A Model Predictive Control Implementation of Guidance for Fuel Optimal Large Diverts (G-FOLD)

Govind Chari*
Cornell University, Ithaca, NY, 14853

In this paper I will present my recreation of JPL's G-FOLD algorithm for six-degree-of-freedom (6DOF) fuel-optimal powered descent and its implementation as a model predictive controller. At the end of the paper, I will present trajectories generated by the controller as well as modifications I made to the algorithm to make it more robust to disturbances. This paper is written as the final report for ECE5555 Stochastic Systems: Estimation and Control.

I. Nomenclature

$\boldsymbol{r}$	=	position vector (m)
$\boldsymbol{q}$	=	orientation quaternion
$\boldsymbol{v}$	=	velocity vector (m/s)
$\boldsymbol{\omega}$	=	angular velocity vector (rad/s)
$\boldsymbol{e}_i$	=	i^{th} column of the identity matrix
g_0	=	gravitational acceleration in earth at sea level (m/s^2)
$\boldsymbol{g}$	=	gravitational acceleration vector (m/s^2)
m_{dry}	=	dry mass of vehicle (kg)
m_{wet}	=	initial mass of vehicle (kg)
R	=	vehicle radius (m)
l	=	vehicle length (m)
z	=	natural log of mass
$\mathbb{I}$	=	inertia tensor $(kg \cdot m^2)$
I_{sp}	=	specific impulse (s)
ρ_1	=	lower throttle bound (N)
ρ_2	=	upper throttle bound (N)
α	=	mass depletion parameter
γ	=	glideslope $(rads)$
θ_{max}	=	pointing constraint
$\boldsymbol{T}$	=	thrust vector
$\boldsymbol{T}_{des}$	=	desired thrust vector
$\boldsymbol{u}$	=	acceleration induced by thrusters
$\boldsymbol{M}$	=	moment vector
$\boldsymbol{M}_{des}$	=	desired moment vector
Γ	=	slack variable
σ	=	mass normalized slack variable
Δt	=	discretization timestep
t_f	=	time of flight (s)
t_f^*	=	optimal time of flight (s)
$\|\boldsymbol{x}\|$	=	ℓ_2 norm of $\boldsymbol{x}$
$\odot$	=	quaternion product

*Undergraduate Student, Mechanical and Aerospace Engineering

II. Introduction

The powered descent guidance problem is concerned with finding the optimal sequence of thrust vectors that allows a vehicle to decelerate itself from some initial position and velocity to perform a soft landing at some specified target on the ground while consuming as little fuel as possible. By minimizing fuel usage, the divert capabilities of the vehicle are maximized. This problem is of great importance to companies like SpaceX, who land their boosters propulsively, and for JPL, who need to land their rovers near important scientific targets on Mars. For both SpaceX and JPL, it would be ideal to solve this problem online due to uncertainty in initial conditions.

However, the general form of the powered descent problem is nonconvex, so it is not feasible to solve online during vehicle descent, since nonconvex problems can have up to an exponential runtime and local minima are not guaranteed to be global minima. This leads to difficulty in solution convergence. On the other hand, some convex problems, such as second order cone problems (SOCPs), can be solved in polynomial time using interior point methods.

The main source of nonconvexity is the lower throttle bound on the propulsion system. Bipropellant chemical engines such as the ones used on SpaceX's Falcon 9 cannot continuously be throttled from 0% to 100% thrust. There exists a lower bound on the thrust achievable by the engine. Attempting to throttle below this limit can cause improper fuel mixing leading to combustion instabilities which can severely damage the engines. However, it can be shown that the 3DOF problem (where the vehicle is modeled as a point mass) can be reformulated as a finite dimensional second order cone problem (SOCP) by introducing a slack variable to lift the dimensionality of the thrust vector [1]. This technique is referred to as "lossless" convexification, meaning that the problem relaxation does not remove any part of the feasible set and a solution to the relaxed convex problem constitutes a solution to initial nonconvex problem. A proof for this is given in [1].

The solution to this SOCP is a solution to the fixed terminal time problem, meaning that the time-of-flight is specified *a priori* to the solution of the SOCP, and the solution to this problem generates the optimal thrust sequence which results in minimal fuel consumption for that specified time-of-flight. However, solving this SOCP does not allow one to determine whether the specified time of flight is fuel optimal across all times-of-flight. For example, a time-of-flight of 20 seconds may result in 30kg of fuel being consumed, but a time-of-flight of 18 seconds may result in 20kg of fuel being consumed. To find the optimal time-of-flight, the SOCP must be solved many times by varying the time-of-flight until a minimum fuel cost is found. To turn the fixed terminal time problem into a free final time problem, the fixed time SOCP should be wrapped in a time-of-flight search loop.

This scheme can then be implemented as an online model predictive controller where at the beginning of the timestep, the free-final time problem is solved using the current state, the first thrust vector is implemented, and the dynamics is stepped forward in time. At the beginning of the next timestep, the free-final time problem is solved again given the current state and yet again only the first thrust vector is implemented. This process is then repeated until the vehicle lands.

To get this algorithm to work on actual vehicles rather than idealized point masses, attitude control need to be considered. The "proper" way of solving the powered descent problem for a rigid body would be to explicitly consider attitude dynamics in the formulation of the optimization problem. However, attitude dynamics occurs on much shorter timescales than translational dynamics, so we can get away with solving the SOCP to get a desired thrust vector, run an attitude controller to get the desired moment, then blend the two to get the throttle (thrust magnitude) and gimbal angle. A sketch of this process is given in figure (1).

