Convex Programming Approaches to Pinpoint Fuel-Optimal Planetary Landing with Non-Convex Constraints

Pádraig Lysandrou*
Cornell University, Ithaca, New York, 14853

In this paper, I present a summary and implementations of work done in convex programming approaches [1][2][3][4] to the fuel-optimal powered descent, or "soft-landing", problem. These trajectory optimization problems are nontrivial due to nonconvex 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. I will initially pose and solve a single trajectory optimization program with a time-of-flight search and then explore a successive approach. The successive approach, with nonlinearities, mass-depletion dynamics, and free final time presents a challenge for real-time implementation. To provide a tractable solution we must employ a methods of lossless convexification, linearization, trust regions, and relaxations. Again formulating the problem as an SOCP and initialized with a coarse time-of-flight estimation, it can be solved iteratively, hence successive. These algorithms can be implemented on autonomous real-time systems with powerful modern Interior Point Method (IPM) solvers. This is written as the final project report for Cornell Engineering course ECE5555 Stochastic Systems: Estimation and Control.

Nomenclature

r	=	position
$\mathbf{T}$	=	Thrust vector (N)
g	=	Gravity vector of planet (m/s^2)
$m(t)$	=	Mass of vehicle
α	=	Constant describing mass consumption rate
T_1	=	Lower Thrust Bound
T_2	=	Upper Thrust Bound
ρ_1	=	Lower Thrust Bound
ρ_2	=	Upper Thrust Bound
m_{wet}	=	The total mass of the vehicle including propellant
t_f	=	Length of finite time horizon
x	=	Augmented state vector defined as $[r, \dot{r}]^T$
$r_z(t)$	=	Vertical position
$r_x(t)$	=	Downrange position
Θ_{alt}	=	Glideslope constraint
S_j, v_j, a_j	=	Matrices, vectors, and scalars defining convex state constraints
I_{sp}	=	Specific impulse of the rocket engine
u	=	Control acceleration put on by thrusters
Γ	=	Slack variable required for convexification
log	=	Natural logarithm
$\|\|x\|\|$	=	Euclidean 2-norm of vector x

*Undergraduate Student, Electrical and Computer Engineering, AIAA Student Member

I. Introduction

In this paper, I present implementations of convex programming algorithms for the powered descent guidance problem. I will focus on these two papers:[1][2]. Traditional landing algorithms have tolerances of tens of kilometers, where the one presented has tolerances as little as meters. So far, all Mars landings have been for exploration missions with relaxed landing criteria. However, given that landing has gained renewed interest due to privatization and commercialization of space exploration, other techniques are being researched. 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 industry and has already had a huge impact in reducing the cost of getting to space. We have seen companies like SpaceX and Blue Origin make incredible progress with landing reusable vehicles and expect more developments from them in the future.

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 from initial condition to terminal state. We define powered descent guidance as the fuel-optimal trajectory that takes the vehicle at some initial state condition to a prescribed final state in a uniform gravitational field with standard vehicle given thrust magnitude and direction constraints. I will start with the two-dimensional formulation and extend it to three dimensions in my implementation.

The convex optimization framework is exploited because it is conducive of real-time onboard 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 onboard implementation. I will summarize the problem given in [1], provide a solution, extend it to three dimensions, and then summarize the successive convexification approach found in [3] with my own successive implementation under noise.

II. The General Problem[1]

Given that we will solve our system in two dimensions (x and z), the position and velocity vectors will be of size $\mathbb{R}^2$. We have assumed everything is the the *surface fixed frame*. The translational dynamics of the system are given as

$$\ddot{r}(t) = g + \frac{T_c(t)}{m(t)} \tag{1}$$

$$\dot{m}(t) = -\alpha \|T_c(t)\| \tag{2}$$

Where $\ddot{r}(t) \in \mathbb{R}^2$ is acceleration of the vehicle, g is the gravitational acceleration vector of the planet, $T_c(t)$ is the thrust vector, $m(t)$ is the mass of the vehicle, and α is a positive constant describing the fuel consumption rate. For n identical thrusters each with nominal thrust level T and specific impulse I_{sp} we can denote $\boldsymbol{T}_c = (nT\cos\phi)\boldsymbol{e}$ and $\alpha = (I_{sp} g \cos\phi)^{-1}$ as the consumption rate with ϕ being the cant angle of each engine. However, moving forward I will assume one engine per spacecraft and, therefore, a cant angle of 0 degrees.

Engine thrust cannot be continuously throttled down to 0 and therefore has upper and lower bounds

$$0 < T_1 < T(t) < T_2, \qquad \forall t \in [0, t_f] \tag{3}$$

$$\rho_1 \leq \|T(t)\| \leq \rho_2 \tag{4}$$

$$\rho_2 > \rho_1 > 0 \tag{5}$$

Where ρ_1 is the deepest throttle and ρ_2 is the nominal largest thrust value. The final time is denoted by t_f. Note that eqn.4 defines a nonconvex region in the control space. We must also provide the initial conditions: $m(0) = m_{wet}$, $x(0) = x_0$, $x(tf) = 0$, where x denotes the state. The value m_{wet} is the wet mass of the rocket. This means the mass of the structure, avionics, and other non-consumables plus the mass of the fuel in the tanks. The state is constructed as such: $x = [r \quad \dot{r}]^T$. We also assume that $r_z(t) > 0, \quad \forall t \in [0, t_f)$ which is to say that the vehicle cannot travel through the ground.

