = (Q,U,I,f)$ be a control system in $\mathbb{R}^n$. Assume that $S \subseteq Q$, $a \leq b$, $a,b \in I$, $\xi_* : [a,b] \to Q$ is a trajectory of Σ corresponding to a control $\eta_* : [a,b] \to U$, and $\xi_*(b) \in S$. Let C be a tangent cone to S at $\xi_*(b)$ which is not a linear subspace of $\mathbb{R}^n$. Then a necessary condition for $\mathcal{R}^{\Sigma}_{[a,b]}(\xi_*(a))$ to be locally separated from S at $\xi_*(b)$ is that there exist a nonzero row-vector-valued map $[a,b] \ni t \to \psi(t) \in \mathbb{R}_n$ that satisfies:*

1. the **Adjoint Equation (AE)**

$$\dot{\psi}(t) = -\frac{\partial H^{\Sigma}}{\partial q}\Big(\xi_*(t), \eta_*(t), \psi(t), t\Big),$$

2. the **Hamiltonian Maximization Condition (HMC)**

$$H^{\Sigma}(\xi_*(t), \eta_*(t), \psi(t), t) = \max\Big\{H^{\Sigma}(\xi_*(t), u, \psi(t), t) : u \in U\Big\},$$

3. the **Transversality Condition (TC)**

$$-\psi(b) \in C^{\dagger}\,.$$

To see the equivalence, consider the standard optimal control problem, Problem 2.1.3 (GEOCP) with the special endpoint condition (2.19). Suppose (ξ_*, η_*) is a solution. Form the "augmented system" $\Sigma^{\#} = (Q^{\#}, U, I, f^{\#})$, where

(A.1) $Q^{\#} = \mathbb{R} \times Q$,

(A.2) $f^{\#} : Q^{\#} \times U \times I \to \mathbb{R}^{n+1}$ is the map $(r, q, t, u) \to (-L(q, u, t), f(q, u, t))\,,$

(A.3) $S^{\#} \coloneqq \Big\{(c, q) : q \in S\,,\, c \geq \hat{c} + \|q - \hat{q}\|^2\Big\}$,

(A.4) $c_*(t) = -\int_a^t L(\xi_*(s), \eta_*(s), s)\, ds$ for $a \leq t \leq b$,

(A.5) $\hat{c} = c_*(b)$ and $\hat{q} = \xi_*(b)$.

If we define $\xi_*^{\#}(t) = (c_*(t), \xi_*(t))$, then $(\xi_*^{\#}, \eta_*)$ is a trajectory-control pair for $\Sigma^{\#}$, such that $\xi_*^{\#}(a) = \bar{q}^{\#} \coloneqq (0, \bar{q})$ and $\xi_*^{\#}(b) \in S^{\#}$. The fact that (ξ_*, η_*) is a solution of (GEOCP) implies that $\mathcal{R}^{\Sigma^{\#}}_{[a,b]}(\bar{q}^{\#}) \cap S^{\#} = \{\bar{q}^{\#}\}$. If we then apply Theorm 2.1.3 (SSMP), and make the reasonable assumption that

Assumption 2.1.1. *if $S \subseteq \mathbb{R}^n$, $\hat{q} \in S$, $\hat{c} \in \mathbb{R}$, C is a tangent cone to S at $\hat{q}$, and $S^\#$ is defined by (A.3), then $C^\# \coloneqq [0, +\infty] \times C$ is a tangent cone to $S^\#$ at $(\hat{c}, \hat{q})$,*

then we get the usual Pontryagin's Maximum Principle (Theorem 2.1.2). Note that in this case it is not necessary to require that the cone C not be a linear subspace, because $C^\#$ will never be a linear subspace, even if C is.

Remark. In Theorem 2.1.3, the requirement that C is not a linear subspace of $\mathbb{R}^n$ is necessary, for otherwise (SSMP) can fail for trivial reasons. To see this, take $Q = \mathbb{R}^n$, let S be a linear subspace of $\mathbb{R}^n$, let T be a complementary subspace of S, and consider the control system $\dot{q} = u$, $u \in T$, $q \in \mathbb{R}^n$. The reachable set from 0 over any interval $[a, b]$ with $a < b$ is T. So this set is separated from S at 0. On the other hand, the adjoint equation just says that $\dot{\psi} = 0$, and the Hamiltonian maximization condition implies that $\psi(t)$ must annihilate T. Then an obvious choice for C is S itself, and thus the transversality condition $-\psi(b) \in C^\dagger$ says that $\psi(b)$ must annihilate S. So $\psi(t) \equiv 0$, contradicting the nontriviality of ψ. □

In addition to implying the usual necessary conditions for optimality, Theorem 2.1.3 (SSMP) has the additional advantage that, properly interpreted, it also implies a *sufficient condition for local controllability along a reference trajectory*, or equivalently, a *necessary condition for a system not to be locally controllable along a reference trajectory.*

Local controllability can be easily interpreted by the reachable set. That is, Σ is *locally controllable* along a trajectory-control pair (ξ_*, η_*) defined on an interval $[a, b]$ if $\xi_*(b)$ is an interior point of $\mathcal{R}^\Sigma_{[a,b]}(\xi_*(a))$. So, if Σ is not locally controllable along (ξ_*, η_*), then there must exist a sequence $\{q_j\}$ of points of Q that converge to $\xi_*(b)$ and do not belong to $\mathcal{R}^\Sigma_{[a,b]}(\xi_*(a))$. By taking a subsequence, if necessary, we can assume that the limiting direction

$$v = \lim_{j \to \infty} \frac{q_j - \hat{q}}{\|q_j - \hat{q}\|} \tag{2.23}$$

exists. Let

$$S = \{\hat{q}\} \cup \{q_j : j = 1, 2, \ldots\}\,. \tag{2.24}$$

Then it is clear that the sets S and $\mathcal{R}^{\Sigma}_{[a,b]}(\xi_*(a))$ are separated at $\xi_*(b)$. Suppose we could take the tangent cone C to S at $\xi_*(b)$ to be the half-line $\{rv : r \geq 0\}$. Then we could apply (SSMP) to get the following theorem.

Theorem 2.1.4 (**Local Controllability Maximum Principle (LCMP)**). *Let $\Sigma = (Q, U, I, f)$ be a control system in $\mathbb{R}^n$. Assume that $a \leq b$, $a, b \in I$, and $\xi_* : [a, b] \to Q$ is a trajectory of Σ corresponding to a control $\eta_* : [a, b] \to U$. Then a necessary condition for $\mathcal{R}^{\Sigma}_{[a,b]}(\xi_*(a))$ not to be a neighborhood of $\xi_*(b)$ is that there exist a nonzero map $[a, b] \ni t \to \psi(t) \in \mathbb{R}_n$ such that (HMC) and (AE) hold.*

However, for the general (SSMP), in addtion to Assumption 2.1.1, we would still need a "tangent cone" that satisfies

Assumption 2.1.2. If S is given by (2.24), where $\{q_j\}$ is a sequence such that $q_j \to \hat{q}$, $q_j \neq \hat{q}$, and the limit (2.23) exists, then the half-line $\{rv : r \geq 0\}$ is a tangent cone to S at $\hat{q}$.

2.1.6 Weakly Approximating Cones and Transversality

In the following, we will use tools from convex analysis to derive a "tangent cone" that satisfies both assumptions and is also usable in the Maximum Principle (Theorem 2.1.3). We give the definition upfront:

Definition 2.1.3. Suppose $S \subseteq \mathbb{R}^n$, $\hat{q} \in S$, and $C \subseteq \mathbb{R}^n$. We say that C is a *weakly approximating cone* to S at $\hat{q}$ if

i) C is a closed convex cone;

ii) there exist a closed subset C^* of C, and a continuous map $F : C^* \to S$, such that

$$\lim_{\varepsilon \downarrow 0} \widetilde{\sup}\Big\{ \|\xi(1)\| : \xi \in C^0([0,1], C \backslash C^*), ||\xi(0)|| \leq \varepsilon \Big\} = 0\,, \tag{2.25}$$

$$F(v) = \hat{q} + v + o(\|v\|) \qquad \text{as} \qquad v \to 0\,,\ v \in C^*\,. \tag{2.26}$$

In this definition, $C^0([0,1], C\backslash C^*)$ is the set of continuous maps from $[0,1]$ to $C\backslash C^*$. The expression "$\widetilde{\sup}$" stands for "supremum in the set $[0,+\infty]$" so that $\widetilde{\sup}(\emptyset) = 0$. It then follows that (2.25) holds, in particular, if C^* is a full relative neighborhood of 0 in C.

Notice that Definition 2.1.3 ii) implies that $0 \in C^*$ because, if $0 \notin C^*$, then there would exist a convex neighborhood U of 0 such that $U \cap C^* = \emptyset$, since C^* is closed. But then there would be a curve $\xi \in C^0([0,1], C\backslash C^*)$ such that $\xi(0) = 0$ and $\xi(1) \neq 0$, contradicting (2.25).)

Condition (2.25) says roughly that "within C, it is not possible to join points that are close to 0 to points that are far from 0 without hitting the set C^*."

Definition 2.1.4. If the set C^* can be taken to be a full relative neighborhood of 0 in C, then we call C an *approximating cone* to S at $\hat{q}$. In other words, C is an *approximating cone* to S at $\hat{q}$ if

i) C is a closed convex cone;

ii) there exist a neighborhood U of 0 and a continuous map F from $C \cap U$ to S, such that $F(v) = \hat{q} + v + o(\|v\|)$ as $v \to 0$ via values in C.

A fundamental difference between the Transversality Condition (TC) here in Theorem 2.1.3 (SSMP) and that of [59] is from the fact that (SSMP) uses a much weaker concept of "tangent cone" than that of [59]. Indeed, the concept of tangent cone implicitly used in [59] in the formulation of TC is in fact exactly the "approximating cone" in Definition 2.1.4. The following example shows how this makes a difference.

Remark. Consider the optimal control problem in $\mathbb{R}^2$–with coordinate point $q = (x, y)$–in which it is desired to minimize the integral

$$J = -\int_0^1 u(t)dt\,,$$

subject to the dynamical law $\dot{x} = u$, $\dot{y} = v$ and the endpoint constraints $\xi(0) = (0,0)$,

$\xi(1) \in S$, where

$$S = \left\{(x, y) : x > 0 \wedge y = x \sin \frac{1}{x}\right\} \cup \{(0, 0)\}.$$

Then the trajectory $[0, 1] \ni t \to (0, 0)$, corresponding to the controls $u(t) \equiv v(t) \equiv 0$, obviously fails to be a minimizer, since the choice of controls $v(t) \equiv 0$, $u(t) \equiv \frac{\pi}{k}$, for any positive integer k, will yield a trajectory satisfying our endpoint constraints and having a lower cost.

However, the version of the Maximum Principle given in [59] will not exclude this trajectory as a candidate for a minimum. This can be seen by explicitly finding an adjoint vector that satisfies all the conditions, which is quite easy to do. There is, however, an even simpler argument. It suffices to notice that the only approximating cone for the set S at $(0, 0)$ is $\{(0, 0)\}$. Hence the conclusion of the Maximum Principle for our optimization problem is identical to that arising from any modified problem with the same dynamics and cost functional, and with the set S replaced by another set S' having $\{(0, 0)\}$ as an approximating cone at $(0, 0)$. In particular, we can take $S' = \{(0, 0)\}$. Since the cost of any feasible trajectory $t \to \xi(t) = (x(t), y(t))$ is just $-x(1)$, we see that all feasible trajectories of our original problem that end at $(0, 0)$ are actually minimizers for the modified problem, so they satisfy the conditions of the Maximum Principle of [59].

In contrast, the transversality condition in Theorem 2.1.3 does exclude the trajectory $[0, 1] \ni t \to (0, 0)$. To see this, notice that the cone $C = \{(x, y) : |y| \leq x\}$ is a "tangent cone" to S at $(0, 0)$ in our sense. To satisfy the necessary condition of the Maximum Principle we need a vector $(\psi_1, \psi_2) \in \mathbb{R}_2$ and a nonnegative ψ_0 such that, to begin with, $\psi_1 u + \psi_2 v + \psi_0 u$ is maximized by $u = v = 0$, for which we must have $\psi_2 = 0$ and $\psi_1 + \psi_0 = 0$. Moreover, since $(\psi_0, \psi_1, \psi_2) \neq (0, 0, 0)$, we need $\psi_0 > 0$ and $\psi_1 < 0$. But the transversality condition yields $-(\psi_1, \psi_2) \in C^\dagger$, so $\psi_1 \geq 0$, since $(1, 0) \in C$. Hence, we have reached a contradiction. □

It turns out that (SSMP) holds if we use the"weakly approximating cone" as the "tangent cone". To understand how this works, we first need the following theorem proved in [70]:

Theorem 2.1.5. *Let C_1 be a closed convex cone in $\mathbb{R}^m$, let Ω be a neighborhood of 0 in $\mathbb{R}^m$, let $f : \Omega \cap C_1 \to \mathbb{R}^n$ be a continuous map, and let $A : \mathbb{R}^m \to \mathbb{R}^n$ be linear. Suppose that f is*

differentiable at 0 *with differential* A*, i.e. that*

$$f(x) = f(0) + Ax + o(||x||) \qquad \text{as} \qquad x \to_{C_1} 0\,. \tag{2.27}$$

Let S *be a subset of* $\mathbb{R}^n$ *having a cone* C_2 *as weakly approximating cone at* $f(0)$*. Assume that* C_2 *is not a linear subspace of* $\mathbb{R}^n$*. Then a necessary condition for the sets* $f(\Omega \cap C_1)$ *and* S *to be locally separated at* $f(0)$ *is that* $(AC_1)^\dagger \cap (-C_2)^\dagger \neq \{0\}$*.*

In other words, if $f(\Omega \cap C_1) \cap S \cap V = \{f(0)\}$ for some neighborhood V of $f(0)$, then there exists a nonzero row vector $\bar{z} \in \mathbb{R}_n$ such that $\bar{z}Av_1 \leq 0$ for all $v \in C_1$ and $\bar{z}.v_2 \geq 0$ for all $v \in C_2$. Note that this result closely relates to the *polar cone* of the linearized constriant set AC_1 and the terminal set S in the from of its "tangent set" C_2, which is in turn also reflected in the *transversality condition.*

The main proof strategy for Theorem 2.1.3 (SSMP) consists of constructing needle variations, studying the corresponding endpoint maps, computing their differentials, and then applying a separation result. For conciseness, the rest of the proof will be omitted. The reader is referred to [70] if interested. Classically, when the differentials used are classical differentials, the appropriate separation result is the aforementioned Theorem 2.1.5. Next we will show how, using essentially the same approach but relying on more general concepts of differential, it is possible to derive much stronger results that contain, extend, and strengthen various "nonsmooth" versions of the Maximum Principle.

2.1.7 Clarke's Nonsmooth Maximum Principle

To every map $\eta : [a, b] \to U$, we associate a time-varying vector field $f_\eta : Q \times [a, b] \to \mathbb{R}^n$ by letting $f_\eta(q, t) \coloneqq f(q, \eta(t), t)$.

Definition 2.1.5. We say that a time-varying $\mathbb{R}^n$-valued vector field F, defined on some subset $\mathrm{Dom}(F)$ of $\mathbb{R}^n \times \mathbb{R}$, satisfies the C^1*-Carathéodory condition* near a curve $\xi : [a, b] \to \mathbb{R}^n$

if there exists a tube

$$\mathcal{T}(\xi,\varepsilon) \coloneqq \{(q,t) : a \leq t \leq b, \|q-\xi(t)\| \leq \varepsilon\}, \tag{2.28}$$

such that

i) $F(q,t)$ is measurable with respect to t–on the compact set $\{t : (q,t) \in \mathcal{T}(\xi,\varepsilon)\}$–for each fixed q, and of class C^1 with respect to q–on the closed ball $\{q \in \mathbb{R} : \|q-\xi(t)\| \leq \varepsilon\}$–for each fixed $t \in [a,b]$,

ii) there is a function $\varphi \in L^1([a,b],\mathbb{R})$ such that

$$\|F(q,t)\| + \left\|\frac{\partial F}{\partial q}(q,t)\right\| \leq \varphi(t)$$

for all $(q,t) \in \mathcal{T}(\xi,\varepsilon)$.

In the proof of Theorem 2.1.3 (SSMP), there is this following "classical technical hypothesis" that has to be met:

Assumption 2.1.3. For every control η which is either constant or equal to the reference control η_*, the corresponding time-varying vector field f_η satisfies a C^1-Carathéodory condition (See Definition 2.1.5) near ξ_*.

F.H. Clarke in [16] and [18] proposed an extension to the classical version of the Maximum Principle, which has become known as the "nonsmooth Maximum Principle.". In Clarke's version, Assumption 2.1.3 is replaced by the weaker requirement that the vector fields $(q,t) \to f(q,\eta(t),t)$ be Lipschitz continuous, with an integral bound on the Lipschitz constant, or more precisely described in Assumption 2.1.4.

Definition 2.1.6. A time-varying vector field F defined on some subset $\mathrm{Dom}(F)$ of $\mathbb{R}^n \times \mathbb{R}$, satisfies a *Lipschitz-Carathéodory condition* near a curve $\xi : [a,b] \to \mathbb{R}^n$ if there exists a tube

$\mathcal{T}(\xi, \varepsilon)$ such that Definition 2.1.5 i) holds, and there is a function $\varphi \in L^1([a,b], \mathbb{R})$ such that

$$\|F(q,t)\| \leq \varphi(t) \quad \text{and} \quad \|F(q',t) - F(q,t)\| \leq \varphi(t).\|q' - q\|$$

whenever $(q,t) \in \mathcal{T}(\xi, \varepsilon)$ and $(q',t) \in \mathcal{T}(\xi, \varepsilon)$.

Assumption 2.1.4. For every control η that is either constant or equal to the reference control η_*, the corresponding time-varying vector field f_η satisfies a Lipschitz-Carathéodory (See Definition 2.1.6)condition near ξ_*.

Yet another "classical technical hypothesis" needed in the proof of Theorem 2.1.3 is that

Assumption 2.1.5. a solution of the adjoint equation (AE) along the reference trajectory-control pair $\gamma_* = (\xi_*, \eta_*)$ is an absolutely continuous map $\psi : [a,b] \to \mathbb{R}_n$ such that the equality

$$-\dot{\psi}(t) = \psi(t).\frac{\partial f}{\partial q}(\xi_*(t), \eta_*(t), t)$$

holds for almost all $t \in [a_\eta, b_\eta]$.

Naturally, when the weaker condition Assumption 2.1.4 holds, it is no longer possible to interpret the adjoint equation in the classical sense, so Assumption 2.1.5 has to be modified. Clarke proposed a modification that the adjoint equation be interpreted as a *differential inclusion* with the classical Jacobian matrix replaced by a Clarke generalized Jacobian. In other words, Clarke suggested that, instead of Assumption 2.1.5, the following be used:

Assumption 2.1.6. A solution of the adjoint equation (AE) along the reference trajectory-control pair $\gamma_* = (\xi_*, \eta_*)$ is an absolutely continuous map $\psi : [a,b] \to \mathbb{R}_n$ such that

$$-\dot{\psi}(t) \in \psi(t) \cdot \partial_1 f(\xi_*(t), \eta_*(t), t) \quad \textit{for a. e. } t \in [a,b]\,. \tag{2.29}$$

Here $\partial_1 f(\bar{q}, u, t)$ is the *Clarke's generalized Jacobian* (CGJ) at $\bar{q}$ of the map $q \to f(q,u,t)$. For a Lipschitz continuous map $\mu : Q \to \mathbb{R}^m$, where Q is open in $\mathbb{R}^n$, the CGJ of μ at $q \in Q$

is the convex hull $\partial\mu(q)$ of the set of all linear maps $A : \mathbb{R}^n \to \mathbb{R}^m$ such that $L = \lim_{j\to\infty} A_j$ for some sequence $\{A_j\}$ such that there are $q_j \in Q$ for which $q_j \to q$ and μ is differentiable at q_j with differential A_j. The existence of many such sequences $\{q_j\}$ follows because μ is differentiable almost everywhere. The set $\partial\mu(q)$ is then nonempty, compact and convex.

The appearance of CGJ renders the problem nonsmooth, but with the corresponding conditions (Assumption 2.1.4 and Assumption 2.1.6), Clarke was still able to prove that the nonsmooth Maximum Principle holds [16]. The concept of CGJ and the closely related *Clarke's generalized differential* and *Clarke's generalized gradient*, etc. will be extensively used in the author's convergence proof for the `SCvx` algorithm, due to the presence of nonsmooth penalty functions.

2.2 Finite-dimensional Optimization

This section gives an overview of the field of finite-dimensional optimization. The notion of finite dimensions is to contrast with the optimal control problems we just covered in Section 2.1, which can be seen as infinite-dimensional optimization problems. Finite-dimensional optimization problems are everywhere. The engineer who designs an aircraft with minimal fuel consumption given its aerodynamics, the investor who maximizes profit within constraints imposed by the risks/resources, the bike courier who seeks the shortest path between two points in a city, and the AI recommending you movies to watch/products to buy etc., all solve optimization problems.

2.2.1 General Finite-dimensional Optimization Problem and Examples

Mathematically, we can formulate an important class of such problems as follows:

Problem 2.2.1 (General Finite-dimensional Optimization Problem).

$$\begin{aligned} \text{(P)} \qquad & \min_{x\in\mathbb{R}^n} f(x) \\ & \text{s.t.} \quad g_i(x) \geq 0, \quad (i = 1, \ldots, p), \\ & \qquad\; h_j(x) = 0, \quad (j = 1, \ldots, q), \end{aligned}$$

where f, g_i and h_j are sufficiently smooth functions: typically we require them to be twice continuously differentiable.

The function f represents an objective (such as energy, cost etc.) that has to be minimized under side constraints defined by the functions g_i and h_j. We therefore call f the *objective function*, the functions g_i the *equality constraint functions*, and the functions h_j the *inequality constraint* functions of (P). Note that by replacing f by $-f$ we can of course treat *maximization problems* in the same framework.

Example 2.2.1 (Linear Programming (LP)). The *transportation network problem* occurs when the cheapest way of shipping a certain amount of a commodity across a transportation network has to be determined. This can be a network of gas pipelines, a computer network, a power grid, a road network etc..

A network of mail services is given in Figure 2.1. An arrow from node i to node j represents a service route with transport capacity c_{ij} in the given direction. Transporting one unit of mails along the edge (ij) costs d_{ij}. The amount of mails collected at node i is p_i, and the amount of mail dispatched is q_i. We assume that the total amount mail dispatched equals the total amount of collected. Then the question is to determine how the quantities x_{ij} of mails shipped along the edges (ij) should be chosen so as to satisfy all the mail collection/dispatch demands and at the same time to minimize costs.

We set $c_{ij} = 0$ for all edges (ij) that do not exist, and thus we can assume that the

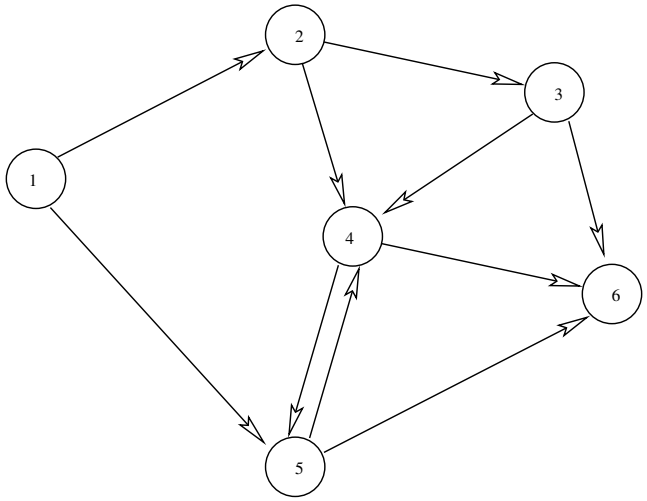


Figure 2.1: *Mail services network illustration.*

network is a complete graph. The problem can then be formulated as

$$\begin{aligned} &\min_x \sum_{i,j=1}^{6} d_{ij} x_{ij} \\ &\text{s.t.} \quad \sum_{k=1}^{6} x_{ki} + p_i = \sum_{j=1}^{6} x_{ij} + q_i, \qquad (i = 1, \ldots, 6), \\ &\qquad 0 \le x_{ij} \le c_{ij}, \qquad (i, j = 1, \ldots, 6). \end{aligned}$$

This is an example of a *linear programming* problem, as the objective function and all the constraint functions are linear.

Note that it is not a priori clear that this problem has feasible solutions. One is therefore interested in algorithms that not only find optimal LP solutions if exist but also detect when a problem instance is infeasible.

Example 2.2.2 (Quadratic Programming (QP)). In the portfolio management problem, an investor considers a fixed time interval and wants to decide which fraction of the capital

he/she wants to invest in each from n different given assets when the expected return of asset i is μ_i and the covariance between assets i and j is σ_{ij}. The vector $\mu = [\mu_i]$ and the matrix $\sigma = [\sigma_{ij}]$ are assumed to be known and the investor aims at a total return of at least b. Subject to this constraint, he/she aims to minimise the risk as quantified by the variance of the overall portfolio.

This problem can be modeled as

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mu_i x_i \geq b, \\ & \sum_{i=1}^{n} x_i = 1, \\ & x_i \geq 0 \quad (i = 1, \ldots, n). \end{aligned}$$

The constraint $\sum_{i=1}^{n} x_i = 1$ expresses the requirement that 100% of the initial capital has to be invested.

Example 2.2.3 (Semidefinite Programming (SDP)). In optimal control, variables $u_1, \ldots, u_m$ have to be chosen so as to design a system that is driven by the linear ODE

$$\dot{x} = M(u)x,$$

where $M(u) = \sum_{i=1}^{m} u_i A_i + A_0$ is an affine combination of given symmetric matrices A_i $(i = 0, \ldots, m)$. To stabilize the system, one would like to choose u so as to minimize the largest eigenvalue of $M(u)$.

Note that $\lambda_1(M) \leq \eta$ if and only if $\eta I - M \succeq 0$ (is positive semidefinite). Therefore, the

problem we need to solve becomes

$$\begin{aligned} &\max_{\eta,u} \; -\eta \\ &\text{s.t.} \quad \eta I - A_0 - \sum_{i=1}^{m} u_i A_i \succeq 0. \end{aligned}$$

2.2.2 Local versus Global Optimality

In order to rigorously define the finite-dimensional optimization problems and their solutions, we need to first explain what the "optimal solution" to the problem (P) means. A point $x \in \mathbb{R}^n$ is *feasible* for the optimization problem (P) if $g_i(x) \geq 0 \; \forall i$ and $h_j(x) = 0 \; \forall j$, that is, if x satisfies all the constraints of the problem. The set $\mathcal{F}$ of feasible points is called the *domain of feasibility* of (P). A feasible point x^* is a *local minimizer* if there exists a ball $B_\epsilon(x^*)$ around x^* such that

$$f(x^*) \leq f(x) \qquad \forall x \in B_\epsilon(x^*) \cap \mathcal{F},$$

that is, x^* is a minimizer among all the feasible points in a neighborhood of x^*, but there might be feasible points further away from x^* where the objective function takes an even smaller value. A feasible point x^* is a *global minimizer* if

$$f(x^*) \leq f(x) \qquad \forall x \in \mathcal{F},$$

that is, x^* minimizes the objective function among all feasible points of the problem, although there might exist several of these points.

Example 2.2.4. The problem

$$(P) \qquad \min_{x\in\mathbb{R}} f(x) = x^3 + 9x^2$$
$$\text{s.}t. \quad -10 \leq x \leq 2$$

has a local minimizer at $x = 0$, and a global minimizer at $x^* = -10$, see Figure 2.2.

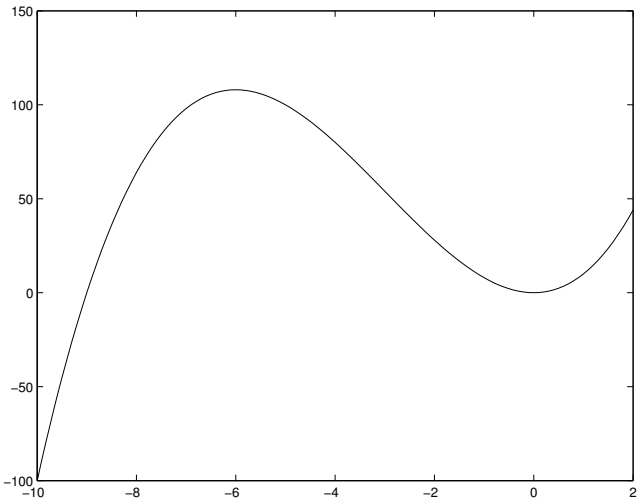


Figure 2.2: *Objective function of Example 2.2.4.*

In the general framework, efficient algorithms can only be devised for the problem of finding a *local* minimizer. The problem of finding a global minimizer is indeed important in practice, but its solution is typically based on heuristics that rely on local minimization as a subproblem. We therefore restrict the scope of this section to only local minimization.

A slightly confusing terminology is the following: an iterative algorithm for solving (P) converges *globally* if the output sequence $(x_k)_{\mathbb{N}}$ converges to a *local minimizer* x^* for all starting points $x_0 \in \mathcal{F}$. On the other hand, an iterative algorithm is called *locally convergent* if the output sequence $(x_k)_{\mathbb{N}}$ converges to a *local minimizer* x^* for all feasible starting points

x_0 close enough to x^*, that is, for all $x_0 \in B_r(x^*) \cap \mathcal{F}$ for some $r > 0$.

Example 2.2.5. Back to the problem of Example 2.2.4 and consider the following algorithm:

S0 Choose x_0. Set $\alpha = 1$, $k = 0$.

S1 $x = x_k + \alpha f'(x_k)$.

S2 If x is feasible then goto **S3**, else $\alpha \leftarrow \alpha/2$ and goto **S1**.

S3 Set $x_{k+1} = x$, $k \leftarrow k + 1$, $\alpha = 1$, and goto **S1**.

This algorithm converges to the local minimizer $x^* = 0$ for all starting points $x_0 \in (-6, 2]$, and to the global minimizer $x^* = -10$ for $x_0 \in [-10, -6)$. For $x_0 = -6$ it remains stuck. If we exclude x_0 as a starting point, then this algorithm is globally convergent, even though it only converges to local minimizers. The focus here is that the algorithm converges no matter what the starting point is.

On the other hand, if we omit the judicious choice of α, we obtain the following algorithm:

S0 Choose x_0. Set $k = 0$.

S1 Set $x_{k+1} = x_k + f'(x_k)$, $k \leftarrow k + 1$, and goto **S1**.

This algorithm converges locally to the local minimizer $x^* = 0$, but for values $x_0 < -6$ the algorithm diverges to $-\infty$ and becomes infeasible instead of finding the local (and global) minimizer $x^* = -10$.

2.2.3 Convergence Rates

Since much of numerical analysis is devoted to the construction of algorithms that converge quickly, we should be able to quantify convergence speed.

A converging sequence $(x_k)_{\mathbb{N}} \to x^*$ has Q-convergence rate $r \geq 1$ (Q stands for "quotient") if $\exists \rho > 0$ and $k_0 \in \mathbb{N}$ such that

$$\|x_{k+1} - x^*\| \leq \rho \|x_k - x^*\|^r, \quad \forall k \geq k_0.$$

Note that when $r = 1$, only bounds with $\rho < 1$ are useful. If $r = 1$ or $r = 2$ we speak of Q-linear and Q-quadratic convergence respectively. Finally, $(x_k)_{\mathbb{N}}$ converges Q-superlinearly if

$$\lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0.$$

Example 2.2.6. Let $z \in (0, 1)$ and consider the sequence $(x_k)_{\mathbb{N}}$ defined by

$$x_k := \sum_{n=0}^{k} z^n.$$

Then $(x_k)_{\mathbb{N}}$ converges Q-linearly but not Q-superlinearly to $x^* = (1 - z)^{-1}$. Indeed, we have $|x_k - x^*| = \sum_{n=k+1}^{\infty} z^n$. Therefore,

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|} = z < 1,$$

which shows the claim.

We say that an iterative algorithm has Q-convergence rate r if every output sequence produced by it converges at least at the Q-convergence rate r.

The practical significance of Q-linear convergence is that asymptotically the point x_{k+1} approximates x^* with $\log_{10} \rho$ more correct digits than x_k. This means that the number of

correct digits grows linearly in the number of iterations taken. For Q-convergence of order r on the other hand, the number of additional correct digits asymptotically grows by a factor of r, that is, the number of correct digits is exponential in the number of iterations taken and the convergence is super fast.

In practice though, Q-convergence of any order $r > 1$ is qualitatively similar to convergence of any other order because applying an order r algorithm j steps at a time yields an order r^j algorithm.

2.2.4 Convex Sets and Convex Functions

The notion of convexity plays a central role in optimization. A set $C \subseteq \mathbb{R}^n$ is *convex* if

$$x, y \in C \Rightarrow \lambda x + (1 - \lambda)y \in C \quad \forall\, \lambda \in [0, 1],$$

that is, if the straight line segment joining any two elements of C lies in C. The empty set, half spaces $\{x : a^{\mathrm{T}}x \geq 0\}$, polyhedra $\{x : Ax \geq b\}$, open balls $B_\rho(\bar{x}) = \{x : \|x - \bar{x}\| < \rho\}$, ellipsoids $\{x : x^{\mathrm{T}}Bx \leq r\}$ (with B a positive definite matrix) and affine subspaces $\{x : a^{\mathrm{T}}x = b\}$ are all examples of convex sets.

If $C, D \subseteq \mathbb{R}^n$ are convex sets, $\lambda \in \mathbb{R}$ and $\varphi : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map, then $C + D := \{x + y : x \in C, y \in D\}$, $\lambda C := \{\lambda x : x \in C\}$, $\varphi(C) := \{\varphi(x) : x \in C\}$ and $C \cap D$ are convex sets.

Functions $f : \mathbb{R}^n \to (-\infty, +\infty]$ into the real line extended by $+\infty$ are called *proper*. A proper function is *convex* if its epigraph

$$\mathrm{epi}(f) := \big\{(x, z) \in \mathbb{R}^{n+1} : f(x) \leq z\big\}$$

is a convex set in $\mathbb{R}^{n+1}$.

A proper function f (assumed to be defined on all of $\mathbb{R}^n$) is convex if and only if

$$f\big(\lambda x + (1-\lambda)y\big) \leq \lambda f(x) + (1-\lambda) f(y) \tag{2.30}$$

for all $x, y \in \mathbb{R}^n$, $\lambda \in [0,1]$. If this becomes a strict inequality $<$ for all $\lambda \in (0,1)$ we say that f is *strictly convex*.

If f is convex then its *effective domain* $\mathrm{dom}(f) := \{x : \ f(x) < +\infty\}$ is a convex set in $\mathbb{R}^n$. On the other hand, we call any function $f : C \to \mathbb{R}$ which is defined on a convex set C and satisfies (2.30) for all $x, y \in C$ and $\lambda \in [0,1]$ convex, and any such function can be extended to a convex proper function by setting $f(x) := +\infty$ for all $x \notin C$.

If f and g are convex proper functions then so are $f + g$ and λf for any $\lambda \geq 0$. If $\mathcal{F}$ is a set of convex proper functions then the pointwise supremum

$$\sup_{\mathcal{F}} : \ x \mapsto \sup\{f(x) : \ f \in \mathcal{F}\}$$

is a convex proper function. In particular, the pointwise maximum of finitely many convex proper functions is convex.

If f is a convex proper function then all its level sets $\{x : \ f(x) \leq z\}$ (where $z \in (-\infty, +\infty]$ is fixed) are convex. Any convex proper function f is continuous on the topological interior $\mathrm{intr}\big(\mathrm{dom}(f)\big)$ of its effective domain.

A proper function $g : \mathbb{R}^n \to [-\infty, +\infty)$ or a function $g : C \to [-\infty, +\infty)$ defined on a convex set is called *concave* if $-g$ is convex.

To see why convexity is so important, look no further than its first order optimality conditions. Simply put, without convexity, finding a global minimizer is mostly out of reach.

Theorem 2.2.1 (First order differential properties of convex functions)**.**
Let $f : D \to \mathbb{R}$ be a function defined on a convex open domain $D \subset \mathbb{R}^n$.

(i) If f is convex then x^* is a local minimizer if and only if it is a global minimizer.

(ii) If f is C^1 on D, then f is convex if and only if for all $x, y \in D$,

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x), \tag{2.31}$$

that is, the graph of the first order approximation of f at x lies below the graph of f.

(iii) If f is convex and $\nabla f(x^*) = 0$ then x^* is a global minimizer of f. If $D = \mathbb{R}^n$ then this condition is both sufficient and necessary.

(iv) f is both convex and concave if and only if f is an affine function.

Proof. Suppose $x^* \in D$ is a local but not a global minimiser. Then there exists a $y \in D$ such that $f(y) < f(x^*)$, and then $f(\lambda y + (1-\lambda)x^*) \leq \lambda f(y) + (1-\lambda)f(x^*) < f(x^*)$ for all $\lambda \in [0, 1)$ and x^* cannot be a local minimiser because λ can be chosen arbitrarily close to 0. On the other hand, every global minimiser is a local minimiser. This proves (i). Suppose now that f satisfies (2.31). Given $\lambda \in [0, 1]$ and $x, y \in D$, let $z = (1-\lambda)x + \lambda y$. (2.31) implies

$$\begin{aligned} f(x) &\geq f(z) + \nabla f(z) \cdot (x - z) \quad \text{and} \\ f(y) &\geq f(z) + \nabla f(z) \cdot (y - z). \end{aligned}$$

Multiplying the first inequality by $(1-\lambda)$ and the second by λ, and adding the two inequalities we get $f(z) \leq (1-\lambda)f(x) + \lambda f(y)$. Hence, f is convex. Suppose on the other hand that f is convex. Then $f\big(x + \lambda(y-x)\big) \leq f(x) + \lambda\big(f(y) - f(x)\big)$, and hence

$$\frac{f\big(x + \lambda(y-x)\big) - f(x)}{\lambda} \leq f(y) - f(x)$$

Taking limits as $\lambda \to 0$ we get (2.31). This proves (ii). (iii) is a trivial consequence of (i) and (ii). If f is affine, then it is clearly both convex and concave. On the other hand, if f is both convex and concave, and if f is differentiable at least at one point x^* then it follows

from (2.31) that $f(y) \geq f(x^*) + \nabla f(x^*) \cdot (y - x^*)$ and $-f(y) \geq -f(x^*) - \nabla f(x^*) \cdot (y - x^*)$ for all y. Hence, $f(y) \equiv f(x^*) + \nabla f(x^*) \cdot (y - x^*)$. The general case can be proved in a similar way using the notion of subdifferential. One can also prove that there are always points where f is differentiable, but this is technically more difficult. □

Theorem 2.2.2 (Second order differential properties of convex functions)**.**
Let $f : D \to \mathbb{R}$ be a function defined on a convex open domain $D \subset \mathbb{R}^n$.

