A Successive Convexification Optimal Guidance Implementation for the Pinpoint Landing of Space Vehicles

by

Pádraig S. Lysandrou

B.S. ECE, Cornell University, 2018

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Masters of Science

Department of Aerospace Engineering Sciences

2019

This thesis entitled:
A Successive Convexification Optimal Guidance Implementation for the Pinpoint Landing of Space Vehicles
written by Pádraig S. Lysandrou
has been approved for the Department of Aerospace Engineering Sciences

Prof. Robert D. Braun

Dr. Jay McMahon

Dr. Hanspeter Schaub

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Lysandrou, Pádraig S. (M.S., Aerospace Engineering)

A Successive Convexification Optimal Guidance Implementation for the Pinpoint Landing of Space Vehicles

Thesis directed by Prof. Robert D. Braun

Autonomous propulsive spacecraft have enabled the exploration of other planets, moons, asteroids, and other celestial bodies. Recent advances in real-time optimal guidance algorithms allow spacecraft to perform precision landing maneuvers enabling a new age autonomous and crewed missions. These algorithms compute trajectories that are locally optimal to a given objective and are able to account for key mission aspects and dynamic constraints. Therefore, these methods maximize the agility and divert capability of space vehicles subject to environmental dispersions while they are maneuvering to their terminal destinations.

This thesis focuses on the implementation and development of a 6 degree-of-freedom (DoF) free-final-time guidance algorithm that solves the powered descent guidance (PDG) problem subject to vehicular, environmental, and mission constraints. I present a solution to this problem utilizing the successive convexification nonlinear Model Predictive Control (MPC) framework and modified Rodrigues parameters (MRPs) as the attitude formalism.

Dedication

In no particular order, to my loving family: Plato, Carolyn, Helena, Maria, and animals.

Acknowledgements

Foremost, I am thankful to Dr. Robert D. Braun for his support and advising through my work at Colorado Boulder. I would like to thank Professor Jay McMahon and Professor Hanspeter Schaub for serving on my committee and for their fulfilling coursework and advising. Additionally, I would like to thank Dr. Mason Peck for his four years of support at Cornell, which serve as my foundation in this field.

As an unordered set, I am grateful for the insightful technical discussions I have had with Shez Virani, Daniel Aguilar-Marsillach, the EsDL research team, Sven Niederberger, Benjamin Chung, Janis Maczijewski, Behçet Açikmeşe, Miki Szmuk, Skye Mceowen, Ian Garcia, Joe Barnard, and many more. Similarly, I would like to thank my SpaceX colleagues for grounding me in reality while letting me play with vehicles of the heavens. Go Falcon, Dragon, and Starlink!

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

1.1 Background

Pin-point landing has been of significant interest for a variety of applications. These include safely landing scientific payloads and humans on other planets, returning them back to Earth, and reusable launch vehicles (RLVs). The ability to soft-land a rocket is fundamentally disruptive to the launch industry and has already showed promise in reducing the cost of getting to space [11]. For planetary exploration, it will be a requirement to land near a site of scientific interest, base, or refueling stations. It is apparent that having a robust and reliable powered descent guidance routine will be a necessity for our future space transport infrastructure.

Every pinpoint landing problem begins with an entry phase where the vehicle descends through an atmosphere to a point where the landing regime begins. Many Mars entry, descent, and landing schemes enter the atmosphere and decelerate via an ablative shield and supersonic parachutes. The parachute is then cut away to allow powered descent to occur. With atmospheric qualities being non-deterministic, the position in which the descent phase must begin is uncertain. Therefore, landing algorithms which maximize the divert capability of the vehicle by minimizing the fuel consumption of the final landing time are of interest. We define powered descent guidance as the generation of a fuel-optimal trajectory and/or input sequence that takes the vehicle from some initial state condition to a prescribed final state in a uniform gravitational field with standard vehicle given thrust magnitude and direction constraints in finite time. Figure 1.1 shows an example of an RLV in a return-to-launch-site maneuver (RTLS).

The convex optimization framework is exploited because it is conducive of real-time on-board implementation and has guaranteed convergence properties with deterministic criteria. The convex programming algorithm to solve powered descent guidance that I will present has non-convex controls constraints and will be posed as a finite-dimensional SOCP problem. SOCPs have low complexity and can be solved in polynomial time [8]. Interior-point numerical methods compute optimal solutions with deterministic stopping criteria and are conducive to on-board implementation.

This thesis focuses on the implementation and development of a 6 degree-of-freedom (DoF) guidance algorithm that solves the non-convex nonlinear powered descent guidance (PDG) problem. I employ a method called successive convexification or SCvx. In this method, we first initialize the algorithm with a reference trajectory, then linearize and discretize the problem as an SOCP problem. We then structure the problem as an iterative solution process where the current problem is linearized about the previous trajectory. We do this in such a way that the solution satisfies the original nonlinear dynamics, non-convex constraints, and other state and control constraints.

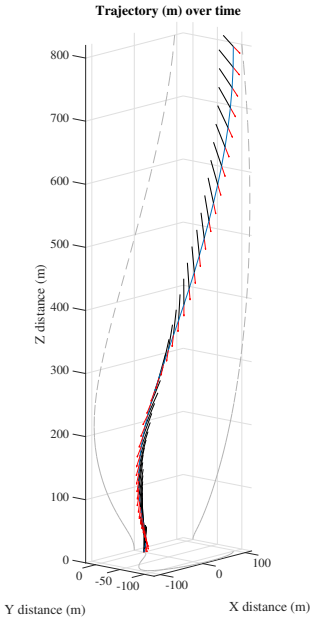

Figure 1.1: Landing Trajectory Example

1.2 Motivation

Autonomous optimal guidance strategies enable vehicle recovery from a wider range disturbances accrued from the entry and descent process. This allows the vehicle to optimally maximize the divert and transport capability after dispersions, increasing the probability of safe landing and mission success. The proof of this value is apparent in the efforts and successes of commercial space companies landing and reusing space launch vehicles in recent years. The presented work has a number of desirable traits which make it amenable to implementation in this context. To sufficiently motivate and further our discussion, we must discuss what factors are important in the selection of a flight guidance algorithm.

1.2.1 Attributes of a Good Guidance Algorithm

Nominally, the guidance routine must be computationally tractable. If it is run iteratively, the cycle time could be from 1Hz for slow systems, up to 20Hz for fast systems. This is one of the main reasons guidance tends to be run "offline," or not during flight, and stored in look up tables (LUTs). This may also be called "implicit" guidance. These are simply stored solutions for the flight computer to use a later time. Sophisticated guidance algorithms that have a full understanding of the problem at hand tend to be quite intensive. So the engineer is always looking for simple and fast solutions.

A guidance routine must be tractable not only for speed, but also to reduce resource congestion. Other processes, namely navigation and control, also require compute power and should not be throttled. Real-time operating systems have strict schedulers where each task is guaranteed to run to completion in finite time. If a deadline is unable to be met, a priority inversion system is put in place to pause one task and move on to another, for single-threaded operations.

The guidance routine must be accurate to the vehicle, actuator, and environmental dynamics. Often the routine is not aware of larger perturbations and can accrue significant errors. Even offline LUT methods fall short here, especially when a guidance phase starts with high uncertainty in the

initial condition.

An online method, or one "explicitly" computed onboard the flight computer during operation, is a good solution for systems where the initial condition of that phase of flight may disperse widely. If there is high confidence in the navigation solution at the beginning of the guidance routine, but it errs from the nominal point, an online method is attractive. This algorithm can calculate the new, optimal solution for the exact problem at hand. If a rocket is 20 meters from the nominal position, a new state trajectory and input history can be computed on the spot and fed to the control loop. A LUT method would require a neighboring optimal control solution to bring it back to the original proposed trajectory, which at this point is suboptimal. This scenario has no guarantees of meeting the mission requirements and constraints.

Optimality is important. If the vehicle has fast moving dynamics and has stringent fuel requirements, a sense of optimality with respect to these key metrics becomes important. If your vehicle must land on the surface of Mars at a specific position, would you not want to get there while minimizing fuel usage to ensure you don't overspend and crash? What if the vehicle must be somewhere in the least amount of time? A trajectory and input sequence which is tailored to these metrics will then be required.

Time ambiguity is another important factor in a good guidance algorithm. Will this algorithm create a control history which meets your objective with free or fixed-final-time? The former means that the terminal time is free and is a solution of the algorithm itself: I will land on Mars with this trajectory, these inputs, and at exactly 34.5235 seconds from now. The latter first asks you what time you'd like to be there, then produces the required trajectory. The first tends to have a larger sense of optimality than the smaller, more restricted second problem.

A good algorithm must also be conducive to the inclusion of mission, vehicular, and environmental constraints. Do the crew members enjoy 10g's of acceleration and 45 degrees per second of rotation? Would you like to stay above the ground and never use more fuel than you have? These are important pieces of information to have when making a vital decision about the motion and plan of your space vehicle.

Guarantees of convergence and aforementioned optimality are important for validation and testing of the GN&C subsystem. What if the algorithm actually never converges in flight, or the output is entirely suboptimal and performs poorly. Having these guarantees mathematically proven becomes very important when it comes to flying critical hardware and humans.

Fault detection, isolation, and recovery (FDIR) is another critical component. Can the state machine of the vehicle or craft detect a fault and correctly recover? If a system, for some reason, behaves anomalously, there must be a robust way to detect this and communicate to other subsystems what has happened. If a guidance routine does not converge, another trajectory or control strategy must be chosen and used quickly before something off-nominal occurs.

The performance with respect to dispersions and sensitive parameters is also important. The algorithm performance in the face of randomness must not degrade significantly. Additionally, the routine should not be overly sensitive to any single parameter. If any algorithm is highly sensitive to a single parameters, say a valve actuation speed variation, it could significantly limit the operational dynamic range of that algorithm. These analyses should be performed rigorously a priori before flight implementation.

1.3 Literature Review and Related Research

Optimal control and guidance strategies are generally split into two categories: Indirect and Direct methods. Classical indirect methods are applications of calculus of variations and Pontryagin's Maximum Principle to derive the necessary conditions of optimality, such as adjoint equations and transversality conditions. Although the optimality of the indirect method can be guaranteed, solving the resulted two-point boundary value problem is a difficult task, and good initial guesses of the adjoint variables are hard to come by.

The direct method does not require explicit derivation of first-order necessary conditions; instead, the original optimal control problem is approximated with a parameter optimization problem and solved using nonlinear programming (NLP) algorithms, such as sequential quadratic programming (SQP) algorithms. Although large-scale problems can be handled due to the development of

NLP algorithms, the solution process is still time-consuming for complicated problems and dependent upon good initial guesses. Convex optimization approaches are direct optimization methods, however can be reliably solved in polynomial time [18].

Polynomial Guidance was famously used on the Apollo lunar lander, first landing men on the moon in 1969. The Apollo guidance computer (AGC) had limited compute capability with the algorithms requiring manual re-targeting and flight by the mission pilot. This algorithm was an analytical solution to a boundary valued problem with a fixed time-to-go [12]. The final braking law is represented by a quartic polynomial function which connect the initial and terminal states. This result does not satisfy vehicle thrust constraints, and errors introduced by this law must be compensated for by other calculations. Besides the boundary values, there are no state constraints and no optimality with respect to fuel or time. Although compute power was so limited, all lunar powered descent missions were successful thanks to the polynomial and astronaut guidance. Modified versions of this law are still in the literature with an outer time-of-flight search mechanism to effectively make it a free final time-to-go algorithm as well as adding in fuel optimality [10].

Powered Explicit Guidance or PEG is another prevalent guidance routine. This was developed to handle all phases of the Shuttle exoatmospheric powered flight and their cutoff constraints [17]. This is a simplified vector form of a linear tangent steering law. It provides robust guidance for situations which vary widely in thrust-to-weight ratios (TWRs). During the second stage of Shuttle ascent, the TWRs range from 1 to 3 with altitude, velocity, flightpath angle, and orbital plane cutoff constraints. The objective of this algorithm is to generate steering and throttle commands such that the cutoff constraints are satisfied in a fuel-efficient manner. This linear tangent scheme is based on an indirect calculus of variations solution to the minimum-fuel ascent trajectory problem. The thrust attitude takes the form of $\lambda_F = \lambda_v + \dot{\lambda}(t - t_\lambda)$ where λ_F is the vector defining the thrust direction, λ_v is the unit vector in the direction of velocity-to-be-gained, $\dot{\lambda}$ is a vector normal to λ_v representing the rate of change of λ_F. The variable t is continuous time where t_λ is a chosen time such that the total velocity change due to thrust is along the vector λ_v. To satisfy the required cutoff constraints, the algorithm takes the form of a predictor corrector mechanism where

λ_v is computed such that the terminal velocity constraints are satisfied.

A backwards recursive version of this steering law was developed for the powered descent guidance problem as well. This law minimizes the control required for soft landing but does not guarantee to meet actuator constraints and is naive to the nonlinear dynamics.

Lossless Convexification is a method to produce a perfect convex second order cone program (SOCP) sub-problem from an originally non-convex problem statement. It is used for convexifying a non-convex constraint without loosing any precision. This method, championed by Blackmore, Açikmeşe, and Ploen, is at the heart of the G-FOLD landing algorithm [7] [3] [1]. The Guidance algorithm for Fuel Optimal Large Diverts, or G-FOLD, is a recent advancement in PDG solutions. It very quickly calculates a 3DoF fuel optimal divert maneuver with free-final-time, inequality and equality constraints on the states and inputs, and losslessly convexifying the lower thrust constraint. Some of these constraints include a glideslope around the target and maximum velocity. The routine solves a lossless convex SOCP problem iteratively with an outer time of flight search loop to find the optimal final time. This was successfully demonstrated in a family of flight experiments on Masten Space landing vehicles [2] [19].

