An Investigation of Optimal Powered Descent

Peter L. Brodkin

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics and Astronautics

University of Washington
2021

Committee:
Mehran Mesbahi
Justin Little

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

An Investigation of Optimal Powered Descent

Peter L. Brodkin

Chair of the Supervisory Committee:
Mehran Mesbahi
Aeronautics and Astronautics

Achieving fuel-optimal pinpoint landings is a vital component of many missions. In this paper, the foundational components of optimal control theory known as Pontryagin's Maximum Principle are derived, and an example is provided. The fuel-optimal trajectory for a one-dimensional lunar landing is then presented. The problem is then formulated in three-dimensions as a convex optimization problem. The main issue with this formulation is dealing with a non-convex constraint on the thrust, due to a non-zero lower bound on the thrust. However, the constraint can be made to be convex through the use of a slack variable. Some results for a simulated landing on Mars are presented. Finally, a problem formulation using the so-called indirect method is shown. Principles of optimal control are applied, and a system of equations including the state variables and Hamiltonian is derived. Achieving convergence for the root finding algorithm is difficult due to sensitivities to the initial guess and numerical scaling.

Contents

Chapter 1

Introduction

1.1 Problem Definition

Achieving soft, pinpoint landings on planets, moons, asteroids and comets is a crucial aspect of successful missions. Soft landings require a powered descent as many celestial objects, such as the moon, don't have an appreciable atmosphere and therefore parachutes and other aerodynamic breaking techniques won't work. Some celestial bodies, such as Mars, have enough atmosphere for a parachute to be useful for part of the landing process. However, depending on how fragile the payload is, or when there are humans on board, a soft landing is often required. Powered descents require expensive and heavy fuel. Using as little fuel as possible allows for a larger payload and a reduction in cost. Therefore, landing trajectories that use the least amount of fuel are often desired.

The choice of landing site is a combination of maximizing scientific discovery and the chances for a successful landing. Landings need to be as precise as possible for a number of reasons. For example, we may want a future spacecraft to land as closely as possible to other equipment that is already there. Moreover, the most interesting locations are often those that are geologically diverse and involve rough terrain [26]. In particular, canyons, large pits and caves are often scientifically significant but clearly require a higher degree of precision in order to avoid crashing. Lunar and Martian pits also experience less radiation, have milder temperatures and little to no meteorite activity making them desirable places for permanent equipment or human habitation. [10] Furthermore, those types of locations are often ideal spots to search for signs of extraterrestrial life. [5]

Due to the great distances involved when traveling through space, the spacecraft must be able to carry out the landing entirely on its own. Even with a lunar landing, it would take several seconds to get a message back and forth to a lander, which is too long during crucial moments. The goal of this research is to analyze the final stage of powered descent in order to find a fuel optimal trajectory that can be computed quickly and autonomously on board the spacecraft.

1.2 Approaches to the Problem

1.2.1 Apollo Guidance

From 1969 to 1972 the Apollo program landed the only 12 people to have ever set foot on the moon. Apollo powered descent and guidance was relatively simple, extremely innovative and highly successful. The main idea of Apollo guidance was to get a lander from one point to another from a given set of initial and final conditions. In order to work within the computing limits of the time, principles of optimal control were not applied. Instead, the focus was on finding a feasible solution for the trajectory. [26] First a quadratic acceleration function of time was defined. A quadratic was used because that is what is required if there are three target states that we want to specify. [12] Therefore the acceleration can be written as

$$a(t) = C_0 + C_1 t + C_2 t^2 \tag{1.1}$$

Where C_0, C_1 and C_2 are 3-dimensional constants. Integrating gives

$$v(t) = C_0 t + \frac{1}{2} C_1 t^2 + \frac{1}{3} C_2 t^3 + v_0 \tag{1.2}$$

$$r(t) = \frac{1}{2} C_0 t^2 + \frac{1}{6} C_1 t^3 + \frac{1}{12} C_2 t^4 + v_0 t + r_0 \tag{1.3}$$

Putting the system of equations into matrix form and solving for the coefficients at the final time gives

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1 & t_f & t_f^2 \\ t_f & \frac{1}{2} t_f^2 & \frac{1}{3} t_f^3 \\ \frac{1}{2} t_f^2 & \frac{1}{6} t_f^3 & \frac{1}{12} t_f^4 \end{bmatrix}^{-1} \begin{bmatrix} a_f \\ v_f - v_0 \\ r_f - v_0 t_f - r_0 \end{bmatrix} \tag{1.4}$$

To get closed form solutions, an inspection of (1.4) reveals that the final position, velocity, acceleration and time, as well as the initial position and velocity must be fixed. The initial acceleration, however, is free provided that the acceleration can change instantaneously. It is important to note that there were no

constraints on the states. For example, there is no guarantee that the solution won't result in a trajectory that goes below the surface or an acceleration that exceeds the maximum acceleration that the thrusters can provide. Therefore, it was essential that the initial conditions be within a range that allowed for a soft, physically possible landing. [16] In addition, the solutions for the constants are inversely proportional to different powers of t_f. To avoid a singularity, the update to the acceleration was stopped and a minimum time was used in place of values that were too close to zero. [35]

A main drawback of Apollo guidance is the fixed final time. Not only is fixing the final time not fuel-optimal but it may not be practical when, for unforeseen reasons, the initial conditions end up being significantly different than expected or the landing spot needs to be adjusted. However, with the computing speeds of today, the optimal final time can be found quite quickly via a line search method for the given objective function

$$J = \int_t^{t_f} ||a_T|| d\tau \tag{1.5}$$

where a_T is the acceleration thrust vector. [16] The optimal final time can then be used in (1.4) which can yield much better results without adding any significant complexity.

1.2.2 Missions to Mars

Over the last 25 years or so, there has been increasing interest in landing on Mars, in no small part due to all of the excitement surrounding the success of the various Mars rovers. Five rovers have been successfully landed on Mars beginning with Sojourner in 1997 and culminating with Perseverance just a few months ago, in February of 2021. In addition, the Curiosity rover that landed in 2012 is still operational as of May 12, 2021. The rovers have all made important scientific discoveries such as early evidence of rivers and lakes on Mars, as well as various organic compounds that are building blocks for life. [24]

Landing on Mars consists of several steps called Entry, Descent and Landing (EDL). Upon entering the Martian atmosphere, the spacecraft will heat up considerably, but the payload is protected by an aeroshell that helps keep important components at safe temperatures. The atmosphere also slows down the spacecraft considerably. Once the desired velocity is reached, a parachute deploys that further slows the spacecraft. The Perseverance rover that just recently landed on Mars in February of 2021, was the first to utilize a Range Trigger which helped to open the parachute at the optimal time. Previous missions deployed the parachute

as soon as a specific velocity was reached whereas the Range Trigger deployed the parachute based on the spacecraft's velocity and location, allowing for a more precise landing. During the parachute descent, the heat shield is dropped and the payload continues to slow down. However, due to the thin atmosphere on Mars, a parachute is not sufficient to slow the payload down enough to enable a soft landing. Therefore, a final powered descent phase is required to safely land the payload. [24] Not surprisingly, the landing ellipse, which is the region in which the payload is expected to land, has shrunk considerably over the last 25 years. For Pathfinder, which landed in 1997, the ellipse was about 200 by 70 kilometers. In contrast, Perseverance had a landing ellipse of about 8 by 7 kilometers. [25]

In this paper, the focus is on the final powered descent phase of the landing. In addition to the modified Apollo guidance which was used to help land the Mars Science Lab that carried Curiosity in 2012, there are two main approaches to the fuel-optimal pinpoint landing problem. The first method, sometimes referred to as the direct method, involves discretizing an optimization problem and then applying a nonlinear programming technique such as convex optimization (see chapter 4). Possible solutions are found using algorithms that identify states that satisfy the Karush-Kuhn-Tucker (KKT) conditions. These methods don't require a determination of the necessary conditions for optimality. The second method, referred to as the indirect method, involves applying the principles of optimality as given by Pontryagin's Maximum Principle [27]. The problem can then be formulated as two point boundary problem and a numerical algorithm is used to find a solution.

The direct method using convex optimization is advantageous because there are robust, efficient algorithms that solve convex optimization problems in a relatively small number of iterations, even when there are hundreds of variables and constraints. On the other hand, the indirect method can provide a pinpoint landing that is more accurate by an order of magnitude [17]. However, the indirect method suffers from the fact that convergence is not guaranteed, and the required root-finding algorithms are highly susceptible to the initial guess (See Chapter 5). Further complicating the matter is that the initial guess will likely include the initial costates. Since costates are not physical quantities, it can be difficult to know what a good initial guess might be. In addition, the problem formulation can be more complicated as to derive the necessary equations, a command of optimal control theory is required. Ultimately, these two methods are both important as they are both suitable for different types of applications.

1.3 Algorithms

The fundamental advantage to the convex optimization approach is the existence of numerous fast, robust and reliable algorithms that can guarantee a solution. G-FOLD is very robust in that it can handle any retargeting range, if required. In addition, the algorithm will provide the fuel-optimal solution and is numerically stable. What makes the algorithm particularly innovative is that it takes a non-convex constraint that appears due to a lower bound on the thrust magnitude and converts it to a convex constraint through the use of a slack variable (see Chapter 4). Therefore, all of the advantages of convex optimization, such as local optimal solutions also being global optimal solutions, can be utilized.

The solver used in this paper is CVX which solves convex programs and is implemented in Matlab. The main problem with CVX is that it's slow compared to other solvers. However, for problems that are relatively small and avoid large loops, CVX works well. [11]

For the indirect method (see Chapter 5), principles of optimal control are applied which leads to a seven-equation, seven-unknown root-finding problem. One issue with root-finding algorithms is that they can be numerically unreliable and not converge to a solution. Dr. Lu, and others in [19] have determined that numerical scaling is essential, in addition to using the right type of algorithm. In particular, Powell's dog leg method is used in [18] and in chapter 5. However, it is highly susceptible to the initial guess and the problem must be formulated in a numerically friendly fashion.

Chapter 2

Optimal Control Theory

2.1 Introduction

In this chapter, building off of the work of David Luenberger [20], the necessary conditions for an optimal trajectory are presented. At the heart of all of this optimal control theory is Pontryagin's Maximum Principle. Lev Pontryagin's "The Mathematical Theory of Optimal Processes," [27] contains detailed and rigorous proofs of the findings shown here. The goal here is to provide a more accessible derivation of some of the main findings of the Maximum Principle without duplicating the proofs. The principles are then applied to a problem with terminal constraints on the state and a free final time. Finally, a simple example is provided of how to derive an optimal trajectory that demonstrates the desired thrust profile.

2.2 Conditions for Optimality

A general form for a dynamic system can be written as

$$\dot{x} = f\big(x(t), u(t)\big) \tag{2.1}$$

with an initial condition $x(0) = x_0$ and a valid control $u(t)$, with an objective function

$$J = \psi\big(x(t_f)\big) + \int_0^{t_f} L\big(x(t), u(t)\big)dt \tag{2.2}$$

Note that there are two parts of the objective function. The first part is sometimes called the terminal cost, and represents how some aspect of the final state affects J. So if the goal is to maximize or minimize

some aspect of the final state, such as final velocity, that can be incorporated into this part of the objective function. The second part, often called the integral cost, represents how something that builds up over time affects the cost function. This part of the objective function is important for minimizing the amount of fuel used, for example. A given set of equations for the dynamics, along with the initial state and control function define a unique trajectory. For optimal control problems, the goal is to find the control function that results in maximizing the objective function.

If $u(t)$ is optimal, then any change in u will result in a decrease in the objective function. In general, to determine if a certain control function is optimal, we can change u a little bit and see what happens to J. Of course, J also depends on x and changing u also changes x. To start to find a way to determine how changes in u affect J, first rewrite J as

$$\bar{J} = J - \int_0^{t_f} \lambda(t)^T [\dot{x}(t) - f\big(x(t), u(t)\big)] dt \tag{2.3}$$

Using (2.1), we can see that for any $\lambda(t)$, $\bar{J} = J$. This will prove very useful shortly, as $\lambda(t)$ can be chosen in such a way as to vastly simplify the problem. The Hamiltonian can then be defined as

$$\mathcal{H}(\lambda, x, u) = \lambda^T f(x, u) + L(x, u) \tag{2.4}$$

Then with some basic substitution using (2.2) and (2.4), (2.3) can be written as

$$\bar{J} = \psi\big(x(t_f)\big) + \int_0^{t_f} \mathcal{H}(\lambda, x, u) - \lambda(t)^T \dot{x}(t) dt \tag{2.5}$$

Now assume that u is changed in some small way. Let the new trajectory be represented as $x(t) + \delta x(t)$. That leads to a change in the objective function, represented as

$$\delta\bar{J} = \psi\big(x(t_f) + \delta x(t_f)\big) - \psi\big(x(t_f)\big) + \int_0^{t_f} \big[\mathcal{H}(\lambda, x + \delta x, u + \delta u) - \mathcal{H}(\lambda, x, u) - \lambda(t)^T \delta\dot{x}\big] dt \tag{2.6}$$

The third term in the integrand can be integrated by parts. In addition, a first order approximation of the form

$$\frac{\mathcal{H}(\lambda, x + \delta x, u) - \mathcal{H}(\lambda, x, u)}{\delta x} = \mathcal{H}_x$$

can be used to simplify (2.6). After a fair amount of tedious algebra, (2.6) can be written as

$$\begin{aligned}\delta\bar{J} = &\left[\psi_x\big(x(t_f)\big) - \lambda(t_f)^T\right]\delta x(t_f) + \lambda(0)^T\delta x(0) \\ &+ \int_0^{t_f}\left[\mathcal{H}_x(\lambda, x, u) + \dot{\lambda}^T\right]\delta x dt \\ &+ \int_0^{t_f}\left[\mathcal{H}(\lambda, x, u+\delta u) - \mathcal{H}(\lambda, x, u)\right]dt\end{aligned} \tag{2.7}$$

where higher terms are ignored. First note that $\delta x(0) = 0$ since changing the control won't change the initial conditions. So that eliminates one term. To simplify further, we can take full advantage of the fact that $\lambda(t)$ can be anything. The first integral can be made to go to zero by requiring that

$$-\dot{\lambda} = \mathcal{H}_x \tag{2.8}$$

In addition, the first term in (2.7) can be eliminated by requiring a terminal condition on (2.8):

$$\lambda(t_f)^T = \psi_x\big(x(t_f)\big) \tag{2.9}$$

With this definition of $\lambda(t)$, all terms except for the last integral in (2.7) go to zero. So δJ can be written as

$$\delta\bar{J} = \int_0^{t_f}\left[\mathcal{H}(\lambda, x, u+\delta u) - \mathcal{H}(\lambda, x, u)\right]dt \tag{2.10}$$

If u is optimal, then $\delta\bar{J}$ must be negative at all times. Using (2.10) then gives

$$\mathcal{H}(\lambda, x, u+\delta u) \leq \mathcal{H}(\lambda, x, u) \tag{2.11}$$

which leads to the very important conclusion that the optimal control is the control that maximizes the Hamiltonian.

2.3 Terminal Constraints

The analysis in the previous section assumed that there were no restrictions on the final state. However, oftentimes there is a desire to restrict the final state in some way. For example, in the case of landing a spacecraft, we may want the final position and velocity to be zero in order to achieve a soft, pinpoint landing. A further inspection of (2.7) shows the effect that these constraints have on the problem. If a certain final state, $x_i(t_f)$ is constrained, then $\delta x_i(t_f) = 0$. If this is case, then the first term in (2.7) shows

that $\lambda_i(t_f)$ can be anything. If $x_i(t_f)$ is not constrained, then $\delta x_i(t_f)$ doesn't have to be zero. In that case $\lambda_i(t_f) = \psi_x\big(x(t_f)\big)_i$.