(i) If f is convex, $x \in D$ and the Hessian $H(x) = f''(x)$ exists, then $H(x) \succeq 0$ (positive semidefinite, that is, $z^{\mathrm{T}} H(x) z \geq 0$ for all $z \in \mathbb{R}$).

(ii) If $H(x)$ exists for all $x \in D$ and $H(x) \succeq 0$ then f is convex.

(iii) If $H(x)$ exists for all $x \in D$ and $H(x) \succ 0$ (positive definite, that is, $z^{\mathrm{T}} H(x) z > 0$ for all $z \in \mathbb{R} \setminus \{0\}$) then f is strictly convex.

The proof of Theorem 2.2.2 is omitted here for simplicity. The reader is referred to [14] for more details.

2.2.5 Unconstrained Optimization

The subject of this section is the unconstrained minimization problem, we will only introduce the problem and its optimality conditions, along with some notable solution methods that are related to the author's work. The problem formulation is in fact quite simple:

$$\min_{x \in \mathbb{R}^n} \; f(x), \tag{2.32}$$

where f is a continuous objective function. Note that no constraints imposed on the decision variables! Furthermore, we usually assume that f is C^2 with Lipschitz-continuous Hessian,

that is, there exists $\Lambda > 0$ such that

$$\|D^2 f(x) - D^2 f(y)\| \leq \Lambda \|x - y\| \quad \forall\, x, y \in \mathbb{R}^n.$$

In all that follows $\|x\| = \sqrt{\sum x_i^2}$ denotes the Euclidean norm of a vector $x \in \mathbb{R}^n$ and $\langle \cdot, \cdot \rangle$ is the corresponding Euclidean inner product. If $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map, then $\|A\|$ denotes the operator norm defined by the Euclidean norms on $\mathbb{R}^n$ and $\mathbb{R}^m$, that is,

$$\|A\| = \inf\{\lambda > 0 : \ \|Ax\| \leq \lambda \|x\| \ \forall\, x \in \mathbb{R}^n\}.$$

The gradient $\nabla f(x)$ of a function $f : \mathbb{R}^n \to \mathbb{R}$ is sometimes denoted by $g_f(x)$, and its Hessian $D^2 f(x)$ by $H_f(x)$. The Jacobian $Df(x)$ of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is sometimes denoted by $J_f(x)$. Note: if $m = 1$ then $J_f(x) = g_f(x)^{\mathrm{T}}$. We will also use the so-called "big O" notation: we say that a function $g(x)$ is *of order* $\|x\|^k$ and write $g(x) = O(\|x\|^k)$ if there exists a constant $c > 0$ and a $\delta > 0$ such that $|g(x)| \leq c\|x\|^k$ whenever $\|x\| < \delta$.

A well designed optimization algorithm should be able to recognize when an approximate minimum has been attained. We therefore need a mathematical characterization of local minimizers. In the univariate case, we all know that a necessary condition is $f'(x) = 0$, and that second derivatives help deciding whether x is a local maximizer or minimizer. The same idea works in higher dimensions:

Theorem 2.2.3 (Optimality Conditions for Unconstrained Minimization)**.** *(i) If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at $x^* \in \mathbb{R}^n$ and has a local minimum there, then $\nabla f(x^*) = 0$, that is, x^* is a* stationary point *of f. This is a first order necessary optimality condition, because it involves first derivatives, or the first order Taylor approximation of f.*

(ii) If $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable at $x^* \in \mathbb{R}^n$ and has a local minimum there, then the Hessian $D^2 f(x^*)$ is positive semidefinite, that is, $h^{\mathrm{T}} D^2 f(x^*) h \geq 0$ for all $h \in \mathbb{R}^n$.

This is a second order *necessary optimality condition.*

(iii) If $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable at $x^* \in \mathbb{R}^n$, and if $\nabla f(x^*) = 0$ and $D^2 f(x^*)$ is positive definite, that is, if $h^{\mathrm{T}} D^2 f(x^*) h > 0$ for all $h \in \mathbb{R}^n \setminus \{0\}$, then x^* is a local minimizer of f. These are sufficient optimality conditions.

Again, the proof is omitted here.

The optimality conditions in Theorem 2.2.3 play an important role in the construction of algorithms: Solving the simultaneous system of nonlinear equations

$$\nabla f(x) = 0$$

by an iterative procedure generating a sequence of points $(x_k)_{\mathbb{N}}$, if we can assure that $f(x_k)$ decreases in each iteration,

$$f(x_{k+1}) \leq f(x_k) \quad \forall\, k,$$

then in practice $(x_k)_{\mathbb{N}}$ can only converge to a *local minimizer* x^* and

$$\|\nabla f(x^*)\| < \epsilon$$

can be used as a stopping criterion. Thus, solving unconstrained optimization problems is closely related to the problem of solving simultaneous equations with the added feature that progress can be controlled by monitoring a naturally defined *merit function* f.

Most competitive algorithms for unconstrained minimisation are based on this idea. There are two main families of such methods: *line-search methods* and *trust region methods.* We start with a description of the former.

The minimal assumption we need to make for this algorithm to work is $f \in C^1$ with Lipschitz continuous gradient. The generality of Algorithm 2.2.5 leaves flexibility both in

Algorithm 1 Descent method

S0 Choose a starting point $x_0 \in \mathbb{R}^n$ and a tolerance parameter $\epsilon > 0$. Set $k = 0$.

S1 If $\|\nabla f(x_k)\| \leq \epsilon$ then stop and output x_k as an approximate local minimizer.

S2 Otherwise choose a search direction $d_k \in \mathbb{R}^n$ such that $\langle \nabla f(x_k), d_k \rangle < 0$.

S3 Choose a step size $\alpha_k > 0$ such that $f(x_k + \alpha_k d_k) < f(x_k)$.

S4 Set $x_{k+1} := x_k + \alpha_k d_k$, replace k by $k + 1$, and go back to S1.

the choice of the step length α_k and the search direction d_k. Different methods are devised based on these choices, and we will only cover the ones related to the author's research.

2.2.6 Newton's Method

The Newton–Raphson method, is probably the most-expensive-to-compute but fastest-to-converge method for unconstrained optimization, which sets the benchmark for convergence speed, both in theory and in practice.

Let us first consider how to find zeros of a differentiable univariate function g, that is, suppose we want to find $x^* \in \mathbb{R}$ such that $g(x^*) = 0$. If we are given an approximation x_k of x^* such that $g'(x_k) \neq 0$, then the linear function

$$\varphi : x \mapsto g(x_k) + g'(x_k)(x - x_k)$$

has the unique zero $x_{k+1} = x_k - g(x_k)/g'(x_k)$, see Figure 2.3. But φ is the first order Taylor approximation of g around x_k, which locally approximates g well. Therefore, it is reasonable to expect that under benevolent conditions x_{k+1} should be an even better approximation of x^* than x_k, and that $(x_k)_{\mathbb{N}}$ converges to x^* if the same process is applied in an iterative fashion. This is the *Newton–Raphson method* for zero-solving. The "benevolent" conditions

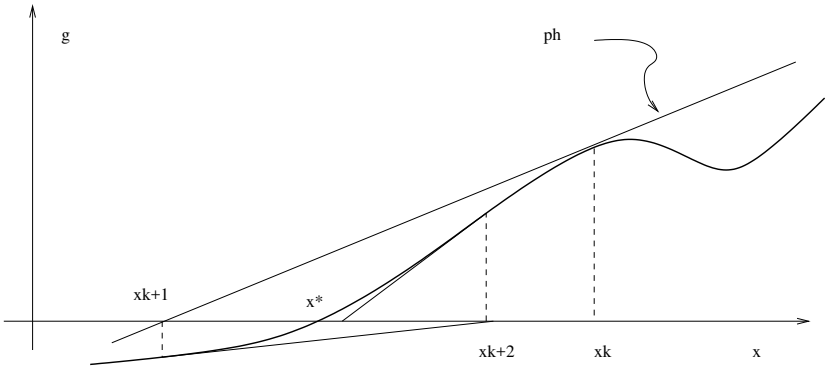


Figure 2.3: *Newton's method for univariate zero-finding.*

we need are simply that $g'(x^*) \neq 0$ and that x_k lies in sufficient proximity to x^*. This will follow from Theorem 2.2.4 below.

The multivariate version of the Newton–Raphson method is a direct generalization: we now seek a root (or zero) of a multivariate function $g : \mathbb{R}^n \to \mathbb{R}^n$. Given an approximate root $x_k \in \mathbb{R}^n$, we find the next iterate x_{k+1} as the zero of the first-order Taylor approximation of g around x_k, or in other words, as the root of the linear system of equations $g(x_k) + J_g(x_k)(x - x_k) = 0$, where $J_g(x) = \left[\frac{\partial g_i}{\partial x_j}\right]$ is the Jacobian of g at x. When $J_g(x)$ is nonsingular, this linear system has the unique solution

$$x_{k+1} = x_k - J_g(x_k)^{-1} g(x_k).$$

Note that if $g \in C^1$ and $\det J(x^*) \neq 0$ then $J_g(x)$ is nonsingular for all x_k in a neighborhood of x^*, and x_{k+1} is well defined.

Note that in this approach a difficult nonlinear problem is replaced by a sequence of easy linear problems. Much of numerical analysis follows similar approaches. Now, if we go back to the unconstrained optimisation problem (2.32), the first order necessary optimality condition of Theorem 2.2.3 (i) tells us that (2.32) can be replaced by the multivariate root-

finding problem

$$\nabla f(x) = 0. \tag{2.33}$$

When we apply the Newton–Raphson method to the zero-finding problem (2.33), we obtain the updating rule

$$x_{k+1} = x_k - \big(D^2 f(x_k)\big)^{-1} \nabla f(x_k), \tag{2.34}$$

which is well-defined as long as $D^2 f(x_k)$ is nonsingular. We call

$$n_f(x_k) := -\big(D^2 f(x_k)\big)^{-1} \nabla f(x_k)$$

the *Newton-step*.

Algorithm 2 Newton-Raphson method

Let $f \in C^2(\mathbb{R}^n)$ and let x_0 be an initial approximation of a local minimizer x^* of f. The Newton–Raphson method for solving (2.32) consists in computing the sequence of points $(x_k)_{\mathbb{N}}$ defined by

$$x_{k+1} = x_k + n_f(x_k).$$

This is an excellent method for minimizing convex functions, since $D^2 f(x_k) \succ 0$ implies

$$\langle n_f(x_k), \nabla f(x_k) \rangle = -\nabla f(x_k)^{\mathrm{T}} D^2 f(x_k) \nabla f(x_k) < 0,$$

that is, $n_f(x_k)$ is a descent direction. Furthermore, Theorem 2.2.4 below shows that the Newton–Raphson method converges to a stationary point x^* Q-quadratically, and since the local minimizers of a convex function are exactly characterized by the optimality condition $\nabla f(x^*) = 0$, x^* must be a local minimizer.

Regrettably, when f is not convex, the Newton–Raphson method is not guaranteed to converge to local minimizer. Cycling can occur (i.e., $x_{k+j} = x_k$ for some $k, j \in \mathbb{N}$) but is unlikely in practice, but more importantly, the method can converge to a local maximizer or

a saddle point of f.

As for the convergence speed of the Newton–Raphson method, we have

Theorem 2.2.4. *Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$ have Λ-Lipschitz continuous Hessian and let $x^* \in \mathbb{R}^n$ be a stationary point of f. If $D^2 f(x^*)$ is nonsingular then there exists a neighborhood $B_\rho(x^*)$ of x^* such that choosing $x_0 \in B_\rho(x^*)$ as a starting point and applying the Newton-Raphson method generates a sequence $(x_k)_\mathbb{N}$ that is is well-defined, lies in $B_\rho(x^*)$ and converges to x^* Q-quadratically.*

Theorem 2.2.4 shows that the Newton–Raphson method produces an output sequence $(x_k)_\mathbb{N}$ in which every x_{k+1} approximates x^* with roughly twice as many correct digits as x_k, as long as the process is started within $B_\rho(x^*)$. In applications it is a drawback that the radius ρ depends on the Hessian $D^2 f(x^*)$, which is of course unknown because x^* is unknown. The ball $B_\rho(x^*)$ is called the *domain of quadratic attraction* of x^*. During the 1990's, interesting new theories have been developed (most notably Nesterov and Nemirovskii's theory of self–concordant functions [57]) which show that for certain classes of objective functions there exist mathematical criteria which make it possible to detect that x_k lies in the domain of quadratic attraction without knowing the location of x^*. The ensuing interior-point methods based on these principles have revolutionized the way in which many important optimization problems are solved in practice, including linear programming.

The Newton–Raphson method plays indeed a uniquely important role as a basic element in many practical optimization algorithms. However, its usefulness has limitations in situations where computing the Hessian $D^2 f$ and solving the linear system $D^2 f(x_k) d_k = -\nabla f(x_k)$ to determine the Newton step is computationally too expensive. In some applications the problem dimension is so large that it is not possible to keep all the entries of $D^2 f$ in the main memory of a computer.

2.2.7 The Quasi-Newton Methods and BFGS update

To avoid the expensive Hessian $D^2 f$ computation, the family of algorithms called *quasi-Newton* methods are devised. The idea goes as follows: If instead of the Hessian $D^2 f(x_k)$ we use an approximation $B_k \approx D^2 f(x_k)$, we can generalize this idea and consider the minimization problem $\min_{x \in \mathbb{R}^n} p(x)$, where $p(x)$ is a quadratic model

$$p(x) = f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2}(x - x_k)^{\mathrm{T}} B_k (x - x_k)$$

of the objective function f. The minimization of $p(x)$ yields the iterate

$$x^* = x_k - B_k^{-1} \nabla f(x_k).$$

This update is well-defined when B_k is nonsingular, and in particular when B_k is positive definite symmetric. Since B_k is only an approximation of $D^2 f(x_k)$, the update $d_k = x^* - x_k$ is used as a search direction rather than an exact update. A line-search then yields a new Quasi-Newton iterate

$$x_{k+1} = x_k + \alpha_k d_k.$$

Suppose we are given a rule for cheaply computing B_{k+1} as a function of the previously computed gradients $\nabla f(x_0), \ldots, \nabla f(x_{k+1})$, or as a function of the previously computed quantities $B_k, \nabla f(x_k)$ and $\nabla f(x_{k+1})$. Then a generic quasi Newton algorithm proceeds as follows:

To turn the generic procedure outlined in Algorithm 2.2.7 into a practical algorithm, we have to specify how to update the approximate Hessian B_k. The most widely used optimization algorithm in unconstrained optimization is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [25]. Maintaining positive definiteness comes at the slight price of requiring rank 2 updates: B_{k+1} is of the form

$$B_{k+1} = B_k + uu^{\mathrm{T}} + vv^{\mathrm{T}},$$

Algorithm 3 Generic Quasi-Newton Method

S0 Choose a starting point $x_0 \in \mathbb{R}^n$, a nonsingular $B_0 \in S^n$ (often the choice is $B_0 = I$), and a termination tolerance $\epsilon > 0$. Set $k = 0$.

S1 If $\|\nabla f(x_k)\| \leq \epsilon$ then stop and output x_k as an approximate local minimizer of f. Else go to **S2**.

S2 Compute the quasi-Newton search direction $d_k = -B_k^{-1}\nabla f(x_k)$.

S3 Perform a practical line-search for the minimization of $\phi(\alpha) = f(x_k + \alpha d_k)$: find a step length α_k that satisfies the Wolfe conditions and compute the new iterate $x_{k+1} = x_k + \alpha_k d_k$.

S4 Compute the new approximate Hessian B_{k+1} according to the specified rule.

S5 Replace k by $k + 1$ and go to **S1**.

that is, the update consists of a sum of two symmetric rank 1 matrices, and such matrices are of rank 2 if u and v are linearly independent.

BFGS updates are described by the relation

$$B_{k+1} = B_k - \frac{B_k \delta_k \delta_k^{\mathrm{T}} B_k}{\delta_k^{\mathrm{T}} B_k \delta_k} + \frac{\gamma_k \gamma_k^{\mathrm{T}}}{\gamma_k^{\mathrm{T}} \delta_k}, \tag{2.35}$$

where $\gamma_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and $\delta_k = x_{k+1} - x_k = \alpha_k d_k$.

If d_k was known, computing the update would take $O(n^2)$ operations, but the computation of d_k requires the solution of the linear system $B_k d_k = -\nabla f(x_k)$. Here we achieve a complexity reduction by maintaining B_k as a Cholesky factorization $B_k = L_k L_k^{\mathrm{T}}$. In fact, as a by-product of this approach it is not too hard to find that B_{k+1} inherits the positive definiteness property from B_k.

Suppose B_k is positive definite symmetric and that we know its Cholesky factorization $B_k = L_k L_k^{\mathrm{T}}$. Then solving the linear system $B_k d_k = -\nabla f(x_k)$ is the same as solving the two triangular systems

$$L_k g_k = -\nabla f(x_k), \tag{2.36}$$

$$L_k^{\mathrm{T}} d_k = g_k. \tag{2.37}$$

Triangular systems are great because they can be solved by direct back-substitution.

Thus, if the Cholesky factorization of B_k is available, the computation of d_k incurs an $O(n^2)$ cost and the BFGS method takes $O(n^2)$ work per iteration. The main idea of the BFGS method then becomes to seek a sensible updating rule $L_k \leftarrow L_{k+1}$ for the Cholesky factor and to take $B_{k+1} = L_{k+1} L_{k+1}^{\mathrm{T}}$, which automatically guarantees that B_{k+1} is symmetric positive definite.

In order to avoid cluttering when summarizing the ideas, we write B, L, α d δ and γ for B_k, L_k, α_k, d_k, δ_k and γ_k respectively, and B_+, L_+ etc. instead of B_{k+1}, L_{k+1} and so forth. At first, we will neglect the condition that L is lower triangular and replace it by an arbitrary nonsingular matrix J such that $JJ^{\mathrm{T}} = B$ and $J_+ J_+^{\mathrm{T}} = B_+$.

BFGS will attempt to solve the minimization problem

$$\min_{J_+} \|J_+ - J\|_F \tag{2.38}$$

$$\text{s.t. } J_+ g = \gamma, \tag{2.39}$$

where $\|\cdot\|_F$ is the Frobenius norm and $g \in \mathbb{R}^n$ is a parameter vector. Choose g so that

$$J_+^{\mathrm{T}} \delta = g. \tag{2.40}$$

The result will be that $B_+ = J_+ J_+^{\mathrm{T}}$ satisfies the quasi-Newton equation and that $\|J_+ - J\|_F$, a measure of distance between B_+ and B, is minimized under this condition.

This minimization problem (2.38) is clearly the same as the smooth strictly convex optimization problem as follows

$$\begin{aligned} &\min_{J_+} \operatorname{tr}\big((J_+ - J)(J_+ - J)^{\mathrm{T}}\big) \\ &\text{s.t. } J_+ g = \gamma, \end{aligned} \tag{2.41}$$

where $\operatorname{tr}(A) = \sum_i A_{ii}$ denotes the trace of a square matrix A. This problem can be reformulated as

$$\min_{J_+} \langle J_+ - J, J_+ - J \rangle \tag{2.42}$$

$$\text{s.t. } \langle J_+, e_i g^{\mathrm{T}} \rangle = \gamma_i \quad (i = 1, \ldots, n), \tag{2.43}$$

where e_i is the i-th coordinate vector and

$$\begin{aligned} &\langle \cdot, \cdot \rangle : \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times n} \to \mathbb{R} \\ &\qquad (A, B) \mapsto \langle A, B \rangle := \operatorname{tr}(AB^{\mathrm{T}}) = \sum_{i,j} a_{ij} b_{ij} \end{aligned}$$

is a Euclidean inner product on the vector space $\mathbb{R}^{n \times n}$ of $n \times n$ real matrices. It is easy to check that the Euclidean norm that corresponds to this inner product is the usual Frobenius norm, defined by

$$\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}.$$

Thus, $(\mathbb{R}^{n \times n}, \langle \cdot, \cdot \rangle)$ is a Euclidean space of dimension n^2. The problem (2.42) is then the task of finding the point closest to J in the affine subspace defined by the constraints (2.43). The minimum is achieved at the unique point J_+^* where $(J_+^* - J)$ is orthogonal to the affine subspace, that is, $(J_+^* - J) \in \operatorname{span}(e_1 g^{\mathrm{T}}, \ldots, e_n g^{\mathrm{T}})$.

Condition (2.40) now becomes

$$J^{\mathrm{T}}\delta = \left(1 - \frac{(\gamma - Jg)^{\mathrm{T}}}{g^{\mathrm{T}}g}\delta\right) g, \tag{2.44}$$

and since $J^{\mathrm{T}}\delta \neq 0$, this can only be satisfied for $g = \beta J^{\mathrm{T}}\delta$ with $\beta \neq 0$. Substituting into (2.44), we obtain

$$1 = \left(1 - \frac{\beta(\gamma^{\mathrm{T}}\delta - \alpha\delta^{\mathrm{T}}B\delta)}{\beta^2\delta^{\mathrm{T}}B\delta}\right)\beta,$$

or

$$\beta = \pm\sqrt{\frac{\gamma^{\mathrm{T}}\delta}{\delta^{\mathrm{T}}B\delta}}. \tag{2.45}$$

Note that if α satisfies the Wolfe conditions of Lecture 2, then

$$\gamma^{\mathrm{T}}\delta = \big(\nabla f(x + \alpha d) - \nabla f(x)\big)^{\mathrm{T}}\alpha d \geq (c_2 - 1)\phi'(0) > 0.$$

Therefore, the square-root in (2.45) is real.

Expressing the update of J in terms of β, we find

$$J_+ = J + \frac{(\gamma - \beta B\delta)\delta^{\mathrm{T}}J}{\beta\delta^{\mathrm{T}}B\delta}. \tag{2.46}$$

The corresponding update of B is

$$\begin{aligned} B_+ &= J_+J_+^{\mathrm{T}} \\ &= JJ^{\mathrm{T}} + \frac{(\gamma - \beta B\delta)\delta^{\mathrm{T}}B}{\beta\delta^{\mathrm{T}}B\delta} + \frac{B\delta(\gamma - \beta B\delta)^{\mathrm{T}}}{\beta\delta^{\mathrm{T}}B\delta} + \frac{(\gamma - \beta B\delta)(\gamma - \beta B\delta)^{\mathrm{T}}}{\beta^2\delta^{\mathrm{T}}B\delta} \\ &= \dots \\ &= B - \frac{B\delta\delta^{\mathrm{T}}B}{\delta^{\mathrm{T}}B\delta} + \frac{\gamma\gamma^{\mathrm{T}}}{\beta^2\delta^{\mathrm{T}}B\delta}. \end{aligned}$$

Using (2.45), this yields the BFGS formula (2.35):

$$B_+ = B - \frac{B\delta\delta^{\mathrm{T}}B}{\delta^{\mathrm{T}}B\delta} + \frac{\gamma\gamma^{\mathrm{T}}}{\gamma^{\mathrm{T}}\delta}.$$

Note that this formula is independent of the sign of β, and more importantly of J, that is to say, we could have chosen *any* factorization $B = JJ^{\mathrm{T}}$ of B into a nonsingular matrix J and its transpose and we would have ended up with the same update for B. This property makes it possible to choose $J = L$, where L is the Cholesky factor of B. With this choice the transpose of Equation (2.46) becomes

$$J_+^{\mathrm{T}} = L^{\mathrm{T}} + \frac{L^{\mathrm{T}}\delta(\gamma - \beta B\delta)^{\mathrm{T}}}{\beta\delta^{\mathrm{T}}B\delta}.$$

This is an upper triangular matrix plus a rank 1 update.

Next we have the following lemma:

Lemma 2.2.1. *Let $X \in \mathbb{R}^{n\times n}$ be a square matrix. Then there exist unique matrices $Q, R \in \mathbb{R}^{n\times n}$ such that Q is orthogonal (that is, $Q^{\mathrm{T}}Q = \mathrm{I}$) and R is upper triangular (that is $R_{ij} = 0$ if $i > j$) and with nonnegative diagonal entries (that is, $R_{ii} \geq 0$ for all i), and such that $X = QR$.*

Using the triangular structure of J_+^{T}, it is easy to show that the QR factorization of $J_+^{\mathrm{T}} = Q_+R_+$ can be computed in $O(n^2)$ operations. But then we have

$$B_+ = J_+J_+^{\mathrm{T}} = R_+^{\mathrm{T}}Q_+^{\mathrm{T}}Q_+R_+ = R_+^{\mathrm{T}}R_+,$$

and this shows that $L_+ := R_+^{\mathrm{T}}$ is the Cholesky factor of B_+.

Having derived the BFGS update and a rule for computing the updates of the Cholesky factor of the approximate Hessian B_k, we can now formulate the BFGS algorithm with all the pieces put together:

Algorithm 4 BFGS method

S0 Choose a starting point x_0 and a lower triangular matrix L_0 with positive diagonal entries (the usual choice is $L_0 = I$, if no other information about the problem is known). Set a termination tolerance $\epsilon > 0$. Set $k = 0$.

S1 If $\|\nabla f(x_k)\| \leq \epsilon$ then stop and output x_k as an approximate local minimizer.

S2 Otherwise solve the triangular system $L_k g_k = -\nabla f(x_k)$ for g_k, and then the triangular system $L_k^{\mathrm{T}} d_k = g_k$ for d_k.

S3 Perform a line search to find a positive step length $\alpha_k > 0$ such that $f(x_k + \alpha_k d_k) < f(x_k)$, and such that α_k satisfies the Wolfe conditions.

S4 Set $\delta_k := \alpha_k d_k$, $x_{k+1} = x_k + \delta_k$. Compute $\gamma_k := \nabla f(x_{k+1}) - \nabla f(x_k)$ and $\beta_k = \pm\sqrt{\gamma_k^{\mathrm{T}} \delta_k / \delta_k^{\mathrm{T}} B_k \delta_k}$.

S5 Compute

$$J_{k+1}^{\mathrm{T}} = L_k^{\mathrm{T}} + \frac{L_k^{\mathrm{T}} \delta_k (\gamma_k - \beta_k B_k \delta_k)^{\mathrm{T}}}{\beta_k \delta_k^{\mathrm{T}} B \delta_k},$$

and then compute the QR factorization $J_{k+1}^{\mathrm{T}} = Q_{k+1} R_{k+1}$. Set $L_{k+1} := R_{k+1}^{\mathrm{T}}$ and return to **S1**.

The BFGS algorithm spends $O(n^2)$ computation time per iteration, yet its convergence behavior is not too much slower than that of the Newton–Raphson method. With additional conditions, it can be shown that the BFGS method has local Q-superlinear convergence (but not Q-quadratic). It can also be shown that if the BFGS algorithm is used on a strictly convex quadratic function and in conjunction with exact line searches, then B_k becomes the exact (constant) Hessian after n iterations.

However, it also has drawbacks. It needs to keep a $n \times n$ matrix H_k (the inverse of the approximate Hessian B_k) or L_k (the Cholesky factor of B_k) in the computer memory. In other words, there is an $O(n^2)$ *memory requirement* for quasi-Newton methods. In comparison, the steepest descent method only occupies $O(n)$ memory at any given time: after x_{k+1} has

been computed, the registers that were previously used to store the n components of $\nabla f(x_k)$ can be overwritten with the data $\nabla f(x_{k+1})$, then x_{k+1} can be overwritten with x_{k+2} etc. In situations where n is very large, the steepest descent method can therefore still cope with keeping all the necessary data in the main memory when quasi-Newton methods have long ceased to be effective and spend most of their time in data transfers.

2.2.8 Trust Region Methods

Apart from line-search methods, *Trust region methods* constitute a second fundamental class of algorithms. In this approach (2.32) is again replaced by a sequence of easier problems, but instead of reducing the problem dimension, the simplicity is achieved by replacing f with a degree 2 polynomial (conventional). Conceptually, the idea can be described as follows:

- In iteration k, replace $f(x)$ by a locally valid quadratic model function $m_k(x)$:

$$m_k(x) = f(x_k) + \nabla f(x_k)^{\mathrm{T}}(x - x_k) + \frac{1}{2}(x - x_k)^{\mathrm{T}} B_k (x - x_k) \tag{2.47}$$

- Choose a neighborhood R_k of the current iterate x_k in which $m_k(x)$ can be trusted to approximate f well (we do not care about how well m_k approximates f outside R_k).

- The next iterate x_{k+1} is found by approximately minimising the model function over the trust region,

$$x_{k+1} \approx \arg\min_{x \in R_k} m_k(x). \tag{2.48}$$

The linear part of (2.47) coincides with the first order Taylor approximation of f around x_k, so that $m_k(x)$ will be a good local approximation of $f(x)$ if $B_k \approx D^2 f(x_k)$. To make the method work, one thus has to worry about how to update B_k cheaply.

Let y_{k+1} be the approximate minimizer of the trust region subproblem (2.48). In principle, this is the point we would like to select as our next iterate x_{k+1}. However, y_{k+1} is computed

on the basis of the model function m_k, and it could happen that moving to y_{k+1} leads to an increase rather than decrease in of the *true* objective function f. Trust-region methods therefore accept y_{k+1} only if the decrease achieved in f is at least a fixed proportion of the decrease "predicted" by m_k,

$$x_{k+1} = \begin{cases} y_{k+1}, \text{ if } \frac{f(x_k)-f(y_{k+1})}{m_k(x_k)-m_k(y_{k+1})} > \rho, \\ x_k, \text{ otherwise,} \end{cases} \tag{2.49}$$

where $\rho \in (0, 1/4)$ is fixed. Note that rejecting the update does not imply that the algorithm will stall, because we can still shrink the trust region so that $y_{k+2} \neq y_{k+1}$.

The easiest way to define a trust region R_k is to choose the closed ball of radius Δ_k around x_k in some norm $\|\cdot\|$,

$$R_k = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}.$$

For simplicity, we will assume that $\|\cdot\|$ is the Euclidean norm. Δ_k is called the *trust region radius*.

In order to define a new trust region R_{k+1} around x_{k+1}, it suffices to fix a rule on how to select Δ_{k+1}. The following rule is a popular choice.

$$\Delta_{k+1} = \begin{cases} \frac{\Delta_k}{2} \text{ if } \frac{f(x_k)-f(y_{k+1})}{m_k(x_k)-m_k(y_{k+1})} < \frac{1}{4}, \\ \min(2\Delta_k, \Delta_{\max}) \text{ if } \frac{f(x_k)-f(y_{k+1})}{m_k(x_k)-m_k(y_{k+1})} > \frac{3}{4}, \\ \Delta_k \text{ otherwise.} \end{cases} \tag{2.50}$$

The rule is designed so that Δ_k never exceeds $\Delta_{\max}$, and it is motivated by comparing the objective function decrease $f(x_k) - f(y_{k+1})$ with the decrease $m_k(x_k) - m_k(y_{k+1})$ "predicted" by the model function:

- If the actual decrease was below our expectations, this indicates that m_k should be regarded as a more local model than before. We thus find a reasonable Δ_{k+1} by shrinking Δ_k.

- If the actual decrease was above our expectations, we feel confident to expand the trust region by selecting Δ_{k+1} as an expansion of Δ_k.

- If there is neither reason for expanding or shrinking, we stick to the previous value $\Delta_{k+1} = \Delta_k$.

By now we assembled the necessary elements to formulate a generic trust region algorithm:

Algorithm 5 Generic Trust region Method

S0 Choose $\Delta_{\max} > 0$, $\Delta_0 \in (0, \Delta_{\max})$, $\eta \in (0, 1/4)$, $x_0 \in \mathbb{R}^n$, B_0, $\epsilon > 0$.

S1 While $\|\nabla f(x_k)\| \geq \epsilon$ repeat

Compute y_{k+1} as the approximate minimizer of (2.48).
Determine x_{k+1} via (2.49).
Compute Δ_{k+1} using (2.50).
Build a new model function $m_{k+1}(x)$.
$k \leftarrow k + 1$.

end

S2 Return x_k.

In step **S1** of the algorithm, the approximate minimizer y_{k+1} can be computed in many different ways. For them to work out though, we need to assume that the method chosen

for computing y_{k+1} compares favorably to a specific benchmark, the so-called *Cauchy point*. This point is obtained when a steepest descent line-search is applied to m_k at x_k and is restricted to R_k.

An unrestricted line-search in the direction $-\nabla f(x_k)$ yields the step-length multiplier

$$\begin{aligned}\alpha_k^u &:= \arg\min_{\alpha\geq 0} m_k(x_k - \alpha\nabla f(x_k)) \\ &= \arg\min_{\alpha\geq 0}\ f(x_k) - \alpha\nabla f(x_k)^{\mathrm{T}}\nabla f(x_k) + \frac{\alpha^2}{2}\nabla f(x_k)^{\mathrm{T}} B_k \nabla f(x_k) \\ &= \begin{cases} +\infty \text{ if } \nabla f(x_k)^{\mathrm{T}} B_k \nabla f(x_k) \leq 0, \\ \frac{\nabla f(x_k)^{\mathrm{T}}\nabla f(x_k)}{\nabla f(x_k)^{\mathrm{T}} B_k \nabla f(x_k)} \text{ otherwise.} \end{cases}\end{aligned}$$

If we want to stay within R_k we have to "clip" α_k^u to a constrained step-length multiplier α_k^c. Note that $\alpha \mapsto m_k(x_k - \alpha\nabla f(x_k))$ is strictly decreasing on $[0, \alpha_k^u)$. Moreover, the radius $\|x_k - \alpha\nabla f(x_k)\|$ is strictly increasing over the same interval. Therefore, the correct clipping rule is given by

$$\alpha_k^c = \min\left(\frac{\Delta_k}{\|\nabla f(x_k)\|}, \alpha_k^u\right) \tag{2.51}$$

and $y_k^c := x_k - \alpha_k^c \nabla f(x_k)$ is the Cauchy point of the trust region subproblem (2.48).

Using the concept of *Cauchy point*, one may indeed obtain a global convergence result for the Trust region method.

Theorem 2.2.5. *Let Algorithm 2.2.8 be applied to the minimization of $f \in C^2(\mathbb{R}^n, \mathbb{R})$, and for all k let y_{k+1} be computed such that $m_k(y_{k+1}) \leq m_k(y_k^c)$ holds. Let there exist $\beta > 0$ such that for all k, $\|B_k\|, \|D^2 f(x_k)\| \leq \beta$, and finally, let $\Delta_0 \geq \epsilon/(14\beta)$. Then exactly one of two following alternatives occurs:*

(i) The algorithm does not terminate, but $\lim_{k\to\infty} f(x_k) = -\infty$ and f is unbounded below.

(ii) The algorithm terminates in finite time, returning an approximate minimizer.

The proof will be omitted here for simplicity. The reader is referred to [19] if interested.

2.2.9 Constrained Optimization and the KKT Conditions

In this section we will consider the problem of minimizing objective functions over constrained domains. The general nonlinear programming problem of this kind can be written as

$$\begin{aligned} \text{(NLP)} \quad & \min_{x \in \mathbb{R}^n} f(x) \\ & \text{s.t.} \quad g_i(x) = 0 \quad (i \in \mathcal{E}), \\ & \qquad g_j(x) \geq 0 \quad (i \in \mathcal{I}), \end{aligned}$$

where $\mathcal{E}$ and $\mathcal{I}$ are the finite index sets corresponding to the equality and inequality constraints, and where $f, g_i \in C^k(\mathbb{R}^n, \mathbb{R})$ for all $(i \in \mathcal{I} \cup \mathcal{E})$.

In unconstrained optimization we found that we can use the optimality conditions to transform optimization problems into zero-finding problems for systems of nonlinear equations. A similar approach can be developed for constrained optimization: in this case the optimal solutions can be characterized by systems of nonlinear equations and inequalities.

A natural by-product of this analysis will be the notion of a Lagrangian dual of an optimization problem: every optimization problem - called the primal - has a sister problem in the space of Lagrange multipliers - called the dual. In constrained optimization it is often advantageous to think of the primal and dual in a combined primal-dual framework where each sheds light from a different angle on a certain saddle-point finding problem.

Theorem 2.2.6 (Farkas' Lemma)**.**
Let $a_1, \ldots, a_m, b \in \mathbb{R}^n$ be a set of vectors. Then exactly one of the two following alternatives occurs:

(i) $\exists y \in \mathbb{R}^m_+$ *such that* $b = \sum_i^m y_i a_i$.

(ii) $\exists d \in \mathbb{R}^n$ *such that* $d^{\mathrm{T}}b < 0$ *and* $d^{\mathrm{T}}a_i \geq 0$ *for all* $(i = 1, \ldots, m)$.

Note that Alternative (i) says that b lies in the convex cone generated by the vectors a_i:

$$b \in \text{cone}(a_1, \ldots, a_m) := \left\{ \sum_{i=1}^{n} \lambda_i a_i : \lambda_i \geq 0 \; \forall i \right\}.$$

Alternative (ii) on the other hand says that the hyperplane $d^{\perp} := \{x \in \mathbb{R}^n : d^{\mathrm{T}}x = 0\}$ strictly separates b from the convex set $\text{cone}(a_1, \ldots, a_m)$. Thus, Theorem 2.2.6 is a result about convex separation: either b is a member of $\text{cone}(a_1, \ldots, a_m)$ or there exists a hyperplane that strictly separates the two objects. See Figure 2.4 for an illustration of the two cases.