III. Dynamics

The dynamics can be split into three main parts: translational dynamics, attitude dynamics, and mass depletion dynamics. For translational dynamics, we will assume that the gravitational field is uniform, which is a good assumption if the vehicle is close the the surface of the planet. We will also assume that there is no air resistance. Under these assumptions, the translational equation of motion is

$$\ddot{\boldsymbol{r}} = \boldsymbol{g} + \frac{\boldsymbol{T}}{m} \tag{1}$$

For attitude dynamics, we will consider applied moments in the body frame, which is centered at the vehicle's center of mass and whose axes are the principle axes of the vehicle. Considering moments in the body frame makes it easier to turn throttle and gimbal angle into a net moment. We will neglect aerodynamic moments. Since we are also using quaternions as the attitude convension, we must also present the quaternion kinematics equation. Both the attitude dynamics equation of motion and the quaternion kinematic equation are given below.

$$\mathbb{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbb{I}\boldsymbol{\omega}) = \boldsymbol{M} \tag{2}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{3}$$

As the engines fire and fuel is burnt, the mass of the vehicle decreases. We can define a mass depletion parameter and then the mass depletion dynamics.

$$\alpha = \frac{1}{I_{sp} g_0} \tag{4}$$

$$\dot{m} = -\alpha \|\boldsymbol{T}\| \tag{5}$$

The above equation assumes that the engine's specific impulse is independent of throttle. For real engines this is not true, since I_{sp} is a function of combustion chamber pressure, and as an engine is throttled, the chamber pressure fluctuates. However, we will use this simplifying assumption.

IV. Translational Controller

In this section, we will assume the vehicle is a point mass. Firstly, we will look at the general non-convex problem, convixify it, then discretize it so it can be solved using interior point methods.

A. The General Non-Convex Problem

In the general minimum fuel powered descent problem, we are looking for the sequence of thrust vectors that minimize the amount of fuel burned while guiding the vehicle from some initial position and velocity to the ground at a specified landing site. This problem has state and input constraints. The main state constraints are the dynamics given by equations (1) and (5) and a glideslope constraint that keeps the vehicle within an upwards opening cone with apex at the landing site. This glideslope constraint is to prevent translation near the ground which could result in the vehicle hitting a rock or other objects near the surface. The input constraints are the lower and upper throttle bounds. We can also implicitly define a pointing constraint where we constrain all thrust vectors to being within some angle of vertical. Formally, this problem can be written as

Problem 1

$$\min_{t_f, \boldsymbol{T}(t)} \quad \int_0^{t_f} \alpha \|\boldsymbol{T}(t)\| \, dt$$

$$\text{subject to} \quad \ddot{\boldsymbol{r}}(t) = \boldsymbol{g} + \boldsymbol{T}(t)/m(t) \quad \dot{m}(t) = -\alpha \|\boldsymbol{T}(t)\|$$

$$\boldsymbol{T}(t) \cdot \boldsymbol{e}_3 \leq \|\boldsymbol{T}(t)\| cos(\theta_{max}) \quad \forall t \in [0, t_f]$$

$$0 < \rho_1 \leq \|\boldsymbol{T}(t)\| \leq \rho_2 \quad \forall t \in [0, t_f]$$

$$m(0) = m_{wet} \quad m(t_f) \geq m_{dry}$$

$$\boldsymbol{r}(0) = \boldsymbol{r}_0 \quad \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0 \quad \boldsymbol{r}(t_f) = \boldsymbol{0} \quad \dot{\boldsymbol{r}}(t_f) = \boldsymbol{0}$$

$$\|S\boldsymbol{r}\| + c^\top \boldsymbol{r} \leq 0 \quad \forall t \in [0, t_f]$$

where

$$S \coloneqq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{6}$$

$$c^\top \coloneqq \begin{bmatrix} 0 & 0 & -tan\left(\frac{\pi}{2} - \gamma\right) \end{bmatrix} \tag{7}$$

The lower throttle bound results in the nonconvexity of the input set.

B. Lossless Convexifying Throttle Bounds

The lower throttle bound can be convexified by introducting the slack variable Γ. The new problem becomes

Problem 2

$$\min_{t_f, \boldsymbol{T}(t), \Gamma(t)} \quad \int_0^{t_f} \Gamma(t)\, dt$$

$$\text{subject to} \quad \ddot{\boldsymbol{r}}(t) = \boldsymbol{g} + \boldsymbol{T}(t)/m(t) \quad \dot{m}(t) = -\alpha\Gamma(t)$$

$$\boldsymbol{T}(t) \cdot \boldsymbol{e}_3 \leq \Gamma(t) cos(\theta_{max}) \quad \forall t \in [0, t_f]$$
$$\|\boldsymbol{T}(t)\| \leq \Gamma(t) \quad 0 < \rho_1 \leq \Gamma \leq \rho_2 \quad \forall t \in [0, t_f]$$
$$m(0) = m_{wet} \quad m(t_f) \geq m_{dry}$$
$$\boldsymbol{r}(0) = \boldsymbol{r}_0 \quad \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0 \quad \boldsymbol{r}(t_f) = \boldsymbol{0} \quad \dot{\boldsymbol{r}}(t_f) = \boldsymbol{0}$$
$$\|S\boldsymbol{r}\| + c^\top \boldsymbol{r} \leq 0 \quad \forall t \in [0, t_f]$$

Looking at figure (2), one can geometrically see why introducing Γ convexifies the thrust bounds. The initial thrust bounds is an annulus and thus is nonconvex, but by introducing Γ and the constraint $\|\boldsymbol{T}(t)\| \leq \Gamma(t)$, the annulus is converted into a frustum which is convex.

Introducing the slack variable Γ expands the feasible set, so clearly the optimal solution to Problem 1 is in the feasible set for Problem 2. In addition, the solution to Problem 2 is the solution to Problem 1 [1].

C. Change of Variables

In order to more easily formulate Problem 2 as a SOCP, we must perform the following change of variables:

$$\sigma = \frac{\Gamma}{m} \quad \text{and} \quad \boldsymbol{u} = \frac{\boldsymbol{T}}{m} \tag{8}$$

The translational and mass depletion dynamics in Problem 2 can be rewritten as

$$\ddot{\boldsymbol{r}}(t) = \boldsymbol{g} + \boldsymbol{u}(t) \tag{9}$$

$$\frac{\dot{m}(t)}{m(t)} = -\alpha\sigma(t) \tag{10}$$

Solving equation (10) yields

$$m(t) = m_0 \exp\left[-\alpha \int_0^{t_f} \sigma(\tau)\, d\tau\right] \tag{11}$$

Minimizing the fuel consumed is equivalent to maximizing the final vehicle mass. Since $\alpha > 0$, minimizing fuel is equivalent to minimizing

$$\int_0^{t_f} \sigma(t)\, dt \tag{12}$$

The control input constraints now become

$$\|\boldsymbol{u}(t)\| \leq \sigma(t) \tag{13}$$

$$\frac{\rho_1}{m(t)} \leq \sigma(t) \leq \frac{\rho_2}{m(t)} \tag{14}$$

The constraint given by equation (14) is nonconvex. To geometrically see this, look at the doubly shaded region in figure (3).

