AUTOMATING INTERPLANETARY TRAJECTORY GENERATION FOR ELECTRIC PROPULSION TRADE STUDIES

by

Prashant R. Patel

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2008

This thesis is dedicated to Elena Spatoulas for showing me that the beauty in life is in the things we do everyday. I love you.

ACKNOWLEDGEMENTS

I would like to thank the NASA GSRP program and Celeste Satter for providing funding for this research and allowing me to pursue a few tangents. I would like to thank my research advisors Daniel Scheeres, Thomas Zurbuchen, and Alec Gallimore for giving me the freedom to try, fail, and succeed. Alec I want to thank you for asking me to take this journey, as I did not believe this was the path for me. Thomas, I want to thank you for showing me that research is more than a set of equations, its value is the story it tells. Dan thank you for keeping me on track and forcing me to document my work. Without you I'm sure I would never have gotten to this point. Dan you have shown me the beauty and elegance of math and how a solid theory does more then a bunch of pretty pictures. Mom, dad, and sis you have supported me and let me go down my own road, for which I can never thank you enough. Bill I need to thank you for keeping me sane and continuously motivating me, you have been a great friend.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

α_{power}	Power system mass divided by output power.
α_{ppu}	Power processing unit mass divided by input power.
$\alpha_{thruster}$	Thruster mass divided by input power.
ΔV	Impulsive velocity change
$\dot{m}$	mass flow rate
η	Total thruster system efficiency
η_{cap}	Maximum efficiency attainable by thruster
ϕ_i	Thruster control state, bounded between -1 and 1
$\boldsymbol{C}$	Constraint set
$\boldsymbol{F}$	Dynamics
$\boldsymbol{p}$	parameters, constant over trajectory
$\boldsymbol{T}$	Thrust vector
$\boldsymbol{U}$	Control set
$\boldsymbol{x}$	State variables, dynamic over trajectory
ξ	First propellant ionization specific energy
$c_{i,j}$	coefficient for the i^{th} positional coordinate and the j^{th} Chebyshev polynomial
$F(C_3)$	Mass delivered to orbit by the launch vehicle
g_0	gravitational constant 9.8 m/s
I_{sp}	specific impulse
J	Cost function
k_i	Ionization factor cost
k_s	Structural mass fraction. Mass of structure divided by initial spacecraft mass.
k_T	Tank mass fraction. Mass of propellant tank divided by propellant mass.
m	mass, subscript denotes the object that is being referenced.
m^*	Normalized mass delivered to escape orbit by launch vehicle
n	inverse mass, $1/m$
P_{elec}	Power input into thruster system
P_{jet}	Jet power, amount of power that produces thrust
Pd	Orbital period of departure and destination body
q	constraints on a trajectory
s	Thruster state on or off, 1 or 0
T_j	j^{th} Chebyshev polynomial

u_e	effective exhaust velocity
V	Cost to go function

LIST OF APPENDICES

CHAPTER I

INTRODUCTION

1.1 Problem Statement

The goal of this work is to provide methods that automate the generation of feasible interplanetary trajectories for use in electric propulsion trades studies. Automation enables engineers to focus on the design of the system and allows non trajectory specialists to create solutions. The output of the automated methods are feasible trajectories which solves the problem of interest. While they are not necessarily the best solution, they place an upper bound on the cost and are good initial guesses for optimizers. Evaluating different technological options to identify the best option or trend is called a trade study. Trade studies benefit the designers because they quantify the cost of using a set of systems. Electric propulsion is used because it increases the capability of spacecraft. With the launch of the Dawn mission electric propulsion will demonstrate its potential. The multi asteroid tour will require an approximate ΔV of 11 $\frac{km}{s}$ which was previously unobtainable with chemical propulsion. In comparison, the Cassini spacecraft, a chemical mission, was only able to provide a ΔV of 2.4 $\frac{km}{s}$. The Dawn spacecraft is able to provide a mission capability four times greater than one of the largest chemically propelled interplanetary missions for the same mass ratio.

1.2 Challenges

Automating the generation of feasible interplanetary trajectories that utilize electric propulsion is challenging due to the coupling between the propulsion, power system, and trajectory. For electric propulsion the increased performance is traded for longer burn times and low thrust to weight ratios. The larger burn times and low thrust to weight ratios require designing trajectories that are fundamentally different from those previously used for chemical based missions. Because chemical propulsion systems have high thrust to weight ratios and low burn times the thrusting periods can be modeled as instantaneous changes in the velocity vector. This results in trajectories that follow conic sections in the two body problem (Keplarian orbits). Conic sections shown in Fig. 1.1, are relatively simple shapes that govern the motion of chemically propelled spacecraft. Determining the velocity change that connects the departure orbit and destination orbit with a Keplarian orbit is known as Lambert's problem. These trajectories can be generated using a Lambert Solver, which is documented in many astrodynamical texts. Furthermore, because the thruster performance is not required in Lambert's problem the thruster used for the mission can be determined after the trajectory is designed.

Due to the low thrust to weight ratio when using electric propulsion, each thrust segment only affects an electric propulsion orbit slightly, causing the orbit to morph over time, requiring that the entire trajectory be modeled. This requires that the electric propulsion trajectories have a large number of control segments and long burn times in order to offset the low thrust to weight ratio. Fig. 1.2 provides two views of an electric propulsion trajectory. Furthermore, because the thrust constraints are dependent on the specific thruster and power system, the thruster

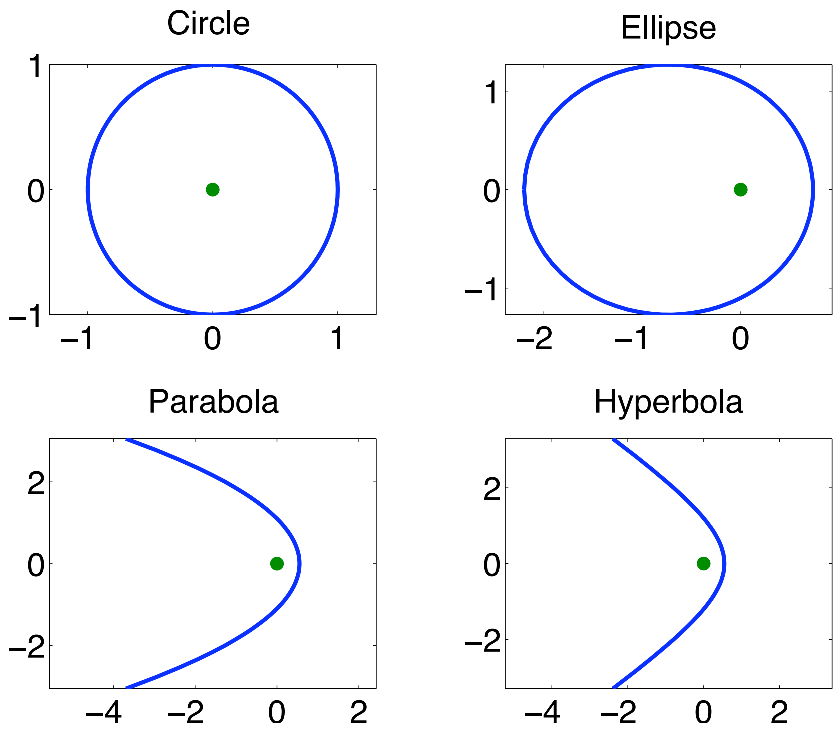


Figure 1.1: The four types of conic sections, circles, ellipses, parabolas, and hyperbolas. All shapes are generated using a single equation that governs the properties of the conic section.

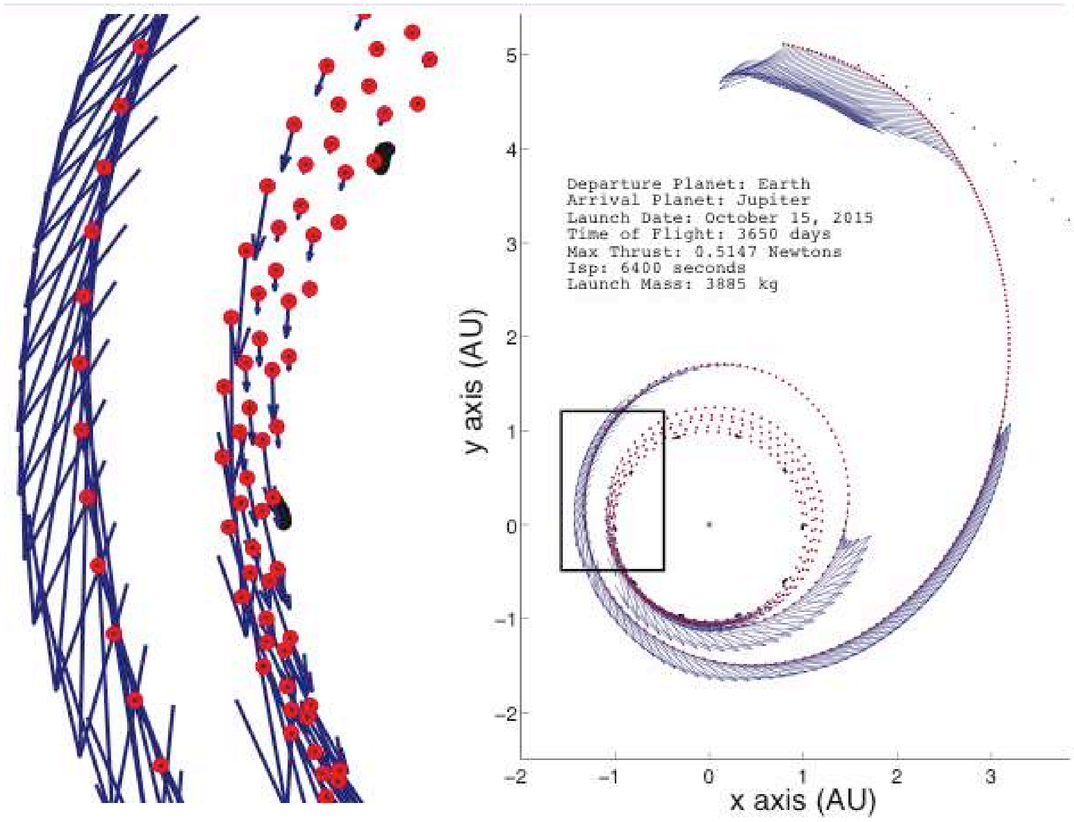


Figure 1.2: An Earth to Jupiter electric propulsion trajectory. The zoomed in portion on the left shows that the thrust magnitude and direction can change substantially. The zoomed out view, on the right, shows that these trajectories can go through periods where the trajectory changes very little followed by times where the trajectory changes substantially.

and power system have to be selected a priori. The coupling between the trajectory, thruster, and power system requires the evaluation of various different thruster and power systems to identify the best technological option given the mass, cost, and risk.

1.3 Research Overview

Conventionally, trade studies have been conducted by supplying user generated initial guesses to optimizers. The conventional process to finding trajectories is shown in Fig. 1.3. The major drawback of the conventional approach is that it

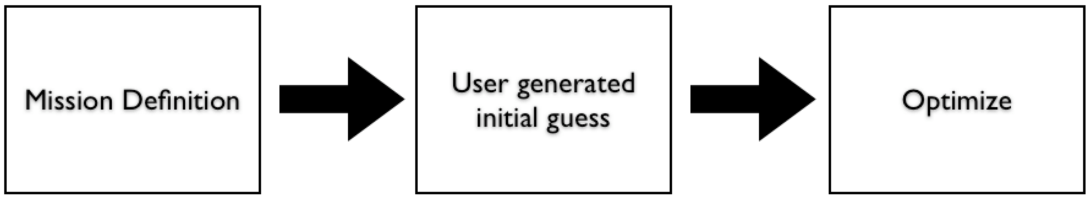


Figure 1.3: The conventional trade study process utilizes a user supplied initial guess to generate optimal trajectories. The speed of convergence is highly dependent on the initial guess that is supplied by the user. This limits the trade space that can be searched.

relies on a user supplied initial guess, the quality of which will vary from problem to problem and user to user. Several attempts have been made to automate the initial guess generation. Russell [54] used a super computer to randomly generate initial guesses for the optimizer. The work was successful but only $\leq 0.1\%$ of the initial guesses lead to successful solutions and required the use of super computer. Petropoulos used the shape based approach to generate solutions to low thrust gravity assist problems[50, 47]. The work was successful in identifying some very complicated trajectories, but because of the exponential sinusoid's limitations it cannot handle rendezvous problems or non-planar trajectories.

The work presented here is designed to provide a front end for optimizers. The methods used here automatically generate trajectories, which can then be used to conduct trade studies. Fig. 1.4 shows the proposed approach for the automating the generation of feasible direct trajectories. The trajectories generated by the process in Fig. 1.4 are feasible solutions so they should be good initial guesses for the optimizer.

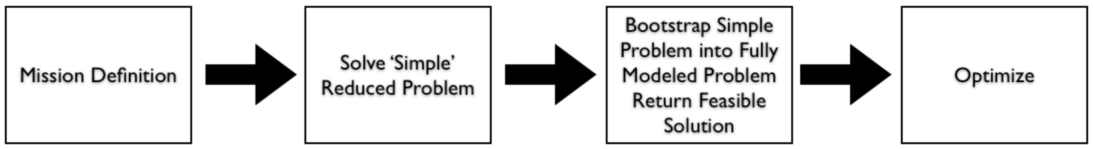

Mission Definition
Solve 'Simple' Reduced Problem
Bootstrap Simple Problem into Fully Modeled Problem Return Feasible Solution
Optimize

Figure 1.4: A two stage process is utilized to generate feasible solutions for optimization. The process works by reducing the feasible trajectory problem into a simpler computationally tractable problem which is then converted into the fully feasible problem. This bootstrap process allows for the use of simple numerical methods.

1.3.1 High Level Trades

High level trade studies use reduced order models to quickly identify optimal mission parameters. Both Kluever[26] and Lorenzo[9, 32, 8] evaluated electric propulsion trajectories using standard efficiency[1] and thrust models. The trajectory, power level, and efficiency are all optimized for a given mission, and the power system mass is assumed to scale linearly with the power generated. In addition, the power system specific power is varied in order to see how the power system impacts the optimal engine and trajectory characteristics. This information can then be used to evaluate the feasibility of a potential mission. High level trades can also be used to evaluate the benefits of proposed thrusters. Lorenzo[8] shows that dual specific impulse thrusters can have similar performance to variable specific impulse thrusters. Reduced order models are useful in evaluating the feasibility of missions and in evaluating the potential impact of new technologies. By returning the optimal power level, specific impulse, and trajectory high level trade studies provide an initial design point.

