

6DOF Free-Final-Time Successive Convexification Powered Descent Guidance Algorithm Implementation

Pádraig Lysandrou*

University of Colorado Boulder, Boulder, Colorado, 80301

Final Project for ASEN6519: Aerospace Vehicle Guidance & Control

In this paper, I present a summary and implementation of a successive convexification algorithm written by Szmuk and Acikmese[1] with an accompanying attitude control system utilizing these reference trajectories. The algorithm is an extension of previous work on maximizing final mass, therefore minimizing fuel consumption [2] [3] [4]. However the algorithm presented is free-final-time formulation that is solved iteratively with a 'successive convexification' framework. We ignore aerodynamic forces in this presentation. To provide a tractable solution we must employ a methods of lossless convexification, linearization, trust regions, and relaxations. These trajectory optimization problems are nontrivial due to non-convex control restraints and will be formulated as a finite-dimensional second-order cone program (SOCP). SOCPs are a convex program subclass with efficient solvers that have deterministic convergence properties. These algorithms can be implemented on autonomous real-time systems with powerful modern Interior Point Method (IPM) solvers. Towards the end, I then apply these attitude trajectories and engine inputs to an attitude controller and simulation.

Nomenclature

\mathbf{r}_I	=	Position vector, inertial frame
\mathbf{v}_I	=	Velocity vector, inertial frame
ω_B	=	Angular velocity vector, body frame
$q_{B/I}$	=	Unit attitude quaternion
\mathbf{T}_B	=	Body Frame Thrust vector (N)
\mathbf{g}_I	=	Gravity vector of planet (m/s^2)
$m(t)$	=	Mass of vehicle wrt time
α	=	Constant describing mass consumption rate
T_{min}	=	Lower Thrust Bound
T_{max}	=	Upper Thrust Bound
m_{wet}	=	The total mass of the vehicle including propellant
t_f	=	Time of flight
\mathbf{x}	=	State Vector
\mathbf{u}	=	Input Vector
τ	=	Normalized Trajectory Time
σ	=	Time dilation coefficient
δ	=	Gimbal angle
γ_{gs}	=	Glideslope angle constraint

*PhD Student, Aerospace Engineering, AIAA Student Member

I. Introduction

In this paper, I present implementations of convex programming algorithms for the powered descent guidance problem. I have implemented the algorithm by Szmuk and Acikmese here[1]. I have also used the attitude profiles generated by the algorithm to try a cold gas thruster attitude control system on a cylindrical shaped vehicle mimicking a Vertical-Takeoff-Vertical-Landing (VTVL) rocket.

Having the ability to land near a site of scientific interest, a base, or refueling station is valuable. The convex methods I will present can be used for landing a spacecraft on Mars as well as other planetary bodies like Earth for application in reusable launch vehicles. The ability to soft-land a rocket is fundamentally disruptive to the launch industry and has already had a huge impact in reducing the cost of getting to space. Additionally, similar methods, given navigational upgrades, will be conducive of the development of colonies on other planets. Pin-point landing and divert capability is a necessity in these situations.

Every pinpoint landing problem begins with an entry phase where the vehicle descends through an atmosphere to a point where powered descent must begin. Many Mars entry, descent, and landing schemes enter the atmosphere and slow down via parachute. This parachute is then cut away to allow powered descent. With atmospheric qualities being stochastic, the position in which the descent phase must begin is uncertain. Therefore, we would like to look at algorithms which maximize the divert capability of the craft by minimizing the fuel consumption or final time from initial condition to terminal state. We define powered descent guidance as the generation of a fuel-optimal trajectory that takes the vehicle from some initial state condition to a prescribed final state in a uniform gravitational field with standard vehicle given thrust magnitude and direction constraints.

The convex optimization framework is exploited because it is conducive of real-time on-board implementation and has guaranteed convergence properties with deterministic criteria. The convex programming algorithm to solve powered descent guidance that I will present has non-convex controls constraints and will be posed as a finite-dimensional SOCP problem. SOCPs have low complexity and can be solved in polynomial time [5]. Interior-point numerical methods compute optimal solutions with deterministic stopping criteria and are, again conducive to on-board implementation.

II. Problem Description

The goal is to be able to generate optimal attitude and translational trajectories. I will use these attitude trajectories later on in a control system implementation.

We shall define the 6DOF minimum final time powered descent problem in the original continuous non-convex dynamic form. The final time of the problem is now an optimization variable. Going forward we will use t_f s the final time of the trajectory, and will refer to this as the time-of-flight.

We shall define our two frames. The \mathcal{F}_I frame defines an inertially fixed Up-East-North reference frame with the origin located at the landing site. The \mathcal{F}_B frame is a body fixed frame follows similarly with the X-axis is aligned vertically with the vehicle, are with the thrust vector at zero gimbal angle. The Y-axis points out of the side of the cylindrical vehicle. The Z-axis completes the right handed triad.

A. Dynamics

In our formulation, the vehicle is treated as a rigid body. We assume that aerodynamic forces are negligible but that the body is subject to planetary gravitation \mathbf{g}_I . The algorithm as presented in [1] has the vehicle actuated by a single gimbaled thruster at the bottom of the vehicle. This engine has feasible thrust magnitude and efficiency, as well as standard gimbal range for agile vehicles. Modifications can be made here to include other actuators in the dynamics. I could have adapted my own actuator dynamics here, but to save time I have implemented the original algorithm.