Unfortunately only a couple types of constraint can be convexified in this fashion, where the remaining nonlinear dynamics are the primary sources of non-convexity. These must be included into the problem statement with another process.

1.4 Brief Introduction to Convex Optimization

A convex optimization problem is one that takes the following form:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, \quad i = 1, \cdots, m \end{aligned}$$

where each of the functions $f_0, ..., f_m \ : \mathbb{R}^n \to \mathbb{R}$ are convex. This means they satisfy the inequality $f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \quad \forall x, y \in \mathbb{R}^n$ where all $\alpha, \beta \in \mathbb{R}$ with $\alpha + \beta = 1$ and each positive

semi-definite. Many optimization problems are just special cases of this problem, including the general least-squares and linear programming problems.

Using convex optimization is much like using any other optimization framework. If a problem can be identified or formulated as a convex problem, then we can solve it efficiently. Recognizing a convex function is nontrivial and there are many tricks for transforming non-convex problems into convex ones. The Boyd and Vandenberghe text [8] gives significant insight into these tricks and the skills required to recognize and formulate convex problems.

1.5 On Successive Convexification

The successive convexification framework (SCvx) is able to solve optimal control problems with nonlinear dynamics and non-convex state and control constraints. It does this by iteratively solving convex optimization sub-problems, obtained by linearizing non-convexities in dynamics and constraints around the previous iteration solution. These sub-problems employ techniques of virtual control (dynamic relaxation), virtual buffer zones, and trust regions to prevent solution artificial infeasibility and artificial unboundedness. This linearization acts as an approximation, but the solution is driven to convergence within the user's tolerances to solve exactly the originally proposed non-convex optimal control problem with local optimality.

For general real-time autonomy tasks where safety and determinism are prioritized, it is often much better to find a locally optimal solution quickly rather than a globally optimal solution slowly. Generally speaking, nonlinear programming tends to be the method of choice for locally optimal solutions. However, their convergence behaviour is dependent upon the initial guess provided to the solver and do not offer bounds on computational effort required for convergence. These facts are at odds with requirements for real-time embedded applications.

So we turn to convex optimization methods with can be reliably solved in polynomial time [18]. Sequential convex programming (SCP) offers a way to solve problems with more general nonlinear dynamics and non-convex constraints. While SCP performs well empirically, no general convergence results have been reported. SCvx differs from other SCP approaches in many ways,

including proofs for (weak and strong) global convergence and superlinear convergence rate [14].

1.6 Statement of Scope

In this thesis, we shall describe the nonlinear equations of motion for a space vehicle, pose a guidance problem, and then modify it such that it is iteratively solveable. We do this by creating a linear time varying version of the dynamics, discretizing it, and convexifying the problem contraints. Once this is done, we generate a reference trajectory as a linearization path for the SCvx to solve the problem with and iteratively feed the output trajectory and input history back into the problem until convergence. The derivations and analysis are shown in this document.

Chapter 2

The Powered Descent Guidance Problem

2.1 Problem Description

The goal of the presented algorithm is to generate optimal translational and attitude trajectory profiles that are dynamically feasible and amenable. This means that the modeled vehicle should abide by all state boundary conditions, actuator constraints, and the proposed dynamics. We shall initially define this problem as a continuous-time non-convex dynamical form and then explore converting this to a disciplined convex program. We shall discuss the dynamic, control, initial, and terminal constraints that must be met throughout the problem.

In order to maximize the divert capability of the vehicle, we propose the objective of minimizing the final time of the solution. Although not proven here, this can be considered a proxy to the minimum-fuel consumption problem, as the non-convex constraint of minimum-thrust and single-ignition requires that the engine be on for the duration of the landing phase. In this scenario, the fuel-consumption cost is strictly increasing monotonic, and the sooner the terminal conditions can be met, the fewer the total cost. A similar free-ignition-time modification can be made to further decrease this cost and optimize the ignition time [21].

2.2 Definitions and Notation

Now we shall define the two frames of importance. The $\mathcal{F}_{\mathcal{N}} : \{\mathcal{O}_{\mathcal{N}}, \hat{\boldsymbol{n}}_1, \hat{\boldsymbol{n}}_2, \hat{\boldsymbol{n}}_3\}$ frame defines an inertially fixed Up-East-North reference frame where the origin $\mathcal{O}_{\mathcal{N}}$ located at the landing site. This can easily be changed to the local-vertical local-horizontal (LVLH) or other useful frame

definition. The $\mathcal{F}_\mathcal{B} : \{\mathcal{O}_\mathcal{B}, \hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2, \hat{\boldsymbol{b}}_3\}$ frame is a body fixed frame where the x-axis is aligned vertically with the vehicle, or aligned with the thrust vector at zero thrust vector control (TVC) deflection angle. The Y-axis points out of the side of the cylindrical vehicle and the Z-axis completes the right handed triad.

Here forward, it should also be assumed that the vectorial derivative, shown by $\dot{\mathbf{r}}$, is an inertial time derivative. Derivatives in frames other than our inertial will be indicated otherwise as ${}^{\mathcal{X}}\frac{d\mathbf{r}}{dt}$. Any vector shown as ${}^{\mathcal{X}}\mathbf{r}$ is in the $\mathcal{X}$ frame, and similarly anything without this left superscript is frameless. Therefore the vector $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$ is the frameless angular velocity vector of the body frame with respect to the inertial frame. We also use the notation $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{R}_{++}$ to denote the set of real values, non-negative real values, and positive real values respectively.

2.3 Vehicle Dynamics

2.3.1 Translational Dynamics

Given that most powered descent maneuvers are done within kilometers of a site, and at speeds much less than orbital velocities, we can use a simplified gravitational acceleration assumption with a non-rotation planet. Similarly, we assume aerodynamic forces to be negligible, representative of a Mars landing scenario. However, as we will show in a later section, any nonlinear dynamics can be incorporated, as they will be represented as a linear time-varying (LTV) system which is successively convexified to solve the nonlinear problem exactly to a locally optimum solution.

The algorithm as presented has the vehicle actuated by a single gimbaled thruster at the bottom of the vehicle. It should be stated that the algorithm can readily accommodate other actuator geometries and configurations. This engine has feasible thrust magnitude and efficiency, as well as standard gimbal range for agile landing or vertical-takeoff-vertical-landing (VTVL) vehicles. To be inclusive of actuator dynamics, we assume that the engine has maximum and minimum thrust bounds. Most rocket engines have a minimum throttle percentage, below which the engine does

not perform well or in a stable manner. During the powered descent routine, once ignition occurs, the engine is not turned off or re-ignited until commanded at the terminal state. This minimum thrust constraint is a source of non-convexity.

It is critical to capture the mass depletion dynamics, proportional to the magnitude of the thrust generated by the engine. For simplicity, we assume that the inertia matrix and the position of the center-of-mass is constant throughout the trajectory although these modifications can also be included in the dynamic formulation. We use the constant $\alpha_{\dot{m}}$, a function of the specific impulse, as the mass depletion parameter. This is the inverse of the mass flow rate, which is the total flow rate of the propellants to the engine. Additionally, we assume a constant specific impulse throughout the throttleable region, which may not always be true. Normally an engine operating at a lower thrust than nominal may may be less efficient and have a lower specific impulse. These effects are small enough to be ignored in this problem statement. Therefore we have that

$$\alpha_{\dot{m}} = \frac{1}{I_{sp} g_0} \tag{2.1}$$

$$\dot{m}(t) = -\alpha_{\dot{m}} \left\| \mathbf{F}_{thrust}(t) \right\|_2 \tag{2.2}$$

With our assumptions, we can trivially express the translational dynamics and forces acting on the vehicle in the inertially frame. They are as follows:

$$\dot{\mathbf{r}}(t) = \mathbf{v}(t) \tag{2.3}$$

$$\dot{\mathbf{v}}(t) = \frac{\mathbf{F}_{thrust}(t)}{m(t)} + \mathbf{g} \tag{2.4}$$

As stated previously, any extra nonlinear terms can be added to this formulation and still be solved exactly. This may become useful in the hypersonic entry, supersonic retropropulsive, or any perturbing regimes. Moving forward, we will assume the translational dynamics to be in the inertial frame and all rotational dynamics to be in the body fixed frame.

2.3.2 Attitude Dynamics and Formalisms

In this formulation, the vehicle is treated as a rigid body, although solutions with non-rigid structures could be readily supported. From Euler's principal rotation theorem, any coordinate reference frame can be brought from an arbitrary initial condition to an arbitrary final orientation by a single rigid rotation through a principle angle ϕ about a principal axis $\hat{\boldsymbol{e}}$. This axis is fixed in both the initial and final orientation. Therefore $\hat{\boldsymbol{e}} = [C]\hat{\boldsymbol{e}}$, where $[C] \in SO(3)$ is a rotation mapping in $\mathbb{R}^{3\times3}$ taking a vector from the initial orientation to the final, shows that $\hat{\boldsymbol{e}}$ is an unit eigenvector of the $[C]$ transform whose eigenvalue is $+1$. This rotation mapping matrix, a direction cosine matrix (DCM), has nine parameters and can be cumbersome although exact. Generally, we prefer to use mappings with fewer parameters. The right-handed $SO(3)$ group DCM has a determinant $+1$ with inverse/transpose both being the inverse mapping: $[\mathcal{BN}]^T = [\mathcal{NB}] = [\mathcal{N} \leftarrow \mathcal{B}]$. Similarly, they can be multiplied to encode compound rotations, say, from sensor frame to body frame to inertial.

Quaternions The attitude formalism initially used in [23] are Euler parameters, or quaternions. They to denote the attitude of the vehicle between the $\mathcal{F}_\mathcal{B}$ and $\mathcal{F}_\mathcal{N}$ frames, $q_{\mathcal{B}/\mathcal{N}}(t)$ on the unit sphere. This produces the DCM $[\mathcal{BN}]$. We use the angle-axis form of the quaternion, noted here:

$$\boldsymbol{q}_{\mathcal{B}/\mathcal{N}} \triangleq \begin{bmatrix} \cos(\frac{\phi}{2}) \\ \hat{\boldsymbol{e}}\sin(\frac{\phi}{2}) \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{2.5}$$

Note that the Euclidean norm $||\boldsymbol{q}_{\mathcal{B}/\mathcal{N}}||_2 = 1$ must be constrained to the unit sphere at all times. Lie group methods of integration are normally employed to ensure that the unit norm constraint is satisfied [5]. It soon becomes apparent that there can become a hemispherical ambiguity in the quaternion parameter as ϕ grows larger than π. However, this is resolved as the sign of the q_0 parameter can be checked to switch from long to short rotation quaternions. We similarly see that quaternion symmetries are such that $\boldsymbol{q}$ and $-\boldsymbol{q}$ produce the same rotation mapping. As it will

become useful later, we must now introduce the quaternion-to-DCM mapping in equation 2.6 [20].

$$C_{\mathcal{B}/\mathcal{N}} = [\mathcal{BN}] = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{2.6}$$

In attitude dynamics, we use $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}(t) \in \mathbb{R}^3$ to denote the angular velocity vector of the vehicle rigid body frame with respect to the inertial frame. This should recognized as the nominal output of your gyroscope, the body frame angular rate. We also use the $\left[\mathbf{r}^{\times}\right]$ operator to denote the skew symmetric matrix form of the vector $\mathbf{r}$. The inertia tensor instantiated in the $\mathcal{F}_B$ frame, about the body center of mass, is written as ${}^{\mathcal{B}}[I_c] \in \mathbb{R}^{3\times3}$. The body frame inertia matrix is the solution to the ${}^{\mathcal{B}}[I_c] = \int_B -[\boldsymbol{r}^{\times}][\boldsymbol{r}^{\times}]dm$ where $\boldsymbol{r}$ are the vectors to each infinitesimal mass element from the center of mass. This matrix must be symmetric positive semi-definite and abide by the triangle inequality.

