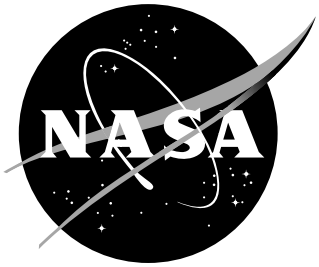
NASA/TM–20230009811



Tutorial: MATLAB Implementation of a Successive Convexification Algorithm for 3 DoF Rocket Landings

Alex Hayes, Jing Pei, Zachary May
NASA Langley Research Center, Hampton, Virginia

February 2025

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

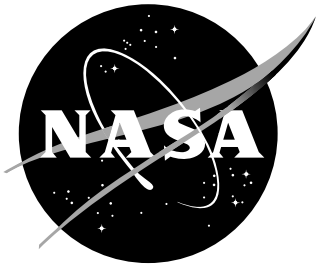
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.
- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov
- E-mail your question to help@sti.nasa.gov
- Phone the NASA STI Information Desk at 757-864-9658
- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM–20230009811



Tutorial: MATLAB Implementation of a Successive Convexification Algorithm for 3 DoF Rocket Landings

Alex Hayes, Jing Pei, Zachary May
NASA Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

NASA Langley Research Center
Hampton, Virginia 23681-2199

February 2025

Available from:

NASA STI Program / Mail Stop 050
NASA Langley Research Center
Hampton, VA 23681-2199

Abstract

The primary objective of this work is to fill in gaps and explore an alternate way of solving the 3 DoF rocket-powered landing problem presented in the 2016 AIAA paper by Szmuk, Ackimese, and Berning using successive convexification (SCvx). In the original paper, CVX, an automatic parsing package, was used to transcribe the high-level trajectory optimization problem into a format that could be read by a conic solver. The parsing step, generally computationally intensive, is hidden from the user. The use of CVX is sufficient for the generation of trajectories off-line due to the lack of runtime and flight software implementation constraints. For on-line applications, it is necessary to parse the problem for flight software implementation. References on hand-parsing powered descent guidance (PDG) problems are sparse. In this Tech Memo, the process of transcribing the 3 DoF PDG problem into the format required by MATLAB's built-in second-order cone solver, *coneprog.m*, is presented in detail. Due to the abridged 3 DoF dynamics and the relatively simple nonlinearities, this reference is the natural starting point for anyone interested in grasping the concepts behind SCvx pertaining to PDG and the parsing step. Simulation results shown in this report were independently created by solving the problem using *coneprog.m*. The intent of this memo is to serve as a supplemental material to the original paper by breaking down the concept behind successive convexification and shed light into the parsing process. Readers are encouraged to first familiarize themselves with the material laid out in the original reference.

Contents

Nomenclature . . . **3**

1 Introduction . . . **4**
1.1 Purpose . . . 4
1.2 Second Order Cone Programming in MATLAB . . . 4
1.3 Global LTV Dynamics . . . 5

2 3 DoF Rocket Dynamics . . . **8**

3 First Iteration . . . **10**
3.1 Dynamics . . . 10
3.2 Optimization Problem . . . 14
3.2.1 Boundary Condition Constraints . . . 15
3.2.2 State Constraints . . . 17
3.2.3 Control Constraints . . . 18
3.2.4 Objective Function Constraints . . . 20

4 Successive Iterations . . . **22**
4.1 Dynamics . . . 22
4.2 Optimization Problem . . . 25
4.2.1 Boundary Condition Constraints . . . 26
4.2.2 Control Constraints . . . 26
4.2.3 Objective Function Constraints . . . 26
4.2.4 Trust Region Constraints . . . 27

5 Results . . . **29**

6 Comparison with Quickshot^{TM} . . . **34**

7 Comparison with E-guidance . . . **36**

8 Conclusion . . . **37**

9 Acknowledgements . . . **37**

10 Appendix: Partial Derivatives . . . **38**

References . . . **41**

Selected Nomenclature

Symbols

$\tilde{\boldsymbol{x}}$	Vector that contains all optimization variables
$\mathbf{X}$	State vector for the entire discrete trajectory
$\mathbf{T}$	Thrust, N
$\mathbf{a}_R$	Synthetic acceleration, m/s^2
Γ	Thrust magnitude slack variable
$\kappa_{\mathbf{a},R}$	Synthetic acceleration slack variable
$\boldsymbol{\kappa}_{\mathbf{a},R}$	Vector containg synthetic acceleration slack variables
Δt	Discretized sample time, s
N	Number of temporal nodes
M, Υ	Matrices that pull out elements of a vector
γ	Glidescope angle, deg
θ	Vehicle pointing angle with respect to vertical, deg
Ψ	Generic representation of the state and control values at nodes k and $k+1$
η	Trust region variables
w	Cost function weights
$\mathbf{1}$	Identity matrix of the appropriate dimension
n_x	Number of states
n_u	Number of inputs
n_p	Number of pseudo inputs
$\|\|\mathbf{x}\|\|$	2-norm of a generic vector

Subscripts

i	i$^{\text{th}}$ iteration
k_f	Final node

Superscripts

-1	Inverse
T	Transpose

Acronyms

LCvx	Lossless Convexification
SCvx	Successive Convexification
SCP	Sequential Convex Programming
SOCP	Second-order Cone Program
LTV	Linear Time Varying
gs	Glidescope

1 Introduction

1.1 Purpose

The seminal paper by Szmuk, Acikmese, and Berning in 2016 [1] on the application of successive convexification (SCvx) to the 3 DoF rocket powered landing problem introduced a new paradigm in the field of real-time trajectory optimization. Reference [1] builds off the theory behind lossless convexification (LCvx) and uses sequential convex programming (SCP) for trajectory optimization in the presence of nonlinearities and non-convex constraints. The SCvx method is far more general compared to LCvx and solves the discretized convex problem in the local neighborhood where the linearization is accurate. The problem is re-linearized about the new solution and the process is repeated until a set of stopping criteria has been met [2]. Ensuing publications over the past 6+ years have extended the SCvx algorithm outlined in Ref. [1] to include rotational dynamics and additional constraints.

Many major aerospace companies including SpaceX, Blue Origin and Astrobotics have already embraced leveraging the power behind convex optimization. Since 2015, SpaceX has relied on high-speed onboard convex optimization algorithms for the Falcon 9 booster landings [3]. Similiarly, Blue Origin is looking at using lossless convexification and dual quaternion guidance for lunar powered descent guidance [4,5]. Despite the disruptive and emerging nature of the SCvx algorithm for the purpose of trajectory generation, it is still an active area of research [2, 6]. Furthermore, expertise in this area within NASA appears to be sparse at best.

Due to the abridged 3 degree-of-freedom (DoF) dynamics and the relatively simple nonlinearities associated with the aerodynamic drag term and mass depletion, Ref. [1] is the natural starting point for anyone interested in grasping the concepts and procedures involved with SCvx and its application to trajectory generation for powered descent. The primary objective of this work is to provide details into the successive convexification procedure and shed light onto the parsing process which involves transcribing the high-level trajectory optimization problem, involving multiple equality, inequality, and second-order cone constraints, into a standard format that can be fed to the conic solver. This is step is generally hidden to the user when using an automatic parsing package such as CVX. A MATLAB implementation of the problem using the built-in *coneprog.m* function is available to interested readers. The overall intent is to share knowledge and foster research/development in this field within the NASA GN&C community. Note: the intent of this TM is to serve as supplemental material to Ref. [1]. Readers are encouraged to first familiarize themselves with the concepts presented in Ref. [1]. Nomenclature used in this TM is consistent with Ref. [1] where applicable.

The TM is organized as follows: the remainder of Sec. 1 provides background information on the structure of the second-order cone programming (SOCP) solver in MATLAB and propagation of a general linear-time varying (LTV) discrete-time system. Section 2 provides detailed descriptions of the 3 DoF continuous- and discrete-time dynamics described in Problems 1–3 of Ref. [1]. Section 3 covers details involving the first iteration, or Problem 4 of the SCvx algorithm. Section 4 addresses the detailed formulation and implementation of the optimal control problem in subsequent iterations, or Problem 5. Independent results of the example provided in Ref. [1] are shown in Sec. 5 for verification purposes. Section 6 provides comparison between SCP solution with QuickShot$^{\text{TM}}$, a commercial trajectory optimization software, based on nonlinear programming methods. Section 7 provides a comparison between the SCP solution and the polynomial E-guidance used during the approach phase on the Apollo missions.

1.2 Second Order Cone Programming in MATLAB

As described in Ref. [1], the discretized parameter optimization problem including the dynamics along with the various state, control, and trust region constraints are formulated as a sequence of SOCP. There are many software packages available for solving SOCP problems such as SDPT3 [7], SeDuMi [8] and MOSEK [9]. The use of a parser such as CVX [10] can then be used as an interface allowing these solvers to be used within the MATLAB framework. MATLAB version R2020b introduced the function *coneprog.m* [11] as part of the Optimization Toolbox which enables SOCP optimization problems to be solved in MATLAB, using

built-in capabilities, without the need for third-party software. While the use of *coneprog.m* avoids the need to use additional software within MATLAB, the trajectory optimization problem must be hand-parsed into a standard form.

In order to use the *coneprog.m* solver, the optimization problem must conform to the format specified in the function documentation. The solver minimizes the objective function,

$$\min_{\tilde{\mathbf{x}}} \mathbf{f}^T \tilde{\mathbf{x}}, \tag{1}$$

where $\tilde{\mathbf{x}}$ is a vector containing all of the variables over which the objective function is optimized and $\mathbf{f}$ is a vector of the same dimension as $\tilde{\mathbf{x}}$ that encodes the cost associated with a particular solution. The objective function is minimized subject to second-order cone (SOC) constraints, linear equality, and inequality inequality constraints while also ensuring that the optimization variables are within specified lower and upper bounds. The SOC constraints must of be the form,

$$||\mathbf{A}_{\text{sc}}(i)\tilde{\mathbf{x}} - \mathbf{b}_{\text{sc}}(i)|| \leq \mathbf{d}_{\text{sc}}(i)^T \tilde{\mathbf{x}} + \gamma_{\text{sc}}(i), \tag{2}$$

while the inequality constraints is expressed as,

$$\mathbf{A}_{\text{ineq}}\tilde{\mathbf{x}} \leq \mathbf{b}_{\text{ineq}}. \tag{3}$$

Similarly, the equality constraint takes the form of,

$$\mathbf{A}_{\text{eq}}\tilde{\mathbf{x}} = \mathbf{b}_{\text{eq}}, \tag{4}$$

and the lower and upper bounds on the optimization vector are expressed as,

$$\tilde{\mathbf{x}}_{\min} \leq \tilde{\mathbf{x}} \leq \tilde{\mathbf{x}}_{\max}, \tag{5}$$

where $\tilde{\mathbf{x}}_{\min}$ and $\tilde{\mathbf{x}}_{\max}$ are vectors containing the lower and upper bounds on the optimization variables, respectively.

While *coneprog.m* is capable of accepting an array containing multiple SOC constraints, multiple individual equality or inequality constraints must be stacked to form a single constraint of each type. For example, a set of N_{eq} equality constraints are arranged as

$$\begin{bmatrix} \mathbf{A}_{\text{eq},1} \\ \vdots \\ \mathbf{A}_{\text{eq},N_{\text{eq}}} \end{bmatrix} \tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{b}_{\text{eq},1} \\ \vdots \\ \mathbf{b}_{\text{eq},N_{\text{eq}}} \end{bmatrix} \tag{6}$$

to form a single equality constraint of the form $\mathbf{A}_{\text{eq}}\tilde{\mathbf{x}} = \mathbf{b}_{\text{eq}}$.

The primary purpose of this tutorial is to describe the process of hand parsing, or manipulating the dynamics and constraints, as described in Ref. [1], into the form required by *coneprog.m* described in this section and shed light onto the process of successive convexification. The algorithm in Ref. [1] solves two separate optimization problems: one for the first iteration, referred to as Problem 4, and another for all subsequent iterations, referred to as Problem 5. As such, Section 3 and Section 4 describe the process of solving each optimization problem in MATLAB using *coneprog.m*.

1.3 Global LTV Dynamics

An important step in the discretized parameter optimization problem is the formulation of the linear time varying (LTV) global dynamics. This detail was left out in Ref. [1]. In order to generate an optimal sequence of inputs and trajectory, the entire trajectory of the full state vector must be expressible as a function of the initial condition and the optimization variables. Beginning with a discrete-time linear model of the dynamics of the form,

$$\mathbf{x}[k+1] = \mathbf{A}[k]\mathbf{x}[k] + \mathbf{B}[k]\mathbf{u}[k] + \mathbf{F}[k]\mathbf{p}[k] + \mathbf{w}[k], \tag{7}$$

where k identifies the temporal node in the discrete-time dynamics, $\mathbf{A}$ and $\mathbf{B}$ are the discrete time matrices related to the linearized dynamics and the controls. The vector $\mathbf{u}[k]$ contains the physical control inputs involved in propagating the state from node k to node $k+1$ and depends on how the continuous-time control is paramerized in discrete time (e.g., a zero-order hold or a first-order hold). The vector $\mathbf{p}$ is similar to $\mathbf{u}$ and represents pseudo control terms such as synthetic acceleration which is penalized in the cost function. $\mathbf{w}$ represent any constant disturbances that affects the dynamics. A dynamic model of this form is derived in Sec. 3 for the first iteration and Sec. 4 for all subsequent iterations. The global expression of the dynamics is derived by first propagating the initial condition through Eq. (7) to obtain the state at $k = 1$

$$\mathbf{x}[1] = \mathbf{A}[0]\mathbf{x}[0] + \mathbf{B}[0]\mathbf{u}[0] + \mathbf{F}[0]\mathbf{p}[0] + \mathbf{w}[0] \tag{8}$$

Subsequently, propagate from $k = 1$ to $k = 2$ by substituting the expression for $k = 1$ shown in Eq. (8)

$$\begin{aligned}
\mathbf{x}[2] =& \mathbf{A}[1]\mathbf{x}[1] + \mathbf{B}[1]\mathbf{u}[1] + \mathbf{F}[1]\mathbf{p}[1] + \mathbf{w}[1] \\
=& \mathbf{A}[1]\left(\mathbf{A}[0]\mathbf{x}[0] + \mathbf{B}[0]\mathbf{u}[0] + \mathbf{F}[0]\mathbf{p}[0] + \mathbf{w}[0]\right) \\
&+ \mathbf{B}[1]\mathbf{u}[1] + \mathbf{F}[1]\mathbf{p}[1] + \mathbf{w}[1] \\
=& \mathbf{A}[1]\mathbf{A}[0]\mathbf{x}[0] + \mathbf{A}[1]\mathbf{B}[0]\mathbf{u}[0] + \mathbf{A}[1]\mathbf{F}[0]\mathbf{p}[0] \\
&+ \mathbf{A}[1]\mathbf{w}[0] + \mathbf{B}[1]\mathbf{u}[1] + \mathbf{F}[1]\mathbf{p}[1] + \mathbf{w}[1].
\end{aligned} \tag{9}$$

Propagate once more to obtain the expression for $\mathbf{x}$ at $k = 3$

$$\begin{aligned}
\mathbf{x}[3] =& \mathbf{A}[2]\mathbf{x}[2] + \mathbf{B}[2]\mathbf{u}[2] + \mathbf{F}[2]\mathbf{p}[2] + \mathbf{w}[2] \\
=& \mathbf{A}[2]\left(\mathbf{A}[1]\mathbf{A}[0]\mathbf{x}[0] + \mathbf{A}[1]\mathbf{B}[0]\mathbf{u}[0] + \mathbf{A}[1]\mathbf{F}[0]\mathbf{p}[0]\right. \\
&\left. + \mathbf{A}[1]\mathbf{w}[0] + \mathbf{B}[1]\mathbf{u}[1] + \mathbf{F}[1]\mathbf{p}[1] + \mathbf{w}[1]\right) \\
&+ \mathbf{B}[2]\mathbf{u}[2] + \mathbf{F}[2]\mathbf{p}[2] + \mathbf{w}[2] \\
=& \mathbf{A}[2]\mathbf{A}[1]\mathbf{A}[0]\mathbf{x}[0] + \mathbf{A}[2]\mathbf{A}[1]\mathbf{B}[0]\mathbf{u}[0] \\
&+ \mathbf{A}[2]\mathbf{A}[1]\mathbf{F}[0]\mathbf{p}[0] + \mathbf{A}[2]\mathbf{A}[1]\mathbf{w}[0] + \mathbf{A}[2]\mathbf{B}[1]\mathbf{u}[1] \\
&+ \mathbf{A}[2]\mathbf{F}[1]\mathbf{p}[1] + \mathbf{A}[2]\mathbf{w}[1] + \mathbf{B}[2]\mathbf{u}[2] + \mathbf{F}[2]\mathbf{p}[2] + \mathbf{w}[2]
\end{aligned} \tag{10}$$

Subsequently, the same procedure can be carried out to obtain the entire time history of $\mathbf{x}$ from $k = 1$ to $k = k_f$. Equation (11) is a compact representation of the LTV global dynamics. The natural or free-response is represented by the first term on the RHS, subsequent terms denotes the forced responses. The various terms in Eq. (11) are expanded in Eqs. (12) – (17).

$$\mathbf{X} = \mathbf{M}\mathbf{x}[0] + \mathbf{S}_1\mathbf{U} + \mathbf{S}_2\mathbf{P} + \mathbf{S}_3\mathbf{W}, \tag{11}$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}[1] \\ \vdots \\ \mathbf{x}[N-1] \end{bmatrix} \in \mathbb{R}^{n_x(N-1)\times 1}, \tag{12}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}[0] \\ \vdots \\ \mathbf{u}[N-2] \end{bmatrix} \in \mathbb{R}^{n_u(N-1)\times 1}, \tag{13}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}[0] \\ \vdots \\ \mathbf{p}[N-2] \end{bmatrix} \in \mathbb{R}^{n_p(N-1)\times 1}, \tag{14}$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}[0] \\ \vdots \\ \mathbf{w}[N-2] \end{bmatrix} \in \mathbb{R}^{n_x(N-1)\times 1}. \tag{15}$$

and the global dynamic matrices take the form

$$\mathbf{M} = \begin{bmatrix} \mathbf{A}[0] \\ \mathbf{A}[1]\mathbf{A}[0] \\ \mathbf{A}[2]\mathbf{A}[1]\mathbf{A}[0] \\ \vdots \\ \prod_{j=0}^{N-2} \mathbf{A}[k] \end{bmatrix} \in \mathbb{R}^{n_x(N-1)\times n_x} \tag{16}$$

and

$$\mathbf{S}_1 = \begin{bmatrix} \mathbf{B}[0] & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}[1]\mathbf{B}[0] & \mathbf{B}[1] & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}[2]\mathbf{A}[1]\mathbf{B}[0] & \mathbf{A}[2]\mathbf{B}[1] & \mathbf{B}[0] & & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \left(\prod_{j=1}^{N-2} \mathbf{A}[j]\right)\mathbf{B}[0] & \left(\prod_{j=2}^{N-2} \mathbf{A}[j]\right)\mathbf{B}[1] & \left(\prod_{j=3}^{N-2} \mathbf{A}[j]\right)\mathbf{B}[2] & \cdots & \mathbf{B}[N-2] \end{bmatrix}, \tag{17}$$

and $\mathbf{S}_2 \in \mathbb{R}^{n_x(N-1)\times n_p(N-1)}$ and $\mathbf{S}_3 \in \mathbb{R}^{n_x(N-1)\times n_x(N-1)}$ are obtained by swapping out $\mathbf{B}[k]$ in Eq. (17) for $\mathbf{F}[k]$ and $\mathbf{1}$ respectively.

2 3 DoF Rocket Dynamics

The heart of solving optimization-based trajectory planning problems is the theory of optimal controls as described by Problem 4 of Ref. [6]. The goal is to find a set of inputs that minimizes a cost function while satisfying the system dynamics and various constraints. Hence, the dynamics of the vehicle must be embedded into the optimization problems. However, continuous-time dynamics result in an infinite-dimensional optimization problem which is not tractable to be implemented on a computer. Therefore, the trajectory of the vehicle must be discretized into a finite number of temporal nodes and Problem 4 in Ref. [6] is solved as a parameter optimization exercise via numerical optimization. This necessitates the use of discrete-time dynamics to propagate the vehicle state from one node to the next. In order to obtain the discrete-time dynamics of the rocket, first the continuous-time trajectory of the rocket is divided into N temporal nodes connected by $N-1$ time segments of length Δt, the discretized sample time. These nodes are identified by their index, $k \in [0, k_f]$, where $k = 0$ is the initial time of the trajectory and $k_f = N - 1$ represents the end of the trajectory.