We also capture mass depletion dynamics, proportional to the magnitude of the thrust generated by the engine. For tractability, we assume that the inertia matrix and the position of the center-of-mass is constant throughout the trajectory. We use the constant $\alpha_{\dot{m}}$, a function of the specific impulse, as the mass depletion parameter. Therefore we have that

$$\alpha_{\dot{m}} = \frac{1}{I_{sp} g_0} \quad (1)$$

$$\dot{m}(t) = -\alpha_{\dot{m}} \|\mathbf{T}_B t\|_2 \quad (2)$$

We will express the translational dynamics and forces acting on the vehicle in the inertially fixed frame. They are as follows:

$$\dot{\mathbf{r}}_I(t) = \mathbf{v}_I(t) \quad (3)$$

$$\dot{\mathbf{v}}_I(t) = \frac{\mathbf{F}_I}{m(t)} + \mathbf{g}_I \quad (4)$$

The attitude formalism used in the paper are Euler parameters, or quaternions. They are used to denote the attitude of the vehicle between the \mathcal{F}_B and \mathcal{F}_I frames, $q_{B/I}(t)$ on the unit sphere. We use the angle-axis form of the quaternion, noted here:

$$q_{B/I}(t) \triangleq \begin{bmatrix} \cos(\phi/2) \\ \hat{n} \sin(\phi/2) \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T \quad (5)$$

The $\hat{n} \in \mathcal{R}^3$ vector is the euler vector in which the single angle ϕ displaces the attitude about. In our derivation we will also need the direction cosine matrix produced by this quaternion. DCMs are part of the $SO(3)$ group with properties that its determinant is 1 and can be transposed/inverted for the reverse mapping. They can be multiplied to encode multiple rotations. The mapping from inertial to body frame is the following:

$$C_{B/I} = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix} \quad (6)$$

In our attitude dynamics, we will use $\omega_B(t) \in \mathcal{R}^3$ to denote the angular velocity vector of the vehicle in the \mathcal{F}_B frame with respect to \mathcal{F}_I . The torque acting on the vehicle is written as $\mathbf{M}_B(t) \in \mathcal{R}^3$ in the body frame. We also use the $[\mathbf{r}^\times]$ operator to denote the skew symmetric form of the vector \mathbf{r} . The inertia tensor instantiated in \mathcal{F}_B is written as $J_B \in \mathcal{R}^{3 \times 3}$.

Given that we have chosen quaternions as our formalism, we must understand the kinematics as such:

$$\dot{q}_{B/I}(t) = \frac{1}{2}B(\omega_B(t))q_{B/I}(t) \quad (7)$$

where the matrix $B(\omega_B(t))$ is defined as

$$B(\omega_B(t)) = \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \quad (8)$$

We can also derive the simplified angular dynamics as the following

$$J_B \dot{\omega}_B(t) = -[\omega_B^\times] J_B \omega_B + \mathbf{M}_B(t) \quad (9)$$

With the torque applied to the spacecraft.

B. Constraints and Boundary Conditions

We must now state the constraints and boundary values for the problem required by the optimization. We know that we can never use more fuel than we have, therefore the convex constraint is

$$m(t) \geq m_{dry} \quad (10)$$

Additionally we want to apply a glide slope constraint in the case that our vehicle has terrain relative navigation

sensors. We form the convex constraint using the angle γ_{gs} :

$$\mathbf{e}_1 \cdot \mathbf{r}_I(t) \geq \tan(\gamma_{gs}) \left\| [\mathbf{e}_2 \quad \mathbf{e}_3]^T \mathbf{r}_I(t) \right\|_2 \quad (11)$$

This effectively draws an upward facing cone slope that the vehicle must not penetrate. This type of convex constraint can also be useful in avoiding rocky terrain and enforcing a landing from directly above an area, minimizing lateral movement close to the ground. The author also included that the vehicle must avoid excessive tilt angles such that it stays relatively upright throughout the trajectory, we can constrain a portion of the direction cosine matrix as such:

$$\cos(\theta_{max}) \leq 1 - 2(q_2^2 + q_3^2) \quad (12)$$

Along the same line of thought, we can constrain the angular rate using convex constraints as well:

$$\|\boldsymbol{\omega}_B(t)\|_2 \leq \omega_{max} \quad (13)$$

Finally we must encode that the commanded thrust vector needs to be constrained between two magnitudes, which is appropriate. We assume that the engine cannot be re-lit during operation (commonly becoming a bad assumption to make). We also know that there is some limit to the amount of gimbal angle we can perform δ_{max} .

$$0 < T_{min} \leq \|\mathbf{T}_B(t)\|_2 \leq T_{max} \quad (14)$$

$$\cos(\delta_{max}) \|\mathbf{T}_B(t)\|_2 \leq \mathbf{e}_1 \cdot \mathbf{T}_B(t) \quad (15)$$

We see that the upper thrust bound is clearly convex but the lower bound creates a non-convex constraint. Because of the range of δ_{max} turns out to be convex.

For the powered descent guidance problem, the boundary conditions are trivial to identify. The initial mass, position, velocity, attitude, and angular rates are given by the navigation subsystem at the moment in time. We assume the dry mass is known and that an estimate of the wet mass can be backed out. The terminal position is the landing site. The terminal translational and angular velocities can be zero, with the terminal attitude in the upright position. The final mass should be free. The initial thrust vector should be free, while the terminal vector should be along the vehicle up, X-axis. The author intends on the initial attitude being free, which can free up the optimization for shorter final times.

C. Continuous Time Problem

Putting this all together we can pose the continuous time optimization problem. In this form, it is non-convex and requires significant conditioning to work into the convex framework. As stated, the objective is to minimize the time-of-flight required to get to the terminal conditions subject to the aforementioned constraints, dynamics, and boundary conditions. We leave the commanded thrust, gimballed angles, and final time free. It is as follows:

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

Cost Function:

$$\min_{\mathbf{T}_B, t_f} t_f$$

Boundary Conditions:

$$\begin{aligned} m(0) &= m_{wet} & \mathbf{r}_I(0) &= \mathbf{r}_i & \mathbf{v}_I(0) &= \mathbf{v}_i & q_{B/I}(0) &= q_{B/I_i} & \boldsymbol{\omega}_B(0) &= \boldsymbol{\omega}_{B_i} \\ \mathbf{r}_I(t_f) &= \mathbf{0} & \mathbf{v}_I(t_f) &= \mathbf{0} & q_{B/I}(t_f) &= q_{B/I_f} & \boldsymbol{\omega}_B(t_f) &= \mathbf{0} \\ \mathbf{e}_2 \cdot \mathbf{T}_B(t_f) &= \mathbf{e}_3 \cdot \mathbf{T}_B(t_f) &= 0 \end{aligned}$$