1.3.2 Low Level Trades

Low level trade studies quantify the relative merits of different components for a particular mission and have become more important with the development of new Hall[19] and Ion thrusters[46] for interplanetary missions. With the ability to choose between different thrusters, it is critical to quantify the benefit of a thruster for a mission. Work by David Oh[42, 41] evaluated different Hall and Ion thrusters for various missions. In particular, Oh evaluated 7 different propulsion configurations and the dual hall thruster configuration had the largest net payload mass. Hofer[19] evaluated the use of a BPT-4000 thruster and NSTAR Ion thruster for an asteroid and comet sample return mission. The use of a the BPT-4000 over the NSTAR in Hofer's study showed a reduction in the mission ΔV by about 2 km/s. These two examples demonstrate the need to evaluate different propulsion system configurations to identify the best option.

Starting with an initial design, low level trades then iterate on the design point until it satisfies the various system and mission constraints. These constraints can include power usage, launch vehicle mass, launch dates, downlink time, pointing requirements, and thermal restrictions. The goal of the process is to produce a solution that satisfies system and mission constraints. Engineering experience, high level trades, previous mission studies, or parametric studies can all be used to generate an initial design point.

1.4 Overview of Thesis

1.4.1 Chapter II

The chapter describes the progression of previous research and provides an overview of key concepts. The chapter begins by discussing the difference between chemical and electric propulsion. Then an overview of Ion and Hall thrusters is provided along with the performance curves for the NSTAR, NEXT, and BPT-4000 thrusters. Also, previous research into high level trades is described and analyzed. Next trajectory optimization and the generation of initial guesses is reviewed. The chapter provides a chronological review of trajectory optimization programs and discusses the shortcomings of the previous work and provides an outline for addressing them.

1.4.2 Chapter III

Chapter III describes the algorithm for automating high level trades. First, reduced order models for the engine, power system, and launch vehicle are directly coupled to the trajectory dynamics through the thrust acceleration term in the equations of motion giving a general model for the acceleration profile. Then variable and constant specific impulse assumptions are applied to the general model acceleration model allowing for some simplification of the equations. For the constant specific impulse model the reduction couples the power system and the propellant mass, indicating that the power and propulsion system should be considered a single system. This analysis extends the work of Jahn[23] by incorporating propellant ionization costs and generalizing the power system. Key parameters relating to the engine, power system, and launch vehicle are then optimized along with the trajectory to maximize the payload mass fraction. An optimization procedure is formulated and coded up in a trade study tool. The tool allows for high level trades to be conducted quickly and autonomously. The algorithm is then applied to an Interstellar Probe vision mission study. A technology trade study is conducted over the power system specific power and the feasible technology envelope is identified.

1.4.3 Chapter IV

Chapter IV focuses on generating feasible trajectories, specifically the chapter describes the formulation and implementation of the first stage. In the first stage the algorithm models the trajectory with a low order polynomial, which is then optimized. The underlying formulation is discussed along with the differences between this formulation and the shape based methods. Next, an overview of the trajectory search and optimization algorithm is described. A proof of concept tool that implements the algorithms and showcases some of the advantages of the method is described. Finally, several example trajectories are shown.

1.4.4 Chapter V

Chapter V focuses on second stage of the feasible trajectory algorithm. First, a description is provided that converts the polynomial trajectories into a thrust sequence based on the user selected engine and launch vehicle. This results in a solution that no longer solves the rendezvous problem. A correction algorithm is defined that uses linear approximations to solve the rendezvous problem. Then the power system is incorporated into the problem. The power system is treated as a constraint on the trajectory and the problem is resolved. Finally, several different trajectories are computed using different thruster and launch vehicle combinations to showcase the utility of the method.

1.4.5 Chapter VI

Chapter VI describes a method for solving optimal control problems. The method uses a second order control update which is constructed to minimize the cost function and satisfy the constraints to the second order. The theory and construction of the algorithm is described and several examples that showcase the performance and benefit of the algorithm are shown along with examples that point out specific problems with the algorithm. A discussion of several critical issues with the algorithm is provided.

1.4.6 Chapter VII

Chapter VII describes the results of the research and avenues for potential future research. The advantages of an automated high level trade study algorithm is described along with the advantages of feasible trajectory generators. Several shortcomings with the current methods are discussed along with potentials solutions that need to be investigated. The benefits of the optimization algorithm is provided along with remaining numerical issues.

CHAPTER II

BACKGROUND

Previously, Chapter I provided an overview of the thesis and discussed three major topics: high level trades, feasible trajectory generators, and optimization. Feasible trajectory generators and optimization are two important steps in automating low level trade studies. This chapter provides background information on the research topic and provides an overview of Hall and Ion thrusters, the primary forms of propulsion considered. Ion and Hall thruster performance curve fits are utilized in the generation of feasible trajectories; specifically, the thrust and mass flow rate profiles are provided for several thrusters. Also, a research overview of high level trade studies is provided as this work focused on optimizing the propulsion system and trajectory. Finally, previous research into trajectory generation and optimization is reviewed.

2.1 Chemical Propulsion

Two different propulsion systems are used for interplanetary exploration, electric propulsion and chemical propulsion. Chemical propulsion missions typically have large thrust to weigh ratios and short burn times. For example, Cassini's thrust to weight ratio at 1 AU is about 27,[1] and the Saturn orbital insertion burn was projected to last about 90 minutes; typical interplanetary missions last on the order of months to years. The large thrust to weight ratio and relatively small burn time of chemical propulsion allows the thrusting periods to be modeled as discrete changes in the velocity. When the engine is off, the trajectory is a conic section (two body problem), also called ballistic trajectories or coast arcs.

Modeling the chemical propulsion system as a series of instantaneous velocity changes reduces the rendezvous problem to finding the velocity vectors that allow the spacecraft to transition from the initial orbit to the final orbit, given a launch date, transit time, departure body, and destination body. The resulting two point boundary value problem is called Lambert's Problem[12, 57]. The velocity vectors that allow a transfer between the desired orbits are the solution to Lambert's problem. Many different Lambert solvers[12, 57] exist to solve Lambert's problem. Once the orbit is computed the mass cost of the transfer can be estimated.

The velocity change provided by an engine in gravity free space is the ΔV, which is an important metric because it is related to the amount of propellant required for a mission. For chemical propulsion, the ΔV is the sum of the L_1 norm of the instantaneous velocity changes. The L_1 norm of a vector $\boldsymbol{V}$ with elements V_i is $|\boldsymbol{V}|_1 = \sqrt{\sum_{i=1}^{N} V_i}$. The ideal rocket equation, Eq. (2.1), provides a relationship between the initial mass, m_0, final mass, m_f, ΔV, and the effective exhaust velocity, u_e.

$$\frac{m_f}{m_0} = e^{-\Delta V/u_e} \tag{2.1}$$

[1]Thrust to weight ratio computed from NASA Cassini fact sheets. http://saturn.jpl.nasa.gov/news/fact-sheets.cfm

The effective exhaust velocity is given by

$$u_e = g_0 I_{sp} = g_0 \frac{\int_{t_0}^{t_f} T d\tau}{\int_{t_0}^{t_f} \dot{m} d\tau} \tag{2.2}$$

where g_0 is a constant equal to 9.8 m/s^2. I_{sp} is an abbreviation for the specific impulse and is proportional to the effective exhaust velocity. The rocket equation demonstrates that higher specific impulses allow for larger ΔV for a fixed mass ratio.

2.2 Electric Propulsion

Electric propulsion as defined by Jahn[23] is

> The acceleration of gases for propulsion by electrical heating and/or by electric and magnetic forces.

Jahn[23] goes further and breaks electric propulsion into three categories, electrothermal, electrostatic, and electromagnetic. The three categories are defined by Jahn[23] as

> 1. Electrothermal propulsion, wherein the propellant gas is heated electrically, then expanded in a suitable nozzle.
> 2. Electrostatic propulsion, wherein the propellant is accelerated by direct application of electric body forces to ionized particles
> 3. Electromagnetic propulsion, wherein an ionized propellant stream is accelerated by interactions of external and internal magnetic fields with electric currents driven through the stream

Electric propulsion has allowed previously unfeasible missions to become reality, due to the higher specific impulse (> 2000 s). DAWN[6] recently began its mission and is using the NSTAR Ion engine; the thrusters are expected to provide about 11 km/s of ΔV, while Cassini generated a total ΔV of about 2.4 km/s[18]. The DAWN spacecraft has a thrust to weight ratio at 1 AU of 0.013 which is much less than Cassini's thrust to weight ratio. The DAWN mission clearly demonstrates the benefit of electric propulsion.

The rocket equation (Eq. (2.1)) determines the propellant mass but does not account for the power system mass. Electric propulsion has large power requirements ($>$1 kW) so the power system can represent a significant fraction of the spacecraft mass budget. For this reason the power system must be taken into account when conducting trade studies.

2.2.1 Hall and Ion Thrusters

Hall and Ion thrusters are the two primary propulsion devices considered in this thesis. Hall and Ion thrusters are electrostatic devices[23, 17]. They have been used on several space exploration missions including Deep Space 1[33, 4, 52], DAWN[6], SMART-1[27, 15], and Hayabusa[29, 30, 28]. DAWN and Hayabusa used ion thrusters while SMART-1 used a Hall thruster.

From Goebel and Katz[17] Ion thrusters are described.

> Ion thrusters employ a variety of plasma generation techniques to ionize a large fraction of the propellant. These thrusters then utilize biased grids to electrostatically extract ions from the plasma and accelerate them to high velocity at voltages up to and exceeding 10 kV. Ion thrusters feature the highest efficiency (from 60% to > 80%) and very high specific impulse (from 2000 to over 10,000 s) compared to other thruster types.

Also, Hall thrusters are described.

> This type of electrostatic thruster utilizes a cross-field discharge described by the Hall effect to generate the plasma. An electric field established perpendicular to an applied magnetic field electrostatically accelerates ions to high exhaust velocities, while the transverse magnetic field inhibits electron motion that would tend to short out the electric field. Hall thruster efficiency and specific impulse is somewhat less than that achievable in ion thrusters, but the thrust at a given power is higher and the device is much simpler and requires fewer power supplies to operate.

The three primary thrusters used in this research are the NSTAR and NEXT Ion thrusters, and the BPT-4000 Hall thruster.

The NSTAR Ion thruster was used in the Deep Space 1 mission[33, 4, 52] and is currently being used on the DAWN mission[33, 4, 52]. The engine has a specific impulse of about 2000-3000 seconds, provides a thrust of 20-92 mN, and uses 0.525-2.6 kW of power[19, 42, 41]. The thrust and mass flow rate are parameterized from the models in references [19], [42], and [41].

The NEXT Ion thruster is able to throttle over 0.620-7.3 kW and has a specific impulse range of 1000-4000 seconds. The thrust range is 102-341 mN. The NEXT thrust and mass flow rate is parameterized from[41].

The BPT-4000 is a Hall thruster which is able to throttle over 1-4.5 kW of power and has a thrust range of 82-250 mN with a specific impulse range of 2000-3000 seconds. The BPT-4000 thrust and mass flow rate is parameterized from[19].

2.3 High Level System Trades

High level system trades of electric propulsion based missions are designed to identifying the important parameters that couple the propulsion (thruster and launch vehicle), power, and trajectory. Some of the earliest work was conducted by Jahn[23] and Marec[34]. Jahn derived the optimal exhaust velocity assuming a known thrusting time and constant efficiency. Jahn also pointed out that for electric propulsion, the power system can have a significant mass cost. Marec analyzed ideal variable specific impulse electric propulsion system with constant efficiency. Auweter-Kurtz[1] took thruster analysis further by analyzing different electric propulsion thrusters, Hall, Ion, MPD, and others, and relating them to the ΔV. The ΔV is a function of the trajectory and Auweter-Kurtz[1] did not optimize the trajectory with the propulsion system.

Another approach is to simply optimize the trajectory and propulsion system model[26, 25, 32, 8]. Work by Kluever[26, 25] and Lorenzo[32, 8] took into account the change in efficiency with different exhaust velocities and optimized the power level, launch vehicle, exhaust velocity, and trajectory.

Kluever[26, 25] directly optimized the propulsion system and trajectory for a lunar and heliospheric exploration missions. The work numerically, mapped out the relationship between the trip time and specific impulse, payload mass fraction, and power level. However, no analytic formulas or reductions were found.

Lorenzo[32, 8] also optimized the trajectory, power, and propulsion system for several missions using variable and constant specific impulse thruster models. For the constant specific impulse thruster model with a constant power source the specific impulse is removed from the equations and is a function of the power level, burn time, and propellant mass. While the relationship between the specific impulse and the power level, burn time, and propellant mass does eliminate one variable, the other three variables have to be found through the optimization process and cannot be easily estimated.

2.4 Low Level System Trades and the Trajectory

Low level trades quantify the relative merits of different components for a particular mission. These types of trade studies have become even more important with the development of new Hall[19] and Ion thrusters[46] for interplanetary missions. The system trades are primarily focused on how different systems affect the projected mass, cost, and risk to a mission[3, 5, 11, 14, 13, 19, 42, 41, 63, 6, 46].

Mission studies focus on evaluating technologies for a particular mission. Some mission studies include a Titan explorer[11], a comet sample return[3, 63], and an Interstellar probe[14]. These studies were focused on evaluating the feasibility of a particular mission.

Other studies focus directly on the technology and on how new technologies can benefit future missions. Work by David Oh[42, 41] evaluated various different Hall and Ion thrusters for different missions. Hofer[19] evaluated the use of a BPT-4000 thruster and NSTAR Ion thruster for and asteroid and comet sample return mission. Brophy[5] evaluated solar electric propulsion for solar system exploration.

