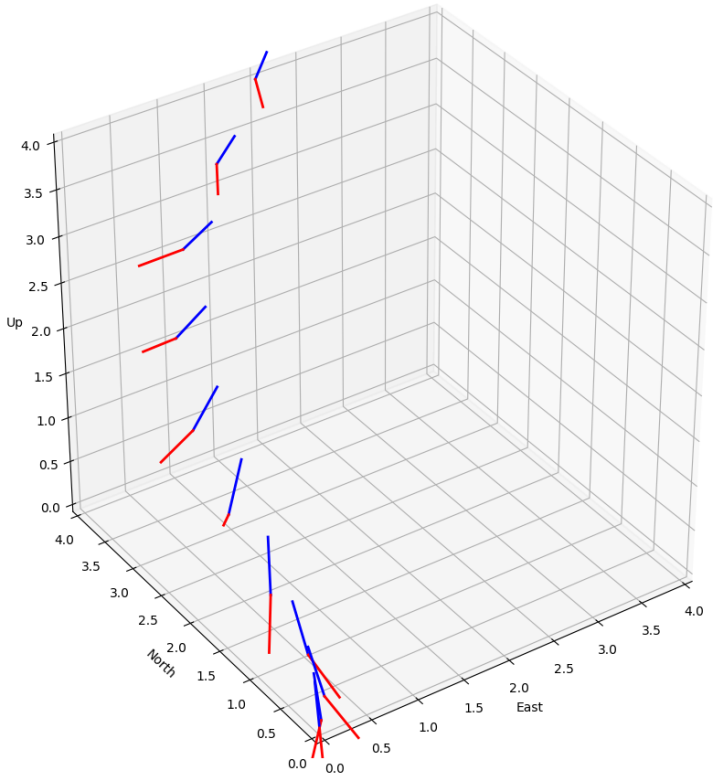
Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free Final Time

Spencer Powers

July 28, 2025



A walkthrough of the title paper by Szmuk and Açıkmeşe with implementation notes.

Contents

1 Introduction **1**
- 1.1 Notation . . . 1
- 1.2 Convexity . . . 1

2 Original Problem Formulation **3**
- 2.1 Kinematics and Dynamics . . . 3
- 2.2 State Constraints . . . 5
- 2.3 Control Constraints . . . 6
- 2.4 Boundary Conditions . . . 6
- 2.5 Problem Statement . . . 7

3 Convex Formulation **8**
- 3.1 Looking Ahead . . . 8
- 3.2 Linearization . . . 9
  - 3.2.1 Kinematics and Dynamics . . . 9
  - 3.2.2 Thrust Lower Bound Constraint . . . 9
  - 3.2.3 Quaternion Magnitude Constraint . . . 10
- 3.3 Discretization . . . 10
- 3.4 Successive Convexification . . . 13
  - 3.4.1 Trust Regions . . . 14
  - 3.4.2 Virtual Controls . . . 15
- 3.5 Miscellaneous Constraints . . . 16
- 3.6 Problem Statement . . . 16

4 Successive Convexification Algorithm **18**
- 4.1 Initialization . . . 18
- 4.2 Core Algorithm . . . 18

5 Demonstration **19**
- 5.1 In-plane Maneuver . . . 19
- 5.2 Out-of-plane Maneuver . . . 22

6 Future Work **25**

1 Introduction

This work concerns optimal guidance for precision powered landing of launch vehicles. I chose the title paper[1] because I was specifically looking for a real-time optimal control method that supports a full 6 DoF vehicle model.

The purpose of this document is to capture what I learned in the process of implementing the title paper. The accompanying source code can be found here. The target audience is anyone who finds the paper fascinating and wants to better understand how to close the gap between the presented theory and practical application.

Think of this like an annotated version of the original paper. Consequently, much of the content will be identical to the paper, but I will add or modify content along the way to in an effort to improve clarity and practicality.

1.1 Notation

I use slightly different notation than the original paper because my original notes were hand-written and you can't really differentiate between scalars and vectors using boldface in real life.

Standard lowercase symbols such as $m(t)$ denote scalars. Symbols with an arrow above them such as $\vec{F}(t)$ denote vectors. Standard uppercase symbols such as $A(t)$ denote matrices. Fancy F symbols ($\mathcal{F}$) represent coordinate frames. Any other variants (e.g., $\hat{\vec{x}}$, $\bar{A}$, $\tilde{A}$) will be defined as-needed later because they don't have generic definitions.

1.2 Convexity

Given that this method revolves around successive convexification, a brief primer on convexity is in order. I found another work by the same authors [2] helpful in understanding convexity in the context of optimization, but I'll briefly summarize the relevant components below (largely directly from that source).

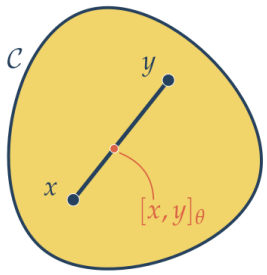
Let's start with the concepts of convex sets and convex functions. Figure 1 is from [2] and illustrates these concepts with notional examples. Convex sets are important because they are interchangeable with convex constraints, which we'll use extensively in later sections. By definition, $\mathcal{C} \subseteq \mathbb{R}^n$ is a convex set if and only if it contains the line segment connecting any two of its points:

$$\vec{x}, \vec{y} \in \mathcal{C} \Rightarrow [\vec{x}, \vec{y}]_\theta \in \mathcal{C} \tag{1}$$

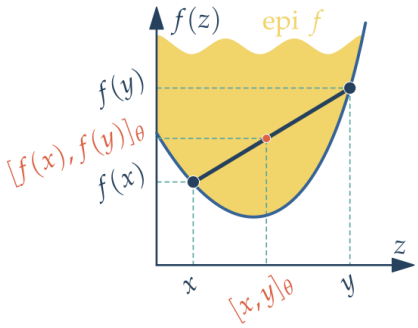
for all $\theta \in [0, 1]$, where $[\vec{x}, \vec{y}]_\theta \doteq \theta\vec{x} + (1-\theta)\vec{y}$.

Using the same θ machinery as above, a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if $dom(f)$ is a convex set and evaluating f at any point in its domain between $\vec{x}$ and $\vec{y}$ parameterized by θ yields a value under or on the associated value on the line segment connecting $f(\vec{x})$ and $f(\vec{y})$:

$$\vec{x}, \vec{y} \in dom(f) \Rightarrow f\left([\vec{x}, \vec{y}]_\theta\right) \leq [f(\vec{x}), f(\vec{y})]_\theta \tag{2}$$



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



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

Figure 1: Illustration from [2] of **(a)** a notional convex set and **(b)** a convex function. In both cases, the variable $\theta \in [0, 1]$ generates a line segment between two points. The epigraph $epi\ f \subseteq \mathbb{R}^n \times \mathbb{R}$ is the set of points which lie above the function and is thus a convex set itself.

Putting these two concepts together, we can now formally describe a convex optimization problem, which is central to the title paper. A convex optimization problem is simply the minimization of a convex function subject to a number of convex constraints that act to restrict the search space:

$$\min_{\vec{x} \in \mathbb{R}^n} \quad f(\vec{x}) \tag{3a}$$

$$\text{s.t.} \quad g_i(\vec{x}) \leq 0,\ i = 1, \ldots, n_{ineq} \tag{3b}$$

$$h_j(\vec{x}) = 0,\ j = 1, \ldots, n_{eq} \tag{3c}$$

where $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ is a convex cost function, $g_i(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ are convex inequality constraints, and $h_j(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ are affine equality constraints. Note that affine just means that the function is linear in x and can have a constant offset.

One important property of convex sets is that convexity is preserved under set intersection. Consequently, 3b and 3c combine to form a single convex set of

feasible values that the optimization decision variable x can take. Convex functions also have many properties that make their optimization quite tractable, but for now let's just say that off-the-shelf solvers are *really* good at solving convex optimization problems.

2 Original Problem Formulation

We will now building up the original continuous-time, non-convex problem description. There will be two coordinate frames of interest going forward:

1. $\mathcal{F}_{\mathcal{I}}$: An inertially-fixed Up-East-North reference frame with its origin located at the landing site.
2. $\mathcal{F}_{\mathcal{B}}$: A body-fixed frame centered at the vehicle center-of-mass, with its X-axis pointing along the vertical axis of the vehicle (i.e., along the thrust vector when the engine gimbal angle is zero), its Y-axis pointing out the side of the vehicle, and its Z-axis completing the right-handed system.

2.1 Kinematics and Dynamics

We treat the vehicle as a rigid body subject to constant gravitational acceleration $\vec{g}_{\mathcal{I}} \in \mathbb{R}^3$ and negligible aerodynamic forces. The vehicle is assumed to actuate a single gimbaled rocket engine to generate a thrust vector within a feasible range of magnitudes and gimbal angles.