Dynamics:

$$\begin{aligned} \dot{m}(t) &= -\alpha_{\dot{m}} \|\mathbf{T}_B t\|_2 \\ \dot{\mathbf{r}}_I(t) &= \mathbf{v}_I(t) \\ \dot{\mathbf{v}}_I(t) &= \frac{C_{I/B}(t)\mathbf{T}_B(t)}{m(t)} + \mathbf{g}_I \\ \dot{q}_{B/I}(t) &= \frac{1}{2} B(\boldsymbol{\omega}_B(t)) q_{B/I}(t) \\ J_B \dot{\boldsymbol{\omega}}_B(t) &= -[\boldsymbol{\omega}_B^\times] J_B \boldsymbol{\omega}_B + [r_{com}^\times] \mathbf{T}_B(t) \end{aligned}$$

State Constraints:

$$\begin{aligned} m(t) &\geq m_{dry} \\ \mathbf{e}_1 \cdot \mathbf{r}_I(t) &\geq \tan(\gamma_{gs}) \left\| [\mathbf{e}_2 \ \mathbf{e}_3]^T \mathbf{r}_I(t) \right\|_2 \\ \cos(\theta_{max}) &\leq 1 - 2(q_2^2 + q_3^2) \\ \|\boldsymbol{\omega}_B(t)\|_2 &\leq \omega_{max} \end{aligned}$$

Control Constraints:

$$\begin{aligned} 0 < T_{min} &\leq \|\mathbf{T}_B(t)\|_2 \leq T_{max} \\ \cos(\delta_{max}) \|\mathbf{T}_B(t)\|_2 &\leq \mathbf{e}_1 \cdot \mathbf{T}_B(t) \end{aligned}$$

III. Convex Form

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

A. Linearization

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

$$\mathbf{x}(t) \triangleq \begin{bmatrix} m(t) & \mathbf{r}_I^T(t) & \mathbf{v}_I^T(t) & q_{B/I}^T(t) & \boldsymbol{\omega}_B^T(t) \end{bmatrix}^T \quad (16)$$

$$\mathbf{u}(t) \triangleq \mathbf{T}_B(t) \quad (17)$$

Therefore we can express the nonlinear dynamics in the following form

$$\frac{d}{dt}\mathbf{x}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) = \begin{bmatrix} \dot{m}(t) & \dot{\mathbf{r}}_I^T(t) & \dot{\mathbf{v}}_I^T(t) & \dot{q}_{B/I}^T(t) & \dot{\omega}_B^T(t) \end{bmatrix}^T \quad (18)$$

In order to make the problem free-final-time, we must manipulate the how time works. We are going to replace t with a normalized trajectory time $\tau \in [0, 1]$. We apply the chain run:

$$\frac{d}{dt}\mathbf{x}(t) = \frac{d\tau}{dt} \frac{d}{d\tau}\mathbf{x}(\tau) \quad (19)$$

And we can translate between the two by using the dilation coefficient σ which is defined

$$\sigma \triangleq \left(\frac{d\tau}{dt}\right)^{-1} \quad (20)$$

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

$$\mathbf{x}'(\tau) \triangleq \frac{d}{d\tau}\mathbf{x}(\tau) = \sigma f(\mathbf{x}(\tau), \mathbf{u}(\tau)) \quad (21)$$

We can fit the nonlinear dynamics from problem 1 into a linear framework by performing a first-order Taylor approximation about a reference state, input, and dilation coefficient $(\hat{\mathbf{x}}, \hat{\mathbf{u}}, \hat{\sigma})$. Therefore we can represent the system using normalized time:

$$\mathbf{x}'(\tau) = \hat{\sigma} \frac{\partial}{\partial x} f(x, u) \Big|_{\hat{x}, \hat{u}} (x(\tau) - \hat{x}(\tau)) + \hat{\sigma} \frac{\partial}{\partial u} f(x, u) \Big|_{\hat{x}, \hat{u}} (u(\tau) - \hat{u}(\tau)) + \sigma f(\hat{x}(\tau), \hat{u}(\tau)) \quad (22)$$

$$\mathbf{x}'(\tau) = A(\tau)\mathbf{x}(\tau) + B(\tau)\mathbf{u}(\tau) + \Sigma(\tau)\sigma + \mathbf{z}(\tau) \quad (23)$$

We can break the Taylor expansion into matrices to make things simpler later on:

$$A(\tau) \triangleq \hat{\sigma} \frac{\partial}{\partial x} f(x, u) \Big|_{\hat{x}, \hat{u}} \quad (24)$$

$$B(\tau) \triangleq \hat{\sigma} \frac{\partial}{\partial u} f(x, u) \Big|_{\hat{x}, \hat{u}} \quad (25)$$

$$\Sigma(\tau) \triangleq f(\hat{x}(\tau), \hat{u}(\tau)) \quad (26)$$

$$\mathbf{z}(\tau) \triangleq -A(\tau)\hat{\mathbf{x}}(\tau) - B(\tau)\hat{\mathbf{u}}(\tau) \quad (27)$$

With this done, we tackle the only source of non-convexity: the non-zero lower bound to our actuator thrust. With a Taylor series approximation we can say that $T_{min} \leq B_g(\tau)(u(\tau))$ for which $B_g(\tau) \triangleq \frac{\hat{\mathbf{u}}^T(\tau)}{\|\hat{\mathbf{u}}\|_2}$.

B. Discretization Scheme

This is where things get hairy. We need to "cast" the problem into a finite dimensional optimization problem where the trajectory is discretized into K evenly separated points with respect to the normalized trajectory time τ . Let us define the set $\mathcal{K} \triangleq \{0, 1, \dots, K-1\}$ with which many of the parameter arrays will have the same dimensions.

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

$$\tau_k \triangleq \frac{k}{K-1}, \quad \forall k \in \mathcal{K} \quad (28)$$

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

$$\mathbf{u}(\tau) \triangleq \alpha_k(\tau)\mathbf{u}_k + \beta(\tau)\mathbf{u}_{k+1} \quad (29)$$

$$\alpha_k(\tau) = \frac{d\tau - \tau}{d\tau} \quad (30)$$

$$\beta_k(\tau) = \frac{\tau}{d\tau} \quad (31)$$

$$d\tau = \frac{1}{K-1} \quad (32)$$

I have shown these differently than the author for simplicity.