Another set of constraints that we must consider are those imposed upon the system by instruments and sensors. For example, when landing on a planetary body without global positioning, terrain relative navigation might be used and restrict the attitude of the vehicle. For this we implement what is called the glideslope constraint.

$$\Theta_{alt} = \arctan \frac{r_z(t)}{r_x(t)} \tag{6}$$

Additionally, it is assumed that constraints take the following affine form:

$$||S_j x(t) - v_j|| + c_j^T x(t) + a_j \leq 0 \tag{7}$$

where $S_j \in \mathbb{R}^{n_j * 4}$, $c_j \in \mathbb{R}^4$, $n_j \leq 4$, $v_j \in \mathbb{R}_j^n$, and $a_j \in \mathbb{R}$. Similarly we can express the glideslope constraint like

$$||Sx|| + c_j^T x \leq 0 \tag{8}$$

where

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \qquad c = \begin{bmatrix} -\tan \Theta_{alt} & 0 & 0 & 0 \end{bmatrix}^T$$

So we can now pose

Problem 1:

$$\max_{t_f, T} m(t_f) = \min_{t_f, T} \int_0^{t_f} ||T(t)|| dt$$

Subject to:

$$\ddot{r}(t) = g + \frac{\boldsymbol{T}_c(t)}{m(t)}, \quad \dot{m}(t) = -\alpha ||T_c(t)||$$

$$\rho_1 \leq ||T(t)|| \leq \rho_2, \quad r_z(t) \leq 0$$

$$||S_j x(t) - v_j|| + c_j^T x(t) + a_j \leq 0, \quad j = 1, ..., n_s$$

$$m(0) = m_{wet}, r(0) = r_0, \dot{r}(0) = \dot{r_0}, r(t_f) = \dot{r}(t_f) = 0$$

III. Convexification of the Control Magnitude Constraint

I will now introduce a modified version of Problem 1:

Problem 2:

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

Subject to:

$$\ddot{r}(t) = g + \frac{\boldsymbol{T}_c(t)}{m(t)}, \quad \dot{m}(t) = -\alpha ||\Gamma(t)||$$

$$||T(t)|| \leq \Gamma(t)$$

$$\rho_1 \leq \Gamma(t) \leq \rho_2, \quad r_z(t) \leq 0$$

$$||S_j x(t) - v_j|| + c_j^T x(t) + a_j \leq 0, \quad j = 1, ..., n_s$$

$$m(0) = m_{wet}, r(0) = r_0, \dot{r}(0) = \dot{r_0}, r(t_f) = \dot{r}(t_f) = 0$$

This is derived by introducing Γ as a slack variable to the initial problem. Γ replaces $||T||$ with an additional constraint $||T(t)|| \leq \Gamma(t)$. Because this problem is a relaxation of the initial problem, Problem 1 defines a feasible solution of Problem 2. However, a feasible solution for problem 2 does not necessarily define one for Problem 1. Lemma 1 of

[1] shows that the optimal solution to Problem 2 also is a feasible solution of Problem 1. This is done by using the Hamiltonian of Problem 2 and using the necessary conditions for optimality, pointwise maximum principle, and the transversality condition. This paper shows, with lemma 1, that modified constraints of Problem 2, namely the scalar slack variable Γ convexifies Problem 1. The lemma states that if there is an optimal solution to Problem 2, then there also exists one for Problem 1, and it can be found from the optimal solution of Problem 2. We also see that that the admissible set of controls for the second problem are within the set of admissible controls for the first. Together this means that Problem 2 can be solved with convex control constraints in order to find a solution to the nonconvex Problem 1.

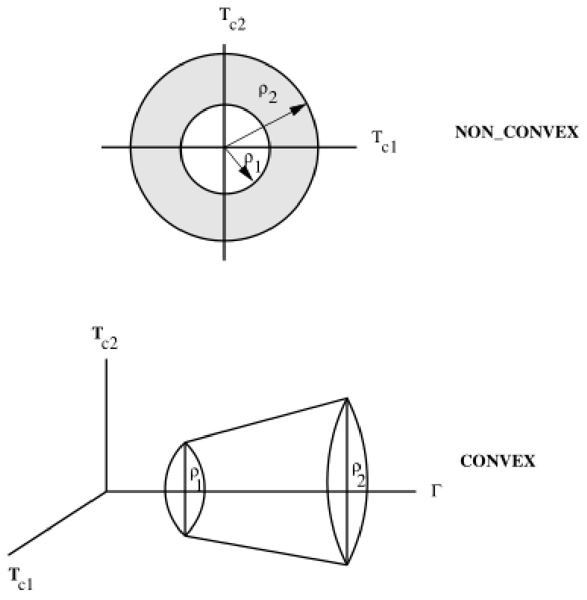


Fig. 1 Problem 1 to Problem 2 convexification of the thrust constraint. From [1]

See Fig. 1 to see how the original nonconvex annulus turns into a three-dimensional convex cone. The [1] paper gives a optimal solution existence proof and continues onto a change of variables step which will make construction of the algorithm much simpler.

A. Change of Variables

A change of variables will allow for a much cleaner numerical algorithm for expanding Problem 2 into the third form which will be a continuous time SOCP. Let us perform

$$\sigma = \frac{\Gamma}{m} \tag{9}$$

$$u = \frac{T}{m} \tag{10}$$

and consequently the dynamics change

$$\ddot{r}(t) = u(t) + g \tag{11}$$

$$\frac{\dot{m}(t)}{m(t)} = -\alpha\sigma(t) \tag{12}$$

$$\text{and clearly } m(t) = m_0 \exp\big[-\alpha \int_0^{t_f} \sigma(t) d\tau\big] \tag{13}$$

Therefore we must minimize the integral term in the last mass dynamical equations. The original constraints the follow that

$$\|u(t)\| \leq \sigma(t), \quad \forall t \in [0, t_f] \tag{14}$$

$$\frac{\rho_1}{m(t)} \leq \sigma(t) \leq \frac{\rho_2}{m(t)}, \quad \forall t \in [0, t_f] \tag{15}$$