We assume the vehicle depletes its mass $m(t) \in \mathbb{R}_{++}$ at a rate proportional to the magnitude of the commanded thrust vector $\vec{T}_{\mathcal{B}}(t) \in \mathbb{R}^3$, which is expressed in $\mathcal{F}_{\mathcal{B}}$ coordinates. For tractability, we assume that the inertia tensor and the position of the center-of-mass are constant despite the depletion of mass. The proportionality constant $\alpha_{\dot{m}}$ is given in terms of the vacuum-specific-impulse I_{sp} (an exception to the aforementioned notational rules) and Earth's standard gravity constant g_0 as follows:

$$\alpha_{\dot{m}} \doteq \frac{1}{I_{sp} g_0} \tag{4}$$

We can now write our mass depletion dynamics in the following form:

$$\dot{m}(t) = -\alpha_{\dot{m}} \left\| \vec{T}_{\mathcal{B}}(t) \right\|_2 \tag{5}$$

We express the position, velocity, and force acting on the vehicle in $\mathcal{F}_{\mathcal{I}}$ coordinates and write them as $\vec{r}_{\mathcal{I}}$, $\vec{v}_{\mathcal{I}}$, and $\vec{F}_{\mathcal{I}}$, respectively. We can thus write the vehicle's translational dynamics as:

$$\dot{\vec{r}}_{\mathcal{I}}(t) = \vec{v}_{\mathcal{I}}(t) \tag{6}$$

$$\dot{\vec{v}}_{\mathcal{I}}(t) = \frac{1}{m(t)} \vec{F}_{\mathcal{I}}(t) + \vec{g}_{\mathcal{I}} \tag{7}$$

We use unit quaternions to parameterize the attitude of $\mathcal{F}_\mathcal{B}$ relative to $\mathcal{F}_\mathcal{I}$, and denote them by $\vec{q}_{\mathcal{B}/\mathcal{I}}(t) \in \mathcal{S}^3$. This paper uses the *leading scalar element* convention. Let us define their elements as $\vec{q}_{\mathcal{B}/\mathcal{I}}(t) \doteq [q_0 \; q_1 \; q_2 \; q_3]^T$.

The direction cosine matrix (DCM) that encodes the attitude transformation from $\mathcal{F}_\mathcal{I}$ to $\mathcal{F}_\mathcal{B}$ is denoted by $C_{\mathcal{B}/\mathcal{I}}(t) \in SO(3)$, where $C_{\mathcal{B}/\mathcal{I}}(t)$ is related to $\vec{q}_{\mathcal{B}/\mathcal{I}}(t)$ through the following relation:

$$C_{\mathcal{B}/\mathcal{I}} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{8}$$

This was one initial point of confusion because if you go to the Wikipedia entry discussing deriving the rotation matrix from a given quaternion, you get the transpose of the above matrix. The same Wikipedia page discourages the use of this particular convention, but I wanted to stay reasonably true to the original paper's theory, so I kept it in my implementation.

Since $C_{\mathcal{B}/\mathcal{I}}(t) \in SO(3)$, we can write the inverse transformation from $\mathcal{F}_\mathcal{B}$ to $\mathcal{F}_\mathcal{I}$ as $C_{\mathcal{I}/\mathcal{B}} = C_{\mathcal{B}/\mathcal{I}}^{-1} = C_{\mathcal{B}/\mathcal{I}}^T$. We can now update our translational dynamics (7) to use the vehicle thrust vector in place of the inertial force vector, as the former is what we'll actually be controlling:

$$\dot{\vec{v}}_\mathcal{I}(t) = \frac{1}{m(t)} C_{\mathcal{I}/\mathcal{B}}(t)\vec{T}_\mathcal{B}(t) + \vec{g}_\mathcal{I} \tag{9}$$

We use $\vec{\omega}_\mathcal{B}(t) \in \mathbb{R}^3$ to denote the angular velocity vector of $\mathcal{F}_\mathcal{B}$ relative to $\mathcal{F}_\mathcal{I}$, expressed in $\mathcal{F}_\mathcal{B}$ coordinates. Additionally, for some $\vec{\xi} \in \mathbb{R}^3$, we define the skew-symmetric matrices $\vec{\xi}^\wedge$ and $\Omega(\vec{\xi})$ as follows:

$$\vec{\xi}^\wedge = \begin{bmatrix} 0 & -\xi_z & \xi_y \\ \xi_z & 0 & -\xi_x \\ -\xi_y & \xi_x & 0 \end{bmatrix} \tag{10}$$

$$\Omega(\vec{\xi}) = \begin{bmatrix} 0 & -\xi_x & \xi_y & -\xi_z \\ \xi_x & 0 & \xi_z & -\xi_y \\ \xi_y & -\xi_z & 0 & \xi_x \\ \xi_z & \xi_y & -\xi_x & 0 \end{bmatrix} \tag{11}$$

We denote the torque acting on the vehicle as $\vec{M}_\mathcal{B}(t) \in \mathbb{R}^3$, and the inertia tensor of the vehicle in $\mathcal{F}_\mathcal{B}$ coordinates as $J_\mathcal{B}$. Based on our earlier assumption that the center of mass does not move, it follows that the moment arm from the center of mass to the gimbal point of the engine is constant. We denote this constant position vector in $\mathcal{F}_\mathcal{B}$ coordinates as $\vec{r}_{T,\mathcal{B}} \in \mathbb{R}^3$. It then follows that the body-frame torque is just $M_\mathcal{B}(t) = \vec{r}_{T,\mathcal{B}}^\wedge \vec{T}_\mathcal{B}(t) = \vec{r}_{T,\mathcal{B}} \times \vec{T}_\mathcal{B}(t)$. We can then finally write the quaternion kinematics and attitude dynamics as follows:

$$\dot{\vec{q}}_{\mathcal{B}/\mathcal{I}}(t) = \frac{1}{2}\Omega(\vec{\omega_\mathcal{B}}(t))\vec{q}_{\mathcal{B}/\mathcal{I}}(t) \tag{12}$$

$$J_{\mathcal{B}}\dot{\vec{\omega}}_{\mathcal{B}}(t) = \vec{r}^{\wedge}_{T,\mathcal{B}}\vec{T}_{\mathcal{B}}(t) - \vec{\omega}^{\wedge}_{\mathcal{B}}(t)J_{\mathcal{B}}\vec{\omega}_{\mathcal{B}}(t) \tag{13}$$

Let us define our complete vehicle state vector $\vec{x} \in \mathbb{R}^{14}$ as:

$$\vec{x}(t) \doteq \left[m(t)\ \vec{r}^{T}_{\mathcal{I}}(t)\ \vec{v}^{T}_{\mathcal{I}}(t)\ \vec{q}^{T}_{\mathcal{B}/\mathcal{I}}(t)\ \vec{\omega}^{T}_{\mathcal{B}}(t)\right]^{T} \tag{14}$$

As a practical note, we'll later need to write our full nonlinear dynamics in the form $\dot{\vec{x}} = f(\vec{x}, \vec{u})$. We can immediately use Equations 5, 6, 9, and 12, but we'll have to left-multiply both sides of 13 by $J_{\mathcal{B}}^{-1}$ to leave only $\dot{\omega}_{\mathcal{B}}(t)$ on the left-hand side:

$$\dot{\omega}_{\mathcal{B}}(t) = J_{\mathcal{B}}^{-1}\left[\vec{r}^{\wedge}_{T,\mathcal{B}}\vec{T}_{\mathcal{B}}(t) - \vec{\omega}^{\wedge}_{\mathcal{B}}(t)J_{\mathcal{B}}\vec{\omega}_{\mathcal{B}}(t)\right] \tag{15}$$

2.2 State Constraints

The first implied state constraint is that the kinematics and dynamics in the previous section are obeyed. We'll need to convexify this constraint later, as it's an equality constraint and it clearly doesn't fit the affine requirement described by 3c.

We additionally restrict the mass of the vehicle to remain above the dry mass m_{dry} using the following convex constraint:

$$m_{dry} \leq m(t) \tag{16}$$

The path of the vehicle is also restricted to lie within an upward-facing glide-slope cone that makes an angle $\gamma_{gs} \in [0°, 90°)$ with the horizontal and is centered at the origin of $\mathcal{F}_{\mathcal{I}}$. This can be expressed with the following convex constraint:

$$\vec{e}_1 \cdot \vec{r}_{\mathcal{I}}(t) \geq \tan\gamma_{gs}\left\|H_{23}^{T}\vec{r}_{\mathcal{I}}(t)\right\|_2 \tag{17}$$

where $\vec{e}_i$ is the unit vector along the ith axis and $H_{23} \doteq [\vec{e}_2\ \vec{e}_3]$. If we look at the expression as an equality expression, we can see from trigonometry that it just restricts the altitude of some point $\vec{r}_{\mathcal{I}}(t)$ to exist on the surface of the inverted cone; the norm expression on the right-hand side is just a compact way of getting the Euclidean distance to the origin of $\mathcal{F}_{\mathcal{I}}$ on the East-North plane. By making it an inequality, we allow $\vec{r}_{\mathcal{I}}(t)$ to exist on the inside of the inverted cone in addition to its surface.