To summarize:

$$\begin{aligned} x_i \text{ constrained: } & \lambda_i(t_f) \text{ is free} \\ x_i \text{ free: } & \lambda_i(t_f) = \psi_x\big(x(t_f)\big)_i \end{aligned} \tag{2.12}$$

To account for the fact that the terminal conditions may result in no solution to the problem, the Hamiltonian from (2.4) is rewritten as

$$\mathcal{H}(\lambda, x, u) = \lambda^T f(x, u) + \lambda_0 L(x, u) \tag{2.13}$$

where $\lambda_0 \geq 0$. If the problem is well formulated, we can take $\lambda_0 = 1$ and the Hamiltonian reduces to (2.4). However, in the case of no solution, $\lambda_0 = 0$.

Very often, the final time can be kept free, although for long journeys that involve human passengers, the final time might need to be constrained. However, as is done several times in this paper, there is no reason to fix the final time if all that is desired is to optimize fuel consumption. To derive the conditions for the free final time problem, the same method as the previous section is used. Using (2.5) and (2.13), the objective function can be written as

$$\bar{J} = \psi\big(x(t_t)\big) + \int_0^{t_f} \big[L(x, u) + \lambda^T f(x, u) - \dot{\lambda}^T x\big] dt \tag{2.14}$$

As before, there is a new control $v(t) = u(t) + \delta u(t)$, an associated new trajectory $x(t) + \delta x(t)$, and a new final time $t_f + \delta t_f$. The new value of the final state is now $x(t_f) + \delta x(t_f + dt_f)$ since the final time itself also changes. Much as before, with a fair amount of simplifying and using first order approximations, we can write

$$\begin{aligned} \delta\bar{J} = {} & \psi_x\big(x(t_f)\big) dx(t_f) - \lambda(t_f)^T \delta x(t_f) + \lambda(0)^T \delta x(0) + L\big(x(t_f), u(t_f)\big)\delta T \\ & + \int_0^{t_f} \big[\mathcal{H}_x(\lambda, x, u) + \dot{\lambda}^T\big] \delta x dt \\ & + \int_0^{t_f} \big[\mathcal{H}(\lambda, x, v) - \mathcal{H}(\lambda, x, u)\big] dt \end{aligned} \tag{2.15}$$

We can apply similar reasoning as before and determine that $\delta x(0) = 0$ and that the first integral goes to zero by (2.8). In addition, using a first order approximation for $\delta x(t_f + \delta t_f)$, applying (2.12) and doing some

substitution gives

$$\delta \bar{J} = \mathcal{H}(t_f) + \int_0^{t_f} \big[\mathcal{H}(\lambda, x, v) - \mathcal{H}(\lambda, x, u)\big] dt \tag{2.16}$$

The integral part is the same as before in (2.10). But here, we can also say that

$$\mathcal{H}(t_f) = 0 \tag{2.17}$$

Otherwise there could be some final time that would tend make $\delta J > 0$ and therefore the control would be suboptimal. As shown by Pontryagin [27] in a lengthy and rigorous proof, the Hamiltonian actually will be zero everywhere along an optimal trajectory as long as it doesn't depend explicitly on time and the final time is free. If the final time is fixed and there is no explicit time dependence, the Hamiltonian is equal to a constant.

To summarize, there are four main necessary conditions for optimal control:

$$-\dot{\lambda} = \mathcal{H}_x \tag{2.18}$$

$$\lambda(t_f)^T = \psi_x\big(x(t_f)\big) \tag{2.19}$$

$$\mathcal{H}(\lambda, x, u + \delta u) \leq \mathcal{H}(\lambda, x, u) \quad \text{for an optimal control } u \tag{2.20}$$

$$\mathcal{H}(t_f) = 0 \qquad \text{if the final time is free} \tag{2.21}$$

2.4 Thrust Profile

Now the principles from the previous section can be applied to the minimum fuel problem. In problems such as this, the dynamics often take a form similar to

$$\dot{x}_1 = x_2 \tag{2.22}$$

$$\dot{x}_2 = u + a \tag{2.23}$$

where x_1 is the position, x_2 is the velocity, u is the control that could equal specific thrust, and a is a constant often equal to gravitational acceleration. The goal for this example is to minimize the fuel burned for a trajectory from some starting point to the origin and it will be shown in section 3.2 that this is equivalent to minimizing the final time. Therefore, the objective function can be written using (2.2) as

$$J = t_f - t_i \tag{2.24}$$

Using (2.4), the Hamiltonian can be written as

$$\mathcal{H} = \lambda_1 x_2 + \lambda_2 (u + a) \tag{2.25}$$

Applying (2.18) gives

$$\dot{\lambda}_1 = 0 \qquad \dot{\lambda}_2 = -\lambda_1 \tag{2.26}$$

$$\lambda_1 = c_1 \qquad \lambda_2 = -c_1 t + c_2 \tag{2.27}$$

where c_1 and c_2 are constants. Because it makes good sense to have an upper and lower bound on the thrust, $u_{\min} \leq u \leq u_{\max}$. By applying (2.20),

$$u = \begin{cases} u_{\min} & \text{if} \quad \lambda_2 < 0 \\ u_{\max} & \text{if} \quad \lambda_2 > 0 \end{cases} \tag{2.28}$$

Therefore, the optimal control is always equal to the maximum thrust or the minimum thrust. This important result holds when the Hamiltonian has a linear dependence on u, as is this case in these types of optimization problems. If the coefficient of u in the Hamiltonian is positive, then $u_{\max}$ optimizes the Hamiltonian and is therefore optimal. Similarly, $u_{\min}$ maximizes the Hamiltonian and is optimal if the coefficient is negative. Therefore, the coefficient of u has special significance and is referred to as the switching function, as when this function is equal to zero, it indicates that a switch between maximum and minimum thrust occurs. Here the switching function is λ_2. If the switching function equals zero, then u cannot be determined. This is called the singularity condition and can sometimes present a problem. It will be shown in future chapters, however, that for the approaches in this paper, the singularity condition cannot hold over any finite time interval. Furthermore, by the second equation in (2.27), λ_2 can only equal zero at most once on any time interval. Therefore, in this example there is at most one switch for the optimal trajectory.

For simplicity, let $u \in [-1, 1]$ and $a = 0$. Then plugging in the maximum and minimum value for u and integrating $\dot{x}_1$ and $\dot{x}_2$ gives

$$x_1 = \pm \frac{1}{2} t^2 + c_1 t + c_2 \tag{2.29}$$

$$x_2 = \pm t + c_1 \tag{2.30}$$

where c_1 and c_2 are integration constants and the $\pm$ is for max/min u. Eliminating t results in

$$x_1 = \pm\frac{1}{2}x_2^2 + C \tag{2.31}$$

where C is some other constant. Figure 2.1 represents two possible optimal trajectories. In the case where the initial conditions dictate that the trajectory starts above the red curve, the initial control will be at a minimum until the trajectory hits the red curve. At that point, the control will switch to a maximum value until the endpoint of the origin is reached. Similarly, if the trajectory starts below the red curve, the control will be maximized until it hits the red curve, at which point the control will be minimized until the origin is reached. The red curve here represents the switching function.

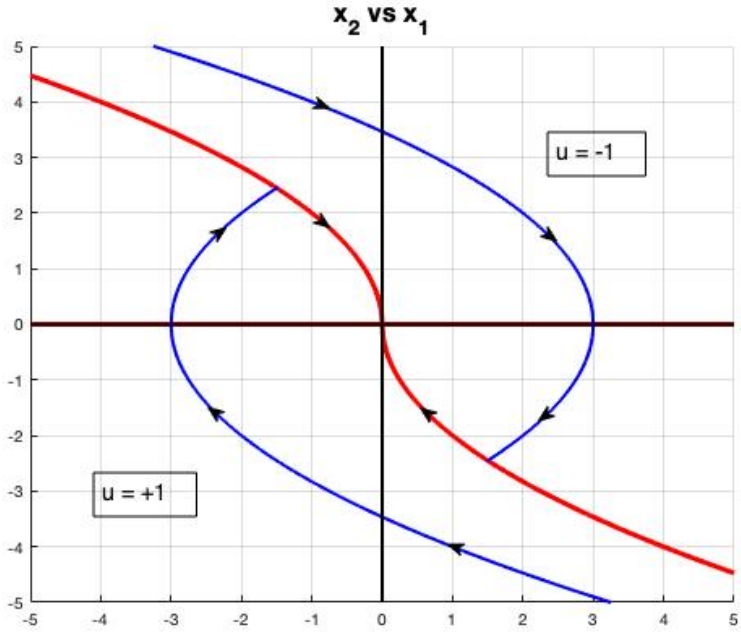

Figure 2.1: Optimal Trajectory

Formal proofs of this bang-bang control are given in [27] and are expanded to multiple dimensions. While the proofs won't be duplicated here, some important findings are worth summarizing. First, it can be dangerous to assume that an optimal control exists and there are certainly some systems for which that is the case (consider $\dot{x} = u$ and we want to get x from 0 to 1 in minimal time) . However, if the problem is well formulated, then u can be assumed to exist. Mathematically, it turns out that if the dynamics are given in the general form $\dot{x} = f(t, x, u)$ and we assume u exists and that $f(t, x, u)$ is compact and convex for all (t, x) pairs along the trajectory and for all u, then the reachable set of points from x_0 is compact. This is known

as Filippov's Theorem [15]. Around the same time, Pontryagin developed a similar theorem for a linear time optimal process. If the set of allowable controls is contained in a parallelepiped and the eigenvalues of the state matrix are real (as is the case for these types of optimization problems), then there is a unique optimal control function that will be piecewise continuous and always have either maximum or minimum value. In addition, the theorem states that there will be at most $n-1$ switchings where n is the number of states. Moreover, as proven in [14], there can be at most three thrust arcs. We can also say that for a linear system, as long as the set of allowable controls is compact and convex, then there is a time optimal control [15].

To summarize, the most important results for thrust profiles are:

1. An optimal control exists if the problem is well formulated.
2. Control is bang-bang, meaning it is always at the maximum or minimum allowable value.
3. Most optimal thrust profiles will follow a max-min-max pattern.

Chapter 3

Vertical Fuel-Optimal Descent

3.1 Introduction

In this chapter, the problem of a fuel-optimal trajectory for a small lander in the terminal descent phase is analyzed. This work was presented originally by Dr. James Meditch in a paper from 1964 entitled "On the Problem of Optimal Thrust Programming for a Lunar Soft Landing." [22] First, the problem of a fuel-optimal, vertical descent is shown to be equivalent to the problem of a time-optimal descent. Then, using principles of optimal control and Pontryagin's Maximum Principle, a switching function is derived that determines all of the states of altitude and velocity from which a soft landing can be achieved. Building on the concepts developed in chapter 2, the optimal thrust profile is applied and it can be seen that the optimal trajectory consists of a period of free fall followed by full thrust until the spacecraft lands. Finally, a simulation is presented using matlab for a fuel-optimal terminal descent of a small lunar lander.

3.2 Problem Formulation

The dynamics of the problem can be represented as

$$\ddot{x} = -\frac{k\dot{m}}{m} - g \tag{3.1}$$

where $k > 0$ is the velocity of the exhaust gas with respect to the spacecraft and $\dot{m} \leq 0$ is the rate at which fuel is consumed. To achieve a soft, pinpoint landing, both the final position and velocity are zero, so $x(\tau) = 0$ and $\dot{x}(\tau) = 0$. Here, τ represents the final time, which is kept free for reasons that will soon be clear. In this example, a trajectory that minimizes fuel consumption is considered. Therefore, the cost

function can be written as

$$J = -\int_0^\tau \dot{m}(t)dt = m(0) - m(\tau) \tag{3.2}$$

since minimizing the change in mass of the spacecraft is equivalent to minimizing the amount of fuel burned. To apply principles of time optimal control, we can show that the minimum fuel problem is equivalent to the minimum time problem. To see this, first note that

$$\frac{\dot{m}}{m} = \frac{d}{dt}\ln m.$$

Then, (3.1) can be rewritten as

$$\ddot{x} = -k\frac{d}{dt}\ln m - g. \tag{3.3}$$

Integrating both sides from 0 to t gives

$$\dot{x}(t) - \dot{x}(0) = -k(\ln m(t) - \ln m(0)) - gt$$

$$\dot{x}(t) = -k\ln\frac{m(t)}{m(0)} - gt + \dot{x}(0). \tag{3.4}$$

Remembering that the velocity at the final time, τ, is zero gives

$$0 = -k\ln\frac{m(\tau)}{m(0)} - g\tau + \dot{x}(0)$$

$$k\ln\frac{m(\tau)}{m(0)} = -g\tau + \dot{x}(0)$$

$$m(\tau) = m(0)\exp\left[\frac{-g\tau + \dot{x}(0)}{k}\right] \tag{3.5}$$

Now we can substitute (3.5) into (3.2) to get

$$J = m(0)\left[1 - \exp\left(\frac{-g\tau + \dot{x}(0)}{k}\right)\right] \tag{3.6}$$

For obvious physical reasons, $m(0) > 0$ and, as stated previously, $k > 0$. $\dot{x}(0)$ has no restrictions other than the assumption that for a given initial velocity, the spacecraft can generate enough thrust to perform a soft landing. With these constraints, J will always increase as τ increases, so minimizing τ will minimize J as well. Therefore, we can state that minimizing the final time is equivalent to minimizing the amount of fuel consumed. Now it is clear why the final time was kept free, as this important result allows for the use of principles of time optimal control.

3.3 Optimal Control

The states of the system can be defined as $x_1 =$ altitude, $x_2 =$ velocity and $x_3 =$ mass, so

$$\dot{x}_1 = x_2 \qquad \dot{x}_2 = \frac{-k}{x_3}u - g \qquad \dot{x}_3 = u. \tag{3.7}$$

The control variable, u, is equal to $\dot{m}$ which is directly related to the thrust of the spacecraft. The Hamiltonian for optimal control is given by

$$\mathcal{H} = \lambda_1 x_2 + \lambda_2 \left(-\frac{k}{x_3}u - g \right) + \lambda_3 u \tag{3.8}$$

where λ_i represents the costates. Using the fact that $\dot{\lambda}_i = -\frac{\partial \mathcal{H}}{\partial x_i}$ gives

$$\dot{\lambda}_1 = 0 \qquad \dot{\lambda}_2 = -\lambda_1 \qquad \dot{\lambda}_3 = -\lambda_2 \frac{k}{x_3^2}u. \tag{3.9}$$

So λ_i for $i = 1, 2, 3$ are the solutions to this set of differential equations.

Applying Pontryagin's Maximum Principle (see chapter 2), the control that maximizes the Hamiltonian is optimal. Given that there's a limit on how much fuel can be burned at once and that $\dot{m} \leq 0$, upper and lower limits on u can be assigned such that $-\alpha \leq u \leq 0$ where α is a constant related to the maximum possible thrust. Regrouping terms in the Hamiltonian (3.8) gives

$$\mathcal{H} = \lambda_1 x_2 - \lambda_2 g + \left(-\frac{k}{x_3}\lambda_2 + \lambda_3 \right) u.$$

Since $u \leq 0$, $\mathcal{H}$ is maxed as follows:

$$u = \begin{cases} -\alpha & \text{if} \quad -\frac{k}{x_3}\lambda_2 + \lambda_3 < 0 \\ 0 & \text{if} \quad -\frac{k}{x_3}\lambda_2 + \lambda_3 > 0 \end{cases} \tag{3.10}$$

Note that if

$$\frac{\partial \mathcal{H}}{\partial u} = -\frac{k}{x_3}\lambda_2 + \lambda_3 = 0$$

then u cannot be determined. This is the singularity condition and it is shown in the next section that this condition can't hold on a finite time interval. In addition, a proof given in [22] shows that the optimal trajectory consists of either full thrust for the whole descent, or a period of free fall followed by a period of full thrust until landing. The problem now becomes finding the switching function which gives us the state

at which the thrust should begin firing.