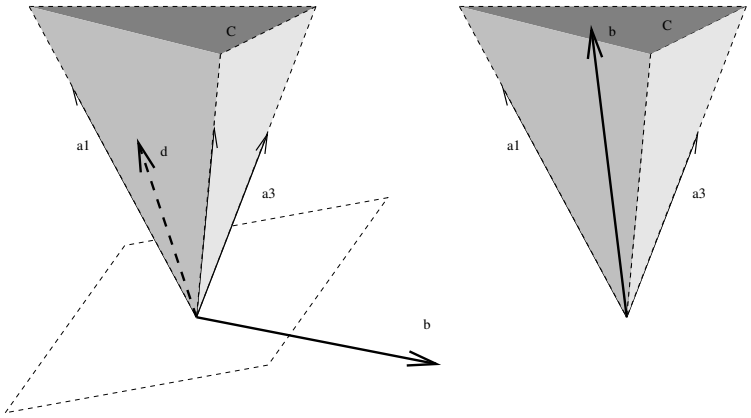


Figure 2.4: *Illustration of the two alternative situations described by Theorem 2.2.6, where* $C = \text{cone}(a_1, \ldots, a_m)$. *On the left, Alternative (ii) is depicted, and on the right, Alternative (i).*

Now for the optimality condition, a useful picture is to imagine a point mass m moves

on a hard smooth surface

$$G := \big\{(x, z) \in \mathbb{R}^n \times \mathbb{R} : \, g(x) = 0\big\}$$

which is parallel to the z-axis everywhere and keeps the point mass from rolling into the domain where $g(x) < 0$. Such a surface can exert only a normal force that points towards the domain $\{x : \, g_j(x) > 0\}$. Therefore, the reaction force must be of the form $\vec{N}_g = \mu_g \big(\begin{smallmatrix} \nabla g(x) \\ 0 \end{smallmatrix} \big)$, where $\mu_g \geq 0$. In the case depicted in Figure 2.5 where there is only one inequality constraint,

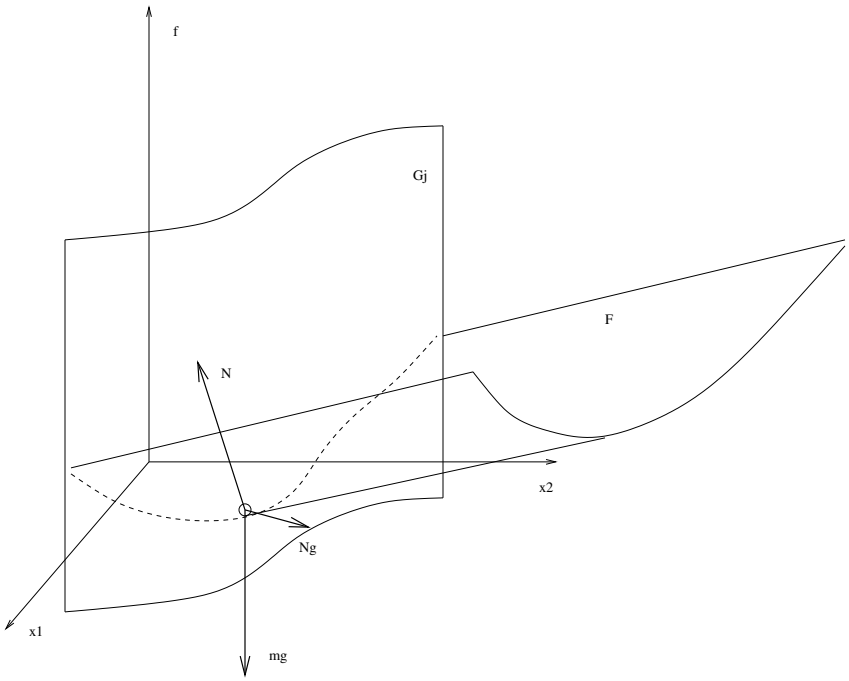


Figure 2.5: *Mechanistic interpretation of constrained first order optimality conditions: the sum of external forces has to be zero.*

the point mass is at rest and does not roll to lower terrain if the sum of external forces is

zero, that is, $\vec{N}_f + \vec{N}_g + m\vec{g} = 0$. Since $\vec{N}_f = \mu_f \binom{-\nabla f(x)}{1}$ for some $\mu_f \geq 0$, we find

$$\mu_f \begin{bmatrix} -\nabla f(x) \\ 1 \end{bmatrix} + \mu_g \begin{bmatrix} \nabla g(x) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -mg \end{bmatrix} = 0,$$

from where it follows that $\mu_f = mg$ and

$$\nabla f(x) = \lambda \nabla g(x) \tag{2.52}$$

with $\lambda = \mu/mg \geq 0$. In other words, $\vec{N}_f$ is determined by the condition that its vertical component counter-balances the force of gravity, and $\vec{N}_g$ by the condition that it counter-balances the horizontal component of $\vec{N}_f$. This second condition is expressed in the balance equation (2.52).

When multiple inequality constraints are present, then the horizontal component of $\vec{N}_f$ must be counter-balanced by the *sum* of the reaction forces exerted by the constraint manifolds that touch the test mass. The balance equation (2.52) must thus be replaced with

$$\nabla f(x) = \sum_{j \in \mathcal{I}} \lambda_j \nabla g_j(x)$$

for some $\lambda_j \geq 0$, and since constraints for which $g_j(x) > 0$ cannot exert a force on the test mass, we must set $\lambda_j > 0$ for these indices, or equivalently, the equation $\lambda_j g_j(x) = 0$ must hold for all $j \in \mathcal{I}$.

It remains to discuss the influence of equality constraints when they are present. Replacing $g_i(x) = 0$ by the two inequality constraints $g_i(x) \geq 0$ and $-g_i(x) \geq 0$, the mechanistic interpretation yields two parallel surfaces G_i^+ and G_i^-, leaving an infinitesimally thin space between them within which the point mass is constrained to move, see Figure 2.6. The net reaction force of the two surfaces is of the form

$$\lambda_i^+ \nabla g_i(x) + \lambda_i^- \nabla(-g_i)(x) = \lambda_i \nabla g_i(x),$$

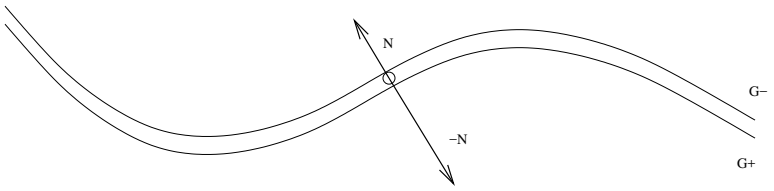


Figure 2.6: *Mechanistic interpretation of an equality constraint: the net reaction force can point to either side of G_i. Here, we are looking down the z-axis.*

where we replaced the difference $\lambda_i^+ - \lambda_i^-$ of the bound-constrained variables $\lambda_i^+, \lambda_i^- \geq 0$ by a single unconstrained variable $\lambda_i = \lambda_i^+ - \lambda_i^-$. Note that in this case the conditions $\lambda_i^+ g_i(x) = 0$, $\lambda_i^-(-g_i(x)) = 0$ are satisfied automatically, since $g_i(x) = 0$ if x is feasible.

In summary, the mechanistic motivation suggests that if x is a local minimizer of (NLP), then there exist *Lagrange multipliers* $\lambda \in \mathbb{R}^{|\mathcal{I}\cup\mathcal{E}|}$ such that

$$\begin{aligned} \nabla f(x) - \sum_{i\in\mathcal{I}\cup\mathcal{E}} \lambda_i \nabla g_i(x) &= 0 \\ g_i(x) &= 0 \qquad (i \in \mathcal{E}) \\ g_j(x) &\geq 0 \qquad (j \in \mathcal{I}) \\ \lambda_j g_j(x) &= 0 \qquad (j \in \mathcal{I}) \\ \lambda_j &\geq 0 \qquad (j \in \mathcal{I}). \end{aligned}$$

These are the so-called Karush-Kuhn-Tucker (KKT) conditions. The above intuitive motivation cannot replace a rigorous proof, but the physical interpretation provides an easy explanation and a jog for memory.

So far, the fundation has been laid for the presentation of the author's work. In the following chapters, we will revisit many of the concepts mention in this chapter.

Chapter 3

SUCCESSIVE CONVEXIFICATION FOR NON-CONVEX OPTIMAL CONTROL

Successive Convexification is a family of iterative algorithms that solves non-convex optimal control problems. Depending on what the sources of non-convexity are and what the system dynamics look like, one can choose to a certain variant of the algorithm that is better suited for the problem. In this section, we will first focus on the main and general-purpose algorithm, `SCvx`. It can handle complex nonlinear system dynamics and non-convex state constraints that have minimal assumptions placed on them. Therefore, it is the go-to choice for many types of optimal control problems, from spacecraft guidance to optimal power flow.

3.1 Problem Statement

A continuous-time optimal control problem has to be discretized before it can be solved using optimization methods on a digital computer [36]. One may use, for instance, Gauss collocation method [31] or pseudospectral method [27] to achieve that. Therefore in this section, it is sufficient for us to consider just the discrete-time finite-horizon optimal control problem as in Problem 3.1.1.

Problem 3.1.1. Non-Convex Optimal Control Problem

$$\min_{u} \quad C(x,u) \coloneqq \sum_{i=1}^{N} \phi(x_i, u_i), \tag{3.1a}$$

subject to:

$$x_{i+1} = f(x_i, u_i), \qquad i = 1, 2, \ldots, N-1, \tag{3.1b}$$

$$s(x_i, u_i) \leq 0, \qquad i = 1, 2, \ldots, N, \tag{3.1c}$$

$$x_i \in X_i \subseteq \mathbb{R}^{n_x}, \qquad i = 1, 2, \ldots, N, \tag{3.1d}$$

$$u_i \in U_i \subseteq \mathbb{R}^{n_u}, \qquad i = 1, 2, \ldots, N-1. \tag{3.1e}$$

In Problem 3.1.1, x_i and u_i represent the state and control vectors at the i^{th} discrete time step, X_i and U_i are assumed to be convex and compact sets which include the boundary conditions, and N denotes the total number of time steps. For simplicity, and without loss of generality, we assume that N is fixed (i.e. fixed final time; see [73] for a treatment of free-final-time problems). More concisely, these variables can be written as

$$x \coloneqq [x_1^T, x_2^T, \ldots, x_N^T]^T \in X \subseteq \mathbb{R}^{n_x N}$$

$$u \coloneqq [u_1^T, u_2^T, \ldots, u_{N-1}^T]^T \in U \subseteq \mathbb{R}^{n_u (N-1)},$$

where X is the Cartesian product of X_i, and U is the Cartesian product of U_i. We assume that the function $\phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ in (3.1a) is convex and continuously differentiable. This is a reasonable assumption for many real-world optimal control problems. For example, the minimum-fuel problem has $\phi(x_i, u_i) = \|u_i\|_2$. The system dynamics are represented by (3.1b), where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is assumed to be a continuously differentiable nonlinear function. The state constraints are represented by 3.1c, where $s : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_s}$ is assumed to be a continuously differentiable non-convex function. Often, lossless convexification can be leveraged to handle the non-convex control constraints (see e.g. [3, 4]). This should be done whenever possible. In summary, the non-convexity of Problem 3.1.1 is due to (3.1b) and (3.1c).

3.2 Algorithm Description

The basic operating principle of the `SCvx` algorithm involves linearizing the non-convex parts of Problem 3.1.1 about the solution of the k^{th} iteration. This results in a convex subproblem that is solved to full optimality (which makes this different than standard trust-region based methods), resulting in a new solution for the $(k+1)^{th}$ iteration. This process is repeated in succession until convergence is achieved. In general, the solution to the convex subproblem will not be optimal with respect to the original non-convex problem. To recover optimality, the algorithm must eventually converge to a solution that satisfies the first-order optimality conditions of Problem 3.1.1.

To achieve this, we begin by denoting the solution to the k^{th} iteration as (x^k, u^k). At each time step i, let $A_i^k \coloneqq \dfrac{\partial}{\partial x_i} f(x_i, u_i)\Big|_{x_i^k, u_i^k}$, $B_i^k \coloneqq \dfrac{\partial}{\partial u_i} f(x_i, u_i)\Big|_{x_i^k, u_i^k}$, $S_i^k \coloneqq \dfrac{\partial}{\partial x_i} s(x_i, u_i)\Big|_{x_i^k, u_i^k}$, $Q_i^k \coloneqq \dfrac{\partial}{\partial u_i} s(x_i, u_i)\Big|_{x_i^k, u_i^k}$, and define $d \coloneqq x - x^k$, $d_i \coloneqq x_i - x_i^k$, $w \coloneqq u - u^k$, and $w_i \coloneqq u_i - u_i^k$ in terms of the solution to the current iteration, (x, u). At the i^{th} time step, the first-order approximation of (3.1b) and (3.1c) about (x_i^k, u_i^k) is given by

$$x_{i+1}^k + d_{i+1} = f(x_i^k, u_i^k) + A_i^k d_i + B_i^k w_i, \tag{3.2a}$$

$$s(x_i^k, u_i^k) + S_i^k d_i + Q_i^k w_i \leq 0, \tag{3.2b}$$

which is a linear system with respect to the incremental state and control variables, d_i and w_i, of the convex subproblem. The linearization procedure affords the benefit of convexity, but introduces two issues that obstruct the convergence process: *artificial infeasibility* and *artificial unboundedness*.

Artificial infeasibility happens when a feasible non-convex problem is linearized about a point (x, u) and it results in an infeasible convex subproblem. This undesirable phenomenon is often encountered during the early iterations of the process, and to mitigate its effects from the linearization of nonlinear dynamics, we augment the linearized dynamics in (3.2a)

with an unconstrained *virtual control* term, $v_i \in \mathbb{R}^{n_v}$:

$$x_{i+1}^k + d_{i+1} = f(x_i^k, u_i^k) + A_i^k d_i + B_i^k w_i + E_i^k v_i, \tag{3.3}$$

where $E_i^k \in \mathbb{R}^{n_x \times n_v}$ is selected such that $\text{im}(E_i^k) = \mathbb{R}^{n_x}$, and thus guaranteeing one-step controllability. Since v_i is left unconstrained, any state in the feasible region of the convex subproblem is reachable in finite time. For example, for a mass-spring-damper system, the virtual control can be interpreted as a synthetic force to ensure feasibility with respect to state and control constraints. Consequently, the resulting augmented convex subproblem is no longer vulnerable to artificial infeasibility arising from the linearization of (3.1b). While the virtual control term makes any state reachable in finite time, it does not allow the problem to retain feasibility when the state and control constraints in (3.2b) define an empty feasible set (e.g. consider the union of the state constraints $[1\,,\,0,\ldots,\,0]x \geq \epsilon$ and $[1\,,\,0,\ldots,\,0]x \leq -\epsilon$, for $\epsilon > 0$).

To mitigate artificial infeasibility caused by the linearization of non-convex state and control constraints, we introduce an unconstrained *virtual buffer zone* term, $s'_i \in \mathbb{R}_+^{n_s}$ (meaning $s'_i \geq 0$), to (3.1c):

$$s(x_i^k, u_i^k) + S_i^k d_i + Q_i^k w_i - s'_i \leq 0. \tag{3.4}$$

s'_i can be understood as a relaxation term that allows the non-convex state and control constraints in (3.2b) to be violated. In an obstacle avoidance example, s'_i can be interpreted as a measure of the obstacle constraint violation necessary to retain feasibility at the i^{th} time step.

To ensure that v_i and s'_i are used only when necessary, we define

$$v \coloneqq [v_1^T, v_2^T, \ldots, v_{N-1}^T]^T \in \mathbb{R}^{n_v(N-1)}$$
$$s' \coloneqq [s'^T_1, s'^T_2, \ldots, s'^T_{N-1}]^T \in \mathbb{R}_+^{n_s(N-1)},$$

and augment the linear cost function with the term $\sum_{i=1}^{N-1} \lambda_i P(E_i v_i, s'_i)$, where the λ_i's are

sufficiently large positive penalty weights, and $P : \mathbb{R}^{n_x} \times \mathbb{R}^{n_s} \to \mathbb{R}$ is an exact penalty function (which will be more precisely defined later in (3.12)). Thus, to obtain the solution for the $(k+1)^{th}$ iteration, we use the penalized cost given by

$$L^k(d, w) \coloneqq C(x^k + d, u^k + w) + \sum_{i=1}^{N-1} \lambda_i P(E_i^k v_i, s'_i). \tag{3.5}$$

Note that we omitted the dependence of v_i and s'_i in L^k for simplicity. Its corresponding nonlinear penalized cost is given by

$$J(x, u) \coloneqq C(x, u) + \sum_{i=1}^{N-1} \lambda_i P\big(x_{i+1} - f(x_i, u_i), s(x_i, u_i)\big). \tag{3.6}$$

The second adverse effect of linearization is *artificial unboundedness*. This occurs when the local properties of the non-convex problems are extrapolated well beyond the neighborhood of the linearization point. For example, consider the following simple non-convex optimization problem: $\{\min x_2 \,|\, x_2 = x_1^2\}$, whose solution is $(x_1^*, x_2^*) = (0, 0)$. Linearizing this non-convex problem about any point other than (x_1^*, x_2^*) renders the linearized problem unbounded. To mitigate the risk of artificial unboundedness, we impose trust-region constraint $\|u - u^k\| \leq r^k$, i.e., $\|w\| \leq r^k$ to ensure that u does not deviate significantly from the previous control input u^k. By selecting r^k appropriately, this constraint, in conjunction with (3.3), ensures that x remains sufficiently close to the state vector obtained in the previous iteration, x^k, thus keeping the solution within the region where the linearization is accurate.

We are now able to present the final problem formulation and a summary of the `SCvx` algorithm. At the $(k+1)^{th}$ iteration, we solve the convex optimal control subproblem, Problem 3.2.1. For many applications, U and X are simple convex sets, e.g. second order cones, in which case Problem 3.2.1 can be readily solved by SOCP solvers (e.g. [24]) in a matter of milliseconds.

Problem 3.2.1. Convex Optimal Control Subproblem

$$\min_{d,w} L^k(d,w),$$

$$\text{subject to:} \quad u^k + w \in U, \quad x^k + d \in X, \quad \|w\| \leq r^k.$$

The `SCvx` algorithm solves Problem 3.1.1 according to the steps outlined in Algorithm 6. It follows standard trust-region radius updating rules, but differs from conventional trust-region methods in some significant ways, as discussed later in Section 3.2.

In the following we will discuss some of the algorithm details. In line 2, we initialize the algorithm by using random initial state x^1 and control u^1, which need **not** to be dynamically feasible. This is important because finding a feasible solution itself is a difficult non-convex problem, and `SCvx` fortunately does not require that. There are no strict rules to following in choosing parameters. Generally speaking, ρ_0 is very close to 0, ρ_1 is slightly greater than 0, and ρ_2 can be marginally less than 1. r_l can be slightly greater than 0.

The quality of the linear approximation used in Problem 3.2.1 can be understood by inspecting the ratio ρ^k in (3.9), which compares the realized (nonlinear) cost reduction ΔJ^k in (3.7), to the predicted (linear) cost reduction ΔL^k in (3.8) based on the previous (i.e. k^{th}) iteration. In line 13, when ρ_k is small (i.e. when $\rho^k < \rho_0 \ll 1$), the approximation is considered inaccurate, since the linear cost reduction over-predicts the realized nonlinear cost reduction. Consequently, the solution (d, w) is rejected, the trust region is contracted by a factor of $\alpha < 1$, and the step is repeated. The contraction of the trust region ensures that this condition can occur only a finite number of times, thus guaranteeing that the algorithm will not remain in this state indefinitely.

In line 15, if ρ^k is such that the linearization accuracy is deficient, yet acceptable (i.e. when $\rho_0 \leq \rho^k < \rho_1$), then the solution (d, w) is accepted, but the trust region is contracted. The former is done to avoid unnecessarily discarding the solution that has already been computed, while the latter is done to improve the linearization accuracy of the next succession.

Algorithm 6 The `SCvx` Algorithm

procedure `SCvx`$(x^1, u^1, \lambda, \epsilon_{tol})$

input Select initial state $x^1 \in X$ and control $u^1 \in U$. Initialize trust region radius $r^1 > 0$. Select penalty weight $\lambda > 0$, and parameters $0 < \rho_0 < \rho_1 < \rho_2 < 1$, $r_l > 0$ and $\alpha > 1, \beta > 1$.

while not converged, i.e. $\Delta J^k > \epsilon_{tol}$ **do**

step 1 At $(k+1)^{th}$ succession, solve Problem 3.2.1 at (x^k, u^k, r^k) to get an optimal solution (d, w).

step 2 Compute the *actual* change in the penalty cost (3.6):

$$\Delta J^k = J(x^k, u^k) - J(x^k + d, u^k + w), \tag{3.7}$$

and the *predicted* change by the convex cost (3.5):

$$\Delta L^k = J(x^k, u^k) - L^k(d, w). \tag{3.8}$$

if $\Delta J^k = 0$ **then**

stop, and **return** (x^k, u^k);

else

compute the ratio

$$\rho^k := \Delta J^k / \Delta L^k. \tag{3.9}$$

end if

step 3

if $\rho^k < \rho_0$ **then**

reject this step, contract the trust region radius, i.e. $r^k \leftarrow r^k/\alpha$ and go back to **step 1**;

else

accept this step, i.e. $x^{k+1} \leftarrow x^k + d$, $u^{k+1} \leftarrow u^k + w$, and update the trust region radius r^{k+1} by

$$r^{k+1} = \begin{cases} r^k/\alpha, & \text{if } \rho^k < \rho_1; \\ r^k, & \text{if } \rho_1 \le \rho^k < \rho_2; \\ \beta r^k, & \text{if } \rho_2 \le \rho^k. \end{cases}$$

end if

$r^{k+1} \leftarrow \max\{r^{k+1}, r_l\}$, $k \leftarrow k + 1$, and go back to **step 1**.

end while

return (x^{k+1}, u^{k+1}).

end procedure

When ρ^k is sufficiently large, yet significantly less than unity (i.e. when $\rho_1 \leq \rho^k < \rho_2$), the linearization is deemed sufficiently accurate. Consequently, the solution (d, w) is accepted, and the trust region size is retained.

Lastly, when ρ^k is close to unity, the linear cost reduction accurately predicts the realized nonlinear cost reduction. Moreover, if ρ^k is greater than unity, then the linear approximation under-predicts the cost reduction, and is thus conservative. These conditions indicate that the linearization is accurate or conservative enough to enlarge the trust region. Hence, when $\rho_k \geq \rho_2$, the solution (d, w) is accepted, and the trust region size is increased by a factor of $\beta > 1$ to allow for larger w, and therefore d, in the next succession.

Comparison with Conventional Nonlinear Programming Methods

One important distinction between `SCvx` and typical trust-region-type algorithms lies in the subproblem solved at each iteration. Conventional trust-region algorithms usually perform a line search along the Cauchy arc to achieve a sufficient reduction [19]. However, in the `SCvx` algorithm, each convex subproblem is solved to full optimality, thus increasing the number of inner solver iterations at each succession, while decreasing the number of outer `SCvx` iterations. Qualitatively speaking, the number of successions is reduced by achieving a greater cost reduction at each succession. Thanks to the capabilities of existing IPM algorithms, and due to recent advancements in IPM customization techniques (e.g. [20, 24]), each convex subproblem can be solved quickly enough to outweigh the negative impacts of solving it to full optimality.

It is also crucial to point out some key differences between the `SCvx` algorithm and SQP-based methods (e.g. [12]). SQP-based methods typically make use of second-order information when approximating the Hessian of the cost function (and in some cases, of the constraints). This requires techniques like the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, which can be computationally expensive. Furthermore, additional conditions must be satisfied in order to ensure that the computed Hessian is positive-definite, and therefore, to guarantee that the resulting subproblem is convex [25]. For these two reasons, SQP-based

methods are not well suited for real-time autonomous applications. On the other hand, the `SCvx` algorithm relies only on first-order information obtained through linearization. While the first-order approximation may be less accurate than its second-order counterpart, the resulting error is properly regulated by the trust region updating mechanism outlined in Algorithm 6. Furthermore, since the Jacobian matrices can be determined analytically, very little computational effort is expended in setting up each succession. Most importantly, as a consequence of linearization, the resulting subproblems are automatically *guaranteed* to be convex, thus further ensuring the robustness of the convergence process.

Lastly, the Successive Linear Programming (SLP) method (e.g. [26]) may sound similar to our method by name, but in fact it is more closely related to SQP as it has a second phase that still requires a quadratic model. That aside, our method has another advantage over SLP, the retention of convex constraints. We only linearize system dynamics and non-convex constraints, which improves accuracy and speeds up convergence as shown in both theoretical convergence analysis and practice.

3.3 Global Convergence

In this section, we extend the global convergence analysis from [51] to facilitate the addition of state and control constraints. Since the state and control variables become indistinguishable once the optimal control problem is implemented as a numerical parameter optimization problem, we perform the following analysis accordingly. In lieu of the state and control variables, x and u, we will use $z \coloneqq [x^T, u^T]^T$ as our optimization variable, where $z \in \mathbb{R}^{n_z}$ and where $n_z = n_x N + n_u(N-1)$. Consequently, the constraints will be rewritten as a set of inequalities expressed in terms of z. Thus, we have the finite-dimensional non-convex optimization problem in Problem 3.3.1.

Problem 3.3.1. Original Non-Convex Optimization Problem

$$\min_{z} \quad g_0(z), \tag{3.10a}$$

$$\text{subject to:} \quad g_i(z) = 0, \qquad \forall i \in \mathcal{I}_{eq}, \tag{3.10b}$$

$$g_i(z) \leq 0, \qquad \forall i \in \mathcal{I}_{ineq}, \tag{3.10c}$$

$$h_j(z) \leq 0, \qquad \forall j \in \mathcal{J}_{ineq}, \tag{3.10d}$$

In Problem 3.3.1, $\mathcal{I}_{eq} \coloneqq \{1, 2, \ldots, e\}$ represents the set of non-convex equality constraint indices, $\mathcal{I}_{ineq} \coloneqq \{e+1, \ldots, p\}$, represents the set of non-convex inequality constraint indices, and $\mathcal{J}_{ineq} \coloneqq \{1, 2, \ldots, q\}$ represents the set of convex inequality indices. Correspondingly, equations (3.10b)-(3.10d) represent the nonlinear system dynamics, the non-convex state and control constraints, and the convex state and control constraints, respectively. We assume that $g_i(z)$ and $h_j(z)$ are continuously differentiable for all $i \in \mathcal{I}_{eq} \cup \mathcal{I}_{ineq}$ and $j \in \mathcal{J}_{ineq}$, respectively. To simplify the analysis, we assume that $g_0(z) \in C^1$, but note that $g_0(z)$ can be an element of C^0 in practice.

Since we are restricting our analysis to discrete-time systems, z is a finite dimensional vector. Consequently, the 1-norm used in [51] manifests itself as a finite sum of absolute values. With state and control inequality constraints incorporated, the exact penalty function in [51] can be extended as follows

$$J(z) \coloneqq g_0(z) + \sum_{i \in \mathcal{I}_{eq}} \lambda_i |g_i(z)| + \sum_{i \in \mathcal{I}_{ineq}} \lambda_i \max\big(0, g_i(z)\big), \tag{3.11}$$

where $\lambda_i \geq 0$ and $\tau_j \geq 0$ are scalars, and $\lambda \coloneqq [\lambda_1, \lambda_2, \ldots, \lambda_p]$ and $\tau \coloneqq [\tau_1, \tau_2, \ldots, \tau_q]$ represent the penalty weights. Next, we can now express the corresponding penalized problem in Problem 3.3.2.

Problem 3.3.2. Non-Convex Penalty Optimization Problem

$$\min_{z} \; J(z), \quad s.t.\; h_j(z) \leq 0,\; \forall j \in \mathcal{J}_{ineq}.$$

Note that $J(z)$ is non-convex, and thus needs to be convexified. According to the `SCvx` algorithm, at $(k+1)^{th}$ succession, we linearize $g_0(z)$, $g_i(z) = 0$ for all $i \in \mathcal{I}_{eq} \cup \mathcal{I}_{ineq}$ about z^k. This produces a sequence of convex penalty functions given by

$$\begin{aligned} L^k(d) \coloneqq g_0(z^k) + \nabla g_0(z^k)^T d + \sum_{i \in \mathcal{I}_{eq}} \lambda_i |g_i(z^k) + \nabla g_i(z^k)^T d| \\ + \sum_{i \in \mathcal{I}_{ineq}} \lambda_i \max\big(0, g_i(z^k) + \nabla g_i(z^k)^T d\big). \end{aligned} \tag{3.12}$$

Note that $L^k(d)$ is **not** exactly linearized $J(z)$, since the linearization is applied inside the $|\cdot|$ and $\max(0,\cdot)$ functions. This gives us the convexified penalty subproblem as in Problem 3.3.3.

Problem 3.3.3. Convex Penalty Subproblem

$$\min_{d} \quad L^k(d), \qquad \text{s.t.} \quad \|d\| \leq r^k,\; h_j(z) \leq 0,\; \forall j \in \mathcal{J}_{ineq}.$$

The corresponding *actual* cost reduction in (3.7) can be rewritten as $\Delta J(z,d) = J(z) - J(z+d)$, while the *predicted* cost reduction in (3.8) becomes $\Delta L(d) = J(z) - L(d)$.

To facilitate subsequent analysis, we first introduce some preliminary results and assumptions.

Theorem 3.3.1 (**Local Optimum**, Theorem 4.1 in [32])**.** *Let $N(\bar{z})$ denote an open neighborhood of $\bar{z}$ that contains feasible point of Problem 3.3.1. Then, if there exist $\bar{\lambda} \geq 0$, $\bar{\tau} \geq 0$, and $\bar{z} \in \mathbb{R}^{n_z}$ such that $J(\bar{z}) \leq J(z)$ for all penalty weights $\lambda \geq \bar{\lambda}$ and $\tau \geq \bar{\tau}$, and for all $z \in N(\bar{z})$, then $\bar{z}$ is a local optimum of Problem 3.3.1.*

Assumption 3.3.1 (**LICQ**)**.** Define the following sets of indices corresponding to the active inequality constraints: $I_{ac}(\bar{z}) \coloneqq \{\, i \mid g_i(\bar{z}) = 0,\ i \in \mathcal{I}_{ineq}\} \subseteq \mathcal{I}_{ineq}$, and $J_{ac}(\bar{z}) \coloneqq \{\, j \mid h_j(\bar{z}) = 0,\ j \in \mathcal{J}_{ineq}\} \subseteq \mathcal{J}_{ineq}$. Furthermore, define $G_{eq}(\bar{z})$ as a matrix whose columns comprise of $\nabla g_i(\bar{z})$ for all $i \in \mathcal{I}_{eq}$, $G_{ac,g}(\bar{z})$ as a matrix whose columns comprise of $\nabla g_i(\bar{z})$ for all $i \in I_{ac}(\bar{z})$, and $G_{ac,h}(\bar{z})$ as a matrix whose columns comprise of $\nabla h_i(\bar{z})$ for all $j \in J_{ac}(\bar{z})$. Then, the Linear Independence Constraint Qualification (LICQ) is satisfied at $\bar{z}$ if the columns of matrix $G_{ac}(\bar{z}) \coloneqq [\, G_{eq}(\bar{z}),\ G_{ac,g}(\bar{z}),\ G_{ac,h}(\bar{z})\,]$ are linearly independent.

The following theorem states the first-order necessary conditions (i.e. the KKT conditions) for a point $\bar{z}$ to be a local optimum of Problem 3.3.1 (in the of Theorem 3.3.1).

Theorem 3.3.2 (**Karush–Kuhn–Tucker (KKT)**, Theorem 3.2 in [32])**.** *If the constraints of Problem 3.3.1 satisfy LICQ (Assumption 3.3.1), and $\bar{z}$ is a local optimum, then there exist Lagrange multipliers $\bar{\mu}_i$ for all $i \in \mathcal{I}_{eq}$, $\bar{\mu}_i \geq 0$ for all $i \in \mathcal{I}_{ineq}$, and $\bar{\sigma}_j \geq 0$ for all $j \in \mathcal{J}_{ineq}$ such that*

$$\nabla g_0(\bar{z}) + \sum_{i \in \mathcal{I}_{eq}} \bar{\mu}_i \nabla g_i(\bar{z}) + \sum_{i \in I_{ac}(\bar{z})} \bar{\mu}_i \nabla g_i(\bar{z}) + \sum_{j \in J_{ac}(\bar{z})} \bar{\sigma}_j \nabla h_j(\bar{z}) = 0. \tag{3.13}$$

We refer to a point that satisfies the above conditions as a *KKT point*.

We say a penalty function is *exact* if there exists finite penalty weights $\bar{\mu}_i$ and $\bar{\sigma}_j$ such that Problem 3.3.1 and Problem 3.3.2 produce equivalent optimality conditions. Since Theorem 3.3.2 already gives the optimality condition for Problem 3.3.1, we now examine the first-order conditions for Problem 3.3.2.

For fixed λ, $J(z)$ is not differentiable everywhere because of the non-smoothness of $|\cdot|$ and $\max(0, \cdot)$. However, since $g_i(\cdot)$ and $h_j(\cdot)$ are both continuously differentiable, $J(z)$ is locally Lipschitz continuous. Therefore, we have the *generalized directional derivative* of $J(z)$ at some $\bar{z}$ in any direction s exists, and is defined as

$$dJ(\bar{z}, s) \coloneqq \limsup_{\substack{z \to \bar{z} \\ \delta \to 0^+}} \frac{J(z + \delta s) - J(z)}{\delta},$$

and also the *generalized differential* as

$$\partial J(\bar{z}) = \{\nu \,|\, J(\bar{z}+y) \geq J(\bar{z}) + \nu^T y, \forall\, y \in \mathcal{Z}\}, \tag{3.14}$$

where $\mathcal{Z}$ is a closed convex set representing the convex constraints.

Next we present a well-established lemma (Proposition 1 in [17]):

Lemma 3.3.1. *$J : \mathbb{R}^{n_z} \to \mathbb{R}$, and $J'_y(z)$ is its generalized directional derivative in direction y at z. Then for any $y \in \mathcal{Z}$,*

$$J'_y(z) = \sup_{d \in \partial J(z)} d^T y,$$

where $\partial J(z)$ is the generalized differential.

Following this, we can prove a more complex version of the stationarity condition $0 \in \partial J(\bar{z})$ for constrained case (minimize a non-convex function $J(z)$ over $\mathcal{Z}$): Applying Theorem 1 from [17] with the above definitions, we have

Lemma 3.3.2 (**Necessary Condition for Local Optimality**). *If $\bar{z}$ is a locally optimal solution of the non-convex penalty problem, Problem 3.3.2, then there exists a generalized gradient $d \in \partial J(\bar{z})$ such that $d^T(y - \bar{z}) \geq 0,\ \forall\, y \in \mathcal{Z}$.*

Alternatively, we may define the set of stationary points of Problem 3.3.2 as $S \coloneqq \{z \in \mathbb{R}^{n_z} | \,\exists\, d \in \partial J(z), s.t.\ d^T(y - z) \geq 0,\ \forall\, y \in \mathcal{Z}\}$, then Lemma 3.3.2 states that if $\bar{z}$ solves Problem 3.3.2 then $\bar{z} \in S$.

We can now state the fairly well-known *exactness* result. Its proof can be found in, for example, Theorem 4.4 and 4.8 of [32].

Theorem 3.3.3 (**Exactness of the Penalty Function**). *If $\bar{z}$ is a KKT point of the original Problem 3.3.1 with multipliers $\bar{\mu}_i$ for all $i \in \mathcal{I}_{eq} \cup \mathcal{I}_{ineq}$ and $\bar{\sigma}_j$ for all $j \in \mathcal{J}_{ineq}$, and if the penalty weights λ and τ satisfy $\lambda_i > |\bar{\mu}_i|,\ \forall\ i \in \mathcal{I}_{eq} \cup \mathcal{I}_{ineq}$, and $\tau_j > |\bar{\sigma}_j|,\ \forall\ j \in \mathcal{J}_{ineq}$ respectively, then $\bar{z} \in S$.*

Conversely, if a stationary point $\bar{z} \in S$ of the penalty Problem 3.3.2 is feasible for the original Problem 3.3.1, then it is also a KKT point of the original Problem 3.3.1, and therefore, (3.13) *holds at $\bar{z}$.*

Although Theorem 3.3.3 does not suggest a way to find such λ and τ, it nevertheless has important theoretical implications. In our current implementation, we select "sufficiently" large constant λ's and τ's, a strategy that has been shown to work well in practice. Alternatively, the values of λ and τ can be adjusted after each succession, based on the value of the dual solution obtained in the previously solved subproblem. However, it is the second (i.e. "converse") part of Theorem 3.3.3 that is particularly important to our subsequent convergence analysis. Specifically, it guarantees that as long as we can find a stationary point for Problem 3.3.2 that is also feasible for Problem 3.3.1, then that point also satisfies the KKT conditions for Problem 3.3.1.

To facilitate further analysis, we first note that for any z,

$$J(z+d) = L^k(d) + o(\|d\|), \tag{3.15}$$

where $o(\|d\|)$ denotes higher order terms of $\|d\|$, i.e.,

$$\lim_{\|d\|\to 0} \frac{o(\|d\|)}{\|d\|} = 0,$$

and is independent of z. This can be verified by writing out the Taylor expansion of $g_i(z)$ and $h_j(z)$ in $J(z)$, and then using the fact that $o(\|d\|)$ can be taken out of $|\cdot|$ and $\max(0, \cdot)$.

The next lemma is a key preliminary result, and its proof also provides some geometric insights into the `SCvx` algorithm. The proof is similar to and based on that of [51, Lemma 3], with modifications made to facilitate convex constraints.