Quaternions are non-unique and non-singular with their kinematic differential equation being elegant and bilinear 2.7. These can be attractive for problems where the dynamics are linearized.

$$\dot{\boldsymbol{q}}_{\mathcal{B}/\mathcal{N}} = \frac{1}{2}B_w({}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}})\boldsymbol{q}_{\mathcal{B}/\mathcal{N}} = \frac{1}{2}B_q(\boldsymbol{q}_{\mathcal{B}/\mathcal{N}})\ {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \tag{2.7}$$

where the B_w and B_q matrices are defined as

$$B_w(\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}) = \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \qquad B_q(\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & -q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{2.8}$$

Modified Rodrigues Parameters Now that we are intimately familiar with Euler parameters, we shall introduce Modified Rodrigues Parameters (MRPs), another useful and popular rotation formalism. These form a three parameter set without a norm constraint. We derive them from quaternions, as shown in 2.9

$$\sigma_i = \frac{q_i}{1+q_0} \quad i = 1,2,3 \tag{2.9}$$

$$\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}} = [\sigma_1\ \sigma_2\ \sigma_3]^T = \tan\frac{\phi}{4}\hat{\boldsymbol{e}} \tag{2.10}$$

taking an argument of $\boldsymbol{q}_{\mathcal{B}/\mathcal{N}}$. The reverse mapping is simply $q_0 = \frac{1-\sigma^2}{1+\sigma^2}$ and $q_{1:3} = \frac{2\sigma_{1:3}}{1+\sigma^2}$ where σ^2 here forward will represent the norm squared of the MRP. We see that small angle approximations can be made for a wider range of displacements. It is clear that there exists a singularity at $\pm 2\pi$. However, handling this singularity, in a computational sense, is very simple. We perform a switch to the MRP shadow set when the angular displacement is $\phi \geq \pi$ and define the shadow set as $\boldsymbol{\sigma}^{\mathcal{S}} = \frac{-\boldsymbol{\sigma}}{\sigma^2} = \tan\frac{\phi-2\pi}{4}\hat{\boldsymbol{e}}$. Simply:

Algorithm 1 MRP Switching

```
procedure MRP_SWITCH(σ_k)
    if ‖σ_k‖ > 1 then
        σ_k = −σ_k/σ_k^2
    end if
end procedure
```

However, if we constrain ourselves to never encounter a full rotation, we will not need such a structure. In optimization routines where large angular displacements must be made, it could be possible to encode switches as a state triggered constraint (STC) [21]. Now we shall recognize the MRP-to-DCM mapping that will be required 2.11:

$$C_{\mathcal{B}/\mathcal{N}} = [\mathcal{BN}] = \mathbb{I}_3 + \frac{8[\boldsymbol{\sigma}^\times]^2 - 4(1-\sigma^2)[\boldsymbol{\sigma}^\times]}{(1+\sigma^2)^2} \tag{2.11}$$

taking an argument of $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ and where $\mathbb{I}_3$ is the identity matrix. The kinematic differential equation is 2.12

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}[(1-\sigma^2)\mathbb{I}_3 + 2[\boldsymbol{\sigma}^\times] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T]\ {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} = \frac{1}{4}B_\sigma(\boldsymbol{\sigma})\ {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \tag{2.12}$$

as a function of the body frame vehicle angular rate.

Differentiating the angular momentum vector of system, and making the rigid body assumption, we have:

$$\dot{\boldsymbol{H}} = \mathbf{L}_c = [I_c]\dot{\boldsymbol{\omega}} + [\boldsymbol{\omega}^\times][I_c]\boldsymbol{\omega} \tag{2.13}$$

where the torque acting on the vehicle is written as $\mathbf{L}_c(t) \in \mathbb{R}^3$ in the body frame. Rearranged, we

have the canonical Euler rotational equations of motion:

$$\dot{\boldsymbol{\omega}} = [I_c]^{-1}(\mathbf{L}_c - [\boldsymbol{\omega}^{\times}][I_c]\boldsymbol{\omega}) \tag{2.14}$$

While the moment of inertia matrix would normally be determined from the design of the vehicle, we shall simply use a cylinder rotating about it's center with a homogeneous mass distribution for out presentation. We also assume that the mass is constant throughout the optimization for this matrix, although that modification can be made if required. We use the following expression:

$$[I_c] = \begin{bmatrix} \frac{1}{2}mr^2 & 0 & 0 \\ 0 & \frac{1}{12}\left(3r^2 + h^2\right) & 0 \\ 0 & 0 & \frac{1}{12}\left(3r^2 + h^2\right) \end{bmatrix} \tag{2.15}$$

2.4 Boundary Conditions and State Constraints

The boundary conditions for the proposed guidance routine are simple. We must have hard constraints for the initial conditions, being the state vector of the vehicle when the routine is initialized, and terminal state constraints. At the start of the guidance routine, we initialize the states with the current vehicle state vector:

$$m(0) = m_0, \quad {}^{\mathcal{N}}\mathbf{r}(0) = \mathbf{r}_0, \quad {}^{\mathcal{N}}\mathbf{v}(0) = \mathbf{v}_0, \quad \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}(0) = \boldsymbol{\sigma}_0, \quad {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}(0) = \boldsymbol{\omega}_0 \tag{2.16}$$

with our given terminal state constraints:

$${}^{\mathcal{N}}\mathbf{r}(0) = \mathbf{0}, \quad {}^{\mathcal{N}}\mathbf{v}(0) = \mathbf{0}, \quad \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}(0) = \mathbf{0}, \quad {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}(0) = \mathbf{0} \tag{2.17}$$

leaving the final mass unconstrained and assuming the landing site to be the origin. An upright attitude assuming the vehicle has landing hardware. We leave the terminal mass unconstrained. Of course these can be modified to fit arbitrary landing requirements. The problem proposed in [22] does not constrain the initial attitude of the vehicle, but we will in this formulation.

Now, let us look at the state constraints that must be met. The vehicle propulsion system is limited in fuel which manifests itself as this inequality constraint:

$$m_{dry} - m(t) \leq 0 \tag{2.18}$$

We shall apply a glide-slope constraint such that the vehicle approaches the landing point from above, limiting large lateral diverts in the terminal phase. We form the convex constraint using the angle γ_{gs}. This becomes a simple geometrical argument that $\tan\gamma_{gs} \leq \frac{r_{Up}}{\left|\left|[r_{East}\ \ r_{North}]\right|\right|}$. We can form this with the following:

$$\tan\gamma_{gs}||[\hat{\boldsymbol{n}}_2 \quad \hat{\boldsymbol{n}}_3]^T\mathbf{r}(t)||_2 - \hat{\boldsymbol{n}}_1^T\mathbf{r}(t) \leq 0 \tag{2.19}$$

This creates an upward facing cone about the landing point that the vehicle must not lie outside of. This type of convex constraint can also be useful in avoiding rocky terrain and enforcing a landing from directly above an area, minimizing lateral movement close to the ground.

It is also helpful to restrict the attitude of the vehicle such that it does not tilt over a prescribed angular displacement. This could be to maintain visibility for terrain relative navigation sensors or to give human passengers visibility over the terrain or landing surface. Therefore we can constrain the angle between the inertial frame Up unit vector and the bore-sight body vector of the vehicle. We start with the constraint derivation $\hat{\boldsymbol{b}}_1 \cdot \hat{\boldsymbol{n}}_1 \geq \cos\psi_{max}$:

$$\left([\mathcal{N}\mathcal{B}] \begin{bmatrix}1\\0\\0\end{bmatrix}^{\mathcal{B}} \right)^T \begin{bmatrix}1\\0\\0\end{bmatrix}^{\mathcal{N}} \geq \cos\psi_{max} \tag{2.20}$$

this selects the $(1,1)$ element of the MRP-to-DCM matrix in equation 2.11. Therefore the constraint becomes:

$$1 - \frac{8(\sigma_2^2+\sigma_3^2)}{(1+\sigma^2)^2} \geq \cos\psi_{max} \tag{2.21}$$

This quaternion version of this same constraint is $1-2(q_2^2+q_3^2) \geq \cos(\psi_{max})$. The quaternion unity identity must be used in this derivation. The quaternion definition is convex, but the MRP version is not. However, we can circumnavigate this issue by using the direct MRP identity $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}} = \tan\frac{\phi}{4}\hat{\boldsymbol{e}}$. Therefore, we can constrain the entire MRP such that the vehicle does not exceed ψ_{max} radians in total angular displacement from the zero angle. The zero MRP is defined as the upward unit vector in the target inertial frame.

$$\left|\left|\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}(t)\right|\right|_2 \leq \tan\left(\frac{\psi_{max}}{4}\right) \tag{2.22}$$

Nominally the vehicle should not have large angular rates throughout the descent and landing, so we can simply constrain this as well.

$$\left\|\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}(t)\right\|_2 \leq \omega_{max} \tag{2.23}$$

Finally we must constrain the commanded thrust magnitude. As stated before, engines have a minimum and maximum thrust region $[T_{min}, T_{max}] \in \mathbb{R}_{++}$ in which they operate. Also, recall that we have made the single-ignition assumption. The engine thrust vector control system there is some limit to the amount of gimbal angle we can perform $\delta_{\mathrm{TVC}_{max}}$.

$$0 < F_{min} \leq \left\|\boldsymbol{F}_{thrust}(t)\right\|_2 \leq F_{max} \tag{2.24}$$

$$\cos(\delta_{\mathrm{TVC}_{max}}) \left\|\boldsymbol{F}_{thrust}(t)\right\|_2 \leq \hat{\boldsymbol{b}}_1^T \boldsymbol{F}_{thrust}(t) \tag{2.25}$$

it is clear that the upper thrust bound is clearly convex but the lower bound creates a non-convex constraint. This shall be handled later in our discretization step.

2.5 Continuous Time Problem

Putting this all together, we can pose the continuous time optimization problem. In this form, it is non-convex and requires conditioning to work into the convex programming framework. As stated, our objective is to minimize the time-of-flight required to get to the terminal conditions subject to the aforementioned constraints, dynamics, and boundary conditions. The state vector, thrust commands, gimbal angles, and final time are optimization variables and are considered the solution to this problem. We pose this in Problem one: 2.5.

Problem 1: Continuous Time Non-Convex Free-Final-Time

Cost Function:

$$\min_{\mathbf{x},\ \mathbf{F}_{th}} \ t_f$$

Boundary Conditions:

$$m(0) = m_0, \qquad {}^{\mathcal{N}}\mathbf{r}(0) = \mathbf{r}_0, \qquad {}^{\mathcal{N}}\mathbf{v}(0) = \mathbf{v}_0, \qquad \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}(0) = \boldsymbol{\sigma}_0, \qquad {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}(0) = \boldsymbol{\omega}_0$$

$$ {}^{\mathcal{N}}\mathbf{r}_T = \mathbf{0}, \qquad {}^{\mathcal{N}}\mathbf{v}_T = \mathbf{0}, \qquad \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}_T} = \mathbf{0}, \qquad {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}_T} = \mathbf{0}$$

Dynamics:

$$\dot{m}(t) = -\alpha_{\dot{m}} \left\| \mathbf{F}_{th}(t) \right\|_2$$

$${}^{\mathcal{N}}\dot{\mathbf{r}}(t) = \mathbf{v}(t)$$

$${}^{\mathcal{N}}\dot{\mathbf{v}}(t) = \frac{[\mathcal{N}\mathcal{B}(\boldsymbol{\sigma})]\ {}^{\mathcal{B}}\mathbf{F}_{th}(t)}{m(t)} + {}^{\mathcal{N}}\mathbf{g}$$

$$\dot{\boldsymbol{\sigma}}_{\mathcal{B}/\mathcal{N}} = \frac{1}{4}\left[(1-\sigma^2)\mathbb{I}_3 + 2[\boldsymbol{\sigma}^{\times}] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T\right]\ {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$$

$${}^{\mathcal{B}}\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} = [I_c]^{-1}\left([\boldsymbol{r}^{\times}_{\mathrm{COM}}]\ {}^{\mathcal{B}}\mathbf{F}_{th}(t) - [\boldsymbol{\omega}^{\times}][I_c]\boldsymbol{\omega}\right)$$

State and Control Constraints:

$$m_{dry} - m(t) \leq 0$$

$$||[\hat{\boldsymbol{n}}_2 \ \ \hat{\boldsymbol{n}}_3]^T \mathbf{r}(t)||_2 \tan\gamma_{gs} - \hat{\boldsymbol{n}}_1^T \mathbf{r}(t) \leq 0$$

$$\left\| \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}(t) \right\|_2 \leq \tan\left(\frac{\psi_{max}}{4}\right)$$

$$\left\| \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}(t) \right\|_2 \leq \omega_{max}$$

$$0 < F_{min} \leq \left\| \boldsymbol{F}_{th}(t) \right\|_2 \leq F_{max}$$

$$\cos(\delta_{max}) \left\| \boldsymbol{F}_{th}(t) \right\|_2 \leq \hat{\boldsymbol{b}}_1^T \boldsymbol{F}_{th}(t)$$

$$\left\| \delta_{\mathrm{TVC}_{t+dt}} - \delta_{\mathrm{TVC}_t} \right\|_2 \leq \dot{\delta}_{\mathrm{TVC}_{max}} dt$$

Chapter 3

Convex Formulation

Now we shall derive the convex form of Problem 1. We will convert the non-convex continuous free-final-time problem to a convex fixed-final-time problem. This will be a second order cone sub-problem. We will solve this sub-problem repeatedly to convergence or "successively." This successive process turns each subproblem into a larger free-final-time algorithm.