To convexify this constraint, we will introduce

$$z \coloneqq \ln m \tag{15}$$

Now we can rewrite equations (10) and (14) as

$$\dot{z}(t) = -\alpha\sigma \tag{16}$$

$$\rho_1 \exp[-z(t)] \leq \sigma(t) \leq \rho_2 \exp[-z(t)] \tag{17}$$

The constraint given by equation (17) is still nonconvex. To geometrically see this, look at the doubly shaded region in figure (4).

To convexify equation (17), we can take the Taylor expansion of both sides of the inequality and keep the quadratic term on the lower bound and linear term on the upper bound. This result in

$$\mu_1(t)\left[1 - (z(t) - z_0(t)) + \frac{(z(t) - z_0(t))^2}{2}\right] \leq \sigma(t) \leq \mu_2(t)[1 - (z(t) - z_0(t))] \tag{18}$$

where

$$z_0(t) = \ln(m_{wet} - \alpha\rho_2 t) \tag{19}$$

$$\mu_1(t) = \rho_1 e^{-z_0(t)} \tag{20}$$

$$\mu_2(t) = \rho_2 e^{-z_0(t)} \tag{21}$$

To geometrically see the convexity of this constraint, refer to the doubly shaded region in figure (5). It can also be seen that $z_0(t)$ is a lower bound on $z(t)$ at time t. We can also construct an upper bound on $z_0(t)$. This gives us another constraint,

$$\ln(m_{wet} - \alpha\rho_2 t) \leq z(t) \leq \ln(m_{wet} - \alpha\rho_1 t) \tag{22}$$

Problem 2 can now be approximated as the following continuous time SOCP as,

Problem 3

$$\min_{t_f, \boldsymbol{u}(t), \sigma(t)} \quad \int_0^{t_f} \sigma(t)\, dt$$

$$\text{subject to} \quad \ddot{\boldsymbol{r}}(t) = \boldsymbol{g} + \boldsymbol{u}(t) \quad \dot{z}(t) = -\alpha\sigma(t)$$

$$\boldsymbol{u}(t) \cdot \boldsymbol{e}_3 \leq \sigma(t) cos(\theta_{max}) \quad \forall t \in [0, t_f]$$

$$\|\boldsymbol{u}(t)\| \leq \sigma(t) \quad \forall t \in [0, t_f]$$

$$\mu_1(t)\left[1 - (z(t) - z_0(t)) + \frac{(z(t)-z_0(t))^2}{2}\right] \leq \sigma(t) \leq \mu_2(t)[1 - (z(t) - z_0(t))] \quad \forall t \in [0, t_f]$$

$$\ln(m_{wet} - \alpha\rho_2 t) \leq z(t) \leq \ln(m_{wet} - \alpha\rho_1 t) \quad \forall t \in [0, t_f]$$

$$z(0) = \ln(m_{wet}) \quad z(t_f) \geq \ln(m_{dry})$$

$$\boldsymbol{r}(0) = \boldsymbol{r}_0 \quad \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0 \quad \boldsymbol{r}(t_f) = \boldsymbol{0} \quad \dot{\boldsymbol{r}}(t_f) = \boldsymbol{0}$$

$$\|S\boldsymbol{r}\| + c^\top \boldsymbol{r} \leq 0 \quad \forall t \in [0, t_f]$$

D. Discretization

Problem 3 is an continuous time SOCP. However, to solve this problem using interior point methods, it needs to be converted into a finite size SOCP with a fixed time-of-flight. To do this, we must discretize the problem. The discretization scheme I implemented is a zero order hold.

$$i = \frac{t}{\Delta t} \quad i = 1, \ldots, T \tag{23}$$

$$\boldsymbol{u}(t) = \boldsymbol{u}_i \quad \forall t \in [t_i, t_{i+1}) \tag{24}$$

$$z(t) = z_i \quad \forall t \in [t_i, t_{i+1}) \tag{25}$$

With a zero order hold, the dynamics simply become the constant acceleration kinematic equations and the objective becomes negative of the natural log of the terminal mass, which corresponds to maximizing the terminal vehicle mass.

Problem 4

$$\min_{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{T-1},\sigma_0,\ldots,\sigma_{T-1}} \quad -z_T$$

$$\text{subject to} \quad \boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \dot{\boldsymbol{r}}_i\Delta t + \tfrac{1}{2}(\boldsymbol{u}_i - \boldsymbol{g})\Delta t^2$$
$$\dot{\boldsymbol{r}}_{i+1} = \dot{\boldsymbol{r}}_i + (\boldsymbol{u}_i - \boldsymbol{g})\Delta t$$

$$z_{i+1} = z_i - \alpha\sigma_i\Delta t$$

$$\boldsymbol{u}_i \cdot \boldsymbol{e}_3 \leq \sigma_i cos(\theta_{max})$$

$$\|\boldsymbol{u}_i\| \leq \sigma_i \quad \forall i \in [0, T]$$

$$\mu_{1,i}\left[1 - (z_i - z_{0,i}) + \frac{(z_i-z_{0,i})^2}{2}\right] \leq \sigma_i \leq \mu_{2,i}[1 - (z_i - z_{0,i})] \quad \forall i \in [0, T]$$

$$\ln(m_{wet} - \alpha\rho_2 i\Delta t) \leq z_i \leq \ln(m_{wet} - \alpha\rho_1 i\Delta t) \quad \forall i \in [0, T]$$

$$z(0) = \ln(m_{wet}) \quad z(T) \geq \ln(m_{dry}) \quad \forall i \in [0, T]$$

$$\boldsymbol{r}(0) = \boldsymbol{r}_0 \quad \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0 \quad \boldsymbol{r}(T) = \boldsymbol{0} \quad \dot{\boldsymbol{r}}(T) = \boldsymbol{0}$$

$$\|S\boldsymbol{r}_i\| + c^\top \boldsymbol{r}_i \leq 0 \quad \forall i \in [0, T]$$

When problem 4 is solved given some initial conditions, a trajectory is generated along with the sequence of control inputs that produce the trajectory. To implement this as a model predictive controller, we will take the first thrust vector and call that our desired thrust vector.