The results from Brophy, Oh, and Hofer, show that each mission needs to be evaluated independently and over a wide range of thrusters in order to determine the best system. Also, the studies highlight that mission designers are not just concerned about mass but must take into account the cost, reliability, and mass to decide which system is best. Hofer showed that the BPT-4000 Hall thruster can decrease the cost of a mission and increase the net payload mass. Oh's work showed that varying the thruster type and number of thrusters the payload mass can vary by about 50% and the N^{th} mission costs can vary by 6.5 million dollars. Because the lowest cost mission does not necessarily, have the largest payload mass, it is critical to evaluate the different possible options then weigh the cost and mass trade. Similarly, Brophy[3] analyzed several thruster configurations for a comet sample return mission. The results showed that the thruster configuration with the largest net payload mass was not the best configuration because it is technically risky. Instead, Brophy recommended using a less risky option that delivers slightly less payload mass.

2.4.1 Trajectory Optimization Programs

The trajectory is critical to evaluating thruster performance for mission and technology studies. In order to ascertain how well a thruster will perform the trajectory has to be optimized for the thruster. Once the trajectory has been optimized, the cost can be estimated. Numerous optimization programs have been developed to optimize electric propulsion trajectories.

Low thrust trajectory optimization problems have been posed and solved since the 1960s[37]. CHEBYTOP[24] was developed by Boeing and JPL for NASA. Sauer is one of the earliest researchers that attempted to tackle the problem and developed the JPL version of CHEBYTOP. CHEBYTOP attempted to solve electric propulsion trajectories using collocation and approximating the constant I_{sp} trajectories as "equivalent" variable I_{sp} trajectories. CHEBYTOP was designed as a performance analysis tool and so does not model engine or dynamics particularly accurately, this can cause CHEBYTOP to return erroneous results for trajectories with thrust to weight ratios > 0.001 or for trajectories with significant inclination changes. The benefit of CHEBYTOP is that it can generate solutions quickly.

VARITOP and SEPTOP[51] are trajectory optimization programs that use Pontryagin's minimum principle[7]. Using the minimum principal results in coupled nonlinear two point boundary value problem. SEPTOP was successfully used to plan and execute the trajectory for Deep Space 1[53]. SEPTOP and VARITOP have been successfully used, but their application is limited because they use calculus of variations. The costates in calculus of variations approach are unstable and generally have small radius of convergence. This can lead to problems when a solution cannot be found because the initial guess is not sufficiently close to the optimal solution.

GALLOP[36, 64, 35] and MALTO[47, 55] are two similar trajectory optimization methods. They are generally used in preliminary mission designs. GALLOP was developed at Purdue, while MALTO was produced at JPL. MALTO approximates the low thrust arcs as small ballistic impulses. The ΔV magnitude is dependent on the thruster and the power system. GALLOP and MALTO use SNOPT[16] to conduct the optimization, which uses a modified sequential quadratic programming method (SQP)[39, 16, 2]. Sequential quadratic programming is a popular and standard method for solving nonlinear optimization problems. MALTO and GALLOP require initial guesses that are close to the solution but because they use a direct method the radius of convergence is larger.

Two high fidelity optimizers are Mystic[59, 60, 61, 62] and Copernicus[51, 40]. Mystic uses static dynamic control (SDC)[59, 60, 61, 62], which is derived from differential dynamic programming (DDP)[22, 21, 38, 31]. Mystic is designed to handle multi body forces as well as solar radiation pressure. Copernicus is designed to be a general optimization program that does not depend on any one specific method but instead uses currently available numerical techniques. Mystic has a very large radius of convergence because it uses a penalty method to optimize the trajectory. The penalty method allows optimizers to converge when the initial guess is far from the optimal solution.

Most optimization methods are based around an iterative method for optimizing, where the current solution is used to generate the next solution. Critical to this process is providing an initial guess that starts the iterative process. The initial guess is the trajectory/control sequence that the optimizer begins the optimization process around, usually supplied by the user. A feasible trajectory is defined as a trajectory that satisfies the constraints for a given problem. The constraints are typically thrust limitations, power limitations, launch vehicle constraints, and arrival constraints. Furthermore, feasible trajectories place an upper bound on the propellant cost and serve as a back up solution if for some reason the optimizer wanders away from the optimal solution. If the initial guess is feasible and near the optimal solution, the optimizer should converge quickly.

2.4.2 Trajectory Approximations

Finding optimal or feasible EP trajectories can be difficult due to the numerous control choices and the nonlinear nature of the dynamics. Over past few decades many approximations have been used to model EP trajectories[49]. The most widely used approximation is the shape based approach. The shape based approach models the trajectory as a function of a free parameter, which is not time. The trajectory model or shape is a holonomic constraint on the system, reducing the problem to a single degree of freedom.

Shape based methods are attractive because of their low computation cost, due to the single degree of freedom. In the shape based approach, the shape parameterizes the trajectory in space but not in time. The shape provides the location of the spacecraft but does not provide any information about when it will be there, so the timing over the trajectory has to be found.

The most successful application of the shape based[20, 45, 48, 50] approach is the use of the exponential sinusoid[50, 48, 47]. The exponential sinusoid is given by

$$r\left(\theta\right) = k_0 e^{k_1 sin(k_2\theta+\phi)} \tag{2.3}$$

where r is the heliocentric radius and θ is the polar angle. The exponential sinusoid is primarily used to identify low thrust gravity assist trajectories. This approach was successfully applied to the ACT 1 competition[47]. In general, the shape based approach can handle any set of state boundary constraints, assuming the shape selected has the appropriate degrees of freedom. The limitation with the shape based approach is that for an arbitrary shape the acceleration profile may be undesirable. This arises because the shape is a holonomic constraint (fixed in space but not time). In order to constrain the acceleration profile, the shape profile would also have to be freely varied. A further limitation of the exponential sinusoid is that it was not designed to handle non-planer trajectories or to explicitly take into account engine and power system limitations.

2.5 Conclusions

Work by Kluever and Lorenzo shows that optimization of trajectory, power, and propulsion system for high level studies is beneficial. Their work was limited because no general method for solving the optimization problem was provided, and the equations were not reduced to allow for scaling of the systems. Chapter III combines the trajectory optimization with the propulsion system and the payload mass is nondimensionalized allowing the system to scale. For the constant specific impulse thruster, the optimal exhaust velocity is found to be a function of the several technological parameters and the throttle time history. Previously, the exhaust velocity was a function of the power level, the propellant mass, and the burn time[32]. Also, Automating the optimization process allows for high level trades to be rapidly computed over a wide range of technology parameters.

The exponential sinusoid has been successfully applied to numerous problems, however the trajectories it generates are not feasible solutions. Furthermore, due to the limited degrees of freedom in the shape, the path will not necessarily solve the rendezvous problem. This points to the need for a new method that can maintain the advantage of the shape method (low computational cost), but can handle the rendezvous problem and generate feasible solutions. Automating the process allows for quick computation of the trajectory while satisfying the thrust and power system limitations. Chapter IV presents a new method that can handle general boundary conditions and allows the trajectory to be optimized while incurring a low computational cost. Chapter V uses the trajectory generated from Chapter IV and introduces the launch vehicle, thruster, and power system to generate a feasible trajectory.

Numerous trajectory optimization algorithms have been developed and tested. Direct methods that have been previously developed use linear control updates and only satisfy the constraints to the first order. A new algorithm is derived in Chapter VI that uses a second order control update that satisfies the constraints to the second order. The method is designed to keep the trajectory feasible while optimizing.

CHAPTER III

HIGH LEVEL SYSTEM TRADE STUDY ALGORITHM FOR INTERPLANETARY ELECTRIC PROPULSION MISSIONS

3.1 Introduction

Chapter II provided an overview of the several electric propulsion thruster and trajectory optimization programs and discussed the nature and importance of high and low level trade studies. Previous research into high level trades has analyzed some of the coupling between the propulsion system and the power system, however the relationships between the propulsion, launch vehicle, and power system were neither fully exploited nor explored. Analyzing the system coupling is important because it can provide scaling laws and insight without the need for solving difficult problems. Furthermore, treating the entire coupled problem allows for mass savings that would not otherwise be apparent. In order to handle a wide class of electric propulsion thrusters, several assumptions have to be made that reduce the accuracy but increase the utility of the analysis. The reduced order models capture the general scaling and trends of thrusters, but do not replicate the performance of a particular thruster. General models that incorporate physical and design parameters are used to model the efficiency and exhaust velocity relationship. Next, the models are used to derive specific relationships between the variables of interest, also the cost function and scaling parameters are nondimensionalized, which allows the systems to scale. The relationships are then applied to two hypothetical missions.

The chapter begins by outlining the models that characterize the launch vehicle, dynamics, power, propellant, and propulsion system. These models are critical to the analysis and optimization because they determine the level of detail captured and the difficulty of the optimization process. For high level trades, low order models are used to capture the major trends and identify the optimal solution quickly. The reduced order models extend the work by Jahn[23] and provide correction terms which take into account that efficiency of the thruster varies depending on the specific impulse and propellant utilized. The models are then used in an optimization program which conducts trade studies over a range of power system mass to power ratios.

3.2 System Models

Electric propulsion allows for propellant mass savings over chemical propulsion systems due to the increased exhaust velocity. However, electric propulsion requires a larger power system, detracting from the potential mass savings. The proper balance between the power level and the propellant mass must be found in order to ensure the feasibility of the mission. Along with the power system the other important systems should also be taken into account. For instance, the launch vehicle determines the initial orbit of the spacecraft and the initial mass of the spacecraft, both of which significantly impact the trajectory and final mass of the spacecraft. It is critical to find the correct balance between the propulsion system, power system, and launch vehicle in order to maximize the payload mass available for science. The payload mass is defined as the mass not devoted to the launch vehicle, structure, power system, or propulsion system.

3.2.1 Dynamics

The electric propulsion thrusters affect the motion of the spacecraft through the thrust term, $\boldsymbol{T}$, in the equations of motion (Eq. (3.1) and Eq. (3.2)).

$$\ddot{\vec{r}} = -\frac{\mu_{sun}}{|\boldsymbol{r}|^3}\boldsymbol{r} + \frac{\boldsymbol{T}}{m} \tag{3.1}$$

and

$$\dot{m} = \frac{|\boldsymbol{T}|}{u_e} \tag{3.2}$$

where $\boldsymbol{r}$ is the positional coordinates, and m is the mass of the spacecraft. Simple dynamics are utilized for high level trades because they reduce computation time and simplify the problem.

3.2.2 Launch Vehicle

The launch vehicle constrains the mass that can be delivered to a specified orbit and determines the initial orbit that the spacecraft is launched into. For interplanetary missions, the mass is a parameterized as a function of twice the specific orbital energy, commonly called C_3, which can be computed using

$$C_3 = \left(\boldsymbol{V}_{s/c} - \boldsymbol{V}_{earth}\right)^T \left(\boldsymbol{V}_{s/c} - \boldsymbol{V}_{earth}\right) \tag{3.3}$$

Here $\boldsymbol{V}_{s/c}$ is the escape velocity of the spacecraft at launch, and $\boldsymbol{V}_{earth}$ is the velocity of Earth. The initial mass of the spacecraft is a function of the launch vehicle and the C_3. The relationship is functionally modeled as

$$m_{Launch} = F\left(C_3\right) \tag{3.4}$$

where m_{Launch} is the launch mass, and F is a predefined parameterization of the launch vehicle. A normalized version of F is defined as

$$m^* = f(C_3) = \frac{F(C_3)}{m_0} \tag{3.5}$$

where

$$m_0 = F(0) \tag{3.6}$$

The normalized function f allows the launch vehicle to scale up or down in mass. The normalized launch vehicle model is assumed to be a known function that places an upper bound on the normalized mass, m^*, that can be delivered to the destination. The mass fraction of several launch vehicles is given in Fig. 3.1. The spacecraft structural mass scales with the initial mass and is

$$m_{struct} = k_s m_{Launch} \tag{3.7}$$

Here k_s is the structural coefficient, which is a dimensionless number that determines how quickly the mass of the structure scales with the mass of the spacecraft. The structural coefficient is bounded between 0 and 1.

3.2.3 Propulsion System Model

A simple reduced order thruster model[26, 32, 8] is utilized to model the propulsion system. The reduced order models links the power generated by the power system to the thrust delivered by the engine. The total efficiency of the power processing unit and thruster, η, is modeled as

$$\eta = \frac{P_{jet}}{P_{elec}} = \frac{\eta_{cap}}{1 + \frac{2k_i\xi}{u_e^2}} \tag{3.8}$$

where η_{cap} represents the maximum possible efficiency, ξ is the first specific ionization energy of the propellant. The parameter k_i is a dimensionless factor that is

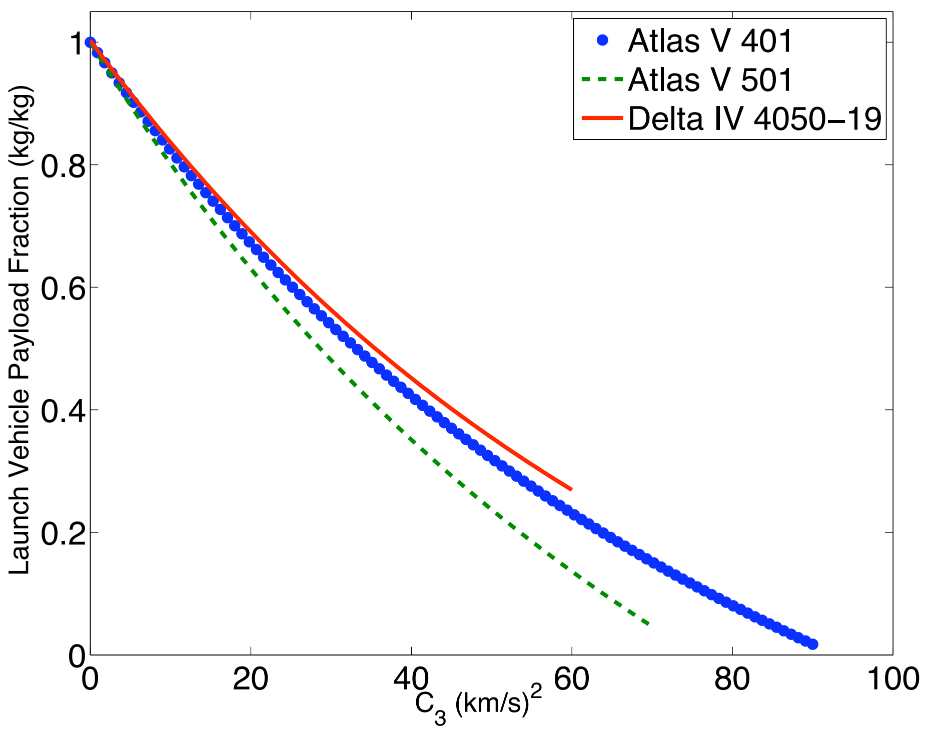


Figure 3.1: Several launch vehicle's delivered mass fraction, $f\left(C_3\right)$, as a function of C_3. As C_3 increases the orbital energy increases, but the delivered mass fraction delivered to space decreases.