3.4 Singularity Condition

Some types of optimization problems, such as the Goddard rocket problem of trying to obtain the highest altitude during an ascent, may require singular controls. This can present a problem as the Maximum Principle doesn't provide any information on what the thrust should be during a singular control arc. For the problem presented here, however, we can show that the singularity condition only can exist instantaneously. First note that the Hamiltonian given in (3.8) does not depend explicitly on time and that the final time is kept free. Therefore, by the Maximum Principle,

$$\mathcal{H} = \lambda_1 x_1 - \lambda_2 \frac{k}{x_3} u - \lambda_2 g + \lambda_3 u \geq 0 \tag{3.11}$$

The singularity condition is

$$\lambda_3 - \lambda_2 \frac{k}{x_3} = 0 \tag{3.12}$$

and plugging that into (3.11) gives

$$\lambda_1 x_2 - \lambda_2 g = \text{constant} \tag{3.13}$$

Differentiating (3.19) with respect to time gives

$$\dot{\lambda}_1 x_2 + \lambda_1 \dot{x}_2 - \dot{\lambda}_2 g = 0 \tag{3.14}$$

Using (3.7) and (3.9) and substituting gives

$$\begin{aligned} \lambda_1 \Big(- \frac{k}{x_3} u - g \Big) - \lambda_1 g &= 0 \\ \lambda_1 \frac{k}{x_3} u &= 0 \end{aligned} \tag{3.15}$$

Since k and x_3 are both greater than zero,

$$\lambda_1 u = 0 \tag{3.16}$$

Using (3.9), $\lambda_1 = \text{constant}$ so $u(t) = 0$ if $\lambda_1 \neq 0$. So either $\lambda_1 = 0$ or $u = 0$ during the singularity condition. First consider $\lambda_1 = 0$ for all t. From (3.9), this means that $\lambda_2 = \text{constant}$.

1. $\lambda_2 = 0$.

   If this is the case, then by (3.9), $\dot{\lambda}_3 = 0$. The transversality condition (2.12) gives $\lambda_3 = 0$ since the final

mass is kept free. So that means that all costates are equal to zero. The Maximum Principle states that there exists a non-zero solution, so this case can't happen.

2. $\lambda_2 < 0$

   By (3.9) $\dot{\lambda}_3 \leq 0$ since $u \leq 0$. But again by transversality, $\lambda_3 = 0$. This means that $\lambda_3 \geq 0$ since the function is decreasing or constant due to the derivative being negative or zero. However, since $\lambda_2 < 0$, $\frac{k}{x_3}\lambda_2 < 0$. So

   $$\lambda_3 - \frac{k}{x_3}\lambda_2 > 0$$

   since that is a positive number minus a negative number. This violates (3.12) and therefore can't happen.

3. $\lambda_2 > 0$

   By (3.9) $\dot{\lambda}_3 \geq 0$. Using the same argument as the previous case means that $\lambda_3 \leq 0$ and $\frac{k}{x_3}\lambda_2 > 0$. So

   $$\lambda_3 - \frac{k}{x_3}\lambda_2 < 0$$

   since that is a negative number minus a positive number which contradicts (3.12).

So the singularity condition cannot hold on a closed time interval for $\lambda_1 = 0$ Now consider the case where $u(t) = 0$. From (3.9), $\dot{\lambda}_3 = 0$ which means that $\lambda_3 =$ constant. Also, k is constant and by (3.7), $x_3 =$ constant. Therefore, by (3.12), $\lambda_2 =$ constant. But $\lambda_2 =$ constant only if $\dot{\lambda}_2 = -\lambda_1 = 0$. We already know that the singularity can't hold if $\lambda_1 = 0$ so it can't hold if $u = 0$ either. Finally, we can conclude that no singular optimal controls exist.

3.5 Derivation of Switching Function

First, let x_1^*, x_2^* and M_0 represent the initial altitude, velocity and mass respectively when thrusting begins. Also, let the time interval over which thrusting occurs be $[0, t_1]$. Furthermore, assuming that the mass loss is linear gives

$$x_3(t) = M_0 - \alpha t$$

Then plugging into equation (3.4) gives

$$x_2(t) = -k \ln \frac{M_0 - \alpha t}{M_0} - gt + x_2^*. \tag{3.17}$$

Using the fact that $\dot{x}_1 = x_2$, using (3.17) and integrating both sides gives

$$x_1(t) - x_1^* = \int_0^t x_2(s)ds$$

$$x_1(t) = \left(\frac{kM_0}{\alpha}\right)\left(\frac{M_0 - \alpha t}{M_0}\right)\ln\left(\frac{M_0 - \alpha t}{M_0}\right) + kt - \frac{1}{2}gt^2 + x_2^* t + x_1^*. \tag{3.18}$$

Keeping in mind that a soft landing is desired, at $t = t_1$ we need $x_1(t_1) = 0$ and $x_2(t_1) = 0$. So (3.17) and (3.18) become

$$x_2(t_1) = -k\ln\frac{M_0 - \alpha t_1}{M_0} - gt_1 + x_2^* = 0$$

$$x_1(t_1) = \left(\frac{kM_0}{\alpha}\right)\left(\frac{M_0 - \alpha t_1}{M_0}\right)\ln\left(\frac{M_0 - \alpha t_1}{M_0}\right) + kt_1 - \frac{1}{2}gt_1^2 + x_2^* t + x_1^* = 0$$

Solving for x_2^* gives

$$x_2^* = k\ln\left(\frac{M_0 - \alpha t_1}{M_0}\right) + gt_1. \tag{3.19}$$

That result can then be plugged into the expression for $x_1(t_1)$ to obtain

$$x_1^* = -\frac{kM_0}{\alpha}\ln\left(\frac{M_0 - \alpha t_1}{M_0}\right) - kt_1 - \frac{1}{2}gt_1^2. \tag{3.20}$$

To get an analytic function in terms of x_1^* and x_2^*, t_1 needs to be eliminated. One way to do this is to use a Taylor expansion for the natural log term. The ratio $\frac{\alpha t_1}{M_0}$ represents the ratio of the mass of fuel consumed to the mass of the spacecraft. Certainly this is less than one, and depending on the circumstances, it may be very much less than one. Therefore the natural log term can be approximated using a second order Taylor expansion.

$$\ln\left(\frac{M_0 - \alpha t_1}{M_0}\right) \approx -\frac{\alpha}{M_0}t_1 - \frac{\alpha^2}{2M_0^2}t_1^2 \tag{3.21}$$

Using this result and plugging into (3.20) gives

$$x_1^* = -\frac{kM_0}{\alpha}\left(-\frac{\alpha}{M_0}t_1 - \frac{\alpha^2}{2M_0^2}t_1^2\right) - kt_1 - \frac{1}{2}gt_1^2$$

$$x_1^* = \left(\frac{\alpha k}{2M_0} - \frac{1}{2}g\right)t_1^2$$

$$x_1^* = at_1^2 \tag{3.22}$$

where

$$a = \frac{\alpha k}{2M_0} - \frac{1}{2}g$$

Similarly, plugging into (3.19) gives

$$x_2^* = k\left(-\frac{\alpha}{M_0}t_1 - \frac{\alpha^2}{2M_0^2}t_1^2 \right) + gt_1$$

$$x_2^* = \left(-\frac{\alpha k}{M_0} + g \right)t_1 - \frac{\alpha^2 k}{2M_0^2}t_1^2$$

$$x_2^* = 2at_1 - bt_1^2 \tag{3.23}$$

where

$$b = \frac{\alpha^2 k}{2M_0^2}$$

Solving (3.22) for t_1 and plugging that result into (3.23) gives

$$x_2^* = -2a\sqrt{\frac{x_1^*}{a}} - b\frac{x_1^*}{a}$$

Rearranging terms results in the switching function:

$$f(x_1^*, x_2^*) = x_2^* + 2a\sqrt{\frac{x_1^*}{a}} + \frac{b}{a}x_1^* = 0 \tag{3.24}$$

This switching function represents all of the initial sets of altitude and velocity for which a soft landing can be achieved while firing at full thrust.

3.6 Discussion

Figure 1.1 shows the velocity vs altitude graph for a vertical lunar descent of a small, 2 kg spacecraft from an initial height of 30 meters, an initial velocity of 1 m/s downwards, with a specific impulse of $30/g_{\text{earth}}$, a max thrust of 6.5 N, and value of $g = 1\ m/s^2$ for lunar gravitational acceleration. Note that the graph depicted is actually the fourth quadrant as $x(t) > 0$ and $\dot{x}(t) < 0$. Physically, this means that the spacecraft is always above the ground and is always moving in a downwards direction. The trajectory of the descent begins at the initial conditions, and the spacecraft will follow the free fall arc until it reaches an altitude of approximately 9 meters with a downward velocity of about 6.5 m/s. At that point, the thrust turns on and the spacecraft follows the arc of the switching function until a soft landing is achieved.

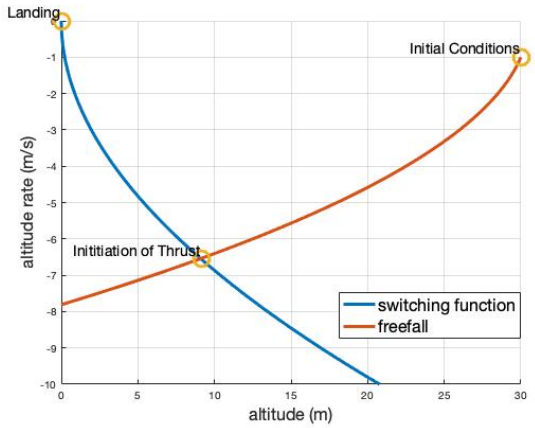

Figure 3.1: Velocity vs Altitude

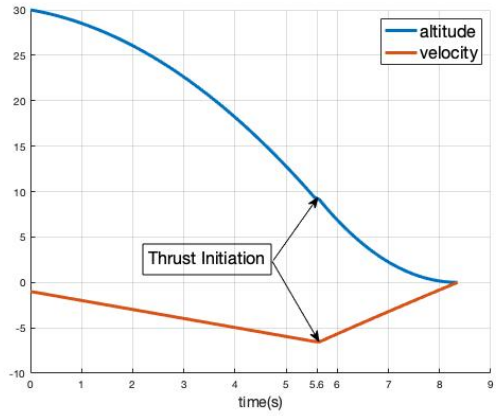

Figure 3.2: Altitude and Velocity vs Time

It is important to note that the free fall trajectory and the switching function intersect only once. That means that there is only one, unique switching time. Since there is only one possible time, that time must be optimal. If the initial conditions of the spacecraft are anywhere above the graph of the switching function, then the spacecraft should free fall until its current state intersects with the switching function, at which point the thrust should fire and remain on until landing. If the state of the spacecraft places it below the switching function, then the thrust of spacecraft is insufficient to produce a soft landing.

To present a more conventional view of the trajectory, figure 1.2 shows the altitude and velocity vs time for the spacecraft. The initiation of the thrust occurs at about 5.6 seconds after the initial free fall, and the total time of the landing takes about 8.3 seconds.

Given that every sensor has some sort of inherent noise, the actual trajectory and the measured trajectory will not align perfectly. To get a better picture of the error, the conditions under which the natural log in (3.21) can be approximated as indicated need to be determined. If the ratio of consumed fuel to initial mass is 0.25, then the error in the approximation of (3.21) is

$$\left| \frac{\ln 0.75 - (-0.25 - 0.25^2/2)}{\ln 0.75} \right| * 100 = 2.24\%.$$

Depending on the mission parameters, this may or may not be acceptable. But expecting that the fuel burned will be about 25% of the total mass of the spacecraft seems like a reasonable assumption for a number of missions.

Another source of error is evident by looking at (3.17). The velocity is dependent on the correct determination of the velocity at the initiation of thrusting. When calculating the altitude, (3.18) reveals that the altitude is dependent on the correct determination of the altitude at the initiation of thrusting. Moreover, when calculating the altitude, any error in the velocity when thrust is initiated will increase linearly with time. One possible solution to the inherent error is to fire a thrust that will maintain a constant descent velocity until landing. Once a velocity is reached at which the spacecraft can reasonably land, say 1 m/s, a thrust would fire that would land the spacecraft at that same velocity. While this would not be fuel optimal, it would help ensure the success of the mission. Regardless, the method presented in this chapter could at the very least be used as a guide to help determine the minimum amount of fuel required for the terminal descent phase of a lander.

Chapter 4

Convex Optimization Approach

4.1 Introduction

Having simulated the one dimensional case in the previous section, we now tackle the more complex three dimensional problem using papers by Ufuk Topcu [34] and Behçet Açikmeşe [2] as a guide. In addition, for practical reasons, the minimum thrust cannot be zero as spacecraft engines typically cannot be turned totally off. Therefore an additional constraint of a non-zero minimum thrust is considered. After formulating the problem and investigating the behavior of the costates, a convex optimization framework is developed using the work presented in [2] and a numerical solution is shown.

Stating the problem as a convex optimization problem has several advantages. First, there are a variety of efficient algorithms that can solve convex optimization problems much faster than non-convex problems. In addition, solutions to convex optimization problems have the desirable property that a local minimum is also a global minimum. Therefore, once an optimal solution is found, there is no need to ensure that it is globally optimal.

One of the main difficulties with framing fuel-optimal landings as a convex optimization problem is that the lower bound on the thrust results in a non-convex constraint. However, introducing a slack variable along with a change in variables as shown in [2] results in a convex constraint. In addition, a proof is offered that shows that a solution to the more relaxed optimization problem with the slack variable is also a solution to the original problem with the original thrust constraint. The problem is then discretized in order to construct the numerical solution.

4.2 Problem Formulation

First the problem can be set up as

$$\dot{m} = -kT \tag{4.1}$$

where k is the inverse of the exhaust velocity and is assumed to be constant, and T is magnitude of the thrust. In addition,

$$\dot{C} = \Lambda \tag{4.2}$$

where $\Lambda = T/m$ is the specific thrust and C is the characteristic velocity with an initial condition of $C(t_0) = 0$. For practical purposes, the minimum thrust cannot be zero, as the engine can never completely shut off. Therefore, bounds are placed on the thrust.

$$0 \leq \Lambda_{min}(t) \leq \Lambda(t) \leq \Lambda_{max}(t) \tag{4.3}$$

Taking the state vetor to be $x = [\vec{r} \quad \vec{V} \quad C]$, the dynamics are represented as

$$\dot{\vec{r}} = \vec{V} \qquad \dot{\vec{V}} = \vec{g} + \vec{\Lambda} \qquad \dot{C} = \Lambda \tag{4.4}$$

where $\vec{r}$ and $\vec{V}$ are position and velocity and $\vec{g}$ is the gravitational acceleration. Minimizing the fuel used means maximizing the final mass, which in turn means maximizing $-C(t_f)$. Therefore, the cost function can be written as

$$J = -C(t_f) \tag{4.5}$$

The Hamiltonian is then given by

$$\mathcal{H} = \vec{\lambda}_r \cdot \vec{V} + \vec{\lambda}_V \cdot (\vec{g} + \vec{\Lambda}) + \lambda_c \Lambda \tag{4.6}$$

By Pontryagin's Maximum Principle (see chapter 2), the control that optimizes the Hamiltonian is optimal. Therefore $\vec{\Lambda}$ will be in the same direction as $\vec{\lambda}_V$. Using that result and rearranging the Hamiltonian gives

$$\mathcal{H} = \vec{\lambda}_r \cdot \vec{V} + (\lambda_V + \lambda_C)\Lambda + \vec{\lambda}_V \cdot \vec{g}. \tag{4.7}$$

Maximizing the Hamiltonian gives

$$\Lambda^* = \begin{cases} \Lambda_{\min} & \text{if} \quad \lambda_V + \lambda_C < 0 \\ \Lambda_{\max} & \text{if} \quad \lambda_V + \lambda_C > 0 \end{cases}$$

where Λ^* is the optimal specific thrust.

$$\frac{\partial \mathcal{H}}{\partial \Lambda} = \lambda_V + \lambda_C \tag{4.8}$$

represents the switching function. The thrust will switch from maximum to minimum or from minimum to maximum when the switching function equals zero.