The continuous-time 3 DoF dynamics are described in Problem 1 of Ref. [1]. While the zeroth-order-hold is frequently used in the discretization of continuous-time systems, Ref. [1] makes use of a first-order approximation on the acceleration of the vehicle in order to improve the fidelity of the discrete-time dynamic equations. At time t, the acceleration of the rocket is expressed as,

$$\mathbf{a}(t) = \frac{1}{m(t)}\left(\mathbf{T}(t) - \frac{1}{2}\rho S_D C_D ||\mathbf{v}(t)||\mathbf{v}(t)\right) + \mathbf{g}, \tag{18}$$

where m is the vehicle mass, $\mathbf{T}$ is the thrust vector, ρ is the density of the atmosphere, S_D and C_D are the vehicle cross-sectional area and drag coefficient, respectively, $\mathbf{v}$ is the vehicle velocity and $\mathbf{g}$ is the acceleration due to gravity. This expression of the acceleration accounts for a rocket with time-varying mass being acted on by engine thrust, drag, and gravity. Note: Ref. [1] uses a crude approximation of C_D by assuming the rocket is a sphere which makes incorporation of aerodynamic drag into the 3 DoF dynamics tractable.

The first-order approximation assumes that, at some time $t \in [0, \Delta t]$ seconds after t_k but before $t_{k+1} = t_k + \Delta t$ (the times that mark the beginning and end of the k^{th} time segment of length Δt), the acceleration can be found through,

$$\mathbf{a}(t_k + t) \approx \frac{\mathbf{a}\left(t_{k+1}\right) - \mathbf{a}\left(t_k\right)}{\Delta t} t + \mathbf{a}\left(t_k\right). \tag{19}$$

Integrating this expression gives the change in velocity resulting in,

$$\begin{aligned}\mathbf{v}(t_k + t) &= \mathbf{v}(t_k) + \int_0^t \left[\frac{\mathbf{a}\left(t_{k+1}\right) - \mathbf{a}\left(t_k\right)}{\Delta t}\tau + \mathbf{a}\left(t_k\right)\right] d\tau \\ &= \mathbf{v}(t_k) + \frac{1}{2}\frac{\mathbf{a}\left(t_{k+1}\right) - \mathbf{a}\left(t_k\right)}{\Delta t} t^2 + \mathbf{a}\left(t_k\right) t,\end{aligned} \tag{20}$$

which describes the velocity during the k^{th} time segment. Integrating the velocity gives the change in position such that,

$$\begin{aligned}\mathbf{r}\left(t_k + t\right) &= \mathbf{r}(t_k) + \int_0^t \left[\frac{1}{2}\frac{\mathbf{a}\left(t_{k+1}\right) - \mathbf{a}\left(t_k\right)}{\Delta t}\tau^2 + \mathbf{a}\left(t_k\right)\tau + \mathbf{v}(t_k)\right] d\tau \\ &= \frac{1}{6}\frac{\mathbf{a}\left(t_{k+1}\right) - \mathbf{a}\left(t_k\right)}{\Delta t} t^3 + \frac{1}{2}\mathbf{a}\left(t_k\right) t^2 + \mathbf{v}(t_k) t + \mathbf{r}\left(t_k\right),\end{aligned} \tag{21}$$

which similarly describes the evolution of the vehicle position. Substituting $t = \Delta t$ to propagate the state from t_k to $t_{k+1} = t_k + \Delta t$ gives

$$\mathbf{v}(t_{k+1}) = \mathbf{v}(t_k + \Delta t) = \frac{1}{2}\left(\mathbf{a}\left(t_k\right) + \mathbf{a}\left(t_{k+1}\right)\right)\Delta t + \mathbf{v}\left(t_k\right) \tag{22}$$

while repeating the same process with Eq. (21) gives

$$\mathbf{r}\left(t_{k+1}\right) = \mathbf{r}(t_k + \Delta t) = \frac{1}{3}\left(\mathbf{a}\left(t_k\right) + \frac{1}{2}\mathbf{a}\left(t_{k+1}\right)\right)\Delta t^2 + \mathbf{v}\left(t_k\right)\Delta t + \mathbf{r}\left(t_k\right). \tag{23}$$

Substituting the notation $x[k] = x(t_k)$ gives

$$\mathbf{v}[k+1] = \frac{1}{2}\left(\mathbf{a}[k] + \mathbf{a}[k+1]\right)\Delta t + \mathbf{v}[k], \tag{24}$$

and

$$\mathbf{r}[k+1] = \frac{1}{3}\left(\mathbf{a}[k] + \frac{1}{2}\mathbf{a}[k+1]\right)\Delta t^2 + \mathbf{v}[k]\Delta t + \mathbf{r}[k], \tag{25}$$

where

$$\mathbf{a}[k] = \frac{1}{m[k]}\left(\mathbf{T}[k] - \frac{1}{2}\rho S_D C_D ||\mathbf{v}[k]||\mathbf{v}[k]\right) + \mathbf{g}, \tag{26}$$

which will serve as the discrete-time dynamic model used to optimize the eventual trajectory. It should be noted that these dynamics are nonlinear due to the time-varying mass term, as well as the squared velocity in the formulation of the drag term. Additionally, as this is a free final time optimization problem, products of Δt also introduce nonlinearities.

The mass of the vehicle is considered as a state along with position and velocity, and is depleted at a rate dependent on the commanded thrust magnitude. In continuous time, the mass-depletion dynamics are given by

$$\dot{m}(t) = -\alpha||\mathbf{T}(t)|| - \dot{m}_{bp}. \tag{27}$$

The thrust-dependent fuel depletion rate is given by,

$$\alpha \triangleq \frac{1}{I_{sp} g_0}, \tag{28}$$

where I_{sp} is the specific impulse of the rocket engine and g_0 is standard gravity. The term that compensates for back-pressure losses due to the engine operating in atmosphere is found through,

$$\dot{m}_{bp} \triangleq \frac{P_{amb} A_{nozzle}}{I_{sp} g_0}, \tag{29}$$

where P_{amb} is the ambient pressure, which is assumed constant, and A_{nozzle} is the area at the exit of the engine nozzle.

The thrust is interpolated linearly between temporal nodes. As a result, throughout a time segment, mass will be depleted at a rate equal to the average of the rate at the beginning and end of a time segment, or,

$$\dot{m}_{avg}[k] = \frac{\dot{m}[k] + \dot{m}[k+1]}{2} = -\left[\frac{\alpha}{2}\left(||\mathbf{T}[k]|| + ||\mathbf{T}[k+1]||\right) + \dot{m}_{bp}\right]. \tag{30}$$

Integrating this rate over a time segment leads to the discrete mass-depletion dynamics given by,

$$m[k+1] = m[k] - \left[\frac{\alpha}{2}\left(||\mathbf{T}[k]|| + ||\mathbf{T}[k+1]||\right) + \dot{m}_{bp}\right]\Delta t. \tag{31}$$

The discrete time dynamics presented here can also be seen in Problem 2 of Ref. [1]. Subsequently, in Problem 3 lossless convexification was applied to the non-convex thrust magnitude constraint and the slack variable Γ was introduced. In order to utilize these dynamics within a convex optimization scheme, they must first be linearized. The linearization is accomplished differently between the first and subsequent iterations corresponding to Problems 4 and 5 in Ref. [1], which will be discussed in Sec. 3 and 4.

3 First Iteration

In the absence of an a-priori estimate of the trajectory, the initial iteration in the trajectory optimization problem solves Problem 4 as described in Ref. [1]. This chapter describes, in detail, the process of implementing the Problem 4 optimization problem in MATLAB using the *coneprog.m* solver.

While the equations of motion in Eqs. (24)-(26) and (31) are in discrete time, which is required to propagate the states of the vehicle from one node to another, they are nonlinear and cannot be incorporated into a convex optimization format without modification. The typical approach would be to linearize these equations about a reference trajectory. However, for the first iteration of the optimization algorithm, no such trajectory is available. Instead, the nonlinearities are removed by assuming linear variations in the mass and $\|\mathbf{v}\|$ with a guess in the time-of-flight as described in Sec. III.C.4 of Ref. [1] which are denoted by the terms $\mu[k]$ and $s[k]$ respectively.

These simplifications result in a linear discrete-time representation of the dynamics. However, this model relies on a guess of the final time, which can result in infeasibility as the vehicle may not be able to reach the desired final state in the allotted time due to state and control constraints. Artificial infeasibility can also arise due to the use of a linearization to approximate the dynamics. In order to avoid infeasibility, the dynamics are relaxed through the addition of a synthetic acceleration, $\mathbf{a}_R[k]$, which can push the vehicle toward the desired final state.

The first iteration yields a set of thrust commands and the resulting trajectory that takes the rocket to the desired terminal conditions. However, as a result of the guess on the time-of-flight and the potentially non-zero synthetic acceleration, the resultant trajectory may not be dynamically feasible. Furthermore, due to the assumptions and simplifications made in linearizing the dynamics, the optimal thrust commands will not guide the vehicle along the desired trajectory when propagated through the nonlinear dynamics.

The trajectory obtained in the first iteration serves as a reference about which the dynamics will be linearized about during the second iteration. Thereafter, the final time of the trajectory will be freed and included as an optimization variable in order to obtain the optimal final time and a dynamically feasible trajectory by penalizing the use of synthetic acceleration in the cost function.

3.1 Dynamics

Linearization and Convexification Equations (24)-(26) and (31) represent the discrete-time nonlinear dynamics of the system. However, due the nonlinearities they cannot be restructured into a format required by a convex optimization solver. One source of nonlinearity is the presence of Δt, which is an optimization variable for the free-final-time problem. This nonlinearity is removed in the first iteration by replacing Δt with $\Delta\tau$, a constant, hence converting the free-final-time problem to a fixed-final-time problem. The value of $\Delta\tau$ comes from an initial guess of the time of flight, t_f, using the following relationship

$$\Delta\tau = \frac{t_f}{N-1}. \tag{32}$$

Furthermore, as discussed in earlier literature [12, 13], rocket engines have some minimum and maximum throttle/thrust magnitude constraints. The minimum thrust constraint is considered non-convex and can be convexified through the use of a slack variable $\Gamma[k]$ where

$$||\mathbf{T}[k]|| \leq \Gamma[k]. \tag{33}$$

It can be shown that for an optimal solution, $\Gamma[k] = ||\mathbf{T}[k]||$ and that an optimal solution for the convexified problem recovers the optimal solution to the original, non-convex problem. Therefore, Γ can be used as a proxy for $||\mathbf{T}||$ in the mass depletion dynamics shown in Eq. (31).

In order to remove the nonlinearity due to time-varying mass, $m[k]$ in Eq. (26) is replaced with $\mu[k]$, the mass along the reference trajectory used in linearization. Similarly, in order to remove the nonlinearity due to aerodynamic drag, $||\mathbf{v}[k]||$ is replaced with $s[k]$, the velocity magnitude along the reference trajectory.

By fixing the final time, removing nonlinearities due to drag and time-varying mass, convexifying the minimum thrust constraints, and inclusion of the synthetic acceleration term leads to a new set of linear discrete-time equations of motion shown in Eqs. (34) – (37) that can be incorporated into a convex optimization solver:

$$m[k+1] = m[k] - \left[\frac{\alpha}{2}(\Gamma[k] + \Gamma[k+1]) + \dot{m}_{bp}\right]\Delta\tau, \tag{34}$$

$$\mathbf{r}[k+1] = \mathbf{r}[k] + \mathbf{v}[k]\Delta\tau + \frac{1}{3}\left(\mathbf{a}[k] + \frac{1}{2}\mathbf{a}[k+1]\right)\Delta\tau^2, \tag{35}$$

$$\mathbf{v}[k+1] = \mathbf{v}[k] + \frac{1}{2}\left(\mathbf{a}[k] + \mathbf{a}[k+1]\right)\Delta\tau, \tag{36}$$

$$\mathbf{a}[k] = \frac{1}{\mu[k]}\left(\mathbf{T}[k] - \frac{1}{2}\rho S_D C_D s[k]\mathbf{v}[k]\right) + \mathbf{a}_R[k] + \mathbf{g}, \tag{37}$$

$$\mathbf{a}[k+1] = \frac{1}{\mu[k+1]}\left(\mathbf{T}[k+1] - \frac{1}{2}\rho S_D C_D s[k+1]\mathbf{v}[k+1]\right) + \mathbf{a}_R[k+1] + \mathbf{g}.$$

Velocity Propagation To propagate the velocity forward by one time segment, Eq. (37) is substituted into Eq. (36) such that

$$\begin{aligned}\mathbf{v}[k+1] &= \mathbf{v}[k] + \frac{1}{2}\left(\mathbf{a}[k] + \mathbf{a}[k+1]\right)\Delta\tau \\ &= \mathbf{v}[k] + \frac{1}{2}\left(\frac{1}{\mu[k]}\left[\mathbf{T}[k] - \frac{1}{2}\rho S_D C_D s[k]\mathbf{v}[k]\right] + \mathbf{g} + \mathbf{a}_R[k]\right. \\ &\quad \left. + \frac{1}{\mu[k+1]}\left[\mathbf{T}[k+1] - \frac{1}{2}\rho S_D C_D s[k+1]\mathbf{v}[k+1]\right] + \mathbf{g} + \mathbf{a}_R[k+1]\right)\Delta\tau\end{aligned}$$

Collecting terms gives

$$\begin{aligned}&\left(1 + \frac{1}{4\mu[k+1]}\rho S_D C_D s[k+1]\Delta\tau\right)\mathbf{v}[k+1] = \\ &\left(1 - \frac{1}{4\mu[k]}\rho S_D C_D s[k]\Delta\tau\right)\mathbf{v}[k] + \frac{\Delta\tau}{2}\left(\mathbf{a}_R[k] + \mathbf{a}_R[k+1]\right) + \Delta\tau\mathbf{g} + \frac{\Delta\tau}{2}\left(\frac{1}{\mu[k]}\mathbf{T}[k] + \frac{1}{\mu[k+1]}\mathbf{T}[k+1]\right)\end{aligned}$$

which can be written as

$$\begin{aligned}\mathbf{v}[k+1] =& \frac{c_{\mathbf{v}}[k]}{d_{\mathbf{v}}[k]}\mathbf{1}\mathbf{v}[k] + \frac{1}{d_{\mathbf{v}}[k]}\begin{bmatrix}\frac{\Delta\tau}{2\mu[k]}\mathbf{1} & \frac{\Delta\tau}{2\mu[k+1]}\mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{T}[k] \\ \mathbf{T}[k+1]\end{bmatrix} \\ &+ \frac{1}{d_{\mathbf{v}}[k]}\begin{bmatrix}\frac{\Delta\tau}{2}\mathbf{1} & \frac{\Delta\tau}{2}\mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{a}_R[k] \\ \mathbf{a}_R[k+1]\end{bmatrix} + \frac{\Delta\tau}{d_{\mathbf{v}}[k]}\mathbf{g}\end{aligned} \tag{38}$$

$$= \mathbf{A}_{\mathbf{v}}[k]\mathbf{v}[k] + \mathbf{B}_{\mathbf{v}}[k]\mathbf{u}[k] + \mathbf{F}_{\mathbf{v}}[k]\mathbf{p}_{\mathbf{rv}}[k] + \mathbf{w}_{\mathbf{v}}[k], \tag{39}$$

where

$$c_{\mathbf{v}}[k] = \left(1 - \frac{1}{4\mu[k]}\rho S_D C_D s[k]\Delta\tau\right) \in \mathbb{R}^{1\times 1}, \tag{40}$$

$$d_{\mathbf{v}}[k] = \left(1 + \frac{1}{4\mu[k+1]}\rho S_D C_D s[k+1]\Delta\tau\right) \in \mathbb{R}^{1\times 1}, \tag{41}$$

$$\mathbf{A}_{\mathbf{v}}[k] = \frac{c_{\mathbf{v}}[k]}{d_{\mathbf{v}}[k]}\mathbf{1} \in \mathbb{R}^{3\times 3}, \tag{42}$$

$$\mathbf{B}_{\mathbf{v}}[k] = \frac{1}{d_{\mathbf{v}}[k]}\begin{bmatrix}\frac{\Delta\tau}{2\mu[k]}\mathbf{1} & \frac{\Delta\tau}{2\mu[k+1]}\mathbf{1}\end{bmatrix} \in \mathbb{R}^{3\times 6}, \tag{43}$$

$$\mathbf{F}_{\mathbf{v}}[k] = \frac{1}{d_{\mathbf{v}}[k]}\begin{bmatrix}\frac{\Delta\tau}{2}\mathbf{1} & \frac{\Delta\tau}{2}\mathbf{1}\end{bmatrix} \in \mathbb{R}^{3\times 6}, \tag{44}$$

$$\mathbf{w}_{\mathbf{v}}[k] = \frac{\Delta\tau}{d_{\mathbf{v}}[k]}\mathbf{g} \in \mathbb{R}^{3\times 1}, \tag{45}$$

$$\mathbf{u}[k] = \begin{bmatrix}\mathbf{T}[k]^T & \mathbf{T}[k+1]^T\end{bmatrix}^T \in \mathbb{R}^{6\times 1}, \tag{46}$$

$$\mathbf{p}_{\mathbf{rv}}[k] = \begin{bmatrix}\mathbf{a}_R[k]^T & \mathbf{a}_R[k+1]^T\end{bmatrix}^T \in \mathbb{R}^{6\times 1}. \tag{47}$$

Position Propagation In a similar way to deriving the velocity propagation equation, Eq. (37) is substituted into Eq. (35) giving

$$\begin{aligned}\mathbf{r}[k+1] &= \mathbf{r}[k] + \mathbf{v}[k]\Delta\tau + \frac{1}{3}\left(\mathbf{a}[k] + \frac{1}{2}\mathbf{a}[k+1]\right)\Delta\tau^2 \\ &= \mathbf{r}[k] + \mathbf{v}[k]\Delta\tau + \frac{1}{3}\left(\frac{1}{\mu[k]}\left[\mathbf{T}[k] - \frac{1}{2}\rho S_D C_D s[k]\mathbf{v}[k]\right] + \mathbf{g} + \mathbf{a}_R[k]\right. \\ &\quad \left. + \frac{1}{2}\left(\frac{1}{\mu[k+1]}\left[\mathbf{T}[k+1] - \frac{1}{2}\rho S_D C_D s[k+1]\mathbf{v}[k+1]\right] + \mathbf{g} + \mathbf{a}_R[k+1]\right)\right)\Delta\tau^2.\end{aligned}$$

Collecting terms gives

$$\begin{aligned}\mathbf{r}[k+1] = \mathbf{r}[k] &+ \left(1 - \frac{1}{6\mu[k]}\rho S_D C_D s[k]\Delta\tau\right)\Delta\tau\mathbf{1}\mathbf{v}[k] \\ &+ \frac{\Delta\tau^2}{2}\mathbf{g} + \frac{\Delta\tau^2}{3}\left(\mathbf{a}_R[k] + \frac{1}{2}\mathbf{a}_R[k+1]\right) + \frac{\Delta\tau^2}{3}\left(\frac{1}{\mu[k]}\mathbf{T}[k] + \frac{1}{2\mu[k+1]}\mathbf{T}[k+1]\right) \\ &- \frac{\Delta\tau^2}{12\mu[k+1]}\rho S_D C_D s[k+1]\mathbf{v}[k+1],\end{aligned}$$

which, after substituting Eq. (39), can be written as

$$\begin{aligned}\mathbf{r}[k+1] = &\begin{bmatrix}\mathbf{1} & \mathbf{c}_{\mathbf{r}}[k]\end{bmatrix}\begin{bmatrix}\mathbf{r}[k] \\ \mathbf{v}[k]\end{bmatrix} + \begin{bmatrix}\frac{\Delta\tau^2}{3\mu[k]}\mathbf{1} & \frac{\Delta\tau^2}{6\mu[k+1]}\mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{T}[k] \\ \mathbf{T}[k+1]\end{bmatrix} \\ &+ \begin{bmatrix}\frac{\Delta\tau^2}{3}\mathbf{1} & \frac{\Delta\tau^2}{6}\mathbf{1}\end{bmatrix}\begin{bmatrix}\mathbf{a}_R[k] \\ \mathbf{a}_R[k+1]\end{bmatrix} + \frac{\Delta\tau^2}{2}\mathbf{g} \\ &- \frac{\Delta\tau^2}{12\mu[k+1]}\rho S_D C_D s[k+1]\left(\mathbf{A}_{\mathbf{v}}[k]\mathbf{v}[k] + \mathbf{B}_{\mathbf{v}}[k]\begin{bmatrix}\mathbf{T}[k] \\ \mathbf{T}[k+1]\end{bmatrix} + \mathbf{F}_{\mathbf{v}}[k]\mathbf{p}_{\mathbf{rv}}[k] + \mathbf{w}_{\mathbf{v}}[k]\right)\end{aligned} \tag{48}$$

which simplifies to

$$\mathbf{r}[k+1] = \mathbf{A}_{\mathbf{r}}[k]\begin{bmatrix}\mathbf{r}[k] \\ \mathbf{v}[k]\end{bmatrix} + \mathbf{B}_{\mathbf{r}}[k]\mathbf{u}[k] + \mathbf{F}_{\mathbf{r}}[k]\mathbf{p}_{\mathbf{rv}}[k] + \mathbf{w}_{\mathbf{r}}[k] \tag{49}$$

where

$$\mathbf{c}_\mathbf{r}[k] = \left(1 - \frac{1}{6\mu[k]}\rho S_D C_D s[k]\Delta\tau\right)\Delta\tau\mathbf{1} \in \mathbb{R}^{3\times 3}, \tag{50}$$

$$d_\mathbf{r}[k] = -\frac{\Delta\tau^2}{12\mu[k+1]}\rho S_D C_D s[k+1] \in \mathbb{R}^{1\times 1}, \tag{51}$$

$$\mathbf{A}_\mathbf{r}[k] = \begin{bmatrix}\mathbf{1} & \mathbf{c}_\mathbf{r}[k] + d_\mathbf{r}[k]\mathbf{A}_\mathbf{v}[k]\end{bmatrix} \in \mathbb{R}^{3\times 6}, \tag{52}$$

$$\mathbf{B}_\mathbf{r}[k] = \begin{bmatrix}\frac{\Delta\tau^2}{3\mu[k]}\mathbf{1} & \frac{\Delta\tau^2}{6\mu[k+1]}\mathbf{1}\end{bmatrix} + d_\mathbf{r}[k]\mathbf{B}_\mathbf{v}[k] \in \mathbb{R}^{3\times 6}, \tag{53}$$

$$\mathbf{F}_\mathbf{r}[k] = \begin{bmatrix}\frac{\Delta\tau^2}{3}\mathbf{1} & \frac{\Delta\tau^2}{6}\mathbf{1}\end{bmatrix} + d_\mathbf{r}[k]\mathbf{F}_\mathbf{v} \in \mathbb{R}^{3\times 6}, \tag{54}$$

$$\mathbf{w}_\mathbf{r}[k] = \frac{\Delta\tau^2}{2}\mathbf{g} + d_\mathbf{r}\mathbf{w}_\mathbf{v}[k] \in \mathbb{R}^{3\times 1}. \tag{55}$$