We define the tilt angle of the vehicle $\theta(t)$ as the angle between the X-axes of $\mathcal{F}_{\mathcal{B}}$ and $\mathcal{F}_{\mathcal{I}}$:

$$\cos\theta(t) = \vec{e}_1 \cdot C_{\mathcal{I}/\mathcal{B}}(t)\vec{e}_1 = 1 - 2\left(q_2^2(t) + q_3^2(t)\right) \tag{18}$$

To avoid excessive tilt angles in the trajectory, we limit $\theta(t)$ to a maximum value of θ_{max}. If we immerse our quaternion $\vec{q}_{\mathcal{B}/\mathcal{I}}$ in $\mathbb{R}^4$, we can impose this tilt angle limit through the following convex constraint:

$$\cos\theta_{max} \leq 1 - 2\left(q_2^2(t) + q_3^2(t)\right) \tag{19}$$

We can write this in a more compact form as follows:

$$\cos\theta_{max} \leq 1 - 2\left|H_q \vec{q}_{\mathcal{B}/\mathcal{I}}\right|_2^2 \tag{20a}$$

$$H_q \doteq \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20b}$$

We also limit the vehicle's body angular rates using the following convex constraint:

$$\left\|\vec{\omega}_{\mathcal{B}}(t)\right\|_2 \leq \omega_{max} \tag{21}$$

Unlike the paper, I also added a quaternion magnitude constraint of unity because the direction cosine matrix used in the dynamics is strictly defined for unit quaternions. This will require more work later because it clearly does not fit the affine equality constraint formulation in 3.

2.3 Control Constraints

We'll assume that the engine is throttleable, but this particular problem formulation requires that the singular engine is already restarted at the beginning of the problem. It also implies that we can't shut it down and restart it later. The result is that in addition to the natural upper bound on the thrust produced by the engine, we have a positive lower bound as well:

$$0 < T_{min} \leq \left\|\vec{T}_{\mathcal{B}}(t)\right\|_2 \leq T_{max} \tag{22}$$

The upper bound is convex, but the lower bound is not. You can intuitively see why if you picture the constraint in 3D space. The upper bound just defines a ball around the origin that we have to exist within (or on its surface), and naturally a ball contains every line segment connecting points within it. The lower bound removes a smaller ball around the origin from the valid set, which means any line segment between points on opposite sides of the inner ball will no longer be contained within the valid set.

Our gimbal mechanism also has a maximum gimbal angle of $\delta_{max} \in [0°, 90°)$, which constrains the possible directions of our thrust vector in body frame via the following convex constraint:

$$\cos\delta_{max}\left\|\vec{T}_{\mathcal{B}}(t)\right\|_2 \leq \vec{e}_1 \cdot \vec{T}_{\mathcal{B}}(t) \tag{23}$$

2.4 Boundary Conditions

For initial conditions, the original paper constrains the vehicle mass, position, velocity, and angular rate to match some known conditions. Surprisingly, they leave the initial attitude unconstrained. My implementation also constrains the

initial attitude because presumably if this were ever deployed on a vehicle, we'd have some estimate of our current attitude.

For terminal conditions, the original paper constrains the vehicle position (all zeros in $\mathcal{F}_\mathcal{I}$), velocity (small), attitude (aligned with $\mathcal{F}_\mathcal{I}$), and angular rates (all zeros), as well as the requiring that the final thrust is aligned with the vehicle's vertical axis. I opted for a terminal tilt constraint instead of requiring that the frames be exactly aligned, as that felt more realistic; it shouldn't matter if the vehicle is rotated about its vertical axis, and any landing mechanism should be able to tolerate a couple degrees of tilt.

As a practical note, I constructed the solver interface around the idea of continuously re-solving the optimal control problem during descent. The config file defines the terminal conditions, as those conditions shouldn't really change. The only public method takes in a current vehicle state vector and solves the optimal control problem, which should be a nice interface for using this library in an MPC-like manner.

2.5 Problem Statement

We can now write our continuous-time non-convex optimization problem:

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

$$\min_{t_f, \vec{T}_\mathcal{B}(t)} \quad t_f$$

subject to:

Dynamics:

$$\dot{m}(t) = -\alpha_{\dot{m}} \left\| \vec{T}_\mathcal{B} \right\|_2$$

$$\dot{\vec{r}}_\mathcal{I}(t) = \vec{v}_\mathcal{I}(t)$$

$$\dot{\vec{v}}_\mathcal{I}(t) = \frac{1}{m(t)} C_{\mathcal{I}/\mathcal{B}}(t) \vec{T}_\mathcal{B}(t) + \vec{g}_\mathcal{I}$$

$$\dot{\vec{q}}_{\mathcal{B}/\mathcal{I}}(t) = \frac{1}{2} \Omega(\vec{\omega}_\mathcal{B}(t)) \vec{q}_{\mathcal{B}/\mathcal{I}}(t)$$

$$\dot{\omega}_\mathcal{B}(t) = J_\mathcal{B}^{-1} \left[\vec{r}^{\wedge}_{T,\mathcal{B}} \vec{T}_\mathcal{B}(t) - \vec{\omega}^{\wedge}_\mathcal{B}(t) J_\mathcal{B} \vec{\omega}_\mathcal{B}(t) \right]$$

State Constraints:

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

$$\tan \gamma_{gs} \left\| H_{23}^T \vec{r}_\mathcal{I}(t) \right\|_2 \leq \vec{e}_1 \cdot \vec{r}_\mathcal{I}(t)$$

$$\cos \theta_{max} \leq 1 - 2 \left(q_2^2(t) + q_3^2(t) \right)$$

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

$$\left\| \vec{q}_{\mathcal{B}/\mathcal{I}}(t) \right\|_2 = 1$$

Control Constraints:

$$0 < T_{min} \leq \left\| \vec{T}_{\mathcal{B}}(t) \right\|_2 \leq T_{max}$$

$$\cos \delta_{max} \left\| \vec{T}_{\mathcal{B}}(t) \right\|_2 \leq \vec{e}_1 \cdot \vec{T}_{\mathcal{B}}(t)$$

Boundary Conditions:

$$m(0) = m_{wet}$$

$$\vec{r}_{\mathcal{I}}(0) = \vec{r}_{\mathcal{I},i}$$

$$\vec{v}_{\mathcal{I}}(0) = \vec{v}_{\mathcal{I},i}$$

$$\vec{q}_{\mathcal{B}/\mathcal{I}}(0) = \vec{q}_{\mathcal{B}/\mathcal{I},i}$$

$$\vec{\omega}_{\mathcal{B}}(0) = \vec{\omega}_{\mathcal{B},i}$$

$$\vec{r}_{\mathcal{I}}(t_f) = \vec{0}$$

$$\vec{v}_{\mathcal{I}}(t_f) = \vec{v}_{\mathcal{I},f}$$

$$\cos \theta_{max,f} \leq 1 - 2\left(q_2^2(t_f) + q_3^2(t_f)\right)$$

$$\vec{\omega}_{\mathcal{B}}(0) = \vec{0}$$

$$\vec{e}_2 \cdot \vec{T}_{\mathcal{B}} = \vec{e}_3 \cdot \vec{T}_{\mathcal{B}} = 0$$

To reiterate, this formulation differs from the original paper's problem formulation in a couple ways; I constrain initial attitude, and I relax the terminal attitude constraint to be a max tilt constraint instead of rigidly aligning with the inertial frame, both of which seemed more realistic to me.

3 Convex Formulation

3.1 Looking Ahead

Let's first give a brief overview of what we plan to do with the convexified problem. The original problem is so non-convex that just solving a convexified version of the problem once will almost certainly not yield a solution that satisfies the original constraints. The fundamental premise of the successive convexification algorithm repeatedly solves the convexified problem, and the converged solution should satisfy the original dynamics and constraints of Problem 1. The exact mechanisms that drive the solution towards a feasible solution will be discussed in Section 3.4.

We will be specifically formulating the convex sub-problem as a Second-Order Cone Problem (SOCP). This imposes additional limitations on the forms of the cost function and constraints, but it allows us to use specific SOCP solvers such as ECOS to efficiently solve each iteration of the successive convexification algorithm.