E. Time-of-Flight Search

Problem 4 is a fixed time-of-flight problem, meaning that the time horizon for the optimization is specified *a priori* to the solution of the problem. While a solution to Problem 4 is optimal for that specified time-of-flight t_f, it may not be optimal across all time-of-flights.

The mass of fuel consumed is a function of t_f, and we are looking for the time of flight, t_f^* that minimizes the fuel consumed. In my simulation, I noticed that t_f^* tended to be the smallest t_f that resulted in a feasible solution to Problem 4. In my implementation, I did not explicity optimize over all t_f, but this heuristic worked very well. To properly conduct a time-of-flight search, one can utilize techniques from derivative free optimization or estimate the gradient of this function via finite differences then use gradient descent.

The time-of-flight search was the most computationally intensive part of the simulation, since Problem 4 must be solved repeatedly until t_f^* is found. To greatly speed up this process, I initialized the time-of-flight search with a guess. This guess was a few seconds fewer than t_f^* from the previous time-of-flight search. Computing t_f^* for the first stage took a while, but for all following stages, t_f^* was computed quickly due to the initialization.

After the time-of-flight search is concluded we take the first thrust vector that results from solving Problem 4 using t_f^* and call that our desired thrust vector.

V. Attitude Controller

We will now focus on designing our attitude controller. The key assumption that we made which allowed us to safely ignore attitude dynamics in our formulation of the optimization problem is that attitude control occurs on a much shorter timescale than translational control. Thus, when we design our attitude controller, we must ensure that it is able to track references quickly with very little lag. If nonidealizations such as an offcentered engine gimbal were to be considered, an integrator would be needed to remove the resulting steady state error, but for simplicity the attitude controller implemented will be PD.

The attitude controller will attempt to point the vehicle along the desired thrust vector, which is generated by the translational controller. Feedback will be performed on the vector term of the orientation quaternion. If we simplify our system from three degrees of rotational freedom to a single degree of freedom, we can define transfer functions from the error quaternion to controller torque and from controller torque to vehicle pointing angle, where I is the moment of inertia of the vehicle. It is important to note that while a desired moment about the long axis of the vehicle is computed, it can never be satisfied, since the engine on the vehicle has no moment arm to create a rolling moment.

$$\frac{T(s)}{e(s)} = K_p + sK_d \tag{26}$$

$$\frac{\theta(s)}{T(s)} = \frac{1}{Is^2} \tag{27}$$

From iteration in my simulation and using SISOTool in Matlab, I found that the following gains led to quick enough rise-time and good disturbance rejection.

$$K_p = I$$

$$K_d = 0.6I$$

At each timestep, based on the desired thrust and the attitude controller, we can generate a desired moment.

VI. Thrust Allocator

The vehicle's engine is able to impart both a force and a moment via thrust vectoring. There is a gimbal system that is able to point the nozzle of the engine. For our vehicle, we have two inputs: the throttle of the engine (the magnitude of the applied force) and the gimbal angle of the engine (a unit vector that points along the direction of thrust). Using these two inputs it is impossible to simultaneously satisfy the desired thrust vector and desired moment. For example, consider a perfectly upright rocket with a desired thrust vector pointing up and to the right. This will result in a desired clockwise moment as computed by our attitude controller. In order to impart this clockwise moment, we need a thrust

vector pointing up and to the left, which is characteristic of a non-minimum phase system. Thus, we need an intelligent way of setting the throttle and gimbal angle.

The solution to this lies in our assumption that attitude dynamics occurs on a much shorter timescale than translational dynamics. We will firstly set the throttle to the magnitude of the desired thrust vector, then set the gimbal angle to the angle necessary to impart the desired moment on the vehicle. Since this attitude adjustement would occur relatively quickly, the vehicle will soon be aligned along the desired thrust vector and be able to satisfy it. This scheme is described in [2].

We are using a scheme that completely satisfies the desired moment, but we are introducing a bit of lag between the desired thrust vector and imparted thrust vector. This lag results in loss of optimality, but if the attitude controller is quick enough, the loss will be small.

VII. Simulation

From the start of this project I knew that conducting the time-of-flight search and reoptimizing trajectory would be very computationally expensive, so I wrote the entire simulation in C++ as opposed to Matlab for quicker runtime (although I still used Matlab for plotting).

The state variable for this simulation is in $\mathbb{R}^{14}$ and contains the position, velocity, orientation quaternion, angular velocity, and mass. The simulation uses a fixed stepsize Runge Kutta 4th order integrator to step the state forward according to the dynamics, which are encoded in equations (1), (2), (3), (5). The pseudocode for the simulation is given below.

Algorithm 1: 6DOF Simulation

while *Altitude* ≥ 0 **do**
- $\boldsymbol{T}_{des}$ = *PositionController(*$\boldsymbol{r}$, $\boldsymbol{v}$, m*);*
- $\boldsymbol{M}_{des}$ = *AttitudeController(*$\boldsymbol{T}_{des}$, $\boldsymbol{q}$, $\boldsymbol{\omega}$*);*
- *throttle, gimbal angle = ThrustAllocator(*$\boldsymbol{T}_{des}$, $\boldsymbol{M}_{des}$*);*
- $\boldsymbol{F}_{net}$, $\boldsymbol{M}_{net}$ = *GenerateForcesMoments(throttle, gimbal angle,* m, $\boldsymbol{q}$*);*
- $\boldsymbol{F}_{net}$, $\boldsymbol{M}_{net}$ = *ProcessNoise(*$\boldsymbol{F}_{net}$, $\boldsymbol{M}_{net}$*);*
- *state = step(state,* $\boldsymbol{F}_{net}$, $\boldsymbol{M}_{net}$*);*

end

Initially, I wanted to use CVXGEN to autocode a custom solver for Problem 4, but CVXGEN only supports linear and quadratic programs, so I turned to alternatives. I ended up using Embedded Conic Solver (ECOS) [3] with Epigraph [4] as an interface for problem formulation. Epigraph made it easy to specify each of the constraints separately rather than performing the matrix packing by hand and feeding ECOS a problem of the form:

$$\begin{aligned} \min \quad & c^\top x \\ s.t. \quad & Ax = b \\ & \mathrm{G}x \leq_K h \end{aligned} \tag{28}$$

In this simulation, the changing center of mass of the vehicle, the changing inertia tensor, gimbal lag, thrust lag, and sensor noise are not implemented. If I had more time I would add these to the simulation as well as some type of state estimator.