This inequality now describes a convex set. However, when we consider m by itself to be variable of the problem, these inequalities become bilinear and do not define a convex region. We will now introduce z to resolve this issue and convexify these inequalities. Lets define $z = \log m$. And then the mass depletion rate now becomes $\dot{z}(t) = -\alpha\sigma(t)$. The inequalities now become

$$\rho_1 \exp(-z(t)) \leq \sigma(t) \leq \rho_2 \exp(-z(t)). \quad \forall t \in [0, t_f] \tag{16}$$

The left side of the inequality in eqn. 16 defines a convex feasible region, but the right part does not. We will now use a Taylor expansion of the exponential to achieve second-order cone and linear approximations to be used in our problem. The first part of this eqn. 16 can be approximate by the first three terms of the series, or second-order cone:

$$\rho_1 \exp(-z_0)\left[1 - (z - z_0) + \frac{(z - z_0)^2}{2}\right] \leq \sigma \tag{17}$$

Where z_0 is a given constant. The right side of the inequality can be linear, or the first two terms of the Taylor expansion. Therefore

$$\sigma \leq \rho_2 \exp(-z_0)\left[1 - (z - z_0)\right] \tag{18}$$

$$\text{with } \mu_1 = \rho_1 \exp(-z_0), \text{and } \mu_2 = \rho_2 \exp(-z_0) \tag{19}$$

We now have a second order cone formulation where $z_0(t) = \log(m_{wet} - \alpha\rho_2 t)$ where $z_0(t)$ serves as a lower bound on $z(t)$ at each time. We must now ensure that $z(t)$ is constrained properly with

$$\log(m_{wet} - \alpha\rho_2 t) \leq z(t) \leq \log(m_{wet} - \alpha\rho_1 t) \tag{20}$$

Let us know pose the third problem, substituting in these simplifications, which remains continuous:

Problem 3:

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

Subject to:

$$\begin{aligned}
\ddot{r}(t) = u(t) + g, \quad \dot{z}(t) = -\alpha\sigma(t) \\
\|u(t)\| \leq \sigma(t) \\
\mu_1(t)\left[1 - (z - z_0) + \frac{(z - z_0)^2}{2}\right] \leq \sigma \leq \mu_2(t)\left[1 - (z - z_0)\right] \\
\log(m_{wet} - \alpha\rho_2 t) \leq z(t) \leq \log(m_{wet} - \alpha\rho_1 t) \\
\|r_x(t)\| \leq \beta r_z(t) \\
m(0) = m_{wet}, r(0) = r_0, \dot{r}(0) = \dot{r}_0, r(t_f) = \dot{r}(t_f) = 0
\end{aligned}$$

Note that we have replaced the general affine constraint form with the glideslope constraint, where β is the defined slope.

IV. Discretization

Discretization must occur for implementation in computer program. We base this on a piecewise linear control input for different time intervals. This means that for $\Delta t > 0$, and for each instance $t_k = k\Delta t, \forall k = 0, ..., N$. Therefore we replace the continuous time with

$$u(t) = u_k + (u_{k+1} - u_k)\tau \tag{21}$$

$$\sigma(t) = \sigma_k + (\sigma_{k+1} - \sigma_k)\tau \tag{22}$$

$$\text{where } \tau = \frac{t - t_k}{\Delta t}, \quad \forall t \in [t_k, t_{k+1}), \quad k = 0, ..., N-1 \tag{23}$$

Using this discretization, we present Problem 4:

Problem 4:

$$\min_{u_0...u_N, \sigma_0...\sigma_N} -z_N$$

Subject to: for $k = 0, ..., N$,

$$r_{k+1} = r_k + \frac{\Delta t}{2}(\dot{r}_k + \dot{r}_{k+1}) + \frac{\Delta t^2}{12}(u_{k+1} - u_k)$$

$$\dot{r}_{k+1} = \dot{r}_k + \frac{\Delta t}{2}(u_k + u_{k+1}) - g\Delta t$$

$$z_{k+1} = z_k - \frac{\alpha \Delta t}{2}(\sigma_k + \sigma_{k+1})$$

$$\|u_k\| \leq \sigma_k$$

$$\mu_{1,k}(t)\left[1 - (z_k - z_{0,k}) + \frac{(z_k - z_{0,k})^2}{2}\right] \leq \sigma \leq \mu_{2,k}(t)\left[1 - (z_k - z_{0,k})\right]$$

$$\log(m_{wet} - \alpha\rho_2 k\Delta t) \leq z_k \leq \log(m_{wet} - \alpha\rho_1 k\Delta t)$$

$$\|r_x(t)\| \leq \beta r_z(t)$$

$$m(0) = m_{wet}, r(0) = r_0, \dot{r}(0) = \dot{r}_0, r(t_f) = \dot{r}(t_f) = 0, z_0 = log(m_{wet}), N\Delta t = t_f$$

We now have the general lossless-convexified algorithm and can use vehicle parameters to generate trajectories.

V. 2D Mars Landing Divert for Small Single-Engined Vehicle

I have now implemented this algorithm in MATLAB using the CVX package to define the convex problem. I use the SEDUMI solver. In Fig.4 we see the parameters that I have given the vehicle. I have modeled these off of parameters similar to Masten Space System's vertical takeoff and vertical landing class of vehicles. The specific impulse reflects that of an ethanol liquid oxygen rocket engine, as well as the throttleability of their engine. Notice that the gravity vector reflects that of Mars. I have also limited the maximum thrust vector control angle to 25 degrees as a constraint on the thrust vector. This is a feasible upper limit for agile landing thrust vector control systems. Notice the initial conditions placed on the position and velocity. These values are consistent with those of Mars entry after the drag/parachute phase. The trajectory in Fig.5 shows how the vehicle overshot the target during the entry phase and how it works its way back towards the landing site. Keep in mind that the t_f in this situation is static at 100 seconds and is not necessarily globally optimal because we have yet to implement the time-of-flight search. Fig. 6 shows the position and velocity of the vehicle along the path. Additionally, I have rewritten the vehicle parameters, environmental parameters, and initial conditions to simulate the re-entry of SpaceX's Falcon 9 vehicle as it approaches the landing barge. Note that this was done as a demonstration and does not accurately represent their implementation as it does not include aerodynamic control surfaces or disturbances.