3.2 Linearization

3.2.1 Kinematics and Dynamics

To get the original kinematics and dynamics to fit into a convex optimization framework, step one is linearizing them. We already defined the total state vector in 14, but just for notational consistency let's define the total system control vector as $\vec{u}(t) \doteq \vec{T}_{\mathcal{B}}(t)$.

We begin by expressing the original kinematics and dynamics as a nonlinear vector-valued function $f : \mathbb{R}^{14} \times \mathbb{R}^3 \to \mathbb{R}^{14}$ of the state and control vectors:

$$\frac{d}{dt}\vec{x}(t) = f\left(\vec{x}(t), \vec{u}(t)\right) \doteq \left[\dot{m}(t)\ \dot{\vec{r}}^T_{\mathcal{I}}(t)\ \dot{\vec{v}}^T_{\mathcal{I}}(t)\ \dot{\vec{q}}^T_{\mathcal{B}/\mathcal{I}}(t)\ \dot{\vec{\omega}}^T_{\mathcal{B}}(t)\right]^T \tag{24}$$

where the individual terms can be substituted with Eqs. 5, 6, 9, 12, and 15.

Recall that we need to keep the final time free. To do this, let's express Eq. 24 in terms of a normalized time $\tau \in [0, 1]$. You can think of this normalized dimensionless time as a function of standard time: $\tau(t) = \frac{t}{t_f}$. Naturally, we can also go the other direction, i.e., $t(\tau) = \tau t_f$. We can now define the dilation coefficient, an important quantity that will be at the heart of the final optimization:

$$\sigma \doteq \left(\frac{d\tau}{dt}\right)^{-1} = t_f \tag{25}$$

We can then use chain rule to rewrite Eq. 24 in terms of our normalized time τ and dilation coefficient σ:

$$\dot{\vec{x}}(\tau) \doteq \frac{d}{d\tau}\vec{x}(\tau) = \sigma f(\vec{x}(\tau), \vec{u}(\tau)) \tag{26}$$

We now have our dynamics in a fixed-final-time setting, but they still need to be linearized to fit into the convex optimization framework. Let's replace the right-hand-side with a first-order Taylor series approximation evaluated at a reference state trajectory $\hat{\vec{x}}(\tau)$, a reference control trajectory $\hat{\vec{u}}(\tau)$, and a reference dilation coefficient $\hat{\sigma}$:

$$\dot{\vec{x}}(\tau) = A(\tau)\vec{x}(\tau) + B(\tau)\vec{u}(\tau) + \Sigma(\tau)\sigma + \vec{z}(\tau) \tag{27a}$$

$$A(\tau) \doteq \hat{\sigma} \cdot \frac{\partial}{\partial \vec{x}} f(\vec{x}, \vec{u})\bigg|_{\hat{\vec{x}}(\tau), \hat{\vec{u}}(\tau)} \tag{27b}$$

$$B(\tau) \doteq \hat{\sigma} \cdot \frac{\partial}{\partial \vec{u}} f(\vec{x}, \vec{u})\bigg|_{\hat{\vec{x}}(\tau), \hat{\vec{u}}(\tau)} \tag{27c}$$

$$\Sigma(\tau) \doteq f(\hat{\vec{x}}(\tau), \hat{\vec{u}}(\tau)) \tag{27d}$$

$$\vec{z}(\tau) \doteq -A(\tau)\hat{\vec{x}}(\tau) - B(\tau)\hat{\vec{u}}(\tau) \tag{27e}$$

3.2.2 Thrust Lower Bound Constraint

The thrust lower bound constraint in Problem 1 is non-convex, so we'll linearize it. First, define function $g : \mathbb{R}^3 \to \mathbb{R}$ as follows:

$$g\left(\vec{u}(\tau)\right) \doteq T_{min} - \|\vec{u}(\tau)\|_2 \leq 0 \tag{28}$$

This just mimics the classic inequality constraint form that we eventually want. We can then replace the left-hand side with a first-order Taylor series approximation about the reference control trajectory $\hat{\vec{u}}(\tau)$:

$$\begin{aligned} g\left(\vec{u}(\tau)\right) &\approx g\left(\hat{\vec{u}}(\tau)\right) + \left.\frac{\partial g\left(\vec{u}(\tau)\right)}{\partial \vec{u}}\right|_{\hat{\vec{u}}(\tau)} \left(\vec{u}(\tau) - \hat{\vec{u}}(\tau)\right) \\ &= T_{min} - \left\|\hat{\vec{u}}(\tau)\right\|_2 + \left(-\frac{\hat{\vec{u}}^T(\tau)}{\left\|\hat{\vec{u}}(\tau)\right\|_2}\right)\left(\vec{u}(\tau) - \hat{\vec{u}}(\tau)\right) \end{aligned}$$

which brings us to our linearized constraint form:

$$T_{min} - \frac{\hat{\vec{u}}^T(\tau)\vec{u}(\tau)}{\left\|\hat{\vec{u}}(\tau)\right\|_2} \leq 0 \tag{29}$$

3.2.3 Quaternion Magnitude Constraint

The quaternion unity magnitude constraint clearly does not fit the affine equality constraint form in 3. We'll replace it with two inequality constraints, one bounding the magnitude from below and one bounding the magnitude from above, both of which use 1 as the bounding value. Just like the thrust magnitude constraints, the upper bound constraint is convex, but the lower bound is not. We'll follow the exact same procedure as above to linearize the lower bound constraint. This yields the following linearized form of the unity magnitude lower bound constraint:

$$1 - \hat{\vec{q}}^T_{\mathcal{B}/\mathcal{I}}(\tau)\vec{q}_{\mathcal{B}/\mathcal{I}}(\tau) \leq 0 \tag{30}$$

You might immediately notice that this is a bit problematic. Since both the reference quaternion and the current quaternion have unit magnitude, the left-hand side can be 1 only if they are exactly the same, and it can never actually be less than zero. If we left this as-is, the quaternions would effectively never be allowed to change from the reference trajectory. To get around this, I added a tolerance ϵ_q and thereby allow the dot product of the quaternions to be slightly less than 1 in order to satisfy the constraint:

$$(1 - \epsilon_q) - \hat{\vec{q}}^T_{\mathcal{B}/\mathcal{I}}(\tau)\vec{q}_{\mathcal{B}/\mathcal{I}}(\tau) \leq 0 \tag{31}$$

3.3 Discretization

We have removed the free final time nature of the problem and linearized it, but it's still continuous in time. In order for us to fit it into a finite-dimensional parameter optimization problem, we discretize the trajectory into K evenly-distributed points. We perform the discretization with respect to normalized

trajectory time τ . For convenience, we define the following two sets:

$$\mathcal{K} \doteq \{0,\ 1,\ \ldots,\ K-2,\ K-1\}$$
$$\bar{\mathcal{K}} \doteq \{0,\ 1,\ \ldots,\ K-3,\ K-2\}$$

Given τ's definition, we can define the normalized time at discretization index k as:

$$\tau_k \doteq \frac{k}{K-1}\ , \quad \forall k \in \mathcal{K} \tag{32}$$

To preserve more feasibility, we assume a first-order-hold on the control over each time step. Thus, over the interval $\tau \in [\tau_k, \tau_{k+1}]$, we can express $\vec{u}(\tau)$ in terms of $\vec{u}_k \doteq \vec{u}(\tau_k)$ and $\vec{u}_{k+1} \doteq \vec{u}(\tau_{k+1})$ as follows:

$$\vec{u}(\tau) = \alpha_k(\tau)\vec{u}_k + \beta_k(\tau)\vec{u}_{k+1}\ , \quad \tau \in [\tau_k, \tau_{k+1}]\ , \forall k \in \bar{\mathcal{K}} \tag{33a}$$
$$\alpha_k(\tau) = \frac{\tau_{k+1} - \tau}{\tau_{k+1} - \tau_k} \tag{33b}$$
$$\beta_k(\tau) = \frac{\tau - \tau_k}{\tau_{k+1} - \tau_k} \tag{33c}$$

Now we come to the problem of discretizing the dynamics. We use $\Phi_A(\tau_{k+1}, \tau_k)$ to denote the state transition matrix that describes the zero-input evolution from $\vec{x}_k$ to $\vec{x}_{k+1}$. We're going to use the state transition matrix in the following derivations because it allows us to encapsulate the dynamics constraint between two discretization points separated by a potentially significant time window.