Lemma 3.3.3. *Let $\bar{z} \in \mathbb{R}^{n_z}$ be a feasible point, but not a stationary point, of Problem 3.3.2. Use $N(\bar{z}, \bar{\epsilon})$ to denote an open neighborhood of $\bar{z}$ with radius $\bar{\epsilon}$, and let $\mathcal{P}(z, r)$ denote Problem 3.3.3 with a linearization evaluated at z and a trust region of radius r. Then, for any*

$c \in (0,1)$, there exist $\bar{r} > 0$ and $\bar{\epsilon} > 0$ such that for all $z \in N(\bar{z}, \bar{\epsilon})$ and $r \in (0, \bar{r}\,]$, any optimal solution d^* for $\mathcal{P}(z,r)$ satisfies $\rho(z,r) \coloneqq \frac{J(z)-J(z+d^*)}{J(z)-L^k(d^*)} \geq c$.

Proof. Since $\bar{z}$ is feasible but not stationary, we know that $\forall\, d \in \partial J(\bar{z})$, we have $d^T(y-\bar{z}) < 0,\ \forall\, y \in \mathcal{Z}$ according to Lemma 3.3.2. Thus we have

$$\forall\, d \in \partial J(\bar{z}),\ \sup_{y \in \mathcal{Z}} d^T y < d^T \bar{z}$$

Hence, there exists a unit vector $s \in \mathcal{Z}$ such that $\sup_{d \in \partial J(\bar{z})} d^T s < d^T \bar{z}$. Since d is a descent direction (from definition in (3.14)), we have $d^T \bar{z} \leq 0$, thus there exist $\kappa > 0$, such that $\sup_{s \in \mathcal{Z}} d^T s \leq -\kappa < 0$. The left hand side is exactly the implicit definition of the generalized direction derivative, as defined together with (3.14). Therefore, we have

$$dJ(\bar{z}, s) \coloneqq \limsup_{\substack{z \to \bar{z} \\ r \to 0^+}} \frac{J(z+rs) - J(z)}{r} \leq -\kappa.$$

This implies that there exist positive $\bar{r}$ and $\bar{\epsilon}$ such that for all $z \in N(\bar{z}, \bar{\epsilon})$ and $r \in (0,\ \bar{r}\,]$,

$$\frac{J(z+rs) - J(z)}{r} < -\frac{\kappa}{2}. \tag{3.16}$$

Now, assume 3.3.3 is solved with $z \in N(\bar{z}, \bar{\epsilon})$ and $r \in (0,\ \bar{r}\,]$, producing an optimal solution d^*. By using (3.15), the (nonlinear) change realized in J is

$$\begin{aligned} \Delta J(z, d^*) &= J(z) - J(z+d^*) = J(z) - L^k(d^*) - o(\|d^*\|) \\ &= \Delta L^k(d^*) - o(r). \end{aligned} \tag{3.17}$$

Thus, the ratio $\rho(z,r)$ is given by $\rho(z,r) = \frac{\Delta J(z,d^*)}{\Delta L^k(d^*)} = 1 - \frac{o(r)}{\Delta L^k(d^*)}$. Next, let $d' = rs \in \mathbb{R}^{n_z}$. From the definition of s, it follows that $\|d'\| = r$, and therefore that d' is within the thrust region of radius $\bar{r}$. Since d^* is the optimal solution, we have $L^k(d^*) \leq L^k(d')$, which in turn

means

$$\Delta L^k(d^*) \geq \Delta L^k(d'). \tag{3.18}$$

Now, as $r \to 0$, we will have $r \in (0,\, \bar{r}\,]$, and from (3.16) we have

$$\Delta J(z, d') = J(z) - J(z + d') > \left(\frac{\kappa}{2}\right) r. \tag{3.19}$$

Replacing d^* with d' in (3.17) and substituting into (3.19), we get $\Delta J(z, d') = \Delta L^k(d') - o(r) > \left(\frac{\kappa}{2}\right) r$. Combining this with 3.18, we obtain

$$\Delta L^k(d^*) - o(r) \geq \Delta L^k(d') - o(r) > \left(\frac{\kappa}{2}\right) r. \tag{3.20}$$

Thus $\Delta L^k(d^*) > (\kappa/2)r + o(r) > 0$, and the ratio

$$\rho(z, r) = 1 - \frac{o(r)}{\Delta L^k(d^*)} > 1 - \frac{o(r)}{(\kappa/2)r + o(r)}. \tag{3.21}$$

Therefore, as $r \to 0$, $\rho(z, r) \to 1$, and thus for any $c \in (0, 1)$, there exists $\bar{r} > 0$ such that for all $r \in (0,\, \bar{r}\,]$, $\rho(z, r) \geq c$ holds. □

Remark. An undesirable situation is one where steps are rejected indefinitely (i.e. by producing solutions that result in $\rho(z, r) < \rho_0$). Lemma 3.3.3 provides an assurance that the `SCvx` algorithm will not produce such behavior. By contracting r^k sufficiently, Lemma 3.3.3 guarantees that the ratio ρ^k will eventually exceed ρ_0, and the algorithm will stop rejecting steps.

Now, the global convergence result will be stated with a sketched proof. The reader is referred to [51] for more details.

Theorem 3.3.4 (**Global Convergence**). *Given Asuumption 3.3.1, regardless of initial conditions, the* `SCvx` *algorithm (Algorithm 6) always has limit points, and any limit point, $\bar{z}$, is a stationary point of the non-convex penalty Problem 3.3.2. Furthermore, if $\bar{z}$ is feasible for the original non-convex Problem 3.3.1, then it is a KKT point of Problem 3.3.1.*

Proof. Since we have assumed the feasible region to be convex and compact, by the Bolzano-Weierstrass theorem (see e.g. [65]), there is at least one convergent subsequence $\{z^{k_i}\} \to \bar{z}$, which is a guaranteed limit point.

The proof of stationarity of $\bar{z}$ is by contradiction, i.e. we assume that $\bar{z}$ is not a stationary point. From Lemma 3.3.3, there exist positive $\bar{r}$ and $\bar{\epsilon}$ such that

$$\rho(z, r) \geq \rho_0 \quad \forall\, z \in N(\bar{z}, \bar{\epsilon}\,) \;\text{ and }\; r \in (0,\, \bar{r}\,],$$

since ρ_0 can be chosen arbitrarily small. Without loss of generality, we can suppose the whole subsequence $\{z^{k_i}\}$ is in $N(\bar{z}, \bar{\epsilon})$, so that

$$\rho(z^{k_i}, r) \geq \rho_0 \quad \forall\, r \in (0,\, \bar{r}\,]. \tag{3.22}$$

If the initial trust region radius is less than $\bar{r}$, then (3.22) will be trivially satisfied. On the other hand, if the initial radius is greater than $\bar{r}$, then the trust region radius may need to be reduced several times by the rejection step in line 13 of Algorithm 6 before condition (3.22) is satisfied. Nevertheless, (3.22) will lead to $J(z^{k_i}) - J(z^{k_i+1}) \geq \rho_0 \Delta L^{k_i}.$, which implies that the penalized cost is monotonically decreasing. Subsequently, we use continuity of J and L^k to find a lower bound for ΔL^{k_i}, i.e. $\Delta L^{k_i} > \theta/2$, where θ is the predicted cost reduction by taking the optimal solution to the convex subproblem. Combining these two inequality, we obtain the following for all $i \geq i_0$:

$$J(z^{k_i}) - J(z^{k_i+1}) \geq \rho_0 \theta/2. \tag{3.23}$$

However, since $k_i + 1 \leq k_{(i+1)}$, and the penalized cost is not increasing, we have $J(z^{k_i+1}) \geq$

$J(z^{k_{(i+1)}})$, and thus

$$\sum_{i=1}^{\infty} \left(J(z^{k_i}) - J(z^{k_i+1}) \right) \leq \sum_{i=1}^{\infty} \left(J(z^{k_i}) - J(z^{k_{(i+1)}}) \right)$$
$$= J(z^{k_1}) - J(\bar{z}) \leq \infty.$$

Therefore, the series is convergent, and necessarily $J(z^{k_i}) - J(z^{k_i+1}) \rightarrow 0$, which contradicts (3.23). This contradiction implies that every limit point $\bar{z}$ is a stationary point of Problem 3.3.2. □

We would like to point out that this convergence on its own Theorem 3.3.4 does not guarantee feasibility of the converged solution, even if the original non-convex problem, Problem 3.3.1, may have feasible regions. However, it is important to also point out that this scenario is rarely observed in practice, and when it actually occurs, trying different initial guesses would often resolve it.

3.4 Strong Convergence

With minimal additional assumptions, the convergence result of `SCvx` will be stronger than classical numerical optimization schemes in the sense that the whole sequence converges to a single limit point rather than a set of accumulation points. Examples of the latter, i.e. weak convergence, include line-search algorithms [25, Theorem 2.5.1], trust-region methods [19, Theorem 6.4.6] and early-stage successive convexification algorithms [43, 51, 52]. Strong convergence, on the other hand, is a relatively recent development. [2] adapts Łojasiewicz's theorem [46] from dynamical systems to show strong convergence of iterative numerical optimization with real-analytic cost functions. [13, 8] extends the single limit point convergence to non-smooth functions, and for optimization problems that satisfy Kurdyka-Łojasiewicz (K-L) inequality, [8] provides handy conditions to check for convergence. In this document, we will demonstrate this type of strong convergence of the `SCvx` algorithm by verifying those conditions.

For the purpose of this section, it is not necessary to distinguish non-convex inequality constraints $g_i(z) \leq 0, \forall\, i \in \mathcal{I}_{ineq}$ and convex inequality constraints $h_j(z) \leq 0, \forall\, j \in \mathcal{J}_{ineq}$. Thus we will use $g_i(z) \leq 0, \forall\, i \in \{e+1, \ldots, p+q\}$ to denote all inequality constraints. For strong convergence, first we need a slightly stronger smoothness assumption.

Assumption 3.4.1. $g_i(z), \forall\, i = 0, \ldots, p+q$ have Lipschitz continuous gradients, that is, $\exists\, L_i \geq 0$, s.t.

$$\|\nabla g_i(z_2) - \nabla g_i(z_1)\| \leq L_i \|z_2 - z_1\|.$$

Using the simplified notation, we can rewrite the original non-convex (penalty) cost functions as

$$J(z) = g_0(z) + \sum_{i=1}^{e} \lambda_i |g_i(z)| + \sum_{i=e+1}^{p+q} \lambda_i \max\big(0, g_i(z)\big), \tag{3.24}$$

and the linearized (penalty) cost function at kth iteration as

$$\begin{aligned} L^k(d) = {} & g_0(z^k) + \nabla g_0(z^k)^T d + \sum_{i=1}^{e} \lambda_i |g_i(z^k) + \nabla g_i(z^k)^T d| \\ & + \sum_{i=e+1}^{p+q} \lambda_i \max\big(0, g_i(z^k) + \nabla g_i(z^k)^T d\big). \end{aligned} \tag{3.25}$$

The key assumption used in [8] to establish single point convergence is that the function been optimized satisfies the (nonsmooth) K-L property [8, Definition 2.4], which means, roughly speaking, that the functions under consideration are "sharp up to a reparametrization". Real semi-algebraic functions provide a very rich class of functions satisfying the K-L property. Many other functions may also satisfy this property, among which an important class is given by functions definable in an o-minimal structure [39]. One can verify that many real-world optimal control problems indeed have K-L property, especially by attesting its semi-algebracity. Therefore, it is reasonable to have the following assumption:

Assumption 3.4.2. The penalized cost function $J(z)$ in (3.24) have K-L property [8, Definition 2.4].

Given Assumption 3.4.2, [8, Section 2.3] establishes three conditions (referred as H1–H3) one can check to ensure strong convergence of an optimization scheme. Note that H3, the *continuity condition*, has already been proved in Theorem 3.3.4. Now, let us proceed to verify the other two conditions in the context of the `SCvx` algorithm.

Condition 3.4.1 (**Sufficient Decrease**)**.** At iteration k, let z^k be the current point, and z^{k+1} be the accepted solution by the `SCvx` algorithm, then the penalized cost function $J(\cdot)$ in (3.24) satisfies

$$J(z^k) - J(z^{k+1}) \geq a\|z^{k+1} - z^k\|^2, \tag{3.26}$$

where $a > 0$ is some constant.

Condition 3.4.2 (**Relative Error**)**.** At iteration k, let z^k be the current point, and z^{k+1} be the accepted solution by the `SCvx` algorithm, then given Assumption 3.4.1, $\exists\, \omega^{k+1} \in \partial J(z^{k+1})$, s.t.

$$\|\omega^{k+1}\| \leq b\|z^{k+1} - z^k\|, \tag{3.27}$$

where $b > 0$ is some constant, and $\partial J(\cdot)$ is the generalized differential defined in (3.14).

The intuition behind these two conditions can be found in [8, p. 92], and roughly speaking, Condition 3.4.1 measures the quality of a descent step, while Condition 3.4.2 reflects relative inexact optimality conditions for subproblems. The proof of Condition 3.4.1 is relatively straightforward, and is given as follows.

Proof of Condition 3.4.1. Note that in (3.17), $d^* = z^{k+1} - z^k$, then from (3.17), (3.20) and the trust region constraint $\|d^*\| \leq r$, we have that

$$J(z^k) - J(z^{k+1}) \geq \frac{\kappa}{2} r \geq \frac{\kappa}{2}\|d^*\|. \tag{3.28}$$

Since $z \in Z = X \times U$, and Z is compact, $\|d^*\| = \|z^{k+1} - z^k\|$ is bounded. Let $D = \max_{z_1, z_2 \in Z} \|z_1 - z_2\|$, then $\|d^*\| \leq D, \forall\, d$.

Now, let $a = \frac{\kappa}{2D}$, then from 3.28, we have

$$J(z^k) - J(z^{k+1}) \geq \frac{\kappa}{2}\frac{1}{\|d^*\|}\|d^*\|^2 \geq \frac{\kappa}{2D}\|d^*\|^2 = a\|d^*\|^2 = a\|z^{k+1} - z^k\|^2,$$

which is exactly (3.26). □

The verification of Conditon 3.4.2 requires a bit more work and a constructive approach. We will leverage convexity of the generalized subdifferential and some special structures of the two types of non-smoothness, $|\cdot|$ and $\max(0, \cdot)$ present in (3.24) and (3.25). Namely, $\partial|g_i(z)|$ is symmetric with respect to 0, and in fact a convex hull spanned by $-\nabla g_i(z)$ and $\nabla g_i(z)$. Thus we have $\forall\, \omega_i \in \partial|g_i(z)|, i = 1, \ldots, e$, $\exists\, \alpha_i \in [0, 1]$, s.t.

$$\begin{aligned} \omega_i &= \alpha_i \nabla g_i(z) + (1 - \alpha_i)(-\nabla g_i(z)), \\ &= (2\alpha_i - 1)\nabla g_i(z). \end{aligned} \tag{3.29}$$

Similarly, $\partial \max\big(0, g_i(z)\big)$ is a convex hull of 0 and $\nabla g_i(z)$, and we also have $\forall\, \omega_i \in \partial \max\big(0, g_i(z)\big), i = e + 1, \ldots, p + q$, $\exists\, \alpha_i \in [0, 1]$, s.t.

$$\omega_i = \alpha_i \nabla g_i(z) + (1 - \alpha_i)0 = \alpha_i \nabla g_i(z). \tag{3.30}$$

Next we present two technical lemmas asserting the close proximity of the generalized differentials at z^{k+1} and their linear approximations around z^k.

Lemma 3.4.1. *For any $\nu_i^k \in \partial_d|g_i(z^k) + \nabla g_i(z^k)^T d|$, $i = 1, \ldots, e$, there exists $\omega_i^{k+1} \in \partial|g_i(z^{k+1})|$ and constant $c_i \geq 0$ s.t.*

$$\|\omega_i^{k+1} - \nu_i^k\| \leq c_i \|z^{k+1} - z^k\|.$$

Proof. The distance between the two generalized differentials is captured by the distance

between the two vectors, ω_i^{k+1} and ν_i^k, and we have

$$\begin{aligned}\|\omega_i^{k+1} - \nu_i^k\| &= \|\omega_i^{k+1} - \nabla g_i(z^{k+1}) + \nabla g_i(z^{k+1}) - \nu_i^k + \nabla g_i(z^k) - \nabla g_i(z^k)\| \\ &\leq \|\omega_i^{k+1} - \nabla g_i(z^{k+1}) - \nu_i^k + \nabla g_i(z^k)\| + \|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\|.\end{aligned} \tag{3.31}$$

Now, using (3.29), we have $\omega_i^{k+1} = (2\alpha_i - 1)\nabla g_i(z^{k+1})$, and $\nu_i^k = (2\bar{\alpha}_i - 1)\nabla g_i(z^k)$. Note that $\alpha_i \in [0, 1]$ is free, while $\bar{\alpha}_i \in [0, 1]$ is fixed for a given ν_i^k. Therefore, we have

$$\|\omega_i^{k+1} - \nu_i^k\| \leq \|2(\alpha_i - 1)\nabla g_i(z^{k+1}) - 2(\bar{\alpha}_i - 1)\nabla g_i(z^k)\| + \|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\|.$$

As illustrated in Figure 3.1, we may choose $\alpha_i = \bar{\alpha}_i$, and get

$$\begin{aligned}\|\omega_i^{k+1} - \nu_i^k\| &\leq 2(1 - \bar{\alpha}_i)\|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\| + \|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\| \\ &= (3 - 2\bar{\alpha}_i)\|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\|.\end{aligned}$$

By Assumption 3.4.1, we have

$$\|\omega_i^{k+1} - \nu_i^k\| \leq (3 - 2\bar{\alpha}_i)L_i\|z^{k+1} - z^k\| := c_i\|z^{k+1} - z^k\|.$$

□

Lemma 3.4.2. *For any $\nu_i^k \in \partial_d \max\left(0, g_i(z^k) + \nabla g_i(z^k)^T d\right)$, $i = e+1, \ldots, p+q$, there exists $\omega_i^{k+1} \in \partial \max\left(0, g_i(z^{k+1})\right)$ and constant $c_i \geq 0$ s.t.*

$$\|\omega_i^{k+1} - \nu_i^k\| \leq c_i\|z^{k+1} - z^k\|.$$

Proof. We still have (3.31) in this case. Now, using (3.30), we have $\omega_i^{k+1} = \alpha_i \nabla g_i(z^{k+1})$, and $\nu_i^k = \bar{\alpha}_i \nabla g_i(z^k)$. Again, note that $\alpha_i \in [0, 1]$ is free, while $\bar{\alpha}_i \in [0, 1]$ is fixed for a given ν_i^k.

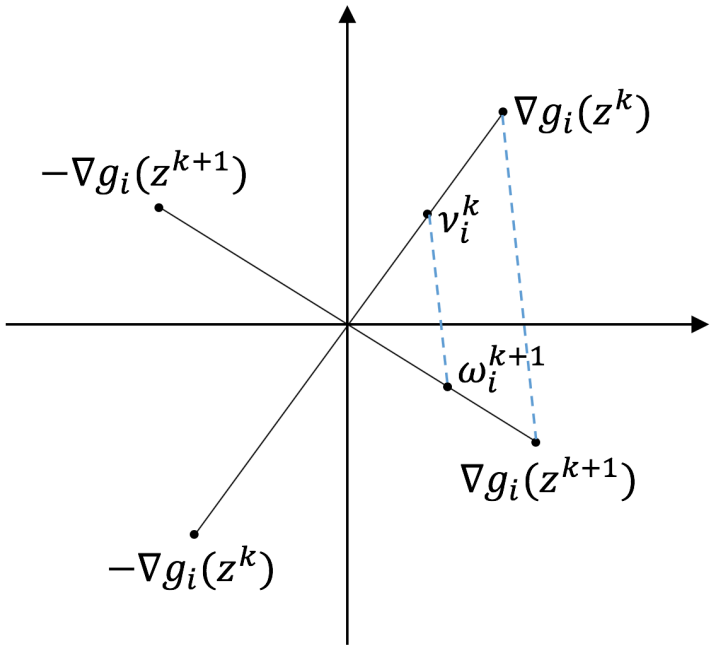


Figure 3.1: *In the proof of Lemma 3.4.1, by choosing $\alpha_i = \bar{\alpha}_i$, we are effectively setting ω_i^{k+1} to have the same convex combination factor as ν_i^k.*

Therefore, from (3.31) we have

$$\|\omega_i^{k+1} - \nu_i^k\| \leq \|(\alpha_i - 1)\nabla g_i(z^{k+1}) - (\bar{\alpha}_i - 1)\nabla g_i(z^k)\| + \|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\|.$$

Again, choose $\alpha_i = \bar{\alpha}_i$, and we get

$$\begin{aligned}\|\omega_i^{k+1} - \nu_i^k\| &\leq (1 - \bar{\alpha}_i)\|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\| + \|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\| \\ &= (2 - \bar{\alpha}_i)\|\nabla g_i(z^{k+1}) - \nabla g_i(z^k)\|.\end{aligned}$$

By Assumption 3.4.1, we have

$$\|\omega_i^{k+1} - \nu_i^k\| \leq (2 - \bar{\alpha}_i)L_i\|z^{k+1} - z^k\| := c_i\|z^{k+1} - z^k\|.$$

□

Now we are ready to present the proof of Condition 3.4.2, which makes use of the optimality conditions of the convex subproblem.

Proof of Condition 3.4.2. First note that

$$\omega^{k+1} = \nabla g_0(z^{k+1}) + \sum_{i=1}^{e} \lambda_i \omega_i^{k+1} + \sum_{i=e+1}^{p+q} \lambda_i \omega_i^{k+1}, \tag{3.32}$$

where $\omega_i^{k+1} \in \partial |g_i(z^{k+1})|$, $i = 1, \ldots, e$ and $\omega_i^{k+1} \in \partial \max\left(0, g_i(z^{k+1})\right)$, $i = e+1, \ldots, p+q$. Now let us turn to the stationarity condition for the convex subproblem:

$$0 \in \partial_d L^k(d) + \mathcal{N}_{\|d\| \le r^k}(d) \tag{3.33}$$

where $\mathcal{N}_{\|d\| \le r^k}(d)$ is the normal cone of set $\|d\| \le r^k$ at d, which turns out to be $c\,d$, i.e. $c\,(z^{k+1} - z^k)$, where $c \ge 0$ is a constant. From the formula of $L^k(d)$ in (3.25), condition in (3.33) is equivalent to

$$\exists\, \nu^k \in \partial_d L^k(d), \quad \text{s.t. } 0 = \nu^k + c\,(z^{k+1} - z^k), \tag{3.34}$$

where $\nu^k = \nabla g_0(z^k) + \sum_{i=1}^{e} \lambda_i \nu_i^k + \sum_{i=e+1}^{p+q} \lambda_i \nu_i^k$, where $\nu_i^k \in \partial_d |g_i(z^k) + \nabla g_i(z^k)^T d|$, $i = 1, \ldots, e$ and $\nu_i^k \in \partial_d \max\left(0, g_i(z^k) + \nabla g_i(z^k)^T d\right)$, $i = e+1, \ldots, p+q$. Therefore, (3.34) can be written

as

$$
\begin{aligned}
& 0 = \nabla g_0(z^k) + \sum_{i=1}^{e} \lambda_i \nu_i^k + \sum_{i=e+1}^{p+q} \lambda_i \nu_i^k + c\,(z^{k+1} - z^k) \\
\Leftrightarrow \quad & 0 = \nabla g_0(z^k) - \nabla g_0(z^{k+1}) + \nabla g_0(z^{k+1}) \\
& \quad + \sum_{i=1}^{e} \lambda_i (\nu_i^k - \omega_i^{k+1} + \omega_i^{k+1}) \\
& \quad + \sum_{i=e+1}^{p+q} \lambda_i (\nu_i^k - \omega_i^{k+1} + \omega_i^{k+1}) + c\,(z^{k+1} - z^k)
\end{aligned}
$$

$$
\begin{aligned}
\Leftrightarrow \quad & \nabla g_0(z^{k+1}) + \sum_{i=1}^{e} \lambda_i \omega_i^{k+1} + \sum_{i=e+1}^{p+q} \lambda_i \omega_i^{k+1} \\
& = \nabla g_0(z^k) - \nabla g_0(z^{k+1}) + \sum_{i=1}^{e} \lambda_i (\nu_i^k - \omega_i^{k+1}) \\
& \quad + \sum_{i=e+1}^{p+q} \lambda_i (\nu_i^k - \omega_i^{k+1}) + c\,(z^{k+1} - z^k)
\end{aligned}
$$

The left-hand-side of the above equation is exactly ω^{k+1} as in (3.32), and apply Assumption 3.4.1 and Lemma 3.4.1 and Lemma 3.4.2 to the right-hand-side, we get

$$
\begin{aligned}
\|\omega^{k+1}\| &\le \|\nabla g_0(z^k) - \nabla g_0(z^{k+1})\| \\
&\quad + \sum_{i=1}^{p+q} \lambda_i \|\nu_i^k - \omega_i^{k+1}\| + c\,\|z^{k+1} - z^k\| \\
&\le (L_0 + \sum_{i=1}^{p+q} \lambda_i c_i + c) \|z^{k+1} - z^k\|.
\end{aligned}
$$

where $c_i = (3 - 2\bar{\alpha}_i)L_i$ for $i = 1, \ldots, e$ and $c_i = (2 - \bar{\alpha}_i)L_i$ for $i = e+1, \ldots, p+q$, where $\bar{\alpha}_i$ is the convex combination factor for ν_i^k. Let $b = L_0 + \sum_{i=1}^{p+q} \lambda_i c_i + c$, we have (3.27). □

Now that we verified the strong convergence conditions, Condition 3.4.1 and Condi-

tion 3.4.2, by using [8, Theorem 2.9] and the weak convergence results in the previous section 3.3.4, we are in the position to claim the following:

Theorem 3.4.1 (**Global Strong Convergence**). *Suppose Assumption 3.3.1, 3.4.1, and 3.4.2 hold, then regardless of initial conditions, the sequence* $\{z^k\}$ *generated by the* `SCvx` *algorithm (Algorithm 6) always converges to a single limit point,* $\bar{z}$*, and* $\bar{z}$ *is a stationary point of the non-convex penalty Problem 3.3.2. Furthermore, if* $\bar{z}$ *is feasible for the original Problem 3.3.1, then it is a KKT point of Problem 3.3.1.*

3.5 Superlinear Convergence Rate

In the following, we will show that the `SCvx` algorithm converges not only globally, but also superlinearly under some additional mild assumptions. Moreover, we will see that the superlinear rate of convergence is enabled by the structure of the underlying optimal control problem. In other words, the `SCvx` algorithm is specifically tailored to solve non-convex optimal control problems, and thus enjoys a faster convergence rate when compared to generic nonlinear programming methods, which often converge linearly, if at all.

First we note that at each succession of Algorithm 6, we are solving a convex programming problem, which is best solved using IPMs that employ self-dual embedding technique introduced by [29]. A particular advantage of this approach is that it always produces a strictly complementary solution, thus satisfying the following assumption:

Assumption 3.5.1 (**Strict Complementary Slackness**). In addition to the KKT conditions in Theorem 3.3.2, we assume that $\bar{\mu}_i > 0,\ \forall\, i \in I_{ac}(\bar{z})$ and $\bar{\sigma}_j > 0,\ \forall\, j \in J_{ac}(\bar{z})$ are satisfied at the local optimum $\bar{z}$.

The next assumption leverages the structure of optimal control problems, and is crucial in subsequent analysis.

Assumption 3.5.2 (**Binding**). Let $z^k \to \bar{z}$. There are at least $n_u(N-1)$ binding constraints in $g_i(\bar{z}) \leq 0$, $i \in \mathcal{I}_{ineq}$ and $h_j(\bar{z}) \leq 0$, $j \in \mathcal{J}_{ineq}$. That is, $|I_{ac}(\bar{z})| + |J_{ac}(\bar{z})| \geq n_u(N-1)$.

Optimal control problems often observes the *bang-bang principle*, provided that the Hamiltonian is affine in controls, the control set is a convex polyhedron, and there are no *singular arcs* [69]. Linear systems with these properties are referred as *normal* (see Corollary 7.3 of [9]). For nonlinear systems, the non-singular condition can be checked by sequentially examine the *Lie bracket* of the system dynamics (see Section 4.4.3 of [42]). If the optimal control is indeed *bang-bang*, then Assumption 3.5.2 is obviously satisfied. Once classic example is the *bang-bang* solution obtained by solving a minimum time optimal control problem [40]. The algorithm proposed in [34] can also be used to ensure *bang-bang* property of the solutions.

Note that even if some control constraints are inactive at the optimal solution, as long as there are an equal or greater number of active state constraints, then Assumption 3.5.2 still holds true. An interesting example of such a case is that of the maximum-divert planetary landing problem [33] containing both control constraints and velocity constraints. In this example, the control constraints are inactive only when the velocity constraints are activated.

Nevertheless, given the special structure of optimal control problems and the properties of their solutions, we have the following lemma:

Lemma 3.5.1. *Given Assumption 3.3.1, 3.5.2, the gradient set of active constraints $G_{ac}(\bar{z})$ contains a basis of $\mathbb{R}^{n_z}$, where $n_z = n_x N + n_u(N-1)$. In other words, there are at least n_z linearly independent vectors in $G_{ac}(\bar{z})$.*

Proof. From Assumption 3.3.1, 3.5.2, we have at least $n_u(N-1)$ linearly independent gradient vectors from active control constraints, In addition, from the formulation of the original optimal control problem, we know that there are at least $n_x N$ equality constraints due to the system dynamics, and their gradient vectors are also linearly independent by Assumption 3.3.1. Therefore, $G_{ac}(\bar{z})$ has at least n_z linearly independent vectors. □

Next, we prove several technical lemmas that are instrumental in obtaining the final convergence rate result. In this section, we will use $\{z^k\} \to \bar{z}$ broadly to denote weak convergence (i.e. $\{z^k\}$ is the subsequence converged to $\bar{z}$).

Lemma 3.5.2. *Let $\{y^k\}$ be a sequence in $\mathbb{R}^{n_z}$ such that $\{y^k\} \neq \bar{z}$ and $\{y^k\} \to \bar{z}$. Then, there exists $\xi \in \mathbb{R}^{n_z}, \|\xi\| = 1$, such that for any function $g(\cdot) \in C^1 : \mathbb{R}^{n_z} \to \mathbb{R}$, we have a subsequence denoted by $k_s, s = 1, 2, \ldots$, such that*

$$\lim_{k_s \to \infty} \frac{g(y^{k_s}) - g(\bar{z})}{\|y^{k_s} - \bar{z}\|} = \nabla g(\bar{z})^T \xi. \tag{3.35}$$

Proof. Define $\xi^k \coloneqq \frac{y^k - \bar{z}}{\|y^k - \bar{z}\|}$, such that $\|\xi^k\| = 1$. Clearly $\{\xi^k\}$ is bounded, and by Bolzano-Weierstrass theorem (see e.g. [65]), this sequence has a convergent subsequence $\{\xi^{k_s}\} \to \xi$, i.e., the limit of the left hand side of (3.35) exists, and

$$\xi = \lim_{k_s \to \infty} \xi^{k_s} = \lim_{k_s \to \infty} \frac{y^{k_s} - \bar{z}}{\|y^{k_s} - \bar{z}\|}, \tag{3.36}$$

as well as $\|\xi\| = \|\xi^{k_s}\| = 1$. Let $y^{k_s} = \bar{z} + \omega^{k_s} \xi^{k_s}$, where $\omega^{k_s} = \|y^{k_s} - \bar{z}\|$, and $\omega^{k_s} \to 0$, as $k_s \to \infty$. Then, we have $\lim_{k_s \to \infty} \frac{g(y^{k_s}) - g(\bar{z})}{\|y^{k_s} - \bar{z}\|} = \lim_{k_s \to \infty} \frac{g(\bar{z} + \omega^{k_s} \xi^{k_s}) - g(\bar{z})}{\|\omega^{k_s} \xi^{k_s}\|}$. Let $\theta^{k_s} \coloneqq \xi^{k_s} - \xi$, such that $\theta^{k_s} \to 0$ as $k_s \to \infty$. Now we have

$$\begin{aligned} \lim_{k_s \to \infty} \frac{g(y^{k_s}) - g(\bar{z})}{\|y^{k_s} - \bar{z}\|} &= \lim_{k_s \to \infty} \frac{g(\bar{z} + \omega^{k_s} \xi + \omega^{k_s} \theta^{k_s}) - g(\bar{z})}{\omega^{k_s}} \\ &= \lim_{k_s \to \infty} \frac{g(\bar{z} + \omega^{k_s} \xi + \omega^{k_s} \theta^{k_s}) - g(\bar{z} + \omega^{k_s} \xi)}{\omega^{k_s}} + \lim_{k_s \to \infty} \frac{g(\bar{z} + \omega^{k_s} \xi) - g(\bar{z})}{\omega^{k_s}}. \end{aligned} \tag{3.37}$$

Since $g(\cdot) \in C^1$, we apply the mean value theorem, such that for the first term of (3.37), we have $\lim_{k_s \to \infty} \frac{\omega^{k_s} {\theta^{k_s}}^T \nabla g(\bar{z} + \omega^{k_s} \xi + \omega^{k_s} \bar{\theta}^{k_s})}{\omega^{k_s}} = \lim_{k_s \to \infty} \nabla g(\bar{z})^T \theta^{k_s} = 0$, where $\bar{\theta}^{k_s}$ denotes a point that lies on the line segment between 0 and θ^{k_s}. Therefore, by definition of the directional derivative, (3.37) becomes

$$\lim_{k_s \to \infty} \frac{g(y^{k_s}) - g(\bar{z})}{\|y^{k_s} - \bar{z}\|} = \lim_{k_s \to \infty} \frac{g(\bar{z} + \omega^{k_s} \xi) - g(\bar{z})}{\omega^{k_s}} = \nabla g(\bar{z})^T \xi,$$

by which we conclude the proof. □

Lemma 3.5.3. *Let $\{z^k\}$ be the convergent sequence generated by the* `SCvx` *algorithm, and*

$\{z^k\} \to \bar{z}$. Assume $\bar{z}$ is feasible to the original problem, Problem 3.3.1, then under Assumption 3.3.1, 3.5.1, 3.5.2, there exist $\beta > 0$ and $\delta > 0$ such that $\forall z \in N(\bar{z}, \delta) := \{z \,|\, \|z - \bar{z}\| \leq \delta\}$, we have

$$J(z) - J(\bar{z}) \geq \beta \|z - \bar{z}\|, \tag{3.38}$$

where $J(z)$ is the penalty cost defined in (3.11).

Proof. We will prove by contradiction. Assume the statement is false. It means that for a given diminishing sequence $\{\varepsilon^k\} \to 0,\ k = 1, 2, \ldots,$ there exists sequence $\{y^k\} \neq \bar{z}$ and $\{y^k\} \to \bar{z},\ k = 1, 2, \ldots,$ such that

$$J(y^k) - J(\bar{z}) \leq \varepsilon^k \|y^k - \bar{z}\|. \tag{3.39}$$

Now let $\{y^{k_s}\}$ denote a subsequence of $\{y^k\}$, such that (3.36) holds and $\|\xi\| = 1$. Since $\bar{z}$ is assumed to be feasible, we have $J(\bar{z}) = g_0(\bar{z})$. Therefore, (3.39) with respect to k_s becomes $g_0(y^{k_s}) + \sum_{i \in \mathcal{I}_{eq}} \lambda_i |g_i(y^{k_s})| + \sum_{i \in \mathcal{I}_{ineq}} \lambda_i \max\left(0, g_i(y^{k_s})\right) + \sum_{j \in \mathcal{J}_{ineq}} \tau_j \max\left(0, h_j(y^{k_s})\right) - g_0(\bar{z}) \leq \varepsilon^{k_s} \|y^{k_s} - \bar{z}\|$, which can be rewritten as

$$\begin{aligned} &g_0(y^{k_s}) - g_0(\bar{z}) + \sum_{i \in \mathcal{I}_{eq}} \lambda_i |g_i(y^{k_s}) - g_i(\bar{z})| + \sum_{i \in I_{ac}(\bar{z})} \lambda_i \max\left(0, g_i(y^{k_s}) - g_i(\bar{z})\right) \\ &+ \sum_{j \in J_{ac}(\bar{z})} \tau_j \max\left(0, h_j(y^{k_s}) - h_j(\bar{z})\right) \leq \varepsilon^{k_s} \|y^{k_s} - \bar{z}\|. \end{aligned}$$

Dividing both sides by $\|y^{k_s} - \bar{z}\|$, we have

$$\begin{aligned} &\frac{g_0(y^{k_s}) - g_0(\bar{z})}{\|y^{k_s} - \bar{z}\|} + \frac{\sum_{i \in \mathcal{I}_{eq}} \lambda_i |g_i(y^{k_s}) - g_i(\bar{z})|}{\|y^{k_s} - \bar{z}\|} + \frac{\sum_{i \in I_{ac}(\bar{z})} \lambda_i \max\left(0, g_i(y^{k_s}) - g_i(\bar{z})\right)}{\|y^{k_s} - \bar{z}\|} \\ &+ \frac{\sum_{j \in J_{ac}(\bar{z})} \tau_j \max\left(0, h_j(y^{k_s}) - h_j(\bar{z})\right)}{\|y^{k_s} - \bar{z}\|} \leq \varepsilon^{k_s}. \end{aligned}$$

Let $k_s \to \infty$, then by (3.5.2) and the definition of ξ from (3.36), we have

$$\nabla g_0(\bar{z})^T\xi + \sum_{i\in\mathcal{I}_{eq}} \lambda_i |\nabla g_i(\bar{z})^T\xi| + \sum_{i\in I_{ac}(\bar{z})} \lambda_i \max\left(0, \nabla g_i(\bar{z})^T\xi\right) + \sum_{j\in J_{ac}(\bar{z})} \tau_j \max\left(0, \nabla h_j(\bar{z})^T\xi\right) \leq 0.$$

Now recall the KKT condition in (3.3.2). We subtracting the product of (3.13) and ξ from the above equation, we have

$$\sum_{i\in\mathcal{I}_{eq}} \left[\lambda_i |\nabla g_i(\bar{z})^T\xi| - \bar{\mu}_i \nabla g_i(\bar{z})^T\xi\right] + \sum_{i\in I_{ac}(\bar{z})} \left[\lambda_i \max(0, \nabla g_i(\bar{z})^T\xi) - \bar{\mu}_i \nabla g_i(\bar{z})^T\xi\right] + \sum_{j\in J_{ac}(\bar{z})} \left[\tau_j \max(0, \nabla h_j(\bar{z})^T\xi) - \bar{\sigma}_j \nabla h_j(\bar{z})^T\xi\right] \leq 0,$$

where $\bar{\mu}_i$ and $\bar{\sigma}_j$ are Lagrange multipliers associated with constraints. Due to the exactness property in Theorem 3.3.3, these three terms are all nonnegative, and by the strict complementary slackness property in Assumption 3.5.1, we have $\nabla g_i(\bar{z})^T\xi = 0$, $\forall\, i \in \mathcal{I}_{eq} \cup I_{ac}(\bar{z})$ and $\nabla h_j(\bar{z})^T\xi = 0$, $\forall\, j \in J_{ac}(\bar{z})$, and therefore we have

$$\left[\nabla g_i(\bar{z})^T,\, \nabla h_j(\bar{z})^T\right]\xi = 0, \quad \forall\, i \in \mathcal{I}_{eq} \cup I_{ac}(\bar{z}),\, j \in J_{ac}(\bar{z}).$$

However, by Lemma 3.5.1 we know that the column space of $[\nabla g_i(\bar{z})^T,\, \nabla h_j(\bar{z})^T]$ contains a basis of $\mathbb{R}^{n_z}$. Since $\xi \in \mathbb{R}^{n_z}$, this implies that $\xi = 0$, which contradicts the fact that $\|\xi\| = 1$, and thus (3.38) holds true. □

Lemma 3.5.3 provides an important condition that is satisfied by many optimal control problems. Next we show that given this condition, the `SCvx` procedure will indeed converge superlinearly. To proceed, first denote the stack of $g_0(\cdot), g_i(\cdot)$, and $h_j(\cdot)$ as $G(\cdot) \in C^1$, and represent $J(\cdot)$ by a function composition $\psi(G(\cdot))$. Qualitatively we can write $\psi(G(\cdot)) = G_{cost}(\cdot) + |G_{eq}(\cdot)| + \max\left(0, G_{ineq}(\cdot)\right)$. Note that $\psi(\cdot)$ is convex since both $|\cdot|$ and $\max(0, \cdot)$

are convex functions.