introduced so the ionization cost, ξ, can be increased due to other loss mechanisms. The electrical power provided by the power system is P_{elec}, and the jet power, P_{jet}, is the power that results in thrust and is given by

$$P_{jet} = \frac{1}{2} T u_e \tag{3.9}$$

Eq. (3.8) can be rewritten to obtain the jet power

$$P_{jet} = \eta P_{elec} = \frac{\eta_{cap} P_{elec}}{1 + \frac{2k_i \xi}{u_e^2}} \tag{3.10}$$

Eq. (3.8) is commonly written as[26]

$$\eta = \frac{b u_e^2}{u_e^2 + d^2} \tag{3.11}$$

The two equations are equivalent if $b = \eta_{cap}$ and $d = \sqrt{2k_i \xi}$. Eq. (3.8) is used in this analysis because it utilizes physical parameters that can be easily estimated. The thrust provided by the propulsion system is

$$T(t, \boldsymbol{r}) = \begin{cases} k_1(t)\, k_2(t, \boldsymbol{r})\, T_0 \\ k_1(t)\, k_2(t, \boldsymbol{r})\, \frac{2P_{jet}}{u_e} \\ k_1(t)\, k_2(t, \boldsymbol{r})\, \frac{2\eta_{cap} P_{elec}}{u_e \left(1 + \frac{2k_i \xi}{u_e^2}\right)} \end{cases} \tag{3.12}$$

where $T_0 = \frac{2P_{jet}}{u_e}$, and $k_1(t)$ is the control that varies between

$$0 \leq k_1(t) \leq 1 \tag{3.13}$$

Here $k_2(t, \boldsymbol{r})$ regulates the amount of power available over time and space. The requirement on k_2 is

$$k_2(t_0, \boldsymbol{r}_0) = 1 \tag{3.14}$$

With this model the mass flow rate is

$$\dot{m} = -k_1(t)\, k_2(t, \boldsymbol{r})\, \frac{T_0}{u_e} \tag{3.15}$$

The mass of the propulsion system is a function of the propellant mass, tank mass, and thruster and power processing unit mass. The propellant mass is a function of the total propellant consumed.

$$m_{propellant} = \begin{cases} \int_{t_0}^{t_f} k_1(t)\, k_2(t, \boldsymbol{r})\, \frac{T_0}{u_e} d\tau \\ \int_{t_0}^{t_f} k_1(t)\, k_2(t, \boldsymbol{r})\, \frac{2P_{jet}}{u_e^2} d\tau \\ \int_{t_0}^{t_f} k_1(t)\, k_2(t, \boldsymbol{r})\, \frac{2\eta_{cap} P_{elec}}{u_e^2 + 2k_i \xi} d\tau \end{cases} \tag{3.16}$$

For convenience the following states are defined

$$Z(t) = \int_{t_0}^{t} k_1(t)\, k_2(t, \boldsymbol{r})\, d\tau \tag{3.17}$$

and

$$\dot{Z} = k_1(t)\, k_2(t, \boldsymbol{r}) \tag{3.18}$$

For convenience Z_f is equal to $Z(t_f)$. The tank mass is assumed to scale with the propellant mass and is

$$m_{tank} = k_T m_{propellant} \tag{3.19}$$

where k_T is the tank fraction, a dimensionless number that determines how the tank mass will scale with the propellant mass. The thruster mass and power processing unit mass scales linearly with the electrical power generated by the power system. The mass of the thruster and power processing unit is

$$m_{thruster+ppu} = \alpha_{thruster+ppu} P_{elec} \tag{3.20}$$

where $\alpha_{thruster+ppu}$ is the specific power of the thruster and power processing unit and P_{elec} is the electrical power generated by the power system.

3.2.4 Power System

The total electrical power produced by the power system at launch is P_{elec}. The mass of the power system scales linearly with the electrical power and is

$$m_{power} = \alpha_{power} P_{elec} \tag{3.21}$$

The electrical power produced by the power system is assumed to be only a function of the time and position of the spacecraft and is

$$P\left(t, \boldsymbol{r}\right) = k_2\left(t, \boldsymbol{r}\right) P_{elec} \tag{3.22}$$

where k_2 is a function that defines how the available power varies with time and position. For example, the power produced by solar arrays vary proportionally to $\frac{1}{r^2}$ because the solar flux decreases as $\frac{1}{r^2}$. Radioisotope thermal generators on the other hand use radioactive materials to generate power and the power output is dependent on the amount of radioactive material remaining so the current power level is a time dependent processes that can be modeled by k_2.

With the models for the thrusters, launch vehicle, and power system defined, different classes of thrusters can be analyzed. Specifically, two thruster types are considered, variable specific impulse and constant specific impulse thrusters. Variable specific impulse thrusters can change the specific impulse arbitrarily while constant specific impulse thrusters operate at a fixed specific impulse over the entire mission. Hall and Ion thrusters can vary the exhaust velocity over a limited range, and for the purposes of this analysis they are considered constant specific impulse thrusters because the throttle range is small.

3.3 Bounded Power Variable Specific Impulse

An ideal variable-specific impulse thruster can vary the specific impulse over the range $0 \leq u_e \leq \infty$, causing the thrust and mass flow rate to change. This analysis begins by attempting to reduce the cost function. The cost function used to evaluate different missions and technologies is the payload mass fraction, J_1, which is the mass not devoted to the power system, propellant, structure, propellant tanks, and launch vehicle, normalized by maximum launch vehicle payload capacity, m_0.

$$\begin{aligned} J_1 &= \underbrace{\frac{m_f}{m_0}}_{final\ mass\ ratio} - \underbrace{k_s f\left(C_3\right)}_{structure\ mass\ fraction} \\ &- \underbrace{k_T\left(f\left(C_3\right) - \frac{m_f}{m_0}\right)}_{tank\ mass\ fraction} - \underbrace{\frac{\alpha P_{elec}}{m_0}}_{power/propulson\ system\ mass\ fraction} \end{aligned} \tag{3.23}$$

where

$$\alpha = \alpha_{thruster+ppu} + \alpha_{power} \tag{3.24}$$

Rearranging J_1

$$J_1 = \frac{m_f + k_T m_f - \alpha P_{elec}}{m_0} - f\left(C_3\right)\left(k_s + k_T\right) \tag{3.25}$$

$$J_1 = f\left(C_3\right)\left(\frac{m_f\left(1 + k_T\right)}{m_0 f\left(C_3\right)} - \frac{\alpha P_{elec}}{m_0 f\left(C_3\right)} - \left(k_s + k_T\right)\right) \tag{3.26}$$

$$J_1 = f\left(C_3\right)\left(\psi m_f^* - \Gamma - \phi\right) \tag{3.27}$$

where

$$m_f^* = \frac{m_f}{m_0 f\left(C_3\right)} = \frac{spacecraft\ final\ mass}{spacecraft\ initial\ mass} \tag{3.28}$$

$$\Gamma = \frac{\alpha P_{elec}}{m_0 f\left(C_3\right)} = \frac{mass\ of\ power\ system\ ,\ thrusters\ ,\ and\ PPU}{spacecraft\ initial\ mass} \tag{3.29}$$

$$\phi = k_T + k_S \tag{3.30}$$

$$\psi = 1 + k_T \tag{3.31}$$

The payload mass fraction is now reduced to a function of two dimensionless variables, the final spacecraft mass fraction and the spacecraft power. Because the specific impulse is variable and unbounded, the throttle function, k_1, is not required and is eliminated from the equations. In Ref. [43] the constant efficiency model allows for an unbounded thrust, however including the variable efficiency model (Eq. (3.8)) results in a maximum thrust of

$$T_{max} = k_2\left(t, \boldsymbol{r}\right)\frac{\eta_{cap}P_{elec}}{\sqrt{2k_i\xi}} \tag{3.32}$$

Unfortunately, coupling the specific impulse and efficiency destroys the simplifications present when using a constant efficiency model[43]. With a variable efficiency model, the maximum thrust is now finite, and for a valid thrust level there are two possible specific impulses. The highest specific impulse is always utilized since it minimizes the mass flow rate.

3.4 Bounded Power Constant Specific Impulse

Constant specific impulse refers to the fact that u_e does not change over the mission. Thus the constant specific impulse case is special case of the variable specific impulse impulse case, implying that the variable specific impulse case will always perform equal to or better than the constant specific impulse case.

For the constant specific impulse case the payload mass fraction is written as

$$\begin{aligned} J_1 = & \underbrace{f\left(C_3\right)}_{Launch\ mass\ fraction} & - & \underbrace{k_s f\left(C_3\right)}_{structural\ mass\ fraction} \\ - & \underbrace{\frac{m_{prop}}{m_0}}_{propellant\ mass\ fraction} & - & \underbrace{k_T\frac{m_{prop}}{m_0}}_{propellant\ tank\ mass\ fraction} \\ - & \underbrace{\alpha\frac{P_{elec}}{m_0}}_{power/propulsion\ system\ mass\ fraction} \end{aligned} \tag{3.33}$$

Collecting terms results in

$$J_1 = f\left(C_3\right)\left(1-k_s\right) - \left(1+k_T\right)\frac{m_{propellant}}{m_0} - \alpha\frac{P_{elec}}{m_0} \tag{3.34}$$

Defining

$$\psi = 1 + k_T \tag{3.35}$$

and

$$\zeta = 1 - k_s \tag{3.36}$$

gives

$$J_1 = f\left(C_3\right)\zeta - \psi\frac{m_{propellant}}{m_0} - \alpha\frac{P_{elec}}{m_0} \tag{3.37}$$

Substituting in Eq. (3.12) and Eq. (3.16) gives

$$J_1 = f\left(C_3\right)\zeta - \psi Z_f\frac{T_0}{u_e m_0} - \alpha\frac{1}{2}\frac{T_0}{\eta_{cap} m_0}\left(1 + \frac{2k_i\xi}{u_e^2}\right)u_e \tag{3.38}$$

Differentiating J_1 with respect to u_e to find the optimal exhaust velocity results in

$$\frac{\partial J_1}{\partial u_e} = 0 = \psi Z_f\frac{T_0}{m_0 u_e^2} - \alpha\frac{1}{2}\frac{T_0}{\eta_{cap} m_0}\left(1 - \frac{2k_i\xi}{u_e^2}\right) \tag{3.39}$$

Solving for the optimal exhaust velocity yields

$$u_{e\,optimal} = \sqrt{2\left(\frac{\psi Z_f \eta_{cap}}{\alpha} + k_i\xi\right)} \tag{3.40}$$

Previously, the optimal exhaust velocity was given as[32]

$$u_e = \sqrt{\frac{2P_{elec}\tau}{m_{propellant}}} \tag{3.41}$$

From Ref. [32] P_{elec} is the electrical power used by the propulsion system, and $m_{propellant}$ is the propellant mass, and τ is the burn time of the engine. The critical difference between Eq. (3.41) and Eq. (3.40) is that Eq. (3.40) does not state the result in terms of the mass or power level, which have to optimized. Instead, Eq.

(3.40) shows that the optimal exhaust velocity does not depend on the power level or propellant mass, but does depend on the propellant, power system, and time history of the throttle profile through Z_f. This result is significant because all the parameters in Eq. (3.40) except for Z_f are known before the optimization problem is solved.

The payload mass fraction, J_1, can be restated in terms of the jet power as

$$J_1 = f\left(C_3\right)\zeta - \psi Z_f \frac{2P_{jet}}{m_0 u_{e\ optimal}^2} - \alpha \frac{P_{jet}}{m_0 \eta_{cap}} \left(1 + \frac{2k_i \xi}{u_{e\ optimal}^2}\right) \tag{3.42}$$

Combining terms

$$J_1 = f\left(C_3\right)\zeta - 2\frac{P_{jet}}{m_0} \left(\frac{\alpha}{2\eta_{cap}} + \frac{\psi Z_f + \frac{\alpha k_i \xi}{\eta_{cap}}}{u_{e\ optimal}^2}\right) \tag{3.43}$$

Using Eq. (3.40) to eliminate the optimal exhaust velocity gives

$$J_1 = f\left(C_3\right)\zeta - 2\frac{P_{jet}\alpha}{m_0 \eta_{cap}} \tag{3.44}$$

Defining

$$\Omega = \frac{P_{jet}\alpha}{m_0 f\left(C_3\right)\eta_{cap}} \tag{3.45}$$

J_1 then becomes

$$J_1 = f\left(C_3\right)\left(\zeta - 2\Omega\right) \tag{3.46}$$

Ω represents the ideal mass fraction of the power and propulsion system if the efficiency was constant[43] and equal to η_{cap}. The propulsion and power system mass fraction is

$$\frac{\alpha P_{elec}}{m_0 f\left(C_3\right)} = \Gamma = \Omega \left(1 + \frac{2k_i \xi}{u_{e\ optimal}^2}\right) \tag{3.47}$$

and the propellant and tank mass fraction is

$$\frac{\psi m_{propellant}}{m_0 f\left(C_3\right)} = \Omega \left(1 - \frac{2k_i \xi}{u_{e\ optimal}^2}\right) \tag{3.48}$$

The power and propulsion system to propellant and tank mass fraction ratio is

$$\frac{mass\ of\ power\ system\ and\ thrusters}{mass\ of\ propellant\ and\ tanks} = \frac{\alpha P_{elec}}{\psi m_{propellant}} = 1 + \frac{2\alpha k_i \xi}{\psi \eta_{cap} Z_f} \tag{3.49}$$

The equations are now dimensionless and allow for scaling; they also provide some insight into the optimal system characteristics. While 2Ω cannot be found without solving the optimal control problem, Eq. (3.49) shows that the optimal solution favors a larger portion of the spacecraft mass being allocated to the power/propulsion system versus the propellant. Furthermore, from the definition of Ω and Eq. (3.44), the power/propulsion system and propellant mass should be treated as one system as the payload mass fraction is a direct function of only two variables, C_3 and Ω. Also, unlike previous analysis[26, 32, 8], the exhaust velocity is not a function of the power or final mass, instead it is a function of the throttle history, Z_f, which indicates that the optimal exhaust velocity is independent of spacecraft mass and is a function of the throttle history, power system scaling, and propellant ionization cost.