3.1 Linearization

Let us define the state vector $\mathbf{x}(t) \in \mathbb{R}^{14\times 1}$ and our control vector $\mathbf{u}(t) \in \mathbb{R}^{3\times 1}$:

$$\mathbf{x}(t) \triangleq \begin{bmatrix} m(t) & {}^{\mathcal{N}}\mathbf{r}^T(t) & {}^{\mathcal{N}}\mathbf{v}^T(t) & \boldsymbol{\sigma}^T_{\mathcal{B}/\mathcal{N}}(t) & {}^{\mathcal{B}}\boldsymbol{\omega}^T_{\mathcal{B}/\mathcal{N}}(t) \end{bmatrix}^T \tag{3.1}$$

$$\mathbf{u}(t) \triangleq {}^{\mathcal{B}}\mathbf{F}_{th}(t) \tag{3.2}$$

Therefore we can express the nonlinear dynamics in the following form

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} \dot{m}(t) & {}^{\mathcal{N}}\dot{\mathbf{r}}^T(t) & {}^{\mathcal{N}}\dot{\mathbf{v}}^T(t) & \dot{\boldsymbol{\sigma}}^T_{\mathcal{B}/\mathcal{N}}(t) & {}^{\mathcal{B}}\dot{\boldsymbol{\omega}}^T_{\mathcal{B}/\mathcal{N}}(t) \end{bmatrix}^T \tag{3.3}$$

In order to formulate our guidance problem with the free-final-time objective we must introduce time dilation. We can evaluate our dynamics on a normalized trajectory time variable $\tau \in [0, 1]$. No matter the resolution of the optimization, the terminal value will end at $\tau = 1$. We can then use a differentiation based on this variable to scale the time back and forth, leaving the unscaled final time as an optimization variable. Applying the chain rule of differentiation:

$$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \frac{d}{dt}\mathbf{x}(t) = \frac{d\tau}{dt}\frac{d}{d\tau}\mathbf{x}(t) = \frac{1}{\eta}\frac{d}{d\tau}\mathbf{x}(t) \tag{3.4}$$

We can now translate between the two by using the dilation coefficient η which we define as

$$\eta \triangleq \left(\frac{d\tau}{dt}\right)^{-1} \tag{3.5}$$

This η will become a variable in the convex subproblem that acts as the non-dimensionalized final time. It is a scaling factor that translates between real work differential time and the normalized version used for our algorithm. We can now write the nonlinear dynamics to take advantage of this normalized time:

$$\mathbf{x}'(\tau) \triangleq \frac{d}{d\tau}\mathbf{x}(\tau) = \eta\mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau)) = \mathbf{g}(\mathbf{x}(\tau), \mathbf{u}(\tau), \eta) \tag{3.6}$$

Taking a first-order Taylor series approximation of the nonlinear dynamics proposed in Problem one, we can write a linear time-varying framework equations 3.7 for the system to use in our algorithm. These dynamics will be instantiated at reference values $(\hat{\mathbf{x}}, \hat{\mathbf{u}}, \hat{\eta})$ at each time.

$$\mathbf{x}'(\tau) = \mathbf{g}(\hat{\mathbf{x}}(\tau), \hat{\mathbf{u}}(\tau), \hat{\eta}) + \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\bigg|_{\hat{x},\hat{u},\hat{\eta}} (\mathbf{x} - \hat{\mathbf{x}}) \; + \frac{\partial \mathbf{g}}{\partial \mathbf{u}}\bigg|_{\hat{x},\hat{u},\hat{\eta}} (\mathbf{u} - \hat{\mathbf{u}}) \; + \frac{\partial \mathbf{g}}{\partial \eta}\bigg|_{\hat{x},\hat{u},\hat{\eta}} (\eta - \hat{\eta}) \tag{3.7a}$$

$$= \hat{\eta}\mathbf{f}(\hat{\mathbf{x}}(\tau), \hat{\mathbf{u}}(\tau)) + \hat{\eta}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\bigg|_{\hat{x},\hat{u}} (\mathbf{x} - \hat{\mathbf{x}}) \; + \hat{\eta}\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\bigg|_{\hat{x},\hat{u}} (\mathbf{u} - \hat{\mathbf{u}}) \; + \mathbf{f}(\hat{\mathbf{x}}(\tau), \hat{\mathbf{u}}(\tau))(\eta - \hat{\eta}) \tag{3.7b}$$

$$= \eta\mathbf{f}(\hat{\mathbf{x}}(\tau), \hat{\mathbf{u}}(\tau)) + \hat{\eta}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\bigg|_{\hat{x},\hat{u}} (\mathbf{x} - \hat{\mathbf{x}}) \; + \hat{\eta}\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\bigg|_{\hat{x},\hat{u}} (\mathbf{u} - \hat{\mathbf{u}}) \tag{3.7c}$$

$$= \Sigma(\tau)\eta + A(\tau)(\mathbf{x} - \hat{\mathbf{x}}) + B(\tau)(\mathbf{u} - \hat{\mathbf{u}}) \tag{3.7d}$$

$$= \Sigma(\tau)\eta + A(\tau)\mathbf{x} + B(\tau)\mathbf{u} + \mathbf{z}(\tau) \tag{3.7e}$$

We can simplify this expression by breaking the Taylor expansion intro matrix subcomponents.

$$A(\tau) \triangleq \hat{\eta}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}},\hat{\mathbf{u}}} \tag{3.8}$$

$$B(\tau) \triangleq \hat{\eta}\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\bigg|_{\hat{\mathbf{x}},\hat{\mathbf{u}}} \tag{3.9}$$

$$\Sigma(\tau) \triangleq \mathbf{f}(\hat{\mathbf{x}}(\tau), \hat{\mathbf{u}}(\tau)) \tag{3.10}$$

$$\mathbf{z}(\tau) \triangleq -A(\tau)\hat{\mathbf{x}} - B(\tau)\hat{\mathbf{u}} \tag{3.11}$$

3.1.1 Convexifying the Minimum Thrust Constraint

With this done, we tackle the last source of non-convexity: the non-zero lower bound to our actuator thrust. Let us define an $\mathbb{R}^3 \to \mathbb{R}$ mapping function: $g(\mathbf{u}(\tau)) = F_{min} - \left\|\mathbf{u}(\tau)\right\|_2 \leq 0$. Taking the first order Taylor series linear approximation, we have the following convexified constraint formulation:

$$g(\mathbf{u}(\tau)) = F_{min} - \left\|\hat{\mathbf{u}}(\tau)\right\| - \frac{\hat{\mathbf{u}}(\tau)^T}{\left\|\hat{\mathbf{u}}(\tau)\right\|}(\mathbf{u}(\tau) - \hat{\mathbf{u}}(\tau)) \qquad \leq 0 \tag{3.12}$$

$$= F_{min} - \frac{\hat{\mathbf{u}}(\tau)^T}{\left\|\hat{\mathbf{u}}(\tau)\right\|}\mathbf{u}(\tau) \qquad \leq 0 \tag{3.13}$$

$$= F_{min} - \Xi(\tau)\mathbf{u}(\tau) \qquad \leq 0 \tag{3.14}$$

this leads us to the linear, convexified constraint $F_{min} \leq \Xi(\tau)\mathbf{u}(\tau)$. Recall the variable $\hat{\mathbf{u}}(\tau)$ is our reference input history.

3.2 Discretization Scheme

The final step to fitting the dynamics, state, and control constraints to an optimization form is to cast the problem as a finite dimensional discretization. We must choose this finite dimensional problem to occur over $K \in \mathbb{Z}$ evenly separated points with respect to the normalized trajectory time τ. We can define the index set $\mathcal{K} \in \mathbb{Z}^K$ and $\mathcal{K}_- \in \mathbb{Z}^{K-1}$ for the state and input histories as the following:

$$\mathcal{K} \triangleq \{0, 1, \cdots, K-1\}$$

$$\mathcal{K}_- \triangleq \{0, 1, \cdots, K-2\}$$

Given that the trajectory time is normalized on the interval $\tau \in [0, 1]$, we can define the discrete time step at point k as such

$$\tau_k \triangleq \frac{k}{K-1}, \quad \forall k \in \mathcal{K} \tag{3.15}$$

For implementation sake, a first-order-hold linear scaling on the applied controls for each time step. Over the interval $\tau \in [\tau_k, \tau_{k+1}]$, we express $\mathbf{u}(\tau)$ in terms of the $\mathbf{u}_k$ and $\mathbf{u}_{k+1}$:

$$\mathbf{u}(\tau) \triangleq \alpha_k(\tau)\mathbf{u}_k + \beta(\tau)\mathbf{u}_{k+1} \tag{3.16}$$

The input is spread linearly, on a first order relationship, from the index position k to the next known control value at $k+1$ in the control history. This also allows us to consider controller interpolation scheme a priori if used in a feed forward regime. The successive convexification algorithm was tested with a number of discretization methods on state and input, where first order hold (FOH) and Legendre-Gauss-Radau (LGR) collocation methods were found to be the most amenable [13]. These two provided the fastest computation times with good performance. The constants we have defined follow the following form:

$$d\tau = \frac{1}{K-1} \tag{3.17}$$

$$\alpha_k(\tau) = \frac{d\tau - \tau}{d\tau} \tag{3.18}$$

$$\beta_k(\tau) = \frac{\tau}{d\tau} \tag{3.19}$$

We use the state transition matrix (STM) of the dynamics $\Phi(\tau_{k+1}, \tau_k)$ to translate the process from a state at time k to future state at $k+1$. This matrix assumes no input is being imparted, but a convolution can be used to describe the time varying input. The STM follows these dynamics:

$$\frac{d}{d\tau}\Phi(\tau, \tau_k) = A(\tau)\Phi(\tau, \tau_k), \quad \forall k \in \mathcal{K} \tag{3.20}$$

Additionally, the STM has the semigroup, inverse, and identity properties which become useful in the derivation:

$$\Phi(t, s) = \Phi(t, \gamma)\Phi(\gamma, s) \tag{3.21}$$

$$\Phi(t, s)^{-1} = \Phi(s, t) \tag{3.22}$$

$$\Phi(s, s) = \mathbb{I}_{n\times n} \tag{3.23}$$

for arbitrary timing parameters t, γ, s. We can take advantage of this property during the discretization steps to minimize some computation. A general homogeneous solution for a system defined by $\dot{\mathbf{x}}$ using the STM is the following:

$$\text{given that} \quad \mathbf{x}(\tau) = \Phi(\tau, \tau_k)\mathbf{x}(\tau_k) \tag{3.24}$$

$$\frac{d}{d\tau}\mathbf{x}(\tau) = \dot{\Phi}(\tau, \tau_k)\mathbf{x}(\tau_k) = A(\tau)\mathbf{x}(\tau) \tag{3.25}$$

$$= A(\tau)\Phi(\tau, \tau_k)\mathbf{x}(\tau_k) \tag{3.26}$$

Integration leads to the following fact

$$\mathbf{x}(\xi) = \mathbf{x}(\tau_k) + \int_{\tau_k}^{\xi} A(\xi)\Phi(\xi, \tau_k)\mathbf{x}(\tau_k)d\xi \tag{3.27}$$

$$= \left(\mathbb{I}_{n\times n} + \int_{\tau_k}^{\xi} A(\xi)\Phi(\xi, \tau_k)d\xi\right)\mathbf{x}(\tau_k) \tag{3.28}$$

Therefore the arbitrary state transition mapping from time τ_k to ξ can be represented as the following:

$$\Phi(\xi, \tau_k) = \mathbb{I}_{n\times n} + \int_{\tau_k}^{\xi} A(\xi)\Phi(\tau, \tau_k)d\xi, \quad \forall \xi \in [\tau_k, \tau_{k+1}] \tag{3.29}$$

Recall our continuous LTV system dynamics expression $\mathbf{x}'(\tau) = A(\tau)\mathbf{x}+B(\tau)\mathbf{u}+\mathbf{z}(\tau)+\Sigma(\tau)\eta$. Employing our control FOH, we want to form this as the following discrete time system:

$$\mathbf{x}_{k+1} = F_k\mathbf{x_k} + G_k^-\mathbf{u_k} + G_k^+\mathbf{u_{k+1}} + \bar{\mathbf{z}}_k + \bar{\Sigma}_k\eta \tag{3.30}$$

Converting the continuous time dynamics to discrete time dynamics, we perform a series of convolution integrals to define the impact of the transformations over each discrete time step. The continuous linear time invariant discretization form is usually written as $G = \int_0^{dt} e^{A\tau}d\tau B$, but given

the LTV dynamics and our FOH control assumption, we reformulate as:

$$F_k \triangleq \Phi(\tau_{k+1}, \tau_k) \tag{3.31a}$$

$$G_k^- \triangleq \int_{\tau_k}^{\tau_{k+1}} \Phi(\tau_{k+1}, \xi)\alpha_k(\xi)B(\xi)d\xi \tag{3.31b}$$

$$\triangleq F_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\xi, \tau_k)\alpha_k(\xi)B(\xi)d\xi \tag{3.31c}$$

$$G_k^+ \triangleq F_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\xi, \tau_k)\beta_k(\xi)B(\xi)d\xi \tag{3.31d}$$

$$\bar{\Sigma}_k \triangleq F_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\xi, \tau_k)\Sigma(\xi)d\xi \tag{3.31e}$$

$$\bar{\mathbf{z}}_k \triangleq F_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\xi, \tau_k)\mathbf{z}(\xi)d\xi \tag{3.31f}$$

It should be noted that the number of temporal nodes K, where $\mathcal{K} \in \mathbb{Z}^K$, chosen for this problem to be evaluated at does not affect the accuracy of the solution, only the optimality. An optimization solution where $K = 10$ will still accurately land the vehicle within all the constraints defined, but may do so in more time and with larger cost than the same problem solved at $K = 25$. However, the former is computed faster and generally produces a cost quite close to the higher temporal-density solution depending on the problem statement.

3.3 Successive Form, Trust Regions and Relaxations

In order to solve a non-convex problem, we iteratively solve a sequence of related convex subproblems. However, before we can reach a concluding framework, we must consider trust regions and dynamic relaxations. In order to make sure that this successive framework stays bounded and feasible through this convergence process, we must bound the divergence of state and inputs from one iteration to another. Unbounded problems can arise from constraints that admit an unbounded cost. To mitigate this issue, we augment the cost function with soft trust regions about the previous

iterate's information. Let us define these deviations at iteration i as such:

$$\delta \mathbf{x}_k^i \triangleq \mathbf{x}_k^i - \mathbf{x}_k^{i-1} \tag{3.32}$$

$$\delta \mathbf{u}_k^i \triangleq \mathbf{u}_k^i - \mathbf{u}_k^{i-1}, \quad \forall k \in \mathcal{K} \tag{3.33}$$

$$\delta \eta^i \triangleq \eta^i - \eta^{i-1} \tag{3.34}$$

We can then fabricate the following constraints with $\bar{\mathbf{\Delta}}^i \in \mathbb{R}^K$ and $\Delta_\eta^i \in \mathbb{R}$

$$\delta \mathbf{x}_k^i \cdot \delta \mathbf{x}_k^i + \delta \mathbf{u}_k^i \cdot \delta \mathbf{u}_k^i \leq \bar{\Delta}_k^i \tag{3.35}$$

$$\delta \eta^i \cdot \delta \eta^i \leq \Delta_\eta^i \tag{3.36}$$

We can now append $w_\Delta^i \left\| \bar{\Delta}^i \right\| + w_{\Delta_\eta} ||\Delta_\eta^i||$ to the cost function to minimize input, state, and final time deviations and keeping their deviation bounded via constraint, where w_Δ^i and w_{Δ_η} are weighting scalars depending on the preferences of the scenario. While the norm magnitudes are bounded, we consider these soft constraints. Given that the trust regions are centered about previous points $(\mathbf{x}_k^{i-1}, \mathbf{u}_k^{i-1}, \eta^{i-1})$, we must evaluate the Jacobian about the nonlinear trajectory beginning at $\mathbf{x}_k^{i-1}$. We can then use the FOH input vector $\mathbf{u}(\tau)$. Doing this $\forall k \in \mathcal{K}$ defines the aforementioned linearization path $(\hat{\mathbf{x}}, \hat{\mathbf{u}}, \hat{\eta})$.