Mass Propagation To write the mass dynamics in a useful form, all that is required is factoring Eq. (34) to obtain

$$m[k+1] = m[k] + \begin{bmatrix}-\alpha\Delta\tau/2 & -\alpha\Delta\tau/2\end{bmatrix}\begin{bmatrix}\Gamma[k] \\ \Gamma[k+1]\end{bmatrix} - \Delta\tau\dot{m}_{bp} \tag{56}$$

$$= m[k] + \mathbf{F}_m\mathbf{p}_m[k] + w_m, \tag{57}$$

where

$$\mathbf{F}_m = \begin{bmatrix}-\alpha\Delta\tau/2 & -\alpha\Delta\tau/2\end{bmatrix} \in \mathbb{R}^{1\times 2}, \tag{58}$$

$$\mathbf{p}_m[k] = \begin{bmatrix}\Gamma[k] & \Gamma[k+1]\end{bmatrix}^T \in \mathbb{R}^{2\times 1}, \tag{59}$$

$$w_m = -\Delta\tau\dot{m}_{bp} \in \mathbb{R}^{1\times 1}. \tag{60}$$

Full State Dynamics Obtaining the dynamics of the full state of the vehicle simply requires stacking Eqs. (39), (49) and (56) yielding

$$\mathbf{x}[k+1] = \mathbf{A}[k]\mathbf{x}[k] + \mathbf{B}[k]\mathbf{u}[k] + \mathbf{F}[k]\mathbf{p}[k] + \mathbf{w}[k] \tag{61}$$

where

$$\mathbf{x}[k] = \begin{bmatrix}m[k] \\ \mathbf{r}[k] \\ \mathbf{v}[k]\end{bmatrix} \in \mathbb{R}^{7\times 1}, \tag{62}$$

$$\mathbf{A}[k] = \begin{bmatrix}1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_\mathbf{r}[k] \\ \mathbf{0} & \mathbf{A}_\mathbf{v}[k]\end{bmatrix} \in \mathbb{R}^{7\times 7}, \tag{63}$$

$$\mathbf{B}[k] = \begin{bmatrix}\mathbf{0} \\ \mathbf{B}_\mathbf{r} \\ \mathbf{B}_\mathbf{v}\end{bmatrix} \in \mathbb{R}^{7\times 6}, \tag{64}$$

$$\mathbf{F}[k] = \begin{bmatrix}\mathbf{F}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_\mathbf{r} \\ \mathbf{0} & \mathbf{F}_\mathbf{v}\end{bmatrix} \in \mathbb{R}^{7\times 8}, \tag{65}$$

$$\mathbf{p}[k] = \begin{bmatrix}\mathbf{p}_m \\ \mathbf{p}_\mathbf{rv}\end{bmatrix} = \begin{bmatrix}\Gamma[k] \\ \Gamma[k+1] \\ \mathbf{a}_\mathbf{R}[k] \\ \mathbf{a}_\mathbf{R}[k+1]\end{bmatrix} \in \mathbb{R}^{8\times 1}, \tag{66}$$

$$\mathbf{w}[k] = \begin{bmatrix}w_m \\ \mathbf{w}_\mathbf{r} \\ \mathbf{w}_\mathbf{v}\end{bmatrix} \in \mathbb{R}^{7\times 1}. \tag{67}$$

These local dynamics are used for the first iteration of the optimization algorithm to propagate the vehicle state between one temporal node and the next. The local dynamics are used to construct the global dynamics of the form described by Eq. (11) in Sec. 1.3, which are needed to write state constraints and embed the dynamics into the optimization problem.

3.2 Optimization Problem

The first iteration of the optimization algorithm aims to solve the optimization problem laid out in Problem 4. This has the objective,

$$\min_{\mathbf{T},\mathbf{\Gamma}} -w_{m,f} m[k_f] + w_{\kappa,\mathbf{a},R} ||\boldsymbol{\kappa}_{\mathbf{a}_R}||, \tag{68}$$

where $w_{m,f}$ and $w_{\kappa,\mathbf{a},R}$ are tuning weights related to the final mass of the vehicle at the end of the trajectory, $m[k_f]$, and the 2-norm of the vector containing the synthetic acceleration slack variables, respectively. This objective function is minimized subject to boundary conditions, state, control, and other constraints.

For the first iteration, the vector containing all of the free parameters over which the objective function is optimized is $\tilde{\mathbf{x}}$ which consists of

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{T}}^T & \tilde{\mathbf{\Gamma}}^T & \tilde{\mathbf{A}}_R^T & \boldsymbol{\kappa}_{\mathbf{a},R}^T & \tilde{x}_{||\boldsymbol{\kappa}_{\mathbf{a},R}||} & \tilde{x}_{m,k_f} \end{bmatrix}^T \in \mathbb{R}^{(8N+2)\times 1}, \tag{69}$$

where $\tilde{mathbfT}$ is a stack of thrust vectors at each temporal node

$$\tilde{\mathbf{T}} = \begin{bmatrix} \mathbf{T}[0]^T & \cdots & \mathbf{T}[N-1]^T \end{bmatrix} \in \mathbb{R}^{3N\times 1}, \tag{70}$$

and $\tilde{\Gamma}$ contains all of the slack variables associated with the convexification of the thrust magnitude and pointing constraints

$$\tilde{\mathbf{\Gamma}} = \begin{bmatrix} \Gamma[0] & \cdots & \Gamma[N-1] \end{bmatrix}^T \in \mathbb{R}^{N\times 1}. \tag{71}$$

Similarly to the thrust vectors, the synthetic accelerations at each node are stacked to form $\tilde{\mathbf{A}}_R$,

$$\tilde{\mathbf{A}}_R = \begin{bmatrix} \mathbf{a}_R[0] & \cdots & \mathbf{a}_R[N-1] \end{bmatrix}^T \in \mathbb{R}^{3N\times 1}. \tag{72}$$

In order to embed the penalty associated with the use synthetic acceleration into the objective function, the magnitude of the synthetic acceleration is bounded point-wise in time by additional slack variables, $\kappa_{\mathbf{a},R}[k]$,

$$||\mathbf{a}_R[k]|| \leq \kappa_{\mathbf{a},R}[k] \tag{73}$$

These point-wise bounds are stacked to form

$$\boldsymbol{\kappa}_{\mathbf{a},R} = \begin{bmatrix} \kappa_{\mathbf{a},R}[0] & \cdots & \kappa_{\mathbf{a},R}[N-1] \end{bmatrix}^T \in \mathbb{R}^{N\times 1}. \tag{74}$$

An optimization variable, $\tilde{x}_{||\boldsymbol{\kappa}_{\mathbf{a},R}||}$, is added to bound the norm of $\boldsymbol{\kappa}_{\mathbf{a},R}$ and penalize the use of synthetic acceleration through

$$||\boldsymbol{\kappa}_{\mathbf{a},R}|| \leq \tilde{x}_{||\boldsymbol{\kappa}_{\mathbf{a},R}||}. \tag{75}$$

Finally, in order to optimize the trajectory with respect to fuel usage, an additional variable, $\tilde{x}_{m,k_f}$, which will later be constrained to be equal to the final mass of the vehicle through the dynamics, is included as an optimization variable.

With $\tilde{\mathbf{x}}$ defined, the objective function in Eq. (68) is bounded from above by Eq. (1) where

$$\mathbf{f} = \begin{bmatrix} \mathbf{0}_{1\times 8N} & w_{\kappa,\mathbf{a},R} & -w_{m,f} \end{bmatrix}^T. \tag{76}$$

Minimizing this objective function results in a tight upper bound on Eq. (68).

3.2.1 Boundary Condition Constraints

Several constraints are needed to ensure that the terminal conditions are satisfied subject to the dynamics. For instance, it desired that the vehicle touches down at the desired landing site with zero velocity and at an upright attitude as described in Eq. (62) of Ref. [1].

Final Position In order for the vehicle to land at the desired position at the final time, the optimization must be constrained to only produce solutions where the final position matches the desired. The vector $\mathbf{X}$ contains all of the vehicle states throughout the entire trajectory. Therefore the final position, at index $k_f = N-1$, can be extracted from $\mathbf{X}$ using

$$\mathbf{r}[k_f] = \mathbf{M}_{\mathbf{r},k_f}\mathbf{X}, \tag{77}$$

where

$$\mathbf{M}_{\mathbf{r},k_f} = \begin{bmatrix}\mathbf{0}_{3\times[n_x(N-2)+1]} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3}\end{bmatrix} \in \mathbb{R}^{3\times n_x(N-1)}. \tag{78}$$

For this problem, $n_x = 7$. Next, the expression for the global dynamics given by Eq. (11) can be substituted for $\mathbf{X}$

$$\mathbf{r}[k_f] = \mathbf{M}_{\mathbf{r},k_f}\left(\mathbf{M}\mathbf{x}_0 + \mathbf{S}_1\mathbf{U} + \mathbf{S}_2\mathbf{P} + \mathbf{S}_3\mathbf{W}\right), \tag{79}$$

where $\mathbf{U}$ and $\mathbf{P}$ are constructed from optimization variables contained in $\tilde{\mathbf{x}}$ according to Eqs. (13), (14), (46) and (66). The vector $\mathbf{U}$ can be obtained through

$$\mathbf{U} = \boldsymbol{\Upsilon}_{\mathbf{U}}\tilde{\mathbf{x}}, \tag{80}$$

where $\boldsymbol{\Upsilon}_{\mathbf{U}}$ is of the form

$$\boldsymbol{\Upsilon}_{\mathbf{U}} = \left[\begin{array}{ccccccc|c} \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} & \cdots & \cdots & \cdots & \cdots & \mathbf{0}_{3\times3} & \\ \mathbf{0}_{3\times3} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} & \cdots & \cdots & \cdots & \mathbf{0}_{3\times3} & \\ \mathbf{0}_{3\times3} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} & \cdots & \cdots & \cdots & \mathbf{0}_{3\times3} & \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} & \cdots & \cdots & \mathbf{0}_{3\times3} & \mathbf{0}_{6(N-1)\times(5N+2)} \\ \vdots & & & & \ddots & & \vdots & \\ \mathbf{0}_{3\times3} & \cdots & \cdots & \cdots & \cdots & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times3} & \\ \mathbf{0}_{3\times3} & \cdots & \cdots & \cdots & \cdots & \mathbf{0}_{3\times3} & \mathbf{1}_{3\times3} & \end{array}\right], \tag{81}$$

which can be constructed as

$$\boldsymbol{\Upsilon}_{\mathbf{U}} = \begin{bmatrix}\boldsymbol{\Upsilon}_{\mathbf{u},0}^T & \cdots & \boldsymbol{\Upsilon}_{\mathbf{u},N-2}^T\end{bmatrix}^T \in \mathbb{R}^{6(N-1)\times(8N+2)}, \tag{82}$$

where

$$\boldsymbol{\Upsilon}_{\mathbf{u},k} = \begin{bmatrix}\mathbf{0}_{6\times3k} & \mathbf{1}_{6\times6} & \mathbf{0}_{6\times[8N-3k-4]}\end{bmatrix} \in \mathbb{R}^{6\times(8N+2)}. \tag{83}$$

Similarly, $\mathbf{P}$ is obtained through

$$\mathbf{P} = \boldsymbol{\Upsilon}_{\mathbf{P}}\tilde{\mathbf{x}}, \tag{84}$$

where

$$\boldsymbol{\Upsilon}_{\mathbf{P}} = \begin{bmatrix}\boldsymbol{\Upsilon}_{\mathbf{p},0}^T & \cdots & \boldsymbol{\Upsilon}_{\mathbf{p},N-2}^T\end{bmatrix}^T \in \mathbb{R}^{8(N-1)\times(8N+2)}, \tag{85}$$

$$\boldsymbol{\Upsilon}_{\mathbf{p},k} = \begin{bmatrix}\mathbf{0}_{2\times[3N+k]} & \mathbf{1}_{2\times2} & \mathbf{0}_{2\times[5N-k]} \\ \mathbf{0}_{6\times[4N+3k]} & \mathbf{1}_{6\times6} & \mathbf{0}_{6\times[4N-3k-4]}\end{bmatrix} \in \mathbb{R}^{8\times(8N+2)}. \tag{86}$$

Substituting Eqs. (80) and (84) into (79) yields the final position in terms of the optimization vector as

$$\begin{aligned}\mathbf{r}[k_f] &= \mathbf{M}_{\mathbf{r},k_f}\left(\mathbf{M}\mathbf{x}_0 + \mathbf{S}_1\boldsymbol{\Upsilon}_{\mathbf{U}}\tilde{\mathbf{x}} + \mathbf{S}_2\boldsymbol{\Upsilon}_{\mathbf{P}}\tilde{\mathbf{x}} + \mathbf{S}_3\mathbf{W}\right) &(87)\\ &= \mathbf{M}_{\mathbf{r},k_f}\left(\mathbf{M}\mathbf{x}_0 + [\mathbf{S}_1\boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2\boldsymbol{\Upsilon}_{\mathbf{P}}]\tilde{\mathbf{x}} + \mathbf{S}_3\mathbf{W}\right). &(88)\end{aligned}$$

In order to write a constraint, the final position is set equal to the desired final position through

$$\mathbf{r}[k_f] = \mathbf{r}_d. \tag{89}$$

Substituting Eq. (87) for $\mathbf{r}[k_f]$ yields

$$\mathbf{r}_d = \mathbf{M}_{\mathbf{r},k_f} \left(\mathbf{M}\mathbf{x}_0 + [\mathbf{S}_1 \boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2 \boldsymbol{\Upsilon}_{\mathbf{P}}] \tilde{\mathbf{x}} + \mathbf{S}_3 \mathbf{W} \right), \tag{90}$$

which is rearranged to obtain

$$\mathbf{M}_{\mathbf{r},k_f} \left(\mathbf{S}_1 \boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2 \boldsymbol{\Upsilon}_{\mathbf{P}} \right) \tilde{\mathbf{x}} = \mathbf{r}_d - \mathbf{M}_{\mathbf{r},k_f} \left(\mathbf{M}\mathbf{x}_0 + \mathbf{S}_3 \mathbf{W} \right), \tag{91}$$

which matches the form described in Eq. (4) and compatible with *coneprog.m*.

Final Velocity Following the same approach for the equality constraint on the final position, the final velocity, at index $k_f = N - 1$, can be extracted from $\mathbf{X}$ using

$$\mathbf{v}[k_f] = \mathbf{M}_{\mathbf{v},k_f} \mathbf{X}, \tag{92}$$

where

$$\mathbf{M}_{\mathbf{v},k_f} = \begin{bmatrix} \mathbf{0}_{3 \times [n_x(N-2)+4]} & \mathbf{1}_{3\times 3} \end{bmatrix} \in \mathbb{R}^{3 \times n_x(N-1)}. \tag{93}$$

Next, the dynamics given by Eq. (11) can be substituted for $\mathbf{X}$ along with Eqs. (80) and (84) to obtain

$$\begin{aligned} \mathbf{v}[k_f] &= \mathbf{M}_{\mathbf{v},k_f} \left(\mathbf{M}\mathbf{x}_0 + \mathbf{S}_1 \mathbf{U} + \mathbf{S}_2 \mathbf{P} + \mathbf{S}_3 \mathbf{W} \right), & (94) \\ &= \mathbf{M}_{\mathbf{r},k_f} \left(\mathbf{M}\mathbf{x}_0 + [\mathbf{S}_1 \boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2 \boldsymbol{\Upsilon}_{\mathbf{P}}] \tilde{\mathbf{x}} + \mathbf{S}_3 \mathbf{W} \right). & (95) \end{aligned}$$

In order to write a constraint, the final velocity is set equal to the desired final velocity through

$$\mathbf{v}[k_f] = \mathbf{v}_d. \tag{96}$$

Substituting Eq. (94) for $\mathbf{v}[k_f]$ yields

$$\mathbf{v}_d = \mathbf{M}_{\mathbf{v},k_f} \left(\mathbf{M}\mathbf{x}_0 + [\mathbf{S}_1 \boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2 \boldsymbol{\Upsilon}_{\mathbf{P}}] \tilde{\mathbf{x}} + \mathbf{S}_3 \mathbf{W} \right), \tag{97}$$

which is rearranged to obtain

$$\mathbf{M}_{\mathbf{v},k_f} \left(\mathbf{S}_1 \boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2 \boldsymbol{\Upsilon}_{\mathbf{P}} \right) \tilde{\mathbf{x}} = \mathbf{v}_d - \mathbf{M}_{\mathbf{v},k_f} \left(\mathbf{M}\mathbf{x}_0 + \mathbf{S}_3 \mathbf{W} \right), \tag{98}$$

which matches the form described in Eq. (4) and is compatible with *coneprog.m*.

Initial Thrust Magnitude The thrust magnitude at the beginning of the trajectory is a user-defined value. This boundary condition is enforced through an equality constraint on $\Gamma[0]$, as this slack variable is equal to the thrust magnitude for an optimal solution [14]. This constraint has the form,

$$\Gamma[0] = \Gamma_0, \tag{99}$$

which must be written in terms of the optimization vector, $\tilde{\mathbf{x}}$. This is accomplished by using

$$\Gamma[0] = \boldsymbol{\Upsilon}_{\Gamma,0} \tilde{\mathbf{x}}, \tag{100}$$

where

$$\boldsymbol{\Upsilon}_{\Gamma,0} = \begin{bmatrix} \mathbf{0}_{1\times 3N} & 1 & \mathbf{0}_{1\times[5N+1]} \end{bmatrix} \in \mathbb{R}^{1\times(8N+2)}, \tag{101}$$

to extract $\Gamma[0]$ from $\tilde{\mathbf{x}}$. Substituting this expression into the constraint yields

$$\boldsymbol{\Upsilon}_{\Gamma,0} \tilde{\mathbf{x}} = \Gamma_0, \tag{102}$$

the final form of the equality constraint, matching Eq. (4).