3.4.1 Extended Analysis

Some of the earliest system level analysis into optimal thruster characteristics was conducted by Jahn[23]. Jahn's analysis did not account for the propellant ionization cost or the power system's power fluctuations over the trajectory. Work by Patel[43] extended Jahn's analysis by taking into account the different power systems and including various other system level costs. Here, the work is extended further by taking into account the propellant ionization costs.

The optimal specific impulse from Ref. [43] is

$$u_{e\ optimal} = \sqrt{2\frac{\psi Z_f \eta_{cap}}{\alpha}} \tag{3.50}$$

while the optimal specific impulse with the propellant ionization cost is

$$u_{e\ optimal} = \sqrt{2\left(\frac{\psi Z_f \eta_{cap}}{\alpha} + k_i \xi\right)} \tag{3.51}$$

The major difference between the two models is that the latter model requires a higher specific impulse in order to compensate for the propellant ionization costs. This indicates that the optimal exhaust velocity obtained by previous models will under report the actual optimal specific impulse.

The power and propulsion system to propellant and tank mass fraction ratio from previous work is

$$\frac{mass\ of\ power\ system\ and\ thrusters}{mass\ of\ propellant\ and\ tanks} = \frac{\alpha P_{elec}}{\psi m_{propellant}} = 1 \tag{3.52}$$

while the new result is

$$\frac{mass\ of\ power\ system\ and\ thrusters}{mass\ of\ propellant\ and\ tanks} = \frac{\alpha P_{elec}}{\psi m_{propellant}} = 1 + \frac{2\alpha k_i \xi}{\psi \eta_{cap} Z_f} \tag{3.53}$$

This result is significant because it shows that an optimal distribution of mass does NOT appropriate an equal amount of mass to the power system and propellant. Instead, the optimal distribution favors a larger power system mass. Depending on the length of the mission, the difference between the power system mass and propellant mass can be significant.

Incorporating the propellant ionization cost in the efficiency models has provided significant insight into the optimal thruster characteristics and updated the classical results of Jahn[23].

3.5 Proof of Concept Tests

While analysis of the payload mass fraction for the variable and constant specific impulse thrusters allows for some reductions, in order to determine if a mission is feasible, the trajectory, power level, and specific impulse has to be found. Several optimization techniques are implemented in a software program[44, 43] to conduct trade studies over a range of α values. The algorithm is designed to autonomously solve the optimization problem and determine the payload that is available for a mission.

For the high level trade study algorithm an indirect optimization problem[7, 2] is constructed and solved using a homotopy and shooting method[39]. An indirect approach is chosen because it results in a small set of unknown variables. The homotopy and shooting method are used because they robustly solve the indirect problem and they are easy to implement. The program varies α and returns the mass fractions of the payload, propellant, power system, and launch vehicle. The algorithm used here is the same method that is used in Ref. [43, 44].

3.5.1 Homotopy Parameters

The homotopy method works by varying a scalar parameter, called the homotopy parameter. Changing the homotopy parameter transforms a simple solvable problem into a more difficult problem. By slowly varying the homotopy parameter the solver has to solve many simpler problems, instead of one difficult problem.

The homotopy method is used in two parts of the optimization algorithm. First, k_2 is set to 1 and the gravitational parameter, μ, is set equal to 0, which reduces the dynamics to a no gravity problem that is solved relatively easily. The gravitational parameter is

$$\mu = 4\pi^2 \frac{i}{100} \frac{AU^3}{Yr^2} \tag{3.54}$$

where i is increased from 0 to 100, and for every value of i the indirect optimization problem is solved. As i increases, the problem is transformed from a gravity free problem to a fully gravitating problem. Because i is increased over many iterations the solver does not have much difficulty converging.

Once the fully gravitating problem is solved, the power system is considered. For an active nuclear power source $k_2 = 1$ and no modification is needed. If solar electric propulsion is utilized, then k_2 is

$$k_2\left(t, \boldsymbol{r}\right) = \left(\frac{|\boldsymbol{r}_0|}{|\boldsymbol{r}|}\right)^{2i/100} \tag{3.55}$$

and once again i is increased from 0 to 100.

3.5.2 Earth to Mars

The Earth to Mars mission tests the algorithms ability to solve constant specific impulse solar electric propulsion missions. The search range for α is 5 $\frac{kg}{kW}$ to 30 $\frac{kg}{kW}$, however the program terminates if $J_1 \leq 0.01$. The tank coefficient and structural coefficent are 0.15 and 0.05, and $\eta_{cap} = 0.7125$. Also, $k_i = 30$, the propellant is Xenon, and the launch vehicle is an Atlas V 401. The program produces several data products such as Fig. 3.2 and Fig. 3.3. Fig. 3.2 shows the mass fractions used by the power, propellant, and the payload. Fig. 3.3 is the optimal C_3 and specific impulse.

3.5.3 Interstellar Probe

The interstellar probe mission[32, 14, 66, 65] is designed to collect science as it escapes from the solar system. For this example, a 20 years to 150 AU constraint is used with the launch vehicle selected as an Atlas V 551. The structural coefficient and tank fraction is set to 5% and 10% respectively, and the propulsion system is selected as a constant specific impulse thruster utilizing Xenon propellant with an ionization cost, $k_i = 10$. The study is conducted assuming only the sun is gravitating

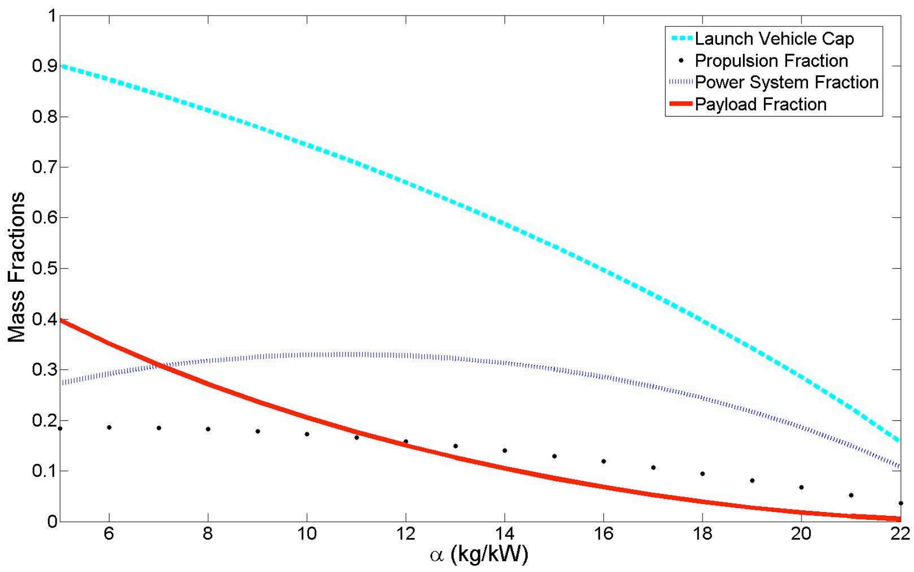


Figure 3.2: The launch vehicle cap is $f\left(C_3\right)$. As α increases the payload mass fraction and the launch vehicle cap decrease. The propellant and power system mass fraction does not follow a monotonic trend and has to be found through the optimization process.

except for a single flyby of Jupiter, which is modeled as an instantaneous change in the velocity vector. The Jupiter flyby distance is 5 Jupiter radii. The power system is a radioisotope thermal generator, which is modeled to decay as

$$k_2(t) = \left(\frac{1}{2}\right)^{\frac{t}{t_{1/2}}} \tag{3.56}$$

where $t_{1/2} = 88\, years$, the approximate half life of $Plutonium - 238$.

The trade study is undertaken to identify the necessary level of technology to make the mission feasible. The program run time is less than 1 minute. Fig. 3.5 shows that a feasible mission requires $\alpha \leq 70 kg/kW$.

3.6 Conclusions

This chapter analyzed the payload mass fraction for variable and constant specific impulse thrusters. No new reductions for the variable specific impulse case were found, however for the constant specific impulse case, the payload mass fraction, J_1, was reduced to a function of the launch vehicle and the power and propellant mass fraction, 2Ω. Also, for the constant specific impulse case, a new result is found and the exhaust velocity is independent of the mass of the spacecraft but is instead dependent on the throttle profile, ionization cost, and α. Two proof of concept optimization programs are created to conduct trade studies over α. The two example cases are, an Earth to Mars mission utilizing solar electric propulsion with a constant specific impulse thruster and an Interstellar Probe mission using radioisotope thermal generators with constant specific impulse thrusters. From the proof of concept test, the homotopy method solves the optimization problem with relative ease over a range of α values.

This high level trade study method is useful because it quickly bounds the problem and returns the maximum payload mass that can be delivered to the destination, while optimizing the power level, thruster performance, and launch vehicle utilization.

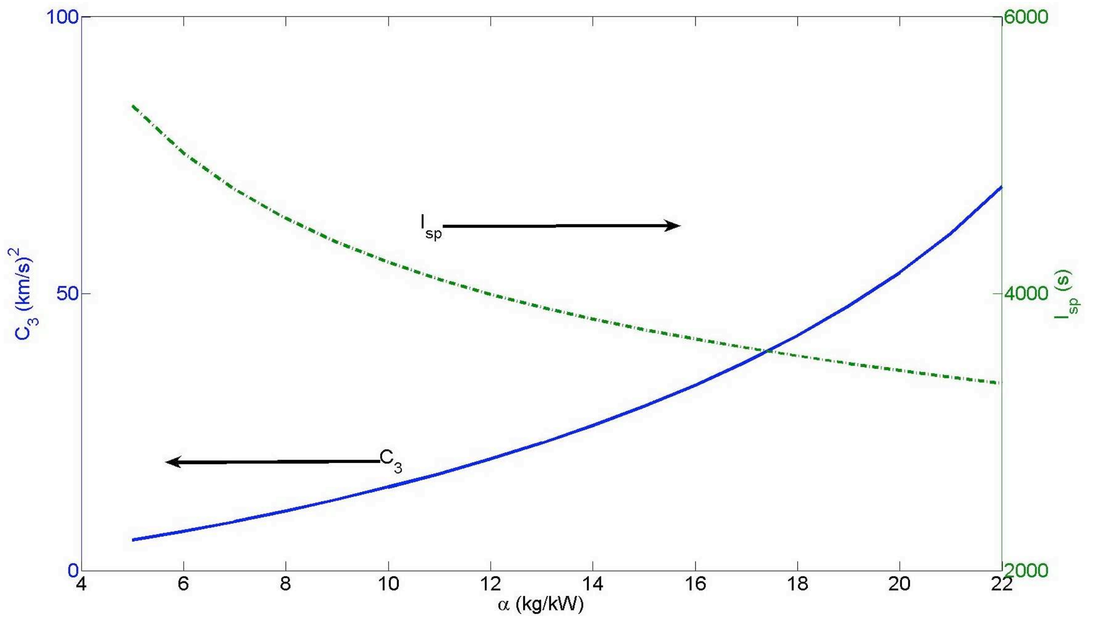


Figure 3.3: The optimal C_3 and specific impulse for a 6 month Earth to Mars mission using solar electric propulsion. As α increases the C_3 increases and the specific impulse decreases which is expected.

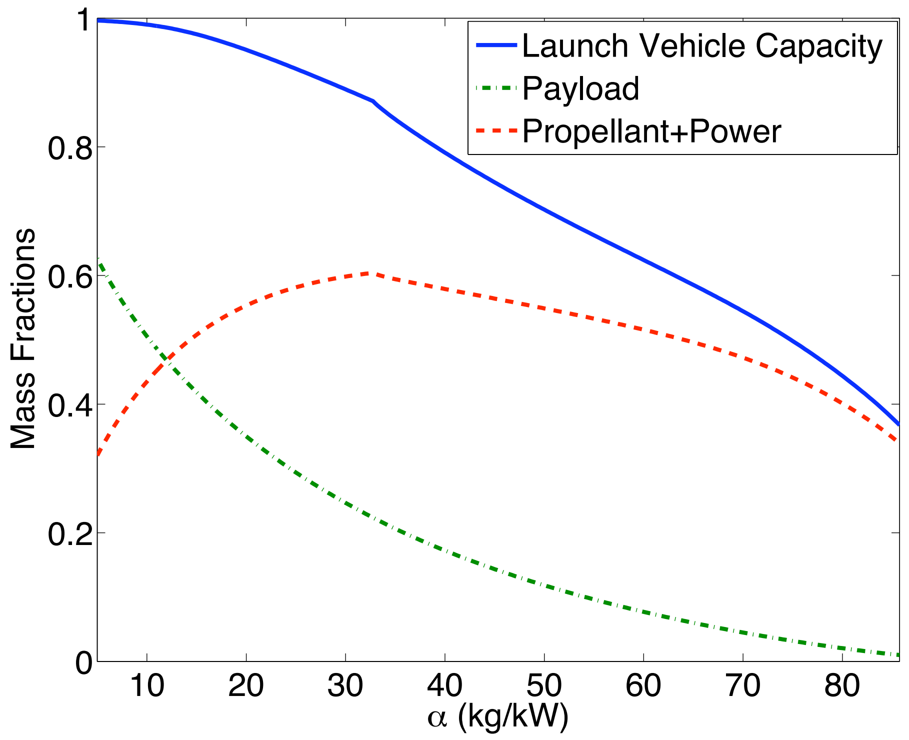


Figure 3.4: Payload mass delivered to 150 AU in 20 years over various α values with a gravity assist at Jupiter. The launch vehicle mass fraction identifies the mass fraction available after launch. Around $\alpha = 35\ kg/kW$ there is a change in slope and the launch vehicle dominates and the power and propellant mass fraction decreases.