Recall that the solution for a linear time-varying system with a single input $\vec{u}$ can be expressed using the state transition matrix as follows (using generic notation):

$$\vec{x}(t) = \Phi(t, t_0)\vec{x}(t_0) + \int_{t_0}^{t} \Phi(t, \tau)B(\tau)\vec{u}(\tau)d\tau \tag{34}$$

Applying this solution form to our problem immediately yields:

$$\vec{x}_{k+1} = \Phi_A(\tau_{k+1}, \tau_k)\vec{x}(\tau_k) + \tag{35}$$
$$\int_{\tau_k}^{\tau_{k+1}} \Phi_A(\tau_{k+1}, \xi)\left[B(\xi)\alpha_k(\xi)\vec{u}_k + B(\xi)\beta_k(\xi)\vec{u}_{k+1} + \Sigma(\xi)\sigma + \vec{z}(\xi)\right] d\xi \tag{36}$$

which we can now rewrite in the form presented in the paper:

$$\vec{x}_{k+1} = \bar{A}_k \vec{x}_k + \bar{B}_k \vec{u}_k + \bar{C}_k \vec{u}_{k+1} + \bar{\Sigma}_k \sigma + \bar{\vec{z}}_k \tag{37a}$$

$$\bar{A}_k \doteq \Phi_A(\tau_{k+1}, \tau_k) \tag{37b}$$

$$\bar{B}_k \doteq \int_{\tau_k}^{\tau_{k+1}} \Phi_A(\tau_{k+1}, \xi) B(\xi) \alpha_k(\xi) d\xi \tag{37c}$$

$$\bar{C}_k \doteq \int_{\tau_k}^{\tau_{k+1}} \Phi_A(\tau_{k+1}, \xi) B(\xi) \beta_k(\xi) d\xi \tag{37d}$$

$$\bar{\Sigma}_k \doteq \int_{\tau_k}^{\tau_{k+1}} \Phi_A(\tau_{k+1}, \xi) \Sigma(\xi) d\xi \tag{37e}$$

$$\bar{\vec{z}}_k \doteq \int_{\tau_k}^{\tau_{k+1}} \Phi_A(\tau_{k+1}, \xi) \vec{z}(\xi) d\xi \tag{37f}$$

How do you actually calculate the bar quantities (Eqs. 37b through 37f)? The paper does not go into this, but Michael Szmuk's doctoral thesis[3] gives us a hint, and Sven Niederberger's SCpp repo[4] provides an example. Buckle up.

Let's start with some important properties of the generic state transition matrix:

$$\Phi(t_1, t_0)^{-1} = \Phi(t_0, t_1) \tag{38a}$$

$$\Phi(t_2, t_0) = \Phi(t_2, t_1)\Phi(t_1, t_0) \tag{38b}$$

$$\frac{\partial}{\partial t}\Phi(t, t_0) = A(t)\Phi(t, t_0) \tag{38c}$$

Using Properties 38a and 38b, we can write:

$$\begin{aligned} \Phi_A(\tau_{k+1}, \xi) &= \Phi_A(\tau_{k+1}, \tau_k)\Phi_A(\tau_k, \xi) \\ &= \Phi_A(\tau_{k+1}, \tau_k)\Phi_A(\xi, \tau_k)^{-1} \end{aligned} \tag{39}$$

The left-hand side is the term that shows up in the integrands of Eqs. 37b through 37f. The first term on the right-hand side is not a function of the integration variable ξ, which means we can pull it outside of the integral. The second term on the right-hand side now has the integration variable as the first argument, which means it's in the correct order to leverage property 38c.

We're going to need to use an integrator to compute the definite integrals in Eqs. 37c through 37f. Eq. 39 allows us to define some intermediate quantities that we can actually feed to an integrator:

$$\tilde{B}_k \doteq \int_{\tau_k}^{\tau_{k+1}} \Phi_A(\xi, \tau_k)^{-1} B(\xi) \alpha_k(\xi) d\xi \tag{40a}$$

$$\tilde{C}_k \doteq \int_{\tau_k}^{\tau_{k+1}} \Phi_A(\xi, \tau_k)^{-1} B(\xi) \beta_k(\xi) d\xi \tag{40b}$$

$$\tilde{\Sigma}_k \doteq \int_{\tau_k}^{\tau_{k+1}} \Phi_A(\xi, \tau_k)^{-1} \Sigma(\xi) d\xi \tag{40c}$$

$$\tilde{\vec{z}}_k \doteq \int_{\tau_k}^{\tau_{k+1}} \Phi_A(\xi, \tau_k)^{-1} \vec{z}(\xi) d\xi \tag{40d}$$

The complete matrix that we're going to feed to the integrator is:

$$V = \left[\vec{x}(\tau) \ \Phi_A(\tau, \tau_k) \ \frac{\partial \tilde{B}_k}{\partial \tau} \ \frac{\partial \tilde{C}_k}{\partial \tau} \ \frac{\partial \tilde{\Sigma}_k}{\partial \tau} \ \frac{\partial \tilde{\vec{z}}_k}{\partial \tau} \right] \in \mathbb{R}^{14 \times 23} \tag{41}$$

where we can get the relevant sections of $\frac{dV}{d\tau}$ from the full nonlinear dynamics, Property 38c, and the integrands of the above definitions of the tilde quantities, respectively.

At the beginning of every discrete segment, we initialize this V matrix to set the first column to be $\vec{x}_k$, the $\Phi_A(\tau, \tau_k)$ block to be the identity matrix, and the rest of the elements are zero. Recall that the quantities in Eqs. 37b through 37f all implicitly use the reference trajectory information. This means that inside the integrator, as we're integrating V over a single discretization step, we'll need to evaluate the full nonlinear dynamics f and its partial derivatives along the reference trajectory. This is why we drag $\vec{x}(\tau)$ through the integration process; by starting at $\vec{x}_k$, computing $\vec{u}(\tau)$ via Eq. 33a, and using the full nonlinear dynamics, we can always evaluate f, A, and B given that first column's current value and the computed $\vec{u}(\tau)$, then use those quantities to populate the derivative information for all of the other blocks in V. Honestly this is easier to understand by looking at the definition of `DiscreteLinearizedVehicleIntegrator::operator()` in the source code linked in the introduction.

Once we integrate $\frac{dV}{d\tau}$ from τ_k to τ_{k+1}, the values in the $\Phi_A(\tau, \tau_k)$ block will yield $\Phi_A(\tau_{k+1}, \tau_k) = \bar{A}_k$, and every block after that yields quantities 40a through 40d. However, we're not done yet, because quantities 40a through 40d all need to be left-multiplied by $\Phi_A(\tau_{k+1}, \tau_k)$ to yield quantities 37c through 37f. This now gives us all of the quantities required to enforce the discrete linear dynamics (Eq. 37a) at each discretization point. The rest of the state and control constraints are just enforced as-is at each discretization point.

3.4 Successive Convexification

In order for successive convexification to work, we must ensure that the problem remains bounded and feasible throughout the convergence process. Section

3.4.1 will address keeping the problem bounded (meaning the costs don't blow up), and Section 3.4.2 will address ensuring the solution is feasible (meaning it satisfies the constraints).

3.4.1 Trust Regions

The linearization of an iterate can result in constraints that admit an unbounded cost. We mitigate the possibility of unbounded solutions in each sub-problem by augmenting the cost function with terms that serve as soft trust regions defined around the previous iterate. To do so, we first define the relative quantities shown below:

$$\delta\vec{x}_k^i \doteq \vec{x}_k^i - \vec{x}_k^{i-1}, \quad \forall k \in \mathcal{K} \tag{42a}$$

$$\delta\vec{u}_k^i \doteq \vec{u}_k^i - \vec{u}_k^{i-1}, \quad \forall k \in \mathcal{K} \tag{42b}$$

$$\delta\sigma^i \doteq \sigma^i - \sigma^{i-1} \tag{42c}$$

where the i superscript denotes the ith iterate. Then, defining $\bar{\vec{\Delta}}^i \in \mathbb{R}_+^K$ and $\Delta_\sigma^i \in \mathbb{R}_+$, we impose the following constraints:

$$\delta\vec{x}_k^i \cdot \delta\vec{x}_k^i + \delta\vec{u}_k^i \cdot \delta\vec{u}_k^i \leq \vec{e}_k \cdot \bar{\vec{\Delta}}^i \tag{43a}$$

$$\delta\sigma^i \cdot \delta\sigma^i \leq \Delta_\sigma^i \tag{43b}$$

We can then use these bounds to add the following term to our cost function:

$$c_\Delta^i \doteq w_\Delta^i \left\| \bar{\vec{\Delta}}^i \right\|_2 + w_{\Delta_\sigma} \left\| \Delta_\sigma^i \right\|_1 \tag{44}$$

The intuition here is straightforward. Recall that the linearization is performed around the reference trajectory, which is calculated between each discrete segment by starting with state $\hat{\vec{x}}_k$ and numerically integrating over the interval $\tau \in [\tau_k, \tau_{k+1}]$ using reference controls generated via Eq. 33a. Given that the original problem is highly nonlinear, the linearization will only be reasonably valid in a small region around the reference trajectory, so we want to penalize the solver from straying too far from the last iterate.