Initial and Final Thrust Direction Due to the limitation of the 3 DoF dynamics, the direction of the thrust vector can be used as a proxy for the vehicle attitude dynamics. To ensure that the vehicle lands upright, the direction of the thrust vector at the end of the trajectory is constrained to be in the vertical direction. Additionally, the initial thrust direction can also be constrained to match the initial attitude. For the initial thrust direction constraint, the thrust vector must satisfy

$$\mathbf{T}[0] = \Gamma[0]\hat{\mathbf{n}}_0, \tag{103}$$

where $\hat{\mathbf{n}}_0$ is the unit vector specifying the direction of the initial thrust vector. To write this equation in terms of $\tilde{\mathbf{x}}$, the initial thrust vector can be extracted through

$$\mathbf{T}_0 = \boldsymbol{\Upsilon}_{\mathbf{T},0}\tilde{\mathbf{x}}, \tag{104}$$

where

$$\boldsymbol{\Upsilon}_{\mathbf{T},0} = \begin{bmatrix}\mathbf{1}_{3\times3} & \mathbf{0}_{3\times(8N-1)}\end{bmatrix} \in \mathbb{R}^{3\times(8N+2)}. \tag{105}$$

This expression is substituted into Eq. (103) along with Eq. (102) to obtain

$$\boldsymbol{\Upsilon}_{\mathbf{T},0}\tilde{\mathbf{x}} = \boldsymbol{\Upsilon}_{\Gamma,0}\tilde{\mathbf{x}}\hat{\mathbf{n}}_0, \tag{106}$$

which can be rearranged into the form of an equality constraint described by Eq. (4)

$$\left(\boldsymbol{\Upsilon}_{\mathbf{T},0} - \boldsymbol{\Upsilon}_{\Gamma,0}\hat{\mathbf{n}}_0\right)\tilde{\mathbf{x}} = 0, \tag{107}$$

Similarly, the final thrust must satisfy

$$\mathbf{T}[k_f] = \Gamma[k_f]\hat{\mathbf{n}}_f, \tag{108}$$

where $k_f = N-1$ is the index of the node at the end of the trajectory and $\hat{\mathbf{n}}_f$ is the unit vector specifying the desired final direction of the thrust vector. Again, this equation is expressed in terms of $\tilde{\mathbf{x}}$ through

$$\mathbf{T}[k_f] = \boldsymbol{\Upsilon}_{\mathbf{T},k_f}\tilde{\mathbf{x}} \tag{109}$$

where

$$\boldsymbol{\Upsilon}_{\mathbf{T},k_f} = \begin{bmatrix}\mathbf{0}_{3\times3(N-1)} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times[5N+2]}\end{bmatrix} \in \mathbb{R}^{3\times(8N+2)} \tag{110}$$

and rearranged into the form of an equality constraint described by Eq. (4),

$$\left(\boldsymbol{\Upsilon}_{\mathbf{T},k_f} - \boldsymbol{\Upsilon}_{\Gamma,k_f}\hat{\mathbf{n}}_f\right)\tilde{\mathbf{x}} = 0, \tag{111}$$

where

$$\Gamma[k_f] = \boldsymbol{\Upsilon}_{\Gamma,k_f}\tilde{\mathbf{x}}, \tag{112}$$

$$\boldsymbol{\Upsilon}_{\Gamma,k_f} = \begin{bmatrix}\mathbf{0}_{1\times[4N-1]} & 1 & \mathbf{0}_{1\times[4N+2]}\end{bmatrix} \in \mathbb{R}^{1\times(8N+2)}. \tag{113}$$

3.2.2 State Constraints

Minimum Mass To ensure that the vehicle consumes less fuel than it is available, the mass of the vehicle is constrained to be larger than the dry mass at each temporal node. The mass at the k^{th} temporal node can be written as

$$m[k] = \mathbf{M}_{m,k}\mathbf{X}, \tag{114}$$

where

$$\mathbf{M}_{m,k} = \begin{bmatrix}\mathbf{0}_{1\times n_x(k-1)} & 1 & \mathbf{0}_{1\times[n_x(N-k)-1]}\end{bmatrix} \in \mathbb{R}^{1\times n_x(N-1)}. \tag{115}$$

The mass $m[k]$ must satisfy

$$m_{\text{dry}} \leq m[k] \tag{116}$$

at nodes $k \in [1, k_f]$ where $k_f = N-1$. Following a process similar to that used to derive Eq. (91) yields

$$-\mathbf{M}_{m,k}\left(\mathbf{S}_1\boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2\boldsymbol{\Upsilon}_{\mathbf{P}}\right)\tilde{\mathbf{x}}_i \leq \mathbf{M}_{m,k}\left(\mathbf{M}_i\mathbf{x}_0 + \mathbf{S}_3\mathbf{W}\right) - m_{\text{dry}}, \tag{117}$$

a linear inequality constraint which is enforced at each k.

Minimum Glideslope For safety reasons and avoiding flying too close to the surface, the vehicle is constrained to remain inside of a prescribed glideslope cone with a half-angle of γ_{gs}, shown graphically in Fig. 1.

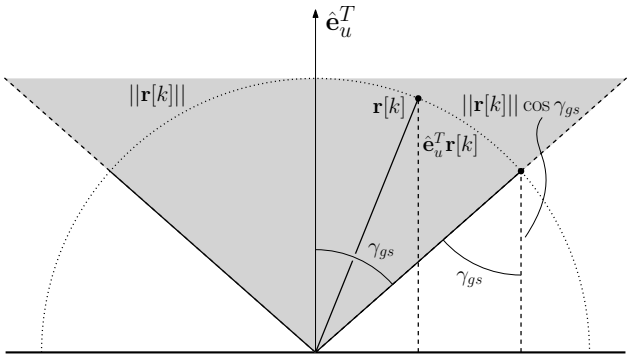


Figure 1: The vertical component of the vehicle position at temporal node k, $\hat{\mathbf{e}}_u^T\mathbf{r}[k]$ must be larger than $||\mathbf{r}[k]|| \cos\gamma_{gs}$ to ensure that the vehicle is within the minimum glide slope cone (grey).

In order for the vehicle to remain inside the glideslope cone,

$$||\mathbf{r}[k]|| \cos\gamma_{gs} \leq \hat{\mathbf{e}}_u^T\mathbf{r}[k], \tag{118}$$

Eq. (118) must be satisfied for nodes $k \in [1, N-2]$. The position at the k^{th} node is obtained through

$$\mathbf{r}[k] = \mathbf{M}_{\mathbf{r},k}\mathbf{X} \tag{119}$$

where

$$\mathbf{M}_{\mathbf{r},k} = \begin{bmatrix}\mathbf{0}_{3\times[n_x(k-1)+1]} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times[n_x(N-k-1)+3]}\end{bmatrix} \in \mathbb{R}^{3\times n_x(N-1)}. \tag{120}$$

Substituting the dynamics and position at k leads to

$$LHS_{gs,k} \leq RHS_{gs,k} \tag{121}$$

where

$$LHS_{gs,k} = ||\mathbf{M}_{\mathbf{r},k}(\mathbf{S}_1\boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2\boldsymbol{\Upsilon}_{\mathbf{P}})\tilde{\mathbf{x}} + \mathbf{M}_{\mathbf{r},k}\left(\mathbf{M}\mathbf{x}_0 + \mathbf{S}_3\mathbf{W}\right)|| \tag{122}$$

and

$$RHS_{gs,k} = \frac{1}{\cos\gamma_{gs}}\hat{\mathbf{e}}_u^T\mathbf{M}_{\mathbf{r},k}(\mathbf{S}_1\boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2\boldsymbol{\Upsilon}_{\mathbf{P}})\tilde{\mathbf{x}} + \frac{1}{\cos\gamma_{gs}}\hat{\mathbf{e}}_u^T\mathbf{M}_{\mathbf{r},k}\left(\mathbf{M}\mathbf{x}_0 + \mathbf{S}_3\mathbf{W}\right) \tag{123}$$

which constitutes an SOC constraint in the form described by Eq. (2).

3.2.3 Control Constraints

Convexification of Thrust Constraints The minimum thrust magnitude and thrust pointing constraints are convexified by enforcing

$$||\mathbf{T}[k]|| \leq \Gamma[k] \tag{124}$$

and

$$T_{\min} \leq \Gamma[k] \leq T_{\max}. \tag{125}$$

These expressions can be written in terms of the optimization vector $\tilde{\mathbf{x}}$ through

$$\mathbf{T}[k] = \boldsymbol{\Upsilon}_{\mathbf{T},k}\tilde{\mathbf{x}} \tag{126}$$

and

$$\Gamma[k] = \boldsymbol{\Upsilon}_{\Gamma,k}\tilde{\mathbf{x}} \tag{127}$$

where

$$\boldsymbol{\Upsilon}_{\mathbf{T},k} = \begin{bmatrix} \mathbf{0}_{3\times 3k} & \mathbf{1}_{3\times 3} & \mathbf{0}_{(8N-3k-1)} \end{bmatrix} \in \mathbb{R}^{3\times(8N+2)}, \tag{128}$$

and

$$\boldsymbol{\Upsilon}_{\Gamma,k} = \begin{bmatrix} \mathbf{0}_{1\times(3N+k)} & 1 & \mathbf{0}_{1\times(5N-k+1)} \end{bmatrix} \in \mathbb{R}^{3\times(8N+2)}, \tag{129}$$

which results in

$$||\boldsymbol{\Upsilon}_{\mathbf{T},k}\tilde{\mathbf{x}}|| \leq \boldsymbol{\Upsilon}_{\Gamma,k}\tilde{\mathbf{x}}, \tag{130}$$

a second order cone constraint in the form given by Eq. (2). Enforcing this constraint for $k \in [0, N-1]$ yields N SOC constraints. The minimum and maximum thrust constraints can be enforced by bounding the optimization variables in the same format as Eq. (5) resulting in

$$\begin{bmatrix} -\infty\mathbf{1}_{3N\times 1} \\ T_{\min}\mathbf{1}_{N\times 1} \\ -\infty\mathbf{1}_{(4N+2)\times 1} \end{bmatrix} \leq \tilde{\mathbf{x}} \leq \begin{bmatrix} \infty\mathbf{1}_{3N\times 1} \\ T_{\max}\mathbf{1}_{N\times 1} \\ \infty\mathbf{1}_{(4N+2)\times 1} \end{bmatrix}. \tag{131}$$

Maximum Tilt Angle The direction of the thrust vector is constrained to be within an angle, $\theta_{\max}$, of the vertical. This constraint is satisfied when

$$\Gamma[k]\cos\theta_{\max} \leq \hat{\mathbf{e}}_u^T\mathbf{T}[k]. \tag{132}$$

Substituting Eqs. (126) and (127) and rearranging yields

$$\left(\boldsymbol{\Upsilon}_{\Gamma,k}\cos\theta_{\max} - \hat{\mathbf{e}}_u^T\boldsymbol{\Upsilon}_{\mathbf{T},k}\right)\tilde{\mathbf{x}} \leq 0, \tag{133}$$

a linear inequality constraint of the form described by Eq. (3). This constraint is enforced for $k \in [1, N-2]$ as the initial and final thrust directions are prescribed.

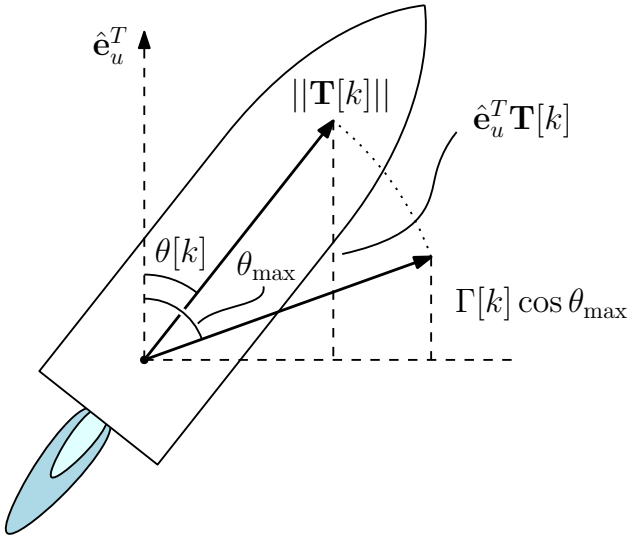


Figure 2: The vertical component of the thrust vector at temporal node k, $\hat{\mathbf{e}}_u^T\mathbf{T}[k]$, must be larger than $\Gamma[k]\cos\theta_{\max}$ to ensure that the vehicle is within the maximum thrust pointing angle. This figure assumes an optimal solution where $||\mathbf{T}[k]|| = \Gamma[k]$.

Thrust Rate In order to prevent the rate of change in thrust magnitude from exceeding the throttle rate capabilities of the engine, which is assumed to be constant over a time segment, the following expression is used

$$\dot{T}_{\min} \leq \frac{\Gamma[k+1] - \Gamma[k]}{\Delta\tau} \leq \dot{T}_{\max}, \tag{134}$$

where $\dot{T}_{\max}$ and $\dot{T}_{\min}$ are constants. Assuming $\dot{T}_{\min} = -\dot{T}_{\max}$, Eq. (134) can be re-written as

$$\begin{bmatrix} \Gamma[k] - \Gamma[k+1] \\ \Gamma[k+1] - \Gamma[k] \end{bmatrix} \leq \Delta\tau \begin{bmatrix} \dot{T}_{\max} \\ \dot{T}_{\max} \end{bmatrix}. \tag{135}$$

Defining

$$\Gamma[k] - \Gamma[k+1] = \boldsymbol{\Upsilon}_{\Delta\Gamma,k}\tilde{\mathbf{x}}, \tag{136}$$

where,

$$\boldsymbol{\Upsilon}_{\Delta\Gamma,k} = \begin{bmatrix} \mathbf{0}_{1\times(3N+k)} & 1 & -1 & \mathbf{0}_{1\times(5N-k)} \end{bmatrix} \in \mathbb{R}^{1\times(8N+2)}, \tag{137}$$

allows the throttle rate constraint to be written as

$$\begin{bmatrix} \boldsymbol{\Upsilon}_{\Delta\Gamma,k} \\ -\boldsymbol{\Upsilon}_{\Delta\Gamma,k} \end{bmatrix} \tilde{\mathbf{x}} \leq \Delta\tau \begin{bmatrix} T_{\max} \\ T_{\max} \end{bmatrix}, \tag{138}$$

which is enforced for $k \in [0, N-2]$.

3.2.4 Objective Function Constraints

Final Mass The optimization variable $\tilde{x}_{m,k_f}$ is included in $\tilde{\mathbf{x}}$ in order to include the final mass of the vehicle in the objective function. This variable is constrained to be equal to the final mass of the vehicle subject to the dynamics according to

$$\tilde{x}_{m,k_f} = m[k_f]. \tag{139}$$

The final mass, $m[k_f]$, is extracted from the concatenated state vector, $\mathbf{X}$, using the definition in Eq. (115) while $\tilde{x}_{m,k_f}$ is extracted from $\tilde{\mathbf{x}}$ through

$$\tilde{x}_{m,k_f} = \boldsymbol{\Upsilon}_{m,k_f}\tilde{\mathbf{x}} \tag{140}$$

where

$$\boldsymbol{\Upsilon}_{m,k_f} = \begin{bmatrix} \mathbf{0}_{1\times(8N+1)} & 1 \end{bmatrix} \in \mathbb{R}^{1\times(8N+2)}. \tag{141}$$

Substituting the dynamics and rearranging yields the linear equality constraint,

$$\left(\boldsymbol{\Upsilon}_{m,k_f} - \mathbf{M}_{m,k_f}\left[\mathbf{S}_1\boldsymbol{\Upsilon}_{\mathbf{U}} + \mathbf{S}_2\boldsymbol{\Upsilon}_{\mathbf{P}}\right]\right)\tilde{\mathbf{x}} = \mathbf{M}_{m,k_f}\left(\mathbf{M}\mathbf{x}_0 + \mathbf{S}_3\mathbf{W}\right) \tag{142}$$

in a form matching Eq. (4).

Synthetic Acceleration In order to penalize the use of synthetic acceleration in the objective function, the vector $\mathbf{a}_R[k]$ is bounded at each temporal node by $\kappa_{\mathbf{a},R}[k]$

$$||\mathbf{a}_R[k]|| \leq \kappa_{\mathbf{a},R}[k], \tag{143}$$

where $\kappa_{\mathbf{a},R}[k]$ are included as optimization variables in $\tilde{\mathbf{x}}$. Substituting the following expressions for $\mathbf{a}_R[k]$ and $\kappa_{\mathbf{a},R}[k]$

$$\mathbf{a}_R[k] = \boldsymbol{\Upsilon}_{\mathbf{a}_R,k}\tilde{\mathbf{x}} \tag{144}$$

$$\kappa_{\mathbf{a},R}[k] = \boldsymbol{\Upsilon}_{\kappa_{\mathbf{a}_R},k}\tilde{\mathbf{x}}, \tag{145}$$

where

$$\boldsymbol{\Upsilon}_{\mathbf{a}_R,k} = \begin{bmatrix} \mathbf{0}_{3\times(4N+3k)} & \mathbf{1}_{3\times3} & \mathbf{0}_{3\times(4N-3k-1)} \end{bmatrix} \in \mathbb{R}^{3\times(8N+2)} \tag{146}$$

$$\boldsymbol{\Upsilon}_{\kappa_{\mathbf{a}_R},k} = \begin{bmatrix} \mathbf{0}_{1\times(7N+k)} & 1 & \mathbf{0}_{1\times(N-k+1)} \end{bmatrix} \in \mathbb{R}^{1\times(8N+2)} \tag{147}$$

leads to

$$||\boldsymbol{\Upsilon}_{\mathbf{a}_R,k}\tilde{\mathbf{x}}|| \leq \boldsymbol{\Upsilon}_{\kappa_{\mathbf{a}_R},k}\tilde{\mathbf{x}}, \tag{148}$$

an SOC constraint in the form described by Eq. (2) which is enforced for $k \in [0, N-1]$.

Next, define $\boldsymbol{\kappa}_{\mathbf{a},R}$ as the vector containing $\kappa_{\mathbf{a},R}[k]$ at all the temporal nodes,

$$\boldsymbol{\kappa}_{\mathbf{a},R} = \begin{bmatrix} \kappa_{\mathbf{a},R}[0] \\ \vdots \\ \kappa_{\mathbf{a},R}[N-1] \end{bmatrix} \in \mathbb{R}^{N\times 1}, \tag{149}$$

the 2-norm of $\boldsymbol{\kappa}_{\mathbf{a},R}$ is bounded by

$$||\boldsymbol{\kappa}_{\mathbf{a},R}|| \leq \tilde{x}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||} \tag{150}$$

where $\tilde{x}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||}$ is an optimization variables that penalizes the use of synthetic acceleration in the objective function. $\boldsymbol{\kappa}_{\mathbf{a},R}$ can be extracted from $\tilde{\mathbf{x}}$ by

$$\boldsymbol{\kappa}_{\mathbf{a},R} = \boldsymbol{\Upsilon}_{\boldsymbol{\kappa}_{\mathbf{a}_R}} \tilde{\mathbf{x}}, \tag{151}$$

where

$$\boldsymbol{\Upsilon}_{\boldsymbol{\kappa}_{\mathbf{a}_R}} = \begin{bmatrix} \mathbf{0}_{N\times 7N} & \mathbf{1}_{N\times N} & \mathbf{0}_{N\times 2} \end{bmatrix} \in \mathbb{R}^{N\times(8N+2)}, \tag{152}$$

Similarly, $\tilde{x}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||}$ can be extracted from $\tilde{\mathbf{x}}$ via

$$\tilde{x}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||} = \boldsymbol{\Upsilon}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||} \tilde{\mathbf{x}}, \tag{153}$$

where

$$\boldsymbol{\Upsilon}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||} = \begin{bmatrix} \mathbf{0}_{1\times 8N} & 1 & 0 \end{bmatrix} \in \mathbb{R}^{1\times(8N+2)} \tag{154}$$

resulting in the SOC constraint in the form described by Eq. (2)

$$||\boldsymbol{\Upsilon}_{\boldsymbol{\kappa}_{\mathbf{a}_R}} \tilde{\mathbf{x}}|| \leq \boldsymbol{\Upsilon}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||} \tilde{\mathbf{x}}. \tag{155}$$

4 Successive Iterations

The first iteration of the trajectory optimization algorithm in Ref. [1] is performed by solving Problem 4. All subsequent iterations are performed by solving Problem 5. This chapter outlines the details involved in implementing Problem 5 in MATLAB using the built-in *coneprog.m* solver.

As described in Sec. 3, Problem 4 is solved as a fixed final time problem, which introduces artificial infeasibility that is circumvented by relaxation of the dynamics, via introduction of the synthetic acceleration term. Subsequent iterations are solved as a free final time problem, which enables the generation of dynamically feasible trajectories. Unlike Problem 4, the size of the time segment, Δt becomes an optimization variable in Problem 5.

Furthermore, in Problem 5 the dynamics are linearized about the trajectory obtained from the previous iteration yielding more accurate solutions so long as the trajectory stays close to the one used in linearization. To ensure that the trajectory of a particular iteration does not stray too far from the reference trajectory used to linearize the dynamics, trust regions are added that penalize large changes in time segment size and thrust commands between iterations shown in Eqs. 51) and 52) of Ref. [1].