We now add a dynamic relaxation variable to discount artificial infeasibility. This is encountered during the convergence process when the linearization becomes infeasible. For example, if the dynamics are linearized about unrealistic conditions, the problem becomes dynamically inconsistent and will not produce solution. We modify the control such that we have guaranteed subproblem solutions that have non-empty feasible sets. This is encountered during the first couple iterations of a successive convexification due to poor initial trajectory or time-of-flight estimation. To get rid of this issue, we employ dynamic relaxation, where a slack variable is added to the dynamics in order to "make room" for the iteration to proceed. You can also think of this as a virtual control or dynamic padding. However, you can guess that this will inevitably be something we want to minimize in the cost function in order to make sure that our final trajectories are as dynamically

consistent as possible. Therefore, we must now write our dynamics as follows:

$$\mathbf{x}_{k+1}^i = F_k^i \mathbf{x}_k^i + G_k^{-,i} \mathbf{u}_k^i + G_k^{+,i} \mathbf{u}_{k+1}^i + \bar{\mathbf{z}}_k^i + \bar{\Sigma}_k^i \eta^i + \boldsymbol{\nu}_k^i \tag{3.37}$$

The right super script ^{i} indicates the iterate of the algorithm, where the subscript $_k$ of course means the ordered parameter entrance in our array shaped by $\mathcal{K}$.

Because we are going to use this in our cost function as a norm, we shall concatenate the time history of the slack variable as

$$\boldsymbol{\nu}^i = \begin{bmatrix} \boldsymbol{\nu}_0^{i\,T} & \cdots & \boldsymbol{\nu}_{K-2}^{i\,T} \end{bmatrix}^T \tag{3.38}$$

And of course we augment the cost function again with $w_v \left\| \boldsymbol{\nu}^i \right\|_1$. As the iteration continues, this value is minimized as the solution becomes more dynamically feasible. Therefore, the magnitude of this norm is indicative of a final solution in the successive iteration process.

3.4 Convex Sub-Problem

Now that we have linearized and convexified the proposed problem, we can put these components together as the full free-final-time problem shown in 3.4.1 where we use the following objective function:

$$\min_{\eta^i, \mathbf{u}_k^i} \quad \eta^i + w_\Delta^i ||\bar{\boldsymbol{\Delta}}^i||_2 + w_{\Delta_\eta} ||\Delta_\eta^i||_1 + w_v ||\bar{\boldsymbol{\nu}}^i||_1 \tag{3.39}$$

3.4.1 Algorithm

The goal is for the successive convexification algorithm 2 to continue iterating until Δ_{tol} and ν_{tol} are met. We define these by the magnitude of each vector. Their magnitudes are checked at the end of each iteration of the routine. Additionally, to ensure boundedness on the trust regions and to prevent an admitted unbounded cost, we turn the cost weighting of the trust region up after each iteration.

Algorithm 2 Successive Convexification

procedure SCvx($x_{ref}, u_{ref}, \eta_{ref}$)
    Generate initial reference trajectory by linearly spanning states and inputs from initial conditions to terminal constraints.
    while $||\Delta|| \geq \Delta_{tol}$ && $||\nu|| \geq \nu_{tol}$ && $i \leq i_{max}$ **do**
        Compute $\bar{A}_k^{i-1}, \bar{B}_k^{i-1}, \bar{C}_k^{i-1}, \bar{\Sigma}_k^{i-1}, \bar{z}_k^{i-1}$ from $\mathbf{x}_k^{i-1}, \mathbf{u}_k^{i-1}, \eta^{i-1}$
        Solve Problem 2 using $\mathbf{x}_k^{i-1}, \mathbf{u}_k^{i-1}, \eta^{i-1}, \bar{A}_k^{i-1}, \bar{B}_k^{i-1}, \bar{C}_k^{i-1}, \bar{\Sigma}_k^{i-1}, \bar{z}_k^{i-1}$
        Store the newly found $\mathbf{x}_k^i, \mathbf{u}_k^i, \eta^i$
        $w_\Delta^{i+1} = 1.7 * w_\Delta^i$
        $i++$;
    end while
    return $\mathbf{x}, \mathbf{u}$
end procedure

Problem 2: Discretized Convex Fixed-Final-Time Sub-Problem (i^{th} iteration)

Cost Function:

$$\min_{\eta^i, \mathbf{u}_k^i} \quad \eta^i + w_\Delta^i ||\bar{\mathbf{\Delta}}^i||_2 + w_{\Delta_\eta} ||\Delta_\eta^i||_1 + w_v ||\bar{\boldsymbol{\nu}}^i||_1$$

Boundary Conditions:

$$m_0 = m_0, \qquad {}^{\mathcal{N}}\mathbf{r}_0 = \mathbf{r}_0, \qquad {}^{\mathcal{N}}\mathbf{v}_0 = \mathbf{v}_0, \qquad \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}_0} = \boldsymbol{\sigma}_0, \qquad {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}_0} = \boldsymbol{\omega}_0$$

$${}^{\mathcal{N}}\mathbf{r}_{K-1} = \mathbf{0}, \qquad {}^{\mathcal{N}}\mathbf{v}_{K-1} = \mathbf{0}, \qquad \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}_{K-1}} = \mathbf{0}, \qquad {}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}_{K-1}} = \mathbf{0}$$

Dynamics:

$$\mathbf{x}_{k+1}^i = F_k^i \mathbf{x}_k^i + G_k^{-,i} \mathbf{u}_k^i + G_k^{+,i} \mathbf{u}_{k+1}^i + \bar{\mathbf{z}}_k^i + \bar{\Sigma}_k^i \eta^i + \boldsymbol{\nu}_k^i$$

State and Control Constraints:

$$m_{dry} - m_k \leq 0$$

$$||[\hat{\boldsymbol{n}}_2 \quad \hat{\boldsymbol{n}}_3]^T \mathbf{r}_k||_2 \tan\gamma_{gs} - \hat{\boldsymbol{n}}_1^T \mathbf{r}_k \leq 0$$

$$\left\| \boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}_k} \right\|_2 \leq \tan\left(\frac{\psi_{max}}{4}\right)$$

$$||\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}_k}||_2 \leq \omega_{max}$$

Control Constraints:

$$F_{min} \leq \Xi_k^i \mathbf{u}_k^i$$

$$||\mathbf{u}_k^i||_2 \leq F_{max}$$

$$\cos(\delta_{max}) ||\mathbf{u}_k^i||_2 \leq \hat{\boldsymbol{b}}_1^T \mathbf{u}_k^i$$

Trust Regions:

$$||\delta {\mathbf{x}_k^i}^T \delta \mathbf{x}_k^i + \delta {\mathbf{u}_k^i}^T \delta \mathbf{u}_k^i||_2 \leq \bar{\Delta}_k^i$$

$$||\delta \eta^i||_2 \leq \Delta_\eta^i$$

Chapter 4

Guidance Results

4.1 Discussion

The successive convexification routine is a very valuable tool in that it can quickly find dynamically feasible trajectories for a wide set of vehicle constraints and parameters. It could be used as an offline tool, or a receding horizon feed-forward guidance strategy where the temporal resolution increases as the terminal state constraint gets closer. It should be stated that trajectory computation employs non-dimensionalized factors for numerical stability. Tables like 4.1 have been included for each of the problem statements given where all real-world values can be calculated using factors U_L, U_T and U_M.

4.2 Planar Problem Solutions

Let us first study a case where the vehicle has a velocity in-plane with the terminal condition and can therefore neglect on dimension of motion. This is the nominal scenario where a vehicle is coming in for landing and has pre-conditioned it's state to do so. This is apparent in the landing of first stages, where the out-of-plane motion is canceled and an in-plane maneuver is performed to adjust to meet the landing pad. In figure 4.1, we see the SCvx guidance solution to a set of initial conditions in-plane with the landing point; the vehicle has a position of 600 meters North and 500 meters East with Eastward velocity of -120m/s. The incoming velocity is higher than nominal, and so the optimal control solution is to maximize the divert capability such that the lateral velocity is arrested while the attitude is up-right at the final time.