We can then use a state transition matrix $\Phi(\tau_{k+1}, \tau_k)$ to translate the system from k to $k+1$ with no input. This matrix follows these dynamics:

$$\frac{d}{d\tau}\Phi(\tau, \tau_k) = A(\tau)\Phi(\tau, \tau_k) \quad \forall k \in \mathcal{K} \quad (33)$$

Using our previous expressions, we can now discretize the dynamics such that each matrix, at each time step requires a short integration:

$$\mathbf{x}_{k+1} = \bar{A}_k \mathbf{x}_k + \bar{B}_k \mathbf{u}_k + \bar{C}_k \mathbf{u}_{k+1} + \bar{\Sigma}_k \sigma + \bar{\mathbf{z}}_k \quad (34)$$

$$\bar{A}_k \triangleq \Phi(\tau_{k+1}, \tau_k) \quad (35)$$

$$\bar{B}_k \triangleq \int_{\tau_k}^{\tau_{k+1}} \alpha_k(\xi) \Phi(\tau_{k+1}, \xi) B(\xi) d\xi \quad (36)$$

$$\bar{C}_k \triangleq \int_{\tau_k}^{\tau_{k+1}} \beta_k(\xi) \Phi(\tau_{k+1}, \xi) B(\xi) d\xi \quad (37)$$

$$\bar{\Sigma}_k \triangleq \int_{\tau_k}^{\tau_{k+1}} \Phi(\tau_{k+1}, \xi) \Sigma(\xi) d\xi \quad (38)$$

$$\bar{\mathbf{z}}_k \triangleq \int_{\tau_k}^{\tau_{k+1}} \Phi(\tau_{k+1}, \xi) \mathbf{z}(\xi) d\xi \quad (39)$$

$$(40)$$

The other boundary, state, and control constraints are already convex and just need to be instantiated for all time or at initial/terminal τ .

C. Successive Form, Trust Regions and Relaxations

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

$$\delta \mathbf{x}_k^i \triangleq \mathbf{x}_k^i - \mathbf{x}_k^{i-1} \quad (41)$$

$$\delta \mathbf{u}_k^i \triangleq \mathbf{u}_k^i - \mathbf{u}_k^{i-1}, \quad \forall k \in \mathcal{K} \quad (42)$$

$$\delta \sigma^i \triangleq \sigma^i - \sigma^{i-1} \quad (43)$$

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

$$\delta \mathbf{x}_k^i \cdot \delta \mathbf{x}_k^i + \delta \mathbf{u}_k^i \cdot \delta \mathbf{u}_k^i \leq \mathbf{e}_k \cdot \bar{\Delta}^i \quad (44)$$

$$\delta \sigma^i \cdot \delta \sigma^i \leq \Delta_{\sigma}^i \quad (45)$$

We can now append $w_{\Delta}^i \|\bar{\Delta}^i\| + w_{\Delta_{\sigma}} \|\Delta_{\sigma}^i\|$ to the cost function to attempt to minimize input, state, and final time deviations and keeping their deviation bounded via constraint, where w_{Δ}^i and $w_{\Delta_{\sigma}}$ are weighting scalars. Here the author makes an important note. Given that the trust regions are centered about previous points $(\mathbf{x}_k^{i-1}, \mathbf{u}_k^{i-1}, \sigma^{i-1})$, we must evaluate the Jacobian about the nonlinear trajectory beginning at \mathbf{x}_k^{i-1} . We can then use the zero-order-hold input vector $\mathbf{u}(\tau)$. Doing this $\forall k \in \mathcal{K}$ defines the linearization path $(\hat{\mathbf{x}}, \hat{\mathbf{u}}, \hat{\sigma})$.

Finally we must speak to dynamic relaxation. "Artificial Infeasibility" is encountered during the convergence process when the linearization becomes infeasible. For example, if the dynamics are linearized about unrealistic conditions, it becomes dynamically inconsistent. It will not produce a feasible solution. This is encountered during the first couple iterations of a successive convexification due to poor initial trajectory or time-of-flight estimation. To get rid of this issue, we employ dynamic relaxation, where a slack variable is added to the dynamics in order to "make room" for the iteration to proceed. However, you can guess that this will inevitably be something we try to minimize in the cost function in order to make sure that our final trajectories are as dynamically consistent as possible. Therefore, we must now write our dynamics as follows:

$$\mathbf{x}_{k+1}^i = \bar{A}_k^i \mathbf{x}_k^i + \bar{B}_k^i \mathbf{u}_k^i + \bar{C}^i \mathbf{u}_{k+1}^i + \bar{\Sigma}_k^i \sigma^i + \bar{\mathbf{z}}_k^i + \mathbf{v}_k^i \quad (46)$$

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

$$\bar{\mathbf{v}}^i = \begin{bmatrix} \mathbf{v}_0^{iT} & \cdots & \mathbf{v}_{K-2}^{iT} \end{bmatrix}^T \quad (47)$$

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

1. Convex Sub-Problem

We can now put all of the components together and form the convex version of the continuous problem, but with fixed final time.