Now, using $\dot{\lambda} = -\frac{\partial \mathcal{H}}{\partial x}$ gives

$$\dot{\vec{\lambda}}_r = 0 \qquad \dot{\vec{\lambda}}_V = -\vec{\lambda}_r. \tag{4.9}$$

Also, using (4.1) and (4.2) gives $m(t) = m_0 e^{-kC}$ or alternatively, $C = -\frac{\ln m - \ln m_0}{k}$. Therefore

$$\begin{aligned} \dot{\lambda}_C &= -\frac{\partial}{\partial C}(\lambda_V + \lambda_C)\Lambda \\ &= -\frac{\partial}{\partial C}(\lambda_V + \lambda_C)\frac{T}{m_0 e^{-kC}} \\ &= -(\lambda_V + \lambda_C)\frac{T}{m_0} k e^{kc} \\ &= -(\lambda_V + \lambda_C)\frac{T}{m_0} k e^{-(\ln m - \ln m_0)} \\ &= -(\lambda_V + \lambda_C) k \Lambda \end{aligned} \tag{4.10}$$

Then by Pontryagin's Maximum Principle (2.12), because $C(t_f)$ is kept free,

$$\lambda_C(t_f) = \left.\frac{\partial J}{\partial C}\right|_{t=t_f} = -1. \tag{4.11}$$

Then, using the second part of (4.9) and integrating both sides gives

$$\begin{aligned} \int_t^{t_f} \dot{\vec{\lambda}}_V dt &= -\int_t^{t_f} \vec{\lambda}_r dt \\ \vec{\lambda}_V(t_f) - \vec{\lambda}_V(t) &= \vec{\lambda}_r(t - t_f) \\ \vec{\lambda}_V(t) &= \vec{\lambda}_V(t_f) - \vec{\lambda}_r(t - t_f) \end{aligned} \tag{4.12}$$

The magnitude of the velocity costate can then be written as

$$\lambda_V(t) = \sqrt{\lambda_r^2 * (t - t_f)^2 - 2(\vec{\lambda}_r \cdot \vec{\lambda}_V(t_f)) * (t - t_f) + \lambda_V^2(t_f)} \tag{4.13}$$

Note that if $\lambda_V + \lambda_C = 0$, then the specific thrust cannot be determined, resulting in the singularity condition. However, as it turns out, the singularity condition can't hold. To see this, first note that if $\lambda_V + \lambda_C = 0$, by (4.10), $\dot{\lambda}_C = 0$ which means that λ_V and λ_C are constant. This implies that Λ would have to be singular over the entire time interval. The boundary condition in (4.11) gives $\lambda_C = -1$ so $\lambda_V = 1$. This implies that $\dot{\lambda}_V = 0$. Now consider the fact that $\lambda_V^2 = \vec{\lambda}_V \cdot \vec{\lambda}_V$. Differentiating gives $2\dot{\vec{\lambda}}_V \cdot \vec{\lambda}_V = 0$. Using (4.9) gives $-2\dot{\vec{\lambda}}_r \cdot \vec{\lambda}_V = 0$ which implies that $\vec{\lambda}_r \perp \vec{\lambda}_V$. Using (4.9) reveals that λ_r is constant. Taking $\lambda_r \neq 0$ results in a rotating λ_V. Therefore, λ_r and λ_V cannot remain orthogonal to each other over any finite time interval. Recall that for free time problems, the Hamiltonian equals zero everywhere along the optimal trajectory. If $\lambda_r = 0$ then by inspection of the Hamiltonian in (4.7), $\lambda_V \perp g$ in order for the Hamiltonian to be zero. Since maximizing the Hamiltonian requires λ_V to be in the same direction as the thrust, this would result in a thrust in the horizontal direction. Therefore, there would be no way to achieve a soft landing given by the boundary conditions. So there can't be any optimal singular thrust arcs.

Using all these expressions for the costates begins to give an idea of what the thrust profile might look like. Given the possible values of λ_r and $\lambda_V(t_f)$, there are three possibilities.

Case 1: $\lambda_r \neq 0$ and $\lambda_V(t_f) \neq 0$. The graph of $\lambda_V(t)$ will be an upwards facing hyperbola. An example of what the graphs of $\lambda_V(t)$ and λ_C might look like can be obtained by solving the set of differential equations for $\dot{\vec{\lambda}}_r$, $\dot{\vec{\lambda}}_V$ and $\dot{\lambda}_C$ given by (4.10) and (4.9) and using reasonable values for λ_r and $\lambda_V(t_f)$, both of which are constant vectors. Looking at the switching function (4.8) reveals that the thrust will switch when $\lambda_V = -\lambda_C$. Figure 2.1 shows a possible example for the time evolution of λ_V and $-\lambda_C$. Note that the intersections of the two curves represents a switch from max to min or min to max thrust. It is also clear that the switch occurs when $\dot{\lambda}_C = 0$. In addition, $-\lambda_C$ can cross the decreasing part of the graph of λ_V only once, and the increasing part only once, resulting in a "bang-bang" profile. Also, in order to maximize the Hamiltonian, when $-\dot{\lambda}_C > 0$ the thrust will be at a maximum and when $-\dot{\lambda}_C < 0$ the thrust will be at a minimum. Therefore, the sample thrust profile in figure 4.1 will be a max-min-max firing.

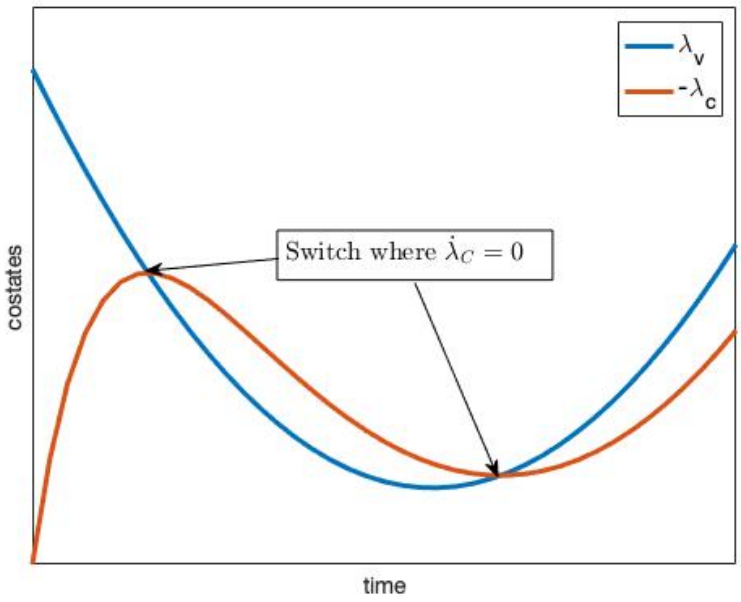

Figure 4.1: Costate Evolution

Case 2:$\lambda_r \neq 0$ and $\lambda_V(t_f) = 0$. Using (4.12) and simplifying gives $\lambda_V(t) = \lambda_r(t - t_f)$ which is a linear graph with a negative slope. Since the final time is free, transversality (2.12) gives $\mathcal{H}(t_f) = 0$ which means that $\lambda_V(t_f) + \lambda_C(t_f) = 0$. However, this isn't possible since $\lambda_V(t_f) = 0$ and $\lambda_C(t_f) = -1$ from (4.11).

Case 3: $\lambda_r = 0$ and $\lambda_V(t_f) \neq 0$. In this case, using the second equation in (4.9) results in a constant value for $\vec{\lambda}_V(t)$. Looking at the Hamiltonian reveals that since $\vec{\lambda}_V \cdot \vec{g}$ is constant, $\lambda_V + \lambda_C$ cannot switch signs. Therefore, there can be only one arc of maximum thrust. That makes this case just a special case of case 1.

4.3 Formulation as a Convex Optimization Problem

Now the main goal shifts to trying to express the problem as a convex optimization problem. The issue is the constraint on the thrust, in that $T_{\min} \leq T \leq T_{\max}$ is non-convex. Dr. Açikmeşe and others [2] deal with this problem by introducing a slack variable as an additional constraint that serves to make the thrust constraint convex at the cost of an extra constraint. The problem can be stated as

$$\max m(t_f) = \min \int_0^{t_f} ||T(t)||dt \tag{4.14}$$

subject to a variety of constraints including the mass loss rate, initial and final conditions, the dynamics and the minimum and maximum thrust constraint. Reformulating this with the slack variable gives

$$\min \int_0^{t_f} \Gamma(t)dt \tag{4.15}$$

where Γ is the slack variable that will now replace T in the previous thrust and mass loss constraints. Furthermore, this problem will have an additional constraint that $||T(t)|| \leq \Gamma(t)$. Because (4.15) is a less restrictive version of (4.14), we must show that an optimal solution of (4.15) is also an optimal solution of (4.14). Recalling that $\dot{m} = -kT$ and $\lambda_0 \leq 0$, the Hamiltonian (2.4) can be written as

$$\mathcal{H} = \vec{\lambda}_r \cdot \vec{V} + \vec{\lambda}_V \cdot \left(\vec{g} + \frac{\vec{T}(t)}{m(t)} \right) - \lambda_m k\Gamma + \lambda_0 \Gamma. \tag{4.16}$$

As usual, applying Pontryagin's Maximum Principle (see chapter 2) and optimal control principles (2.8) gives the following:

$$\dot{\vec{\lambda}}_r = 0 \qquad \dot{\vec{\lambda}}_V = -\vec{\lambda}_r \qquad \dot{\lambda}_m = \frac{1}{m^2}(\vec{\lambda}_V \cdot \vec{T}) \tag{4.17}$$

In addition, transversality (2.12) gives $\lambda_m(t_f) = 0$ and $\mathcal{H}(t_f) = 0$. Note that if $\lambda_V(t) = 0$ over any finite time interval, then the optimal thrust is indeterminate. However, the following proof shows that $\lambda_V(t)$ can only equal zero for at most one point on a finite time interval. If $\lambda_V = 0$, then using (4.17) shows that $\lambda_r = 0$ and $\lambda_m = 0$. Since $\mathcal{H}(t_f) = 0$, then by (4.16) $\lambda_0 = 0$. This would mean that $\begin{bmatrix} \lambda_0 & \lambda_r & \lambda_V & \lambda_m \end{bmatrix} = 0$ which violates the necessary conditions of optimality given by (4.17) and the Maximum Principle [27]. Also, by (4.17), λ_V is linear and since it can't be equal to zero everywhere, it can equal zero at most once on some time interval. In addition, looking at (4.16) reveals that the Hamiltonian depends linearly on the thrust. Since the optimal thrust will maximize the Hamiltonian, $T = \Gamma$ if the coefficient of T is positive and $T = -\Gamma$ if the coefficient is negative. This also leads to $\rho_{\min} \leq T \leq \rho_{\max}$. Finally, one can see that the optimal solution to (4.15) satisfies the constraints for (4.14) and therefore is a solution to both problems.

4.4 Application of Algorithm

A change of variables is now introduced in order to more easily construct a numerical algorithm. Changing the variables results in second-order cone and linear constraints which enables the use of more efficient algorithms. First, let

$$\sigma = \frac{\Gamma}{m}. \tag{4.18}$$

Now we can rewrite (4.1) as

$$\frac{\dot{m}(t)}{m(t)} = -k\sigma(t) \tag{4.19}$$

and write the dynamics as

$$\ddot{r} = g + \Lambda(t). \tag{4.20}$$

Solving (4.19) for $m(t)$ gives

$$m(t) = m_0 \exp\left[-k \int_0^{t_f} \sigma(t)dt \right] \tag{4.21}$$

Again, the goal is to minimize the fuel burned which is equivalent to maximizing the final mass. Therefore, by inspecting (4.21), this problem is equivalent to minimizing

$$\int_0^{t_f} \sigma(t)dt \tag{4.22}$$

The constraints can then be rewritten as

$$||\Lambda(t)|| \leq \sigma(t) \tag{4.23}$$

$$\frac{\rho_{\min}}{m(t)} \leq \sigma(t) \leq \frac{\rho_{\max}}{m(t)}. \tag{4.24}$$

Note that (4.23) becomes an equality for an optimal thrust as shown in the proof at the end of the previous section. By inspection, both inequality constraints in (4.24) are non-convex. To get around that problem, let

$$z = \ln m \tag{4.25}$$

and, using (4.19),

$$\dot{z}(t) = -k\sigma(t). \tag{4.26}$$

Then (4.24) can be written as

$$\rho_{\min} e^{-z(t)} \leq \sigma(t) \leq \rho_{\max} e^{-z(t)}. \tag{4.27}$$

Constraints of the form $g(x) \leq 0$ are convex if the Hessian of $g(x)$ is positive semi-definite. Looking at the first part of the inequality results in a Hessian for $g(x)$ that is positive definite, and therefore that constraint is convex. However, the second part of the inequality is not convex, as the Hessian is in fact negative definite. To fix this problem, the exponentials can be approximated using a Taylor series. Expanding the first inequality using a second order Taylor expansion results in a second-order cone constraint and the second inequality using a first order expansion results in a linear constraint. Keeping the problematic constraint to a first order expansion results in a convex constraint, whereas a second order expansion would not. Regardless,

both of these inequalities are shown to be very accurate in [2]. So the constraints in (4.27) can be written as

$$\mu_1(t)\left[1-[z(t)-z_0(t)]+\frac{[z(t)-z_0(t)]^2}{2}\right]\leq\sigma(t)\leq\mu_2(t)[1-(z(t)-z_0(t))] \tag{4.28}$$

where

$$\mu_1=\rho_{\min}e^{-z_0}\quad\text{and}\quad\mu_2=\rho_{\max}e^{-z_0} \tag{4.29}$$

and

$$z_0(t)=\ln(m_0-k\rho_{\max}t) \tag{4.30}$$

is the lower bound on $z(t)$ as it represents fuel being burned at maximum thrust. An upper bound can be written by substituting $\rho_{\min}$ for $\rho_{\max}$. The optimization problem can now be summarized as

$$\min\int_0^{t_f}\sigma(t)dt$$

subject to

$$\ddot{r}=g+\Lambda(t)$$

$$\dot{z}(t)=-k\sigma(t)\qquad||\Lambda(t)||\leq\sigma(t)$$

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

$$\ln(m_0-k\rho_{\max}t)\leq z(t)\leq\ln(m_0-k\rho_{\min}t)$$

$$m(0)=m_0\quad r(0)=r_0\quad\dot{r}(0)=\dot{r}_0\quad r(t_f)=\dot{r}(t_f)=0$$

Now the problem needs to be discretized in order to use a numerical algorithm. First, let N equal the number of time steps. Then, the control can be represented at each time step by

$$\eta=\begin{bmatrix}\Lambda_x & \Lambda_y & \Lambda_z & \sigma\end{bmatrix}^T \tag{4.31}$$

To get the discretized cost function (4.22), first define a matrix Ω as

$$\Omega=\begin{bmatrix}0 & 0 & 0 & \Delta t\end{bmatrix}^T \tag{4.32}$$

The cost function can then be represented as $\Omega^T \eta$ at each time step. In addition, the thrust constraint can now be written as

$$\left|\eta(1:3)\right| \leq \eta(4) \tag{4.33}$$

where $\eta(1:3)$ represents the first three terms of η and $\eta(4)$ is the last term of η. To discretize the dynamics, first consider the usual state space form, $\dot{x} = Ax + Bu + G$. The dynamics gives

$$A = \begin{bmatrix} 0_{3x3} & I_{3x3} & 0 \\ 0_{3x3} & 0_{3x3} & \vdots \\ 0 & \dots & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0_{3x3} & 0 \\ I_{3x3} & 0 \\ 0 \quad 0 \quad 0 & -k \end{bmatrix} \quad G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.34}$$

The solution to this equation is

$$x_{n+1} = A_{\text{step}} x_n + B_{\text{step}} \eta_n + g_{\text{step}} \tag{4.35}$$

where

$$A_{\text{step}} = e^{A\Delta t} \qquad B_{\text{step}} = \int_0^{\Delta t} e^{A(\Delta t - s)} B ds \qquad g_{\text{step}} = \int_0^{\Delta t} e^{A(\Delta t - s)} G ds \tag{4.36}$$

Finally we have the necessary framework to obtain a numerical solution.