VI. 3D Implementation of the Landing Algorithm

Now that I have shown the mathematical derivation for the two dimensional case, I will extend it to three dimensions in the MATLAB simulation. This was very simple as I augmented the dynamics, gravity, and input vectors. The thruster now has two rotational degrees of freedom. I also re-derived the glideslope constraint and thrust vector control constraints for the new degree of freedom. I will include plots of trajectories using both the 3D algorithm and the time-of-flight search algorithm in the appendix for brevity. I will reference them in the following section.

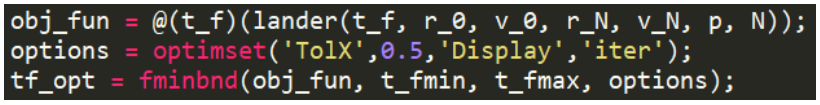
VII. Time-of-Flight (TOF) Search

Now that we can solve the landing problem with static final time, we must perform this SOCP multiple times, performing a linear search, with multiple times in mind to find the global minimum fuel consumption. For example, even though a feasible solution to the convex problem exists with a final time of 150 seconds, it might be suboptimal to the SOCP solution generated with a 148 second time-of-flight. Therefore we must perform an outer iteration of the time horizon to find the globally optimal value corresponding to total minimum fuel consumption.

We will use the Golden search algorithm, a nonlinear program, to determine this optimal time-of-flight value $t_f^\star$. The minimum fuel is a unimodal function of time and has a global minimum and we can perform another optimization to find the optimal value. Therefore, it can be said that we are performing an outer search, where each value of the search is calculated by a convex optimization problem. This is the most time intense portion of this approach and is the least conducive towards onboard implementation. Additionally, it should be noted that finding the proper upper and lower bounds for t_f is nontrivial and a problem on its own.

For the sake of complexity abstraction, I have decided to use MATLAB's $fminbnd$ function to perform this search for me. This function performs:

$$\min_{t_f} lander(t_f, r_0, v_0, r_N, v_N, params, N), \quad \forall t_{min} \leq t_f \leq t_{max} \tag{24}$$



```
obj_fun = @(t_f)(lander(t_f, r_0, v_0, r_N, v_N, p, N));
options = optimset('TolX',0.5,'Display','iter');
tf_opt = fminbnd(obj_fun, t_fmin, t_fmax, options);
```

Fig. 2 Implemented in MATLAB

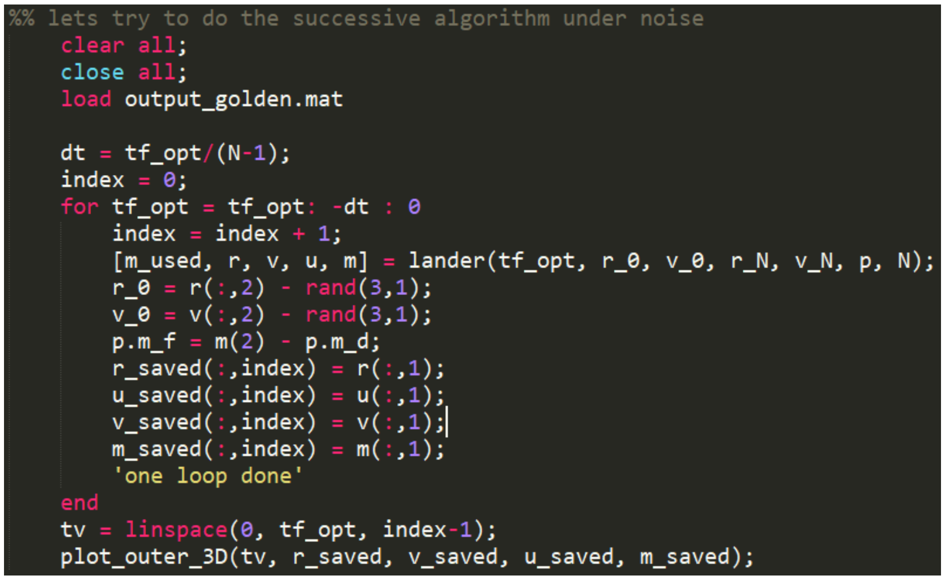
Fig. 2 shows the MATLAB implementation referencing the 3D landing algorithm function `lander.m`. The inputs to the function are the terminal time value, the initial state conditions, terminal state conditions, the vehicle parameters stored in struct `p`, and the granularity of the trajectory generation N. The lander function outputs the mass used, state information, as well as control effort vector u. The `fminbnd` function finds the minimum output of the function for varying the terminal time input value. This means that it takes the Euclidean norm of `lander`'s output vectors and then compares them along its search to find the global minimum. For my experimentation, I have not attempted to derive proper minimum or maximum values for this iteration. Instead, if the individual optimization is infeasible, I set the output consumption value to an arbitrarily large value. This allows me to set the minimum time to zero and the maximum time to an arbitrary value that I know is too large through experimentation.

An output of this search algorithm can be seen in Fig. 9. The x column indicates the terminal time values that have been tested while the $f(x)$ column indicates the total mass consumed in that iteration. Notice that for the infeasible problems, the mass is 2000 which is the arbitrarily large value I have given it. It is the sum of the initial dry and fuel masses. Notice that the search algorithm starts using the golden search method in the initial iterations and then uses a parabolic algorithm to reach the tolerance setting.

This TOF search took 177.585254 seconds to perform. This can drastically be decreased by minimizing the granularity of the calculation and decreasing the minimum and maximum terminal time boundaries.