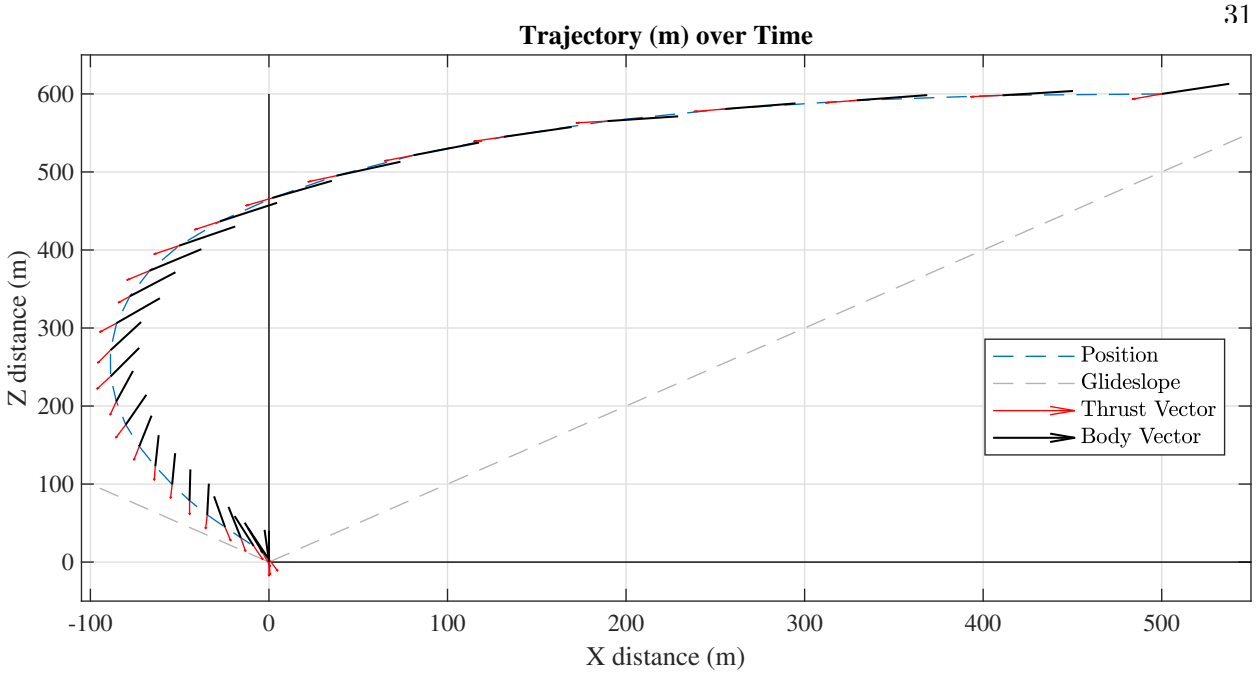

Figure 4.1: Planar Guidance Problem: Vehicle Descends In-line with Target

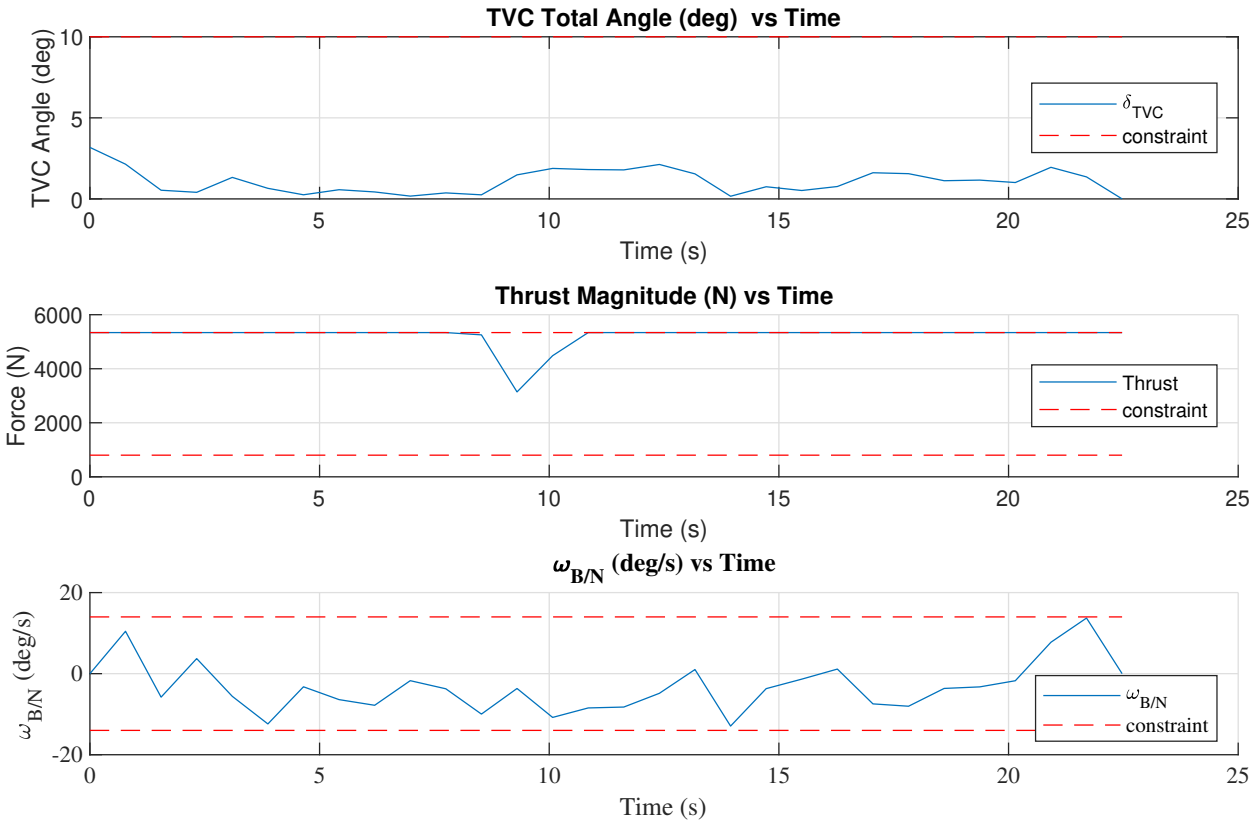

Figure 4.2: Planar Guidance Problem: Control and Angular Rate During Landing

The control solution in figure 4.2 shows how maximum thrust is used nearly the entire duration where lateral thrust is used for attitude control while simultaneously lowering altitude control. The commanded TVC deflection angles stay within our proposed constraints. Our thrust magnitude and angular rate constraints are also met throughout the duration of the flight. For a fast pitch-over maneuvers like this, it is hard to find a trajectory where the vehicle angular rate maintains within reasonable boundaries; this scenario is restricted to below 14 deg/s. We have shown all initial conditions and constraints in 4.1.

Table 4.1: Parameters Used For Planar Problem

Param	Units	Value	Param	Units	Value
${}^{\mathcal{N}}\mathbf{g}$	U_L/U_T^2	$-0.108\hat{\mathbf{n}}_1$	γ_{gs}	deg	45
$\alpha_{\dot{m}}$	-	0.0738	δ_{max}	deg	10
m_{wet}	U_M	1	F_{max}	U_F	0.164
m_{dry}	U_M	0.277	F_{min}	U_F	0.024
${}^{\mathcal{N}}r_0$	U_L	$(0.76, 0.64, 0)$	ψ_{max}	deg	85
${}^{\mathcal{N}}v_0$	U_L/U_T	$(0, -0.48, 0)$	ω_{max}	deg/U_T	43.84
$\sigma_{\mathcal{B}/\mathcal{N}_0}$	-	$(0, 0, 0.32)$	U_M	kg	408.233
$\omega_{\mathcal{B}/\mathcal{N}_0}$	rad/U_T	$(0, 0, 0)$	U_L	m	781.02
			U_T	s	3.132

The vehicle following the planar path in figure 4.1 reached it's terminal state after 22.4 seconds and with a final mass of 373.62kg with a starting mass of 408.23kg only using 11.7% of it's fuel. Additionally, we see that the total alignment MRP constraint is met throughout the flight 4.3.

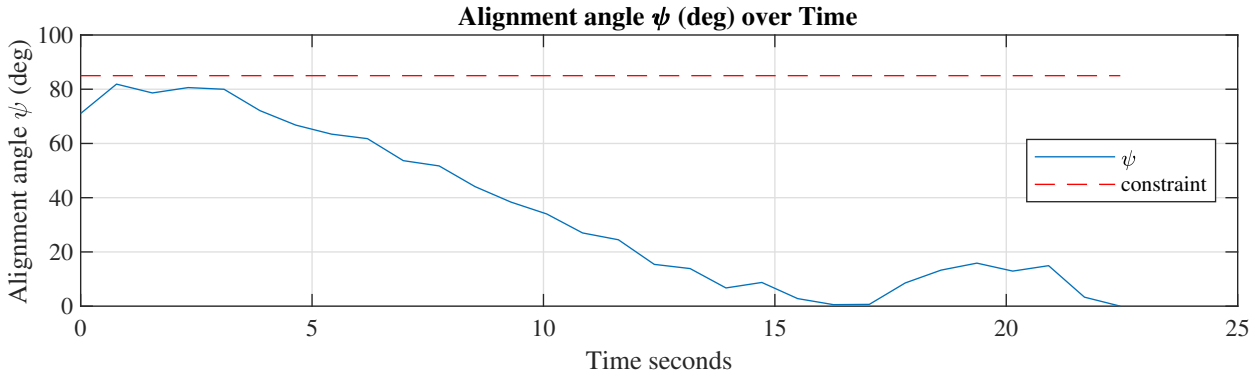

Figure 4.3: Planar Guidance Problem: Total Alignment Constraint Met

4.3 Non-Planar Problem Solutions

We now look at a non-planar problem where the initial condition of the vehicle is off-nominal and must make an out-of-plane maneuver to land on the target. The parameters and initial conditions are found in table 4.2. Figure 4.5 shows a vehicle coming towards the landing pad with an offset lateral position of 100m and with a velocity not pointed directly in-line with the target. All constraints are met and the vehicle uses maximum thrust nearly the entire duration where cosine throttling and attitude control becomes required are employed. The simple MRP constraint is also met.

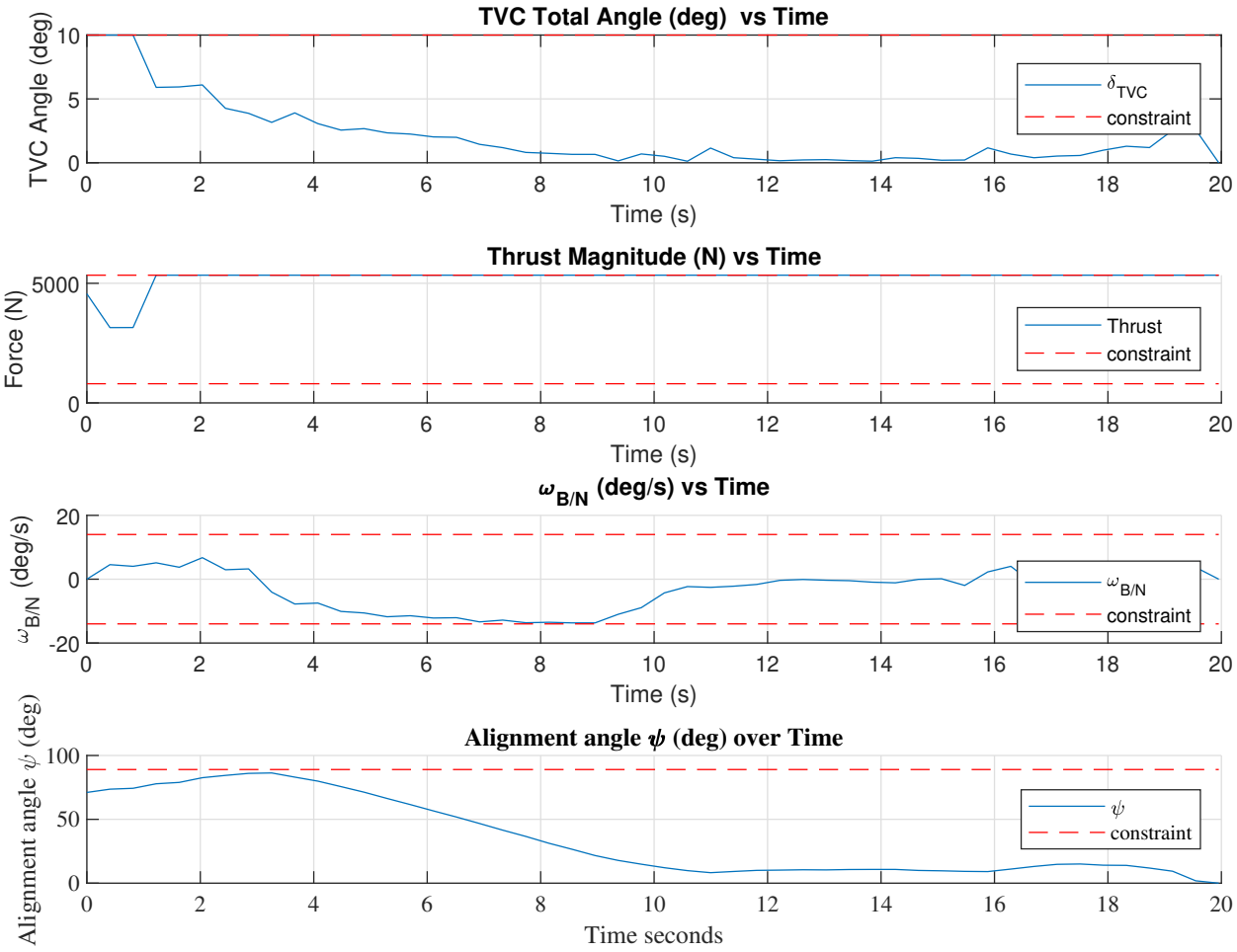

Figure 4.4: Non-Planar Guidance Problem: Controls and Angular rate of Landing Vehicle

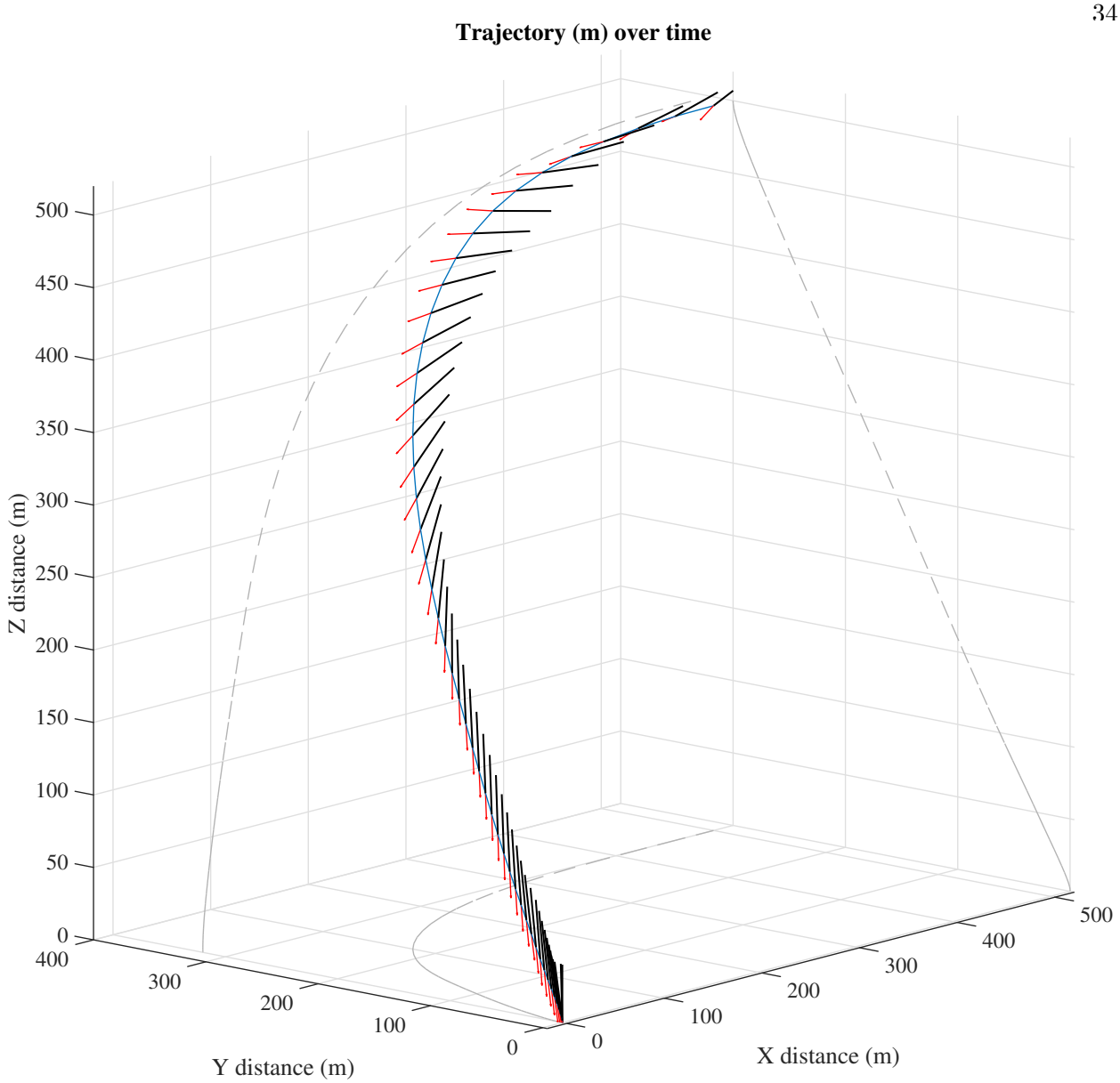

Figure 4.5: Non-Planar Guidance Problem: Vehicle Approaches Offset from Target

The vehicle following the non-planar path in figure 4.1 reached it's terminal state after 19.95 seconds and with a final mass of 376.784kg with a starting mass of 408.23kg only using 10.6% of it's fuel. Additionally, from figure 4.6 we can see the terminal position and velocity constraints are met, and the vehicle reaches the landing pad with a safe terminal velocity