4.1 Dynamics

For convenience to the reader, the discretized dynamics of Problem 5 is represented in Eqs. (156) to (167). In order to linearize the nonlinear dynamics described by Eqs. (24)-(26) and (31), first the quantity $\Psi_x[k]$ is defined in Eq. (156)

$$\boldsymbol{\Psi}_x[k] \triangleq \begin{bmatrix} x[k] & x[k+1] \end{bmatrix}^T, \tag{156}$$

which describes the value of a quantity at two adjacent nodes. This is required due to the first order hold assumption made on the acceleration terms, $\mathbf{a}[k]$, which results in the state evolution of the vehicle depending on information both at the beginning and end of a time segment. Equation (157) shows a compact representation of the state, control, and slack variables that the dynamics are dependent on

$$\boldsymbol{\Psi}[k] = \begin{bmatrix} \Delta t & \boldsymbol{\Psi}_m^T[k] & \boldsymbol{\Psi}_{\boldsymbol{\Gamma}}^T[k] & \boldsymbol{\Psi}_{\mathbf{v}}^T[k] & \boldsymbol{\Psi}_{\mathbf{T}}^T[k] & \boldsymbol{\Psi}_{\mathbf{a},R}^T[k] \end{bmatrix}^T. \tag{157}$$

The discrete-time nonlinear dynamics of the vehicle are shown in Eqs. (158) to (163)

$$m[k+1] = m[k] + f_m\left(\boldsymbol{\Psi}[k]\right) \tag{158}$$

$$\mathbf{r}[k+1] = \mathbf{r}[k] + \mathbf{f}_{\mathbf{r}}\left(\boldsymbol{\Psi}[k]\right) \tag{159}$$

$$\mathbf{v}[k+1] = \mathbf{v}[k] + \mathbf{f}_{\mathbf{v}}\left(\boldsymbol{\Psi}[k]\right) \tag{160}$$

where

$$f_m\left(\boldsymbol{\Psi}[k]\right) = -\left[\frac{\alpha}{2}\left(\Gamma[k] + \Gamma[k+1]\right) + \dot{m}_{bp}\right]\Delta t, \tag{161}$$

$$\mathbf{f}_{\mathbf{r}}\left(\boldsymbol{\Psi}[k]\right) = \mathbf{v}[k]\Delta t + \frac{1}{3}\left(\mathbf{a}[k] + \frac{1}{2}\mathbf{a}[k+1]\right)\Delta t^2, \tag{162}$$

$$\mathbf{f}_{\mathbf{v}}\left(\boldsymbol{\Psi}[k]\right) = \frac{1}{2}\left(\mathbf{a}[k] + \mathbf{a}[k+1]\right)\Delta t, \tag{163}$$

$$\mathbf{a}[k] = \frac{1}{m[k]}\left(\mathbf{T}[k] - \frac{1}{2}\rho S_D C_D ||\mathbf{v}[k]||\mathbf{v}[k]\right) + \mathbf{a}_R[k] + \mathbf{g}, \tag{164}$$

and functions f_m, $\mathbf{f}_{\mathbf{r}}$ and $\mathbf{f}_{\mathbf{v}}$ represent the dynamics of mass, position, and velocity. The equations given in Eqs. (158)-(160) are then linearized by replacing Eqs. (161)-(163) with first-order Taylor series expansions

about the previous trajectory shown in Eqs. (165) to (167)

$$m_i[k+1] = m_i[k] + f_m\left(\mathbf{\Psi}_{i-1}[k]\right) + \left.\frac{\partial f_m}{\partial \mathbf{\Psi}}\right|_{\mathbf{\Psi}_{i-1}[k]} \delta\mathbf{\Psi}_i[k] \tag{165}$$

$$\mathbf{r}_i[k+1] = \mathbf{r}_i[k] + \mathbf{f}_\mathbf{r}\left(\mathbf{\Psi}_{i-1}[k]\right) + \left.\frac{\partial \mathbf{f}_\mathbf{r}}{\partial \mathbf{\Psi}}\right|_{\mathbf{\Psi}_{i-1}[k]} \delta\mathbf{\Psi}_i[k] \tag{166}$$

$$\mathbf{v}_i[k+1] = \mathbf{v}_i[k] + \mathbf{f}_\mathbf{v}\left(\mathbf{\Psi}_{i-1}[k]\right) + \left.\frac{\partial \mathbf{f}_\mathbf{v}}{\partial \mathbf{\Psi}}\right|_{\mathbf{\Psi}_{i-1}[k]} \delta\mathbf{\Psi}_i[k] \tag{167}$$

where $\delta\mathbf{\Psi}_i[k] = \mathbf{\Psi}_i[k] - \mathbf{\Psi}_{i-1}[k]$ describes how the trajectory has changed at the k^{th} node between the i^{th} and $(i-1)^{\text{th}}$ iterations. These dynamics involve evaluating f_m, $\mathbf{f}_\mathbf{r}$ and $\mathbf{f}_\mathbf{v}$ and their partial derivatives about the previous trajectory, represented by $\mathbf{\Psi}_{i-1}[k]$. Because of the inclusion of Δt in $\mathbf{\Psi}[k]$, the equations are linearized about the size of the time segments. This enables Δt to be included as an optimization variable. As the number of time segments, N, is fixed, changing Δt is equivalent to changing the final time. Through successive iterations, the final time is adjusted such that no use of synthetic acceleration is required and final optimized trajectory is therefore, dynamically feasible.

Using Eq. (157), the partial derivative of an arbitrary function, f_x, with respect to the state, control, and slack variables are shown in Eq. (168)

$$\frac{\partial f_x\left(\mathbf{\Psi}[k]\right)}{\partial \mathbf{\Psi}} = \begin{bmatrix} \frac{\partial f_x}{\partial \Delta t} & \frac{\partial f_x}{\partial \mathbf{\Psi}_m} & \frac{\partial f_x}{\partial \mathbf{\Psi}_\Gamma} & \frac{\partial f_x}{\partial \mathbf{\Psi}_\mathbf{v}} & \frac{\partial f_x}{\partial \mathbf{\Psi}_\mathbf{T}} & \frac{\partial f_x}{\partial \mathbf{\Psi}_{\mathbf{a}_R}}, \end{bmatrix} \tag{168}$$

Using the definition for $\delta\Psi$, the last last terms on the RHS of Eqs. (165) to (167) that involves the partials can be expanded and shown in Eq. (169),

$$\delta x_i[k] = x_i[k] - x_{i-1}[k]. \tag{169}$$

For clarity and implementation considerations, Eqs. (165) to (167) can be expanded as follows

$$\begin{aligned} \begin{bmatrix} m_i[k+1] \\ \mathbf{r}_i[k+1] \\ \mathbf{v}_i[k+1] \end{bmatrix} &= \begin{bmatrix} m_i[k] \\ \mathbf{r}_i[k] \\ \mathbf{v}_i[k] \end{bmatrix} + \begin{bmatrix} f_m\left(\mathbf{\Psi}_{i-1}[k]\right) \\ \mathbf{f}_\mathbf{r}\left(\mathbf{\Psi}_{i-1}[k]\right) \\ \mathbf{f}_\mathbf{v}\left(\mathbf{\Psi}_{i-1}[k]\right) \end{bmatrix} \\ &+ \begin{bmatrix} \frac{\partial f_m}{\partial \Delta t} & \frac{\partial f_m}{\partial \mathbf{\Psi}_m} & \frac{\partial f_m}{\partial \mathbf{\Psi}_\Gamma} & \frac{\partial f_m}{\partial \mathbf{\Psi}_\mathbf{v}} & \frac{\partial f_m}{\partial \mathbf{\Psi}_\mathbf{T}} & \frac{\partial f_m}{\partial \mathbf{\Psi}_{\mathbf{a},R}} \\ \frac{\partial \mathbf{f}_\mathbf{r}}{\partial \Delta t} & \frac{\partial \mathbf{f}_\mathbf{r}}{\partial \mathbf{\Psi}_m} & \frac{\partial \mathbf{f}_\mathbf{r}}{\partial \mathbf{\Psi}_\Gamma} & \frac{\partial \mathbf{f}_\mathbf{r}}{\partial \mathbf{\Psi}_\mathbf{v}} & \frac{\partial \mathbf{f}_\mathbf{r}}{\partial \mathbf{\Psi}_\mathbf{T}} & \frac{\partial \mathbf{f}_\mathbf{r}}{\partial \mathbf{\Psi}_{\mathbf{a},R}} \\ \frac{\partial \mathbf{f}_\mathbf{v}}{\partial \Delta t} & \frac{\partial \mathbf{f}_\mathbf{v}}{\partial \mathbf{\Psi}_m} & \frac{\partial \mathbf{f}_\mathbf{v}}{\partial \mathbf{\Psi}_\Gamma} & \frac{\partial \mathbf{f}_\mathbf{v}}{\partial \mathbf{\Psi}_\mathbf{v}} & \frac{\partial \mathbf{f}_\mathbf{v}}{\partial \mathbf{\Psi}_\mathbf{T}} & \frac{\partial \mathbf{f}_\mathbf{v}}{\partial \mathbf{\Psi}_{\mathbf{a},R}} \end{bmatrix} \begin{bmatrix} \delta\Delta t_i \\ \delta\mathbf{\Psi}_{m,i}[k] \\ \delta\mathbf{\Psi}_{\Gamma,i}[k] \\ \delta\mathbf{\Psi}_{\mathbf{v},i}[k] \\ \delta\mathbf{\Psi}_{\mathbf{T},i}[k] \\ \delta\mathbf{\Psi}_{\mathbf{a}_R,i}[k] \end{bmatrix}, \end{aligned} \tag{170}$$

an expression for the linearized dynamics for the full state vector of the vehicle where the partial derivatives are evaluated at $\mathbf{\Psi}_{i-1}[k]$. Furthermore, the equation above can be further expanded by grouping terms that

involve the states, control, and slack variables

$$\begin{aligned}
\begin{bmatrix} m_i[k+1] \\ \mathbf{r}_i[k+1] \\ \mathbf{v}_i[k+1] \end{bmatrix} = \begin{bmatrix} m_i[k] \\ \mathbf{r}_i[k] \\ \mathbf{v}_i[k] \end{bmatrix} &+ \begin{bmatrix} f_m\left(\mathbf{\Psi}_{i-1}[k]\right) \\ \mathbf{f_r}\left(\mathbf{\Psi}_{i-1}[k]\right) \\ \mathbf{f_v}\left(\mathbf{\Psi}_{i-1}[k]\right) \end{bmatrix} \\
&+ \begin{bmatrix} \frac{\partial f_m}{\partial m[k]} & \mathbf{0} & \frac{\partial f_m}{\partial \mathbf{v}[k]} \\ \frac{\partial \mathbf{f_r}[k]}{\partial m[k]} & \mathbf{0} & \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{v}[k]} \\ \frac{\partial \mathbf{f_v}}{\partial m[k]} & \mathbf{0} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{v}[k]} \end{bmatrix} \begin{bmatrix} \delta m_i[k] \\ \delta \mathbf{r}_i[k] \\ \delta \mathbf{v}_i[k] \end{bmatrix} \\
&+ \begin{bmatrix} \frac{\partial f_m}{\partial m[k+1]} & \mathbf{0} & \frac{\partial f_m}{\partial \mathbf{v}[k+1]} \\ \frac{\partial \mathbf{f_r}[k]}{\partial m[k+1]} & \mathbf{0} & \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{v}[k+1]} \\ \frac{\partial \mathbf{f_v}}{\partial m[k+1]} & \mathbf{0} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{v}[k+1]} \end{bmatrix} \begin{bmatrix} \delta m_i[k+1] \\ \delta \mathbf{r}_i[k+1] \\ \delta \mathbf{v}_i[k+1] \end{bmatrix} \\
&+ \begin{bmatrix} \frac{\partial f_m}{\partial \mathbf{T}[k]} & \frac{\partial f_m}{\partial \mathbf{T}[k+1]} \\ \frac{\partial \mathbf{f_r}}{\partial \mathbf{T}[k]} & \frac{\partial \mathbf{f_r}}{\partial \mathbf{T}[k+1]} \\ \frac{\partial \mathbf{f_v}}{\partial \mathbf{T}[k]} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{T}[k+1]} \end{bmatrix} \begin{bmatrix} \delta \mathbf{T}_i[k] \\ \delta \mathbf{T}_i[k+1] \end{bmatrix} \\
&+ \begin{bmatrix} \frac{\partial f_m}{\partial \Delta t} & \frac{\partial f_m}{\partial \Gamma[k]} & \frac{\partial f_m}{\partial \Gamma[k+1]} & \frac{\partial f_m}{\partial \mathbf{a}_R[k]} & \frac{\partial f_m}{\partial \mathbf{a}_R[k+1]} \\ \frac{\partial \mathbf{f_r}}{\partial \Delta t} & \frac{\partial \mathbf{f_r}}{\partial \Gamma[k]} & \frac{\partial \mathbf{f_r}}{\partial \Gamma[k+1]} & \frac{\partial \mathbf{f_r}}{\partial \mathbf{a}_R[k]} & \frac{\partial \mathbf{f_r}}{\partial \mathbf{a}_R[k+1]} \\ \frac{\partial \mathbf{f_v}}{\partial \Delta t} & \frac{\partial \mathbf{f_v}}{\partial \Gamma[k]} & \frac{\partial \mathbf{f_v}}{\partial \Gamma[k+1]} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{a}_R[k]} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{a}_R[k+1]} \end{bmatrix} \begin{bmatrix} \delta \Delta t_i \\ \delta \Gamma_i[k] \\ \delta \Gamma_i[k+1] \\ \delta \mathbf{a}_{R,i}[k] \\ \delta \mathbf{a}_{R,i}[k+1] \end{bmatrix}. \qquad (171)
\end{aligned}$$

Defining the various matrices involving the partials:

$$\hat{\mathbf{A}}_{1,i}[k] = \begin{bmatrix} \frac{\partial f_m}{\partial m[k]} & \mathbf{0} & \frac{\partial f_m}{\partial \mathbf{v}[k]} \\ \frac{\partial \mathbf{f_r}[k]}{\partial m[k]} & \mathbf{0} & \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{v}[k]} \\ \frac{\partial \mathbf{f_v}}{\partial m[k]} & \mathbf{0} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{v}[k]} \end{bmatrix} \in \mathbb{R}^{7\times 7}, \qquad (172)$$

$$\hat{\mathbf{A}}_{2,i}[k] = \begin{bmatrix} \frac{\partial f_m}{\partial m[k+1]} & \mathbf{0} & \frac{\partial f_m}{\partial \mathbf{v}[k+1]} \\ \frac{\partial \mathbf{f_r}[k]}{\partial m[k+1]} & \mathbf{0} & \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{v}[k+1]} \\ \frac{\partial \mathbf{f_v}}{\partial m[k+1]} & \mathbf{0} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{v}[k+1]} \end{bmatrix} \in \mathbb{R}^{7\times 7}, \qquad (173)$$

$$\hat{\mathbf{B}}_i[k] = \begin{bmatrix} \frac{\partial f_m}{\partial \mathbf{T}[k]} & \frac{\partial f_m}{\partial \mathbf{T}[k+1]} \\ \frac{\partial \mathbf{f_r}}{\partial \mathbf{T}[k]} & \frac{\partial \mathbf{f_r}}{\partial \mathbf{T}[k+1]} \\ \frac{\partial \mathbf{f_v}}{\partial \mathbf{T}[k]} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{T}[k+1]} \end{bmatrix} \in \mathbb{R}^{7\times 6}, \qquad (174)$$

$$\hat{\mathbf{F}}_i[k] = \begin{bmatrix} \frac{\partial f_m}{\partial \Delta t} & \frac{\partial f_m}{\partial \Gamma[k]} & \frac{\partial f_m}{\partial \Gamma[k+1]} & \frac{\partial f_m}{\partial \mathbf{a}_R[k]} & \frac{\partial f_m}{\partial \mathbf{a}_R[k+1]} \\ \frac{\partial \mathbf{f_r}}{\partial \Delta t} & \frac{\partial \mathbf{f_r}}{\partial \Gamma[k]} & \frac{\partial \mathbf{f_r}}{\partial \Gamma[k+1]} & \frac{\partial \mathbf{f_r}}{\partial \mathbf{a}_R[k]} & \frac{\partial \mathbf{f_r}}{\partial \mathbf{a}_R[k+1]} \\ \frac{\partial \mathbf{f_v}}{\partial \Delta t} & \frac{\partial \mathbf{f_v}}{\partial \Gamma[k]} & \frac{\partial \mathbf{f_v}}{\partial \Gamma[k+1]} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{a}_R[k]} & \frac{\partial \mathbf{f_v}}{\partial \mathbf{a}_R[k+1]} \end{bmatrix} \in \mathbb{R}^{7\times 9}, \qquad (175)$$

where all partial derivatives are evaluated about $\mathbf{\Psi}_{i-1}[k]$, representing a linearization about the trajectory obtained from the previous iteration. Expressions of all the Jacobians are available in the Appendix. Substituting these definitions allows the dynamics to be expressed as,

$$\mathbf{x}_i[k+1] = \mathbf{x}_i[k] + f_x\left(\mathbf{\Psi}_{i-1}\right) + \hat{\mathbf{A}}_{1,i}[k]\delta\mathbf{x}_i[k] + \hat{\mathbf{A}}_{2,i}[k]\delta\mathbf{x}[k+1] + \hat{\mathbf{B}}_i[k]\delta\mathbf{u}_i[k] + \hat{\mathbf{F}}_i\delta\mathbf{p}_i[k], \qquad (176)$$

where,

$$\mathbf{x}[k] = \begin{bmatrix} m[k] & \mathbf{r}[k]^T & \mathbf{v}[k]^T \end{bmatrix}^T, \qquad (177)$$

$$\mathbf{u}[k] = \begin{bmatrix} \mathbf{T}[k]^T & \mathbf{T}[k+1]^T \end{bmatrix}^T, \qquad (178)$$

$$\mathbf{p}[k] = \begin{bmatrix} \Delta t & \Gamma[k] & \Gamma[k+1] & \mathbf{a}_R[k]^T & \mathbf{a}_R[k+1]^T. \end{bmatrix} \qquad (179)$$

After applying the definition for $\delta\mathbf{\Psi}$, the dynamics become

$$\begin{aligned}
\mathbf{x}_i[k+1] =& \mathbf{x}_i[k] + f_x\left(\mathbf{\Psi}_{i-1}\right) + \hat{\mathbf{A}}_{1,i}[k]\mathbf{x}_i[k] + \hat{\mathbf{A}}_{2,i}[k]\mathbf{x}_i[k+1] + \hat{\mathbf{B}}_i[k]\mathbf{u}_i[k] + \hat{\mathbf{F}}_i\mathbf{p}_i[k] \\
&- \hat{\mathbf{A}}_{1,i}[k]\mathbf{x}_{i-1}[k] - \hat{\mathbf{A}}_{2,i}[k]\mathbf{x}_{i-1}[k+1] - \hat{\mathbf{B}}_i[k]\mathbf{u}_{i-1}[k] - \hat{\mathbf{F}}_i\mathbf{p}_{i-1}[k]. \qquad (180)
\end{aligned}$$

All terms that are purely a function of the previous trajectory are grouped into

$$\hat{\mathbf{w}}_i[k] = f_x\left(\mathbf{\Psi}_{i-1}\right) - \hat{\mathbf{A}}_{1,i}[k]\mathbf{x}_{i-1}[k] - \hat{\mathbf{A}}_{2,i}[k]\mathbf{x}_{i-1}[k+1] - \hat{\mathbf{B}}_i[k]\mathbf{u}_{i-1}[k] - \hat{\mathbf{F}}_i\mathbf{p}_{i-1}[k], \tag{181}$$

which, after substituting into Eq. (180) yields,

$$\mathbf{x}_i[k+1] = \left(\mathbf{1} + \hat{\mathbf{A}}_{1,i}[k]\right)\mathbf{x}_i[k] + \hat{\mathbf{A}}_{2,i}[k]\mathbf{x}_i[k+1] + \hat{\mathbf{B}}_i[k]\mathbf{u}_i[k] + \hat{\mathbf{F}}_i\mathbf{p}_i[k] + \hat{\mathbf{w}}_i[k]. \tag{182}$$

The final form of the local dynamics is shown in Eq. (183)

$$\mathbf{x}_i[k+1] = \mathbf{A}_i[k]\mathbf{x}_i[k] + \mathbf{B}_i[k]\mathbf{u}_i[k] + \mathbf{F}_i[k]\mathbf{p}_i[k] + \mathbf{w}_i[k], \tag{183}$$

where

$$\mathbf{A}_i[k] = \left(\mathbf{1} - \hat{\mathbf{A}}_{2,i}[k]\right)^{-1}\left(\mathbf{1} + \hat{\mathbf{A}}_{1,i}[k]\right), \tag{184}$$

$$\mathbf{B}_i[k] = \left(\mathbf{1} - \hat{\mathbf{A}}_{2,i}[k]\right)^{-1}\hat{\mathbf{B}}_i[k], \tag{185}$$

$$\mathbf{F}_i[k] = \left(\mathbf{1} - \hat{\mathbf{A}}_{2,i}[k]\right)^{-1}\hat{\mathbf{F}}_i[k], \tag{186}$$

$$\mathbf{w}_i[k] = \left(\mathbf{1} - \hat{\mathbf{A}}_{2,i}[k]\right)^{-1}\hat{\mathbf{w}}_i[k]. \tag{187}$$

These dynamics are used to propagate the state of the vehicle from one temporal node to the next for the i^{th} iteration of the optimization algorithm where $i > 1$. Additionally, the local dynamics are used to construct the global dynamics described in Sec. 1) which is required to express the state constraints and embed the dynamics into the optimizer.