VIII. The Successive Convexification Approach

I will now show my efforts in creating a successive algorithm that successively solves the convex problem at each time step under estimation noise. This effectively makes the system closed-loop as it recalculates the trajectory after each input and estimation value. This is, again, another part that is very hard to implement online but can be done with proper computing power.



```
%% lets try to do the successive algorithm under noise
    clear all;
    close all;
    load output_golden.mat

    dt = tf_opt/(N-1);
    index = 0;
    for tf_opt = tf_opt: -dt : 0
        index = index + 1;
        [m_used, r, v, u, m] = lander(tf_opt, r_0, v_0, r_N, v_N, p, N);
        r_0 = r(:,2) - rand(3,1);
        v_0 = v(:,2) - rand(3,1);
        p.m_f = m(2) - p.m_d;
        r_saved(:,index) = r(:,1);
        u_saved(:,index) = u(:,1);
        v_saved(:,index) = v(:,1);
        m_saved(:,index) = m(:,1);
        'one loop done'
    end
    tv = linspace(0, tf_opt, index-1);
    plot_outer_3D(tv, r_saved, v_saved, u_saved, m_saved);
```

Fig. 3 Successive Algorithm

In Fig. 3 we see the general layout of the successive algorithm. First we clear all of the available information and load the output from the TOF search algorithm. The only data we really use are the vehicle parameters, optimal terminal time value, and other constants. We define the time step by the optimal terminal time divided by the total granularity. We then iteratively solve the `lander.m` algorithm with mass depletion dynamics additive noise to the state vectors. The `randn` function in MATLAB returns a random scalar drawn from the standard normal distribution. This is not how aerodynamic uncertainty works, but I am doing this for simplicity and can make it more complicated later. Note that the struct value of fuel mass consistently changes with the known consumption from the last iteration. I then save the values for display later.

Please see Fig. 10 and Fig. 11 for the outputs of the 3D successive algorithm for the originally posed Mars Landing problem. You will see that the trajectory generation is unfinished. This is unfortunately because of my own time constraints and a small glitch. From the data shown, the simulation runs very well and stays within all given constraints. At the last time point shown, the vehicle had only consumed about 150kg of propellant with an additional 350 to its disposal. In optimal conditions it should only consume 315 kg given by Fig. 9. Therefore, the unfinished trajectory and data look very correct and follow the optimal path. Therefore it looks promising and robust under noise. I will continue to work on this solution after the end of the course.

IX. Future Work and Implementation of Successive Algorithm

After the completion of ECE5555, I plan on completing and polishing up my successive solution. I also plan on autocoding my algorithm in C/C++ using the CVXgen tool. I will run it on a small embedded Linux platform and see what kind of cycle times I can get as it converges to an optimal trajectory. Additionally, I plan on implementing an offline version of the algorithm on a small electric-ducted-fan vehicle that will simulate the inertial properties of a rocket body. I will attempt to land this vehicle with some initial conditions.

X. Conclusion

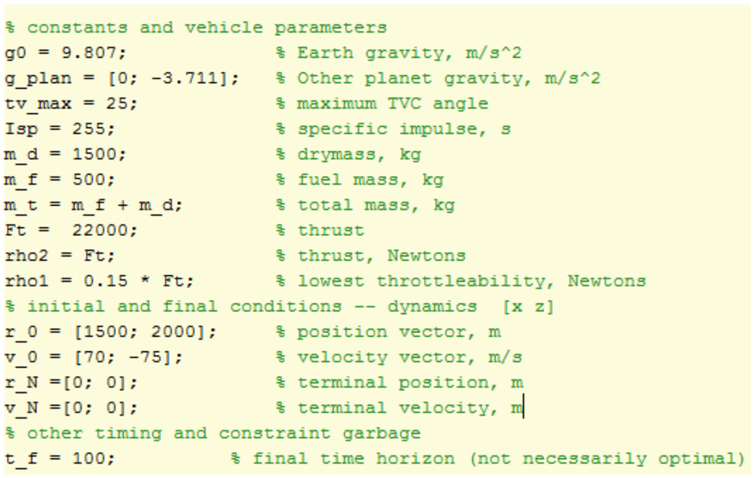
I have presented an algorithm that can generate trajectory and control inputs for fuel-optimal landings. This can be used as an offline optimizer or used online in a successive fashion where the optimal time-of-flight is converged upon. This algorithm has many uses and is useful in the landing of reusable launch vehicles back to Earth and research vehicles on other planets. Implementation of algorithms like this have already had a profound impact on the aerospace industry and are the key to reducing cost to space access. I have shown that a successive approach, effectively closed loop, is robust under noise. Overall, I have learned an incredible amount working on this project in my own time and will continue working on it going forward. I would like to thank Prof. Eilyan Bitar for his help and guidance during this project.

A. APPENDIX

A. Problem 4 CVX Program

```
cvx_begin
        variables u(2,N) z(1,N) s(1,N) r(2,N) v(2,N)
        minimize(-z(N))                % objective function
        subject to                             % constraints and dynamics
                r(:,1) == r_0;         % position IC
                v(:,1) == v_0;         % velocity IC
                r(:,N) == r_N;         % position TC
                v(:,N) == v_N;         % velocity TC
                z(1) == log(m_t);      % mass IC
                r(2,:) >= 0;           % ground clearance
                u(2,:) >= s.*cos(degtorad(tv_max));     % thrust vector control constraint
                r(1,:) <= r(2,:)/tan(degtorad(gs));     % glide slope
        z(:) >= 0;
                for  k = 1:N-1
                        r(:,k+1) == r(:,k) + ((dt/2)*(v(:,k) + v(:,k+1))) +(((dt^2)/12)*(u(:,k+1) - u(:,k)));
                        v(:,k+1) == v(:,k) + ((dt/2)*(u(:,k) + u(:,k+1))) +(g_plan*dt);
                        z(1,k+1) == z(1,k) - (((a*dt)/2)*(s(1,k) + s(1,k+1)));
        end
                for k=1:N
                        norm(u(:,k)) <= s(1,k);
                        z_0 = m_t - (a*rho2*dt*(k-1));
                        m_1 = rho1/z_0;
                        m_2 = rho2/z_0;
                        z1 = log(m_t-(a*rho1*dt*(k-1)));
                        z0 = log(z_0);
                        z(1,k) >= z0;
                        z(1,k) <= z1;
                        s(1,k) <= m_2*(1 - (z(1,k) - z0));
                        s(1,k) >= m_1*(1 - (z(1,k) - z0) + (((z(1,k) - z0)^2)/2));
        end
cvx_end
```