4.5 Discussion

For this simulation, a spacecraft landing on Mars with six thrusters, firing at an angle of 27° with respect to the vertical, is considered. Figure 2.2 shows an optimal trajectory, using Matlab and CVX, with initial

conditions as follows:

$$r = \left[r_z, r_x, r_y\right] = \left[1500, 0, 2000\right] \text{ m}$$
$$\dot{r} = \left[\dot{r}_z, \dot{r}_x, \dot{r}_y\right] = \left[-75, 0, 100\right] \text{ m/s}$$
$$I_{sp} = 225 \text{ s}$$
$$g_{\text{mars}} = -3.7114 \text{ m/s}^2$$
$$T_{\max} = 3100 \text{ N/thruster}$$
$$\rho_{\min} = 0.3T_{\max}$$
$$\rho_{\max} = 0.8T_{\max}$$

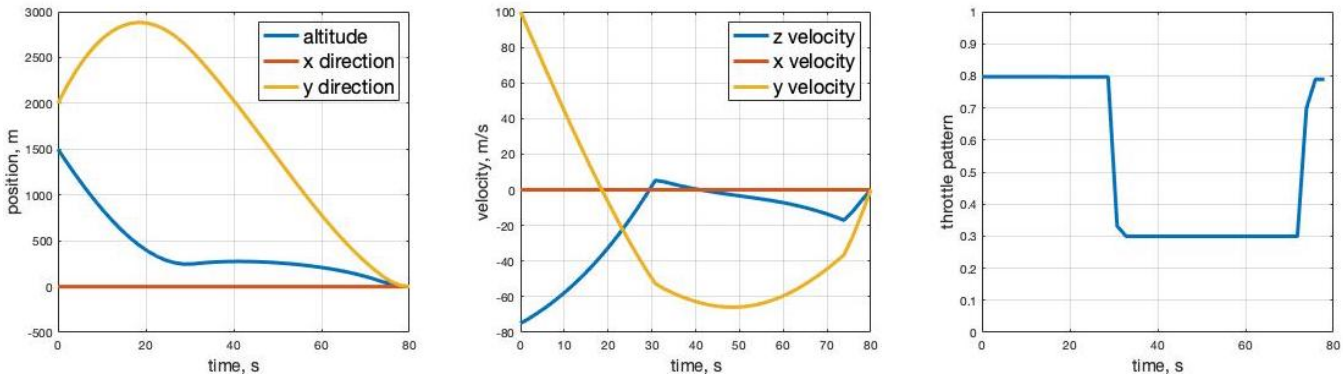

Figure 4.2: Optimization Results

Additionally, two important constraints have been added. First, solutions sometimes will result in a trajectory that goes underground. So the first additional constraint is that $r(1) \geq 0$. Second, trajectories that come in too parallel to the ground are more likely to crash into an unforeseen obstacle. Therefore, a glide slope constraint has been included that limits the angle that the spacecraft can make with the ground and the origin. The first two graphs in figure 2.2 show the position and velocity vs time of the spacecraft. The horizontal component initially moves farther away from the desired landing spot, but eventually gets back to the correct spot. The altitude of the spacecraft actually increases for a brief period of time around 30 seconds, as the velocity goes positive, but then gently descends for the remainder of the way. The third graph shows the optimal thrust profile. Not surprisingly, it shows a max-min-max pattern as predicted in [2]. Figure 2.3 shows the actual trajectory of the spacecraft as an observer on the surface might see it. The graph shows that the trajectory remains above the ground, doesn't come in too shallow, and lands at the

origin.

It is important to note that CVX and Matlab can be comparatively slow. Using a generic solver is fine for educational purposes, but should not be flown. The G-FOLD algorithm mentioned in chapter one, however, is fast enough to be used in real time and has been used to land spacecraft on Mars with this problem formulation. Also, the powered descent phase is the final of several steps in the EDL process (see chapter one). This phase begins when the parachute is cut off from the payload. The optimal time to cut off the parachute is an ongoing area of research and an essential part of the process. In addition, the spacecraft's attitude was not explicitly considered here.

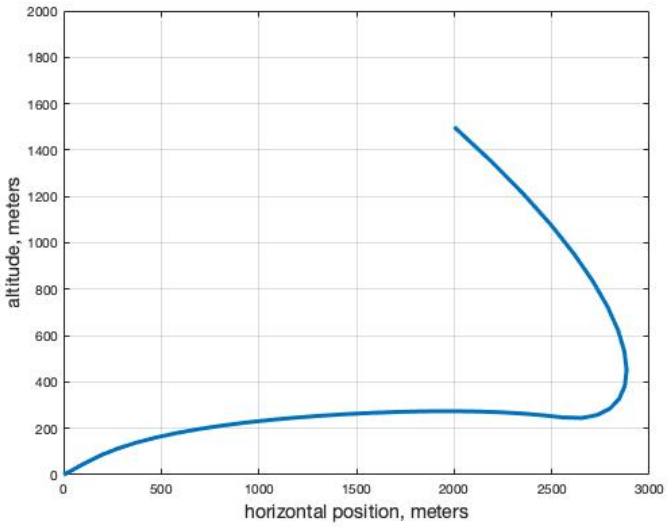

Figure 4.3: Optimal Trajectory

Chapter 5

The Indirect Method

5.1 Introduction

This chapter investigates a formulation of the powered descent optimization problem by deriving closed-form solutions to the state variables in terms of the initial costates and the free final time, as presented by Dr. Lu in several papers [17, 18, 19]. By applying the conditions of Pontryagin's Maximum Principle given by (2.18) through (2.21), the problem can be stated as a root-finding problem of seven equations with seven unknowns. It was found that several popular root-finding algorithms, such as Newton's method and gradient descent, do not converge. At Dr. Lu's suggestion, Powell's dog leg was implemented which did achieve convergence. The algorithm was applied to a small lunar lander in the final stage of powered descent.

5.2 Problem Formulation

While Dr. Lu formulates the problem using a linear gravity model, this paper focuses on the constant gravity model. The dynamics of the descent are given as

$$\dot{r} = \vec{V} \tag{5.1}$$

$$\dot{V} = \vec{g} + \frac{T}{m(t)}\vec{1}_T \tag{5.2}$$

$$\dot{m} = -\alpha T \tag{5.3}$$

where r, V , and m are position, velocity, and mass respectively, T is the magnitude of the thrust, α is the reciprocal of the exhaust velocity, and $\vec{1}_T$ is a unit vector in the direction of the thrust. We wish to maximize

the final mass, leading to the following cost function and Hamiltonian

$$J = \int_0^{t_f} \alpha T dt \tag{5.4}$$

$$H = \lambda_r^T V + \lambda_V^T \left(\vec{g} + \frac{T}{m(t)} \vec{1}_T \right) + \lambda_m (-\alpha T) - \alpha T \tag{5.5}$$

In order to make the problem formulation easier for reasons that will soon be evident, numerical scaling is necessary. Scaling the position by R_0 equal to the lunar radius, acceleration by g_0 equal to lunar gravitational acceleration, velocity by $\sqrt{g_0 R_0}$ and time by $\sqrt{R_0/g_0}$ gives the following for the dimensionless dynamics and the Hamiltonian:

$$\dot{r} = \vec{V} \tag{5.6}$$

$$\dot{V} = \frac{\vec{g}}{g_0} + \frac{T}{g_0 m(t)} \vec{1}_T \tag{5.7}$$

$$\dot{m} = -\alpha T \sqrt{\frac{R_0}{g_0}} \tag{5.8}$$

$$H = \lambda_r^T V + \lambda_V^T \left(\frac{\vec{g}}{g_0} + \frac{T}{g_0 m(t)} \vec{1}_T \right) + \lambda_m (-\alpha T) \sqrt{\frac{R_0}{g_0}} - \alpha T \sqrt{\frac{R_0}{g_0}} \tag{5.9}$$

We also know that $\dot{\lambda}_r = -\frac{\partial H}{\partial r}$ and $\dot{\lambda}_V = -\frac{\partial H}{\partial V}$. Consequently, the costate derivatives can be written as follows.

$$\begin{pmatrix} \dot{\lambda}_r \\ \dot{\lambda}_V \end{pmatrix} = \begin{pmatrix} 0 \\ -\lambda_r \end{pmatrix} \tag{5.10}$$

Writing the dynamics in state space form with scaled, dimensionless units yields

$$\dot{x} = \begin{bmatrix} & & \\ 0_{3x3} & & I_{3x3} \\ & & \\ & & \\ 0_{3x3} & & 0_{3x3} \\ & & \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \\ V_x \\ V_y \\ V_z \end{bmatrix} + \begin{bmatrix} & & & \\ & 0_{3x3} & & \\ & & & \\ \frac{1}{g_0 m(t)} & 0 & 0 & \\ 0 & \frac{1}{g_0 m(t)} & 0 & \\ 0 & 0 & \frac{1}{g_0 m(t)} & \end{bmatrix} \begin{bmatrix} T_x \\ \\ T_y \\ \\ T_z \\ \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \\ g \\ \\ \end{bmatrix} \tag{5.11}$$

and the solution is given by

$$x(t) = \Phi(t, t_0) x_0 + \int_{t_0}^{t} \Phi(t, \tau) B u d\tau + \int_{t_0}^{t} \Phi(t, \tau) E g d\tau \tag{5.12}$$

where E is the matrix preceding g in the state space equation. Let's look at these piece by piece. The first term is simple enough as,

$$\Phi(t,t_0) = e^{A(t-t_0)} = \begin{bmatrix} 1 & 0 & 0 & t-t_0 & 0 & 0 \\ 0 & 1 & 0 & 0 & t-t_0 & 0 \\ 0 & 0 & 1 & 0 & 0 & t-t_0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

and x_0 is just the initial position and velocity. The second and third terms require some manipulation before they can be worked with. First, using properties of the state transition matrix gives

$$\Phi(t,\tau)\Phi(\tau,0) = \Phi(t,0) \tag{5.14}$$

$$\Phi(t,\tau) = \Phi(t,0)\Phi^{-1}(\tau,0) \tag{5.15}$$

$$\Phi(t,\tau) = \Phi(t,0)\Phi(0,\tau) \tag{5.16}$$

So the second term of (5.12) can be written as follows...

$$\int_{t_0}^{t} \Phi(t,\tau)Bud\tau = \int_{t_0}^{t} \Phi(t,0)\Phi(0,\tau)Bud\tau = \Phi(t,0)\int_{t_0}^{t} \Phi(0,\tau)Bud\tau \tag{5.17}$$

Now the integrand can be expanded, which gives

$$\Phi(t,0)\int_{t_0}^{t} \Phi(0,\tau)Bud\tau = \Phi(t,0)\int_{t_0}^{t} \begin{bmatrix} \vec{1}_T \frac{T}{g_0 m(\tau)}(-\tau) \\ \vec{1}_T \frac{T}{g_0 m(\tau)} \end{bmatrix} d\tau \tag{5.18}$$

The third term of (5.12) can be worked with in the same way, with the added benefit that it ends up being a simple integral.

$$\int_{t_0}^{t} \Phi(t,\tau)Egd\tau = \int_{t_0}^{t} \Phi(t,0)\Phi(0,\tau)Egd\tau = \Phi(t,0)\int_{t_0}^{t} \Phi(0,\tau)Egd\tau = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{2}gt^2 - gtt_0 + \frac{1}{2}gt_0^2 \\ 0 \\ 0 \\ g(t-t_0) \end{bmatrix} \tag{5.19}$$

The Maximum Principle states that the control that maximizes the Hamiltonian is optimal. Looking at the Hamiltonian reveals that the control is the direction of the thrust vector. Since that thrust vector is dotted with the velocity costate in the Hamiltonian, maximizing the Hamiltonian results in the velocity costate pointing in the same direction as the thrust vector. Therefore we can rewrite (5.18) as

$$\Phi(t,0)\int_{t_0}^{t}\begin{bmatrix}\vec{1}_T\frac{T}{g_0 m(\tau)}(-\tau)\\ \vec{1}_T\frac{T}{g_0 m(\tau)}\end{bmatrix}d\tau=\Phi(t,0)\int_{t_0}^{t}\begin{bmatrix}\vec{1}_{\lambda_V}\frac{T}{g_0 m(\tau)}(-\tau)\\ \vec{1}_{\lambda_V}\frac{T}{g_0 m(\tau)}\end{bmatrix}d\tau \tag{5.20}$$

Building on the work of Pontryagin, Dr. Meditch showed in his 1964 paper [5] that the optimal thrust profile consists of maximum or minimum thrust over a particular time interval. Therefore, applying these equations to a singular thrust arc results in a constant thrust, T. So that does not present any challenges. However, an expression for $\vec{1}_{\lambda V}$ is necessary in order to evaluate this integral. To get that expression, first note that the costates can be written in state space form as follows

$$\dot{\lambda}=\begin{bmatrix}O_{3x3} & O_{3x3}\\ -I_3 & O_{3x3}\end{bmatrix}\begin{bmatrix}\lambda_r(t)\\ \lambda_V(t)\end{bmatrix} \tag{5.21}$$

which leads to $p=e^{At}p_0$, or

$$\lambda=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\-t&0&0&1&0&0\\0&-t&0&0&1&0\\0&0&-t&0&0&1\end{bmatrix}\lambda_0\implies\begin{pmatrix}\lambda_r=\lambda_{r_0}\\ \lambda_V=-\lambda_{r_0}t+\lambda_{V_0}\end{pmatrix} \tag{5.22}$$

Now $\vec{1}_{\lambda V}=\frac{\vec{\lambda}_V}{|\lambda_V|}$ can be evaluated by performing a second order expansion about the point $\vec{\lambda}_{V0}$. With a little algebra, it can be shown that the expansion leads to

$$\vec{1}_{\lambda_V}\approx\frac{\vec{\lambda}_{V_0}}{|\vec{\lambda}_{V_0}|}+\eta_1 t+\eta_2 t^2 \tag{5.23}$$

where

$$\eta_1=\frac{\lambda_{r_0}^T\lambda_{V_0}}{|\lambda_{V_0}|^3}\vec{\lambda}_{V_0}-\frac{1}{|\lambda_{V_0}|}\vec{\lambda}_{r_0},\qquad \eta_2=\left(2\frac{(\lambda_{r_0}^T\lambda_{V_0})^2}{|\lambda_{V_0}|^5}-\frac{|\lambda_{r_0}|^2}{|\lambda_{V_0}|^3}\right)\vec{\lambda}_{V_0}+\frac{(\lambda_{r_0}^T\lambda_{V_0})^2}{|\lambda_{v_0}|^2}\eta_1 \tag{5.24}$$

Here we are able to ignore terms of order t^3 and higher because of numerical scaling. With scaling, the dimensionless time, t is much less than 1. Therefore, as higher order terms are added, the value of t^n gets very close to zero. Without scaling, this approximation is not valid, and the whole problem formulation falls apart.