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

$$\min_{\sigma^i, \mathbf{u}_k^i} \quad \sigma^i + w_{\Delta}^i \left\| \bar{\Delta}^i \right\|_2 + w_{\Delta\sigma} \left\| \Delta_{\sigma}^i \right\|_1 + w_v \left\| \bar{\mathbf{v}}^i \right\|_1$$

Boundary Conditions:

$$\begin{aligned} m_0 &= m_{wet} & \mathbf{r}_{I_0} &= \mathbf{r}_i & \mathbf{v}_{I_0} &= \mathbf{v}_i & q_{B/I_0} &= q_{B/I_i} & \omega_B(0) &= \omega_{B_i} \\ \mathbf{r}_{I_K} &= \mathbf{0} & \mathbf{v}_{I_K} &= \mathbf{0} & q_{B/I_K} &= q_{B/I_f} & \omega_{B_K} &= \mathbf{0} \\ \mathbf{e}_2 \cdot \mathbf{T}_{B_K} &= \mathbf{e}_3 \cdot \mathbf{T}_{B_K} = 0 \end{aligned}$$

Dynamics:

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

State Constraints:

$$\begin{aligned} m_k &\geq m_{dry} \\ \mathbf{e}_1 \cdot \mathbf{r}_{I_k} &\geq \tan(\gamma_{gs}) \left\| [\mathbf{e}_2 \quad \mathbf{e}_3]^T \mathbf{r}_{I_k} \right\|_2 \\ \cos(\theta_{max}) &\leq 1 - 2(q_{2_k}^2 + q_{3_k}^2) \\ \left\| \omega_{B_k} \right\|_2 &\leq \omega_{max} \end{aligned}$$

Control Constraints:

$$\begin{aligned} T_{min} &\leq B_g(\tau_k) \mathbf{u}_k^i \\ \left\| \mathbf{u}_k^i \right\|_2 &\leq T_{max} \\ \cos(\delta_{max}) \left\| \mathbf{u}_k^i \right\|_2 &\leq \mathbf{e}_1 \cdot \mathbf{u}_k \end{aligned}$$

Trust Regions:

$$\begin{aligned} \delta \mathbf{x}_k^i \cdot \delta \mathbf{x}_k^i + \delta \mathbf{u}_k^i \cdot \delta \mathbf{u}_k^i &\leq \mathbf{e}_k \cdot \bar{\Delta}^i \\ \delta \sigma^i \cdot \delta \sigma^i &\leq \Delta_{\sigma}^i \end{aligned}$$

IV. Algorithm

The goal is for the successive convexification algorithm to continue iterating until Δ_{tol} and v_{tol} are met. These are defined by the magnitude of each vector.

Algorithm 1 Successive Convexification

```

1: procedure SCVX( $x_{ref}, \sigma_{ref}$ )
2:   Generate initial trajectory by linearly spanning from initial condition to terminal conditions on each state
3:   while  $\|\Delta\| \geq \Delta_{tol}$  &&  $\|v\| \geq v_{tol}$  do
4:     Compute  $\bar{A}_k^{i-1}, \bar{B}_k^{i-1}, \bar{C}_k^{i-1}, \bar{\Sigma}_k^{i-1}, \bar{z}_k^{i-1}$  from  $\mathbf{x}_k^{i-1}, \mathbf{u}_k^{i-1}, \sigma^{i-1}$ 
5:     Solve Problem 2 using  $\mathbf{x}_k^{i-1}, \mathbf{u}_k^{i-1}, \sigma^{i-1}, \bar{A}_k^{i-1}, \bar{B}_k^{i-1}, \bar{C}_k^{i-1}, \bar{\Sigma}_k^{i-1}, \bar{z}_k^{i-1}$ 
6:     Store the newly found  $\mathbf{x}_k^i, \mathbf{u}_k^i, \sigma^i$ 
7:      $i++$ ;
8:   end while
9:   return  $\mathbf{x}, \mathbf{u}$ 
10: end procedure

```

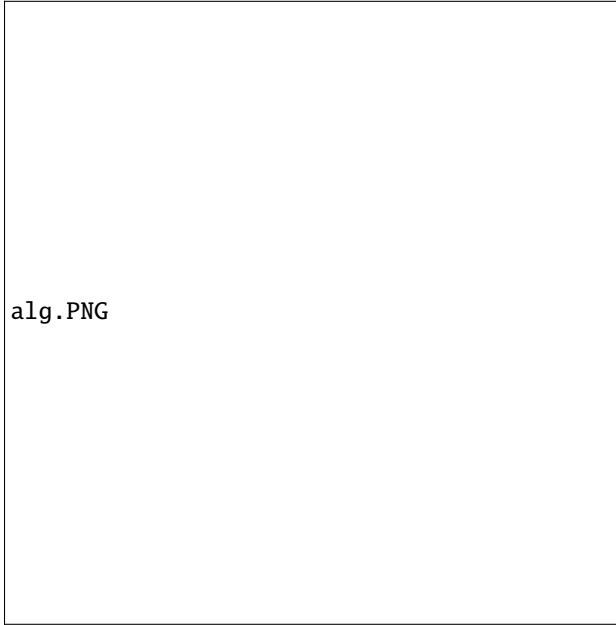
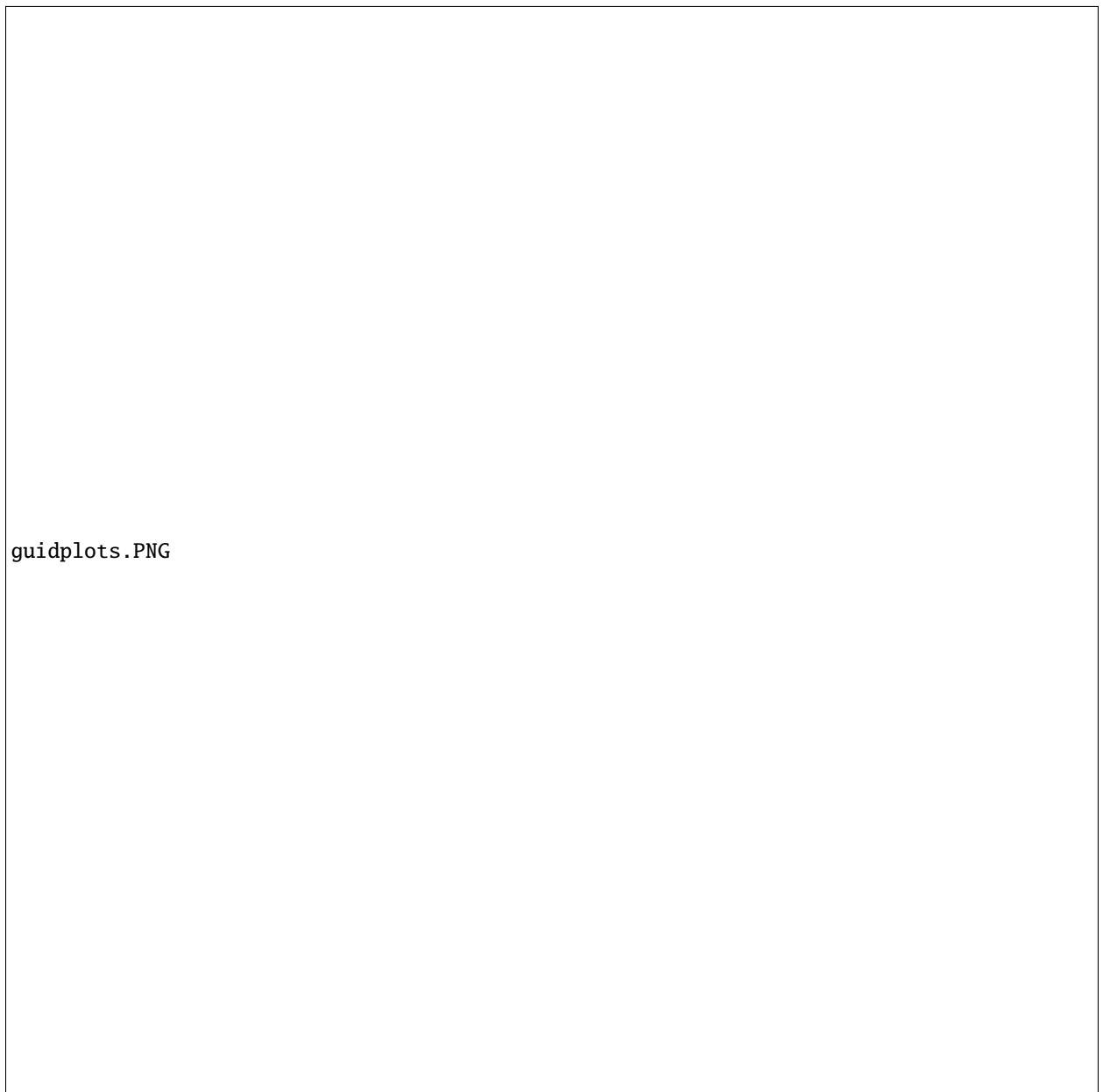