VIII. Results

For all the simulations I conducted, I used the following mass properties and engine properties. The mass properties correspond to a vehicle about the size of SpaceX's Falcon 9 first stage. I also assumed that this powered descent was taking place on earth, so $g = 9.807\ m/s^2$

$m_{dry} = 25600\ kg$
$m_{fuel} = 10000\ kg$
$r = 1.86\ m$
$l = 41.2\ m$
$\rho_1 = 164\ kN$
$\rho_2 = 411\ kN$

$I_{sp} = 311\ s$

To construct the inertia tensor, I modeled the vehicle as a uniformly dense cylinder, whose inertia tensor is given by:

$$\mathbb{I} = \begin{bmatrix} \frac{1}{4}mR^2 + \frac{1}{12}ml^2 & 0 & 0 \\ 0 & \frac{1}{4}mR^2 + \frac{1}{12}ml^2 & 0 \\ 0 & 0 & \frac{1}{2}mR^2 \end{bmatrix} \tag{29}$$

A. 6DOF MPC Trajectory without Disturbances

Figures (6), (7), (8), (9) show two trajectories: one is the 6DOF MPC trajectory, where both translational and attitude control are active, and Problem 4 is solved repeatedly at each timestep to obtain t_f^*, then only the first desired thrust vector is taken and blended with the desired moment vector to get the throttle and gimbal angle. The other trajectory is the optimal trajectory that is obtained by solving Problem 4 with t_f^* at the very first timestage. This trajectory is the one that would be followed if the vehicle were modeled as a point mass, and thus is a 3DOF trajectory.

As you can see from the figures, the 6DOF MPC trajectory takes a wider turn to get to the landing site. This is caused by the lag that attitude dynamics induces between the desired thrust vector and the actual thrust vector. The quicker the attitude controller, the less noticable this lag will be.

B. 6DOF MPC with Disturbances

When I started adding disturbances and attempted to solve the 6DOF MPC trajectories using Problem 4, the trajectory would be partially generated, but then the solver would fail to find a solution. I realized that with disturbances, some of the constraints in problem 4 became too restrictive to allow for a feasible solution, especially if the vehicle is pushed significanly off-course by the disturbances. For example, it may not be possible to find a solution that lands exactly at the origin if the vehicle is near the ground, but far from the origin, similarly it may not be possible to find a solution where the vehicle touches down with zero velocity. To solve these issues, I modified Problem 4 into Problem 5, which is given below:

Problem 5

$$\min_{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{T-1},\sigma_0,\ldots,\sigma_{T-1}} \quad -z_T + \lambda\eta$$

$$\text{subject to} \quad \boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \dot{\boldsymbol{r}}_i\Delta t + \tfrac{1}{2}(\boldsymbol{u}_i - \boldsymbol{g})\Delta t^2$$
$$\dot{\boldsymbol{r}}_{i+1} = \dot{\boldsymbol{r}}_i + (\boldsymbol{u}_i - \boldsymbol{g})\Delta t$$

$$z_{i+1} = z_i - \alpha\sigma_i\Delta t$$

$$\boldsymbol{u}_i \cdot \boldsymbol{e}_3 \leq \sigma_i cos(\theta_{max})$$

$$\|\boldsymbol{u}_i\| \leq \sigma_i$$

$$\mu_{1,i}\left[1 - (z_i - z_{0,i}) + \tfrac{(z_i - z_{0,i})^2}{2}\right] \leq \sigma_i \leq \mu_{2,i}[1 - (z_i - z_{0,i})]$$

$$\ln(m_{wet} - \alpha\rho_2 i\Delta t) \leq z_i \leq \ln(m_{wet} - \alpha\rho_1 i\Delta t)$$

$$z(0) = \ln(m_{wet}) \quad z(T) \geq \ln(m_{dry})$$

$$\boldsymbol{r}(0) = \boldsymbol{r}_0 \quad \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0 \quad 0 \leq \boldsymbol{r}(T) \cdot \boldsymbol{e}_3 \leq \epsilon \quad \|\dot{\boldsymbol{r}}(T)\| \leq \delta$$

$$\eta \geq \|S\boldsymbol{r}(T)\|$$
$$\|S\boldsymbol{r}_i\| + c^\top \boldsymbol{r}_i \leq 0$$

Where η is a new parameter that penalizes deviation from landing at the origin, λ is a landing error aversion parameter that can be tuned to penalize fuel consumption and landing error by different amounts, δ is some acceptable maximum terminal speed that is close to zero, and ϵ is some acceptable terminal height that should be very close to zero. All of these parameters, except η can be tuned to relax the constraints which makes the system more robust to

disturbances rather than binding the terminal state at zero position and velocity, which will frequently lead to no feasible solution under disturbances.

Figures (10) and (11) are generated from a 6DOF MPC simulation with disturbances. I modeled disturbances as additive forces and moments. The standard deviation of the disturbance force was 3000 N and the standard deviation of the disturbance moment was 3000 N.

In figure (10) the red quivers are the thrust vectors at each timestep, and the black lines are projections of the trajectory onto the XY, YZ, and XZ planes. From figure (11), we can see that the optimal control law tends to either apply maximum thrust or minimum thrust. This seems pretty intuitive, since it acts similar to a bang-bang controller, which would be the optimal control law for powered descent in the 1D case.