I hit a couple practical hurdles with the above original formulation. First, the point of this paper is to convert the original optimization problem into a sequence of SOCPs that can be efficiently handled by dedicated SOCP solvers. Inequality constraints for SOCPs can be either linear in the optimization variables or in the 2-norm form of $\|A_i\vec{x} + b_i\|_2$. These new constraints are quadratic in the optimization variables. I opted to take the square root of both sides of Eq. 43a to get around this. Also, because we only care about the 1-norm of the Δ_σ^i term, I opted to skip the square root modification and instead just directly penalize the absolute value of $\delta\sigma^i$. Both of these modifications inherently change the problem and thus require different weights in the final cost function compared to the original paper.

Also, the cost function of an SOCP must be linear in the optimization variables. Consequently, the new cost term (Eq. 44) must be modified. To get around this, I added a couple new optimization variables, $\bar{\Delta}_{norm} \in \mathbb{R}_+$ and $\Delta_{\sigma,abs} \in \mathbb{R}_+$, along with their associated constraints:

$$\left\| \bar{\vec{\Delta}}^i \right\|_2 \leq \bar{\Delta}_{norm} \tag{45a}$$

$$-\Delta_{\sigma,abs} \leq \delta\sigma^i \leq \Delta_{\sigma,abs} \tag{45b}$$

$$0 \leq \Delta_{\sigma,abs} \tag{45c}$$

We can now update our new cost term to be SOCP-compatible:

$$c^i_{\Delta,aug} \doteq w^i_\Delta \bar{\Delta}_{norm} + w_{\Delta_\sigma} \Delta_{\sigma,abs} \tag{46}$$

As a practical note, notice how w^i_Δ has an i superscript. The original paper does not discuss this, but I'm guessing that means that they use some kind of gain scheduling over successive convexification iterations. I ended up doing a very simple schedule where I allow the w_ν weight (discussed in the next section) to dominate for the first few iterations, thereby incentivizing the solver to produce *some* feasible solution, after which point I scale up the w_ν weight for the final iterations. I found that this was actually necessary because the default weight for w_ν is so large in the paper that the trust region cost was never penalized sufficiently heavily to dip below the convergence threshold.

3.4.2 Virtual Controls

Artificial infeasibility can be encountered during the convergence process when the linearization is not favorable for feasibility. For example, if the problem is linearized about an unrealistically short time-of-flight, the linearized equations will likely not admit a feasible solution. Artificial infeasibility is encountered very frequently during the first few iterations of a typical successive convexification sequence, and is largely due to initiating the process with a poor initial guess. To mitigate this, we introduce a virtual control term $\vec{\nu}^i_k \in \mathbb{R}^{14}$, and add it to our discrete linearized dynamics to yield our final convex sub-problem dynamics:

$$\vec{x}_{k+1} = \bar{A}_k \vec{x}_k + \bar{B}_k \vec{u}_k + \bar{C}_k \vec{u}_{k+1} + \bar{\Sigma}_k \sigma + \bar{\vec{z}}_k + \vec{\nu}^i_k \,, \quad \forall k \in \bar{\mathcal{K}} \tag{47}$$

For notational convenience, we concatenate the $\vec{\nu}^i_k$ vectors into a larger vector $\bar{\vec{\nu}}^i \in \mathbb{R}^{14(K-1)}$, and use it to define a new cost term as follows:

$$c^i_\nu \doteq w_\nu \left\| \bar{\vec{\nu}}^i_k \right\|_1 \tag{48}$$

The intuition here is that $\vec{\nu}^i_k$ acts directly on the state variables when necessary to prevent artificial infeasibility, and selecting a large w_ν heavily penalizes the solver from relying on these virtual controls to maintain feasibility. When c^i_ν is negligible, this directly implies that the solution is dynamically feasible.

Since the presented cost term uses the 1-norm of the concatenated virtual controls, this requires us to again modify the formulation to be compatible with the SOCP format. I opted to stack the virtual control vectors side-by-side to make a matrix $\bar{\nu} \in \mathbb{R}^{14\times(K-1)}$, then introduced another auxiliary optimization variable $\bar{\nu}_{abs} \in \mathbb{R}^{14\times(K-1)}$ and its associated constraints:

$$\bar{\nu} \doteq \left[\vec{\nu}_0^i, \ldots, \vec{\nu}_{K-2}^i\right] \tag{49a}$$

$$-\bar{\nu}_{abs} \leq \bar{\nu} \leq \bar{\nu}_{abs} \tag{49b}$$

$$0 \leq \bar{\nu}_{abs} \tag{49c}$$

Now we can write our SOCP-compatible virtual control cost term as follows:

$$c_{\nu,aug}^i \doteq w_\nu \sum_{i=1}^{m}\sum_{j=1}^{n} \bar{\nu}_{abs,i,j} \tag{50}$$

3.5 Miscellaneous Constraints

The max tilt constraint (Eq. 20a) is convex, but it doesn't fit nicely into the SOCP framework because it is quadratic in the optimization variables. By shuffling terms around and taking the square root, we can obtain the following SOCP-compatible form:

$$\left\|H_q \vec{q}_{\mathcal{B}/\mathcal{I}}\right\|_2 \leq \sqrt{\frac{1-\cos\theta_{max}}{2}} \tag{51}$$

3.6 Problem Statement

We are now ready to summarize the convex sub-problem that will be solved repeatedly by the successive convexification algorithm. By virtue of the time normalization introduced in Section 3.2.1, this problem can be viewed as a fixed-final-time optimization problem due to the fact that the final normalized time is always equal to unity. As a consequence, t_f in Problem 1 is substituted with σ^i. Note that the i superscript denotes the i^{th} iterate of the overarching successive convexification loop.

Problem 2: Convex Discrete-Time Fixed-Final-Time Problem

$$\min_{\sigma^i, \vec{u}_k^i} \quad w_\sigma \sigma^i + w_\nu \bar{\nu}_{abs}^i + w_\Delta^i \bar{\Delta}_{norm}^i + w_{\Delta_\sigma} \Delta_{\sigma,abs}^i$$

subject to:

Dynamics:

$$\vec{x}_{k+1}^i = \bar{A}_k^i \vec{x}_k^i + \bar{B}_k^i \vec{u}_k^i + \bar{C}_k^i \vec{u}_{k+1}^i + \bar{\Sigma}_k^i \sigma^i + \bar{\vec{z}}_k^i + \vec{\nu}_k^i$$

State Constraints:

$$m_{dry} \leq m_k^i$$

$$\tan\gamma_{gs}\left\|H_{23}^{T}\vec{r}_{\mathcal{I},k}^{i}\right\|_2 \leq \vec{e}_1\cdot\vec{r}_{\mathcal{I},k}^{i}$$

$$\left\|H_q\vec{q}_{\mathcal{B}/\mathcal{I},k}^{i}\right\|_2 \leq \sqrt{\frac{1-\cos\theta_{max}}{2}}$$

$$\left\|\vec{\omega}_{\mathcal{B},k}^{i}\right\|_2 \leq \omega_{max}$$

$$\left\|\vec{q}_{\mathcal{B}/\mathcal{I},k}^{i}\right\|_2 \leq 1$$

$$(1-\epsilon_q) - \hat{\vec{q}}_{\mathcal{B}/\mathcal{I},k}^{i,T}\vec{q}_{\mathcal{B}/\mathcal{I},k}^{i} \leq 0$$

Control Constraints:

$$T_{min} - \frac{\hat{\vec{u}}_k^{i,T}\vec{u}_k^{i}}{\left\|\hat{\vec{u}}_k^{i}\right\|_2} \leq 0$$

$$\left\|\vec{u}_k^{i}\right\|_2 \leq T_{max}$$

$$\cos\delta_{max}\left\|\vec{u}_k^{i}\right\|_2 \leq \vec{e}_1\cdot\vec{u}_k^{i}$$