Fig. 1 Diagram depicting how the successive convexification loop works. If the tolerance requirements are not met, another iteration is done again to make sure it is producing dynamically feasible sets.

It turns out that this tool is incredibly powerful. One can modify the dynamics quite a bit, and depending on the robustness of the linearization scheme, this could be used as a generalized tool. The only glaring obstacle is state and actuator constraints. These types of nonconvexities need to be managed carefully.

V. Dimensionalized Simulation Results

The initial conditions to this problem must be small for numerical stability. Therefore, you must non-dimensionalize then on entry. Once the problem is solved, you can redimensionalize the solution, as shown in 1. I implemented this algorithm in Python, utilizing the CVXPY library with ECOS solver. Below in 1 are some parameters I tested for an in-plane maneuver. I used the same constants and weightings as in [1]. For this scenario, to redimensionalize, $U_M = 10\text{kg}$, $U_L = 100\text{meters}$, and $U_T = (10)^{1/2}\text{seconds}$.



guidplots.PNG

Fig. 2 Top Left - Planar trajectory plotted in 3D with quivers acting as thrust vector and magnitude indicators. The trajectory looks smooth and dynamically consistent. Top Right - This plot displays the thrust on each axis, the magnitude, and the thrust angle. These look very jagged and non-differentiable. Numerical problems could cause this. Ideally we would like to actuate the engine at lower gimbal rates. Lower Left - The attitudes generated look smooth and dynamically consistent. Lower Right - The mass depletion is shown. It seems to only have consumed 4kg of propellant over the $\sigma = 6.474$.

Table 1 Parameters Used

Param	Units	Value
\mathbf{g}_I	U_L/U_T^2	$-0.981\mathbf{e}_1$
$\alpha_{\dot{m}}$	-	0.004
m_{wet}	U_M	14
m_{dry}	U_M	3
r_{I_0}	U_L	(4, 4, 0)
v_{I_0}	U_L/U_T	(-0.1, 0, 0)
q_{BI_0}	-	(1, 0, 0, 0)
ω_{B_0}	rad/U_T	(0, 0, 0)

VI. Attitude Trajectory Tracking with Cold Gas Thrusters

This portion of the project was developed for ASEN6010 Advanced Attitude Control. I will now show how control can be done with thrusters at the top of the vehicle. I am using the attitude trajectory generated from the convex optimal problem. I have converted those quaternions into MRPs and switched from the UEN frame to a more common ENU frame. This is to say that +x is in the direction of thruster 2 on figure 4; +y goes along thruster 1, and thruster 6/8 generate a positive torque about the z-axis axially through the top.

Please note that I did not implement standard Schmitt trigger "bang bang" control here. This probably would have been a more realistic implementation.

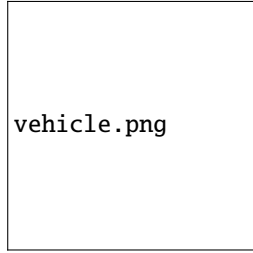


Fig. 3 Example vehicle where circles represent cold gas thruster placement, \mathbf{r}_i represents distance to thruster, and \mathbf{g}_t represents pointing vector.