Lemma 3.5.4. *Following Lemma 3.5.3, there exists $\gamma > 0$, such that*

$$\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d\right) \geq \psi(G(\bar{z})) + \gamma\|d\|, \quad \forall\, d \in \mathbb{R}^{n_z}. \tag{3.40}$$

Proof. First we show that the statement is true for any small step d_δ such that $\|d_\delta\| \leq \delta$, where δ is defined in Lemma 3.5.3. We have $\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d_\delta\right) - \psi(G(\bar{z})) = \psi(G(\bar{z} + d_\delta)) - \psi(G(\bar{z})) + \psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d_\delta\right) - \psi(G(\bar{z} + d_\delta)) \geq \beta\|d_\delta\| + o(\|d_\delta\|) \geq \frac{\beta}{2}\|d_\delta\|$. Let $\gamma \coloneqq \frac{\beta}{2}$, then we have (3.40) hold for a small step d_δ. Note that the first inequality is due to Lemma 3.5.3 and again the fact that $o(\|d_\delta\|)$ can be taken out of $|\cdot|$ and $\max(0, \cdot)$ and $G(\cdot) \in C^1$.

Now to generalize this to any $d \in \mathbb{R}^{n_z}, d \neq 0$, we first define $\zeta \coloneqq \min\left(1, \delta/\|d\|\right)$, and let $z_\zeta \coloneqq \bar{z} + \zeta d$. Denoting $(z_\zeta - \bar{z})$ as d_ζ, we have $d_\zeta = \zeta\, d$. With this definition of ζ, one can verify that $\|d_\zeta\| \leq \delta$. Hence, (3.40) holds true for d_ζ. Therefore, we have

$$\begin{aligned}
\gamma\zeta\|d\| = \gamma\|d_\zeta\| &\leq \psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d_\zeta\right) - \psi(G(\bar{z})) \\
&= \psi\left(G(\bar{z}) + \zeta\nabla G(\bar{z})^T d\right) - \psi(G(\bar{z})) \\
&= \psi\left((1-\zeta)G(\bar{z}) + \zeta\left[G(\bar{z}) + \nabla G(\bar{z})^T d\right]\right) - \psi(G(\bar{z})) \\
&\leq (1-\zeta)\psi(G(\bar{z})) + \zeta\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d\right) - \psi(G(\bar{z})) \\
&= \zeta\left[\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d\right) - \psi(G(\bar{z}))\right].
\end{aligned}$$

The second inequality is due to the convexity of ψ (note that $0 \leq \zeta \leq 1$). Dividing both sides by ζ, we obtain (3.40). □

Theorem 3.5.1 (**Superlinear Convergence**). *Given a sequence $\{z^k\} \to \bar{z}$ generated by the* `SCvx` *algorithm (Algorithm 6) and suppose that Assumption 3.3.1, 3.5.1, 3.5.2 are satisfied, then we have $\|z^{k+1} - \bar{z}\| = o(\|z^k - \bar{z}\|)$.*

Proof. First consider the case without trust region constraints. Since $d^k \coloneqq z^{k+1} - z^k$ is the

unconstrained optimal solution to the convex subproblem, we have $\forall\, d \in \mathbb{R}^{n_z}$, $\psi\big(G(z^k) + \nabla G(z^k)^T\, d\big) \geq \psi\big(G(z^k) + \nabla G(z^k)^T\, d^k\big)$. Let $d = \bar{z} - z^k$, we then have

$$\psi\left(G(z^k) + \nabla G(z^k)^T\,(\bar{z} - z^k)\right) - \psi\left(G(z^k) + \nabla G(z^k)^T\,(z^{k+1} - z^k)\right) \geq 0.$$

Together with (3.40), we get

$$\begin{aligned}\gamma\|z^{k+1} - \bar{z}\| \leq\ & \psi\left(G(\bar{z}) + \nabla G(\bar{z})^T\,(z^{k+1} - \bar{z})\right) - \psi\big(G(\bar{z})\big) \\ & + \psi\left(G(z^k) + \nabla G(z^k)^T\,(\bar{z} - z^k)\right) - \psi\left(G(z^k) + \nabla G(z^k)^T\,(z^{k+1} - z^k)\right).\end{aligned}$$

Since ψ is convex and has a compact domain, it is (locally) Lipschitz continuous (see e.g. [63]). Therefore, we have $\psi\left(G(z^k) + \nabla G(z^k)^T\,(\bar{z} - z^k)\right) - \psi\big(G(\bar{z})\big) \leq L\|G(z^k) - G(\bar{z}) + \nabla G(z^k)^T\,(\bar{z} - z^k)\|$, where L is the Lipschitz constant, and

$$\begin{aligned}&\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T\,(z^{k+1} - \bar{z})\right) - \psi\left(G(z^k) + \nabla G(z^k)^T\,(z^{k+1} - z^k)\right) \\ &\quad \leq L\|G(\bar{z}) - G(z^k) - \nabla G(z^k)^T\,(\bar{z} - z^k) + \left[\nabla G(\bar{z}) - \nabla G(z^k)\right]^T\left(z^{k+1} - \bar{z}\right)\| \\ &\quad \leq L\|G(\bar{z}) - G(z^k) - \nabla G(z^k)^T\,(\bar{z} - z^k)\| + L\|\left[\nabla G(\bar{z}) - \nabla G(z^k)\right]^T\left(z^{k+1} - \bar{z}\right)\|.\end{aligned}$$

Combining the two parts, we obtain $\|z^{k+1} - \bar{z}\| \leq \frac{2L}{\gamma}\|G(\bar{z}) - G(z^k) - \nabla G(z^k)^T\,(\bar{z} - z^k)\| + \frac{L}{\gamma}\|\left[\nabla G(\bar{z}) - \nabla G(z^k)\right]^T\left(z^{k+1} - \bar{z}\right)\|$. Since $G(\cdot) \in C^1$, we have the first term $\|G(\bar{z}) - G(z^k) - \nabla G(z^k)^T\,(\bar{z} - z^k)\| = o(\|z^k - \bar{z}\|)$. Given the fact that as $k \to \infty$, $(\nabla G(\bar{z}) - \nabla G(z^k)) \to 0$, we also have the second term

$$\|\left[\nabla G(\bar{z}) - \nabla G(z^k)\right]^T\left(z^{k+1} - \bar{z}\right)\| = o(\|z^{k+1} - \bar{z}\|).$$

Thus for the unconstrained case, we have $\|z^{k+1} - \bar{z}\| = o(\|z^k - \bar{z}\|)$.

Now, consider the `SCvx` algorithm with trust region constraints. From Lemma 3.3.3, we know that $\rho^k \to 1$, and we have that there exists k' large enough such that $\rho^k > \rho_1$ for all $k \geq k'$, where ρ^k is the ratio defined in (3.9). This means there must be some trust

region radius $r^{k'} > 0$ so that $r^k \geq r^{k'}$ for all $k \geq k'$, because we will not shrink the trust region radius from iteration k' on. On the other hand, from the global convergence result (Theorem 3.3.4), we know that as k increases, $d^k \to 0$. Therefore, there must exist a k', such that $\|d^k\| < r^{k'}$, $\forall\, k \geq k'$, implying that the trust region constraints will eventually become inactive. Hence, the above conclusion holds for the trust-region constrained case as well. □

With Theorem 3.5.1, we have established superlinear convergence rate of the `SCvx` algorithm for a wide range of optimal control problems.

3.6 Numerical Experiments

In this section, we use a non-convex fixed-final-time quad-rotor obstacle avoidance problem to demonstrate the convergence rate of the proposed `SCvx` algorithm. The non-convexity of this problem stems from cylindrical obstacle keep-out zones (see Figure 3.2) and nonlinear aerodynamic drag.

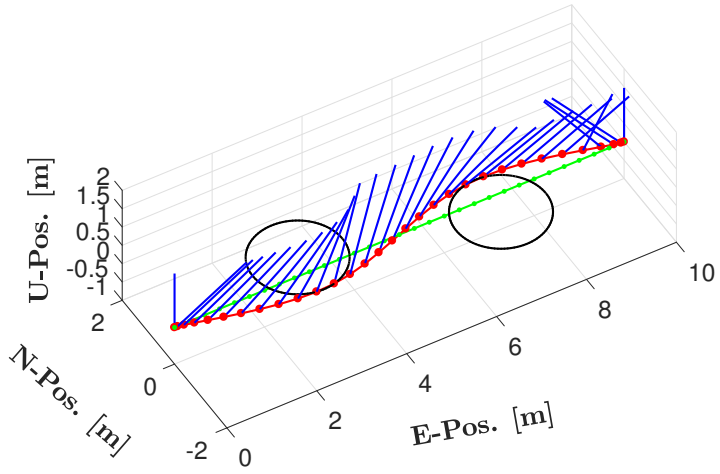


Figure 3.2: *A perspective view of the converged trajectory. The obstacles are represented by the black circles. The red dots and blue lines represent the time discretized positions and thrust vectors, respectively. The green dots represent the initial reference trajectory used to initialize the algorithm (i.e., a straight line). The motion of the vehicle is from left to right.*

We model the quad-rotor using three degree-of-freedom double-integrator translational dynamics [72, 71]. We define the state vector at time t as $x(t) \coloneqq [p^T(t),\ v^T(t)]$, where $p(t) \in \mathbb{R}^3$ and $v(t) \in \mathbb{R}^3$ denote the position and velocity of the vehicle. Likewise, we define the control vector as $u(t) \coloneqq T(t)$, where $T(t) \in \mathbb{R}^3$ is the thrust vector of the vehicle. Additionally, we use $g \in \mathbb{R}^3$ to denote the gravity vector, and $m \in \mathbb{R}_{++}$ and $k_D \in \mathbb{R}_+$ to denote the vehicle's mass and drag constant. With these definitions, the continuous-time dynamics of the system are given by:

$$\dot{x}(t) = f\big(x(t), u(t)\big) \coloneqq \Big[v^T(t), \quad \frac{1}{m}T^T(t) - k_D\|v(t)\|_2 v^T(t) + g^T\Big]^T.$$

To implement this problem numerically, we follow the discretization procedure outlined in [73] to discretize the problem into $N-1$ evenly spaced temporal intervals of duration $\Delta t = t_f/(N-1)$. In this procedure, the system is first linearized about a trajectory $\{\bar{x}, \bar{u}\}$ to obtain

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + z(t), \tag{3.41}$$

where $z(t) \coloneqq f\big(\bar{x}(t), \bar{u}(t)\big) - A(t)\bar{x}(t) - B(t)\bar{u}(t)$.

The first succession evaluates this linearization about a user-specified initialization trajectory, while subsequent successions evaluate the linearization about the solution obtained in the preceding iteration. To generate smoother trajectories, we assume an affine interpolation of $u(t)$ over $t \in [t_i, t_{i+1}]$ for each $i \in \mathcal{I}^- \coloneqq \{1, 2, \ldots, N-1\}$. Thus, $u(t)$ is given by

$$\left.\begin{aligned} u(t) &= \beta^-(t)u_i + \beta^+(t)u_{i+1} \\ \beta^-(t) &\coloneqq (t_{i+1} - t)/\Delta t \\ \beta^+(t) &\coloneqq (t - t_i)/\Delta t \end{aligned}\right\} \begin{array}{l} t \in [t_i, t_{i+1}], \\ \forall i \in \mathcal{I}^-. \end{array}$$

where u_i and u_{i+1} are the temporally-discretized control vectors at the beginning and end of each interval $i \in \mathcal{I}^-$, and are solution variables of the numerical optimization problem solved at each succession. We can now define the following discrete-time equivalents of A, B, and z

from (3.41) for all $i \in \mathcal{I}^-$: $A_{d,i} \coloneqq \Phi_A(t_{i+1}, t_i)$, $B^-_{d,i} \coloneqq \int_{t_i}^{t_{i+1}} \beta^-(\tau)\Phi_A(t_{i+1}, \tau)B(\tau)d\tau$, $B^+_{d,i} \coloneqq \int_{t_i}^{t_{i+1}} \beta^+(\tau)\Phi_A(t_{i+1}, \tau)B(\tau)d\tau$, and $z_{d,i} \coloneqq \int_{t_i}^{t_{i+1}} \Phi_A(t_{i+1}, \tau)z(\tau)d\tau$, where $\Phi_A(\cdot, \cdot)$ is the state transition matrix associated with the dynamics (3.41), and evolves according to the initial-value problem $\dot{\Phi}_A(t, t_i) = A(t)\Phi_A(t, t_i)$, $\Phi_A(t_i, t_i) = I$. At each succession, $A_{d,i}$, $B^-_{d,i}$, $B^+_{d,i}$, and $z_{d,i}$ must be numerically integrated for all $i \in \mathcal{I}_-$. In our experience, the computational burden imposed by this numerical integration is typically negligible compared to that of solving the numerical optimization problems.

Further, we assume that there are n_{obs} stationary cylindrical obstacles, and that the j^{th} obstacle avoidance constraint at the i^{th} time instance is given by $\|p_i - p_{obs,j}\|_2 \geq R_{obs,j}$, $\forall i \in \mathcal{I}$, $\forall j \in \mathcal{J}$, where $\mathcal{I} \coloneqq \{1, 2, \ldots, N\}$, and $\mathcal{J} \coloneqq \{1, 2, \ldots, n_{obs}\}$. To convexify these non-convex state constraints, we linearize them about the reference trajectory: $h_{obs,\bar{x}_i} + H_{obs,\bar{x}_i}(x_i - \bar{x}_i) \geq R_{obs,j}$.

Lastly, our problem definition includes initial conditions denoted by the subscript *ic*; final conditions denoted by the subscript *fc*; minimum and maximum thrust magnitude constraints; and thrust tilt constraints. Notably, the non-convex minimum thrust magnitude constraint can be convexified by introducing the relaxation variable $\Gamma(t) \in \mathbb{R}$, and applying the lossless convexification technique proposed in [3] and employed in [72]. To simplify the presentation of our simulation results, the vehicle is artificially constrained to a constant altitude. However, we emphasize that all three implementations are solving the problem in three-dimensions.

A summary of the convex subproblem solved at each succession is provided in Problem 3.6.1. For notational convenience, we have defined the concatenated solution vector $X \coloneqq [x_0^T, \ldots, x_{N-1}^T, u_0^T, \ldots, u_{N-1}^T]^T$, and use r^0 and r^k to denote the initial and current (i.e. at the $(k+1)^{th}$ iteration) size of the trust region, respectively. As outlined in Section 3.2, the trust region is enforced on the entire solution vector, X.

Problem 3.6.1. Convex Subproblem of Quad-Rotor Obstacle Avoidance

$$\min_{u} \ \sum_{i\in\mathcal{I}} \Gamma_i \Delta t + \lambda \sum_{i\in\mathcal{I}^-} |\nu_i| + \lambda \sum_{j\in\mathcal{J}} \sum_{i\in\mathcal{I}} \max\{0, \eta_{j,i}\}$$

$$\text{subject to: } p_1 = p_{ic},\ v_1 = v_{ic},\ T_1 = T_{ic},\ p_N = p_{fc},\ v_N = v_{fc},\ T_N = T_{fc},$$

$$x_{i+1} = A_{d,i} x_i + B^-_{d,i} u_i + B^+_{d,i} u_{i+1} + z_{d,i} + \nu_i,$$

$$R_{obs,j} - h_{obs,\bar{x}_i} - H_{obs,\bar{x}_i}(x_i - \bar{x}_i) \leq \eta_{j,i},\ \begin{bmatrix}1 & 0 & 0\end{bmatrix} \cdot p_i = 0$$

$$\|T_i\|_2 \leq \Gamma_i,\ T_{min} \leq \Gamma_i \leq T_{max},\ \cos\theta_{max}\Gamma_i \leq \begin{bmatrix}1 & 0 & 0\end{bmatrix} \cdot T_i,\ \|X\|_1 \leq r^k.$$

Algorithm 6 was implemented in MATLAB using `CVX` [30] and `SDPT3` [77]. For comparison, the problem was also implemented using an in-house developed SLP solver, `SNOPT` (an SQP solver) [28] and MATLAB's general purpose `fmincon` IPM solver. Since we lacked sufficient insight into the internal implementation of `SNOPT` and `fmincon`, the results presented here compare the number and values of the iterations obtained by the three implementations, and do not quantify the raw computation times. Nevertheless, our real-time implementations of the `SCvx` algorithm in [72, 71] have yielded promising results. For example, using our in-house IPM solver [24] (whose customized variant was used in recent autonomous rocket landing experiments [23]), aggressive quad-rotor guidance trajectories were computed onboard at rates exceeding 8 Hz.

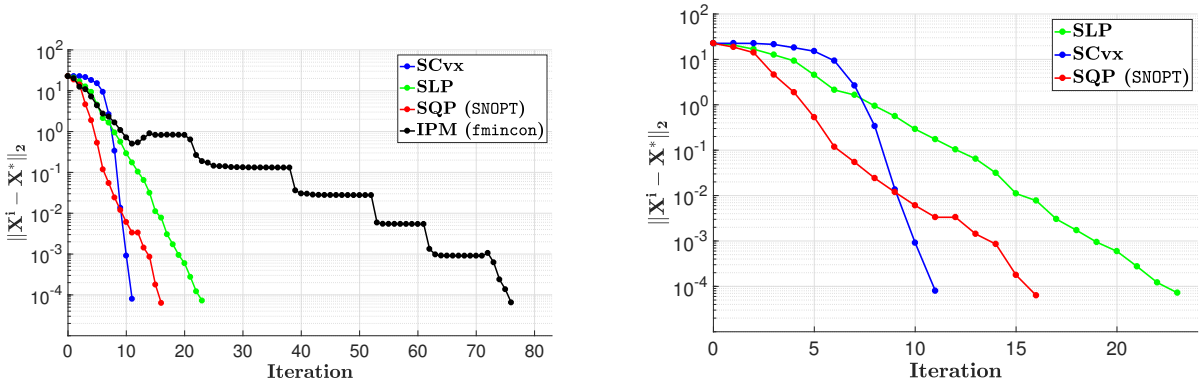
Figure 3.2 shows the trajectory corresponding to the converged solution. The obstacles keep-out zones are shown as the black circles, and the optimized path is shown in red. The red dots indicate each of the N discretization points, and the blue lines represent the thrust vectors at said time points. The large tilt angles generated by the optimization are necessary to counter the drag force dictated by our choice of k_D.

Figure 3.3 shows a comparison between the convergence rates of the three implementations considered here. Convergence was assessed by examining the difference of X across iterations. The same initial guess was used to initialize all three implementations. Our results show that the `SCvx` algorithm initially converges at a slower rate than the other three

implementations, but attains a much faster convergence rate than SLP, `SNOPT`, and `fmincon` after about the 6^{th}, a behavioral curve typical of superlinear convergence. Specifically, `SCvx` converged to a tolerance of 1×10^{-4} in 11 iterations, whereas SLP, `SNOPT`, and `fmincon` achieved the same level of convergence in approximately 23, 16, and 76 iterations, respectively. Notably, `fmincon`'s general purpose IPM performs poorly in solving this problem and struggles to find a descent direction, ultimately taking a large number of iterations to achieve the same level of accuracy.

The first few iterations of the `SCvx` convergence process were spent decreasing the virtual control, ν_i, and virtual buffer zone, $\eta_{j,i}$, terms (see Problem 3.6.1), effectively avoiding artificial infeasibility, and recovering physical feasibility. The remaining iterations were spent refining the trajectory till convergence.

We conclude this section by noting that the `SCvx` algorithm can readily handle free-final-time optimal control problems. For example, the free-final-time version of the problem presented here can be formulated by augmenting the state and dynamics with a time dilation variable as done in [73].



(a) Full convergence history

(b) Zoom-in view without IPM

Figure 3.3: *Convergence history of* ***SCvx*** *(blue), SLP (green),* ***SNOPT*** *(red) and* ***fmincon*** *(black). The lines show the magnitude of the difference between X at each iteration and the converged solution, X^*, and their slopes indicates the rate of convergence.*

Chapter 4

FAST SUCCESSIVE CONVEXIFICATION FOR A CLASS OF NON-CONVEX CONSTRAINTS

This chapter is dedicated to the `SCvx-fast` algorithm, which is a variant of the `SCvx` algorithm, that specializes in handling obstacle avoidance type of problems and is faster to converge than `SCvx`. It uses a *project-and-linearize* procedure (enhanced as a *project-and-convexify* proceure) to handle both the state constraints and the relaxed dynamics. While introducing conservativeness is inevitable in the process, the proposed algorithm preserves much more feasibility than similar results in [64, 43]. One clear advantage of the `SCvx-fast` algorithm is that it does not have to resort to trust regions as `SCvx` does, to guarantee convergence. This property allows the algorithm to potentially take a large step in each succession, thereby greatly accelerating the convergence process, which is exactly the case shown by the numerical simulations.

4.1 Problem Statement

Similar to before, we consider the following discrete-time optimal control problem:

Problem 4.1.1. Discrete-time Optimal Control Problem with Special Constraints

$$\min \quad J(x_i, u_i) := \sum_{i=1}^{T} \phi(x_i, u_i), \tag{4.1a}$$

subject to

$$x_{i+1} - x_i = f(x_i, u_i) \qquad i = 1, 2, T-1, \tag{4.1b}$$

$$h(x_i) \geq 0 \qquad i = 1, 2, T, \tag{4.1c}$$

$$u_i \in U_i \subseteq \mathbb{R}^m \qquad i = 1, 2, T-1, \tag{4.1d}$$

$$x_i \in X_i \subseteq \mathbb{R}^n \qquad i = 1, 2, T. \tag{4.1e}$$

Here, x_i, u_i represent discrete state/control at each temporal point, T denotes the final time, and X_i, U_i are assumed to be convex and compact sets. We also assume that the objective function in (4.1a) is continuous and convex, as is the case in many optimal control applications. For example, the minimum fuel problem has $\phi(x_i, u_i) = \|u_i\|$, and the minimum time problem has $\phi(x_i, u_i) = 1$. Equation (4.1b) represent the system dynamics, where $f(x_i, u_i) \in \mathbb{R}^n$ is, in general, a nonlinear function that is at least twice differentiable. Equation (4.1c) are the additional state constraints, where $h(x_i) \in \mathbb{R}^s$ is also at least twice differentiable, and could be nonlinear as well. Note that we do not impose non-convex control constraints here because we can leverage lossless convexification, see e.g. [3], to convexify them beforehand. Finally, note that (4.1b) and (4.1c) render the problem non-convex.

To reflect the specialty of `SCvx-fast`, we need a few assumptions on $f(x_i, u_i)$ and $h(x_i)$:

Assumption 4.1.1. $f_j(x_i, u_i)$ is a convex function over x_i and u_i, $\forall j = 1, 2, ...n$.

In fact, a wide range of optimal control applications, for example systems with double integrator dynamics and aerodynamic drag (constant speed), satisfy Assumption 4.1.1. It also includes all linear systems. Similarly, we also have

Assumption 4.1.2. $h_j(x_i)$ is a convex function over x_i, $\forall j = 1, 2, ...s$.

An example for Assumption 4.1.2 is collision avoidance constraints, where the shape of each keep-out zone is convex or can be convexly decomposed. See Figure 4.1 for a simple illustration. Note that at this stage, these convexity assumptions do not change the non-convex nature of the problem.

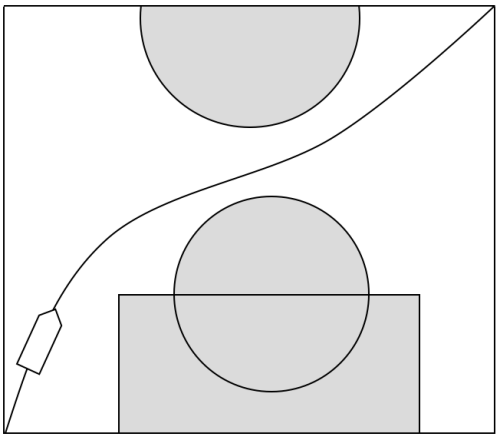


Figure 4.1: *Convex shaped keep-out zone as state constraints*

To convert the optimal control problem into a finite dimensional optimization problem, we treat state variables x_i and control variables u_i as a single variable

$$y = \left(x_1^T, ..., x_T^T, u_1^T, ..., u_{T-1}^T\right)^T \in \mathbb{R}^N,$$

where $N = m(T-1) + nT$. Let Y be the Cartesian product of all X_i and U_i, then $y \in Y$. Y is a convex and compact set because X_i and U_i are. In addition, we let $g_i(x_i, u_i) = f(x_i, u_i) - x_{i+1} + x_i$, and $g(y) = \left(g_1^T, g_2^T, ..., g_{T-1}^T\right)^T \in \mathbb{R}^{n(T-1)}$. Note that each component of $g(y)$ is convex over y by Assumption 4.1.1, and the dynamic equation (3.1b) becomes $g(y) = 0$. We also let $h(y) = \left(h(x_1)^T, h(x_2)^T, ..., h(x_T)^T\right)^T \in \mathbb{R}^{sT}$, then each component of $h(y)$ is convex over y by Assumption 4.1.2, and (3.1c) becomes $h(y) \geq 0$. In summary, we have the following non-convex optimization problem:

$$J(y^*) = \min_y \{J(y) \mid y \in Y, g(y) = 0, h(y) \geq 0\}. \tag{4.2}$$

By leveraging the theory of exact penalty methods, see e.g. [32], [51], we move $g(y) = 0$ into the objective function without compromising optimality:

Theorem 4.1.1 (Exactness). *Let $P(y) = J(y) + \lambda\|g(y)\|_1$ be the penalty function, and $\bar{y}$ be a stationary point of*

$$\min_y\{P(y) \mid y \in Y, g(y) \geq 0, h(y) \geq 0\} \tag{4.3}$$

with Lagrangian multiplier $\bar{\mu}$ for equality constraints. Then, if the penalty weight λ satisfies $\lambda \geq \|\bar{\mu}\|_\infty$, and if $\bar{y}$ is feasible for (4.2), then $\bar{y}$ is a critical point of (4.2).

Since each component of $g(y)$ is convex, and $\|\cdot\|_1$ is convex and nondecreasing due to the constraint $g(y) \geq 0$, then $P(y)$ is a convex function by the composition rule of convex functions, see [14]. This marks our first effort towards the convexification of (4.2).

Let $q(y) = \left(g(y)^T, h(y)^T\right)^T \in \mathbb{R}^M$, where $M = sT + n(T-1)$, then we may rewrite (4.3) as

$$\min_y\{P(y) \mid y \in Y, q(y) \geq 0\} \tag{4.4}$$

where $P(y)$ is continuous and convex, and each component of $q(y)$ is a convex function. By doing this, we are essentially treating constraints due to dynamics as another keep-out zone. Denote

$$F = \{y \mid y \in Y, q(y) \geq 0\}$$

as the feasible set. Note that F is compact but not convex.

4.2 Algorithm Description

First we will introduce the *project-and-linearize* procedure. Since both $f(x_i, u_i)$ and $h(x_i)$ are at least twice differentiable, we have $q(y) \in C^2(Y)$ as well. For any point $z \in F$, let $\nabla_y q(z)$ be the Jacobian matrix of $q(y)$ evaluated at z. Now, if we directly linearize $q(y)$ at z as in [43], there might be a gap between the linearized feasible region and F since z could be in the interior of F. The gap will increase as z moves further away from the boundary $q(y) = 0$. This is not a desirable situation, because a fairly large area of the feasible region is not utilized. In other words, we introduced artificial conservativeness. To address this

issue, we first introduce a *projection* step, which essentially projects z onto each constraint, and obtains each projection point. Then, we linearize each constraint at its own projection point. See Figure 4.2 for an illustration in $\mathbb{R}^2$.

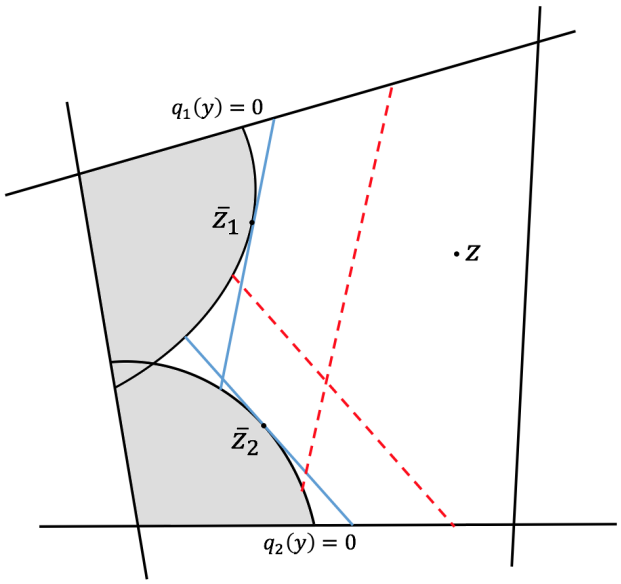


Figure 4.2: *Project-and-linearize vs directly linearize. Note that when the constraint is non-smooth, we pick the supporting hyperplane that is orthogonal to $z - \bar{z}$ as our linearization.*

To formalize, let $q_j(y), j = 1, 2..., M$ represent each component of $q(y)$, i.e. each constraint. Note that $q_j(y)$ is a convex function, hence $q_j(y) \leq 0$ is a closed convex set. Using the well-known Hilbert projection theorem, see e.g.[82], we have

Theorem 4.2.1 (**Uniqueness of the projection**). *For any $z \in F$, there exists a unique point*

$$\bar{z}_j = \underset{y}{argmin} \left\{ \|z - y\|_2 \mid q_j(y) \leq 0 \right\}, \tag{4.5}$$

called the projection of z onto $q_j(y) \leq 0$.

Equation (4.5) is a simple convex program of low dimension that can be solved quickly (sub-milliseconds) using any convex programming solver. Alternatively, for some special

convex sets (e.g. cylinders), (4.5) can be solved analytically, which is even faster. Doing this for each constraint, we obtain a set of projection points, $\{\bar{z}_1, \bar{z}_2, ..., \bar{z}_M\}$. Note that these projection points must lie on the boundary of $q_j(y) \leq 0$, i.e. $q_j(\bar{z}_j) = 0,\ \forall\ j = 1, ..., M$. For a fixed $z \in F$, let $l_j(y, z)$ be the linear approximation of $q_j(y)$:

$$l_j(y, z) = \nabla_y q_j(\bar{z}_j)(y - \bar{z}_j), \tag{4.6}$$

and let $l(y, z) = (l_1^T, l_2^T, ..., l_M^T)^T \in \mathbb{R}^M$. For each $z \in F$, denote

$$F_z = \{y \mid y \in Y,\, l(y, z) \geq 0\}$$

as the feasible region after linearization. F_z also defines a point-to-set mapping, $F_z : z \to F_z$. Note that each component of $l(y, z) \geq 0$, i.e. $l_j(y, z) \geq 0$ represents a half-space. Hence $l(y, z) \geq 0$ is the intersection of half-spaces, which means F_z is a convex and compact set.

Remark. Convexification of F by using F_z also inevitably introduces conservativeness, but one can verify that it is the best we can do to maximize feasibility while preserving convexity.

The following lemma gives an invariance result regarding the point-to-set mapping F_z. It is essential to our subsequent analyses.

Lemma 4.2.1 (Invariance of F_z). *For each $z \in F$, we have $z \in F_z \subseteq F$.*

Proof. For each $z \in F$, and $\forall j = 1, 2, ..., M$, from (4.6), we have

$$l_j(z, z) = \nabla_y q_j(\bar{z}_j)(z - \bar{z}_j).$$

Since $\bar{z}_j$ is the projection, $(z - \bar{z}_j)$ is the normal vector at $\bar{z}_j$, which is aligned with the gradient $\nabla_y q_j(\bar{z}_j)$. Hence $\nabla_y q_j(\bar{z}_j)(z - \bar{z}_j) \geq 0$, i.e. $l_j(z, z) \geq 0, \forall j = 1, 2, ..., M$, i.e. $z \in F_z$.

Furthermore, since $q_j(y)$ is a convex function, we have for any $y \in F_z$,

$$q_j(y) \geq q_j(\bar{z}_j) + \nabla_y q_j(\bar{z}_j)(y - \bar{z}_j) = l_j(y, z) \geq 0,$$

which means $y \in F$. Hence $F_z \subseteq F$. $\square$

Now that we have a convex and compact feasible region F_z and a convex objective function $P(y)$, we are ready to present a successive procedure to solve the non-convex problem in (4.4). Note that the feasible region F_z is defined by z. Therefore, if we start from a point $z^{(0)} \in F$, a sequence $\{z^{(k)}\}$ will be generated, where

$$z^{(k+1)} = \underset{y}{argmin}\{P(y) \mid y \in F_{z^{(k)}}\}, \quad k = 0, 1, ... \tag{4.7}$$

This is a convex programming subproblem, whose global minimizer is attained at $z^{(k+1)}$. At these intermediate steps, $z^{(k+1)}$ may not be the optimal solution to (4.4). Our goal, however, is to prove that this sequence $\{z^{(k)}\}$ converges to a limit point z^*, and that this limit point solves (4.4) by *project-and-linearize* at z^* itself, i.e., it is a "fixed-point" satisfying

$$z^* = \underset{y}{argmin}\{P(y) \mid y \in F_{z^*}\}. \tag{4.8}$$

More importantly, we want to show that z^* gives a local optimum to (4.4) convexified at z^* itself. Then by solving a sequence of convex programming subproblems, we effectively solved the non-convex optimal control problem in (4.2) thanks to Theorem 4.1.1. Notably, we do not use trust regions in this process, hence no trust-region updating mechanism involved. As a result, this procedure converges much faster than `SCvx`, and thus we call it the `SCvx-fast` algorithm. It is summarized in Algorithm 7.

4.3 Global Convergence

In this section, we proceed to show that Algorithm 6 does converge to a point z^* that indeed satisfies (4.8). First we must assume the application of regular constraint qualifications, namely the Linear Independence Constraint Qualification (LICQ) and the Slater's condition. They can be formalized as the following:

Algorithm 7 The `SCvx-fast` Algorithm

procedure SCvx-fast($z^{(0)}, \lambda$)
  input Initial point $z^{(0)} \in F$. Penalty weight $\lambda \geq 0$.
  while not converged **do**
    step 1 Project At each succession k, we have the current point $z^{(k)}$. For each constraint $q_j(y)$, solve the problem in (4.5) to get a projection point $z_j^{(\overline{k})}$.
    step 2 Linearize Construct the convex feasible region $F_{z^{(k)}}$ by using (4.6).
    step 3 Solve the convex subproblem in (4.7) to get $z^{(k+1)}$. Let $z^{(k)} \leftarrow z^{(k+1)}$ and go to the next succession.
  end while
  return $z^{(k+1)}$.
end procedure

Assumption 4.3.1 (**LICQ**)**.** For each $z \in F$, the Jacobian matrix $\nabla_y q(z)$ has full row rank, i.e. $\text{rank}(\nabla_y q(z)) = M$.

Assumption 4.3.2 (**Slater's condition**)**.** For each $z \in F$, the convex feasible region F_z contains interior points.

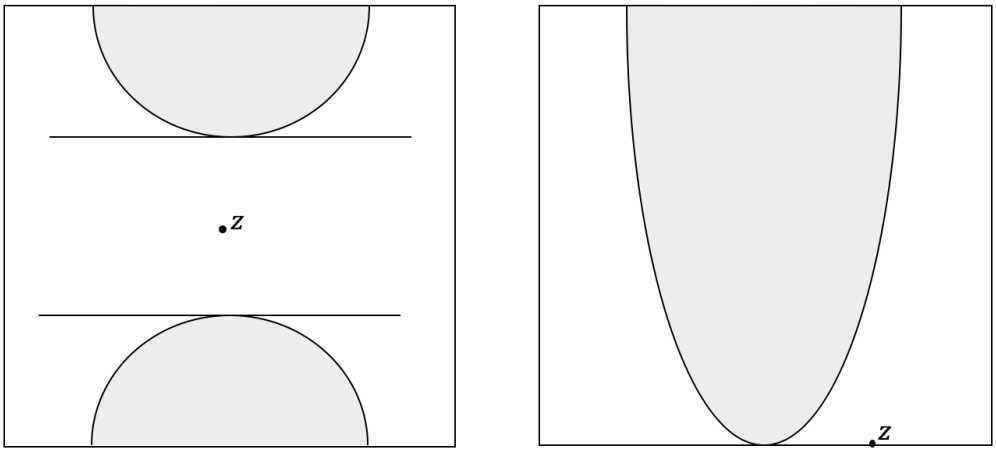
These assumptions do impose some practical restrictions on our feasible region. Figure 4.3 shows some examples where LICQ or Slater's condition might fail. For scenarios like (a), we may perturb our discrete points to break symmetry, For scenarios like (b), we have to assume the connectivity of the feasible region. In other words, our feasible region cannot be degenerate at some point, for example, in collision avoidance the feasible region is not allowed to be completely obstructed.