Using the derived value in (5.23) for $\vec{1}_{\lambda_V}$, the integral (5.20) can now be written as

$$\Phi(t,0)\int_{t_0}^{t}\begin{bmatrix}\vec{1}_{\lambda_V}\frac{T}{g_0 m(\tau)}(-\tau)\\ \vec{1}_{\lambda_V}\frac{T}{g_0 m(\tau)}\end{bmatrix}d\tau = \Phi(t,0)\int_{t_0}^{t}\begin{bmatrix}\vec{1}_{\lambda_{V_0}}\frac{T}{g_0 m(\tau)}(-\tau)+\vec{\eta}_1\frac{T}{g_0 m(\tau)}(-\tau^2)+\vec{\eta}_2\frac{T}{g_0 m(\tau)}(-\tau^3)\\ \vec{1}_{\lambda_{V_0}}\frac{T}{g_0 m(\tau)}+\vec{\eta}_1\frac{T}{g_0 m(\tau)}\tau+\vec{\eta}_2\frac{T}{g_0 m(\tau)}\tau^2\end{bmatrix}d\tau \tag{5.25}$$

$$= \Phi(t,0)\begin{bmatrix}-L\vec{1}_{\lambda_{V_0}}-M\eta_1-N\eta_2\\ K\vec{1}_{\lambda_{V_0}}+L\eta_1+M\eta_2\end{bmatrix} \tag{5.26}$$

where K, L, M, and N are given by

$$K = \frac{T}{g_0}\int_{t_0}^{t}\frac{1}{m_0+\dot{m}\tau}d\tau = \frac{T}{\dot{m}g_0}\ln\left|\left(\frac{m_0+\dot{m}t}{m_0+\dot{m}t_0}\right)\right| \tag{5.27}$$

$$L = \frac{T}{g_0}\int_{t_0}^{t}\frac{\tau}{m_0+\dot{m}\tau}d\tau = \frac{T}{\dot{m}^2 g_0}\left(\dot{m}(t-t_0)+m_0\ln\left|\left(\frac{m_0+\dot{m}t_0}{m_0+\dot{m}t}\right)\right|\right) \tag{5.28}$$

$$M = \frac{T}{g_0}\int_{t_0}^{t}\frac{\tau^2}{m_0+\dot{m}\tau}d\tau = \frac{T}{\dot{m}^2 g_0}\left(\frac{m_0^2}{\dot{m}}\ln\left(\left|\frac{m_0+\dot{m}t}{m_0+\dot{m}t_0}\right|\right)+\frac{1}{2}(\dot{m}(t^2-t_0^2)+2m_0(t-t_0)\right) \tag{5.29}$$

$$N = \frac{T}{g_0}\int_{t_0}^{t}\frac{\tau^3}{m_0+\dot{m}\tau}d\tau = \frac{T}{g_0}\left(\frac{m_0^2 t}{\dot{m}^3}-\frac{m_0 t^2}{2\dot{m}^2}+\frac{t^3}{3\dot{m}}-\frac{m_0^3}{\dot{m}^4}\ln\left(\left|m_0+\dot{m}t\right|\right)-\frac{m_0^2 t_0}{\dot{m}^3}-\frac{m_0 t_0^2}{2\dot{m}^2}+\frac{t_0^3}{3\dot{m}}-\frac{m_0^3}{\dot{m}^4}\ln\left(\left|m_0+\dot{m}t_0\right|\right)\right) \tag{5.30}$$

With this, (5.12) now looks like:

$$x(t) = \Phi(t,t_0)x_0+\Phi(t,0)\begin{bmatrix}-L\vec{1}_{\lambda_{V_0}}-M\eta_1-N\eta_2\\ K\vec{1}_{\lambda_{V_0}}+L\eta_1+M\eta_2\end{bmatrix}+\begin{bmatrix}0\\0\\\frac{1}{2}gt^2-gtt_0+\frac{1}{2}gt_0^2\\0\\0\\g(t-t_0)\end{bmatrix} \tag{5.31}$$

This expression represents the solution for the position and velocity of the spacecraft at time t. Since we want the final position and velocity to be zero, we can set this whole expression equal to zero and solve. However these six equations (each row of x represents a position or velocity dimension) contain seven

unknowns: the three-dimensional initial position and velocity costates, and the final time. However, applying the transversality condition to the Hamiltonian yields

$$H(t_f) = \lambda_{r0}^T V(t_f) + (-\lambda_{r0} t_f + \lambda_{V0})^T \left(\frac{\vec{g}}{g_0} + \frac{T_{max}}{g_0 m(t_f)} \right) - \alpha T_{max} \sqrt{\frac{R_0}{g_0}} = 0 \tag{5.32}$$

5.3 One Arc Solution

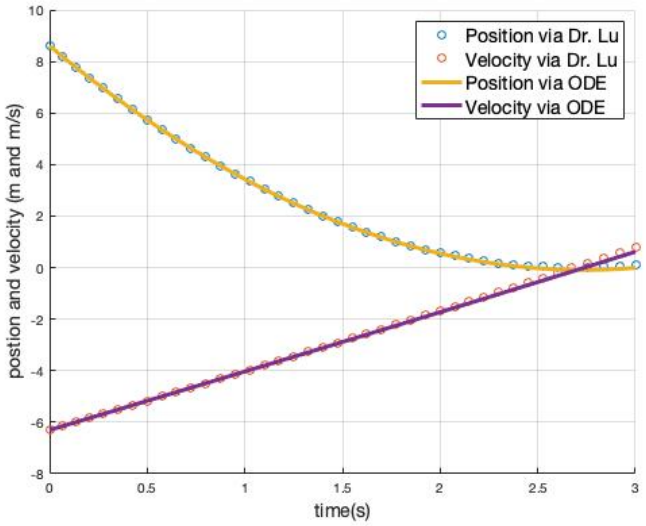

Figure 5.1: One arc trajectory

These seven equations are now applied to a manufactured problem for testing purposes. Figure 5.1 represents a singular arc of thrust in one dimension that when fired continuously results in a soft landing. Position and velocity graphs are plotted versus time using two methods. One method is simply solving the differential equation with all of the given initial conditions. The other method is the one outlined in this paper. We can see that the two methods produce very similar results, which is encouraging. The two methods show nearly identical trajectories and produce a soft landing with x and v both going to zero around $t = 2.7$ seconds. However, it's important to note that the algorithm is *very* sensitive to initial conditions and scaling factors. There is a relatively small range of values that produce the desired results. Therefore the code couldn't be used as it now stands because a slight change in the guess for the initial costates results in a nonsense solution. Still, it is reassuring that the values do line up, which seems to indicate that the approach and the code is valid.

5.4 Full Solution

Now we want to apply the problem to the max-min-max thrust profile. To do this, the states are determined at the end of the first thrust arc and used as the initial states for the next thrust arc, and so on. Therefore, the states can be written as

$$x(t_1) = \Phi(t_1, 0)x_0 + \int_0^{t_1} \Phi(t_1, \tau)Bud\tau + \int_0^{t_1} \Phi(t_1, \tau)Egd\tau \tag{5.33}$$

$$x(t_2) = \Phi(t_2, t_1)x(t_1) + \int_{t_1}^{t_2} \Phi(t_2, \tau)Bud\tau + \int_{t_1}^{t_2} \Phi(t_2, \tau)Egd\tau \tag{5.34}$$

$$x(t_f) = \Phi(t_f, t_2)x(t_2) + \int_{t_2}^{t_f} \Phi(t_f, \tau)Bud\tau + \int_{t_2}^{t_f} \Phi(t_f, \tau)Egd\tau \tag{5.35}$$

Performing substitution, and making use of some properties of the state transition matrix, namely

$$\Phi(t, 0) = \Phi(t, \tau)\Phi(\tau, 0) \quad \text{and} \quad \Phi(t, \tau) = \Phi^{-1}(\tau, t) \tag{5.36}$$

and doing a lot of simplifying, yields

$$x(t_f) = \Phi(t_f, 0)\left(x_0 + \sum_{i=1}^{3} \Gamma_i + \sum_{i=1}^{3} G_i\right) \tag{5.37}$$

where Γ represents the thrust integrals given by

$$\Gamma_1 = \int_0^{t_1} \Phi(0, \tau)Bud\tau, \quad \Gamma_2 = \int_{t_1}^{t_2} \Phi(0, \tau)Bud\tau, \quad \Gamma_3 = \int_{t_2}^{t_f} \Phi(0, \tau)Bud\tau \tag{5.38}$$

G represents the gravitation integrals given by

$$G_1 = \int_0^{t_1} \Phi(0, \tau)Egd\tau, \quad G_2 = \int_{t_1}^{t_2} \Phi(0, \tau)Egd\tau, \quad G_3 = \int_{t_2}^{t_f} \Phi(0, \tau)Egd\tau \tag{5.39}$$

The Γ integrals are defined as in the single thrust arc case...

$$\int \Phi(0, \tau)Bud\tau = \begin{bmatrix} -L\vec{1}_{\lambda_{V_0}} - M\vec{\eta}_1 - N\vec{\eta}_2 \\ K\vec{1}_{\lambda_{V_0}} + L\vec{\eta}_1 + M\vec{\eta}_2 \end{bmatrix} \tag{5.40}$$

The G integrals are somewhat simpler, in that they can be integrated directly as before.

$$\Phi(0,\tau)Eg = \begin{bmatrix} 0 & 0 & -g\tau & 0 & 0 & g \end{bmatrix}' \tag{5.41}$$

Finally we have $x(t_f)$ which is a 6-dimensional vector representing the states as a function of the final time. To get the seventh equation, we use the Hamiltonian, as before, at the final time.

$$H(t_f) = \lambda_{r0}^T V(t_f) + (-\lambda_{r0} t_f + \lambda_{V0})^T \left(\frac{\vec{g}}{g_0} + \frac{T_{max}}{g_0 m(t_f)} \right) - \alpha T_{max} \sqrt{\frac{R_0}{g_0}} = 0 \tag{5.42}$$

where $m(t_f)$ is given by

$$m(t_f) = m_0 + \dot{m}_{max}(t_1 - 0) + \dot{m}_{min}(t_2 - t_1) + \dot{m}_{max}(t_f - t_2) \tag{5.43}$$

5.5 Discussion

Once again, we have seven equations and seven unknowns for the entire thrust profile. The seven equations are represented by the six equations contained in $x(t_f)$ and the single equation $H(t_f)$. Given that we want a soft landing at our landing site, we can say that the states, $\vec{r}$ and $\vec{V}$ should be zero at the final time. Furthermore, applying transversality to the free final time problem means that $H(t_f) = 0$. Therefore, the problem has now become one of finding the roots of seven equations in terms of the seven variables $\vec{\lambda}_{r0}$, $\vec{\lambda}_{V0}$, and t_f.

Unfortunately, the code isn't working for the full thrust arc. Again, the first arc appears to be accurate, but the second and third arcs aren't close to what they should be. It seems evident that the root-finding algorithm is very dependent on, and highly sensitive to, initial conditions and numerical scaling. In some cases, the algorithm converges and the objective function does appear to go to zero. This indicates that a solution is found but, so far, it has never been the correct solution as it does not make physical sense.

A first attempt at solving the root-finding problem involved using Newton's method. The 7x7 Jacobian that shows up in Newton's method was determined both analytically and numerically. However, the matrix was not well conditioned and the inverse could not be determined. Attempts to numerically approximate the inverse of the Jacobian didn't result in convergence. Further attempts were made to achieve convergence using gradient descent with carefully chosen step sizes, such as the Barzilai-Borwein step size given in [3].

Unfortunately, those methods fell short as well. Finally, as suggested in [18], Powell's dog leg method was implemented. The algorithm, as described in [21], was coded and tested. The code output matched the results given in [21], so the algorithm itself appears to be working fine. The algorithm was then applied to the powered descent problem and convergence was finally achieved.

The dog leg method is a modified trust region method. A trust region method restricts how big a step can be taken. If the ratio of the difference in the objective function between successive steps to the linear model is large, then the linear model is good, and the trust region can be expanded. If that ratio is small, then the trust region needs to be reduced. When performing an iteration, the algorithm has three options. First, the Newton step is calculated by solving $J(x)h_n = -f(x)$ where $J(x)$ is the Jacobian matrix, h_n is the Newton step and $f(x)$ is the objective function. If h_n lands within the trust region, then that is the step. If not, then the steepest descent step is calculated using an appropriate step size. If that step is outside the trust region, then it is scaled down to land on the boundary of the trust region. Finally, if the steepest descent step is inside the trust region and Newton step is outside, then the step is a combination of both steps. This is the so-called dog leg step and is represented in figure 5.2 [21].

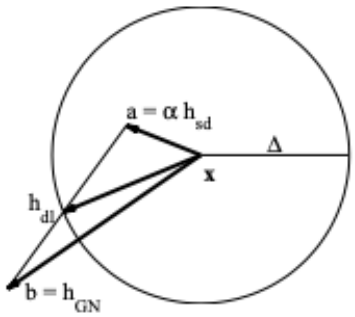

Figure 5.2: Dog leg step [21]

The code for this problem, which is contained in the appendix, is divided into a number of different functions in order to simplify the debugging process. The functions for determining η_1 and η_2 given in (5.24) were verified for scaled values of t. In addition, the K, L, M, and N integrals given in (5.27) through (5.30) were verified numerically using random numbers. The fact that the first thrust arc appears to be correct is promising. However, the bug in the code for the second and third arcs has yet to be found. There may be an issue with the numerical scaling, as converging on the correct solution is highly dependent on the scaling factors used. Furthermore, the initial guess for the costates also has a significant effect on the outcome.

There is some discussion in [18] of replacing the Hamiltonian equation with a different expression that would still satisfy the necessary conditions as well as transversality. Dr. Lu specifically states in this paper that the scaling is tricky if using the Hamiltonian. Future attempts will be made to implement these possible alternatives.

Chapter 6

Conclusion

6.1 Summary of Findings

The research here attempted to tackle the pinpoint landing problem from several different angles. First, an introduction to the problem definition was presented as well as a basic discussion of Apollo guidance. After laying a foundation of optimal control theory, the one-dimensional optimal descent problem was solved and simulated. Then the problem was formulated using the direct method of convex optimization and simulated Martian landing was developed. Finally, an indirect method for deriving the optimal trajectory was presented as a two-point boundary problem using a root-finding algorithm. The beginning of a landing was simulated, although the complete solution has yet to be simulated accurately.

Currently, the convex optimization approach using the G-FOLD algorithm is the preferred method for obtaining a pinpoint landing. The indirect method can be applied to the powered descent phase but may be more useful in other phases of mission, such as determining optimal ascent trajectories [17]. Both methods complement each other well, and each is potentially more suitable for specific applications.

Bibliography

[1] Behçet Açikmeşe, John M Carson, and Lars Blackmore. "Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem". In: *IEEE transactions on control systems technology* 21.6 (2013), pp. 2104–2113. ISSN: 1063-6536.

[2] Behçet Açikmeşe and Scott R Ploen. "Convex Programming Approach to Powered Descent Guidance for Mars Landing". In: *Journal of guidance, control, and dynamics* 30.5 (2007), pp. 1353–1366. ISSN: 0731-5090.

[3] Jonathan Barzilai and Jonathan M Borwein. "Two-Point Step Size Gradient Methods". In: *IMA Journal of Numerical Analysis* 8.1 (1988), pp. 141–148. ISSN: 0272-4979.

[4] Leonard David Berkovitz. *Optimal control theory*. New York, 1974.

[5] P J Boston et al. "Cave biosignature suites: microbes, minerals, and Mars". In: *Astrobiology* 1.1 (2001), pp. 25–55. ISSN: 1531-1074.

[6] Richard P. Brent. *Algorithms for Minimization Without Derivatives*. Englewood Cliffs, N.J., 1972.

[7] John M. Carson et al. "The SPLICE Project: Continuing NASA Development of GN&C Technologies for Safe and Precise Landing". In: *AIAA Scitech 2019 Forum*. DOI: 10.2514/6.2019-0660.

[8] Christopher D'Souza. "An optimal guidance law for planetary landing". In: *Guidance, Navigation, and Control Conference*. DOI: 10.2514/6.1997-3709.

[9] Daniel Dueri et al. "Customized Real-Time Interior-Point Methods for Onboard Powered-Descent Guidance". In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 197–212. DOI: 10.2514/1.G001480.

[10] Junichi Haruyama et al. "Lunar Holes and Lava Tubes as Resources for Lunar Science and Exploration". In: *Moon*. Springer Berlin Heidelberg, pp. 139–163. ISBN: 3642279686.

[11] CVX Research inc. *CVX Users Guide*. http://cvxr.com/cvx/doc/intro.html. Accessed: 05-2021.

[12] Allan R Klumpp. "Apollo Lunar Descent Guidance". In: *Automatica (Oxford)* 10.2 (1974), pp. 133–146. ISSN: 0005-1098.

[13] D. F. Lawden. "Optimal Trajectories for Space Navigation". In: *Planetary and Space Science* 12.2 (1964), p. 188. ISSN: 0032-0633.