Boundary Conditions:

$$m_0^{i} = m_{wet}$$

$$\vec{r}_{\mathcal{I},0}^{i} = \vec{r}_{\mathcal{I},i}$$

$$\vec{v}_{\mathcal{I},0}^{i} = \vec{v}_{\mathcal{I},i}$$

$$\vec{q}_{\mathcal{B}/\mathcal{I},0}^{i} = \vec{q}_{\mathcal{B}/\mathcal{I},i}$$

$$\vec{\omega}_{\mathcal{B},0}^{i} = \vec{\omega}_{\mathcal{B},i}$$

$$\vec{r}_{\mathcal{I},K-1}^{i} = \vec{0}$$

$$\vec{v}_{\mathcal{I},K-1}^{i} = \vec{v}_{\mathcal{I},f}$$

$$\left\|H_q\vec{q}_{\mathcal{B}/\mathcal{I},K-1}^{i}\right\|_2 \leq \sqrt{\frac{1-\cos\theta_{max,f}}{2}}$$

$$\vec{\omega}_{\mathcal{B},K-1}^{i} = \vec{0}$$

$$\vec{e}_2\cdot\vec{u}_{K-1}^{i} = \vec{e}_3\cdot\vec{u}_{K-1}^{i} = 0$$

Auxiliary Constraints:

$$\sqrt{\delta\vec{x}_k^{i}\cdot\delta\vec{x}_k^{i} + \delta\vec{u}_k^{i}\cdot\delta\vec{u}_k^{i}} \leq \vec{e}_k\cdot\bar{\vec{\Delta}}^{i}$$

$$\left\|\bar{\vec{\Delta}}^{i}\right\|_2 \leq \bar{\Delta}_{norm}$$

$$-\Delta_{\sigma,abs} \leq \delta\sigma^{i} \leq \Delta_{\sigma,abs}$$

$$0 \leq \Delta_{\sigma,abs}$$

$$-\bar{\nu}_{abs} \leq \bar{\nu} \leq \bar{\nu}_{abs}$$

$$0 \leq \bar{\nu}_{abs}$$

4 Successive Convexification Algorithm

There are really two parts to the successive convexification algorith presented in the original paper. Since the technique relies on linearizing around a previous iterate, this means we'll need to define how to initialize the problem, i.e., how to define the first reference trajectory. Finally, we'll walk through the core loop that actually solves the problem in an iterative manner until convergence.

4.1 Initialization

One of the benefits of this paper's approach is that the initial guess doesn't have to be dynamically feasible. This allows us to define a simple initialization approach that practically just uses linear interpolation:

Algorithm 1 Initialization

Compute reference trajectory:
for $k \in \mathcal{K}$ **do**
  $\alpha_1 \leftarrow \frac{K-1-k}{K-1}$
  $\alpha_2 \leftarrow \frac{k}{K-1}$
  $m_k^0 \leftarrow \alpha_1 m_{wet} + \alpha_2 m_{dry}$
  $\vec{r}_{\mathcal{I},k}^{\,0} \leftarrow \alpha_1 \vec{r}_{\mathcal{I},i}$ ▷ No α_2 term because $\vec{r}_{\mathcal{I},f} = \vec{0}$
  $\vec{v}_{\mathcal{I},k}^{\,0} \leftarrow \alpha_1 \vec{v}_{\mathcal{I},i} + \alpha_2 \vec{v}_{\mathcal{I},f}$
  $\vec{q}_{\mathcal{B}/\mathcal{I},k}^{\,0} \leftarrow slerp(\vec{q}_{\mathcal{B}/\mathcal{I},i}, q_{id}, \alpha_2)$ ▷ Spherical interpolation, $q_{id} \doteq [1\,,0\,,\,0\,,0]^T$
  $\vec{\omega}_{\mathcal{B},k}^{\,0} = \vec{0}$
  $\vec{x}_k^{\,0} \leftarrow \left[m_k \; \vec{r}_{\mathcal{I},k}^{\,0,T} \; \vec{v}_{\mathcal{I},k}^{\,0,T} \; \vec{q}_{\mathcal{B}/\mathcal{I},k}^{\,0,T} \; \vec{\omega}_{\mathcal{B},k}^{\,0,T}\right]^T$
  $\vec{u}_k^{\,0} \leftarrow -m_k^0 C_{\mathcal{B}/\mathcal{I}} \vec{g}_{\mathcal{I}}$ ▷ Only supports current weight
end for
$\sigma^0 \leftarrow t_{f,guess}$
Compute discrete dynamics quantities:
for $k \in \bar{\mathcal{K}}$ **do**
  Compute $\bar{A}_k^0$, $\bar{B}_k^0$, $\bar{C}_k^0$, $\bar{\Sigma}_k^0$, and $\bar{\bar{z}}_k^0$ using method at end of Section 3.3
end for

4.2 Core Algorithm

Once we have our first guesses for the discrete states, controls, and dilation coefficient, we can begin the core successive convexification loop described below:

Algorithm 2 Successive Convexification Loop

for $i \in \{1\,, N_{max}\}$ **do**
  Solve Problem 2 using $\vec{x}_k^{\,i-1}$, $\vec{u}_k^{\,i-1}$, σ^{i-1}, $\bar{A}_k^{i-1}$, $\bar{B}_k^{i-1}$, $\bar{C}_k^{i-1}$, $\bar{\Sigma}_k^{i-1}$, $\bar{\bar{z}}_k^{i-1}$
  Store newly-computed $\vec{x}_k^{\,i}$, $\vec{u}_k^{\,i}$, σ^i
  if $\left\|\bar{\Delta}^i\right\|_2 \leq \Delta_{tol}$ and $\left\|\bar{\nu}^i\right\|_1 \leq \nu_{tol}$ **then**
    Exit
  else
    Compute $\bar{A}_k^0$, $\bar{B}_k^0$, $\bar{C}_k^0$, $\bar{\Sigma}_k^0$, and $\bar{\bar{z}}_k^0$ using method at end of Sec. 3.3
    Increment i, return to top of loop
  end if
end for

5 Demonstration

I provided a few demonstration cases in `demo.cpp`, namely a simple vertical landing case, an in-plane maneuver requiring a 90° turn, and a more complex out-of-plane maneuver requiring a twisting motion in 3D space. I'll skip discussion of the simple vertical landing case because it really just serves as a nice sanity check for solver unit tests.

You might be wondering if these examples will have anything to do with Mars, given the original paper title. Just like the paper, my examples won't use parameters specific to Mars because everything is non-dimensionalized and scaled for numerical stability. There isn't anything stopping this technique from being applied to a Mars landing problem, it just requires some future work (see Section 6).

5.1 In-plane Maneuver

This mimics the in-plane maneuver presented in the original paper. The vehicle is approaching the landing site horizontally with significant speed and must simultaneously rotate in-plane and decelerate to land vertically. Figure 2 shows the vehicle trajectory in the Up-East plane. Note that just like the original paper, the entire problem works with non-dimensionalized quantities, which is why the length scales are so small. To be clear, the visualized rocket body length and engine plume length are scaled purely for visualization purposes.

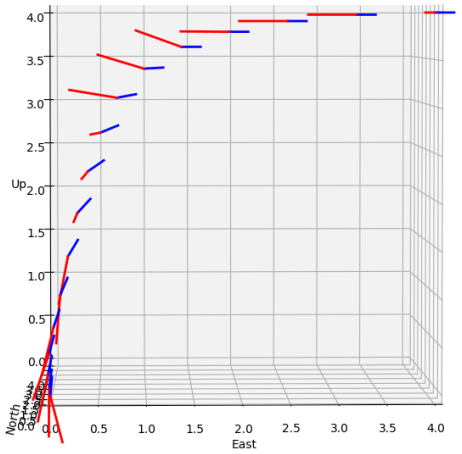


Figure 2: Vehicle path in the planar maneuver case. Blue is the rocket body (not to scale), red is the engine exhaust plume (not to scale).

The vehicle maintains its horizontal orientation for as long as possible so it can purely shave off lateral speed, then gimbals the engine to pivot the vehicle to its upright position while falling towards the landing site before finally firing at max thrust at the end to perform a soft landing. The descent profile is illustrated in Figure 3.

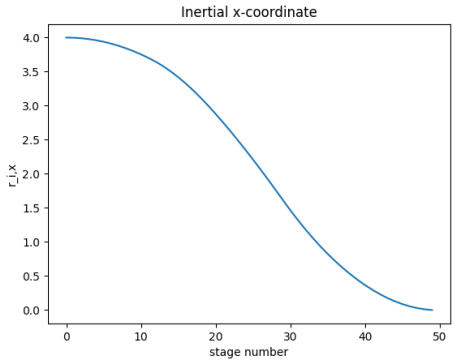


Figure 3: Inertial x-coordinate (non-dimensional altitude) over the trajectory.

We can see that the vehicle rides the 90° tilt angle limit during the first part of the trajectory in Figure 4, then takes advantage of my tilt limit constraint at the end to avoid the effort of landing perfectly vertically. Bang-bang control is allowed in this problem formulation, and since we're minimizing time of flight, the solver takes advantage of this in Figure 5. It also maxes out the body z-axis angular rate limit to make the maneuver as quickly as possible, illustrated in Figure 6. Finally, we can see that the time-of-flight estimate converges within

a small number of iterations in Figure 7.