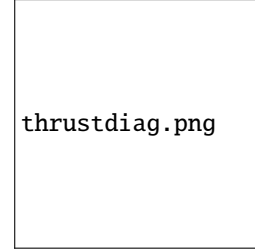


Fig. 4 This represents the placement of thrusters for my implementation. Each arrow is the pointing direction for each engine. For example, thruster 1 would pitch the vehicle over about the x-axis, thruster 2 generates a torque about y-axis, and 6/8 combination rolls about the z-axis

The distribution matrix D for my configuration is the following:

$$[D] = \begin{bmatrix} \vec{r}_1 \times \vec{g}_{t_1} & \cdots & \vec{r}_N \times \vec{g}_{t_N} \end{bmatrix} = \begin{bmatrix} -1.50 & 0 & 1.50 & 0 & 1.50 & -1.50 & -1.50 & 1.50 \\ 0 & 1.50 & 0 & -1.50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.61 & -0.61 & 0.61 & -0.61 \end{bmatrix}$$

We shall no use the MRP version of the Lyapunov control function is the following [6]:

$$\mathbf{L}_r = -K\epsilon - [P] \delta\omega + [I] (\dot{\omega}_r - [\omega]^\times \dot{\omega}_r) + [\omega_r]^\times [I] \omega - \mathbf{L} \quad (48)$$

The value K is a scalar gain on the attitude relative MRP (error) ϵ . The value P is a gain on the angular rate error

$\delta\omega$. All values labeled with ω_r are the reference angular velocities that we get from the successive conovex problem. The value \mathbf{L} is a measure of the exogenous torque. In our case, this is zero.

The torque imparted on the spacecraft from an active thruster is the following:

$$\tau_i = \vec{r}_i \times F_i \vec{g}_{t_i} \quad (49)$$

$$\tau_j = \sum_{i=1}^N (\vec{r}_i \times \vec{g}_{t_i}) \cdot \hat{c}_j F_i \quad (50)$$

Where \hat{c}_j is the body vector we expect to torque about. We can also write this expression as such:

$$\tau_j = [d_1 \cdots d_N] \begin{bmatrix} F_1 \\ \vdots \\ F_N \end{bmatrix} = [D]_j \mathbf{F} \quad (51)$$

We see that $[D] \in \mathbb{R}^{1 \times N}$ for our N thrusters. This matrix maps the thruster forces to the spacecraft torque τ_j along our control axis of interest \hat{c}_j . Now we want to find the thrusters that provide a positive force for the desired torque on the system \mathbf{L}_r . We can perform the minimum norm inverse to find the force matrix as such:

$$\mathbf{F}_j = [D]_j^T ([D]_j [D]_j^T)^{-1} \mathbf{L}_r \cdot \hat{c}_j \quad (52)$$

We then pick which thrusters have positive force values, log their identity, and create a reduced matrix $\bar{\mathbf{F}}_j \in \mathbb{R}^{M \times 1}$. We can do the same with $[\bar{D}]$. And therefore $[\bar{D}] \bar{\mathbf{F}}_j$ is the torque on a single axis. Doing this for all control axes, we apply the final torque

$$\mathbf{L}_{cg} = [\bar{D}]_1 \bar{\mathbf{F}}_1 + [\bar{D}]_2 \bar{\mathbf{F}}_2 + [\bar{D}]_3 \bar{\mathbf{F}}_3 \quad (53)$$

A. Simulation Framework

I created a simulation where I numericall integrated the equations of motion for the vehicle. On each cycle I calculated the required torque and fed it back to the system dynamics.

B. Starting At Nominal Attitude

Let's look at the system response if our initial attitude is the same as the one performed in the convex problem. This would be the case if you performed control immediately after guidance finished, or if it converged quickly. See figure 5.

C. Starting At Off-Nominal Attitude and Angular Rate

In this scenario, I have drifter farther from the convex problem initial condition to roughly 32 degrees off axis and with an angular rate of $\omega_0 = [.3.5.5]^T$ rad/s. This is a large error should be hard to converge to in final time. See figure 6. We can also see the output force of each of the 8 thrusters with figure 7. Notice that the magnitudes don't go over 40N, as that is a hard limit that I set.