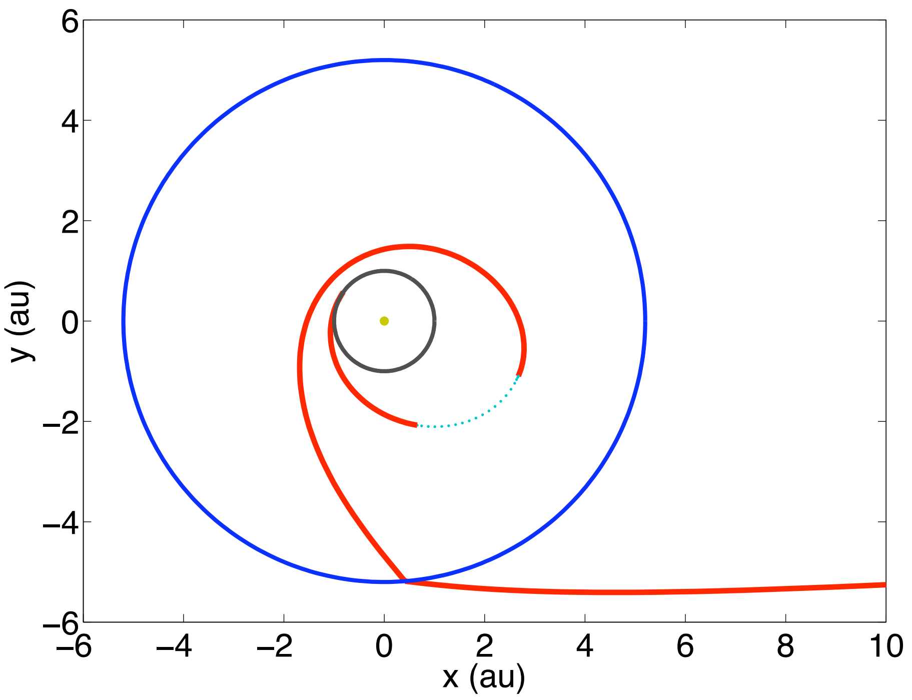


Figure 3.5: Trajectory of Interstellar probe for $\alpha = 60\ \frac{kg}{kW}$. The sold red line indicates the thrust phase and the dotted line indicates a coast phase. The circular orbits are the Earth and Jupiter. For high α values a coast arc appears before escape occurs. This shows that the trajectory can significantly vary depending on α.

CHAPTER IV

APPROXIMATING INTERPLANETARY TRAJECTORIES WITH CHEBYSHEV POLYNOMIALS

4.1 Introduction

The previous chapter focused on the major subsystems: thruster, power, and launch vehicle, and nondimensionalized the cost function. Two generic thruster types were considered, variable specific impulse and constant specific impulse thrusters. The formulation was programed into an automated solver[43] that generated solutions to both an Earth to Mars mission utilizing solar electric propulsion and an Interstellar Probe mission using radioisotope thermal generators with a constant specific impulse thruster.

High level trades are useful for characterizing the technology level that makes a mission feasible. However, high level trades are not useful for quantifying the performance of a specific thruster because they utilize reduced order models, which approximate the performance of an entire class of thrusters. For example, the reduced order efficiency and thrust relationships used in Chapter III approximates the performance of all Hall and Ion thrusters; in reality, different thrusters will perform differently depending on their design. To quantify the performance benefit of a particular thruster, the thruster model, launch vehicle, and trajectory have to be found. Automating the trajectory optimization process is useful because it allows trades to be conduced quickly, but before optimization can begin, an initial guess has to be provided to the optimizer. If the initial guess is “good” then the optimal trajectory is found quickly, however if it is “bad” then the optimizer can fail. Thus finding a good initial guess is critical to the optimization process.

This chapter focuses on approximating interplanetary trajectories independent of the propulsion and power system. These approximate trajectories are used to conduct a search over launch dates, flight times, and heliocentric revolutions. The ΔV cost is estimated from the trajectory, which is then used to evaluate the merits of the trajectory. Once a good trajectory is found, the propulsion and power system are then considered this process is described in Chapter V. Together Chapter IV and Chapter V encompass Fig. 4.1. Chapter IV outlines the methods used to conduct the broad search for trajectories while Chapter V then uses the data to generate feasible trajectories. This chapter sets up and solves the ‘simple’ problem

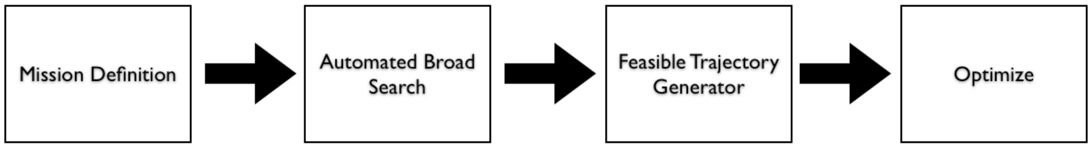


Figure 4.1: The approach used to generate feasible trajectories that can be used by optimization routines. The trajectories generated take into account power and thruster limitations.

in Fig. 1.4.

Currently, the exponential sinusoid[48, 36, 47, 49, 50] is used to approximate interplanetary trajectories. This chapter outlines a new method for approximating interplanetary trajectories. The method outlined in this chapter is designed to be fast, handle rendezvous constraints, and allow for optimization of the trajectory. This method differs from shape based approaches[48, 20, 45, 50, 49] because the trajectory is parameterized in time, and because the trajectory is optimized during the search process.

4.2 Trajectory Parameterization

The trajectory is parameterized by a set of coefficients and Chebyshev polynomials. The coefficients are the free parameters that are varied to generate different trajectories. The Chebyshev polynomials are the underlying functions that represent the trajectories. Each position coordinate is modeled separately, and in cartesian coordinates the position coordinates would be x, y, z, while in spherical coordinates the position coordinates are r, θ, ϕ. For a generic position coordinate, w, the time evolution of the coordinate is

$$w\left(t\right) = \sum_{j=0}^{N-1} c_{w,j} T_j\left(t\right) \tag{4.1}$$

and the time rate of change of the coordinate is

$$\dot{w}\left(t\right) = \sum_{j=0}^{N-1} c_{w,j} \dot{T}_j\left(t\right) \tag{4.2}$$

where $c_{w,j}$ is a coefficient that parameterizes the coordinate w and T_j is the j^{th} order Chebyshev polynomial[58]. The order of the polynomial, N, is also the number of degrees of freedom for that coordinate. In this formulation, only the positions are parameterized, and the velocities and accelerations can be found by differentiating the parameterization. Chebyshev polynomials can be computed recursively with

Eq. (4.3).

$$T_j(\tau) = \begin{cases} 1, & \text{if } j = 0 \\ \tau, & \text{if } j = 1 \\ 2\tau T_{j-1}(\tau) - T_{j-2}(\tau), & \text{if } j \geq 2 \end{cases} \tag{4.3}$$

and

$$\dot{T}_j(\tau) = \begin{cases} 0, & \text{if } j = 0 \\ \dot{\tau}, & \text{if } j = 1 \\ 2\dot{\tau} T_{j-1}(\tau) + 2\tau \dot{T}_{j-1}(\tau) - \dot{T}_{j-2}(\tau), & \text{if } j \geq 2 \end{cases} \tag{4.4}$$

where τ is computed by

$$\tau = 2\frac{t - t_0}{t_f - t_0} - 1 \tag{4.5}$$

$$\dot{\tau} = \frac{2}{t_f - t_0} \tag{4.6}$$

and t is bounded by

$$t_0 \leq t \leq t_f \tag{4.7}$$

Here t_0 is the time when the spacecraft begins it's interplanetary journey, and t_f is the time when the spacecraft reaches its destination. If there are enough degrees of freedom in the trajectory model, the parameterization can satisfy the boundary constraints. For a rendezvous problem, the parameterization requires a minimum of four degrees of freedom. Two degrees of freedom are needed to ensure that the spacecraft leaves the departure body and arrives at the destination body. Another two degrees of freedom are needed so the spacecraft's departure and arrival velocity matches the departure and destination body.

For the rendezvous problem, the trajectory leaves the departure body at a particular time, t_{launch} and rendezvous with the destination body at a specified time, $t_{arrival}$. The position and velocity of the departure and destination body can be obtained from an ephemeris and are assumed to be known for this analysis. With the states of the departure and arrival body known, a constraint problem can be set up that restricts the coefficients such that the trajectory will satisfy boundary conditions.

The states of the departure body at t_0 in polar coordinates are $r_0, \theta_0, \phi_0, \dot{r}_0, \dot{\theta}_0, \dot{\phi}_0$ and the states of the arrival body are t_f in polar coordinates are $r_f, \theta_f, \phi_f, \dot{r}_f, \dot{\theta}_f, \dot{\phi}_f$. The constraint for the departure body is given by Eq. (4.9) and the constraint for the arrival body is given inEq. (4.10).

$$B(\tau) = \begin{bmatrix} T_0(\tau) & \dots & T_{N-1}(\tau) \\ \dot{T}_0(\tau) & \dots & \dot{T}_{N-1}(\tau) \end{bmatrix} \tag{4.8}$$

$$\begin{bmatrix} r_0 \\ \dot{r}_0 \\ \theta_0 \\ \dot{\theta}_0 \\ \phi_0 \\ \dot{\phi}_0 \end{bmatrix} = \begin{bmatrix} B(-1) & 0 & 0 \\ 0 & B(-1) & 0 \\ 0 & 0 & B(-1) \end{bmatrix} \begin{bmatrix} c_{r,0} \\ \vdots \\ c_{r,N-1} \\ c_{\theta,0} \\ \vdots \\ c_{\theta,N-1} \\ c_{\phi,0} \\ \vdots \\ c_{\phi,N-1} \end{bmatrix} \tag{4.9}$$

$$\begin{bmatrix} r_f \\ \dot{r}_f \\ \theta_f \\ \dot{\theta}_f \\ \phi_f \\ \dot{\phi}_f \end{bmatrix} = \begin{bmatrix} B(1) & 0 & 0 \\ 0 & B(1) & 0 \\ 0 & 0 & B(1) \end{bmatrix} \begin{bmatrix} c_{r,0} \\ \vdots \\ c_{r,N-1} \\ c_{\theta,0} \\ \vdots \\ c_{\theta,N-1} \\ c_{\phi,0} \\ \vdots \\ c_{\phi,N-1} \end{bmatrix} \tag{4.10}$$

Any solution that satisfies Eq. (4.9) and Eq. (4.10) will meet the boundary constraints. In order for a solution to Eq. (4.9) and Eq. (4.10) to exist, the order of the polynmials,N, must be greater than or equal to 4. Because Eq. (4.9) and Eq. (4.10) are linear, they can be solved with a variety of numerical methods. Any set of coefficients that satisfyEq. (4.9) and Eq. (4.10) also satisfy the rendezvous problem and so parameterize a valid trajectory. This formulation allows linear state boundary conditions to be satisfied easily, which is a goal of this method.

4.3 Multi Revolution Problem

While any coefficient set that satisfies Eq. (4.9) and Eq. (4.10) is a valid solution to the rendezvous, there exists some freedom in the rendezvous conditions. The freedom exists in Eq. (4.10) because θ_f can be written as

$$\theta_f = \tan^{-1}\left(\frac{x_f}{y_f}\right) + 2\pi m \tag{4.11}$$

where m is an integer that is greater than or equal to zero. Unlike the exponential sinusoid[20], there is no upper bound on m based on the parameterization, which means that m must be bounded based on a different set of criteria. In order to bound the number of heliocentric revolutions, m, a practical method is used that is based on the periods of the departure and destination bodies.

The maximum number of revolutions that the spacecraft will make is equivalent to the maximum number of revolutions that the departure or destination body makes. Equivalently, the lower bound on m is the minimum number of revolutions that the departure or destination body will make. Defining the set Pd as

$$Pd = (Orbital\ period\ of\ departure\ body,\ Orbital\ period\ of\ destination\ body) \tag{4.12}$$

then

$$m \geq \left\lfloor \frac{t_f - t_0}{max\,(Pd)} \right\rfloor \tag{4.13}$$

and

$$m \leq \left\lfloor 0.7 * \frac{t_f - t_0}{min\,(Pd)} \right\rfloor + 1 \tag{4.14}$$

Here the 0.7 is found empirically, and the +1 is a safety factor. Eq. (4.13) and Eq. (4.14) work because as the spacecraft travels from the departure body to the destination body its orbital period will vary between the periods of the target bodies. Equivalently, is is expected that the number of revolutions should also vary between number of revolutions made by the two target bodies.

4.4 Trajectory Cost Function

When the order of the polynomials are greater then four, there exists an infinite number of trajectories that satisfy the rendezvous problem. In order to select a good trajectory, a cost function is needed that can measure the value of a particular trajectory. The cost function is

$$J = \int_{t_0}^{t_f} |\boldsymbol{A}|^2 \, dt \tag{4.15}$$

Eq. (4.15) is used because it is smooth and quadratic, which usually implies that the optimization problem is numerically simple to solve. Although the cost function used here does not minimize the propellant cost, this is not a major concern at this stage because the trajectories are only being approximated.

The dynamics for a point mass in a central gravity field (spacecraft orbiting the sun) are given as

$$\ddot{\boldsymbol{r}} = -\frac{\mu_{sun}}{|\boldsymbol{r}|^3}\boldsymbol{r} + \boldsymbol{A} \tag{4.16}$$

where $\boldsymbol{A}$ is the acceleration required to maintain the path, and $\boldsymbol{r}$ is the position of the spacecraft. Solving for the acceleration, $\boldsymbol{A}$, is

$$\boldsymbol{A} = \ddot{\boldsymbol{r}} + \frac{\mu_{sun}}{|\boldsymbol{r}|^3}\boldsymbol{r} \tag{4.17}$$

Eq. (4.17) shows that the acceleration is now a function of the position of the spacecraft, $\boldsymbol{r}$, and its time derivatives. This is useful because the position of the spacecraft is parameterized by Eq. (4.2), which means that the derivates can be computed simply by differentiating T, the basis function. Substituting in the trajectory function, the acceleration is

$$\boldsymbol{A} = \begin{bmatrix} \sum_{i=0}^{N-1} c_{x,i}\ddot{T}_i + \frac{\mu_{sun}}{|\boldsymbol{r}|^3} \sum_{i=0}^{N-1} c_{x,i}T_i \\ \sum_{i=0}^{N-1} c_{y,i}\ddot{T}_i + \frac{\mu_{sun}}{|\boldsymbol{r}|^3} \sum_{i=0}^{N-1} c_{y,i}T_i \\ \sum_{i=0}^{N-1} c_{z,i}\ddot{T}_i + \frac{\mu_{sun}}{|\boldsymbol{r}|^3} \sum_{i=0}^{N-1} c_{z,i}T_i \end{bmatrix} \tag{4.18}$$

$$|\boldsymbol{r}| = \sqrt{\left(\sum_{i=0}^{N-1} c_{x,i}T_i\right)^2 + \left(\sum_{i=0}^{N-1} c_{y,i}T_i\right)^2 + \left(\sum_{i=0}^{N-1} c_{z,i}T_i\right)^2} \tag{4.19}$$

The acceleration is now a function of the path coefficients, $c_{x,i}$, $c_{y,i}$, and $c_{z,i}$. Since the cost function is a function of the trajectory coefficients, this implies that the cost function can be minimized. With the cost function and constraints defined, the path coefficients can be found such that the constraints are satisfied and the cost is minimized.