C. Monte-Carlo

Finally, I ran a set of 150 Monte-Carlo runs where I varied the initial position and velocity of the vehicle and repeatedly ran the 6DOF MPC simulation with disturbances to generate a set of landing positions. The initial position and velocity were gaussian with mean and covariance matrices given below.

$$\mu_r = \begin{bmatrix} 0 \\ 0 \\ 2000 \end{bmatrix} \quad m \tag{30}$$

$$\Sigma_r = \begin{bmatrix} 250000 & 0 & 0 \\ 0 & 250000 & 0 \\ 0 & 0 & 10000 \end{bmatrix} \quad m^2 \tag{31}$$

$$\mu_v = \begin{bmatrix} 0 \\ 0 \\ -50 \end{bmatrix} \quad m/s \tag{32}$$

$$\Sigma_v = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \quad m^2/s^2 \tag{33}$$

Figure (12) shows the landing locations of the 150 trials. All but five of the landings are shown in this figure. As can be seen from the figure, the majority of landings occur within a two meter radius of the intended landing site.

IX. Future Work

In this paper, I did not consider attitude dynamics or disturbances in the formulation of the convex optimization problem. In the future, I would like to do some work that explicity considers attitude dynamics, for example Successive Convexification [5]. I would also like to consider disturbances and use techniques from [6].

X. Conclusion

In this paper I have recreated JPL's G-FOLD algorithm based on papers [1] and [2], and I have implemented it as a model predictive controller. Then, I reformulated the discretized problem to be more robust to disturbances and conducted a set of Monte-Carlo trials to obtain a landing ellipse. I have learned a great deal about convex optimization, MPC, and writing simulations from this project. I would like to thank Kyle Krol for his help with debugging parts of my simulation and other guidance during this project. The codebase for this project can be found at `https://github.com/govindchari/nsim`.

References

[1] Acikmese, B., and Ploen, S. R., "Convex Programming Approach to Powered Descent Guidance for Mars Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353–1366. https://doi.org/10.2514/1.27553, URL https://doi.org/10.2514/1.27553.

[2] Açıkmeşe, B., Casoliva, J., Carson, J., and Blackmore, L., "G-FOLD: A Real-Time Implementable Fuel Optimal Large Divert Guidance Algorithm for Planetary Pinpoint Landing," *LPI Contributions*, 2012, pp. 4193–.

[3] Domahidi, A., Chu, E., and Boyd, S., "ECOS: An SOCP solver for embedded systems," *European Control Conference (ECC)*, 2013, pp. 3071–3076.

[4] Niederberger, S., "Epigraph," https://github.com/EmbersArc/Epigraph, 2020.

[5] Szmuk, M., Reynolds, T. P., and Açıkmeşe, B., "Successive Convexification for Real-Time Six-Degree-of-Freedom Powered Descent Guidance with State-Triggered Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 8, 2020, p. 1399–1413. https://doi.org/10.2514/1.g004549, URL http://dx.doi.org/10.2514/1.G004549.

[6] Ridderhof, J., and Tsiotras, P., "Minimum-fuel Powered Descent in the Presence of Random Disturbances," 2019. https://doi.org/10.2514/6.2019-0646.

Appendix

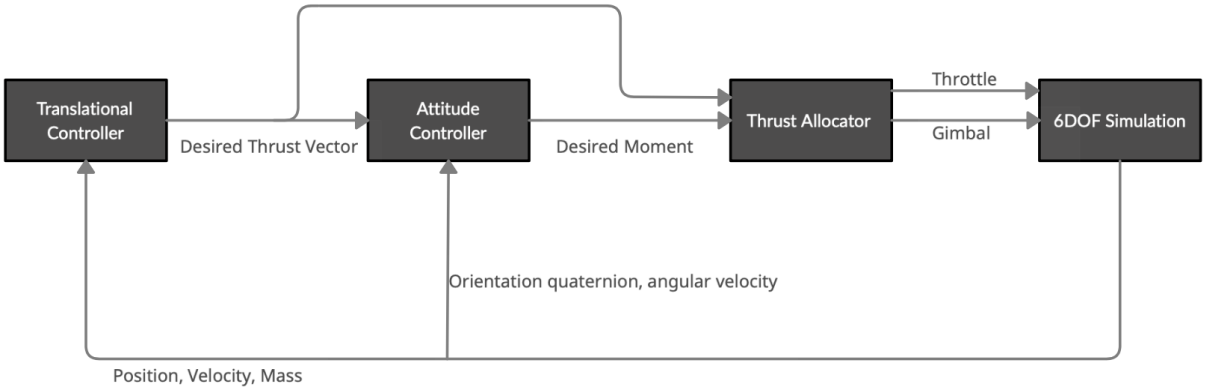

Fig. 1 Block Diagram

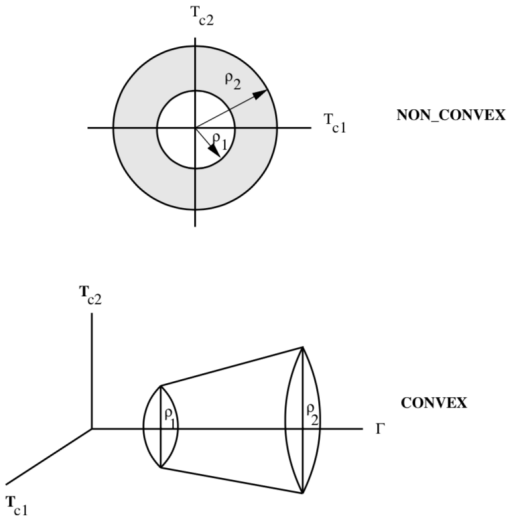

Fig. 2 Nonconvex and Convex Thrust Bounds [1]

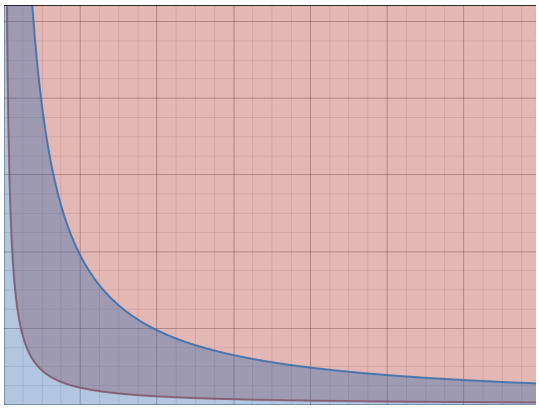

Fig. 3 Nonconvexity of Equation (14)