To analyze convergence, first we need to show that the point-to-set mapping F_z is continuous in the sense that given any point $z^{(1)} \in F$ and $y^{(1)} \in F_{z^{(1)}}$, then for any point $z^{(2)} \in F$ in the neighborhood of $z^{(1)}$, there exists a $y^{(2)} \in F_{z^{(2)}}$ that is close enough to $y^{(1)}$.

First, We have the following lemma:

Lemma 4.3.1 (**Lipschitz continuity of** $l(y, z)$)**.** *The linear approximation* $l(y, z)$ *in* (4.6) *is Lipschitz continuous in* z*, that is*

$$\|l(y, z^{(1)}) - l(y, z^{(2)})\| \leq \gamma \|z^{(1)} - z^{(2)}\|, \tag{4.9}$$



(a) Not full row rank (b) Not have interior point

Figure 4.3: *Cases where LICQ or Slater's condition fails*

for any $z^{(1)}, z^{(2)} \in F$ *and constant* γ.

Proof. Each $l_j(y, z)$ is in fact a composition of two mappings. The first mapping maps z to its projection $\bar{z}_j$. This mapping is defined by the optimization problem in (4.5). It is a well-known result that this mapping is non-expansive (i.e. Lipschitz continuous with constant 1), See e.g. [82]. The second mapping is defined by the auxiliary function:

$$a_j(y, \bar{z}_j) = \nabla_y q_j(\bar{z}_j)(y - \bar{z}_j),$$

Since $q(y) \in C^2(Y)$, $a_j(y, \bar{z}_j)$ is Lipschitz continuous in $\bar{z}_j$. By the composition rule of two Lipschitz continuous functions, we have $l_j(y, z)$ is Lipschitz continuous, for all $j = 1, 2, ..., M$. Therefore, sum over j gives the Lipschitz continuity of $l(y, z)$. □

Now with Assumptions 4.3.1, 4.3.2 and Lemma 4.3.1, we are ready to prove the continuity of point-to-set mapping F_z. The result is given as follows:

Lemma 4.3.2 (**Continuity of mapping** F_z). *Given* $z^{(1)} \in F$ *and* $y^{(1)} \in F_{z^{(1)}}$, *then given* $\epsilon > 0$, *there exists a* $\delta > 0$ *so that for any point* $z^{(2)} \in F$ *with* $\|z^{(2)} - z^{(1)}\| \leq \delta$, *there exists*

a $y^{(2)} \in F_{z^{(2)}}$ *such that* $\|y^{(2)} - y^{(1)}\| \leq \epsilon$.

Proof. From Assumption 4.3.1, we know that the Jacobian matrix $\nabla_y q(z)$ has full row rank. Thus matrix $\nabla_y q(z) \nabla_y q(z)^T$ is symmetric and positive definite for any $z \in F$. Consequently, there exists β such that

$$\|(\nabla_y q(z) \nabla_y q(z)^T)^{-1}\| \leq \beta^2, \quad \forall z \in F. \tag{4.10}$$

If $y^{(1)} \in F_{z^{(2)}}$, then take $y^{(2)} = y^{(1)}$ such that $\|y^{(2)} - y^{(1)}\| = 0 \leq \epsilon$. Now suppose $y^{(1)} \notin F_{z^{(2)}}$, then there exists at least one j, such that $l_j(y^{(1)}, z^{(2)}) < 0$. Let

$$\bar{l}_j = \begin{cases} l_j(y^{(1)}, z^{(2)}) & l_j(y^{(1)}, z^{(2)}) < 0, \\ 0 & l_j(y^{(1)}, z^{(2)}) \geq 0. \end{cases}$$

Note that $l_j(y^{(1)}, z^{(1)}) \geq 0$, then by definition,

$$|\bar{l}_j| \leq |l_j(y^{(1)}, z^{(1)}) - l_j(y^{(1)}, z^{(2)})|, \quad \forall j = 1, 2, ..., M.$$

Hence we have

$$\|\bar{l}\| \leq \|l(y^{(1)}, z^{(1)}) - l(y^{(1)}, z^{(2)})\| \leq \gamma \|z^{(1)} - z^{(2)}\|. \tag{4.11}$$

The second inequality follows from (4.9) in Lemma 4.3.1.

Now we consider two cases:

Case 1: $y^{(1)}$ is an interior point of Y, then $\exists\, \epsilon_1 \in (0, \epsilon]$, such that $y \in Y$ for all y satisfies $\|y - y^{(1)}\| \leq \epsilon_1$.

Let $\|z^{(2)} - z^{(1)}\| \leq \delta$, with $\delta = \epsilon_1 / \beta\gamma$, and let

$$y^{(2)} = y^{(1)} - \nabla_y q(z^{(2)})^T (\nabla_y q(z^{(2)}) \nabla_y q(z^{(2)})^T)^{-1} \bar{l}. \tag{4.12}$$

We need to verify that $y^{(2)} \in F_{z^{(2)}}$. First, we have

$$l_j(y^{(2)}, z^{(2)}) = \nabla_y q_j(\bar{z}_j^{(2)})(y^{(2)} - \bar{z}_j^{(2)}). \tag{4.13}$$

Substitute $y^{(2)}$ from (4.12) into the stacked form of (4.13), and then unstack, we get

$$\begin{aligned} l_j(y^{(2)}, z^{(2)}) &= \nabla_y q_j(\bar{z}_j^{(2)})(y^{(1)} - \bar{z}_j^{(2)}) - \bar{l}_j \\ &= l_j(y^{(1)}, z^{(2)}) - \bar{l}_j \\ &\geq 0. \end{aligned}$$

The last inequality follows from the definition of $\bar{l}_j$. Therefore, $l(y^{(2)}, z^{(2)}) \geq 0$.
Next we need to show that $y^{(2)} \in Y$. Rearrange terms in (4.12), we have

$$\begin{aligned} \|y^{(2)} - y^{(1)}\|^2 &= \bar{l}^T(\nabla_y q(z^{(2)}) \nabla_y q(z^{(2)})^T)^{-1} \bar{l} \\ &\leq \beta^2 \|\bar{l}\|^2 \\ &\leq \beta^2 \gamma^2 \|z^{(1)} - z^{(2)}\|^2 \end{aligned}$$

The inequalities follow from (4.10) and (4.11). Thus we have $\|y^{(2)} - y^{(1)}\| \leq \beta\gamma\delta = \epsilon_1$, i.e. $y^{(2)} \in Y$. So now we have verified $y^{(2)} \in F_{z^{(2)}}$. Since $\epsilon_1 \leq \epsilon$, we have $\|y^{(2)} - y^{(1)}\| \leq \epsilon$, so that F_z is continuous.

Case 2: $y^{(1)}$ is a boundary point of Y. From Assumption 4.3.2, $F_{z^{(1)}}$ has interior points. Also since $F_{z^{(1)}}$ is a convex set, it is a well known fact that there are interior points in the $\epsilon-$Neighborhood of every point in $F_{z^{(1)}}$. Then we can apply the same argument as in Case 1, to get that F_z is continuous. □

One way to show convergence is to demonstrate the convergence of objective functions, $P(z^{(k)})$. Now let's define

$$\Phi(z) \coloneqq \min_y \{P(y) \mid y \in F_z\}$$

to be the function that maps the point z we are solving at in each iteration to the optimal value of the objective function in F_z. By using the continuity of the point-to-set mapping F_z, the following lemma gives the continuity of the function $\Phi(z)$.

Lemma 4.3.3. *$\Phi(z)$ is continuous for $z \in F$.*

Proof. Given any two points $z^{(1)}, z^{(2)} \in F$, and they are close to each other, i.e. $\|z^{(1)} - z^{(2)}\| \leq \delta$. Let

$$y^{(1)} = \underset{y}{argmin}\{P(y) \mid y \in F_{z^{(1)}}\} \text{ and}$$
$$y^{(2)} = \underset{y}{argmin}\{P(y) \mid y \in F_{z^{(2)}}\},$$

then we have $\Phi(z^{(1)}) = P(y^{(1)})$ and $\Phi(z^{(2)}) = P(y^{(2)})$.
Without loss of generality, let $\Phi(z^{(1)}) \leq \Phi(z^{(2)})$, i.e. $P(y^{(1)}) \leq P(y^{(2)})$. From Lemma 4.3.2, the continuity of F_z, there exists $\hat{y}^{(2)} \in F_{z^{(2)}}$, such that $\|\hat{y}^{(2)} - y^{(1)}\| \leq \eta$ for any $\eta > 0$. Then since $P(y)$ is continuous, we have

$$|P(\hat{y}^{(2)}) - P(y^{(1)})| \leq \epsilon. \tag{4.14}$$

Since $y^{(2)}$ is the minimizer of $P(y)$ in $F_{z^{(2)}}$, $P(y^{(2)}) \leq P(\hat{y}^{(2)})$. Therefore, $P(y^{(1)}) \leq P(\hat{y}^{(2)})$ by assumption. Now (4.14) becomes $P(\hat{y}^{(2)}) - P(y^{(1)}) \leq \epsilon$. Again, because $P(y^{(2)}) \leq P(\hat{y}^{(2)})$, we have $P(y^{(2)}) - P(y^{(1)}) \leq \epsilon$, i.e. $\Phi(z^{(2)}) - \Phi(z^{(1)}) \leq \epsilon$, which means $\Phi(z)$ is continuous. □

With the continuity of $\Phi(z)$, we are finally ready to present the final convergence results:

Theorem 4.3.1 (**Global convergence**). *Under Assumptions 4.1.1, 4.1.2, 4.3.1, and 4.3.2, the sequence $\{z^{(k)}\}$ generated by the successive procedure* (4.7) *is in F, and has limit point z^*, at which the corresponding sequence $\{P(z^{(k)})\}$ attains its minimum, $P(z^*)$.*
More importantly, $P(z^*) = \Phi(z^*)$, i.e. z^* is a local optimum of the penalty problem in (4.4) *convexified at z^*.*

Proof. From Lemma 4.2.1, we have $z^{(k)} \in F_{z^{(k)}} \subseteq F$, then

$$P(z^{(k+1)}) = \min_y \{P(y) \mid y \in F_{z^{(k)}}\} \leq P(z^{(k)}),$$

because $z^{(k)}$ is a feasible point to this convex optimization problem, while $z^{(k+1)}$ is the optimum. Therefore, the sequence $\{P(z^{(k)})\}$ is monotonically decreasing.
Also since $F_{z^{(k)}} \subseteq F$ for all k, we have

$$P(z^{(k+1)}) \geq \min_y \{P(y) \mid y \in F\},$$

which means the sequence $\{P(z^{(k)})\}$ is bounded from below. Then by the monotone convergence theorem, see e.g. [65], $\{P(z^{(k)})\}$ converges to its infimum. Due to the compactness of $F_{z^{(k)}}$, this infimum is attained by all the convergent subsequences of $\{z^{(k)}\}$. Let $z^* \in F$ be one of the limit points, then $\{P(z^{(k)})\}$ attains its minimum at $P(z^*)$, i.e.

$$P(z^*) \leq P(z^{(k)}), \quad \forall\, k = 0, 1, ... \tag{4.15}$$

To show $P(z^*) = \Phi(z^*)$, we note that since $z^* \in F$, $z^* \in F_{z^*}$ by Lemma 4.2.1. Thus we have

$$\Phi(z^*) \coloneqq \min_y \{P(y) \mid y \in F_{z^*}\} \leq P(z^*).$$

Now for the sake of contradiction, we suppose $\Phi(z^*) < P(z^*)$. Then by Lemma 4.3.3, the continuity of $\Phi(z)$, there exists a sufficiently large k such that $\Phi(z^k) < P(z^*)$. Then we have $P(z^{(k+1)}) = \Phi(z^k) < P(z^*)$, which contradicts (4.15). Therefore $P(z^*) = \Phi(z^*)$ holds. □

4.4 Enhanced `SCvx-fast` *Algorithm*

In the section, we describe the enhancement made to the `SCvx-fast` algorithm. The extension is in three folds as follows. i) We can now initialize the algorithm from an infeasible starting point, and regain feasibility in just one step; ii) We get rid of the smoothness condi-

tions on the constraints so that a broader range of "obstacles" can be included. Significant changes are made to adjust the algorithm accordingly; iii) We obtain a proof of superlinear rate of convergence, a new theoretical result for `SCvx-fast`. Additonal numerical simulations are performed to validate the enhanced version of `SCvx-fast` and reaffirm the fast convergence rate.

4.4.1 Infeasible Initialization

Previously in [52], we require a feasible starting point. Now we propose a preprocessing routine that linearize the violating constraints at the starting point, which will get us out of the infeasible region in just one step. See Algorithm 8 for details and Figure 4.4 for an geometric illustration of this procedure (in 2-D case).

Algorithm 8 The Infeasible Initialization Routine

procedure INFEASIBLE-INITIALIZATION($z^{(0)}$)
  input Any point $z^{(0)} \notin F$.
  step 1 Approximate any intersecting constraints with their minimum volume ellipsoidal cover as the new constraint.
  step 2 Identify any infeasible constraint $h_i(z_i^{(0)}) < c$, where c represents all the constant parts of that constraint.
  step 3 Linearize $h_i(y)$ with respect to $z_i^{(0)}$ as $l_{z_i^{(0)}}(y)$.
  step 4 Project $z^{(0)}$ onto the half-space intersected by all the $l_{z_i^{(0)}}(y) = c$ to get a new point $\tilde{z}_i^{(0)}$.
  return $\tilde{z}_i^{(0)}$ as the new starting point.
end procedure

Theorem 4.4.1 (**Feasibility of** $\tilde{z}^{(0)}$)**.** *Given Assumption 4.1.2, Algorihtm 8 produces a feasible initial trajectory in terms of constraints, i.e.* $\tilde{z}^{(0)} \in F$.

Proof. By the convexity of $h(y)$ from Assumption 4.1.2, we know that its epigraph $epi(h) = \{(y, w) \,|\, y \in Y, w \in \mathbb{R}, h(y) \leq w\}$ is a convex set. Denote the projection point of $z^{(0)}$ onto $epi(h)$ as z_p, then $l_{z^{(0)}}(y)$ is the supporting hyperplane of $epi(h)$ at z_p. Thanks to the

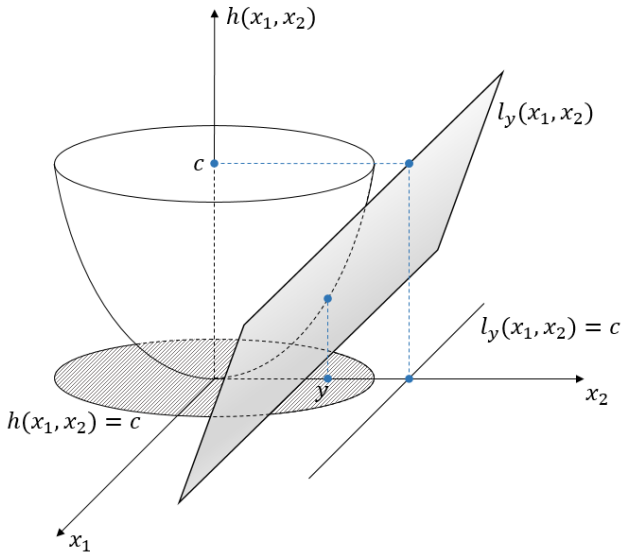


Figure 4.4: *Initialize the algorithm at an infeasible point y. The first linearization will get us to the feasible region $l_y > c$.*

separating hyperplane theorem [14], we have $l_{z^{(0)}}(y) = c$ outside of $epi(h)$, thus $\tilde{z}^{(0)}$, as the projection of $z^{(0)}$, will enjoy the property $h(\tilde{z}^{(0)}) \geq c$, i.e. $\tilde{z}^{(0)} \in F$. □

4.4.2 Project-and-Convexify and the Enhanced `SCvx-fast` *Algorithm*

Next we will introduce the *project-and-convexify* procedure. For any point $z \in F$, let $\tilde{\nabla}_y q(z)$ be the generalized Jacobian matrix of $q(y)$ evaluated at z. Again, in order to avoid artificial conservativeness, we first introduce a *projection* step, which essentially projects z onto each constraint, and obtains each projection point. Then, we linearize each constraint at its own projection point. See Figure 4.2 for an illustration in $\mathbb{R}^2$. Note that here when the constraint function q_j is not differentiable everywhere, $\tilde{\nabla}_y q_j(\bar{z}_j)$ denote the gradient to the supporting hyperplane orthogonal to $z - \bar{z}_j$. Including these modifications, we get a enhanced version of the `SCvx-fast` algorithm, Algorithm 9.

Algorithm 9 The Enhanced `SCvx-fast` Algorithm

procedure SCvx-FAST($z^{(0)}, \lambda$)
  input Initial point $z^{(0)}$. Penalty weight $\lambda \geq 0$.
  if $z^{(0)} \notin F$ **then**
    Infeasibility Preprocessing Execute Algorithm 8 to get a new $\tilde{z}^{(0)} \in F$.
  end if
  while not converged **do**
    step 1 Project At each succession k, we have the current point $z^{(k)}$. For each constraint $q_j(y)$, solve the problem in (4.5) to get a projection point $\bar{z}_j^{(k)}$.
    step 2 Convexify Construct the convex feasible region $F_{z^{(k)}}$ by using (4.6).
    step 3 Solve the convex subproblem in (4.7) to get $z^{(k+1)}$. Let $z^{(k)} \leftarrow z^{(k+1)}$ and go to the next succession.
  end while
  return $z^{(k+1)}$.
end procedure

4.4.3 Superlinear convergence rate

Aside from enhancements made to the `SCvx-fast` algorithm itself, we also obtained a new convergence result quantifying its rate of convergence. We first denote the inverse mapping of F_z as $\bar{F}_z : F_z \to z$, which is a set-to-point mapping. Similar to Lemma 4.3.2, we will first show the continuity of $\bar{F}_z$.

Lemma 4.4.1 (**Lipschitz continuity of** $l(y, z)$)**.** *The linear approximation* $l(y, z)$ *in* (4.6) *is Lipschitz continuous in* z*, that is*

$$\|l(y, z^{(1)}) - l(y, z^{(2)})\| \geq \bar{\gamma}\|z^{(1)} - z^{(2)}\|, \tag{4.16}$$

for any $z^{(1)}, z^{(2)} \in F$ *and constant* $\bar{\gamma}$*.*

Proof. Similar to Lemma 4.3.1. □

Lemma 4.4.2 (**Continuity of mapping** $\bar{F}_z$)**.** *Given* $z^{(1)}, z^{(2)} \in F$ *and* $\epsilon > 0, \delta > 0$*, if for any* $y^{(1)} \in F_{z^{(1)}}$*, there exists* $y^{(2)} \in F_{z^{(2)}}$ *such that* $\|y^{(2)} - y^{(1)}\| \leq \epsilon$*, then* $\|z^{(2)} - z^{(1)}\| \leq \delta$*.*

Proof. If $y^{(1)} \in F_{z^{(2)}}$, then take $y^{(2)} = y^{(1)}$ such that $\|y^{(2)} - y^{(1)}\| = 0 \leq \epsilon$. Now suppose $y^{(1)} \notin F_{z^{(2)}}$, then there exists at least one j, such that $l_j(y^{(1)}, z^{(2)}) < 0$. Let

$$\bar{l}_j = \begin{cases} l_j(y^{(1)}, z^{(2)}) & l_j(y^{(1)}, z^{(2)}) < 0, \\ 0 & l_j(y^{(1)}, z^{(2)}) \geq 0. \end{cases}$$

Note that $l_j(y^{(1)}, z^{(1)}) \geq 0$, then by construction,

$$|\bar{l}_j| \geq |l_j(y^{(1)}, z^{(1)}) - l_j(y^{(1)}, z^{(2)})|, \quad \forall j = 1, 2, ..., M.$$

Hence we have

$$\|\bar{l}\| \geq \|l(y^{(1)}, z^{(1)}) - l(y^{(1)}, z^{(2)})\| \geq \bar{\gamma}\|z^{(1)} - z^{(2)}\|. \tag{4.17}$$

The second inequality follows from (4.16) in Lemma 4.4.1.

Now we consider two cases:

Case 1: $y^{(1)}$ is an interior point of Y, then $\exists\, \epsilon_1 \in (0, \epsilon]$, such that $y \in Y$ for all y satisfies $\|y - y^{(1)}\| \leq \epsilon_1$.

Let $\delta = \epsilon_1/\beta\gamma$, and let

$$y^{(2)} = y^{(1)} - \tilde{\nabla}_y q(z^{(2)})^T (\tilde{\nabla}_y q(z^{(2)}) \tilde{\nabla}_y q(z^{(2)})^T)^{-1} \bar{l}. \tag{4.18}$$

We need to verify that $y^{(2)} \in F_{z^{(2)}}$. First, we have

$$l_j(y^{(2)}, z^{(2)}) = \tilde{\nabla}_y q_j(\bar{z}_j^{(2)})(y^{(2)} - \bar{z}_j^{(2)}). \tag{4.19}$$

Substitute $y^{(2)}$ from (4.18) into the stacked form of (4.19), and then unstack, we get

$$\begin{aligned} l_j(y^{(2)}, z^{(2)}) &= \tilde{\nabla}_y q_j(\bar{z}_j^{(2)})(y^{(1)} - \bar{z}_j^{(2)}) - \bar{l}_j \\ &= l_j(y^{(1)}, z^{(2)}) - \bar{l}_j \\ &\geq 0. \end{aligned}$$

The last inequality follows from the definition of $\bar{l}_j$. Therefore, $l(y^{(2)}, z^{(2)}) \geq 0$.
Next we need to show that $y^{(2)} \in Y$. Rearrange terms in (4.18), we have

$$\begin{aligned} \|y^{(2)} - y^{(1)}\|^2 &= \bar{l}^T(\tilde{\nabla}_y q(z^{(2)})\tilde{\nabla}_y q(z^{(2)})^T)^{-1}\bar{l} \\ &\geq \beta^2\|\bar{l}\|^2 \\ &\geq \beta^2\gamma^2\|z^{(1)} - z^{(2)}\|^2 \end{aligned}$$

The inequalities follow from (4.10) and (4.17). Thus we have $\|y^{(2)} - y^{(1)}\| \leq \beta\gamma\delta = \epsilon_1$, i.e. $y^{(2)} \in Y$. So now we have verified $y^{(2)} \in F_{z^{(2)}}$. Since $\epsilon_1 \leq \epsilon$, we have $\|z^{(2)} - z^{(1)}\| \leq \delta$, so that $\bar{F}_z$ is continuous.

Case 2: $y^{(1)}$ is a boundary point of Y. From Assumption 4.3.2, $F_{z^{(1)}}$ has interior points. Also since $F_{z^{(1)}}$ is a convex set, it is a well known fact that there are interior points in the $\epsilon-$Neighborhood of every point in $F_{z^{(1)}}$. Then we can apply the same argument as in Case 1, to get that $\bar{F}_z$ is continuous. □

By using the continuity of the set-to-point mapping $\bar{F}_z$, we have the following lemma:

Lemma 4.4.3. *Given $z^{(1)}, z^{(2)} \in F$ and $\gamma > 0$, we have*

$$P(y^{(2)}) - P(y^{(1)}) \geq \gamma\|z^{(1)} - z^{(2)}\|. \tag{4.20}$$

Proof. Given any two points $z^{(1)}, z^{(2)} \in F$, and they are close to each other, i.e. $\|z^{(1)} - z^{(2)}\| \leq$

δ. Then let

$$y^{(1)} = \underset{y}{argmin}\{P(y) \mid y \in F_{z^{(1)}}\} \text{ and}$$
$$y^{(2)} = \underset{y}{argmin}\{P(y) \mid y \in F_{z^{(2)}}\},$$

so that we have $\Phi(z^{(1)}) = P(y^{(1)})$ and $\Phi(z^{(2)}) = P(y^{(2)})$.
Without loss of generality, let $\Phi(z^{(1)}) \leq \Phi(z^{(2)})$, i.e. $P(y^{(1)}) \leq P(y^{(2)})$. From Lemma 4.4.2, the continuity of $\bar{F}_z$, there exists $\hat{y}^{(2)} \in F_{z^{(2)}}$, such that $\|\hat{y}^{(2)} - y^{(1)}\| \leq \eta$ for any $\eta > 0$. Then since $P(y)$ is continuous, we have

$$|P(\hat{y}^{(2)}) - P(y^{(1)})| \leq \epsilon. \tag{4.21}$$

Since $y^{(2)}$ is the minimizer of $P(y)$ in $F_{z^{(2)}}$, $P(y^{(2)}) \leq P(\hat{y}^{(2)})$. Therefore, $P(y^{(1)}) \leq P(\hat{y}^{(2)})$ by assumption. Now (4.14) becomes $P(\hat{y}^{(2)}) - P(y^{(1)}) \leq \epsilon$. Again, because $P(y^{(2)}) \leq P(\hat{y}^{(2)})$, we have $P(y^{(2)}) - P(y^{(1)}) \leq \epsilon$, i.e. $\Phi(z^{(2)}) - \Phi(z^{(1)}) \leq \epsilon$, which means $\Phi(z)$ is continuous. $\square$

Lemma 3.5.3 provides an important condition that lower bounds the cost reduction rate. Next we show that given this condition, the `SCvx-fast` procedure will indeed converge superlinearly. To proceed, first denote the stack of $J(\cdot)$ and $g(\cdot)$ as $G(\cdot)$, and represent $P(\cdot)$ by a function composition $\psi(G(\cdot))$. Qualitatively we can write $\psi(G(\cdot)) = G_{cost}(\cdot) + \|G_{eq}(\cdot)\|_1$. Note that $\psi(\cdot)$ is convex since $\|\cdot\|_1$ is a convex function.

Lemma 4.4.4. *Following Lemma 3.5.3, there exists $\gamma > 0$, such that*

$$\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d\right) \geq \psi(G(\bar{z})) + \gamma\|d\|, \quad \forall\, d \in F_{\bar{z}}. \tag{4.22}$$

Proof. First we show that the statement is true for any small step d_γ such that $\|d_\gamma\| \leq \gamma$, where γ is defined in Lemma 3.5.3. We have $\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d_\gamma\right) - \psi(G(\bar{z})) = \psi(G(\bar{z} + d_\gamma)) - \psi(G(\bar{z})) + \psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d_\gamma\right) - \psi(G(\bar{z} + d_\gamma)) \geq \beta\|d_\gamma\| + o(\|d_\gamma\|) \geq \frac{\beta}{2}\|d_\gamma\|$. Let $\gamma := \frac{\beta}{2}$, then we have (4.22) hold for a small step d_γ. Note that the first inequality is due

to Lemma 3.5.3 and again the fact that $o(\|d_\gamma\|)$ can be taken out of $|\cdot|$ and $\max(0, \cdot)$ and $G(\cdot) \in C^1$.

Now to generalize this to any $d \in F_{\bar{z}}, d \neq 0$, we first define $\zeta := \min\left(1, \gamma/\|d\|\right)$, and let $z_\zeta := \bar{z} + \zeta d$. Denoting $(z_\zeta - \bar{z})$ as d_ζ, we have $d_\zeta = \zeta\, d$. With this definition of ζ, one can verify that $\|d_\zeta\| \leq \gamma$. Hence, (4.22) holds true for d_ζ. Therefore, we have

$$\begin{aligned}
\gamma\zeta\|d\| = \gamma\|d_\zeta\| &\leq \psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d_\zeta\right) - \psi(G(\bar{z})) \\
&= \psi\left(G(\bar{z}) + \zeta\nabla G(\bar{z})^T d\right) - \psi(G(\bar{z})) \\
&= \psi\left((1-\zeta)G(\bar{z}) + \zeta\left[G(\bar{z}) + \nabla G(\bar{z})^T d\right]\right) - \psi(G(\bar{z})) \\
&\leq (1-\zeta)\psi(G(\bar{z})) + \zeta\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d\right) - \psi(G(\bar{z})) \\
&= \zeta\left[\psi\left(G(\bar{z}) + \nabla G(\bar{z})^T d\right) - \psi(G(\bar{z}))\right].
\end{aligned}$$

The second inequality is due to the convexity of ψ (note that $0 \leq \zeta \leq 1$). Dividing both sides by ζ, we obtain (4.22). □

Theorem 4.4.2 (**Superlinear Convergence**). *Given a sequence $\{z^k\} \to \bar{z}$ generated by the* `SCvx-fast` *algorithm (Algorithm 7) and suppose that Assumption 4.3.1, 4.3.2, 4.1.1, and 4.1.2 are satisfied, then we have $\|y^{k+1} - \bar{y}\| = o(\|y^k - \bar{y}\|)$.*

Proof. Since $d^k := y^{k+1} - y^k$ is the optimal solution to the convex subproblem within $F_{\bar{z}}$, we have $\forall\, d \in F_{\bar{z}}$, $\psi\big(G(y^k) + \nabla G(y^k)^T d\big) \geq \psi\big(G(y^k) + \nabla G(y^k)^T d^k\big)$. Let $d = \bar{y} - y^k$, we then have

$$\psi\left(G(y^k) + \nabla G(y^k)^T(\bar{y} - y^k)\right) - \psi\left(G(y^k) + \nabla G(y^k)^T(y^{k+1} - y^k)\right) \geq 0.$$

Together with (4.22), we get

$$\begin{aligned}
\gamma\|y^{k+1} - \bar{y}\| \leq\ & \psi\left(G(\bar{y}) + \nabla G(\bar{y})^T(y^{k+1} - \bar{y})\right) - \psi\big(G(\bar{y})\big) \\
& + \psi\left(G(y^k) + \nabla G(y^k)^T(\bar{y} - y^k)\right) - \psi\left(G(y^k) + \nabla G(y^k)^T(y^{k+1} - y^k)\right).
\end{aligned}$$

Since ψ is convex and has a compact domain, it is (locally) Lipschitz continuous (see e.g.

[63]). Therefore, we have $\psi\left(G(y^k)+\nabla G(y^k)^T(\bar{y}-y^k)\right)-\psi\big(G(\bar{y})\big) \leq L\|G(y^k)-G(\bar{y})+\nabla G(y^k)^T(\bar{y}-y^k)\|$, where L is the Lipschitz constant, and

$$\begin{aligned}
&\psi\left(G(\bar{y})+\nabla G(\bar{y})^T(y^{k+1}-\bar{y})\right)-\psi\left(G(y^k)+\nabla G(y^k)^T(y^{k+1}-y^k)\right)\\
&\quad\leq L\|G(\bar{y})-G(y^k)-\nabla G(y^k)^T(\bar{y}-y^k)+\left[\nabla G(\bar{y})-\nabla G(y^k)\right]^T\left(y^{k+1}-\bar{y}\right)\|\\
&\quad\leq L\|G(\bar{y})-G(y^k)-\nabla G(y^k)^T(\bar{y}-y^k)\|+L\|\left[\nabla G(\bar{y})-\nabla G(y^k)\right]^T\left(y^{k+1}-\bar{y}\right)\|.
\end{aligned}$$

Combining the two parts, we obtain $\|y^{k+1}-\bar{y}\| \leq \frac{2L}{\gamma}\|G(\bar{y})-G(y^k)-\nabla G(y^k)^T(\bar{y}-y^k)\|+\frac{L}{\gamma}\|\left[\nabla G(\bar{y})-\nabla G(y^k)\right]^T\left(y^{k+1}-\bar{y}\right)\|$. Since $G(\cdot)\in C^1$, we have the first term $\|G(\bar{y})-G(y^k)-\nabla G(y^k)^T(\bar{y}-y^k)\|=o(\|y^k-\bar{y}\|)$. Given the fact that as $k\to\infty$, $(\nabla G(\bar{y})-\nabla G(y^k))\to 0$, we also have the second term

$$\|\left[\nabla G(\bar{y})-\nabla G(y^k)\right]^T\left(y^{k+1}-\bar{y}\right)\|=o(\|y^{k+1}-\bar{y}\|).$$

Therefore, we have $\|y^{k+1}-\bar{y}\|=o(\|y^k-\bar{y}\|)$. □

4.5 Numerical Experiments

4.5.1 Results for the Original `SCvx-fast` Algorithm

In this section, we present numerical results that apply the `SCvx-fast` algorithm to an aerospace problem. Consider a multi–rotor vehicle with state at time t_i given by $x_i=\left[p_i^T,v_i^T\right]^T$, where $p_i\in\mathbb{R}^3$ and $v_i\in\mathbb{R}^3$ represent vehicle position and velocity at time t_i respectively. We assume that vehicle motion is adequately modeled by double integrator dynamics with constant time step Δt such that $x_{i+1}=Ax_i+B(u_i+g)$, where $u_i\in\mathbb{R}^3$ is the control at time t_i, $g\in\mathbb{R}^3$ is a constant gravity vector, A is the discrete state transition matrix, and B utilizes zero order hold integration of the control input. Further, we impose a speed upper–bound at each time step, $\|v_i\|_2\leq V_{\max}$, an acceleration upper–bound such that $\|u_i\|_2\leq u_{\max}$ (driven by a thrust upper–bound), and a thrust cone constraint $\hat{n}^Tu_i\geq\|u_i\|_2\cos(\theta_{cone})$ that constrains the thrust vector to a cone pointing towards the unit vector $\hat{n}$

(pointing towards the ceiling) with angle θ_{cone}. Finally, the multi–rotor must avoid a known set of cylindrical obstacles with the j^{th} cylinder having center $p_{c,j} \in \mathbb{R}^2$ and radius $r_j \in \mathbb{R}_+$. Therefore, each state must satisfy the non-convex constraint given by $\|Hx_i - p_{c,j}\|_2 \geq r_j$ where H is a linear mapping that maps p_i to its projection on the ground plane.

Given these constraints, the objective is to find a minimum fuel trajectory from a prescribed $x(t_0)$ to a known $x(t_f)$ with fixed final time t_f and N discrete points along with n_{cyl} cylindrical obstacles to avoid:

Problem 4.5.1. Minimum Fuel 3-DoF Multi–rotor Obstacle Avoidance

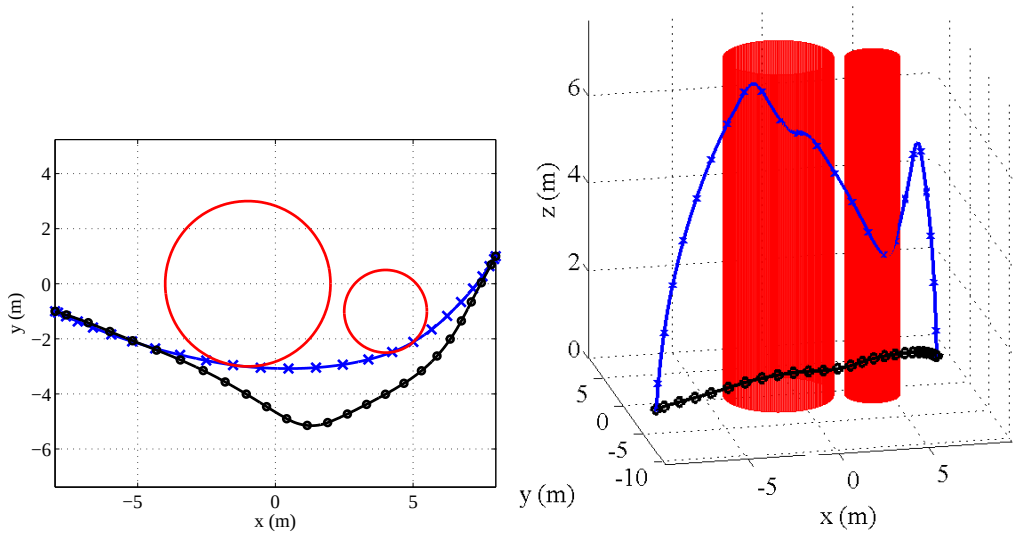
$$
\begin{aligned}
\min \ & \sum_{i=1}^{N} \|u_i\| \\
\text{s.t.} \ & x_{i+1} = Ax_i + B(u_i + g), \ i = 1, \ldots, N-1 \\
& \|u_i\|_2 \leq u_{\max}, \ i = 1, \ldots, N, \\
& \|v_i\|_2 \leq V_{\max}, \ i = 1, \ldots, N, \\
& \hat{n}^T u_i \geq \|u_i\|_2 \cos(\theta_{cone}), \ i = 1, \ldots, N, \\
& \|Hx_i - p_{c,j}\|_2 \geq r_j, \ i = 1, \ldots, N, \ j = 1, \ldots, n_{cyl}, \\
& x_0 = x(t_0), \ x_N = x(t_f).
\end{aligned}
$$

The parameters given in Table 4.1 are used to obtain the numerical results presented herein. An initial feasible trajectory is obtained by using the `SCvx` to solve the feasibility problem associated with this optimization. The feasible initial trajectory, $z^{\{0\}}$ is shown in Figure 4.5 (black circles). Note that $z^{\{0\}}$ is not only feasible for the non-convex state constraints, but also for the convex constraints imposed on the trajectory.