4.5 Broad Search Algorithm

The two disadvantages of the shape based methods have been addressed. These disadvantages are the inability to optimize the trajectory and the inability to handle rendezvous problems. However, for the Chebyshev trajectory approximation to be useful, it must be used in a similar fashion to the shape based approach, as a tool that estimates the ΔV cost and generates trajectory estimates over a large range of launch dates and arrival dates.

Because of the differences between the shape based approach and the Chebyshev trajectory approximation, the algorithm for conducting the broad search is different. Unlike the shape based method in the Chebyshev trajectory approximation, the timing and boundary conditions are explicitly met. However there are extra degrees of freedom, so the trajectory does have room to be optimized. Also, for the Chebyshev trajectory approximation method, the number of heliocentric revolution space has to be searched through because of the freedom in the boundary constraints, Eq. (4.11).

The Chebyshev trajectory approximation broad search algorithm begins by having the user specify the departure body, the destination body, the launch dates, and arrival dates. Then, for each possible launch date and arrival date combination, the method computes the number of heliocentric revolutions and generates a trajectory for each heliocentric revolution. The pseudocode for finding trajectories is Alg. (4.1). The algorithm is missing a single piece of information, how to minimize

Algorithm 4.1 Algorithm for generating Chebyshev approximated trajectories between two bodies over a wide range of launch dates and arrival dates.

```
for all $t_0 \in$ Launch Dates do
  for all $t_f \in$ Arrival Dates do
    for all $m \in$ Possible heliocentric revolutions do
      min J s.t. boundary conditions are satisfied
    end for
  end for
end for
```

J.

4.5.1 Broad Search Initial Guess Generation

With the number of heliocentric revolutions known, an initial guess and optimization algorithm is needed to optimize the trajectory, which will minimize Eq. (4.15). The optimization algorithm used is a Sequential Quadratic Programming[2, 39, 16] algorithm. The Sequential Quadratic Programming method approximates the constraints to the first order and the cost function to the second order. Because the constraints are linear, the Sequential Quadratic Programming method will explicitly satisfy the constraints during every iteration.

While there exist many ways to generate an initial guess, the method used here is to simply set the Chebyshev polynomial order to $N = 4$. Because the number of coefficients is equal to the number of free parameters, there exists only one solution to Eq. (4.9) and Eq. (4.10), which define the rendezvous problems.

Next, N is increased by 1 to 5, and the new coefficients, $c_{r,N}$, $c_{\theta,N}$, $c_{\phi,N}$ are set equal to 0. Now the $N = 5$ trajectory is equivalent to the $N = 4$ trajectory, but the $N = 5$ trajectory has extra degrees of freedom that can be optimized. This trajectory is then used as an initial guess to the sequential quadratic programming method, which takes the initial guess and returns the optimal solution. This process can be generalized to generate higher order trajectory approximations. It was found that $N = 10$ is usually sufficient to model interplanetary trajectories. The generalized algorithm is Alg. (4.2).

Algorithm 4.2 Algorithm for optimizing the Chebyshev trajectory approximation. The algorithm increases the order of the Chebyshev polynomial from $N = 4$ to $N = 10$ and optimizes the trajectory.

```
$N = 4$
find coefficients that solve rendezvous problem
**for** $N = 5$ to 10 **do**
    Set coefficients for polynomial order $N$ to $N - 1$ coefficients
    Set higher order coefficients to 0
    minimize $J$ using Sequential Quadratic Programming method
**end for**
```

Alg. (4.2) represents a self contained method that can generate an initial guess and minimize the cost function. Graphically, Alg. (4.2) is shown in Fig. 4.2. Alg.

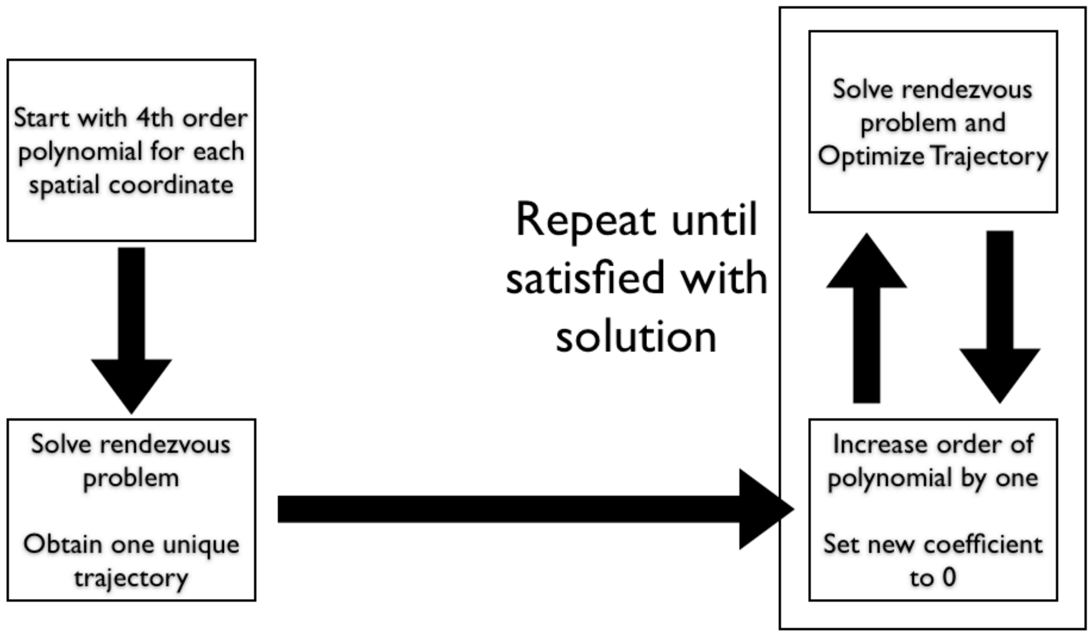

Start with 4th order polynomial for each spatial coordinate
Solve rendezvous problem
Obtain one unique trajectory
Repeat until satisfied with solution
Solve rendezvous problem and Optimize Trajectory
Increase order of polynomial by one
Set new coefficient to 0

Figure 4.2: Graphically shows the algorithm for generating a trajectory using Chebyshev polynomials.

(4.2) is self contained because it begins with a unique solution to the rendezvous problem. This unique trajectory is then used as an initial guess to find subsequent optimal trajectories represented by higher order polynomials.

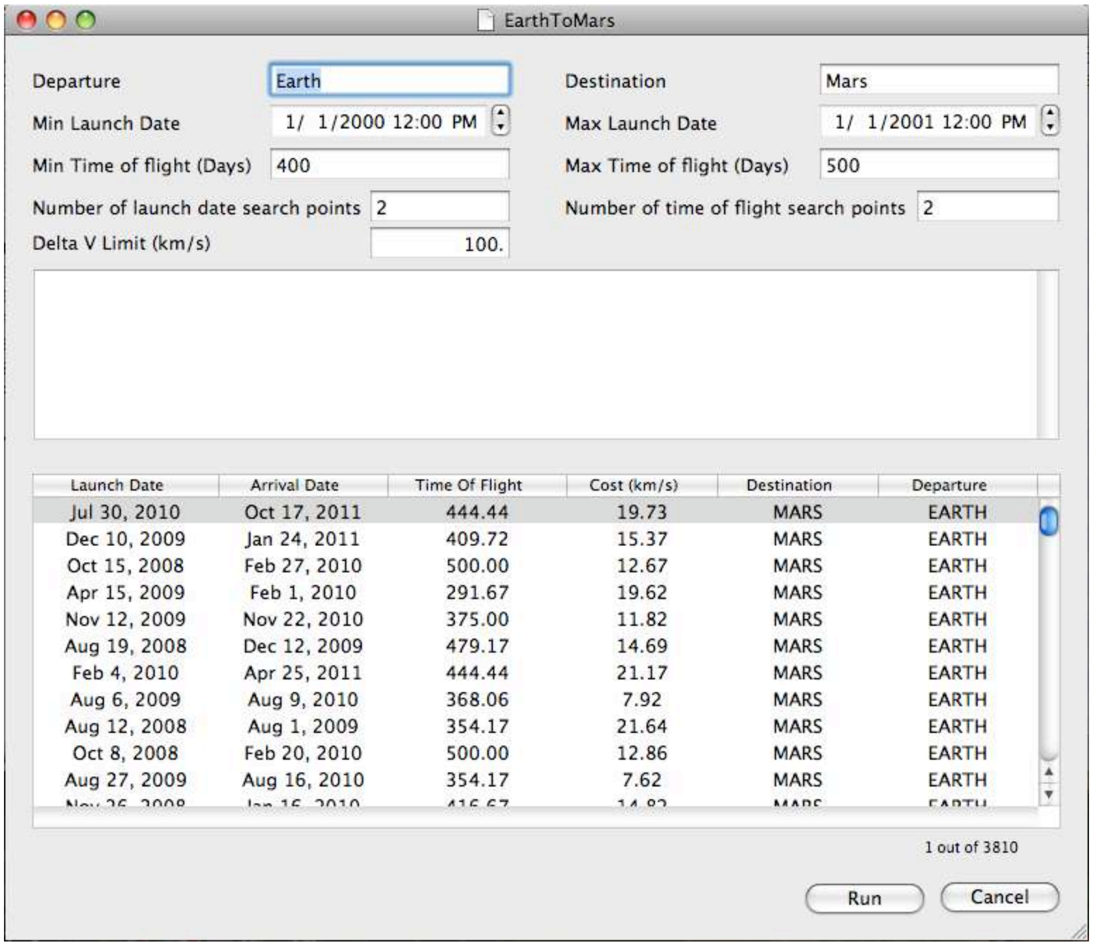
4.5.2 Multi Segment Trajectories

The method outlined previously is designed for point to point rendezvous trajectories however, it can be easily modified to handle multiple segment trajectories. Multiple segment trajectories arise when gravity assists or a multiple tour mission is used. Two modifications to Alg. (4.2) are needed to handle multiple segment trajectories. The first modification is to replace the rendezvous conditions with the appropriate boundary conditions for the mission. For a multiple tour mission no change is required while for a gravity assist problem the gravity assist continuity equations would need to be used to account for the change in the spacecraft velocity. Next, the variable N in Eq. (4.2) would need to be set to the number of free parameters on each segment. With these modifications the algorithm can be generalized.

4.5.3 Chebyshev Trajectory Approximation Examples

A tool to generate feasible trajectories is coded up in C++ and Objective C on an Apple computer using OS 10.5. The program uses the CSPICE libraries. The broad search algorithm runs in multiple threads. The problem scope for this demonstration program is limited to single rendezvous problems.

The broad search program is shown in Fig. 4.3. The program requires the departure body, arrival body, launch dates, and times of flight. The program then searches over the entire space and stores all potentially valid solutions. The limited input requirement makes broad searches easy to conduct. The program interface, shown in Fig. 4.3, allows for the use of multiple processors. Because each subproblem generates its own initial guess, the subproblems can be solved independent of the each other, allowing the methods to be used in a parallel or distributed environment, which reduces the computational time.



EarthToMars

Departure	Earth	Destination	Mars
Min Launch Date	1/ 1/2000 12:00 PM	Max Launch Date	1/ 1/2001 12:00 PM
Min Time of flight (Days)	400	Max Time of flight (Days)	500
Number of launch date search points	2	Number of time of flight search points	2
Delta V Limit (km/s)	100.		

Launch Date	Arrival Date	Time Of Flight	Cost (km/s)	Destination	Departure
Jul 30, 2010	Oct 17, 2011	444.44	19.73	MARS	EARTH
Dec 10, 2009	Jan 24, 2011	409.72	15.37	MARS	EARTH
Oct 15, 2008	Feb 27, 2010	500.00	12.67	MARS	EARTH
Apr 15, 2009	Feb 1, 2010	291.67	19.62	MARS	EARTH
Nov 12, 2009	Nov 22, 2010	375.00	11.82	MARS	EARTH
Aug 19, 2008	Dec 12, 2009	479.17	14.69	MARS	EARTH
Feb 4, 2010	Apr 25, 2011	444.44	21.17	MARS	EARTH
Aug 6, 2009	Aug 9, 2010	368.06	7.92	MARS	EARTH
Aug 12, 2008	Aug 1, 2009	354.17	21.64	MARS	EARTH
Oct 8, 2008	Feb 20, 2010	500.00	12.86	MARS	EARTH
Aug 27, 2009	Aug 16, 2010	354.17	7.62	MARS	EARTH
Nov 26, 2008	Jan 16, 2010	416.67	14.82	MARS	EARTH

1 out of 3810

Figure 4.3: Implementation of broad search routine. Basis functions are given by Chebyshev Polynomials. Program only requires mission parameters to begin searching for trajectories. For an Earth to Mars mission about 4,000 trajectories are found that have $\Delta V \leq 30\ \frac{km}{s}$

To show the flexibility of the Chebyshev polynomials in representing the trajectories, two different Earth to Jupiter missions are considered. The first example is given in Fig. 4.4. The figure shows the trajectory and the estimated thrust vector. The trajectories are continuously thrusting, due to the use of the polynomial basis functions, however the trajectories look like low thrust trajectories. The first