Table 4.2: Parameters Used for Non-Planar Problem

Param	Units	Value	Param	Units	Value
${}^{\mathcal{N}}\mathbf{g}$	U_L/U_T^2	$\hat{\mathbf{n}}_1$	γ_{gs}	deg	5
$\alpha_{\dot{m}}$	-	0.0738	δ_{max}	deg	10
m_{wet}	U_M	1	F_{max}	U_F	0.166
m_{dry}	U_M	0.277	F_{min}	U_F	0.025
${}^{\mathcal{N}}r_0$	U_L	$(0.65, 0.65, 0.39)$	ψ_{max}	deg	85
${}^{\mathcal{N}}v_0$	U_L/U_T	$(0, -0.4, 0)$	ω_{max}	deg/U_T	43.84
$\sigma_{\mathcal{B}/\mathcal{N}_0}$	-	$(0, 0, 0.32)$	U_M	kg	408.233
$\omega_{\mathcal{B}/\mathcal{N}_0}$	rad/U_T	$(0, 0, 0)$	U_L	m	768.114
			U_T	s	3.132

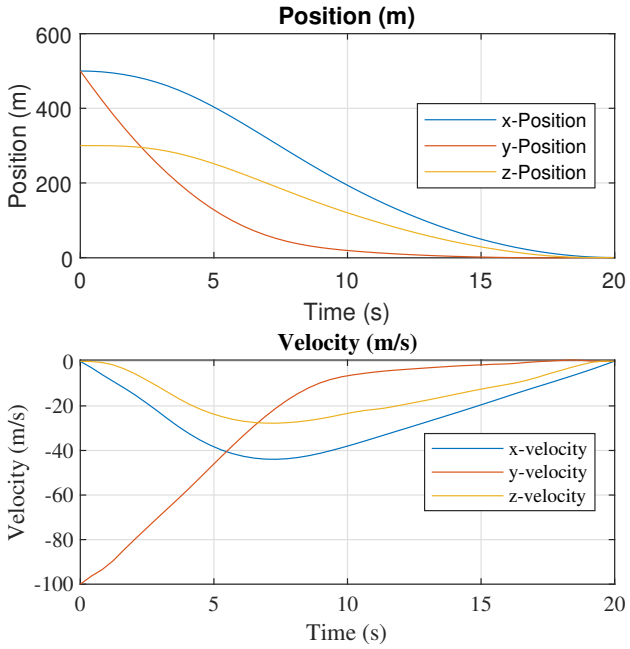

Figure 4.6: Non-Planar Guidance Problem: Position and Velocity Histories

Chapter 5

Analysis

5.1 Computation Preface

For the following analyses, the algorithm was run on a 2018 Dell XPS 15 laptop with an Intel quad-core i7-7700HQ CPU clocked at 2.8GHz with 16GB of ram. The implementation was in Python utilizing CVXpy configured to use the ECOS solver by EmboTech [9]. ECOS is an Mehrotra predictor corrector interior-point method (IPM) solver for second-order cone programs (SOCP) designed specifically for embedded applications.

It should be understood that my Python implementation of this is not optimized to run quickly on embedded platforms and should not be held as a benchmark. These algorithms are amenable for online implementation and should be written in C/C++ ideally with customized solvers that can take advantage of the problem structure. One can reach excellent cycle speeds for these algorithms as evidence by [21]. The performance and computational information presented in this section was performed with the baseline problem outlined in Table 4.1.

Additionally, it should be understood that all runs listed were initialized with a dynamically infeasible and linearly-spanned trajectory as discussed earlier. These are suboptimal with respect to performance and will required larger numbers of iterations compared to a good guess. One could easily improve this with an LQR run or another dynamically-motivated trajectory estimation algorithm. This would significantly decrease computation time for the SCvx algorithm.

5.2 Temporal Node Count Variation

Although the listed runtimes are larger than nominal for an implementation, for the aforementioned reasons, it important to study it's numerical behaviour. Table 5.1 shows the planar problem shown in 4.1 being solved numerous times for the temporal node count quantity K for cases of $K = [10,\ 15,\ 20,\ 30,\ 40,\ 50]$. For trajectories with 10 state and control temporal nodes, the computer took an average of 1.771 seconds to compute. Figure 5.1 shows how the final time solution changes on the order of 100ms from a K=10 solution to a K=50 solution. Additionally, the final time difference between K=20 and K=50 is less than 100ms and shows that the latter is not necessarily better for the extra computation cost. Similarly in Figure 5.2, where the key metric of remaining final mass is displayed with respect to K, we see that increasing the temporal resolution does not necessarily save much more fuel. The difference between a quick calculation and a more intensive one is on the order of hundreds of grams, effectively inconsequential.

Table 5.1: Computation Time (s)

K	Min	Max	Mean	Stdev
10	0.875	1.771	1.3023	0.29
15	1.331	2.713	2.2504	0.365
20	2.289	2.986	2.618	0.230
30	3.38	4.708	4.1419	0.363
40	4.587	5.499	4.9056	0.257
50	6.814	8.637	7.7515	0.688

Figure 5.4 shows the results of Table 5.1 in a graphical form for multiple runs. For the same landing problem, we see the convergence rate in Figure 5.3 of each of the resolution problems is effectively the same. For our K=10 case, a converged solution was extracted after the 13th successive iteration.

For an online system, it makes sense to run a lower resolution optimization with a receding horizon approach. The algorithm is run as fast as the processor allows, and the input and state history is calculated iteratively. Each of the runs would use the previous solution as an initial trajectory, cutting down computation costs. This allows the effective resolution to increase as the

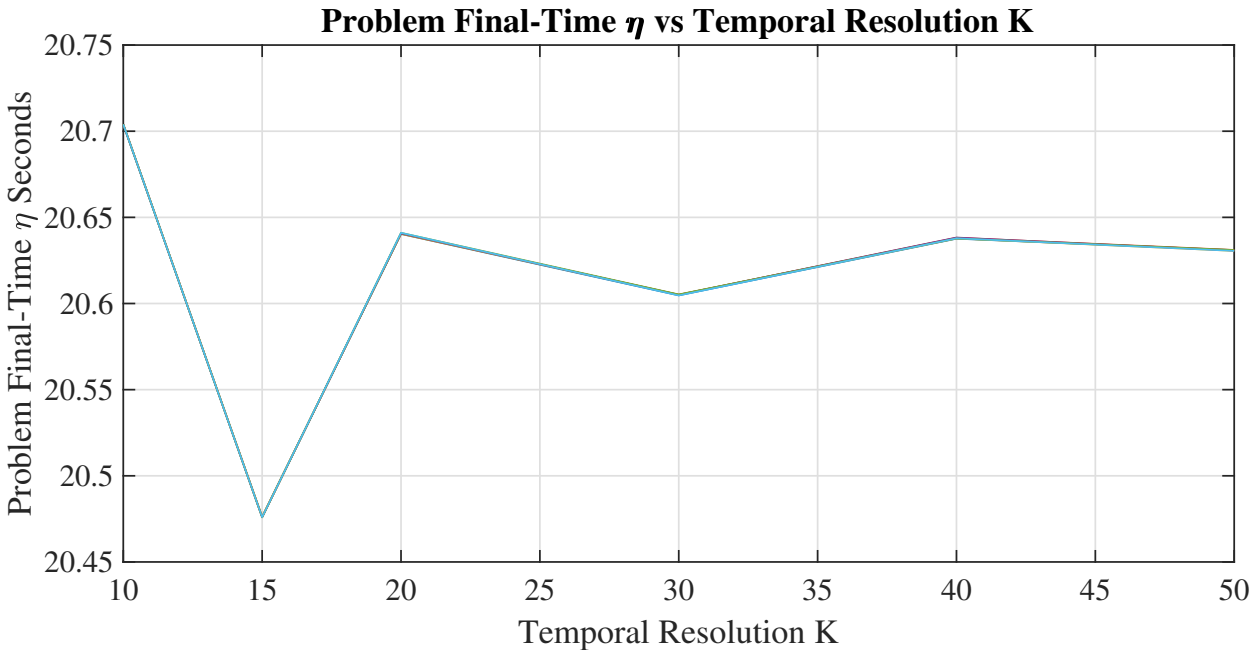

Figure 5.1: Planar Guidance Problem: Final-Time η versus Temporal Node Count K

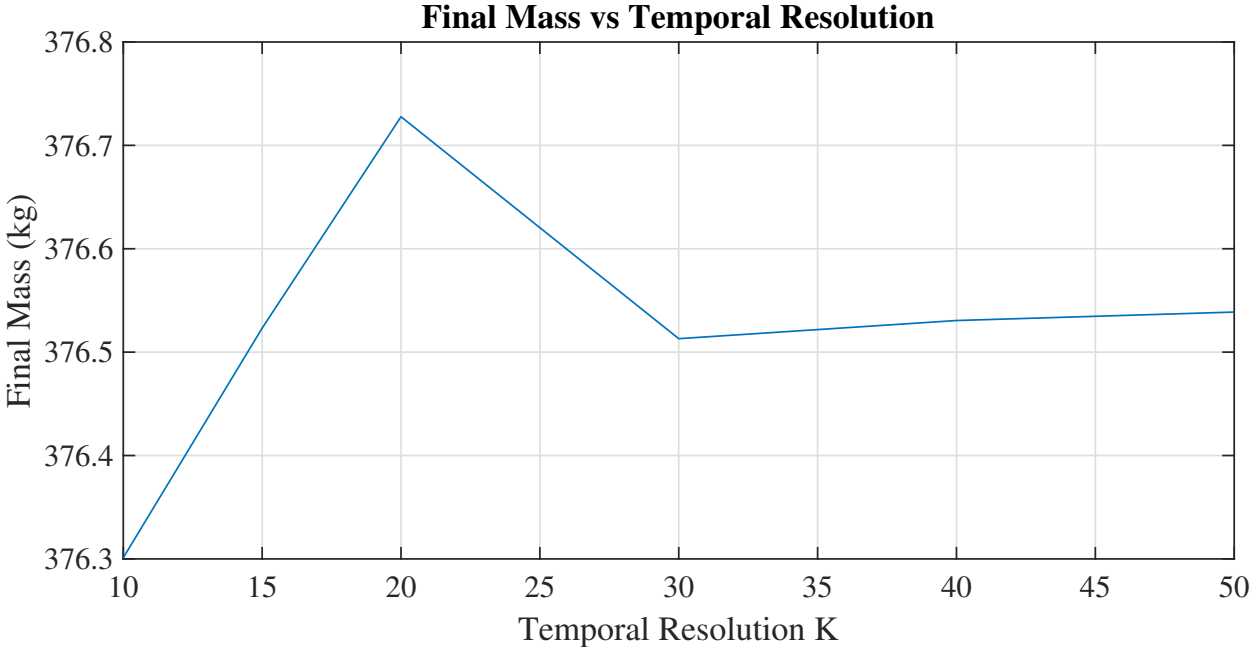

Figure 5.2: Planar Guidance Problem: Terminal Vehicle Mass versus K Node Count

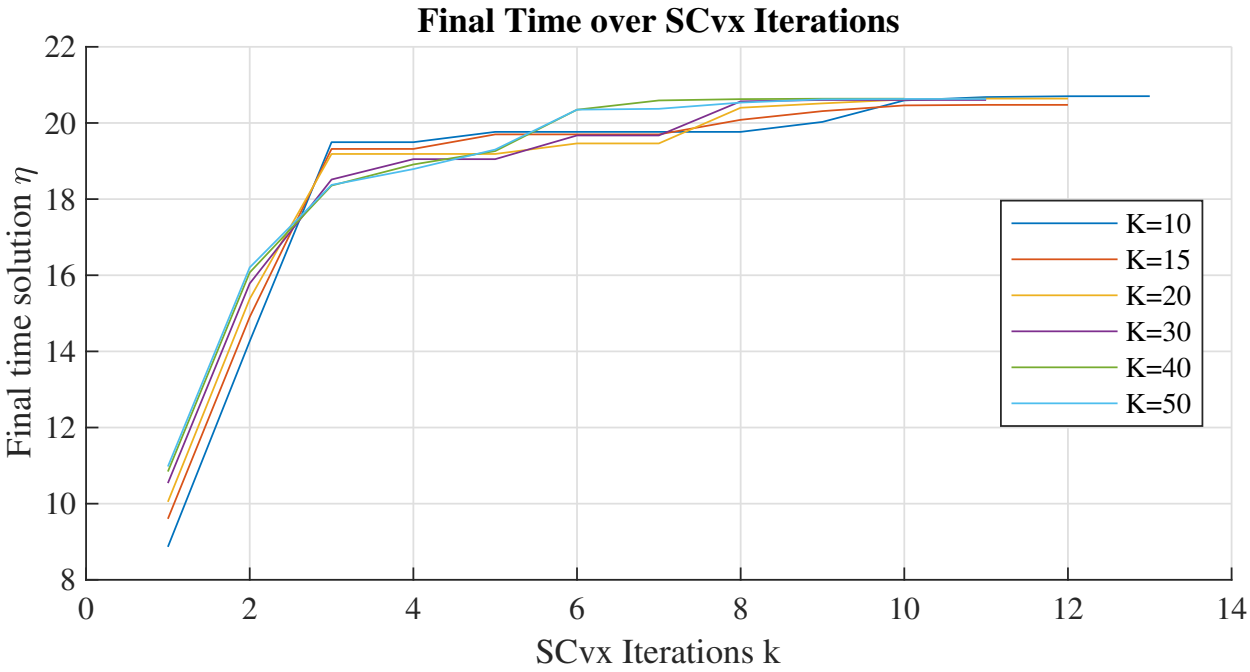

Figure 5.3: Planar Guidance Problem: Final-Time η Convergence for many K Node Count Runs