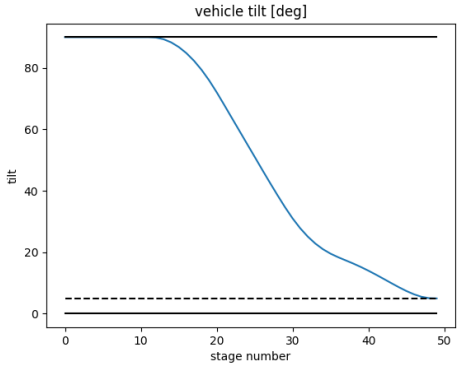


Figure 4: Vehicle tilt in degrees over the trajectory.

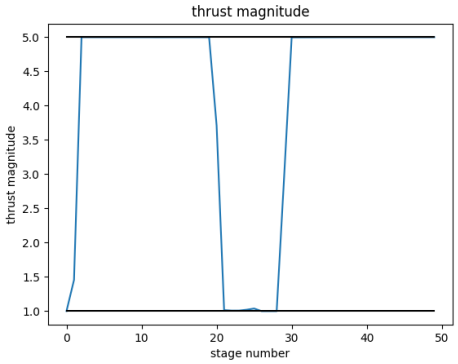


Figure 5: Non-dimensional thrust magnitude over the trajectory.

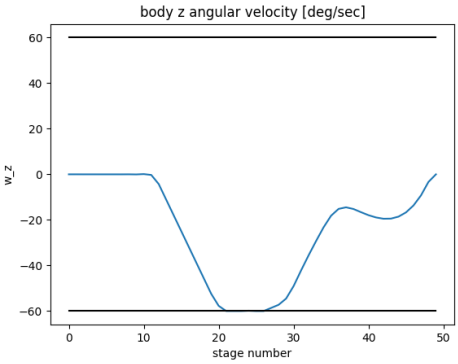


Figure 6: Body z-axis angular rate in degrees over the trajectory.

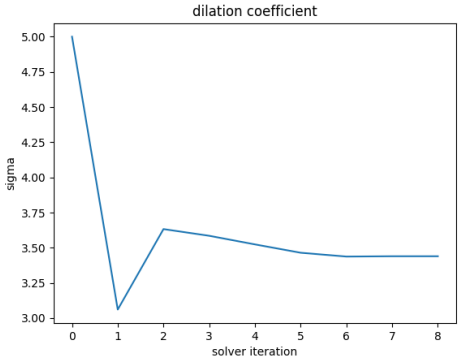


Figure 7: Dilation coefficient estimate over successive convexification iterations.

5.2 Out-of-plane Maneuver

This example forces a more interesting 3D maneuver down to the landing site. The vehicle has to rapidly decelerate, but it also has to re-orient itself and translate over to the landing site over the course of the landing burn. It again starts horizontal with substantial speed, but it's not pointing exactly in the direction of the landing site. The twisting maneuver down to the landing site is shown in Figure 8.

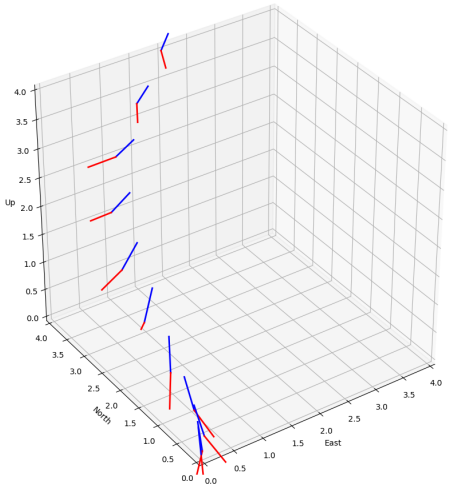


Figure 8: Out-of-plane landing maneuver. Blue is the rocket body (not to scale), red is the engine exhaust plume (not to scale).

The solution first gimbals heavily to re-orient the rocket body while still coasting with substantial speed, then swings the engine back to push it laterally towards the landing site, before finally swinging the engine back the other

direction to cancel out body angular velocities and land vertically at the origin. The descent profile is illustrated in Figure 9.

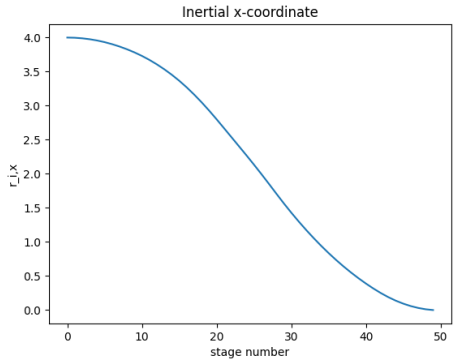


Figure 9: Inertial x-coordinate (non-dimensional altitude) over the trajectory.

We can see that it again uses bang-bang control and saturates the thrust limits for most of the flight in Figure 8. For this particular problem, it doesn't need to saturate the body angular rate limits as shown in Figure 11, but it does ride the 90 deg tilt limit again for the first section of the trajectory in Figure 12, then again takes advantage of my tilt limit constraint at the end to avoid the effort of landing perfectly vertically. Finally, we can see that the dilation coefficient estimate still converges within a small number of successive convexification iterations in Figure 13.

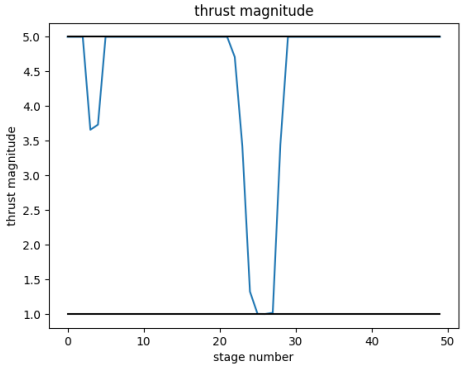


Figure 10: Non-dimensional thrust magnitude over the trajectory.

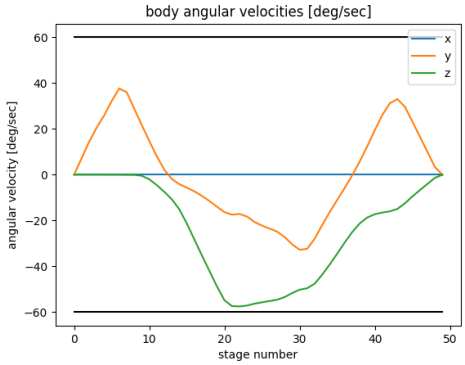


Figure 11: Body angular rates in degrees over the trajectory.

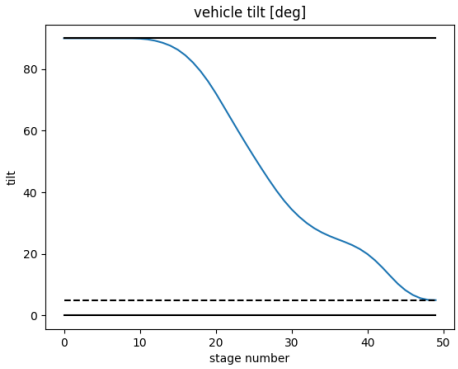


Figure 12: Vehicle tilt in degrees over the trajectory.

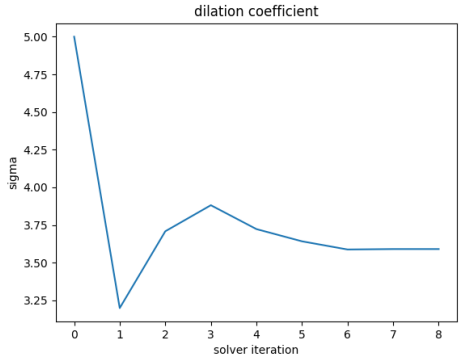


Figure 13: Dilation coefficient estimate over successive convexification iterations.

6 Future Work

For numerical stability purposes, this implementation requires all problem quantities to be non-dimensionalized and appropriately scaled to prevent magnitude differences of many orders of magnitude across optimization quantities. Ideally, this library would take in arbitrary vehicle configurations and boundary conditions and perform all required non-dimensionalization and scaling under the hood for the user.

I currently compute the discrete dynamics quantities (37b through 37f) serially, which consumes a non-trivial percentage of the cycle time. However, given that this calculation process starts at each discrete reference state and integrates forward using controls calculated from Eq. 33a, these quantities can and should be computed in parallel across discretization stages. Further profiling is required to determine other performance bottlenecks to focus on.

References

[1] Michael Szmuk and Behçet Açıkmeşe. Successive convexification for 6-dof mars rocket powered landing with free-final-time. 2018. https://arxiv.org/pdf/1802.03827.

[2] Danylo Malyuta, Taylor P. Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behçet Açıkmeşe. Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently. 2021. https://arxiv.org/pdf/2106.09125.

[3] Michael Szmuk. *Successive Convexification & High Performance Feedback Control for Agile Flight*. PhD thesis, University of Washington, 2019.

[4] Sven Niederberger. https://github.com/EmbersArc/SCpp.