B. 2D Figures for Mars Landing With Glideslope Constraint



```
% constants and vehicle parameters
g0 = 9.807;               % Earth gravity, m/s^2
g_plan = [0; -3.711];     % Other planet gravity, m/s^2
tv_max = 25;              % maximum TVC angle
Isp = 255;                % specific impulse, s
m_d = 1500;               % drymass, kg
m_f = 500;                % fuel mass, kg
m_t = m_f + m_d;          % total mass, kg
Ft =  22000;              % thrust
rho2 = Ft;                % thrust, Newtons
rho1 = 0.15 * Ft;         % lowest throttleability, Newtons
% initial and final conditions -- dynamics  [x z]
r_0 = [1500; 2000];       % position vector, m
v_0 = [70; -75];          % velocity vector, m/s
r_N =[0; 0];              % terminal position, m
v_N =[0; 0];              % terminal velocity, m
% other timing and constraint garbage
t_f = 100;          % final time horizon (not necessarily optimal)
```

Fig. 4 Single engined small Mars vehicle parameters.

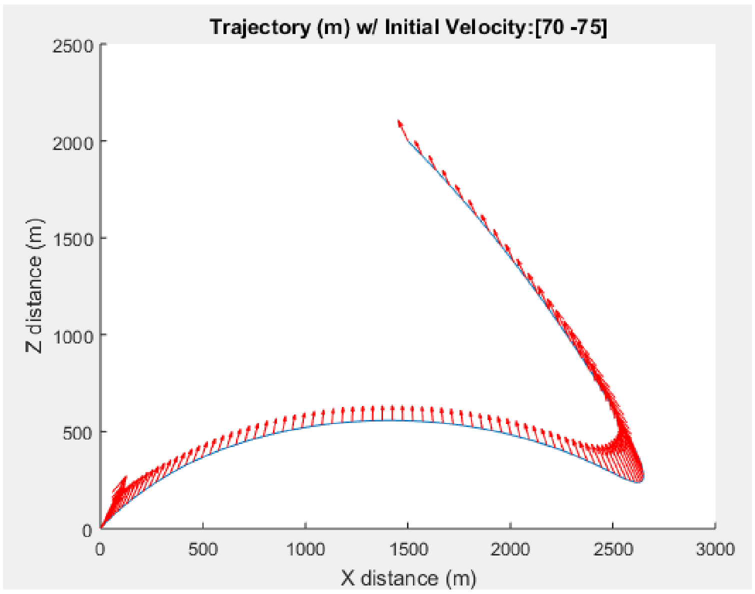


Fig. 5 Single engined small Mars vehicle trajectory generated with static finite time. This shows the magnitude and direction of the thrust as well as outlines the path of movement.

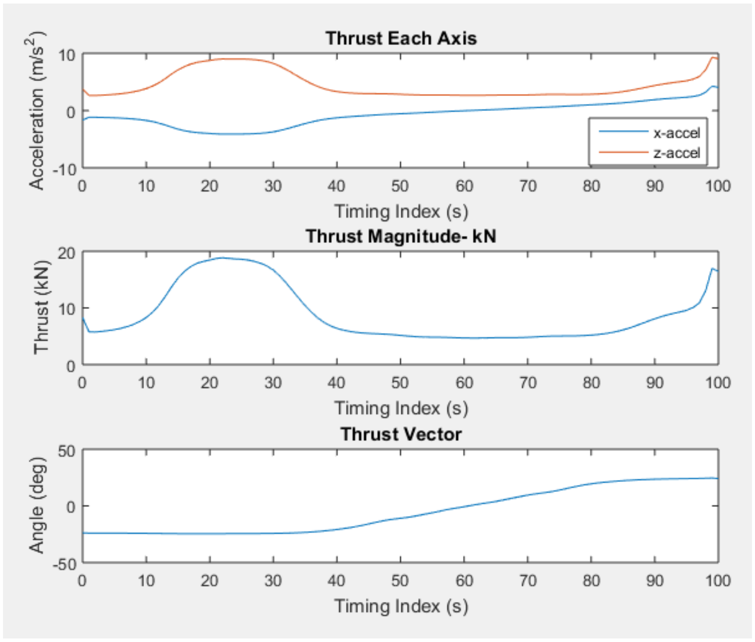


Fig. 6 Single engined small Mars vehicle subplot array shows the position and velocity of the lander along the trajectory.