[14] G Leitmann. "On a Class of Variational Problems in Rocket Flight". In: *Journal of the aerospace sciences* 26.9 (1959), pp. 586–591. ISSN: 1936-9999.

[15] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory : A Concise Introduction*. Princeton, N.J., 2012. ISBN: 9781400842643.

[16] Ping Lu. "Augmented Apollo Powered Descent Guidance". In: *Journal of Guidance, Control, and Dynamics* 42.3 (2019), pp. 447–457. DOI: 10.2514/1.G004048.

[17] Ping Lu. "Propellant-Optimal Powered Descent Guidance". In: *Journal of Guidance, Control, and Dynamics* 41 (Dec. 2018), pp. 1–14. DOI: 10.2514/1.G003243.

[18] Ping Lu, Stephen Forbes, and Morgan Baldwin. "A Versatile Powered Guidance Algorithm". In: *AIAA Guidance, Navigation, and Control Conference*. DOI: 10.2514/6.2012-4843.

[19] Ping Lu et al. "Rapid Optimal Multiburn Ascent Planning and Guidance". In: *Journal of Guidance, Control, and Dynamics* 31.6 (2008), pp. 1656–1664. DOI: 10.2514/1.36084.

[20] David G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications*. New York, 1979.

[21] Kaj Madsen, Hans Nielsen, and O Tingleff. *Methods for Non-Linear Least Squares Problems (2nd ed.)* Jan. 2004.

[22] J Meditch. "On the problem of optimal thrust programming for a lunar soft landing". eng. In: *IEEE transactions on automatic control* 9.4 (1964), pp. 477–484. ISSN: 0018-9286.

[23] Angelo Miele. "The Calculus of Variations in Applied Aerodynamics and Flight Mechanics". In: *Mathematics in Science and Engineering*. Vol. 5. 1962, pp. 99–170. ISBN: 0124429505.

[24] NASA. *Mars Exploration Program*. https://mars.nasa.gov/mars-exploration/missions. Accessed: 05-2021.

[25] NASA. *Mars Probe Landing Ellipses*. https://www.jpl.nasa.gov/images/mars-probe-landing-ellipses. Accessed: 05-2021.

[26] Scott Ploen, Bechet Acikmese, and Aron Wolf. "A Comparison of Powered Descent Guidance Laws for Mars Pinpoint Landing". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. DOI: 10.2514/6.2006-6676.

[27] L. S. (Lev Semenovich) Pontryagin. *The Mathematical Theory of Optimal Processes,* New York, 1962.

[28] M. J. D Powell. "An efficient method for finding the minimum of a function of several variables without calculating derivatives". In: *Computer journal* 7.2 (1964), pp. 155–162. ISSN: 0010-4620.

[29] Philip Rabinowitz, ed. *Numerical Methods for Nonlinear Algebraic Equations*. London, New York: Gordon and Breach Science Publishers, 1970. ISBN: 0677142307.

[30] Taylor Reynolds et al. "A Real-Time Algorithm for Non-Convex Powered Descent Guidance". In: *AIAA Scitech 2020 Forum*. DOI: 10.2514/6.2020-0844.

[31] Taylor P. Reynolds and Mehran Mesbahi. "Optimal Planar Powered Descent with Independent Thrust and Torque". In: *Journal of Guidance, Control, and Dynamics* 43.7 (2020), pp. 1225–1231. DOI: 10.2514/1.G004701.

[32] Daniel P. Scharf et al. "Implementation and Experimental Demonstration of Onboard Powered-Descent Guidance". In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 213–229. DOI: 10.2514/1.G000399.

[33] Ronald Sostaric and Jeremy Rea. "Powered Descent Guidance Methods For The Moon and Mars". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. DOI: 10.2514/6.2005-6287.

[34] Ufuk Topcu, Jordi Casoliva, and Kenneth D Mease. "Minimum-Fuel Powered Descent for Mars Pinpoint Landing". In: *Journal of spacecraft and rockets* 44.2 (2007), pp. 324–331. ISSN: 0022-4650.

[35] Edward C Wong, Gurkirpal Singh, and James P Masciarelli. "Guidance and Control Design for Hazard Avoidance and Safe Landing on Mars". In: *Journal of spacecraft and rockets* 43.2 (2006), pp. 378–384. ISSN: 0022-4650.

Chapter 7

Appendix

7.1 Vertical descent simulation from chapter 3

```
%Dr. Meditch simulation

clear;clc;close all

g = 1;              %g on the moon m/s^2
k = 30*9.806;       %velocity of exhaust wrt spacecraft m/s
alpha = 6.5/k;      %optimal value for mass flow rate:
u = -alpha;         %goes from -alpha to zero, alpha > 0
ksi1 = 30;          %initial altitude in meters
ksi2 = -1;          %initial velocity in m/s
M0 = 2;             %Initial mass at beginning of thrusting in kg

%define constants
a = 1/2 * ((k*alpha-g*M0)/M0);
b = k*alpha^2/(2*M0^2);

%solve dynamics
f = @(t,x) [x(2);-k/x(3)*u-g;u];
[ta,xa] = ode45(f,[0  10],[9  -6.3 M0]);
```

```
%plot powered decent (sanity check)
figure
grid on
hold on
%plot(xa(:,1),xa(:,2))
x1max = .0625*a*(M0/alpha)^2;
xlim([0 30])
x2min = -0.5*a*(M0/alpha)-0.0625*b*(M0/alpha)^2;
ylim([-10 0])
xlabel('altitude (m)','FontSize',16)
ylabel('altitude rate (m/s)','FontSize',16)

%plot powered descent from x1* and x2* (equations 14 and 15)
t=0:.1:10;
x1star = -k*M0/alpha.*log(1-alpha/M0.*t)-k.*t-1/2*g*t.^2;
x2star = -b/a*x1star - 2*a*sqrt(x1star/a);
plot(x1star,x2star,'LineWidth',3)

%plot free fall from kinematics
v = ksi2 - g.*t;
altitude = (v.^2-ksi2^2)/(-2*g)+30;
plot(altitude,v,'LineWidth',3)

%powered descent from direct integration
x1star0 = 9;
x2star0 = -6.3;
x2 = -k*log(1-alpha/M0.*t)-g*t+x2star0;
x1 = k*M0/alpha*(1-alpha/M0.*t).*log(1-alpha/M0.*t)+k.*t-1/2*g.*t.^2+...
    x2star0.*t + x1star0;
%plot(x1,x2,'o')
```

```
%Graph Labels
x = [altitude(1) 9.15 x1star(1)];
y = [v(1) -6.55 x2star(1)];
labels = {'Initial Conditions','Inititiation of Thrust','Landing'};
plot(x,y,'o','MarkerSize',15,'LineWidth',3,'HandleVisibility','off')
text(x,y,labels,'VerticalAlignment','bottom','HorizontalAlignment','right','FontSize',14)
legend('switching function','freefall','FontSize',16)

%plot x vs t and v vs t
x_traj = [altitude(1:56) flip(x1star(1:29))];
v_traj = [v(1:56) flip(x2star(1:29))];
t_traj = [linspace(0,5.55,56) linspace(5.6509,8.35,29)];
figure
hold on
grid on
plot(t_traj,x_traj,'LineWidth',3)
plot(t_traj,v_traj,'LineWidth',3)
xlabel('time(s)','FontSize',14)
set(gca, 'XTick', sort([5.6, get(gca, 'XTick')]));
t = annotation('textbox','String','Thrust Initiation');
sz = t.FontSize;
t.FontSize = 16;
legend('altitude','velocity','FontSize',16)
```

7.2 Convex optimization approach from chapter 4

7.2.1 Dr. Topcu's Paper [34]

```
%Dr. Topcu paper

clear;clc;close all
%these work
% k = 1;
```

```
% gamma_star = 1;
% ti = 0;
% tf = 10;
% rx0=1;ry0=1;rz0=-1;
% vx0=-1;vy0=1;vz0=10;
% lambda_v_tf = [vx0 vy0 vz0];
% lambda_r = [rx0 ry0 rz0];
% delta_t = .01;
% t = ti:delta_t:tf;

k = 1;
gamma_star = 1;
ti = 0;
tf = 10;
rx0=.2;ry0=.2;rz0=-.2;
vx0=-.1;vy0=-.1;vz0=-.1;
lambda_v_tf = [vx0 vy0 vz0];
lambda_r = [rx0 ry0 rz0];
delta_t = .01;
t = ti:delta_t:tf;

%initialize lambda_v
lambda_v = sqrt(dot(lambda_r,lambda_r)*(t-tf).^2 ...
    -2*dot(lambda_r,lambda_v_tf).*(t-tf) ...
    + dot(lambda_v_tf,lambda_v_tf));

%this worked
%plot(t,lambda_v)

% f = @(t,x) [0;0;0;-x(1)*(t-tf)+lambda_v_tf(1);...
%     -x(2)*(t-tf)+lambda_v_tf(2);-x(3)*(t-tf)+lambda_v_tf(3);...
%     -(sqrt(dot(lambda_r,lambda_r)*(t-tf)^2 ...
```

```
%       -2*dot(lambda_r,lambda_v_tf)*(t-tf)...
%       +dot(lambda_v_tf,lambda_v_tf))...
%       +x(7))*k*gamma_star];
% [t,xa] = ode45(f,[ti  tf],[rx0  ry0  rz0  lambda_v(1)  lambda_v(2)  ...
%       lambda_v(3)  1]);


%New stuff
f = @(t,x) [0;0;0;-x(1);-x(2);-x(3);-(norm(x(4:6))+x(7))*k*gamma_star];
[t,xa] = ode45(f,[ti  tf],[rx0  ry0  rz0  lambda_v(1)  lambda_v(2)  ...
    lambda_v(3)  -5.5]);
for  i = 1:length(xa)
    lambda_V(i) = norm(xa(i,4:6));
end
plot(t,lambda_V,'LineWidth',3)
hold on
grid on
plot(t,-xa(:,7),'LineWidth',3)
legend('\lambda_v','-\lambda_c','FontSize',16)
xlabel('time','FontSize',14)
ylabel('costates','FontSize',14)
set(gca,'XTick',[],'YTick',[])
t = annotation('textbox','String','Switch␣where␣$\dot\lambda_C=0$',...
    'Interpreter','latex','Position',[.4,.5,.1,.1]);
sz = t.FontSize;
t.FontSize = 16;
% mag = sqrt(xa(:,4).^2 + xa(:,5).^2 + xa(:,6).^2);
% plot(t,mag,'o')
```

7.2.2 Dr. Açikmeşe's Paper [2]

```
% Simulation of Dr. Acikmese's paper


clear;clc;close all
```

```
deltat = 2;                 %choosen value for time interval in seconds
N = 40;
t = linspace(0,N*deltat,N);

r0 = [1500 0 2000]';        %Initial position with altitude = 1500m
rdot0 = [-75 0 100]';       %Initial velocity m/s
mwet = 1905;                %Initial mass of spacecraft with fuel
Isp = 225;                  %Specific impulse in seconds
alpha = 1/(9.81 * Isp);     %constant associated with fuel consumption
gmars = -3.7114;            %g on mars in m/s^2
Tmax = 3100*6;              %max thrust for 6 thrusters in Newtons
rho1 = 0.3*Tmax;            %lower thrust bound
rho2 = 0.8*Tmax;            %upper thrust bound
phi = 27*pi/180;            %angle of thrusters




%Set up Ac, Bc and gc matrices
Ac = [ zeros(3,3) eye(3)      zeros(3,1);
       zeros(3,3) zeros(3,3) zeros(3,1);
       zeros(1,3) zeros(1,3) zeros(1,1) ];
Bc = [zeros(3,4); eye(3) zeros(3,1); 0 0 0 -alpha];
gc = [0 0 0 gmars 0 0 0]';

%equation for A from the paper
A = expm(Ac*deltat);
%perform integration to get B and g matrices
stepsize = 0.001;
B0 = zeros(7,4);
```

```
g0 = zeros(7,1);
B0 = [B0(:);g0(:)];
[T,Y] = ode45(@(t,x)integration(Ac,deltat,Bc,t,gc,x),[0,deltat],B0);
B = reshape(Y(end,1:28),7,4);
g = reshape(Y(end,29:35),7,1);

%Alternate way to get B and g. Yields same results as above
% for i = 0:stepsize:deltat
%     B = B + expm(Ac*(deltat - i))*Bc*stepsize;
%     g = g + expm(Ac*(deltat - i))*gc*stepsize;
% end

%Initialize everything
%gamma0 = 0;              %Just for kicks, to test freefall dynamics
%Tc = [0 0 0]';           %Also just for kicks, to test freefall dynamics
%sigma0 = gamma0/mwet;
%u0 = Tc/mwet;
z0 = log(mwet);
%zdot0 = -alpha * sigma0;
x0 = [r0; rdot0; z0];
xf = [0 0 0 0 0 0]';
%xa(:,1) = [r0; rdot0; z0];
%eta = [u0; sigma0];

z0 = log(mwet-alpha*rho2*t);
zmax = log(mwet-alpha*rho1*t);
mu1 = rho1 * exp(-z0);
mu2 = rho2 * exp(-z0);

cvx_begin
    esigma = [0 0 0 1]';
    omega = deltat * esigma;
```

```
    variables eta(4,N) x(7,N)
    Omega = repmat(omega,N,1);
    minimize(Omega' * eta(:))
    subject to
    x(:,1) == x0;
    x(1:6,N) == xf;
    for k = 1:N
        if (k<N)
            x(:,k+1) == A*x(:,k) + B*eta(:,k) + g;
        end
    %mass constraint
    z0(k) <= x(7,k) <= zmax(k);

    %thrust constraint concerning gamma
    norm(eta(1:3,k)) <= eta(4,k);

    %thrust constraint concerning upper and lower bounds
    mu1(k)*(1 - (x(7,k) - z0(k))+(x(7,k) - z0(k))^2/2) <= eta(4,k) <= ...
        mu2(k) * (1 - (x(7,k) - z0(k)));

    %Glide slope
    %norm(x(1:3,k))*cos() <= x(3,k)
    norm(x(2:3,k))-tan(86*pi/180)*x(1,k) <= 0;

    %Staying above ground
    x(1,k) >= 0;
    end

cvx_end

%Behcet's loop for freefall only (psi_k = 0)
% lambdak = B;
```

```
% for k = 1:N-1
%       xa(:,k+1) = A^k*xa(:,1) + lambdak*[gmars 0 0 0]';
%       lambdak = lambdak + (A^k)*B;
% end
mass = exp(x(7,:));
for q = 1:N-1%%%% was 1:39
    throttle(q) = -(mass(q+1) - mass(q))/(deltat*alpha);
end
throttle=throttle/(3100*6);
subplot(1,3,1)
plot (t,x(1,:),'LineWidth',3)
xlabel('time, s','FontSize',14)
ylabel('position, m','FontSize',14)
hold on
grid on
plot (t,x(2,:),'LineWidth',3)
plot (t,x(3,:),'LineWidth',3)
legend('altitude','x direction','y direction','FontSize',16)

subplot(1,3,2)
plot (t,x(4,:),'LineWidth',3)
hold on
grid on
xlabel('time, s','FontSize',14)
ylabel('velocity, m/s','FontSize',14)
plot (t,x(5,:),'LineWidth',3)
plot (t,x(6,:),'LineWidth',3)
legend('z velocity','x velocity','y velocity','FontSize',16)

subplot(1,3,3)
plot(t(1:N-1),throttle,'LineWidth',3)%%%%was 1:39
grid on
```

```
ylim([0,1])
xlabel('time, s','FontSize',14)
ylabel('throttle pattern','FontSize',14)

X=linspace(0,3000,10);
m=tan(4*pi/180);
Y=m*X;
figure
plot(x(3,:),x(1,:),'LineWidth',3)
grid on
hold on
%plot(X,Y,'LineWidth',2) %Glideslope line
xlim([0,3000])
ylim([0,2000])
xlabel('horizontal position, meters','FontSize',14)
ylabel('altitude, meters','FontSize',14)
```