The `SCvx-fast` algorithm is initiated with $z^{\{0\}}$, and is considered to have converged when the improvement in the cost of the linearized problem is less than ϵ. Figure 4.5 illustrates the converged trajectory that avoids the cylindrical obstacles while satisfying its actuator and mission constraints (blue x's). Note that the converged trajectory $z^{\{5\}}$ in Figure 4.5 (blue x's) is different than that of $z^{\{0\}}$ and is characterized by having a smooth curve. For

Table 4.1: Parameter Values of the `SCvx-fast` algorithm test

Par.	Value	Par.	Value
N	25	ϵ	1×10^{-6}
$V_{\max}$	2 m/s	$u_{\max}$	13.33 m/s^2
g	$[0,\ 0,\ -9.81]^T$ m/s^2	θ_{cone}	30 deg
p_0	$[-8,\ -1,\ 0]^T$ m	p_f	$[8,\ 1,\ 0.5]^T$ m
v_0	$[0,\ 0,\ 0]^T$ m/s	v_f	$[0,\ 0,\ 0]^T$ m/s
$p_{c,1}$	$[-1,\ 0]^T$ m	$p_{c,2}$	$[4,\ -1]^T$ m
r_1	3 m	r_2	1.5 m
λ	0	$\hat{n}$	$[0,\ 0,\ 1]^T$
t_f	15 s		



(a) 2-DoF Ground plane projection

(b) 3-DoF plot of trajectories

Figure 4.5: *The `SCvx-fast` algorithm test: Initial trajectory (black circles) and the converged trajectory (blue x's)*

$z^{\{0\}}$, $\sum_{i=1}^{N} \|u_i\| = 269.01$ and at $z^{\{5\}}$, we have that $\sum_{i=1}^{N} \|u_i\| = 245.38$, so the cost of the converged trajectory is lower than that of the initial trajectory. At each iteration, the `SCvx-fast` algorithm solves an SOCP, and therefore 5 SOCPs were solved in order to produce these results (in addition to 2 SOCPs for finding a feasible starting point).

For comparison, we solve the same problem using `SCvx`. The converged trajectories for both algorithms are identical to within ϵ. The number of iterations ($k_{\texttt{SCvx}}$ and $k_{\texttt{SCvx-fast}}$) and time elapsed using each method ($t_{\texttt{SCvx}}$ and $t_{\texttt{SCvx-fast}}$) are reported in Table 4.2. The two iterations necessary to find a feasible starting point for the `SCvx` algorithm are also reported. All times were found using the ECOS solver ([20]) on a standard workstation with an Intel Xeon processor at 3.40 GHz and 16 GB of RAM. We can see that the convergence process of `SCvx-fast` takes far fewer iterations than `SCvx`, and the runtimes presented here show its potential in real–time applications.

Table 4.2: Runtimes and Iterations of the `SCvx-fast` and the `SCvx` algorithms

Method	$k_{\texttt{SCvx}}$	$t_{\texttt{SCvx}}$	$k_{\texttt{SCvx-fast}}$	$t_{\texttt{SCvx-fast}}$	k_{total}	t_{total}
`SCvx`	14	149.9 ms	-	-	14	149.9 ms
`SCvx-fast`	2	39.3 ms	5	26.5 ms	7	65.7 ms

4.5.2 Results for the Enhanced `SCvx-fast` Algorithm

In this section, we apply the enhanced `SCvx-fast` algorithm, Algorithm 9 to an aerospace problem and present the numerical results. Consider a multi–rotor vehicle with state at time t_i given by $x_i = \left[p_i^T, v_i^T\right]^T$, where $p_i \in \mathbb{R}^3$ and $v_i \in \mathbb{R}^3$ represent vehicle position and velocity at time t_i respectively. We assume that vehicle motion is adequately modeled by double integrator dynamics with constant time step Δt such that $x_{i+1} = Ax_i + B(u_i + g)$, where $u_i \in \mathbb{R}^3$ is the control at time t_i, $g \in \mathbb{R}^3$ is a constant gravity vector, A is the discrete state transition matrix, and B utilizes zero order hold integration of the control input. Further, we impose a speed upper–bound at each time step, $\|v_i\|_2 \leq V_{\max}$, an acceleration upper–bound such that $\|u_i\|_2 \leq u_{\max}$ (driven by a thrust upper–bound), and a thrust cone constraint $\hat{n}^T u_i \geq \|u_i\|_2 \cos(\theta_{cone})$ that constrains the thrust vector to a cone pointing towards the unit vector $\hat{n}$ (pointing towards the ceiling) with angle θ_{cone}. Finally, the multi–rotor must avoid a known set of obstacles that can be either ellipsoids or polytopes. The j^{th} ellipsoidal

Table 4.3: Parameter Values of the `SCvx-fast` algorithm test

Par.	Value	Par.	Value
N	20	ϵ	1×10^{-4}
$V_{\max}$	2 m/s	$u_{\max}$	13.33 m/s^2
g	$[0,\ 0,\ -9.81]^T$ m/s^2	θ_{cone}	30 deg
p_0	$[-2,\ 6,\ 0]^T$ m	p_f	$[6,\ 2,\ 0.5]^T$ m
v_0	$[0,\ 0,\ 0]^T$ m/s	v_f	$[0,\ 0,\ 0]^T$ m/s
λ	0	$\hat{n}$	$[0,\ 0,\ 1]^T$
t_f	15 s		

is expressed as $x'A_jx + 2b_j'x + c_j \leq 0$, while The k^{th} polytopical is defined by its faces $A_kx + b_k \leq 0$.

Given these constraints, the objective is to find a minimum fuel trajectory from a prescribed $x(t_0)$ to a known $x(t_f)$ with fixed final time t_f and N discrete points along with $n_{ellip} + n_{polyt}$ obstacles to avoid:

Problem 4.5.2. Minimum Fuel 3-DoF Multi–rotor Obstacle Avoidance

$$
\begin{aligned}
\min\ & \sum_{i=1}^{N} \|u_i\| \\
\text{s.t.}\ \ & x_{i+1} = Ax_i + B(u_i + g),\ i = 1, \ldots, N-1 \\
& \|u_i\|_2 \leq u_{\max},\ i = 1, \ldots, N, \\
& \|v_i\|_2 \leq V_{\max},\ i = 1, \ldots, N, \\
& \hat{n}^T u_i \geq \|u_i\|_2 \cos(\theta_{cone}),\ i = 1, \ldots, N, \\
& x_i' A_j x_i + 2b_j' x_i + c_j \geq 0,\ i = 1, \ldots, N,\ j = 1, \ldots, n_{ellip}, \\
& A_k x_i + b_k \geq 0,\ i = 1, \ldots, N,\ k = 1, \ldots, n_{polyt}, \\
& x_0 = x(t_0),\ x_N = x(t_f).
\end{aligned}
$$

The parameters given in Table 4.3 are used to obtain the numerical results presented herein. An initial feasible trajectory is obtained by using Algorithm 8. The infeasible

initial trajectory $z^{\{0\}}$ is shown in Figure 4.6 (black), while the feasible and (locally) optimal converged trajectory $\bar{z}$ is shown in Figure 4.7 (green)

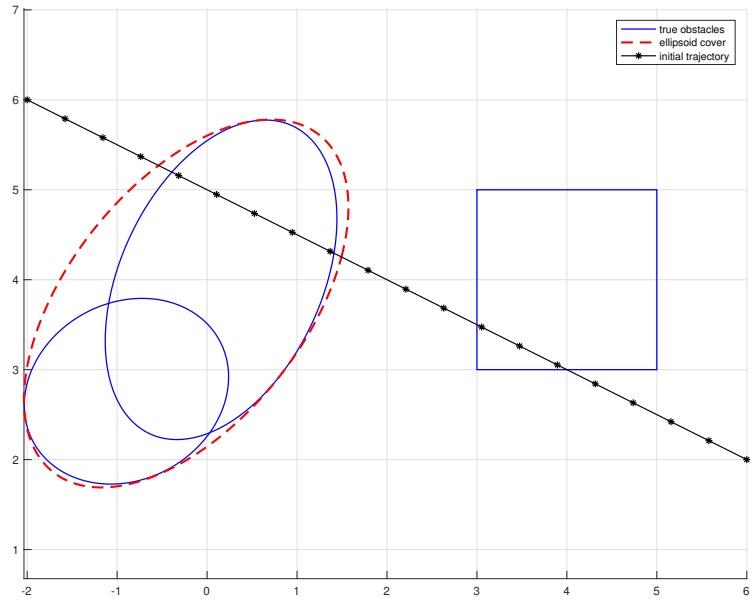


Figure 4.6: *The* ***SCvx-fast*** *algorithm test: Initial trajectory (black) and the minimum volume ellipsoidal cover (red) for intersecting obstacles (blue).*

The `SCvx-fast` algorithm is initiated with $z^{\{0\}}$, and is considered to have converged when the improvement in the cost of the convexified problem is less than ϵ. Figure 4.7 illustrates the converged trajectory that avoids the obstacles while satisfying its actuator and mission constraints (green). Note that the converged trajectory in Figure 4.7 (blue x's) is different than that of $z^{\{0\}}$ and is characterized by having a smooth curve. For $z^{\{0\}}$, $\sum_{i=1}^{N} \|u_i\| = 282.41$ and at the converged solution $z^{\{11\}}$, we have that $\sum_{i=1}^{N} \|u_i\| = 221.76$, so the cost of the converged trajectory is lower than that of the initial trajectory. At each iteration, the `SCvx-fast` algorithm solves an SOCP, and therefore 11 SOCPs were solved in order to produce these results. The full convergence history is shown in Figure 4.8, which clearly demonstrates the pattern of a superlinearly convergent algorithm: It improves

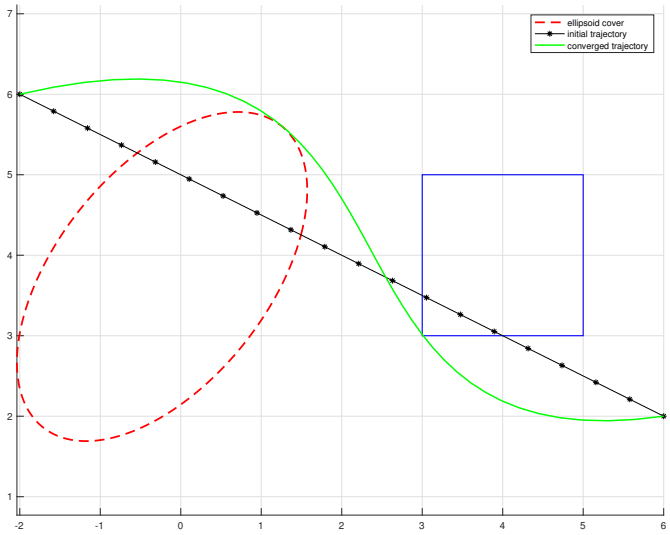


Figure 4.7: *The* ***SCvx-fast*** *algorithm test: converged trajectory (green) and the minimum volume ellipsoidal cover (red) for intersecting obstacles (blue).*

relatively slowly at first, but significantly speeds up later on, especially when approaching the converged solution.

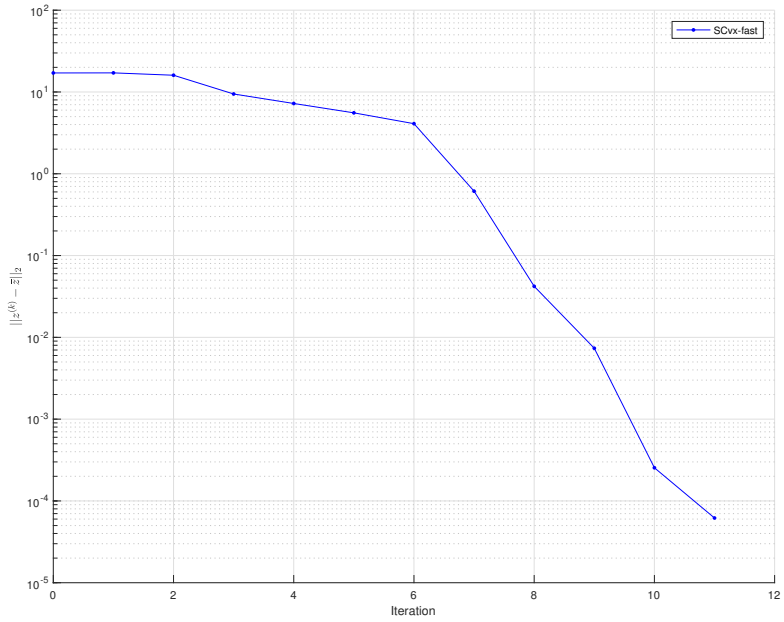


Figure 4.8: *The* ***SCvx-fast*** *algorithm test: Full convergence history of the solution variable z. It affirms a typical pattern for superlinear convergence rate, namely, speeding up when near the converged solution.*

Chapter 5

TOWARD FULL AUTONOMY: APPLICATIONS OF SUCCESSIVE CONVEXIFICATION AS TRAJECTORY PLANNER

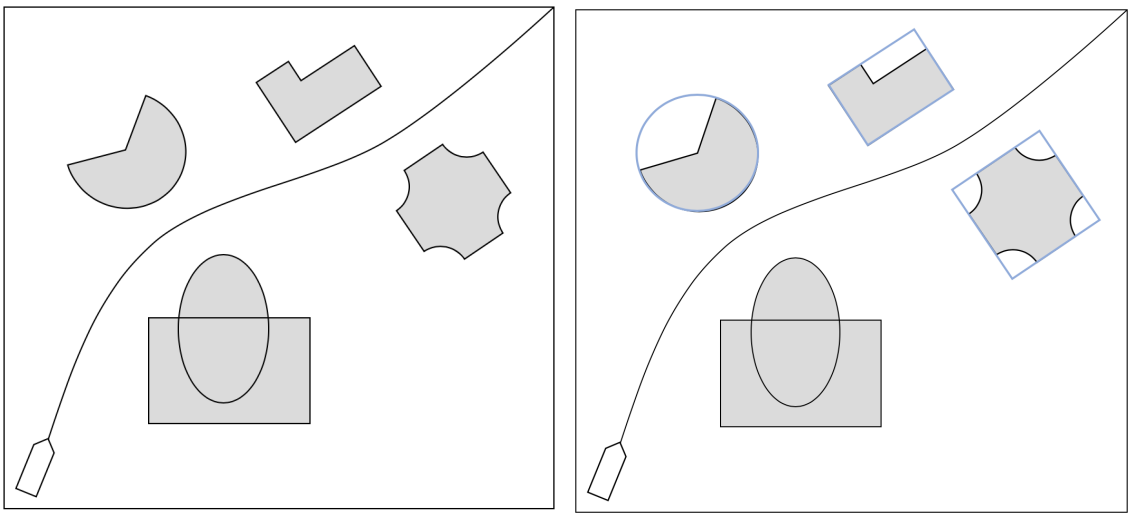
So far in this document we have established the algorithmic framework of *Successive Convexification*, and provided mathematical proofs for convergence properties. The author is also looking outward application-wise for other pieces of the puzzle toward building a fully autonomy system. This chapter represents the application-oriented work that uses *Suceessive Convexification* as a building block to achieve this goal.

5.1 Learning-based Perception

For a typical optimal control problem (e.g. Problem 3.1.1), the parameters $(N, \phi, f, s, X_i, U_i)$ are given to us, and one may proceed to solve it using `SCvx` or some other algorithms. However, in real-world autonomous operations, these parameters may not always be readily usable. For one, we may not know how far into the future we want to plan due to environmental uncertainty. That is, the best final time, N, has to be estimated. Another example would be the obstacle information. We know from [52] that the `SCvx-fast` algorithm performs better in real-time thanks to its simplicity, thus would be the preferred planner in practice. However, one notable requirement of `SCvx-fast` is that the obstacles have to be decomposable into convex shapes, i.e., h in Problem 4.1.1 has to be a vector of convex functions. Meanwhile, the perception data about the environment is typically a stream of occupancy grids, where the obstacles are hardly in convex shapes. Besides, given limited information, the vehicle may also do not know where it should go next that can minimize the overall cost, i.e., $\phi(x_N, u_N)$ in Problem 3.1.1 is unclear and has to be inferred. These real-world scenarios

all bear the question: how can we set up a real-time solvable trajectory optimization problem when some of the parameters are either unknown or not readily usable?

In this section, we aim to provide a preliminary learning-based answer to this question. This research in conducted in two phases. In phase one, the author focuses on using Convolutional Neural Network (CNN) to learn the convex shapes for obstacles from occupancy maps. Figure 5.1 shows how one can generate convex shaped obstacles in an occupancy map.



(a) Original non-convex shaped obstacles (b) Generated convex shaped obstacles

Figure 5.1: *Generating convex shaped obstacles from non-convex ones*

Phase two has not started yet, thus will be an area for future work. Nevertheless, the author plans to look at other potential parameters that learning can help find the best choices of. For instance, we can use learning to identify the optimal planning horizon. As well, we may learn from dynamical simulations the optimal goal placement for certain environment. More specifically, we can train the learning model by running mission simulations with different intermediate goal states and an overall performance metric.

5.1.1 Methodology

In this section, we will discuss the methodology we use for phase one, i.e., the learning of convex shapes. First we need the training and testing data. In this case, that would be artificially generated occupancy maps. The maps are generated in a way that is outlined in [45]. Roughly speaking, First some binary clusters of obstacles are randomly generated on an empty map. Secondly, the obstacle clusters are blurred by a Gaussian filter, which converts the binary map to a probabilistic occupancy grid.

Next step, we manually annotate the obstacles using convex shapes like circles and polygons. Currently the annotation is done by using a web based API provided by `dataturks.com`, but we are also exploring more efficient alternatives.

Once we have enough annotated maps in our dataset, we can start training on it. We use the `Faster R-CNN` model on the `TensorFlow` platform. Finally, we use a separate dataset to test the trained model.

5.1.2 Preliminary Results

In the first batch of tests, we have manually annotated 120 occupancy maps (training/testing: 90/30) and there are 631 obstacles in total. We test both Hausdorff Distance and Intersection-over-Union (IoU) as the accuracy metric. The IoU is a measure of how close two regions are on a scale between 0 and 1. A value of 0 means the regions do not overlap and a scale of 1 means that the regions are exactly the same. Explicitly,

$$\mathrm{IoU}(A, B) = \mathrm{area}(A \cap B) \,/\, \mathrm{area}(A \cup B).$$

Table 5.1 summarizes the preliminary results we get from this limited dataset.

In addition, we find that computing the Hausdorff Distance takes quite a while so in the second batch, we did tests with 80 samples using only the IoU metric. Results is shown in Table 5.2. Note that for the IoU mean, the closer to 1 the better, so we did see a slight increase in accuracy with more samples.

Table 5.1: Preliminary training and testing results

Hausdorff Distance (m)	Training	Testing	IoU	Training	Testing
mean	0.166	0.201	mean	0.924	0.855
std	0.052	0.055	std	0.047	0.053

Table 5.2: Additional training and testing results

IoU	Training	Testing
mean	0.936	0.883
std	0.042	0.057

5.2 `SCvx` *with STL Specifications*

In the envisioned near future of cyber-physical systems, unmanned systems such as Unmanned Aerial Vehicles (UAV) and intelligent ground robots are playing an increasingly important role, as they are assigned newer and more challenging tasks. From package delivery to traffic monitoring, from aerial photograph to fieldwork on modern farms, careful and autonomous trajectory planning is of utmost importance to ensure safety and mission requirements satisfaction. An onboard optimal-control-based guidance system is proved to be highly effective in this regard [50]. A notable line of work is the Successive Convexification(`SCvx`) algorithmic framework [54], which provides both theoretical convergence guarantee and practical real-time solving capability. However, traditional optimal control solvers, including `SCvx`, are designed to handle spatial (state) and physical (control) constraints, not temporal ones. Meanwhile, we have seen a growing trend of stating complex mission requirements as temporal specifications [1], thanks to the expressiveness of temporal logic encoding. Examples of these requirements include deadlines, periodic occurrences, and sequentially triggered events. The downside is that its semantics are combinatorial in nature and thus difficult to incorporate in a smooth optimization solver, which is usually required for onboard computation. Therefore, one may ask: "Is it possible to have the best of both

worlds?" That is, is it possible to have a smooth optimal control solver that also incorporates the temporal logic specifications as constraints?

In this section, we aim to provide a positive answer to this question. For the smooth optimization solver part, we are going to adopt the `SCvx` framework. Hence, the problem becomes how to express the temporal logic encodings as constraints recognizable by `SCvx`. Since `SCvx` deals with continuous-time systems with continuous-time state variable x, we are going to use Signal Temporal Logic (STL) as the primary form of the temporal encoding, which also operates on continuous-time trajectories ("signals"). A more detailed survey on the state of the art of both `SCvx` and STL is given in section 5.2.1. In this section, we propose a four-steps approach, called `SCvx-STL`, to tackle the problem. First, we use robust STL semantics [21] to express these temporal logic specifications as real-valued state constraints. Second, we introduce auxiliary state variables to transform these state constraints into system dynamics (thus creating "STL subdynamics"), by exploiting the recursively structure of certain robust STL semantics. Third, we smooth the auxiliary system dynamics with polynomial smooth min- and max- functions. This also takes advantage of the aforementioned recursive structures. Last, we convexify the resulting smooth optimal control problem and solve with the `SCvx` algorithm.

5.2.1 State of the art

Optimal control (trajectory optimization) of unmanned vehicles has been extensively studied. The reader is referred to [50] for an overview of the field. In particular, convex optimization and convexification based technologies have been developed for more than a decade now [5, 4], while newer 6-DoF (Degrees of Freedom) formulations with attitude control have been studied in recent years [41, 74, 73]. Lately we have also seen commercial usages of these technologies on, for instance, SpaceX's Falcon series reusable rockets [10]. On a much smaller scale, consider quadrotors, systems that are becoming increasingly ubiquitous in the world of cyber-physical systems. Convexification-based real-time guidance are also used in these agile systems [52, 72], which includes non-convex obstacle avoidance constraints. A

commonly used algorithmic paradigm in this line of work is the Successive Convexification (`SCvx`) framework [51, 54]. It is an iterative procedure that solves optimal control problems with nonlinear dynamics and non-convex state and control constraints, and provides proofs of global convergence and superlinear convergence rate. We will use `SCvx` as our smooth optimization solver thanks to its good synergy with STL specifications. More on this later.

The talk of Signal Temporal Logic (STL) is often connected with controlling a cyber-physical system and formal verification methods. STL was originally developed in order to specify and monitor the expected behavior of physical systems [49], including temporal constraints between events. STL allows the specification of properties of dense-time, real-valued signals, and the automatic generation of monitors for testing these properties on individual simulation traces. It has since been applied to the analysis of several types of continuous and hybrid systems, including dynamical systems where the continuous variables represent quantities like position and velocity of a vehicle [61]. STL has the advantage of naturally admitting quantitative semantics which, in addition to the yes/no answer to the satisfaction question, provide a real number that grades the quality of the satisfaction or violation. Such semantics have been defined to assess the robustness of systems to parameter or timing variations [21]. STL as part of the Model Predictive Control (MPC) design has been studied in [61], in which the combinatorial nature of such specifications is preserved, and thus Mixed-Integer Linear Programming (MILP) has to be employed. More recently, for a specific multi-quadrotor setting, [58] relaxes the robustness measures using smooth approximations (log-sum-exponential function for max-) so that the resulting problem can be solved by a smooth optimization solver. It, however, has to employ a two-tier optimization framework because the amount of computation is not real-time implementable.

5.2.2 Signal Temporal Logics

Formal characterization of mission requirements can be expressed as logical statements over states. While first order logic can be typically used to describe state constraints, properties over trajectories can be expressed using temporal logics such as LTL (Linear temporal logics).

These notions have been extended to bounded-horizon continuous time signals in the *Signal Temporal Logics* (STL) and are convenient to express expected behavior of dynamic systems. STL formulas are described using the grammar:

$$\varphi ::= \pi^{\mu} \mid \neg\,\psi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond_{[a,b]}\,\varphi \mid \varphi_1\,\mathcal{U}_{[a,b]}\,\varphi_2,$$

where π^{μ} is an Boolean predicate whose truth value is determined by the sign of a real-valued function μ, and $\varphi, \varphi_1, \varphi_2, \psi$ are STL formulas. As in LTL, $\Diamond_{[a,b]}$ and $\mathcal{U}_{[a,b]}$ denote the eventually and until modalities, respectively, with a restriction to a given time frame $[a, b]$ with $a < b$. A simulation run $\xi(x_0, u)$ satisfies an STL formula φ is denoted by $\xi \models \varphi$.

Informally, we have *Eventually* denoted as $\xi \models \Diamond_{[a,b]}\,\varphi$, meaning φ holds at some time between a and b. Also, we have *Until* denoted as $\xi \models \varphi\,\mathcal{U}_{[a,b]}\,\psi$, meaning φ holds at every time before ψ holds, and ψ holds at some time between a and b. Additionally, we define *Always* as $\Box_{[a,b]}\,\varphi = \neg\,\Diamond_{[a,b]}\,(\neg\,\varphi)$, so that $\xi \models \Box_{[a,b]}\,\varphi$ means φ holds at all times between a and b.

The semantics of a formula φ with respect to a run ξ is defined inductively as in Definition 5.2.1.

Definition 5.2.1 (STL semantics).

$$\begin{aligned}
\xi \models \varphi \quad &\Leftrightarrow \quad (\xi, t_0) \models \varphi \\
(\xi, t_k) \models \pi^{\mu} \quad &\Leftrightarrow \quad \mu(x_k, u_k) > 0 \\
(\xi, t_k) \models \neg\psi \quad &\Leftrightarrow \quad \neg(\xi, t_k) \models \psi \\
(\xi, t_k) \models \varphi \wedge \psi \quad &\Leftrightarrow \quad (\xi, t_k) \models \varphi \ \wedge\ (\xi, t_k) \models \psi \\
(\xi, t_k) \models \Diamond_{[a,b]}\,\varphi \quad &\Leftrightarrow \quad \exists\, t_{k'} \in [t_k + a, t_k + b], (\xi, t_{k'}) \models \varphi \\
(\xi, t_k) \models \varphi\,\mathcal{U}_{[a,b]}\,\psi \quad &\Leftrightarrow \quad \exists\, t_{k'} \in [t_k + a, t_k + b], (\xi, t_{k'}) \models \psi \\
&\qquad \wedge\ \forall\, t_{k''} \in [t_k, t_{k'}], (\xi, t_{k''}) \models \varphi.
\end{aligned}$$

A key challenge is to efficiently relate specifications with optimal control. Robust seman-

tics of STL [21] amounts to characterize a real-valued function ρ^{φ} of signal ξ and time t such that $\rho^{\varphi}(\xi, t) > 0 \Rightarrow (\xi, t) \models \varphi$. That is, ρ^{φ} serves as a robustness measure of how much ξ satisfies φ. Its absolute value can be viewed as the signed distance of ξ from the set of trajectories satisfying or violating φ, in the space of projections with respect to the function μ that define the predicates of φ. The robustness function can also be defined recursively as in Definition 5.2.2.

Definition 5.2.2 (**Robust STL semantics**)**.**

$$
\begin{aligned}
\rho^{\pi^{\mu}}(\xi, t_k) &= \mu(x_k, u_k) \\
\rho^{\neg\psi}(\xi, t_k) &= -\rho^{\psi}(\xi, t_k) \\
\rho^{\varphi_1 \wedge \varphi_2}(\xi, t_k) &= \min\left(\rho^{\varphi_1}(\xi, t_k), \rho^{\varphi_2}(\xi, t_k)\right) \\
\rho^{\varphi_1 \vee \varphi_2}(\xi, t_k) &= \max\left(\rho^{\varphi_1}(\xi, t_k), \rho^{\varphi_2}(\xi, t_k)\right) \\
\rho^{\Diamond_{[a,b]}\,\psi}(\xi, t_k) &= \max_{t_{k'} \in [t_k + a, t_k + b]} \rho^{\psi}(\xi, t_{k'}) \\
\rho^{\varphi_1\, \mathcal{U}_{[a,b]}\, \varphi_2}(\xi, t_k) &= \max_{t_{k'} \in [t_k + a, t_k + b]} \min\big(\rho^{\varphi_2}(\xi, t_{k'}), \\
&\qquad \min_{t_{k''} \in [t_k, t_{k'}]} \rho^{\varphi_1}(\xi, t_{k''})\big).
\end{aligned}
$$

Temporal operators (e.g. $\max_{t_{k'} \in [t_k + a, t_k + b]}$) are treated as conjunctions or disjunctions along the time axis once discretized for computation on digital computers.

An example optimal control problem with robust STL specifications can be written as follows, in Problem 5.2.1.

Problem 5.2.1. Optimal Control with Robust STL Specs

$$
\begin{aligned}
\max_{u} \quad & \rho^{\varphi}(x) - \omega\, C(x, u) \\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k),\ k = 1, 2, \ldots, N-1, \\
& x_k \in X_k, u_k \in U_k,\ k = 1, 2, \ldots, N, \\
& \rho^{\varphi}(x) \geq 0,
\end{aligned}
\tag{5.1}
$$

where $\omega > 0$ is a weighting parameter between the two objectives. In this example, we are trying to maximize the STL robustness measure as well as ensure the satisfaction of STL specifications in the first place. One could just do either of the two depending on the specific needs.

Unfortunately as shown in Definition 5.2.2, ρ^{φ} is mainly built using min- and max-operators leading to difficult combinatorial problems, which necessitates the use of Mixed-Integer Programming solvers [61]. A smooth (infinitely differentiable) approximation of these function has been proposed [58] and the solution is to straightforwardly substitute min- and max- functions in ρ^{φ} with log-sum-exponential functions:

$$\begin{aligned} \tilde{\max}_k(a_1, \ldots, a_n) &:= \frac{1}{k} \ln(e^{ka_1} + \ldots + e^{ka_m}), \\ \tilde{\min}_k(a_1, \ldots, a_n) &:= -\tilde{\max}_k(-a_1, \ldots, -a_n). \end{aligned} \tag{5.2}$$

Despite the improved run-time from the smooth operators, experiments reported [58] were not successful to solve (5.1) with this smooth robust STL semantics, efficiently, in real-time, for a multi-drone fleet trajectory planning problem using the full quadrotor dynamics. It uses a two-tier sampling method instead.

One obvious drawback of this straightforward smoothing method is that the gradient calculation will quickly getting out of hand when the elemental STL specification, such as $\rho^{\varphi_1}(\xi, t_{k''})$ in the formula for the *Until* operator $\rho^{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}(\xi, t_k)$, is moderately complex. Therefore, it is of great interest to explore other ways to systematically encode and smooth the robust STL semantics.

5.2.3 STL subdynamics

In this and subsequent sections, we will present the `SCvx-STL` method that systematically encodes and smooths the robust STL specifications $\rho^{\varphi}(x)$ in Problem 5.2.1. Once we have established Problem 5.2.1 as our main problem to solve, the next step is to take a closer look at the temporal specifications we are trying to encode, namely the *Eventually* $\rho^{\Diamond_{[a,b]} \psi}(\xi, t_k)$

and the *Until* $\rho^{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}(\xi, t_k)$.

The Eventually *operator* $\Diamond_{[a,b]}$

Notice that with the discretized state and control variables, we can denote the first and last time instance for $t_{k'} \in [t_k + a, t_k + b]$ as $k^{(a)}$ and $k^{(b)}$ respectively. Additionally, we define the set of instances in between $k^{(a)}$ and $k^{(b)}$ as $\mathcal{K} \coloneqq \{k^{(a)}, k^{(a)} + 1, \ldots, k^{(b)}\}$. Then, we have

$$\rho^{\Diamond_{[a,b]} \psi}(x, k) \;=\; \max_{k' \in \mathcal{K}} \rho^{\psi}(x, k')$$

We can then introduce an auxiliary variable $y = [y_1, y_2, \ldots, y_k, \ldots, y_N]$ as follows.

$$\begin{aligned} y_k &= \rho^{\psi}(x, k^{(a)}), && \forall\, 1 \leq k \leq k^{(a)}, \\ y_{k+1} &= \max\big(y_k,\, \rho^{\psi}(x, k+1)\big), && \forall\, k^{(a)} < k < k^{(b)}, \\ y_k &= y_{k^{(b)}}, && \forall\, k^{(b)} < k \leq N. \end{aligned}$$

Given $x_{k+1} = f(x_k, u_k)$ from the system dynamics, we have

$$\rho^{\psi}(x, k+1) = \rho^{\psi}(f(x_k, u_k)) \coloneqq \tilde{\rho}^{\psi}(x_k, u_k).$$

Hence, for $k^{(a)} < k < k^{(b)}$, we have

$$y_{k+1} = \max\big(y_k,\, \tilde{\rho}^{\psi}(x_k, u_k)\big) \coloneqq f_y(y_k, x_k, u_k), \tag{5.3}$$

which is recursively defined, and thus y can be treated as a state variable, whose corresponding dynamical equation is exactly (5.3) for $k^{(a)} < k < k^{(b)}$.

The Until *operator* $\mathcal{U}_{[a,b]}$

Similar considerations can be taken for the *Until* operator, albeit more complex. In addition to the same notation for $k^{(a)}, k^{(b)}$ and $\mathcal{K}$, we also define of all time instances in between t_k

and $t_{k'}$ as $\mathcal{K}'$, and all time instances in between t_k and $t_{k^{(a)}}$ as $\mathcal{K}^{(a)}$. Then we can rewrite the formula for the *Until* operator as

$$\rho^{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}(x,k) \;=\; \max_{k' \in \mathcal{K}} \min\big(\rho^{\varphi_2}(x,k'),\, \min_{k'' \in \mathcal{K}'} \rho^{\varphi_1}(x,k'')\big).$$

First, we introduce an auxiliary variable $\zeta = [\zeta_1, \zeta_2, \ldots, \zeta_k, \ldots, \zeta_N]$ as follows.

$$\begin{aligned}
\zeta_k &= \min_{k'' \in \mathcal{K}^{(a)}} \rho^{\varphi_1}(x,k''), && \forall\, 1 \le k \le k^{(a)},\\
\zeta_{k+1} &= \min\big(\zeta_k,\, \rho^{\varphi_1}(x,k+1)\big), && \forall\, k^{(a)} < k < k^{(b)},\\
\zeta_k &= \zeta_{k^{(b)}}, && \forall\, k^{(b)} < k \le N.
\end{aligned}$$

Similarly, given $x_{k+1} = f(x_k, u_k)$ from the system dynamics, we have

$$\rho^{\varphi_1}(x, k+1) = \rho^{\varphi_1}(f(x_k,u_k)) \coloneqq \tilde{\rho}^{\varphi_1}(x_k,u_k).$$

Hence, for $k^{(a)} < k < k^{(b)}$, we have

$$\zeta_{k+1} = \min\big(\zeta_k,\, \tilde{\rho}^{\varphi_1}(x_k,u_k)\big) \coloneqq f_\zeta(\zeta_k, x_k, u_k), \tag{5.4}$$

which is recursively defined, and thus ζ can be treated as a state variable, whose corresponding dynamical equation is exactly (5.4) for $k^{(a)} < k < k^{(b)}$.

Next, for the outer max function, we introduce another auxiliary variable $\eta = [\eta_1, \eta_2, \ldots, \eta_k, \ldots, \eta_N]$ as follows.

$$\begin{aligned}
\eta_k &= \min\big(\rho^{\varphi_2}(x,k^{(a)}),\, \zeta_k\big), && \forall\, 1 \le k \le k^{(a)},\\
\eta_{k+1} &= \max\Big(\min\big(\rho^{\varphi_2}(x,k+1),\, \zeta_{k+1}\big),\, \eta_k\Big), && \forall\, k^{(a)} < k < k^{(b)},\\
\eta_k &= \eta_{k^{(b)}}, && \forall\, k^{(b)} < k \le N.
\end{aligned}$$

Given $x_{k+1} = f(x_k, u_k)$ from the system dynamics, we have

$$\rho^{\varphi_2}(x, k+1) = \rho^{\varphi_2}(f(x_k, u_k)) \coloneqq \tilde{\rho}^{\varphi_2}(x_k, u_k).$$

Combined with the dynamics for ζ (5.4), we have for $k^{(a)} < k < k^{(b)}$,

$$\begin{aligned} \eta_{k+1} &= \max\Big(\min\big(\tilde{\rho}^{\varphi_2}(x_k, u_k),\ f_\zeta(\zeta_k, x_k, u_k)\big),\ \eta_k\Big) \\ &\coloneqq f_\eta(\eta_k, \zeta_k, x_k, u_k), \end{aligned} \tag{5.5}$$

Note that the dynamical time window from $k^{(a)}$ to $k^{(b)}$ for each STL specification can be different, and they are all relative to their respective starting time t_k. Therefore, the above formulation also applies to nested STL specifications. For example, $\rho^{\psi}(\xi, t_{k'})$ in the *Eventually* formula can be an *Until* specification $\rho^{\varphi_1 \mathcal{U}_{[a',b']} \varphi_2}(\xi, t_{k'})$ starting from $t_{k'}$. One just need to adjust the dynamical time window for the associated auxiliary state variable(s) accordingly.

Assuming both the *Eventually* and the *Until* operators are present, we may append the auxiliary states y, ζ and η to the original state variable x and obtain a new state variable

$$z = \begin{bmatrix} x \\ y \\ \zeta \\ \eta \end{bmatrix} \text{ and } z_{k+1} = f_z(z_k, u_k) = \begin{bmatrix} f(x_k, u_k) \\ f_y(y_k, x_k, u_k) \\ f_\zeta(\zeta_k, x_k, u_k) \\ f_\eta(\eta_k, \zeta_k, x_k, u_k) \end{bmatrix}. \tag{5.6}$$

We call the dynamics for the auxiliary variables y, ζ and η (5.3), (5.4), and (5.5) "STL subdynamics", since they correspond to the subvector of the last three lines in (5.6).

To replace the STL specified state constraints with these newly defined dynamical equations, one just need to add a simple convex state constraint with respect to the associated auxiliary variable, to ensure the robust satisfaction of the original constraints. Use the *Until* operator as an example, one can easily verify that the following two problems are equivalent:

Problem 5.2.2. Optimal Control with the *Until* Specifications

$$\begin{aligned} \min_{u} \quad & C(x,u) \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k), \\ & \rho^{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}(x,k) \geq 0. \end{aligned} \tag{5.7}$$

Problem 5.2.3. Optimal Control with the *Until* Subdynamics

$$\begin{aligned} \min_{u} \quad & C(x,u) \\ \text{s.t.} \quad & z_{k+1} = f_z(z_k, u_k), \\ & \eta_{k^{(b)}} \geq 0. \end{aligned} \tag{5.8}$$

Here z and η are defined in the same way as before.