C. 2D Earth Landing of Falcon 9 Without Glideslope Constraint

D. 3D Mars Landing with Glideslope Constraint, Same Vehicle Parameters as 4, and TOF search

E. Successive Algorithm Figures

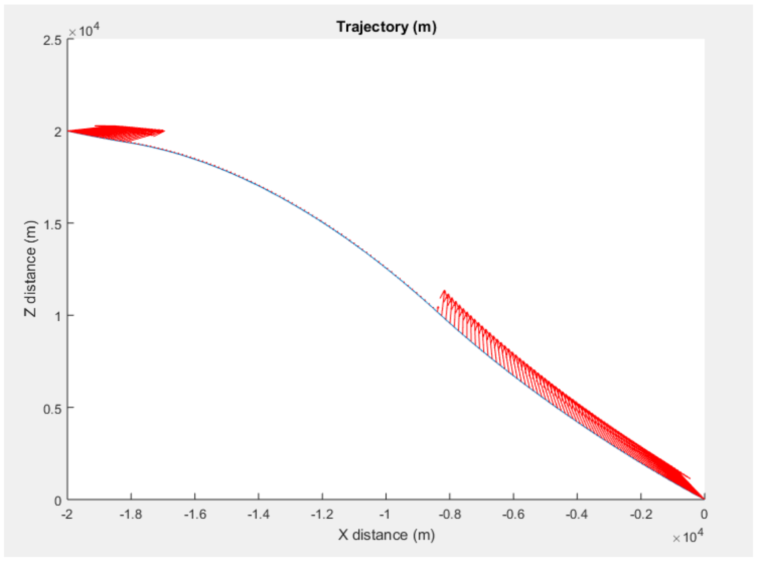


Fig. 7 Earth Landing Trajectory of SpaceX's Falcon 9 first stage rocket as it approaches the barge for landing.

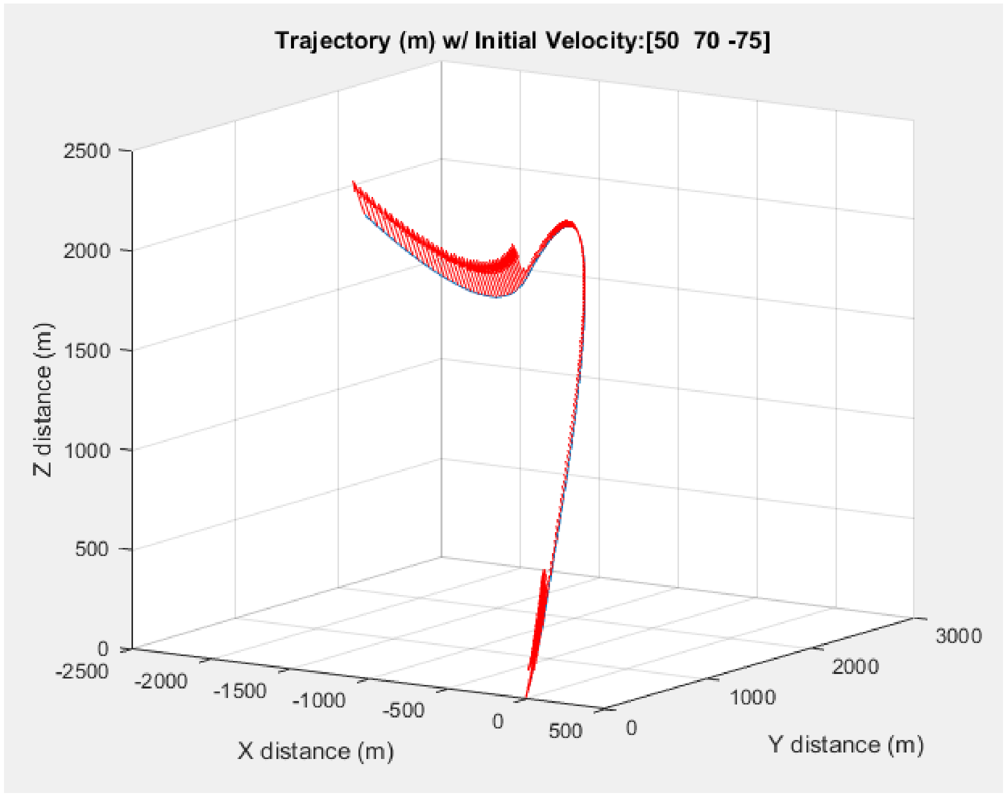
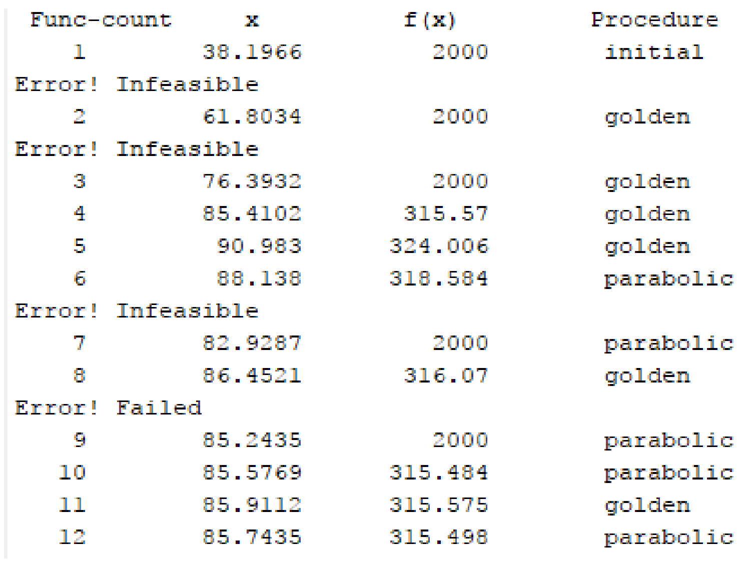


Fig. 8 3D Mars landing trajectory with glideslope constraints and optimal terminal time given from TOF search.



```
 Func-count     x          f(x)         Procedure
    1        38.1966        2000        initial
Error! Infeasible
    2        61.8034        2000        golden
Error! Infeasible
    3        76.3932        2000        golden
    4        85.4102      315.57        golden
    5         90.983     324.006        golden
    6         88.138     318.584        parabolic
Error! Infeasible
    7        82.9287        2000        parabolic
    8        86.4521      316.07        golden
Error! Failed
    9        85.2435        2000        parabolic
   10        85.5769     315.484        parabolic
   11        85.9112     315.575        golden
   12        85.7435     315.498        parabolic
```

Fig. 9 Output from the fminbnd linear search. Notice that it starts using the golden search algorithm in the initial iterations and then uses a parabolic algorithm to reach the tolerance setting.