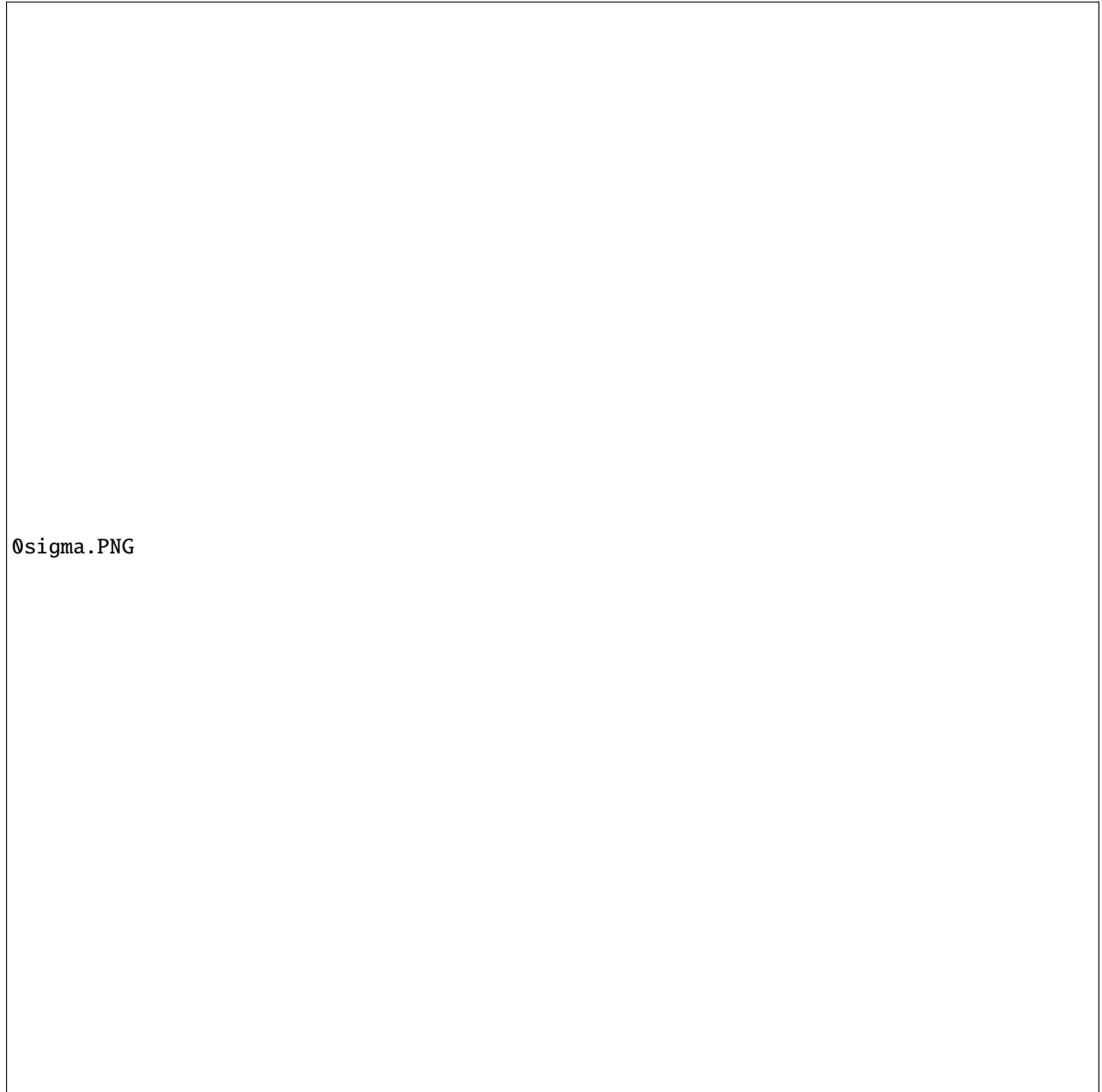


Fig. 5 Top Left - The state MRP and reference MRP are on top of each other, tracking well. Bottom Left - Similar with the state angular rate and the reference. Top Right - The MRP error is in E-4, and very small. Bottom Right - angular error is roughly 0.001 rad/s



Fig. 6 Top Left - Over time the vehicle seemed to track well. There is a very small amount of steady state error on each axis. Bottom Left - The angular rate also converges smoothly over time, with little steady state error. Top Right - The MRP error approaches zero over time, does not quite hit. Bottom Right - Angular rate error goes to zero quite quickly.

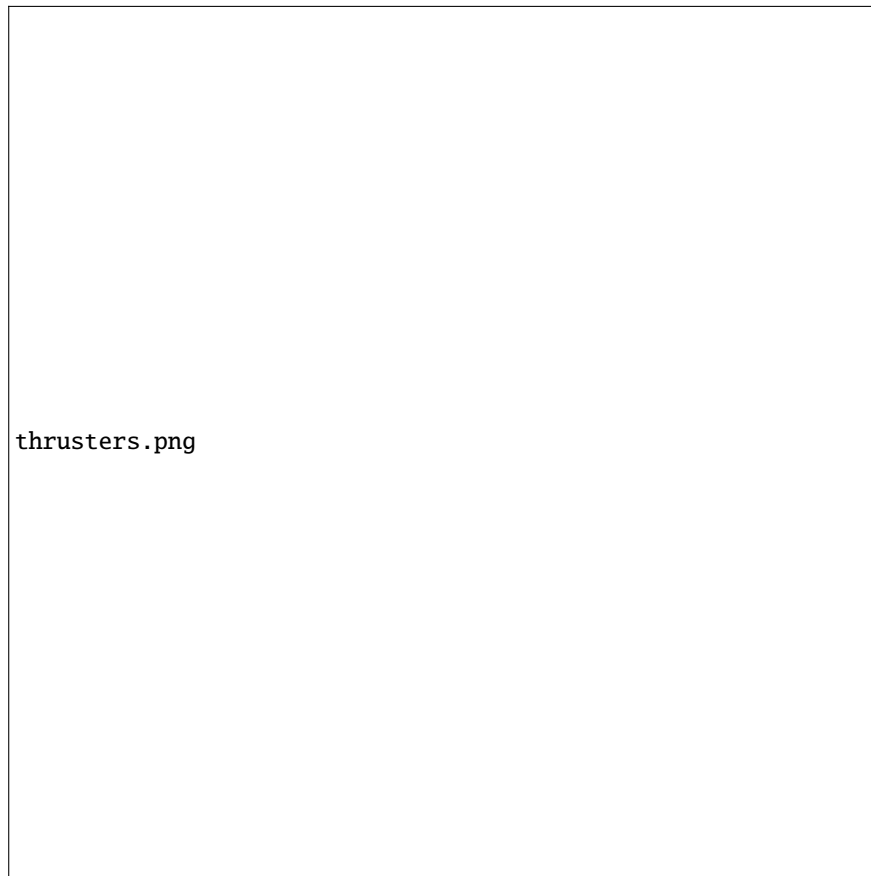


Fig. 7 Plot of the output thrust of each of the thrusters for the non-nominal initial attitude case.

VII. Conclusions and Future Work

In the future I would like to combine translational and attitude controllers in a single simulation. Aerodynamic effects would be interesting to implement and model. Additionally, I think it would be interesting to try new actuators and integrate them into the same code. Aerodynamic control surfaces, while very difficult to simulate, have high attitude control authority through out all phases of flight.

VIII. Acknowledgments

I would like to thank Sven Niederberger. His discretization scheme implementation and code was very helpful with debugging my implementation.

References

- [1] Szmuk, M., and Acikmese, B., “Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time,” *AIAA SciTech Guidance, Navigation, and Control Conference*, 2018. doi:10.2514/6.2018-0617.
- [2] Acikmese, B., and Ploen, S., “Convex Programming Approach to Powered Descent Guidance for Mars Landing,” *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353, 1366. doi:10.2514/1.27553.
- [3] Acikmese, B., Carson, J., and Blackmore, L., “Successive Convexification for Fuel-Optimal Powered Landing with Aerodynamic Drag and Non-Convex Constraints,” *AIAA SciTech Forum*, 2016.
- [4] Acikmese, B., Blackmore, L., Scharf, D., and Wolf, A., “Enhancements on the Convex Programming Based Powered Descent Guidance Algorithm for Mars Landing,” *AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Guidance, Navigation, and Control*, 2008. doi:10.2514/6.2008-6426.
- [5] Stephen Boyd, L. V., *Convex Optimization*, 2009.
- [6] Hanspeter Schaub, J. J., *Analytical Mechanics of Space Systems, Fourth edition*, 2018.