4.2 Optimization Problem

Subsequent iterations of the optimization algorithm aim to solve the optimization problem laid out in Problem 5. This has the objective,

$$\min_{\Delta t, \mathbf{T}, \mathbf{\Gamma}} -w_{m,f}m[k_f] + w_{\eta,\Delta t}\eta'_{\Delta t,i} + w_{\eta,\mathbf{T}}||\eta'_{\mathbf{T},i}|| + w_{\kappa,\mathbf{a},R}||\boldsymbol{\kappa}_{\mathbf{a}_R,i}|| \tag{188}$$

where two additional terms, involving $\eta'_{\Delta t,i}$ and $\eta'_{\mathbf{T},i}$, are added compared to the objective function for Problem 4 shown in Eq. (68). These are referred to as trust region constraints in order to penalize large changes in Δt and $\mathbf{T}[k]$ between iterations. Note: due to the limitations of *coneprog.m*, $\eta'_{\Delta t,i}$ and $\eta'_{\mathbf{T},i}$ differ from the definitions of $\eta_{\Delta t,i}$ and $\eta_{\mathbf{T},i}$ in Ref. [1] as discussed in Sec. 4.2.4.

For subsequent iterations, the vector containing all of the free parameters over which the objective function is optimized is $\tilde{\mathbf{x}}$ which consists of

$$\tilde{\mathbf{x}}_i = \begin{bmatrix}\tilde{\mathbf{T}}_i^T & \tilde{\mathbf{\Gamma}}_i^T & \tilde{\mathbf{A}}_{R,i}^T & \boldsymbol{\kappa}_{\mathbf{a}_R,i}^T & \boldsymbol{\eta}_{\mathbf{T},i}'^T & \Delta t_i & \eta'_{\Delta t,i} & \tilde{x}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||,i} & \tilde{x}_{||\boldsymbol{\eta}'_{\mathbf{T}}||,i} & \tilde{x}_{m,k_f,i}\end{bmatrix}^T \in \mathbb{R}^{(9N+5)\times 1}, \tag{189}$$

which includes all of the optimization variables present in Problem 4 and listed in Eq. (69). The size of the time segments, Δt_i, is added as an optimization variable in order to optimize over the duration of the trajectory. The variables $\boldsymbol{\eta}'_{\mathbf{T},i}$ and $\tilde{x}_{||\boldsymbol{\eta}'_{\mathbf{T}}||,i}$ are added in order to include the radii of the trust regions on the thrust commands in the objective function while $\eta'_{\Delta t_i}$ is used similarly for the trust region on the time segment size, Δt_i.

With $\tilde{\mathbf{x}}$ defined, the objective function in Eq. (188) can be written as

$$\mathbf{f}^T\tilde{\mathbf{x}}, \tag{190}$$

where

$$\mathbf{f} = \begin{bmatrix}\mathbf{0}_{1\times(9N+1)} & w_{\eta,\Delta t} & w_{\kappa,\mathbf{a},R} & w_{\eta,\mathbf{T}} & -w_{m,f}\end{bmatrix}^T, \tag{191}$$

which matches the form specified by Eq. (1).

4.2.1 Boundary Condition Constraints

The optimization problem stated in Problem 5 of Ref. [1] is subject to the same boundary condition constraints as Problem 4, used for the first iteration of the algorithm. However, the optimization vector, $\tilde{\mathbf{x}}$, is different for Problem 5 as compared to Problem 4. As a result, the matrices used to extract appropriate variables from the optimization vector are updated to reflect the new form of $\tilde{\mathbf{x}}$ and are expressed as follows:

$$\boldsymbol{\Upsilon}_{\mathbf{U}} = \begin{bmatrix} \boldsymbol{\Upsilon}_{\mathbf{u},0}^T & \cdots & \boldsymbol{\Upsilon}_{\mathbf{u},N-2}^T \end{bmatrix}^T \in \mathbb{R}^{6(N-1)\times(9N+5)}, \tag{192}$$

$$\boldsymbol{\Upsilon}_{\mathbf{u},k} = \begin{bmatrix} \mathbf{0}_{6\times 3k} & \mathbf{1}_{6\times 6} & \mathbf{0}_{6\times(9N-3k-1)} \end{bmatrix} \in \mathbb{R}^{6\times(9N+5)}, \tag{193}$$

$$\boldsymbol{\Upsilon}_{\mathbf{P}} = \begin{bmatrix} \boldsymbol{\Upsilon}_{\mathbf{p},0}^T & \cdots & \boldsymbol{\Upsilon}_{\mathbf{p},N-2}^T \end{bmatrix}^T \in \mathbb{R}^{9(N-1)\times(9N+5)}, \tag{194}$$

$$\boldsymbol{\Upsilon}_{\mathbf{p},k} = \begin{bmatrix} \mathbf{0}_{1\times 9N} & 1 & \mathbf{0}_{1\times 4} \\ \mathbf{0}_{2\times(3N+k)} & \mathbf{1}_{2\times 2} & \mathbf{0}_{2\times(6N-k+3)} \\ \mathbf{0}_{6\times(4N+3k)} & \mathbf{1}_{6\times 6} & \mathbf{0}_{6\times(5N-3k-1)} \end{bmatrix} \in \mathbb{R}^{9\times(9N+5)}, \tag{195}$$

$$\boldsymbol{\Upsilon}_{\Gamma,0} = \begin{bmatrix} \mathbf{0}_{1\times 3N} & 1 & \mathbf{0}_{1\times(6N+4)} \end{bmatrix} \in \mathbb{R}^{1\times(9N+5)}, \tag{196}$$

$$\boldsymbol{\Upsilon}_{\mathbf{T},0} = \begin{bmatrix} \mathbf{1}_{3\times 3} & \mathbf{0}_{3\times(9N+2)} \end{bmatrix} \in \mathbb{R}^{3\times(9N+5)}, \tag{197}$$

$$\boldsymbol{\Upsilon}_{\mathbf{T},k_f} = \begin{bmatrix} \mathbf{0}_{3\times 3(N-1)} & \mathbf{1}_{3\times 3} & \mathbf{0}_{3\times[6N+5]} \end{bmatrix} \in \mathbb{R}^{3\times(9N+5)} \tag{198}$$

$$\boldsymbol{\Upsilon}_{\Gamma,k_f} = \begin{bmatrix} \mathbf{0}_{1\times(4N-1)} & 1 & \mathbf{0}_{1\times(5N+5)} \end{bmatrix} \in \mathbb{R}^{1\times(9N+5)}, \tag{199}$$

and replace the corresponding matrices in Sec. 3.2.1.

4.2.2 Control Constraints

To reflect the change in the form of $\tilde{\mathbf{x}}$ between Problem 4 and Problem 5, the matrices

$$\boldsymbol{\Upsilon}_{\mathbf{T},k} = \begin{bmatrix} \mathbf{0}_{3\times 3k} & \mathbf{1}_{3\times 3} & \mathbf{0}_{(9N-3k+2)} \end{bmatrix} \in \mathbb{R}^{3\times(9N+5)}, \tag{200}$$

$$\boldsymbol{\Upsilon}_{\Gamma,k} = \begin{bmatrix} \mathbf{0}_{1\times(3N+k)} & 1 & \mathbf{0}_{1\times(6N-k+4)} \end{bmatrix} \in \mathbb{R}^{3\times(9N+5)}, \tag{201}$$

$$\boldsymbol{\Upsilon}_{\Delta\Gamma,k} = \begin{bmatrix} \mathbf{0}_{1\times(3N+k)} & 1 & -1 & \mathbf{0}_{1\times(6N-k+3)} \end{bmatrix} \in \mathbb{R}^{1\times(9N+5)}, \tag{202}$$

replace the corresponding matrices in Sec. 3.2.3 while the lower and upper bounds on $\tilde{\mathbf{x}}$ are changed to

$$\begin{bmatrix} -\infty\mathbf{1}_{3N\times 1} \\ T_{\min}\mathbf{1}_{N\times 1} \\ -\infty\mathbf{1}_{(5N+5)\times 1} \end{bmatrix} \leq \tilde{\mathbf{x}}_i \leq \begin{bmatrix} \infty\mathbf{1}_{3N\times 1} \\ T_{\max}\mathbf{1}_{N\times 1} \\ \infty\mathbf{1}_{(5N+5)\times 1} \end{bmatrix}, \tag{203}$$

for all iterations following the first.

4.2.3 Objective Function Constraints

For Problem 5, the corresponding matrices in Sec. 3.2.4 are replaced with

$$\boldsymbol{\Upsilon}_{m,k_f} = \begin{bmatrix} \mathbf{0}_{1\times(9N+4)} & 1 \end{bmatrix} \in \mathbb{R}^{1\times(9N+5)}, \tag{204}$$

$$\boldsymbol{\Upsilon}_{\mathbf{a}_R,k} = \begin{bmatrix} \mathbf{0}_{3\times(4N+3k)} & \mathbf{1}_{3\times 3} & \mathbf{0}_{3\times(4N-3k-1)} \end{bmatrix} \in \mathbb{R}^{3\times(9N+5)}, \tag{205}$$

$$\boldsymbol{\Upsilon}_{\kappa_{\mathbf{a}_R},k} = \begin{bmatrix} \mathbf{0}_{1\times(7N+k)} & 1 & \mathbf{0}_{1\times(2N-k+4)} \end{bmatrix} \in \mathbb{R}^{1\times(9N+5)}, \tag{206}$$

$$\boldsymbol{\Upsilon}_{\boldsymbol{\kappa}_{\mathbf{a}_R}} = \begin{bmatrix} \mathbf{0}_{N\times 7N} & \mathbf{1}_{N\times N} & \mathbf{0}_{N\times(N+5)} \end{bmatrix} \in \mathbb{R}^{N\times(9N+5)}, \tag{207}$$

$$\boldsymbol{\Upsilon}_{||\boldsymbol{\kappa}_{\mathbf{a}_R}||} = \begin{bmatrix} \mathbf{0}_{1\times(9N+2)} & 1 & \mathbf{0}_{1\times 2} \end{bmatrix} \in \mathbb{R}^{1\times(9N+5)}, \tag{208}$$

to reflect the change in $\tilde{\mathbf{x}}$ between Problem 4 and Problem 5.

4.2.4 Trust Region Constraints

Time Segment Size, Δt_i The discretization time, Δt_i, is allowed to vary in order to optimize the trajectory as a free-final-time problem. However, if Δt_i deviates too much compared to the time segment size of the previous trajectory, Δt_{i-1}, the linearized dynamics will become increasingly inaccurate and the problem may become unbounded. In order to avoid this issue, the change in the time segment size, $\delta\Delta t_i$

$$\delta\Delta t_i = \Delta t_i - \Delta t_{i-1}, \tag{209}$$

can be bounded through Eq. (51) of Ref. [1] in the form of a quadratic constraint:

$$\delta\Delta t_i^2 \leq \eta_{\Delta t,i} \tag{210}$$

However, *coneprog.m* does not allow for constraints of this form. Therefore, the quadratic constraint is modified to the form

$$||\delta\Delta t_i|| \leq \eta'_{\Delta t,i} \tag{211}$$

The trust region radius, $\eta'_{\Delta t,i}$, can be extract from $\tilde{\mathbf{x}}_i$ as follows

$$\eta'_{\Delta t,i} = \boldsymbol{\Upsilon}_{\eta',\Delta t}\tilde{\mathbf{x}}_i, \tag{212}$$

where

$$\boldsymbol{\Upsilon}_{\eta',\Delta t} = \begin{bmatrix}\mathbf{0}_{1\times(9N+1)} & 1 & \mathbf{0}_{1\times 3}\end{bmatrix} \in \mathbb{R}^{1\times(9N+5)}. \tag{213}$$

The discretization time can be extracted from $\tilde{\mathbf{x}}_i$ in a similar fashion:

$$\Delta t_i = \boldsymbol{\Upsilon}_{\Delta t}\tilde{\mathbf{x}}_i \tag{214}$$

where

$$\boldsymbol{\Upsilon}_{\Delta t} = \begin{bmatrix}\mathbf{0}_{1\times(9N)} & 1 & \mathbf{0}_{1\times 4}\end{bmatrix} \in \mathbb{R}^{1\times(9N+5)} \tag{215}$$

Substituting Eqs. (209), (212), and (214) into (211) leads to the expression,

$$||\boldsymbol{\Upsilon}_{\Delta t}\tilde{\mathbf{x}}_i - \Delta t_{i-1}|| \leq \boldsymbol{\Upsilon}_{\eta',\Delta t}\tilde{\mathbf{x}}_i, \tag{216}$$

which is equivalent to Eq. (211) and in the form of an SOC constraint.

Thrust Commands, $\mathbf{T}_i[k]$ If the thrust commands change too much between iterations, the resultant trajectory from the current iteration could differ significantly from the previous iteration, which the dynamics are linearized about. As a result, the accuracy of the linear model will degrade compared to the nonlinear dynamics. Hence, certain constraints may be violated when the solution is propagated through the nonlinear dynamics.

To prevent the thrust command at a particular node, $\mathbf{T}_i[k]$, from changing too much compared to the value from the previous iteration, $\mathbf{T}_{i-1}[k]$, the difference is bounded by $\eta_{\mathbf{T}}[k]$ shown in Eq. (217):

$$\delta\mathbf{T}_i[k]^T\delta\mathbf{T}_i[k] \leq \eta_{\mathbf{T}}[k] \tag{217}$$

for nodes $k \in [0, N-1]$ where

$$\delta\mathbf{T}_i[k] = \mathbf{T}_i[k] - \mathbf{T}_{i-1}[k]. \tag{218}$$

Again, because *coneprog.m* is not compatible with quadratic constraints, the constraint is modified to an SOC constraint:

$$||\delta\mathbf{T}_i[k]|| \leq \eta'_{\mathbf{T}}[k]. \tag{219}$$

The thrust command at a particular node can be extracted from $\tilde{\mathbf{x}}_i$ through

$$\mathbf{T}_i[k] = \boldsymbol{\Upsilon}_{\mathbf{T},k}\tilde{\mathbf{x}}_i, \tag{220}$$

where $\boldsymbol{\Upsilon}_{\mathbf{T},k}$ is given by Eq. (200). Likewise, the point-wise bound on the thrust command is extracted through

$$\eta'_{\mathbf{T}}[k] = \boldsymbol{\Upsilon}_{\eta'_{\mathbf{T}},k}\tilde{\mathbf{x}}_i \tag{221}$$

where

$$\boldsymbol{\Upsilon}_{\eta'_{\mathbf{T}}} = \begin{bmatrix} \mathbf{0}_{1\times(8N+k)} & 1 & \mathbf{0}_{1\times(N+4-k)} \end{bmatrix}, \tag{222}$$

which enables Eq. (219) to be expressed as

$$||\boldsymbol{\Upsilon}_{\mathbf{T},k}\tilde{\mathbf{x}}_i - \mathbf{T}_{i-1}[k]|| \leq \boldsymbol{\Upsilon}_{\eta'_{\mathbf{T}},k}\tilde{\mathbf{x}}_i, \tag{223}$$

a group of N number of SOC constraints in the form shown in Eq. (2).

In order to include the trust region radii in the objective function, the norm of $\boldsymbol{\eta}'_{\mathbf{T},i}$, the vector containing all of the point-wise trust region radii on the thrust vectors, is bounded by $\tilde{x}_{||\boldsymbol{\eta}'_{\mathbf{T}}||}$ through

$$||\boldsymbol{\eta}'_{\mathbf{T},i}|| \leq \tilde{x}_{||\boldsymbol{\eta}'_{\mathbf{T}}||}. \tag{224}$$

The point-wise bounds, $\boldsymbol{\eta}_{\mathbf{T},i}$ can be extract through $\tilde{\mathbf{x}}_i$ as follows

$$\boldsymbol{\eta}'_{\mathbf{T},i} = \boldsymbol{\Upsilon}_{\boldsymbol{\eta}'_{\mathbf{T}}}\tilde{\mathbf{x}}_i \tag{225}$$

where

$$\boldsymbol{\Upsilon}_{\boldsymbol{\eta}'_{\mathbf{T}}} = \begin{bmatrix} \mathbf{0}_{N\times 8N} & \mathbf{1}_{N\times N} & \mathbf{0}_{N\times 5} \end{bmatrix} \tag{226}$$

similarly, $\tilde{x}_{||\boldsymbol{\eta}'_{\mathbf{T}}||}$ can be extracted through $\tilde{\mathbf{x}}_i$ as follows

$$\tilde{x}_{||\boldsymbol{\eta}'_{\mathbf{T}}||} = \boldsymbol{\Upsilon}_{||\boldsymbol{\eta}'_{\mathbf{T}}||}\tilde{\mathbf{x}}_i \tag{227}$$

where

$$\boldsymbol{\Upsilon}_{||\boldsymbol{\eta}'_{\mathbf{T}}||} = \begin{bmatrix} \mathbf{0}_{1\times(9N+3)} & 1 & 0 \end{bmatrix} \tag{228}$$

which allows the constraint in Eq. (224) to be written as

$$||\boldsymbol{\Upsilon}_{\boldsymbol{\eta}'_{\mathbf{T}}}\tilde{\mathbf{x}}_i|| \leq \boldsymbol{\Upsilon}_{||\boldsymbol{\eta}'_{\mathbf{T}}||}\tilde{\mathbf{x}}_i, \tag{229}$$

a single SOC constraint of the form shown in Eq. (2).

5 Results

For verification purposes, independently-generated simulation results of the example provided in Ref. [1] are shown. Coordinate system, simulation parameters, initial/terminal conditions, state/control constraints, and cost function weights are provided in Sec.IV.A of Ref. [1]. Similar to Ref. [1], 10 SCP iterations and 30 discretization points are used. Note: to make the results match the paper, the specified min/max values of the provided vacuum thrust were multiplied by a factor of 0.83. Furthermore, due to the modifications of the Δt_i and $\mathbf{T}_i[k]$ trust regions from a form of a quadratic to SOC constraint shown in Eqs. (211) and (219), the corresponding weights $w_{\eta,\Delta t}$ and $w_{\eta,\mathbf{T}}$ where changed to 0.1 and 1e-6 respectively.

Results from solving Problem 4, not available in Ref. [1], are shown in Figs. 3 to 6. Figure 3 is a 3-D profile of the trajectory with the linearized trajectory with scaled thrust vector at each temporal node shown in blue. Subsequently, the sequence of thrust inputs derived from the linear dynamics were propagated through the nonlinear dynamics and illustrated by the pink dashed curve. Due to the large errors in the linear approximation, the nonlinear results fail to reach the desired terminal condition with the fixed time-of-flight of 15 s. Components of the position and velocity vectors are shown in Fig. 4. Magnitude of thrust and the slack variable, Γ, is shown in the top plot of Fig. 5 and the bottom plot illustrates the magnitude of the synthetic acceleration with the corresponding slack variable, $\kappa_{a,R}[k]$. The trajectory is dynamically infeasible without the additional "assistance" from the synthetic accelerations. Mass time history is shown in Fig. 6.