One clear advantage of using this transformation is the dramatic simplification of gradient calculation in (3.2a) and (3.2b). Previously, the temporal specification is over a time window, which could include a large number of sampling instances. Since one needs to apply the chain rule to calculate the gradient, too many terms will simply make the gradient formula nearly impossible to obtain. In contrast, the "STL subdynamics" approach exploits the recursive structure of the STL semantics, and temporally group all the terms together in the same form (the auxiliary subdynamics). Additionally, by appending the original state vector, we are essentially solving the same problem as vanilla `SCvx` does, just with expanded states and the associated dynamical equations, which also makes implementation much easier.

5.2.4 Polynomial smooth min `smin`()

Now that we have transformed the STL specified constraints into subdynamics, the next step is to make the corresponding dynamical equations smooth so that we can apply `SCvx` to solve. Note that in (5.3), (5.4), and (5.5), we still have non-differentiable min- or max-functions. One can certainly use the log-sum-exponential function to approximate these

min- and max- functions (5.2), but this approximation is usually not accurate around non-differentiable points, especially with only two variables, which is exactly the case we have on hand. To solve this problem, we use the polynomial smooth min function $\texttt{smin}(a, b, k)$ defined as

$$\texttt{smin}(a, b, k) = \begin{cases} a, & \text{if } a - b \geq k, \\ b, & \text{if } a - b \leq -k, \\ g(a, b, k), & \text{if } a - b \in (-k, k), \end{cases} \tag{5.9}$$

where $g(a, b, k)$ is a smooth interpolator, and $k > 0$ controls the interpolation range. The idea of this `smin`() function is simple: we want to smoothly interpolate the values a and b if they are close to each other (i.e. $a - b \in (-k, k)$), otherwise we return the true minimum. A polynomial smooth interpolator function is given in [66] with a nice derivation:

$$g(a, b, k) = a(1 - h) + hb - kh(1 - h), \tag{5.10}$$

where $h = \frac{1}{2} + \frac{a-b}{2k}$. One can easily verify that with this $g(a, b, k)$ in (5.10), the `smin`() function in (5.9) is continuously differentiable.

This concept of polynomial smooth min function goes back to a blog post decade ago [60], though both [60] and [66] use it on smoothing the edges in computer graphics applications. The authors have not seen this technique been used in optimization settings, probably due to the limitation of only accepting two variables. However, it lends itself perfectly for smoothing the STL subdynamics, because we only need to calculate the min or max of two values at a time thanks to the recursive nature of dynamical equations.

5.2.5 Solving with `SCvx`

In (5.3), (5.4), and (5.5), replace the min operator with `smin`() and the max operator with $-$`smin`(). Denoting the resulting smooth dynamics as $\tilde{f}_y(), \tilde{f}_\zeta(), \tilde{f}_\eta()$ and $\tilde{f}_z()$, we arrive at the optimal control problem with smooth STL subdynamics, Problem 5.2.4 (Using *Until* as

an example again).

Problem 5.2.4. Optimal Control with Smooth Subdynamics

$$\begin{aligned} \min_{u} \quad & C(x,u) \\ \text{s.t. } & z_{k+1} = \tilde{f}_z(z_k, u_k), \\ & \eta_{k^{(b)}} \geq 0. \end{aligned} \tag{5.11}$$

This problem is now finally ready to be solved by the `SCvx` algorithm. Based on the same example of the *Until* specifications, we summarize the `SCvx-STL` solution method as a top-level algorithm, Algorithm 10.

Algorithm 10 The `SCvx-STL` Algorithm

1: **procedure** `SCvx-STL`(Problem 3.1.1 with an *Until* specification as in Definiton 5.2.1)
2: **step 1** Use robust *Until* semantics in Definition 5.2.2 to obtain a real-valued problem, Problem 5.2.2.
3: **step 2** Follow the steps in section 5.2.3 to transform STL specified constraints into STL subdynamics (5.6), and thus obtain Problem 5.2.3.
4: **step 3** Smooth the STL subdynamics with polynomial smooth min function `smin`() in (5.9) and obtain the smooth optimal control problem, Problem 5.2.4.
5: **step 4** Solve Problem 5.2.4 with the `SCvx` algorithm, Algorithm 6, and obtain final results.
6: **end procedure**

5.2.6 Numerical Experiments

A Simple Example Problem

We first demonstrate the idea by the following simple example problem with the *Always* specification and using the log-sum-exponential smoothing method. The problem reads as follows.

Problem 5.2.5. 3-DoF minimum fuel problem with double-integrator dynamics that also tries to maximize the robustness measure of the STL specification "*Always* stay below ceil-

Table 5.3: Parameter Values of the preliminary STL specification test

Par.	Value	Par.	Value
N	20	ϵ	1×10^{-4}
g	$[0,\ 0,\ -9.81]^T$ m/s^2	$u_{\max}$	13.33 m/s^2
p_0	$[0.1,\ 0.1,\ 1.5]^T$ m	p_f	$[5,\ 5,\ 5]^T$ m
v_0	$[0,\ 0,\ 0]^T$ m/s	v_f	$[0,\ 0,\ 0]^T$ m/s
c	5.5 m	t_f	2 s

ing":

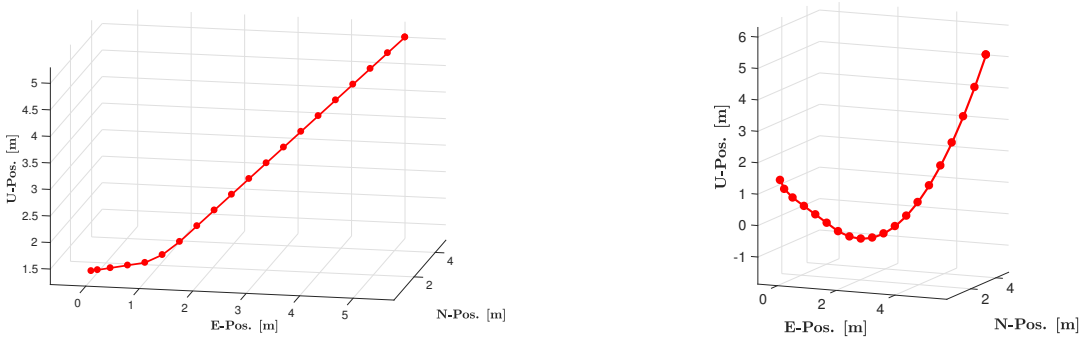
$$\begin{aligned}
\max_{\mathrm{u}} \quad & \sum_{k=1}^{N} \|u_k\| + \lambda\, \rho^{\square_{[t_0,t_f]} x(3)<c}(x) \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k,\ k = 1, \ldots, N-1 \\
& \|u_k\| \le u_{max},\ k = 1, \ldots, N \\
& x_0 = x(t_0),\ x_N = x(t_f),
\end{aligned}$$

where λ is a weighting factor balancing the two objectives. $x(3)$ is the height of the vehicle and c represents the height of the ceiling. $A = \begin{bmatrix} O_3 & I_3 \\ O_3 & O_3 \end{bmatrix}$ and $B = \begin{bmatrix} O_3 \\ I_3 \end{bmatrix}$. u_{max} is the maximum thrust. $x(t_0)$ and $x(t_f)$ are prescribed initial and final state.

The problem was solved in MATLAB using `CVX` [30] and `SDPT3` [77]. The resulting trajectories with two different balancing factor λ's are shown in Figure 5.2. One can see that the more we emphasize the"stay below ceiling" specification, the further away the vehicle tries to keep from the ceiling. This phenomenon illustrates the concept of "robustness" measure, i.e. how safe you want to be with respect to an STL specified constraint.

Spacecraft rendezvous with STL specified mission requirements

In this more elaborated example, we explore the spacecraft rendezvous problem with mission requirements specified in STL semantics. The problem is to autonomously dock the chasing vehicle (CV) onto the target vehicle (TV) with zero relative velocity (See Figure 5.3).



(a) $\lambda = 80$ (b) $\lambda = 1000$

Figure 5.2: *Trajectories of minimum fuel problem with an STL specification. The more we emphasize the "stay below ceiling" specification, the further away the vehicle tries to keep from the ceiling.*

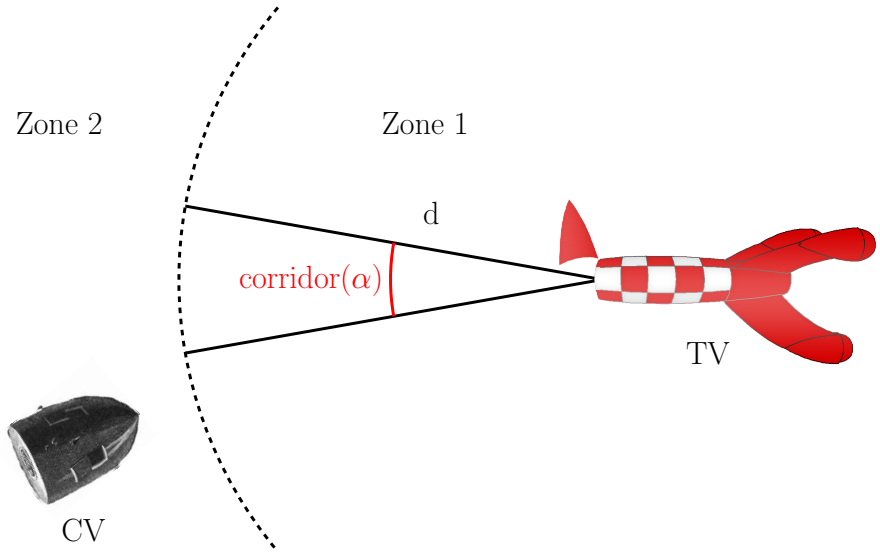


Figure 5.3: *Spacecraft rendezvous with phase transition and safe corridor. The chasing vehicle (CV) enters Zone 1 from Zone 2 due to a range-triggered constraint and needs to stay in the safe corridor cor(α) when in Zone 1 (close proximity to the target vehicle(TV)).*

The system dynamics is given by the Clohessy-Whiltshire equations:

$$\ddot{x} = 3n^2 x + 2n\dot{y}$$

$$\ddot{y} = -2n\dot{x}$$

$$\ddot{z} = -n^2 z,$$

where in lower earth orbit, we typically take $n = 0.00113$. Aside from the docking requirement, we also have multiple STL encoded mission requirements. Let

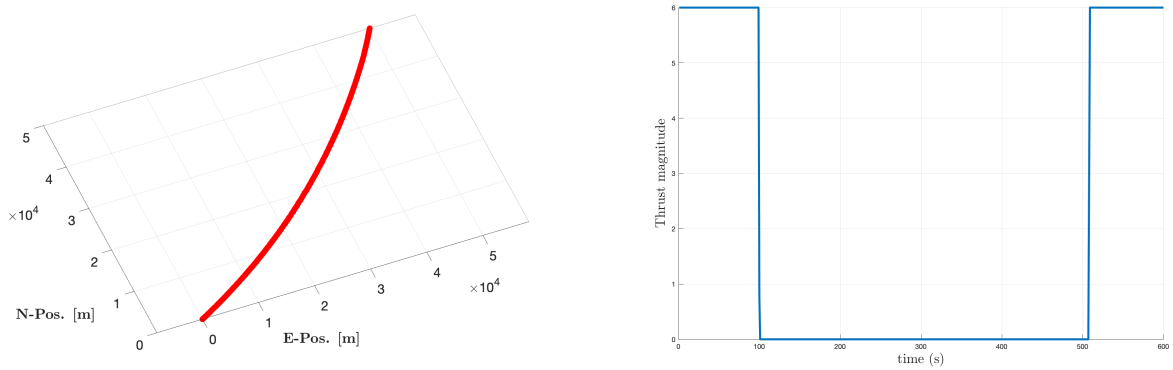
$$
\begin{aligned}
Z_2 &\triangleq \|CV - TV\|_2 - d \geq 0 \\
Z_1 &\triangleq \|CV - TV\|_2 - d \leq 0 \\
D &\triangleq \|CV - TV\|_2 - \epsilon \leq 0 \\
Corr(\alpha) &\triangleq \frac{(CV - TV) \cdot v_{corr}}{\|CV - TV\|_2} \geq cos(\alpha),
\end{aligned}
$$

where D means docking. We then have

- **Req1**: Approach in safe corridor: $Z_1 \implies Corr(\alpha_1)$
- **Req2**: Phase transition: $\square_{[0,T_{\max}]}(Z_2 \implies (Z_2 \, \mathcal{U}_{[0,T_1]} \, Z_1))$
- **Req3**: Dock or in safe zone: $\square_{[0,T_{\max}]}(Z_1 \implies \Diamond_{[0,T_2]}(D \vee Z_2))$
- **Req4**: Dock in safe corridor or abort if outside of safe corridor: $\square_{[0,T_{\max}]}\big(Z_1 \implies Corr(\alpha_1) \, \mathcal{U}_{[0,T_2]}\big(D \vee \big(Corr(\alpha_2) \wedge \Diamond_{[0,T_3]} Z_2\big)\big)\big)$.

For this experiment, we did not incorporate the STL encoded requirements as constraints, rather, we are generating an STL-free trajectory and check the satisfaction of listed STL requirements for that trajectory afterwards. See Figure 5.4 for said trajectory. Note that we have restricted the trajectory to stay in a 2D plane and it comes out no surprise that this setup results in a bang-bang control.

Next, we are going to show the satisfaction or violation of listed STL requirements in Figure 5.5. From the results one can clearly see that although the STL constraints are satisfied most of the time, there are still instances where they are violated. This motivates us to enforce these STL requirements as constraints and solve the problem using `SCvx-STL`, which we will explore in the next example.



(a) computed trajectory of the chaser

(b) thrust profile for this trajectory

Figure 5.4: *Spacecraft rendezvous: computed trajectory (including thrust profile) of the chaser by convex optimization.*

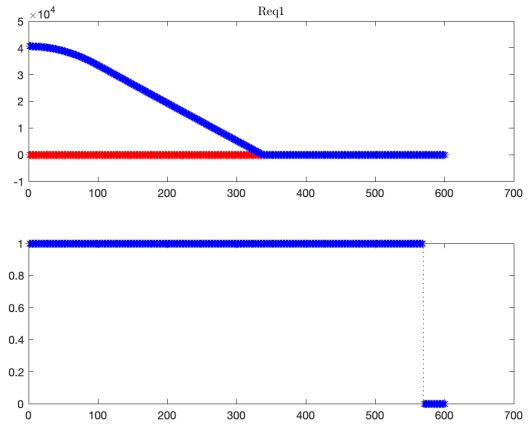
Quadrotor motion planning with STL specifications

In this experiment, we use a non-convex quadrotor obstacle avoidance problem to demonstrate the capability of the proposed `SCvx-STL` algorithm. The non-convexity of this problem stems from cylindrical obstacle keep-out zones (in the same configuration as in Figure 3.2), nonlinear aerodynamic drag, and the additional *Until* specified STL constraints.

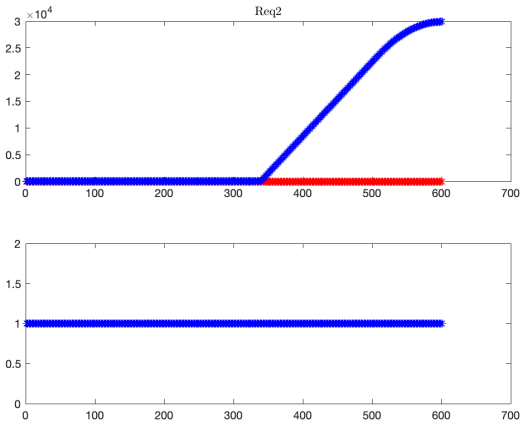
The physical parameters for this numerical experiment closely follows [54], with the exception of STL related parameters and constraints. For a detailed description of the physical model and the problem/parameters setup, the reader is referred to [54].

The STL specified requirement we choose to simulate in this experiment is an *Until* statement: "Do not get in certain range of the destination *Until* a thrust reducing maneuver has been performed within a time window." This is a typical requirement for close proximity operations. This translate to $\rho^{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}(x, t_0) \geq 0$. Here $\rho^{\varphi_1}(x, u) = \|x - x_f\| - R$ and $\rho^{\varphi_2}(x, u) = T_r - \|u\|$, and R is the keep-out range, while T_r is the reduced thrust limit. The value of these parameters selected for this experiment is shown in Table 5.4.

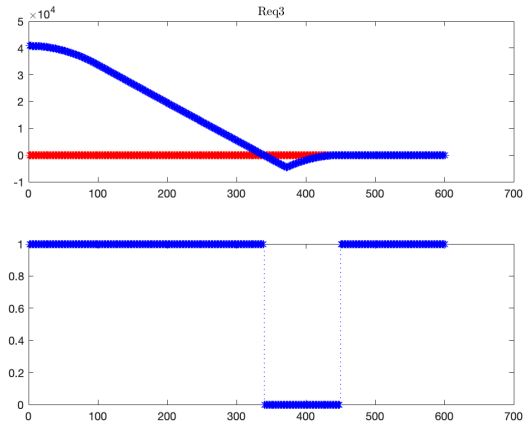
Algorithm 10 was implemented in MATLAB using `CVX` [30] and the `SDPT3` [77] solver.



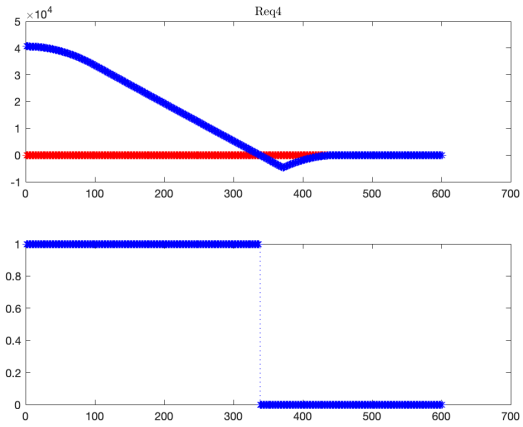
(a) Req1: Approach corridor



(b) Req2: Phase transition



(c) Req3: Dock or safe



(d) Req4: Dock or abort

Figure 5.5: *Spacecraft rendezvous: satisfaction and violation of listed STL requirements. For each subfigure, the top figure shows the value of corresponding robust semantics and the bottom one shows whether the requirement is satisfied (1) or violated (0).*

For onboard real-time computation, one might want to use customized convex optimization solvers that make use of the problem structures. For example, the solver in [23] generates aggressive quadrotor guidance trajectories onboard at rates exceeding 8 Hz.

Figure 5.6 shows the top view of the converged trajectory, as well as thrust profiles (both tilt and magnitude). The converged trajectory is feasible in the physical space as shown in

Table 5.4: STL related parameters

Parameter	Value
a	0 (s)
b	2.4 (s)
R	3 (m)
T_r	$T_{\max}/2$ (N)

the figure. Note that a clear thrust reducing maneuver is performed around time 2.2s, which is within the preset time window [0, 2.4s]. One can clearly observe the differences in the thrust magnitude between here in Figure 5.6 and the one in [54], which is exactly the effect of the additional STL requirement.

Figure 5.7 shows the convergence history of the `SCvx` algorithm for this problem. It takes around 22 successful iterations to converge for this experiment, which just adds a few more iteration to the one in [54]. Not a bad trade-off for including STL requirement.

Figure 5.8 shows the robust measures of ρ^{φ_1} and ρ^{φ_2}. One can easily see that ρ^{φ_1} holds *Until* ρ^{φ_2} holds, which verifies the temporal requirement $\rho^{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}(x, t_0) \geq 0$ for this experiment.

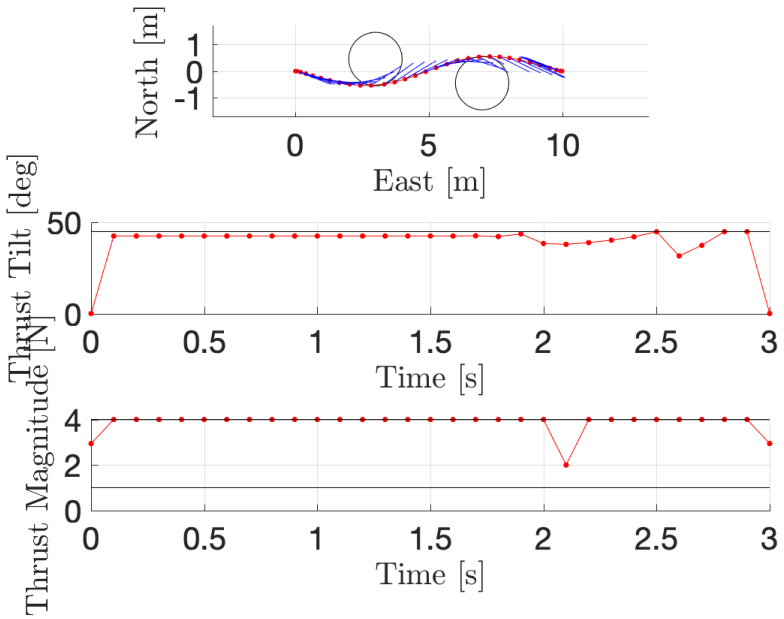


Figure 5.6: *Converged trajectory and thrust profile. Feasibility in physical space and in terms of temporal requirement is validated (based on thrust magnitude).*

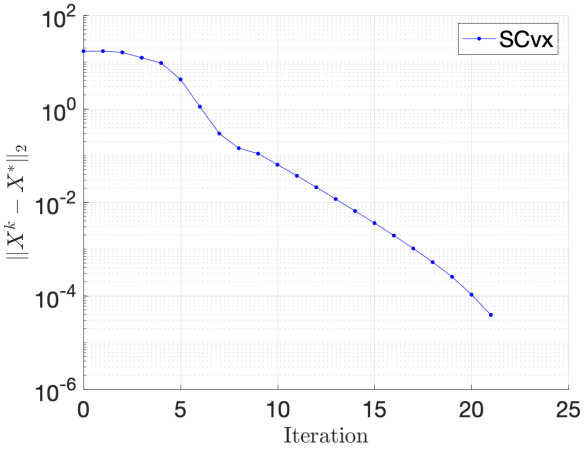


Figure 5.7: *Convergence history of the* `SCvx-STL` *algorithm.*

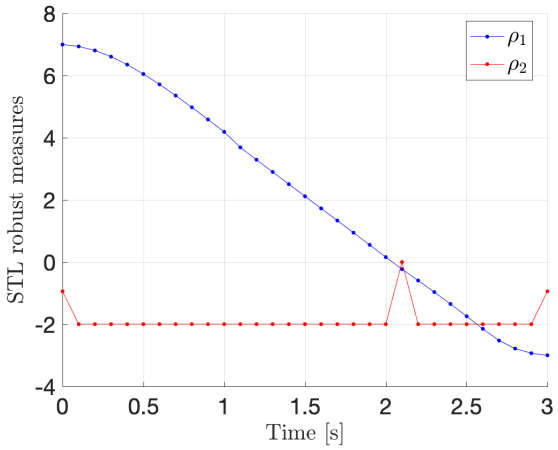


Figure 5.8: *STL robust measures of* ρ^{φ_1} *and* ρ^{φ_2}*. The pattern verifies* $\rho^{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}(x, t_0) \geq 0$.

Chapter 6
CONCLUSIONS

6.1 Summary

In this dissertation, we have presented the author's Ph.D research work, along with some mathematical preliminaries. In Chapter 2, the author gives an overview of the optimal control theory and finite-dimensional optimization theories. Historical and geometrical insight is presented. Connections with the author's work are pointed out. In Chapter 3, the author details every aspect of the `SCvx` algorithm, and gives a thorough treatment of the convergence theories. In short, under some mild conditions, the `SCvx` algorithm is proved to converge globally, strongly and enjoys a superlinear convergence rate. The same applies in Chapter 4 for the `SCvx-fast` algorithm, and the enhanced version of it. Global and superlinear convergence proofs are provided for `SCvx-fast` as well. Numerical examples affirm these convergence properties in both chapters. In Chapter 5, the author presents the application side of the *Successive Convexification* framework. More notably, the author proposed a novel `SCvx-STL` procedure to handle trajectory planning problems with Signal Temporal Logic constraints. Multiple numerical examples are provided.

6.2 Future Work

There are certainly While the theory of *Successive Convexification* presented in this dissertation seems more complete than ever, there are still some enhancement could be made and questions should be answered from a theoretical perspective. For instance, [51] gives global convergence proof for `SCvx` in a continuous-time setting, subsequent efforts such as [54] are, however, all in the discrete-time setting to accommodate the convergence rate proof (e.g. Lemma 3.5.1 involves the number of constraints). This creates an interesting situation: for

strong convergence proof, we need the K-L property, which is actually valid in Banach space (see e.g. [2]). Therefore, it is possible to prove strong convergence without discretizing the dynamical system. Future effort can be made in this regard. As for the unanswered questions, one may be curious about the actual computational complexity of the `SCvx` algorithm, beyond the superlinear convergence rate. Given a set of fixed boundary conditions, it is reasonable for an algorithm practitioner to want to obtain a bound on the number of iterations in terms of the problem size, not just a limiting behavior. Therefore, It would be helpful in practice to have a computational complexity analysis done for `SCvx`. Another important consideration in practice is the "region of attraction" problem. like all the other nonlinear programming solvers, `SCvx` only produces locally optimal solutions and the quality of such solution depends on its region of attraction, and thus is determined by where the algorithm is initialized. We have observed great robustness of `SCvx` in moderately complex settings, thanks to its usage of *virtual control* and *virtual buffer zone*. However, when the vehicle is in a highly complex environment (e.g. many obstacles), `SCvx` may struggle to find a good local optimizer. One may want to integrate some features or techniques of Rapidly-Exploring Random Tree (RRT) into `SCvx`, since RRT is well-suited for finding a feasible path through compelx environment.

One the application side, more complex and possibly nested STL specification can be imposed on the system to further validate the `SCvx-STL` algorithm. Implementing `SCvx-STL` for spacecraft rendezvous is a perfect example of taking on this endeavor, due to the sheer complexity of such missions. A competing method [76, 75] in using logical statement for rendezvous control is to reformulate problems with range-triggered constraints as linear complementary problems. Note that the author's STL based approach can handle state-triggered constraints as well, and arguably in a way more natural. This warrants a comparison study. For the perception unit, it would be beneficial to carry out the second phase of the study and gauge its real-world performance and impact.

BIBLIOGRAPHY

[1] International Rendezvous System Interoperability Standards (IRSIS).

[2] P. Absil, R. Mahony, and B. Andrews. Convergence of the iterates of descent methods for analytic cost functions. *SIAM Journal on Optimization*, 16(2):531–547, 2005.

[3] 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.

[4] B. Açıkmeşe, J.M. Carson, 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.

[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] Behçet Açıkmeşe, Daniel P Scharf, Emmanuell A Murray, and Fred Y Hadaegh. A convex guidance algorithm for formation reconfiguration. In *AIAA Guidance, Navigation, and Control Conference*, page 6070, 2006.

[7] Ross Allen and Marco Pavone. A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance. In *AIAA Guidance, Navigation, and Control Conference*, page 1374, 2016.

[8] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

[9] Leonard David Berkovitz. Optimal control theory. 1974.

[10] Lars Blackmore. Autonomous precision landing of space rockets. In *in Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*, volume 46, pages 15–20.

[11] Lars Blackmore. Autonomous precision landing of space rockets. *The Bridge on Frontiers of Engineering*, 4(46):15–20, 2016.

[12] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[13] Jérôme Bolte, Aris Daniilidis, and Adrian Lewis. The Lojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4):1205–1223, 2007.

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

[15] C. Buskens and H. Maurer. Sqp-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis, and real-time control. *Journal of Computational and Applied Mathematics*, 120:85–108, 2000.

[16] Frank H Clarke. The maximum principle under minimal hypotheses. *SIAM Journal on Control and Optimization*, 14(6):1078–1091, 1976.

[17] Frank H Clarke. A new approach to lagrange multipliers. *Mathematics of Operations Research*, 1(2):165–174, 1976.

[18] Frank H Clarke. *Optimization and nonsmooth analysis*. SIAM, 1990.

[19] Andrew R Conn, Nicholas IM Gould, and Ph L Toint. *Trust region methods*, volume 1. Siam, 2000.

[20] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

[21] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 92–106. Springer, 2010.

[22] D. Dueri, Y. Mao, Z. Mian, J. Ding, and B. Açıkmeşe. Trajectory optimization with inter-sample obstacle avoidance via successive convexification. In *IEEE 56th Conference on Decision and Control (CDC)*, pages 1150–1156, 2017.

[23] Daniel Dueri, Behçet Açıkmeşe, Daniel P Scharf, and Matthew W Harris. Customized real-time interior-point methods for onboard powered-descent guidance. *Journal of Guidance, Control, and Dynamics*, pages 1–16, 2016.

[24] Daniel Dueri, Jing Zhang, and Behçet Açıkmeşe. Automated custom code generation for embedded, real-time second order cone programming. In *19th IFAC World Congress*, pages 1605–1612, 2014.

[25] R. Fletcher. *Practical methods of optimization, 2nd Edition*, volume 2. Wiley, 1987.

[26] Roger Fletcher and E Sainz de la Maza. Nonlinear programming and nonsmooth optimization by successive linear programming. *Mathematical Programming*, 43(1-3):235–256, 1989.

[27] Divya Garg, Michael Patterson, William W Hager, Anil V Rao, David A Benson, and Geoffrey T Huntington. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica*, 46(11):1843–1851, 2010.

[28] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.*, 47:99–131, 2005.

[29] A. J. Goldman and A. W. Tucker. Theory of linear programming. *Linear inequalities and related systems*, 38:53–97, 1956.

[30] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[31] William W. Hager, Jun Liu, Subhashree Mohapatra, Anil V. Rao, and Xiang-Sheng Wang. Convergence rate for a gauss collocation method applied to constrained optimal control. *SIAM Journal on Control and Optimization*, 56(2):1386–1411, 2018.

[32] S-P Han and Olvi L Mangasarian. Exact penalty functions in nonlinear programming. *Mathematical programming*, 17(1):251–269, 1979.

[33] Matthew W. Harris and Behçet Açıkmeşe. Maximum divert for planetary landing using convex optimization. *Journal of Optimization Theory and Applications*, 162(3):975–995, Sep 2014.

[34] Matthew W. Harris and Behçet Açıkmeşe. Minimum time rendezvous of multiple spacecraft using differential drag. *Journal of Guidance, Control, and Dynamics*, 37(2):365–373, 2014.

[35] 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.

[36] D.G. Hull. Conversion of optimal control problems into parameter optimization problems. *Journal of Guidance, Control, and Dynamics*, 20(1):57–60, 1997.

[37] Uros V. Kalabic, Rohit Gupta, Stefano Di Cairano, Anthony M. Bloch, and Ilya V. Kolmanovsky. MPC on manifolds with an application to the control of spacecraft attitude on SO(3). *Automatica*, 76:293–300, 2017.

[38] Yoonsoo Kim and Mehran Mesbahi. Quadratically constrained attitude control via semidefinite programming. *IEEE Transactions on Automatic Control*, 49(5):731–735, 2004.

[39] Krzysztof Kurdyka. On gradients of functions definable in o-minimal structures. In *Annales de l'institut Fourier*, volume 48, pages 769–784. Chartres: L'Institut, 1950-, 1998.

[40] J. P. LaSalle. Study of the basic principles underlying the bang-bang servo. *Tech. Rep. GER-5518*, 1953.

[41] Unsik Lee and Mehran Mesbahi. Constrained autonomous precision landing via dual quaternions and model predictive control. *Journal of Guidance, Control, and Dynamics*, 40:292–308, 2017.

[42] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2012.

[43] Xinfu Liu and Ping Lu. Solving nonconvex optimal control problems by convex optimization. *Journal of Guidance, Control, and Dynamics*, 37(3):750–765, 2014.

[44] Xinfu Liu, Zuojun Shen, and Ping Lu. Entry trajectory optimization by second-order cone programming. *Journal of Guidance, Control, and Dynamics*, 39(2):227–241, 2015.

[45] Y. Liu, A. Xu, and Z. Chen. Map-based deep imitation learning for obstacle avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8644–8649, 2018.

[46] S Lojasiewicz. Sur les trajectoires du gradient dune fonction analytique. *Seminari di geometria*, 1983:115–117, 1982.

[47] Ping Lu and Xinfu Liu. Autonomous trajectory planning for rendezvous and proximity operations by conic optimization. *Journal of Guidance, Control, and Dynamics*, 36(2):375–389, 2013.

[48] Kees Caspert Peter Machielsen. *Numerical solution of optimal control problems with state constraints by sequential quadratic programming in function space*. Technische Universiteit Eindhoven, 1987.

[49] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[50] Danylo Malyuta, Yue Yu, Purnanand Elango, and Behcet Acikmese. Advances in trajectory optimization for space vehicle control, 2021.

[51] 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 (CDC)*, pages 3636–3641, 2016.

[52] Yuanqi Mao, Daniel Dueri, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems with state constraints. *IFAC-PapersOnLine*, 50(1):4063 – 4069, 2017.

[53] Yuanqi Mao, Daniel Dueri, Michael Szmuk, and Behçet Açıkmeşe. *Convexification and Real-Time Optimization for MPC with Aerospace Applications*, pages 335–358. Springer International Publishing, Cham, 2019.

[54] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems. *arXiv e-prints*, page arXiv:1804.06539, February 2019.

[55] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. A tutorial on real-time convex optimization based guidance and control for aerospace applications. In *2018 Annual American Control Conference (ACC)*, pages 2410–2416. IEEE, 2018.

[56] J. Mattingley and S. Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[57] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[58] Yash Vardhan Pant, Houssam Abbas, Rhudii A Quaye, and Rahul Mangharam. Fly-by-logic: control of multi-drone fleets with temporal logic objectives. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 186–197. IEEE, 2018.

[59] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[60] Inigo Quilez. smooth minimum - 2013, 2013.

[61] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87. IEEE, 2014.

[62] Arthur Richards, Tom Schouwenaars, Jonathan P How, and Eric Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *Journal of Guidance, Control, and Dynamics*, 25(4):755–764, 2002.

[63] Wayne State University Mathematics Department Coffee Room. Every convex function is locally lipschitz. *The American Mathematical Monthly*, 79(10):1121–1124, 1972.

[64] Judah Ben Rosen. Iterative solution of nonlinear optimal control problems. *SIAM Journal on Control*, 4(1):223–244, 1966.

[65] Walter Rudin. *Principles of mathematical analysis*, volume 3. 1964.

[66] Vinicius Santos. Smooth min explained, 2016.

[67] Joseph A Starek, Behçet Açıkmeşe, Issa A Nesnas, and Marco Pavone. Spacecraft autonomy challenges for next-generation space missions. In *Advances in Control System Technology for Aerospace Applications*, pages 1–48. Springer, 2016.

[68] Héctor Sussmann. Some optimal control applications of real-analytic stratifications and desingularization. *Banach Center Publications*, 44(1):211–232, 1998.

[69] Héctor J Sussmann. Lie brackets, real analyticity and geometric control. *Differential geometric control theory*, 27:1–116, 1983.

[70] Héctor J Sussmann. Multidifferential calculus: chain rule, open mapping and transversal intersection theorems. In *Optimal Control*, pages 436–487. Springer, 1998.

[71] M. Szmuk, C. A. Pascucci, and B. Aikmee. Real-time quad-rotor path planning for mobile obstacle avoidance using convex optimization. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, Oct 2018.

[72] 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 (IROS)*, pages 4862–4868, 2017.

[73] Michael Szmuk and Behçet Açıkmeşe. Successive convexification for 6-dof mars rocket powered landing with free-final-time. *ArXiv e-prints*, February 2018. arXiv:1802.03827.

[74] Michael Szmuk, Utku Eren, and Behçet Açıkmeşe. Successive convexification for mars 6-dof powered descent landing guidance. In *AIAA Guidance, Navigation, and Control Conference*, page 1500, 2017.

[75] Michael Szmuk, Danylo Malyuta, Taylor P Reynolds, Margaret Skye Mceowen, and Behçet Açikmeşe. Real-time quad-rotor path planning using convex optimization and compound state-triggered constraints. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7666–7673. IEEE, 2019.

[76] Michael Szmuk, Taylor Reynolds, Behcet Acikmese, Mehran Mesbahi, and John M Carson. Successive convexification for 6-dof powered descent guidance with compound state-triggered constraints. In *AIAA Scitech 2019 Forum*, page 0926, 2019.

[77] K.C. Toh, M.J. Todd, and R.H. Tutuncu. SDPT3 — a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11(1):545–581, 1999.

[78] Panagiotis Tsiotras and Mehran Mesbahi. Toward an algorithmic control theory. *Journal of Guidance, Control, and Dynamics*, 40:194–196, 2017.

[79] Abraham P Vinod, Sean Rice, Yuanqi Mao, Meeko MK Oishi, and Behçet Açıkmeşe. Stochastic motion planning using successive convexification and probabilistic occupancy functions. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4425–4432. IEEE, 2018.

[80] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006.

[81] A. Weiss, M. Baldwin, R. S. Erwin, and I. Kolmanovsky. Model predictive control for spacecraft rendezvous and docking: Strategies for handling constraints and case studies. *IEEE Transactions on Control Systems Technology*, 23(4):1638–1647, July 2015.

[82] D.E. Wulbert. Continuity of metric projections. *Transactions of the American Mathematical Society*, 134(2):335–341, 1968.

[83] Yinyu Ye. *Interior point algorithms: theory and analysis*, volume 44. John Wiley & Sons, 2011.

VITA

Yuanqi Mao graduated from Shanghai Jiao Tong University with a Bachelor of Science in Engineering degree. He is a Ph.D student in the William E. Boeing Department of Aeronautics and Astronautics at University of Washington since 2015. He works with Prof. Behçet Açıkmeşe on optimal control theory, numerical optimization, and optimization-based guidance and control technologies with aerospace applications.

He welcomes your comments to `ymao@uw.edu`.