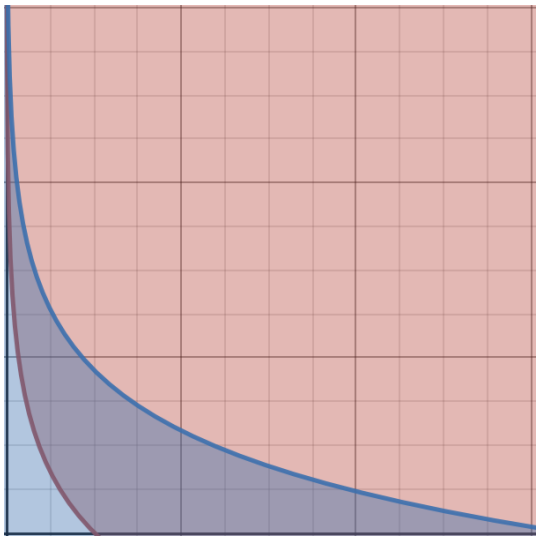

Fig. 4 Nonconvexity of Equation (17)

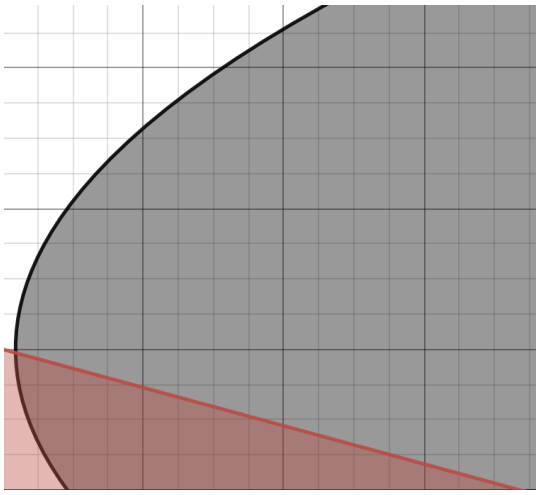

Fig. 5 Convexity of Equation (18)