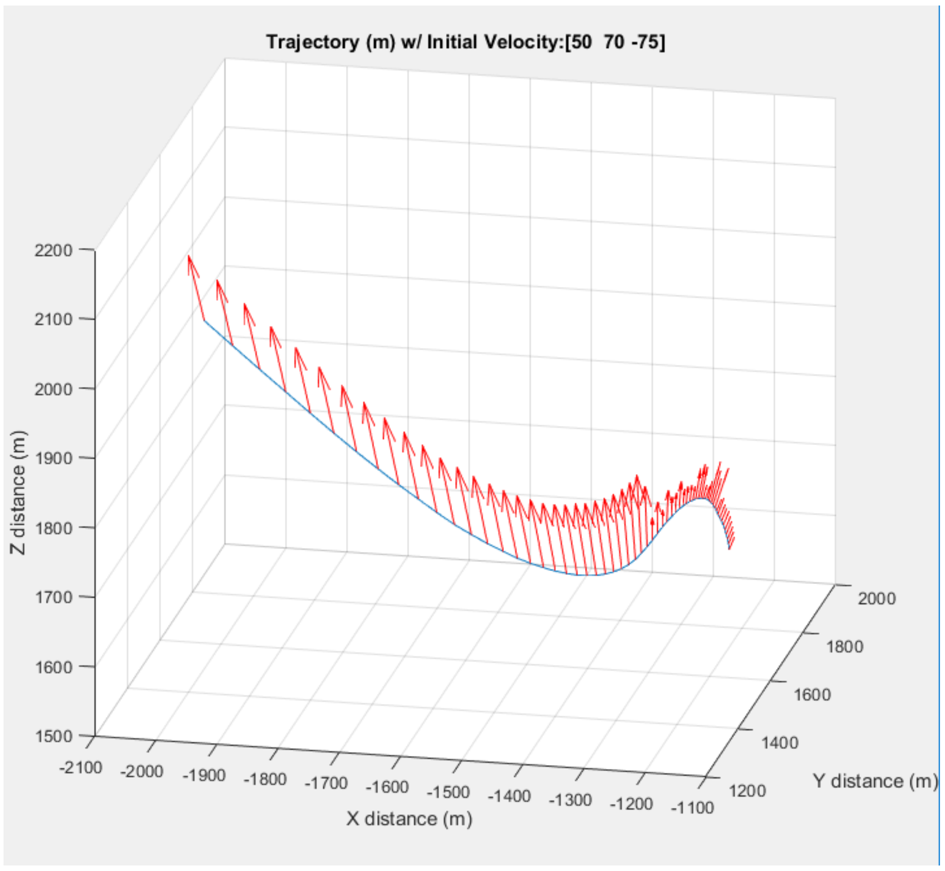


Fig. 10 This is the unfinished 3D trajectory generated by the successive algorithm and noise initial conitions.

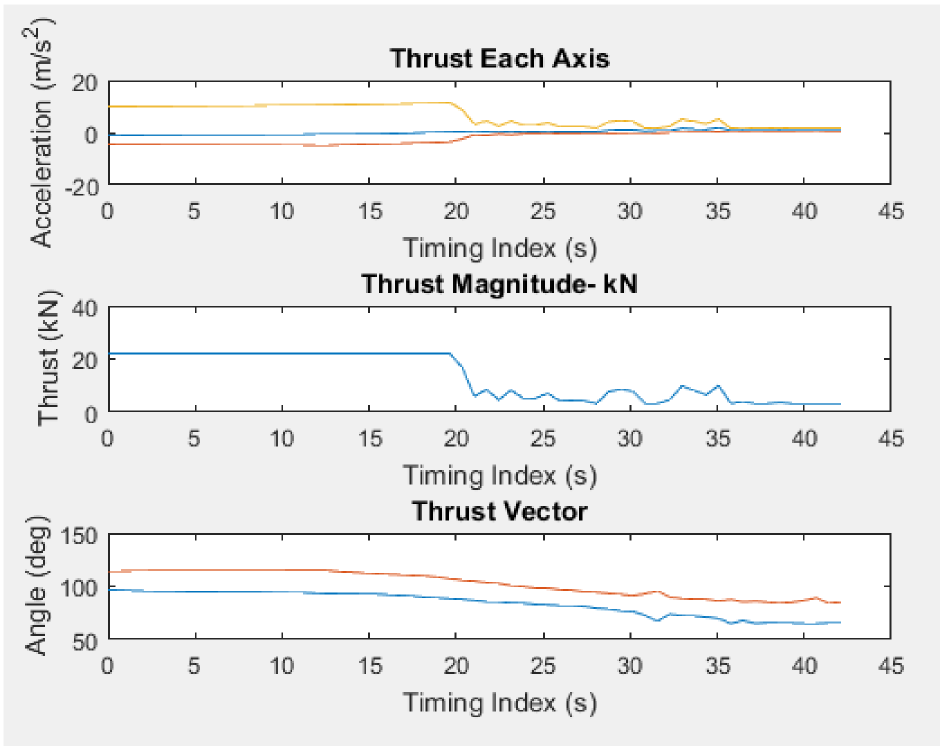


Fig. 11 These are the control inputs from the unfinished 3D trajectory generated by the successive algorithm

References

[1] 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.

[2] 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.

[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., Carson, J., and Blackmore, L., “Lossless Convexification of Nonconvex Control Bound Constraints of the Soft Landing Optimal Control Problem,” *IEEE Transactions on Control Systems Technology*, Vol. 21, No. 6, 2013, p. 2104.

[5] Stephen Boyd, L. V., *Convex Optimization*, 2009.