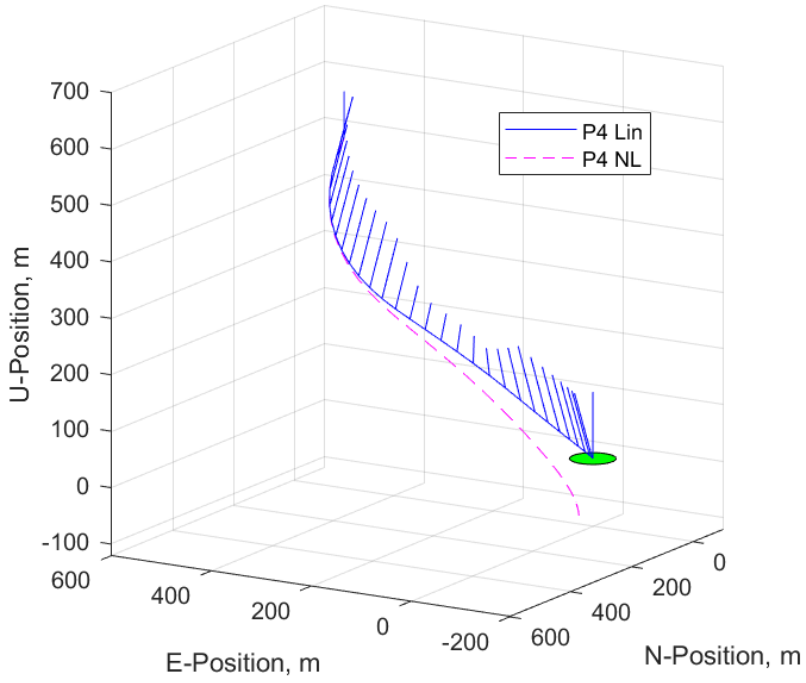


Figure 3: Problem 4 3-D Trajectory

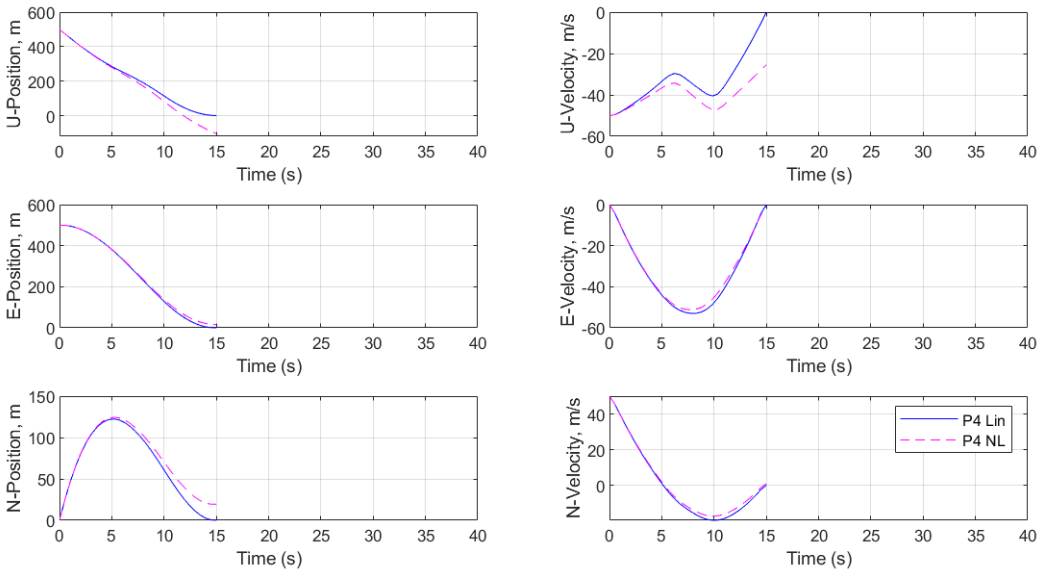


Figure 4: Problem 4 Up, East, and North components of position and velocity

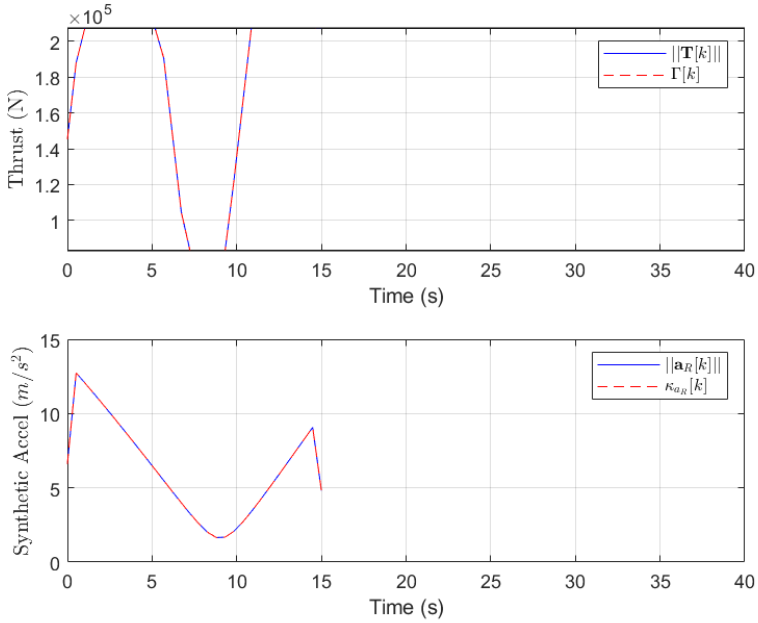


Figure 5: Problem 4 Thrust Magnitude and Synthetic Accel

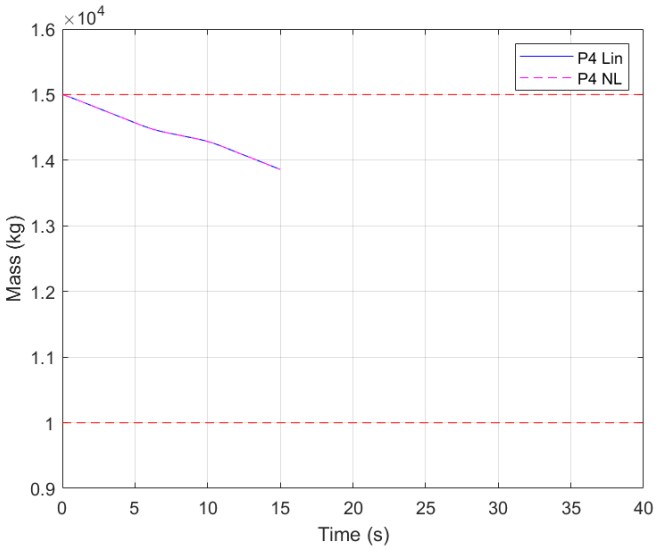


Figure 6: Problem 4 Vehicle Mass

With the availability of an initial trajectory, procedures laid out in Sec. 4 were carried out to obtain the optimal trajectory. Results from the final iteration of Problem 5 are shown in Figs. 7 to 10. Note: weighting term on the thrust trust region differs from the value provided in paper due to the limitations of *coneprog.m*. Figure 7 is a 3-D profile of the trajectory with the linearized dynamics and corresponding scaled thrust vector at each temporal node shown in red. Subsequently, the sequence of thrust inputs derived from the linear dynamics were propagated through the nonlinear dynamics and illustrated by the black dashed curve. It is apparent that at the final iteration, the linear system closely approximates the nonlinear dynamics. Components of the position and velocity vectors are shown in Fig. 8. The resultant optimal trajectory closely matches Figs. 1 and 2 of Ref. [1] with a time-of-flight of 37.75 s. The noticeable "hop" manuever is apparent in the vertical channel.

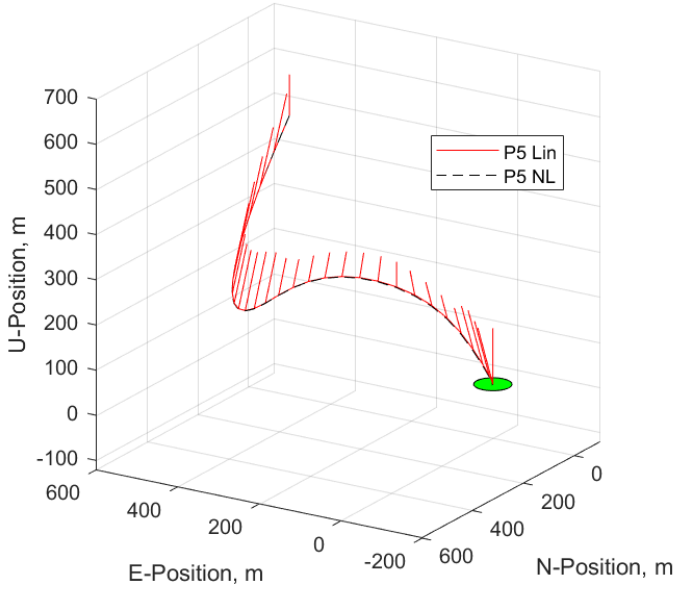


Figure 7: Problem 5 3-D Trajectory

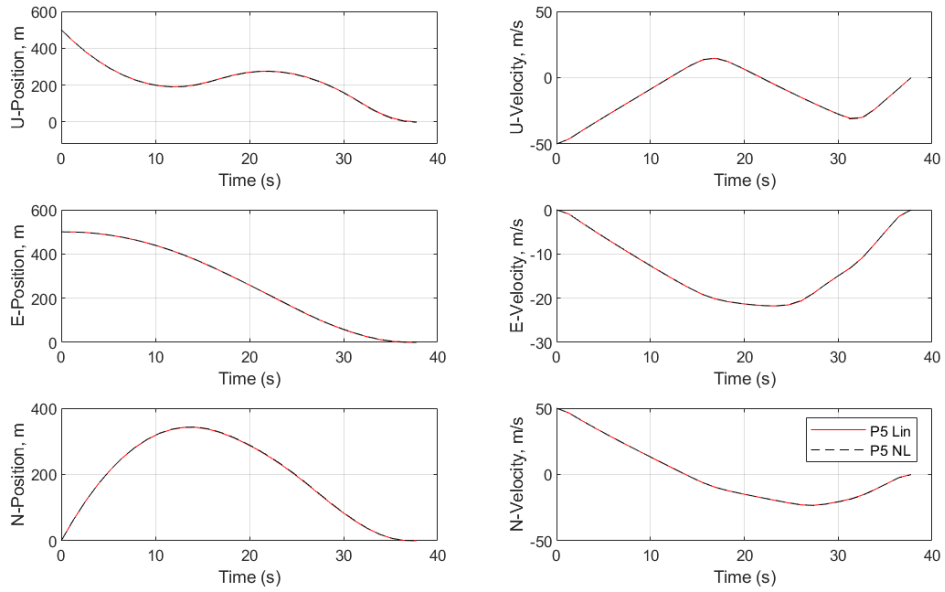


Figure 8: Problem 5 Up, East, and North components of position and velocity

Plots of the thrust magnitude, thrust magnitude rate, synthetic acceleration, tilt angle, azimuth angle, and glidescope angle are shown in Fig. 9. The "bang-coast-bang" nature of the throttle profile is consistent with the top left plot of Fig. 3 in Ref. [1]. The tilt angle starts and ends at the vertical orientation and operates at the maximum prescribed limit of 15 deg for majority of the trajectory with the exception of the slight dip around 27 sec. The throttle rate and azimuth profiles follows the paper with the exception that the throttle rate doesn't quite reach the saturation limit of 100 kN/s at around 30 sec. The glidescope angle, not shown in the paper, stays below the prescribed limit of 80 deg. The synthetic acceleration is driven close to zero. Mass time history is shown in Fig. 10. The final mass at touchdown is 12381 kg and appears to be consistent with Fig. 4 of Ref. [1]

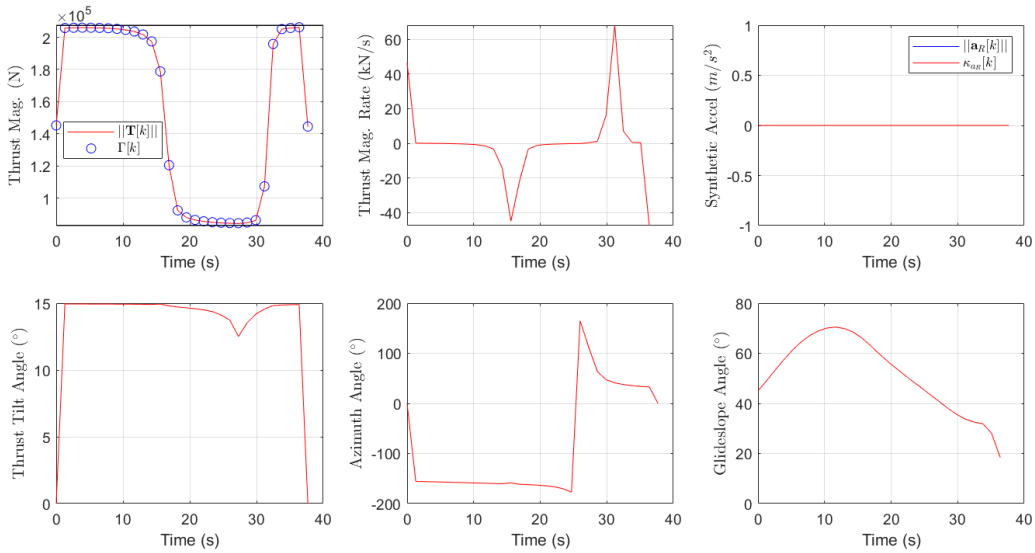


Figure 9: Problem 5 Thrust Mag., Thrust Mag. Rate, Syn. Accel, Tilt Angle, Azimuth, Glidescope

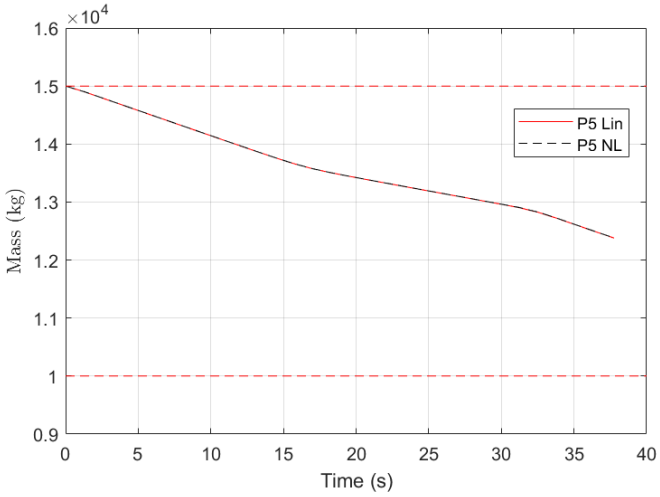


Figure 10: Problem 5 Vehicle Mass

6 Comparison with Quickshot^{TM}

To compare the SCP results to a traditional direct optimization approach using nonlinear programming methods, a QuickShot^{TM} simulation was developed to solve the same continuous-time fuel-optimal powered landing problem described by Problem 1 of Ref. [1]. QuickShot^{TM} [15] is a commercial trajectory optimization software developed by SpaceWorks Enterprises, Inc. which utilizes multi-threaded computing techniques and stochastic optimization routines to solve a variety of 3-4 DoF nonlinear trajectory optimization problems. Standout features include no initial guess being required and the software's ability to find global optimum solutions in multi-modal spaces.

Simulation inputs from Sec.IV.A of Ref. [1] were used along with assumptions of a flat Earth and constant gravity vector. Similar to the SCP results, the min/max values of vacuum thrust provided in Ref. [1] were scaled by a factor of 0.83. The trajectory was discretized into 45 segments with $\Delta t = 1$s for all segments. The vehicle's thrust magnitude, tilt, and azimuth angle were optimized at the beginning and end of each time segment. Furthermore, thrust and attitude angles (using direction of thrust as a proxy) were assumed to vary linearly across each segment. A control variable was used to trigger the end of the simulation as a function of vehicle mass effectively allowing the time of flight to be optimized. While QuickShot^{TM} doesn't require an initial guess, the user must provide bounds for each control variable. Table 1 provides the bounds used for thrust, attitude, and final mass control variables. Table 2 summarizes the objective function and constraints.

Table 1: Control Variable Bounds

Parameter	Minimum	Maximum	Units
$\|\|\boldsymbol{T}\|\|$	83	207.5	KN
θ	0	15	deg
Az	0	360	deg
$m(t_f)$	12000	15000	kg

Table 2: Objective Function and Constraints

Objective Function
Maximize $m(t_f)$
Terminal Constraints
$\boldsymbol{r(t_f) = 0}$
$\boldsymbol{v(t_f) = 0}$
$\theta(t_f) = 0$
State Constraints
$\|\|\mathbf{r(t)}\|\| \cos\gamma_{gs} \leq \hat{\mathbf{e}}_u^T \mathbf{r(t)}$, where $\gamma_{gs} = 80$ deg
Control Constraints
-100 kN/s $\leq \frac{d}{dt}\|\|\boldsymbol{T}\|\| \leq$ 100 kN/s

The thrust magnitude and tilt angle control variable bounds were chosen such that the min/max thrust magnitude and tilt from vertical constraints would always be satisfied. Additionally, the vehicle's initial thrust magnitude and tilt angle were set to 145.25 kN and 0 deg respectively. Figure 11 compares the components of the position and velocity vectors from the SCP and QuickShot trajectory solutions. The optimized mass at the final time, $m(t_f)$, is comparable between Quickshot and SCP (12383 kg vs. 12381 kg). QuickShot's time-of-flight was 37.72 s, electing to not use all of the 45 trajectory segments available.

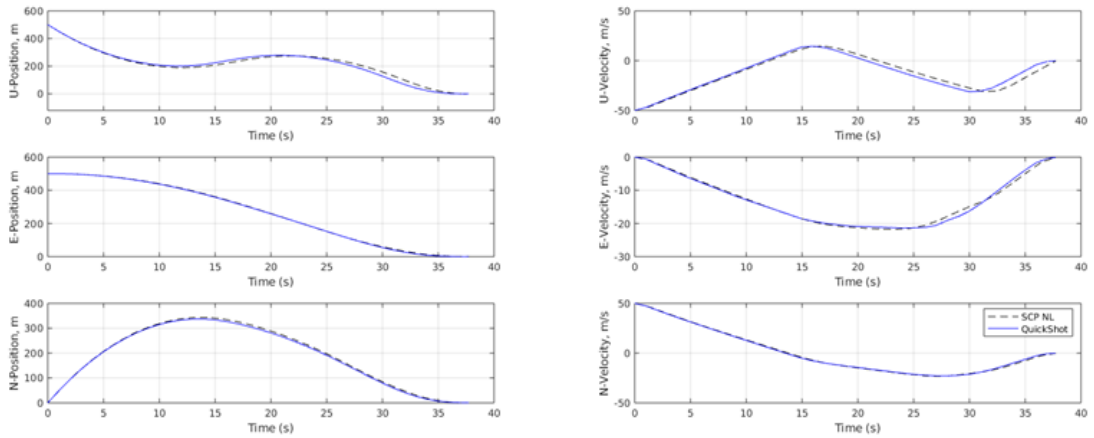


Figure 11: Comparisions of Position and Velocity Profiles

Figure 12 shows a comparison of the control variables, glideslope angle, and vehicle mass time histories. QuickShot exhibits the same "bang-coast-bang" nature in the throttle profile and stays well within the thrust rate saturation limits. The tilt and azimuth angle profiles are comparable between the two approaches with tilt angle holding close to the 15 deg limit for a large portion of the trajectory then turning toward vertical for touchdown. QuickShot's glideslope profile deviates from the SCP solution at around 30 s and exhibits a shallower approach to the landing site.

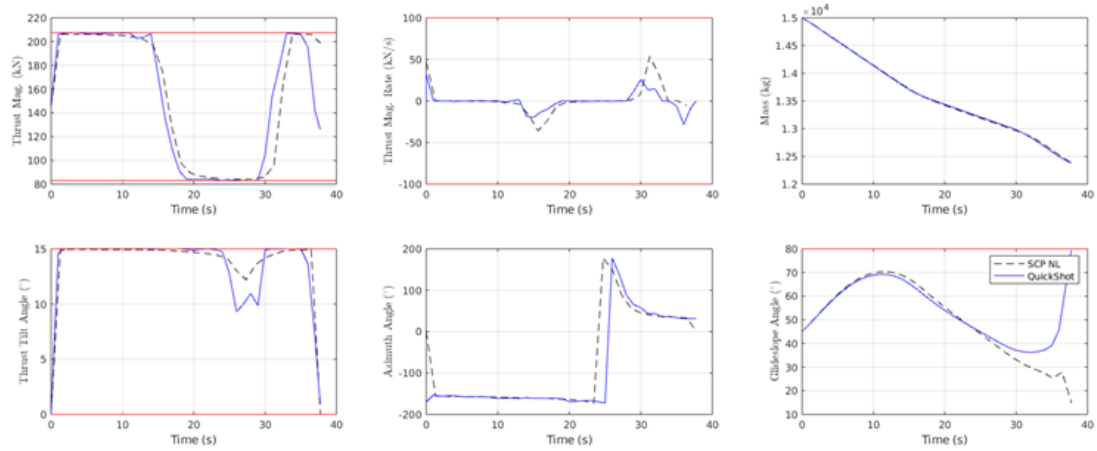


Figure 12: Comparisions Thrust Mag., Thrust Mag. Rate, Syn. Accel, Tilt Angle, Azimuth, Glidescope

One of the main benefits of SCP is the fast convergence characteristics which is ideal for real-time applications. Whereas QuickShot^{TM} is an off-line tool which uses a stochastic optimization routine. The user guide recommends running multiple trials of the same problem with a unique random number seed for each trial and it is the user's responsibility to determine the best combination of trials, runtime cap, and convergence criteria for the problem of interest. Figure 13 shows trial final mass and runtime values with trial 4 being the best solution in the set. Each trial yielded a feasible solution and terminated either due to a runtime cap of 45 minutes or by satisfying a stringent convergence criterion. QuickShot's total runtime was 445 minutes compared to SCP's total runtime of 4 s on a HP EliteBook 840 G7 notebook with four 1.6-4.2 Hz processor.