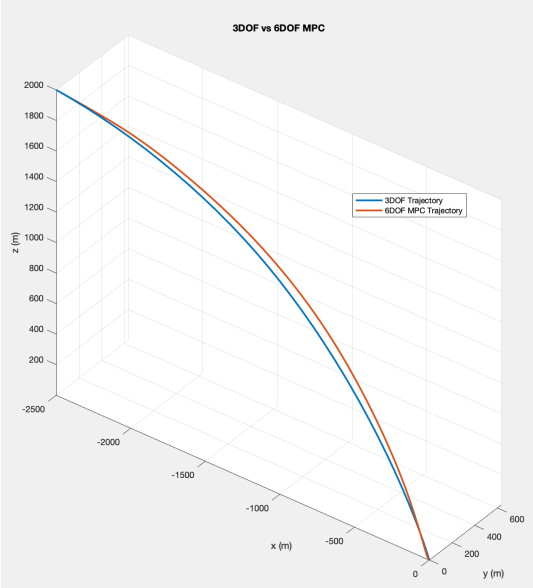

Fig. 6 Isometric view of 6DOF MPC and 3DOF Trajectories

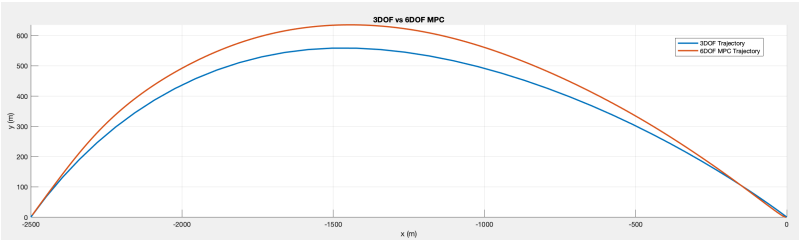

Fig. 7 XY Projection of 6DOF MPC and 3DOF Trajectories

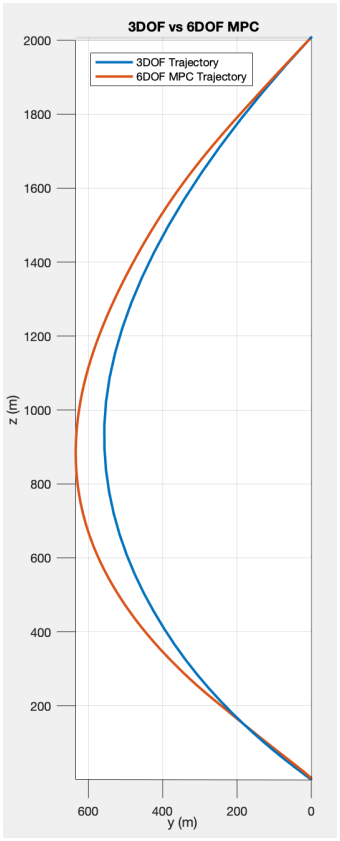

Fig. 8 YZ Projection of 6DOF MPC and 3DOF Trajectories

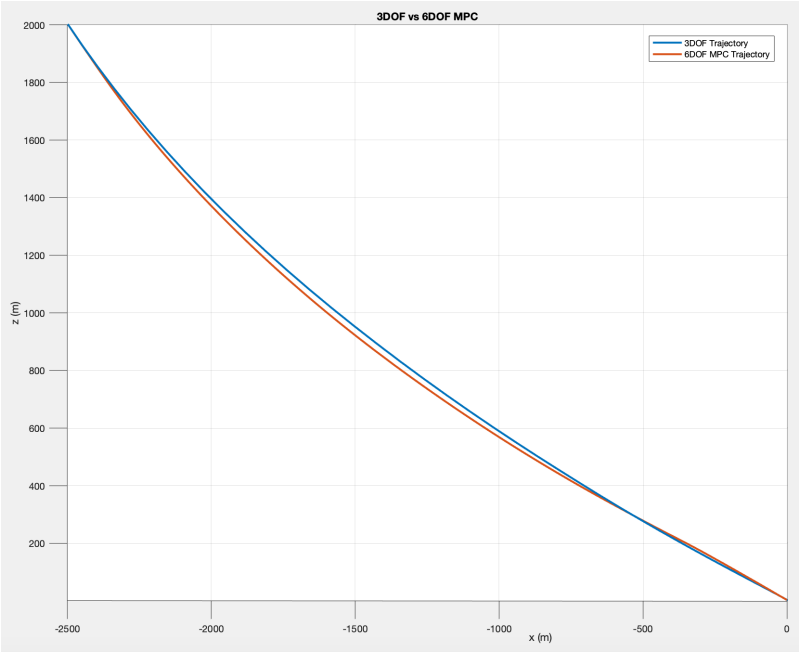

Fig. 9 XZ Projection of 6DOF MPC and 3DOF Trajectories

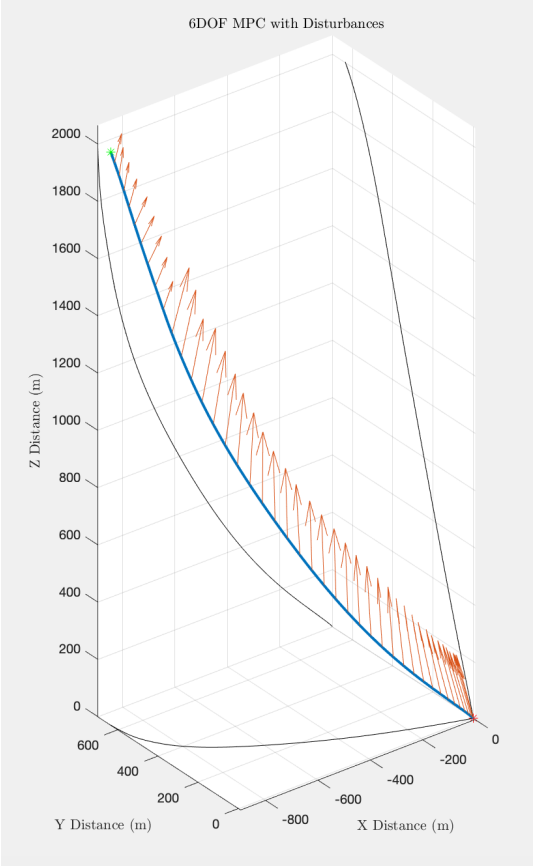

Fig. 10 6DOF MPC Trajectory with Disturbances

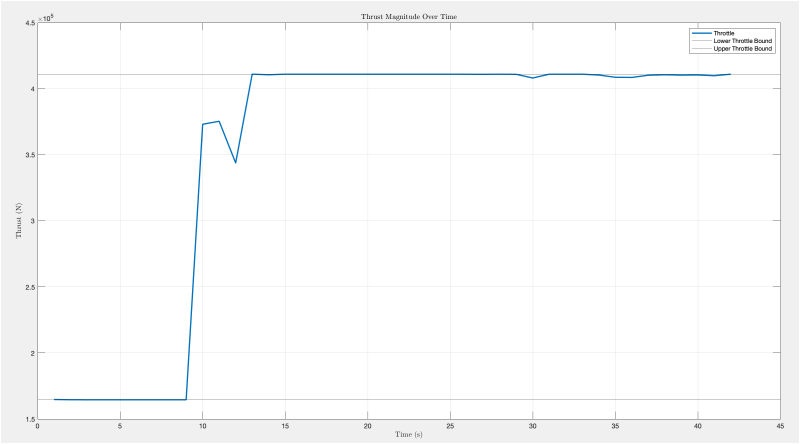

Fig. 11 Throttle for 6DOF MPC Trajectory with Disturbances

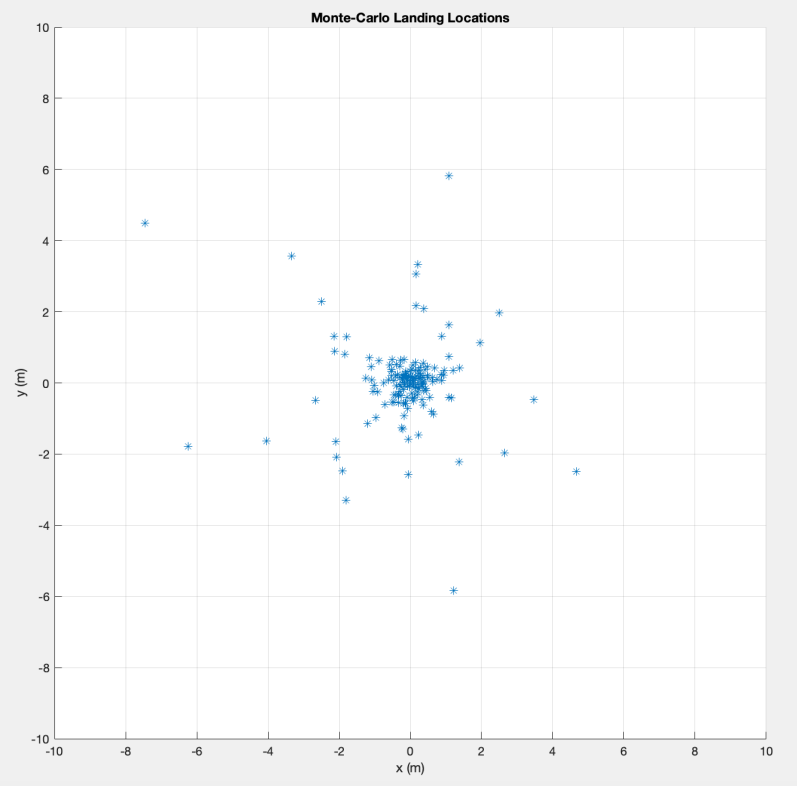

Fig. 12 Results from Monte-Carlo Simulation