7.3 Indirect method from chapter 5

7.3.1 Main Code

```
%required functions:
%getParams, getVars, getK, getL, getM, getN, getEta1, getEta2, getOnePV0, getVtf, getF, Brent,
%getJacobian

clear;clc;close all

p = getParams();

%assigns variables so we don't have to type "p." everytime
[RScale, VScale, gScale, tScale, M0, r0, V0, Tmax, Tmin, m0, alpha, mdotMax, mdotMin, g0, t1, t2, m1, m2]...
    = getVars(p);
```

```
%asymptote values, where KLMN integrals will fail
forbidden_t1 = m0/-mdotMax;
forbidden_t2 = m1/-mdotMin;
forbidden_tf = m2/-mdotMax;

pr0=[0;0;0];
%pr0=r0'./norm(r0);
pV0=-V0'./norm(V0);
%pr0=[1;0;1];
%pV0=-[1;0;1];
tf = 10 / tScale;

[K1,K2,K3] = getK(p);
[L1,L2,L3] = getL(p);
[M1,M2,M3] = getM(p);
[N1,N2,N3] = getN(p);

[eta1,eta1x,eta1y,eta1z] = getEta1();
[eta2,eta2x,eta2y,eta2z] = getEta2(eta1,eta1x,eta1y,eta1z);

[one_pV0,one_pV0x,one_pV0y,one_pV0z] = getOnePV0();

Vtf = getVtf(p,pV0,K1,K2,K3,L1,L2,L3,M1,M2,M3,eta1x,eta1y,eta1z,...
    eta2x,eta2y,eta2z,one_pV0x,one_pV0y,one_pV0z);

f = getF(p,pr0,pV0,K1,K2,K3,L1,L2,L3,M1,M2,M3,N1,N2,N3,eta1,eta1x,eta1y,eta1z,eta2,...
    eta2x,eta2y,eta2z,one_pV0,one_pV0x,one_pV0y,one_pV0z,Vtf);

%X = dogleg(pr0,pV0,tf,f);
X0 = [pr0;pV0;tf];
x = fsolve(f,X0);
```

7.3.2 Required Functions

```
function [p] = getParams()
%scaled


p = struct;


%p.RScale = 1.737e6; %lunar radius
p.RScale = 1e5;
%p.RScale = 6.378e6;
p.gScale = 1;    %lunar g = 1.625
%p.gScale = 9.81;
%M0 = 7.35e22;        %lunar mass
p.M0 = 1;
p.tScale = sqrt(p.RScale/p.gScale);
p.VScale = sqrt(p.gScale*p.RScale);


%p.r0 = [0 0 13.5] ./ p.RScale;
%p.V0 = [0 0 -2] ./ p.VScale;
p.r0 = [30 0 30] ./ p.RScale;
p.V0 = [-4 0 -6] ./ p.VScale;
p.Tmax = 6.5;
p.Tmin = 1.5;
p.m0 = 2 / p.M0;
p.alpha = 1/(30*9.81);
p.mdotMax = -p.alpha*p.Tmax * sqrt(p.RScale/p.gScale);
p.mdotMin = -p.alpha*p.Tmin * sqrt(p.RScale/p.gScale);
p.g0 = -1 / p.gScale;
p.t1 = 1 / p.tScale;
p.t2 = 2 / p.tScale;
p.m1 = p.m0 + p.mdotMax * p.t1;
p.m2 = p.m1 + p.mdotMin * (p.t2 - p.t1);
```

```
end

function [RScale,VScale,gScale,tScale,M0,r0,V0,Tmax,Tmin,m0,alpha,mdotMax,mdotMin,g0,t1,t2
    = getVars(p)
%This function inputs the structure p, and reassigns the variable names so
%that we don't have to type "p." in front of everything.
RScale = p.RScale;
VScale = p.VScale;
gScale = p.gScale;
tScale = p.tScale;
M0 = p.M0;

r0=p.r0;
V0=p.V0;
Tmax=p.Tmax;
Tmin=p.Tmin;
m0=p.m0;
alpha=p.alpha;
mdotMax=p.mdotMax;
mdotMin=p.mdotMin;
g0=p.g0;
t1=p.t1;
t2=p.t2;
m1=p.m1;
m2=p.m2;



end

function [K1,K2,K3] = getK(p)
%
[RScale,VScale,gScale,tScale,M0,r0,V0,Tmax,Tmin,m0,alpha,mdotMax,mdotMin,g0,t1,t2,m1,m2]...
    = getVars(p);
```

```
K1 = (1/gScale)*Tmax/mdotMax*log(abs(m0+mdotMax*t1)/abs(m0));
K2 = (1/gScale)*Tmin/mdotMin*log(abs((m1+mdotMin*t2))/abs((m1+mdotMin*t1)));
K3 = @(t) (1/gScale)*Tmax/mdotMax*log(abs((m2+mdotMax*t))/abs((m2+mdotMax*t2)));

end

function [L1,L2,L3] = getL(p)
%
[RScale,VScale,gScale,tScale,M0,r0,V0,Tmax,Tmin,m0,alpha,mdotMax,mdotMin,g0,t1,t2,m1,m2]...
    = getVars(p);

L1 = (1/gScale)*((mdotMax*Tmax*t1+m0*Tmax*log(abs(m0)/abs((m0+mdotMax*t1))))/mdotMax^2);
L2 = (1/gScale)*(Tmin/mdotMin^2*((t2-t1)*mdotMin+m1*log(abs(m1+mdotMin*t1)/...
    abs(m1+mdotMin*t2))));
L3 = @(t) (1/gScale)*(Tmax/mdotMax^2*((t-t2)*mdotMax+m2*log(abs(m2+mdotMax*t2)/...
    abs(m2+mdotMax*t))));
end

function [M1,M2,M3] = getM(p)
%
[RScale,VScale,gScale,tScale,M0,r0,V0,Tmax,Tmin,m0,alpha,mdotMax,mdotMin,g0,t1,t2,m1,m2]...
    = getVars(p);

M1 = (1/gScale)*(Tmax*(m0^2*log(abs((m0+mdotMax*t1))/abs(m0))/mdotMax^3-m0*t1/mdotMax^2+t1
    (2*mdotMax)));
M2 = (1/gScale)*(Tmin/mdotMin^2*(m1^2/mdotMin*log(abs(m1+mdotMin*t2)/abs(m1+mdotMin*t1))...
    +(mdotMin*(t2^2-t1^2)+2*m1*(t1-t2))/2));
M3 = @(t) (1/gScale)*(Tmax/mdotMax^2*(m2^2/mdotMax*log(abs(m2+mdotMax*t)/abs(m2+mdotMax*t2
    +(mdotMax*(t^2-t2^2)+2*m2*(t2-t))/2));
end

function [N1,N2,N3] = getN(p)
%
[RScale,VScale,gScale,tScale,M0,r0,V0,Tmax,Tmin,m0,alpha,mdotMax,mdotMin,g0,t1,t2,m1,m2]...
```

```
    = getVars(p);

N1 = (1/gScale)*(Tmax*(m0^2*t1/mdotMax^3-m0*t1^2/(2*mdotMax^2)+t1^3/(3*mdotMax)-m0^3/mdotM
    log(abs(m0+mdotMax*t1)/abs(m0))));
N2 = (1/gScale)*(Tmin*(m1^2*t2/mdotMin^3-m1*t2^2/(2*mdotMin^2)+t2^3/(3*mdotMin)-m1^3/mdotM
    log(abs(m1+mdotMin*t2))-(m1^2*t1/mdotMin^3-m1*t1^2/(2*mdotMin^2)+t1^3/(3*mdotMin)-m1^3/
    log(abs(m1+mdotMin*t1)))));
N3 = @(t) (1/gScale)*(Tmax*(m2^2*t/mdotMax^3-m2*t^2/(2*mdotMax^2)+t^3/(3*mdotMax)-m2^3/mdot
    log(abs(m2+mdotMax*t))-(m2^2*t2/mdotMax^3-m2*t2^2/(2*mdotMax^2)+t2^3/(3*mdotMax)-m2^3/m
    log(abs(m2+mdotMax*t2)))));
end

function[eta1,eta1x,eta1y,eta1z]=getEta1()

eta1 = @(y) (y(4:6)'*y(1:3))/norm(y(4:6))^3*y(4:6) - y(1:3)/norm(y(4:6));
eta1x = @(y) (y(4:6)'*y(1:3))/norm(y(4:6))^3*y(4) - y(1)/norm(y(4:6));
eta1y = @(y) (y(4:6)'*y(1:3))/norm(y(4:6))^3*y(5) - y(2)/norm(y(4:6));
eta1z = @(y) (y(4:6)'*y(1:3))/norm(y(4:6))^3*y(6) - y(3)/norm(y(4:6));

end

function[eta2,eta2x,eta2y,eta2z] = getEta2(eta1,eta1x,eta1y,eta1z)

eta2 = @(y) (2*(y(4:6)'*y(1:3))^2/norm(y(4:6))^5-norm(y(1:3))^2/norm(y(4:6))^3)*...
    y(4:6) + y(4:6)'*y(1:3)/norm(y(4:6))^2*eta1(y);
eta2x = @(y) (2*(y(4:6)'*y(1:3))^2/norm(y(4:6))^5-norm(y(1:3))^2/norm(y(4:6))^3)*...
    y(4) + y(4:6)'*y(1:3)/norm(y(4:6))^2*eta1x(y);
eta2y = @(y) (2*(y(4:6)'*y(1:3))^2/norm(y(4:6))^5-norm(y(1:3))^2/norm(y(4:6))^3)*...
    y(5) + y(4:6)'*y(1:3)/norm(y(4:6))^2*eta1y(y);
eta2z = @(y) (2*(y(4:6)'*y(1:3))^2/norm(y(4:6))^5-norm(y(1:3))^2/norm(y(4:6))^3)*...
    y(6) + y(4:6)'*y(1:3)/norm(y(4:6))^2*eta1z(y);

end
```

```
function [one_pV0,one_pV0x,one_pV0y,one_pV0z] = getOnePV0()

one_pV0 = @(y) y(4:6)/norm(y(4:6));
one_pV0x = @(y) y(4)/norm(y(4:6));
one_pV0y = @(y) y(5)/norm(y(4:6));
one_pV0z = @(y) y(6)/norm(y(4:6));

end

function[Vtf] = getVtf(p,pV0,K1,K2,K3,L1,L2,L3,M1,M2,M3,eta1x,eta1y,eta1z,...
    eta2x,eta2y,eta2z,one_pV0x,one_pV0y,one_pV0z)

[RScale,VScale,gScale,tScale,M0,r0,V0,Tmax,Tmin,m0,alpha,mdotMax,mdotMin,g0,t1,t2,m1,m2]...
    = getVars(p);
Vtf = @(x) [
    V0(1)+one_pV0x(x)*(K1+K2+K3(x(7)))+eta1x(x)*(L1+L2+L3(x(7)))+eta2x(x)*(M1+M2+M3(x(7)))
    V0(2)+one_pV0y(x)*(K1+K2+K3(x(7)))+eta1y(x)*(L1+L2+L3(x(7)))+eta2y(x)*(M1+M2+M3(x(7)))
    V0(3)+one_pV0z(x)*(K1+K2+K3(x(7)))+eta1z(x)*(L1+L2+L3(x(7)))+eta2z(x)*(M1+M2+M3(x(7)))
    g0*x(7);%was g0/gScale
    ];

end

function[f] = getF(p,pr0,pV0,K1,K2,K3,L1,L2,L3,M1,M2,M3,N1,N2,N3,eta1,eta1x,eta1y,eta1z,eta
    eta2x,eta2y,eta2z,one_pV0,one_pV0x,one_pV0y,one_pV0z,Vtf)

[RScale,VScale,gScale,tScale,M0,r0,V0,Tmax,Tmin,m0,alpha,mdotMax,mdotMin,g0,t1,t2,m1,m2]...
    = getVars(p);

f = @(x)[
    r0(1)+x(7)*V0(1)+...
    one_pV0x(x)*(x(7)*(K1+K2+K3(x(7)))-(L1+L2+L3(x(7))))+...
    eta1x(x)*(x(7)*(L1+L2+L3(x(7)))-(M1+M2+M3(x(7))))+...
```

```
    eta2x(x)*(x(7)*(M1+M2+M3(x(7)))-(N1+N2+N3(x(7))));

    r0(2)+x(7)*V0(2)+...
    one_pV0y(x)*(x(7)*(K1+K2+K3(x(7)))-(L1+L2+L3(x(7))))+...
    eta1y(x)*(x(7)*(L1+L2+L3(x(7)))-(M1+M2+M3(x(7))))+...
    eta2y(x)*(x(7)*(M1+M2+M3(x(7)))-(N1+N2+N3(x(7))));

    r0(3)+x(7)*V0(3)+...
    one_pV0z(x)*(x(7)*(K1+K2+K3(x(7)))-(L1+L2+L3(x(7))))+...
    eta1z(x)*(x(7)*(L1+L2+L3(x(7)))-(M1+M2+M3(x(7))))+...
    eta2z(x)*(x(7)*(M1+M2+M3(x(7)))-(N1+N2+N3(x(7))))...
    %-1/2*g0*x(7)^2;%was g0/gScale
    +(1/gScale)*(g0*t1^2+g0*t2^2+1/2*g0*x(7)^2-g0*t1*t2-g0*x(7)*t2);

    V0(1)+one_pV0x(x)*(K1+K2+K3(x(7)))+eta1x(x)*(L1+L2+L3(x(7)))+eta2x(x)*(M1+M2+M3(x(7)))
    V0(2)+one_pV0y(x)*(K1+K2+K3(x(7)))+eta1y(x)*(L1+L2+L3(x(7)))+eta2y(x)*(M1+M2+M3(x(7)))
    V0(3)+one_pV0z(x)*(K1+K2+K3(x(7)))+eta1z(x)*(L1+L2+L3(x(7)))+eta2z(x)*(M1+M2+M3(x(7)))+
    g0/gScale*x(7);%was g0/gScale

%Hamiltonian: Should be correct
     x(1:3)'*Vtf(x)+(-x(1:3)*x(7)+x(4:6))'*([0;0;g0]+Tmax/...%was g0/gScale
     (gScale*(m0+mdotMax*t1+mdotMin*(t2-t1)+mdotMax*(x(7)-t2)))*...
     (one_pV0(x)+eta1(x)*x(7)+eta2(x)*x(7)^2))-alpha*Tmax*sqrt(RScale/gScale);

%Hamiltonian: With better scaling
%       x(1:3)'*Vtf(x)+(-x(1:3)*x(7)+x(4:6))'*([0;0;g0]+Tmax/...%was g0/gScale
%       (gScale*(m0+mdotMax*t1+mdotMin*(t2-t1)+mdotMax*(x(7)-t2)))*...
%       (one_pV0(x)+eta1(x)*x(7)+eta2(x)*x(7)^2))-1;

];

end
```

```
function [ Df ] = getJacobian( f,x,dt )
% This function approximates the derivative of the (anonymous) function 'f'
% at the point 'x'. Tolerance 'dt' is optional.

%This is Taylor's code. My notes:
%x and f are columns
%use elements of x like x(1), x(2)... to define f


if nargin < 3
    dt = 1e-4;
end

n       = length(x);
[m,p]   = size(f(x));

if( (m>1) && (p>1) && (n>1) )

    Df   = zeros(m,p,n);
    Erows    = eye(m);
    Ecols    = eye(p);
    Ex       = eye(n);


    for k = 1:n
        for i = 1:m
            for j = 1:p
                Df(i,j,k) = (1/(2*dt)) * Ecols(:,j)' * ...
                 (f(x + dt.*Ex(:,k)) - f(x - dt.*Ex(:,k)))' * Erows(:,i);
            end
        end
    end
```

```
else

    Df      = zeros(m,n);
    E       = eye(m);
    Ex      = eye(n);

    for i = 1:m
        for j = 1:n
            Df(i,j) = (1/(2*dt))*(f(x + dt.*Ex(:,j))'*E(:,i) - f(x - dt.*Ex(:,j))'*E(:,i))
        end
    end

end



end
```