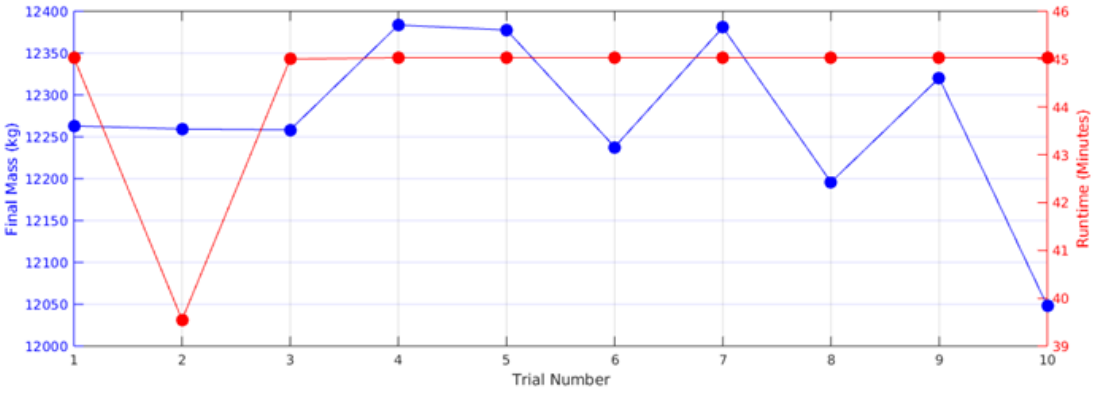


Figure 13: Quickshot^{TM} Convergence Metrics

7 Comparison with E-guidance

In this Section, the SCP solution for the example problem presented in Ref. [1] is compared with the polynomial approach phase guidance used during the Apollo missions [16, 17]. The acceleration profile is restricted to a quadratic form described by Eq. (230). The coefficients can be determined by solving a system of linear equations that involves the current position and velocity, target position, velocity, and acceleration, along with the time-of-flight, t_{go}. For this example, t_{go} was set to the time-of-flight determed by SCP.

$$a = c_o + c_1 t + c_2 t^2 \tag{230}$$

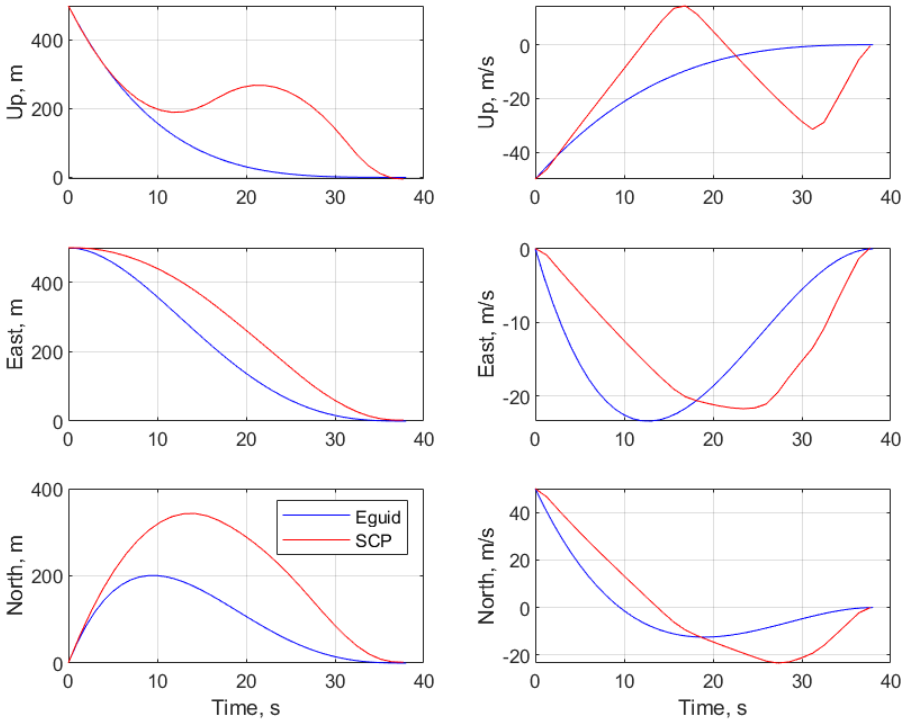


Figure 14: Comparisions of Position and Velocity Profiles

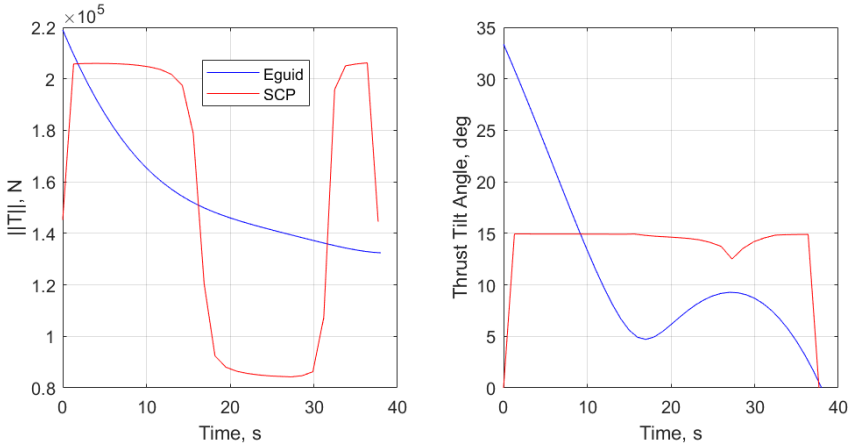


Figure 15: Comparisions of Thrust Magnitude

Position, velocity, thrust magnitude, and tilt angle comparisons are provided in Figs. 14 and 15. It is apparent that early on in the E-guidance solution, the thrust magnitude and tilt angle requirements are violated. This is expected since the solution does not take into consideration state or control constraints. Furthermore, the "bang-coast-bang" thrust profile exhibited by SCP provides a more fuel optimal solution compared to E-guidance.

8 Conclusion

This technical memorandum details the implementation in MATLAB of a successive convexification algorithm for powered descent from the literature, including how the problem is parsed for compatibility with MATLAB's built-in second-order cone programming solver. The MATLAB implementation was used to recreate the results from the original paper then compared to the commercial trajectory optimization software QuickShot^{TM} and Apollo-heritage E-guidance. While QuickShot^{TM} was able to produce nearly the same trajectory and satisfied the state and control constraints, the computation time was orders of magnitude longer than the successive convexification algorithm. E-guidance produced a trajectory using very little computational effort, but was not optimal and violated constraints on the state and control input.

9 Acknowledgements

The authors would like to thank the NASA Engineering Safety Center (NESC) and Human Landing Systems (HLS) Program for supporting this work. The authors also acknowledge Dr. William Elke, Mr. Samuel Buckner and members of the HLS GN&C team for their insight and feedback.

10 Appendix: Partial Derivatives

Acceleration, $\frac{\partial \mathbf{a}[k]}{\partial \mathbf{\Psi}}$

$$\frac{\partial \mathbf{a}[k]}{\partial \Delta t} = \mathbf{0} \tag{231}$$

$$\frac{\partial \mathbf{a}[k]}{\partial \mathbf{\Psi}_m} = \begin{bmatrix} \frac{\partial \mathbf{a}[k]}{\partial m[k]} & \frac{\partial \mathbf{a}[k]}{\partial m[k+1]} \end{bmatrix} \tag{232}$$

$$= \begin{bmatrix} -\frac{1}{m[k]^2}\left(\mathbf{T}[k] - \frac{1}{2}\rho S_D C_D ||\mathbf{v}[k]||\mathbf{v}[k]\right) & \mathbf{0} \end{bmatrix} \tag{233}$$

$$\frac{\partial \mathbf{a}[k]}{\partial \mathbf{\Psi}_{\mathbf{\Gamma}}} = \begin{bmatrix} \frac{\partial \mathbf{a}[k]}{\partial \Gamma[k]} & \frac{\partial \mathbf{a}[k]}{\partial \Gamma[k+1]} \end{bmatrix} \tag{234}$$

$$= \mathbf{0} \tag{235}$$

$$\frac{\partial \mathbf{a}[k]}{\partial \mathbf{\Psi}_{\mathbf{v}}} = \begin{bmatrix} \frac{\partial \mathbf{a}[k]}{\partial \mathbf{v}[k]} & \frac{\partial \mathbf{a}[k]}{\partial \mathbf{v}[k+1]} \end{bmatrix} \tag{236}$$

$$= \begin{bmatrix} \frac{\partial \mathbf{a}[k]}{\partial \mathbf{v}[k]} & \mathbf{0} \end{bmatrix} \tag{237}$$

$$\frac{\partial \mathbf{a}[k]}{\partial \mathbf{v}[k]} = \frac{\partial}{\partial \mathbf{v}[k]}\left(-\frac{1}{2m[k]}\rho S_D C_D ||\mathbf{v}[k]||\mathbf{v}[k]\right) \tag{238}$$

$$= -\frac{1}{2m[k]}\rho S_D C_D \left[\mathbf{v}[k]\left(\frac{\partial}{\partial \mathbf{v}[k]}||\mathbf{v}[k]||\right) + ||\mathbf{v}[k]||\left(\frac{\partial}{\partial \mathbf{v}[k]}\mathbf{v}[k]\right)\right] \tag{239}$$

$$= -\frac{1}{2m[k]}\rho S_D C_D \left(\frac{\mathbf{v}[k]\mathbf{v}^T[k]}{||\mathbf{v}[k]||} + \mathbf{1}||\mathbf{v}[k]||\right) \tag{240}$$

$$\frac{\partial \mathbf{a}[k]}{\partial \mathbf{\Psi}_{\mathbf{T}}} = \begin{bmatrix} \frac{\partial \mathbf{a}[k]}{\partial \mathbf{T}[k]} & \frac{\partial \mathbf{a}[k]}{\partial \mathbf{T}[k+1]} \end{bmatrix} \tag{241}$$

$$= \begin{bmatrix} \frac{\mathbf{1}}{m[k]} & \mathbf{0} \end{bmatrix} \tag{242}$$

$$\frac{\partial \mathbf{a}[k]}{\partial \mathbf{\Psi}_{\mathbf{a},\mathbf{R}}} = \begin{bmatrix} \frac{\partial \mathbf{a}[k]}{\partial \mathbf{a}_R[k]} & \frac{\partial \mathbf{a}[k]}{\partial \mathbf{a}_R[k+1]} \end{bmatrix} \tag{243}$$

$$= \begin{bmatrix} \mathbf{1} & \mathbf{0} \end{bmatrix} \tag{244}$$

Mass Update, $\frac{\partial f_m[k]}{\partial \mathbf{\Psi}}$

$$\frac{\partial f_m[k]}{\partial \Delta t} = -\left[\frac{\alpha}{2}\left(\Gamma[k] + \Gamma[k+1]\right) + \dot{m}_{bp}\right] \tag{245}$$

$$\frac{\partial f_m[k]}{\partial \mathbf{\Psi}_m} = \mathbf{0} \tag{246}$$

$$\frac{\partial f_m[k]}{\partial \mathbf{\Psi}_\Gamma} = \begin{bmatrix} \frac{\partial f_m[k]}{\partial \Gamma[k]} & \frac{\partial f_m[k]}{\partial \Gamma[k+1]} \end{bmatrix} \tag{247}$$

$$= \begin{bmatrix} -\frac{\alpha}{2}\Delta t & -\frac{\alpha}{2}\Delta t \end{bmatrix} \tag{248}$$

$$\frac{\partial f_m[k]}{\partial \mathbf{\Psi}_\mathbf{v}} = \begin{bmatrix} \frac{\partial f_m[k]}{\partial \mathbf{v}[k]} & \frac{\partial f_m[k]}{\partial \mathbf{v}[k+1]} \end{bmatrix} = \mathbf{0} \tag{249}$$

$$\frac{\partial f_m[k]}{\partial \mathbf{\Psi}_\mathbf{T}} = \begin{bmatrix} \frac{\partial f_m[k]}{\partial \mathbf{T}[k]} & \frac{\partial f_m[k]}{\partial \mathbf{T}[k+1]} \end{bmatrix} = \mathbf{0} \tag{250}$$

$$\frac{\partial f_m[k]}{\partial \mathbf{\Psi}_{\mathbf{a},R}} = \begin{bmatrix} \frac{\partial f_m[k]}{\partial \mathbf{a}_R[k]} & \frac{\partial f_m[k]}{\partial \mathbf{a}_R[k+1]} \end{bmatrix} = \mathbf{0} \tag{251}$$

Position Update, $\frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{\Psi}}$

$$\frac{\partial \mathbf{f_r}[k]}{\partial \Delta t} = \mathbf{v}[k] + \frac{2}{3}\left(\mathbf{a}[k] + \frac{1}{2}\mathbf{a}[k+1]\right)\Delta t \tag{252}$$

$$\frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{\Psi}_m} = \begin{bmatrix} \frac{\partial \mathbf{f_r}[k]}{\partial m[k]} & \frac{\partial \mathbf{f_r}[k]}{\partial m[k+1]} \end{bmatrix} \tag{253}$$

$$= \begin{bmatrix} \frac{1}{3}\frac{\partial \mathbf{a}[k]}{\partial m[k]}\Delta t^2 & \frac{1}{6}\frac{\partial \mathbf{a}[k+1]}{\partial m[k+1]}\Delta t^2 \end{bmatrix} \tag{254}$$

$$\frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{\Psi}_\Gamma} = \begin{bmatrix} \frac{\partial \mathbf{f_r}[k]}{\partial \Gamma[k]} & \frac{\partial \mathbf{f_r}[k]}{\partial \Gamma[k+1]} \end{bmatrix} \tag{255}$$

$$= \mathbf{0} \tag{256}$$

$$\frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{\Psi}_\mathbf{v}} = \begin{bmatrix} \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{v}[k]} & \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{v}[k+1]} \end{bmatrix} \tag{257}$$

$$= \begin{bmatrix} \mathbf{1}\Delta t + \frac{1}{3}\frac{\partial \mathbf{a}[k]}{\partial \mathbf{v}[k]}\Delta t^2 & \frac{1}{6}\frac{\partial \mathbf{a}[k+1]}{\partial \mathbf{v}[k+1]}\Delta t^2 \end{bmatrix} \tag{258}$$

$$\frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{\Psi}_\mathbf{T}} = \begin{bmatrix} \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{T}[k]} & \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{T}[k+1]} \end{bmatrix} \tag{259}$$

$$= \begin{bmatrix} \frac{1}{3}\frac{\partial \mathbf{a}[k]}{\partial \mathbf{T}[k]}\Delta t^2 & \frac{1}{6}\frac{\partial \mathbf{a}[k+1]}{\partial \mathbf{T}[k+1]}\Delta t^2 \end{bmatrix} \tag{260}$$

$$\frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{\Psi}_{\mathbf{a}_R}} = \begin{bmatrix} \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{a}_R} & \frac{\partial \mathbf{f_r}[k]}{\partial \mathbf{a}_R} \end{bmatrix} \tag{261}$$

$$= \begin{bmatrix} \frac{1}{3}\frac{\partial \mathbf{a}[k]}{\partial \mathbf{a}_R[k]}\Delta t^2 & \frac{1}{6}\frac{\partial \mathbf{a}[k+1]}{\partial \mathbf{a}_R[k+1]}\Delta t^2 \end{bmatrix} \tag{262}$$

Velocity Update, $\frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{\Psi}}$

$$\frac{\partial \mathbf{f_v}[k]}{\partial \Delta t} = \frac{1}{2}\left(\mathbf{a}[k] + \mathbf{a}[k+1]\right) \tag{263}$$

$$\frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{\Psi}_m} = \begin{bmatrix} \frac{\partial \mathbf{f_v}[k]}{\partial m[k]} & \frac{\partial \mathbf{f_v}[k]}{\partial m[k+1]} \end{bmatrix} \tag{264}$$

$$= \begin{bmatrix} \frac{1}{2}\frac{\partial \mathbf{a}[k]}{\partial m[k]}\Delta t & \frac{1}{2}\frac{\partial \mathbf{a}[k+1]}{\partial m[k+1]}\Delta t \end{bmatrix} \tag{265}$$

$$\frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{\Psi}_\Gamma} = \begin{bmatrix} \frac{\partial \mathbf{f_v}[k]}{\partial \Gamma[k]} & \frac{\partial \mathbf{f_v}[k]}{\partial \Gamma[k+1]} \end{bmatrix} \tag{266}$$

$$= \mathbf{0} \tag{267}$$

$$\frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{\Psi}_\mathbf{v}} = \begin{bmatrix} \frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{v}[k]} & \frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{v}[k+1]} \end{bmatrix} \tag{268}$$

$$= \begin{bmatrix} \frac{1}{2}\frac{\partial \mathbf{a}[k]}{\partial \mathbf{v}[k]}\Delta t & \frac{1}{2}\frac{\partial \mathbf{a}[k+1]}{\partial \mathbf{v}[k+1]}\Delta t \end{bmatrix} \tag{269}$$

$$\frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{\Psi}_\mathbf{T}} = \begin{bmatrix} \frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{T}[k]} & \frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{T}[k+1]} \end{bmatrix} \tag{270}$$

$$= \begin{bmatrix} \frac{1}{2}\frac{\partial \mathbf{a}[k]}{\partial \mathbf{T}[k]}\Delta t & \frac{1}{2}\frac{\partial \mathbf{a}[k+1]}{\partial \mathbf{T}[k+1]}\Delta t \end{bmatrix} \tag{271}$$

$$\frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{\Psi}_{\mathbf{a}_R}} = \begin{bmatrix} \frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{a}_R[k]} & \frac{\partial \mathbf{f_v}[k]}{\partial \mathbf{a}_R[k+1]} \end{bmatrix} \tag{272}$$

$$= \begin{bmatrix} \frac{1}{2}\frac{\partial \mathbf{a}[k]}{\partial \mathbf{a}_R[k]}\Delta t & \frac{1}{2}\frac{\partial \mathbf{a}[k+1]}{\partial \mathbf{a}_R[k+1]}\Delta t \end{bmatrix} \tag{273}$$

References

1. Szmuk, M., Acikmese, B., and Berning, W. A., "Successive Convexification for Fuel-Optimal Powered Landing with Aerodynamic Drag and Non-Convex Constraints," *Proceedings of the AIAA SPACE SciTech Forum 2016*, AIAA 2016-0378.

2. Malyuta, D., Reynolds, T., Szmuk, M., Lew, T., Bonalli, R., Pavone, M., and Acikmese, B., "Convex Optimization for Trajectory Generation: A tutorial on Generating Dynamically Feasible Trajectories Reliability and Efficently," *IEEE Control Systems*, vol. 42, no. 5, October 2022

3. Blackmore, L., "Autonomous Precision Landing of Space Rockets," *Bridge*, vol. 4, no. 46, pp. 15-20, Dec. 2016

4. Berning, W. A., Strohl, D. L., and Bieniawski, S., "Lossless Convex Guidance for Lunar Powered Descent," *Proceedings of the AIAA SciTech Forum 2023*, AIAA 2023-2004.

5. Strohl, L., Doll, J., Fritz, M., Berning, A. W., White, S., Bieniawski, S. R., Carson, J. M., Acikmese, B. "Implementation of a Six Degree of Freedom Precision Lunar Landing Algorithm Using Dual Quaternion Representation," *Proceedings of the AIAA SciTech Forum 2022*, AIAA 2022-1831.

6. Malyuta, D., Yu, Y., Elango, P., Acikmese, B., "Advances in Trajectory Optimization for Space Vehicle Control," *Annual Rev. Control*, vol. 52, pp. 282-315, Dec. 2021

7. Toh, K., Todd, M., Tutuncu,R., "SDPT3 — a Matlab software package for semidefinite programming", *Optimization Methods and Software*, Vol. 11, pp. 545–581, 1999.

8. Sturm, J., "Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones", *Optimization Methods and Software*, Vol. 11, Issue 1-4, pp. 625-653, 1999, DOI: 10.1080/10556789908805766

9. MOSEK ApS, "MOSEK Optimization Toolbox for MATLAB", 2022, https://docs.mosek.com/latest/toolbox/index.html

10. Grant, M., Boyd, S.,"CVX: Matlab software for disciplined convex programming, version 2.0 beta", http://cvxr.com/cvx, September 2013.

11. Mathworks. *Optimization Toolbox*, 2016.

12. Acikmese, B., Ploen, S., "Convex Programming Appraoch to Powered Descent Guidance for Mars," *Journal of Guidance, Control, and Dynamics*, vol. 30, No. 5, Sept-Oct 2007

13. Blackmore, L., Acikmese, B., and Scharf, D., "Minimum-Landing-Error Power Descent Guidance for Mas Landing Using Convex Optimization," *Journal of Guidance, Control, and Dynamics*, vol. 33, No. 4, July-Aug 2010

14. Acikmese, B., Carson, J. M., Blackmore, L., "Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem," *IEEE Transactions on Control Systems Technology*, vol. 21, No. 6, Nov. 2013

15. `https://www.spaceworks.aero/software/quickshot/` [accessed 30 January, 2023].

16. Cherry, G., "A General, Explicit, Optimizing Guidance Law for Rocket-Propelled Spaceflight," *Proceedings of the AAS/ION Astrodynamics Guidance and Control Conference*, August 24-26, No. 64-638

17. Sostaric, R. R., and Rea, R. J., "Powered Descent Guidance Methods for the Moon and Mars," *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit 2005*, AIAA 2005-6287.