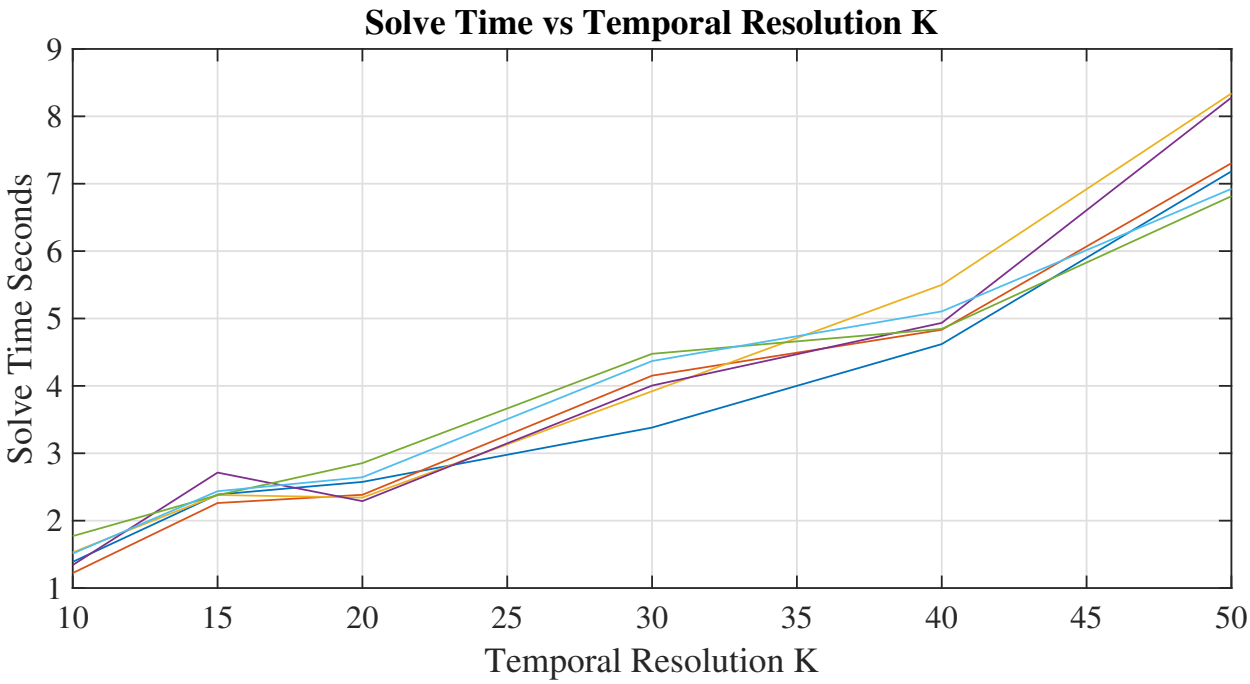

Figure 5.4: Planar Guidance Problem: Computation Time versus Temporal Node Count K

vehicle comes closer to the terminal state constraints.

5.3 A Parameter Variation Study

5.4 Runtime: MRPs vs Quaternions

5.5 Corner Conditions

5.6 Glideslope and Subterminal Constraints

Chapter 6

Conclusions

6.1 Conclusions and Future Work

This type of algorithm is highly versatile and amenable to real-time implementation for agile landing vehicles. This could be used as a trajectory generator, sequential guidance routine, guidance and feed-forward routine, or simply as an offline validation and mission design tool. It's ability to quickly generate dynamically feasible trajectories is impressive.

In the future, it may be interesting to apply a TVC bandwidth constraint to the system as well as state triggered constraints (STCs) for mission specific actions like initial ignition timing as shown in [24]. The hypersonic reentry problem has hybrid-discrete dynamic switching events where the vehicle changes properties. For example, during entry, a capsule may shed it's protective ablative shield and dawn a parachute for a deceleration phase. It may be interesting to explore trajectory optimization and discrete event timing using successive convexification and state triggered constraints. The MRP switching mechanism can easily be encoded with STCs and may yield interesting model predictive attitude slewing algorithms. Tackling the ascent objective with a number of active aerodynamic disturbances seems like a worthwhile objective. The problem can be split up into a couple smaller in-plane and out-of-plane problems, making the full solutions easier and perhaps more computationally tractable.

It would be valuable to write this routine in C/C++ for small embedded single board computers (SBCs). These SBCs could be dedicated path planning computers for small landing testbeds, robotic platforms, and other GN&C validation hardware.

Although the majority of this thesis is implementation and adaptation, I hope it serves as an example of the versatility and usefulness of the SCvx routine and it's capabilities. It is evident that online optimization routines such as this will be valuable and shape the future of autonomous robotics, spacecraft, and real-time decision making.

Bibliography

[1] B Acikmese, J Casoliva, JM Carson, and L Blackmore. G-fold: A real-time implementable fuel optimal large divert guidance algorithm for planetary pinpoint landing. In Concepts and Approaches for Mars Exploration, volume 1679, 2012.

[2] Behcet Acikmese, M Aung, Jordi Casoliva, Swati Mohan, Andrew Johnson, Daniel Scharf, David Masten, Joel Scotkin, Aron Wolf, and Martin W Regehr. Flight testing of trajectories computed by g-fold: Fuel optimal large divert guidance algorithm for planetary landing. In AAS/AIAA spaceflight mechanics meeting, 2013.

[3] Behçet Açıkmeşe and Lars Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. Automatica, 47(2):341–347, 2011.

[4] Behçet Açikmeşe and Scott R Ploen. Convex programming approach to powered descent guidance for mars landing. Journal of Guidance, Control, and Dynamics, 30(5):1353–1366, 2007.

[5] Michael S Andrle and John L Crassidis. Geometric integration of quaternions. Journal of Guidance, Control, and Dynamics, 36(6):1762–1767, 2013.

[6] Lars Blackmore. Autonomous precision landing of space rockets. Winter Bridge on Frontiers of Engineerin, 4(46):15–29, 2016.

[7] Lars Blackmore, Behçet Açikmeşe, and Daniel P Scharf. Minimum-landing-error powered-descent guidance for mars landing using convex optimization. Journal of guidance, control, and dynamics, 33(4):1161–1171, 2010.

[8] Stephen Boyd and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[9] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In 2013 European Control Conference (ECC), pages 3071–3076. IEEE, 2013.

[10] Christopher D'Souza and Christopher D'Souza. An optimal guidance law for planetary landing. In Guidance, Navigation, and Control Conference, page 3709, 1997.

[11] Harry Jones. The recent large reduction in space launch cost. 48th International Conference on Environmental Systems, 2018.

[12] Allan R Klumpp. Apollo lunar descent guidance. Automatica, 10(2):133–146, 1974.

[13] Danylo Malyuta, Taylor Reynolds P., Michael Szmuk, Mehran Mesbahi, and Behçet Açikmeşe. Discretization performance and accuracy analysis for the powered descent guidance problem. 2019.

[14] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In 2016 IEEE 55th Conference on Decision and Control (CDC), pages 3636–3641. IEEE, 2016.

[15] Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behçet Açikmeşe. Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems. arXiv preprint arXiv:1804.06539, 2018.

[16] F Landis Markley and John L Crassidis. Fundamentals of Spacecraft Attitude Determination and Control, volume 33. Springer, 2014.

[17] RL McHenry, AD Long, BF Cockrell, JR Thibodeau III, and TJ Brand. Space shuttle ascent guidance, navigation, and control. Journal of the Astronautical Sciences, 27:1–38, 1979.

[18] Yurii Nesterov and Arkadii Nemirovskii. Interior-point polynomial algorithms in convex programming, volume 13. Siam, 1994.

[19] Daniel P Scharf, Martin W Regehr, Geoffery M Vaughan, Joel Benito, Homayoon Ansari, MiMi Aung, Andrew Johnson, Jordi Casoliva, Swati Mohan, Daniel Dueri, et al. Adapt demonstrations of onboard large-divert guidance with a vtvl rocket. In 2014 IEEE Aerospace Conference, pages 1–18. IEEE, 2014.

[20] Hanspeter Schaub and John L. Junkins. Analytical Mechanics of Space Systems. American Institute of Aeronautics and Astronautics, Inc, Reston, Virginia, fourth edition, 2018.

[21] Michael Szmuk. Successive Convexification & High Performance Feedback Control for Agile Flight. PhD thesis, 2019.

[22] Michael Szmuk and Behçet Açikmeşe. Successive convexification for 6-dof mars rocket powered landing with free-final-time. In 2018 AIAA Guidance, Navigation, and Control Conference, page 0617, 2018.

[23] Michael Szmuk, Utku Eren, and Behçet Açikmeşe. Successive convexification for mars 6-dof powered descent landing guidance. In AIAA Guidance, Navigation, and Control Conference, page 1500, 2017.

[24] Michael Szmuk, Taylor P Reynolds, and Behçet Açikmeşe. Successive convexification for real-time 6-dof powered descent guidance with state-triggered constraints. arXiv preprint arXiv:1811.10803, 2018.

[25] Aron A Wolf, Jeff Tooley, Scott Ploen, Mark Ivanov, Behçet Açikmeşe, and Konstantin Gromov. Performance trades for mars pinpoint landing. In 2006 IEEE Aerospace Conference, pages 16–pp. IEEE, 2006.

Appendix A

Additional Constraint Considerations

A.1 TVC Bandwidth Constraints

It may also be important to restrict the thrust vector control deflection velocities to maintain pragmatic control commands. This effectively limits the commandable bandwidth to the actuator. Therefore we can construct a numerical differentiation between control steps as such: $|\delta_{\text{TVC}_{k+1}} - \delta_{\text{TVC}_k}| \leq \dot{\delta}_{\text{TVC}_{max}} dt$ where of course $\hat{\boldsymbol{b}}_1^T \mathbf{F}_{th} = \|\mathbf{F}_{th}\|_2 \cos\delta_{\text{TVC}}$. The value $\dot{\delta}_{\text{TVC}_{max}}$ is just a constant determined by the guidance engineer a priori. We can use the dimensionality of our desired solution and a time-dilation final time coefficient to maintain scaling of the differentiation for the general free-final-time solution.

As proposed, we have that $|\delta_{\text{TVC}_{t+dt}} - \delta_{\text{TVC}_t}| \leq \dot{\delta}_{\text{TVC}_{max}} dt$. The discrete time equivalent is $|\delta_{\text{TVC}_{k+1}} - \delta_{\text{TVC}_k}| \leq \delta'_{\text{TVC}_{max}} d\tau$. We know that the dilated system works as $dt = \eta d\tau$ and of course that $d\tau = \frac{1}{K}$. It can then simply be encoded as the following constraint:

$$||\delta_{\text{TVC}_{k+1}} - \delta_{\text{TVC}_k}||_2 \leq \frac{\eta}{K} \dot{\delta}_{\text{TVC}_{max}} \tag{A.1}$$

$$||\Delta_\delta|| \leq \frac{\eta}{K} \dot{\delta}_{\text{TVC}_{max}} \tag{A.2}$$

$$\left\| \cos^{-1}\left(\frac{\mathbf{u}_k^T \mathbf{u}_{k+1}}{u_k u_{k+1}} \right) \right\| \leq \frac{\eta}{K} \dot{\delta}_{\text{TVC}_{max}} \tag{A.3}$$

Given that we know the domain of the angle differences is constrained to a positive semidefinite arena of operation in the bodt frame, we can perform a small angle approximation where $\mathbf{u}_k^T \mathbf{u}_{k+1} \leq u_k u_{k+1} \frac{\eta}{K} \dot{\delta}_{\text{TVC}_{max}}$. This is not apparently convex, but there may be simpler approaches to this

constraint which avoid adding an extra order term to the input dynamics. Of course, that may very well be the best way of implementing a first order constraint on TVC deflection.

A.2 Convexifying the Alignment Constraint

Additionally, the convexity of the constraint proposed in 2.21 is not immediately clear. After analysis, it is shown to be non-convex. Performing the first order Taylor linear approximation, we have the convexified form of this constraint:

$$\frac{16\,\hat{\sigma}_2\,\left(\hat{\sigma}_2-\sigma_2\right)\,\left(\hat{\sigma}^2+1-2(\hat{\sigma}_2^2+\hat{\sigma}_3^2)\right)}{\left(\hat{\sigma}^2+1\right)^3}-\frac{32\,\hat{\sigma}_1\,\left(\hat{\sigma}_1-\sigma_1\right)\,\left(\hat{\sigma}_2^2+\hat{\sigma}_3^2\right)}{\left(\hat{\sigma}^2+1\right)^3}-\frac{8(\hat{\sigma}_2^2+\hat{\sigma}_3^2)}{\left(\hat{\sigma}^2+1\right)^2}$$
$$+\frac{16\,\hat{\sigma}_3\,\left(\hat{\sigma}_3-\sigma_3\right)\,\left(\hat{\sigma}^2+1-2(\hat{\sigma}_2^2+\hat{\sigma}_3^2)\right)}{\left(\hat{\sigma}^2+1\right)^3}+1\geq\cos\psi_{max}$$
$$\Psi(\boldsymbol{\sigma},\hat{\boldsymbol{\sigma}})+1\geq\cos\psi_{max}$$

This constraint is not used in lieu of the simple MRP magnitude constraint.