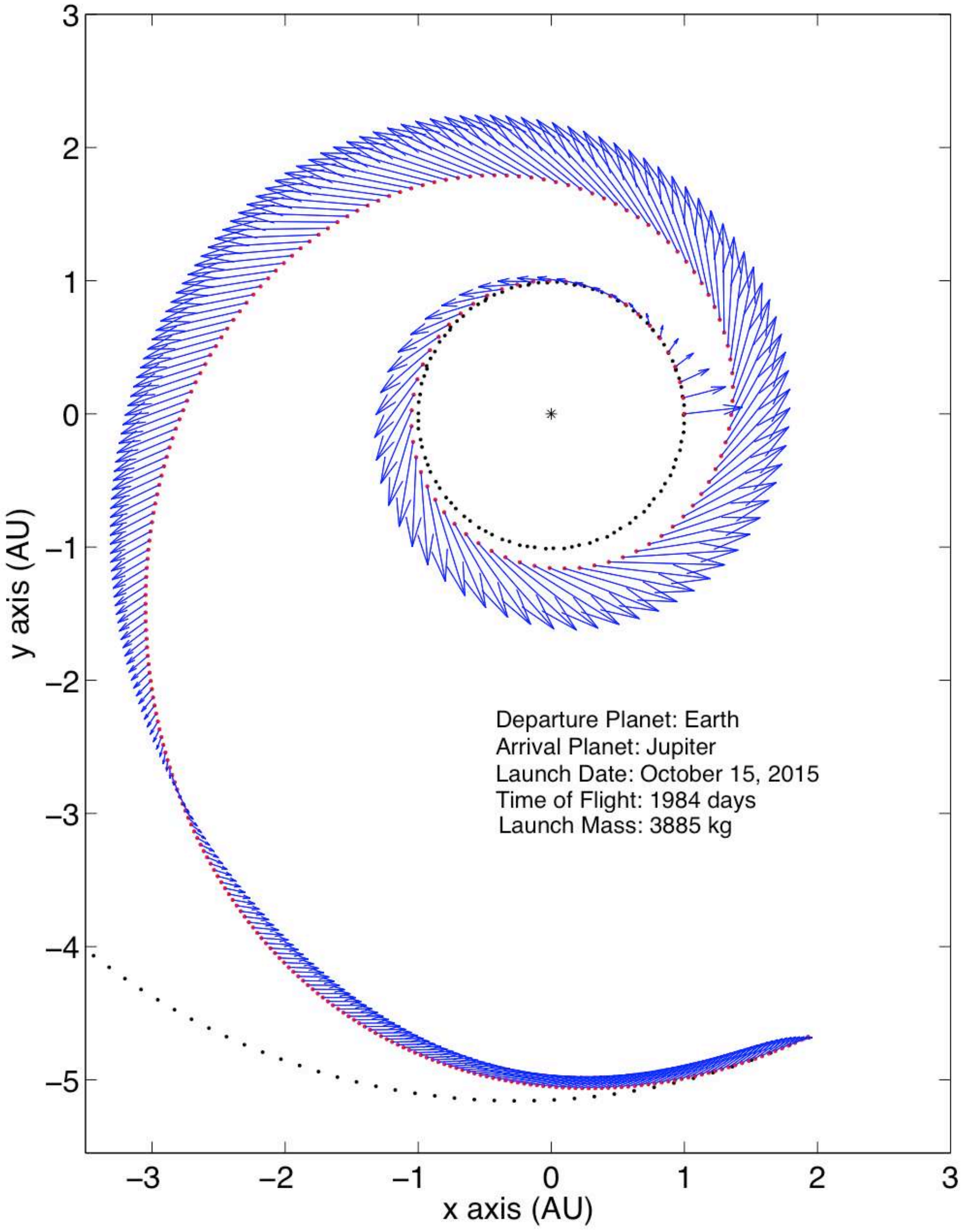


Figure 4.4: Chebyshev polynomial representation of a Earth to Jupiter trajectory. A 10^{th} order polynomial represents the trajectory. The arrows indicate direction and magnitude of the thrust vector.

example demonstrates that a 10^{th} order polynomial can model electric propulsion trajectories well over a low number of revolutions with a large change in radius. The 10^{th} order polynomial is modeling a 5.4 year trajectory that makes 1 full heliocentric revolution.

The second example is given in Fig. 4.5. The figure shows the trajectory and the estimated thrust vector for a 10 year multiple revolution trajectory to Jupiter. This example demonstrates that low order Chebyshev polynomials can represent electric propulsion trajectories over long periods of time and a large number of revolutions. The second example makes 6 heliocentric revolutions, 5 of which are are near 1 AU. This demonstrates that Chebyshev polynomials can remain relatively constant over most of the trajectory then change over a short period of time. The two examples demonstrate that Chebyshev polynomials can represent electric propulsion trajectories.

4.6 Conclusions

In this chapter an automated method for representing electric propulsion trajectories has been described. Unlike the shape based approach, this method uses Chebyshev basis functions to represent the trajectory and the trajectory is directly parameterized in time. Using linear basis functions in time fixes the timing and converts the rendezvous problem into a linear problem which can be solved simply. The ability to solve rendezvous problems addresses one of the shortcomings of the exponential sinusoid. Also, because this method allows for arbitrary order polynomials, the coefficients can be optimized. This allows low cost solutions to be found, which cannot be done with the exponential sinusoid.

The broad search algorithm in conjunction with the optimization method can

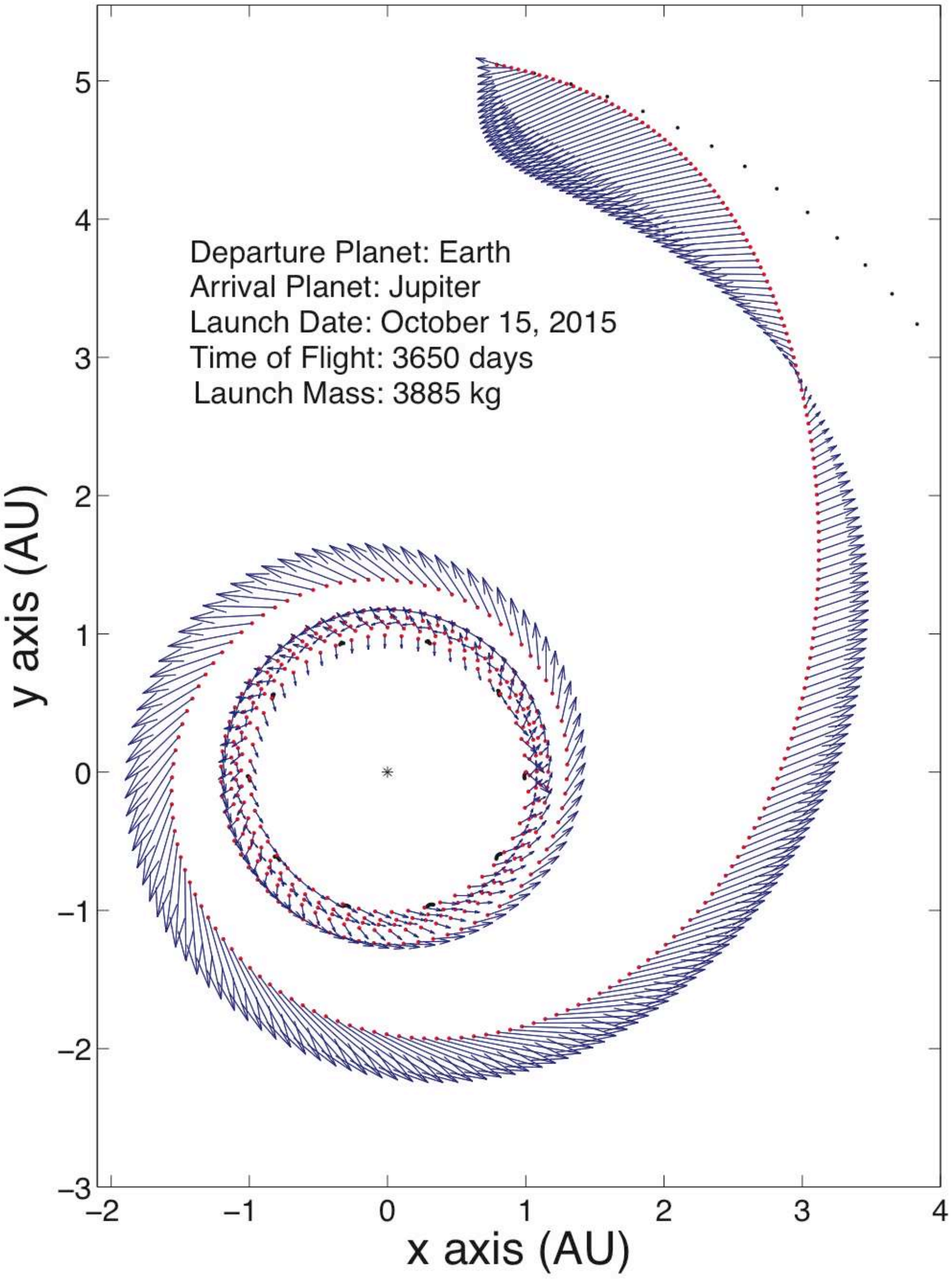


Figure 4.5: Chebyshev polynomial representation of a Earth to Jupiter trajectory with multiple heliocentric revolutions. A 10^{th} order polynomial represents the trajectory. The arrows indicate direction and magnitude of the thrust vector.

rapidly find rendezvous trajectories between two bodies. Furthermore, because the program searches over the number of heliocentric revolutions, it finds the most favorable structure without requiring any a priori knowledge of the user. Also, because each trajectory in the search space can be solved independent of any other trajectory, the search process can use parallel or distributed processing to reduce the computational time. This method is the first step in automating trajectory generation process. While, the algorithms presented here allows for a large number of trajectories to be generated, they do not take into account the thruster, launch vehicle, or power constraints.

CHAPTER V

GENERATING ELECTRIC PROPULSION TRAJECTORIES THAT SATISFY THRUSTER AND POWER CONSTRAINTS

5.1 Introduction

In the previous chapter a method for approximating the electric propulsion trajectories that addressed two of the shortcomings of the exponential sinusoid and shape base methods was presented. The Chebyshev approximation is designed to satisfy rendezvous problems, and it has enough degrees of freedom to permit trajectory optimization. Combining the Chebyshev approximation with the broad search algorithm allows for the rapid generation of trajectories. The ability to rapidly search for trajectories represents the first step in the larger goal of automating the trajectory generation process.

Unfortunately, the Chebyshev approximation does not generate feasible trajectories. Feasible trajectories are trajectories that satisfy the propulsion system's thrust and power system's power limitations. The next step to generating feasible trajectories is to convert the Chebyshev approximations into trajectories that satisfy the thrust, mass, and power constraints of the mission.

This chapter outlines a method for generating feasible trajectories using the

Chebyshev approximation as a starting point. The acceleration profile from the Chebyshev trajectory is converted into a thrust profile, and the trajectory is fully integrated using the equations of motion. Errors between the thrust profile, thruster model, and destination constraints are corrected. Once a trajectory that satisfies the thruster constraints is found, the power system's constraints are included in the set of active constraints, and the errors are corrected for. Once the trajectory satisfies all the constraints, it is a feasible trajectory. A feasible trajectory is a trajectory that satisfies the dynamics, thrust, and power constraints. Feasible trajectories are beneficial because they place an upper bound on the propellant utilized and can be good initial guesses for trajectory optimizers.

5.2 Fully Integrated Model

Once the Chebyshev approximation model generates a trajectory, the trajectory must to be converted into a solution that will satisfy the thruster and power system constraints. While the Chebyshev model minimizes the integral of the acceleration squared, the acceleration profile is not constrained. Hall and Ion thrusters can only provide thrust over a limited finite range. First, the number of control segments, M, must be selected. The user then specifies the launch vehicle, the engine, the duty cycle, and the power system.

The initial thrust profile is generated by converting the acceleration profile from the Chebyshev trajectory approximation into a thrust profile. First the launch vehicle, thruster, and the number of control segments, M, must be determined. From the thruster, the maximum exhaust velocity, u_e, is found, and from the launch vehicle, the initial mass is set. Using the exhaust velocity, initial mass, and the acceleration profile from the Chebyshev approximation a thrust profile can be generated using

Alg. (5.1). The initial thrust profile, T_i, will not satisfy the thruster limitations or

Algorithm 5.1 Algorithm for converting the acceleration profile of the Chebyshev trajectory into a thrust profile based on thruster and launch vehicle specifications. u_e is the maximum exhaust velocity of the specified thruster and d_c is the duty cycle of the thruster.

```
$m_0 = F(0)$
**for** $i = 0$ to $M - 1$ **do**
    $t_i = \frac{i}{M}(t_f - t_0) + t_0$
    $\Delta V_i = \int_{t_i}^{t_{i+1}} |\boldsymbol{A}|\, dt$
    $m_{i+1} = m_i e^{-\Delta V_i / u_e}$
    $\boldsymbol{T}_i = \frac{(m_i - m_{i+1}) u_e}{d_c (t_{i+1} - t_i)} \frac{\boldsymbol{A}(t_i)}{|\boldsymbol{A}(t_i)|}$
**end for**
```

the rendezvous problem. A solver is needed that can take the initial control law and convert it into a control sequence that satisfies the constraints. Ideally, the solver should use simple numerical techniques so that it can be easily implemented.

The equations of motion are

$$\ddot{\boldsymbol{r}} = -\frac{\mu_{sun}}{|\boldsymbol{r}|^3}\boldsymbol{r} + s\boldsymbol{T}n \tag{5.1}$$

$$\dot{n} = -n^2 \dot{m}(\phi)\, s \tag{5.2}$$

where ϕ is the throttle, s is the engine state (on or off), and $\boldsymbol{T}$ is the thrust. n is the inverse of the mass and is equal to $1/m$. The control constraints are defined as

$$q_i = 0 = 0.5\left[T_T(\phi_i)\right]^2 - 0.5\,|\boldsymbol{T}_i|^2 \tag{5.3}$$

and

$$S_i = 0 = 0.5 s_i (s_i - 1) \tag{5.4}$$

In Eq. (5.3) $T_T(\phi_i)$ is the thrust that the propulsions system provides as a function of the throttle. The parameterization of $T_T(\phi_i)$ for the NSTAR, BPT-4000, and the NEXT thrusters is given in Appendix D. The control inequality constraint is

$$-1 \le \phi_i \le 1 \tag{5.5}$$

The launch vehicle constraint is

$$0 \geq n_0 \, F\left(C_3\right) - 1.0 \tag{5.6}$$

The equality constraints are converted into equality constraints using a slack variable method[39]. In the slack variable method, an inequality method is converted into an equality constraint by adding a slack variable, which is added to the control constraint. For example, the launch vehicle inequality constraint, Eq. (5.6), becomes

$$0 = n_0 \, F\left(C_3\right) - 1.0 + \frac{1}{2}s^2 \tag{5.7}$$

where s is a slack variable, which has to be found during the optimization process.

The constraints, $\boldsymbol{C}$, and controls, $\boldsymbol{U}$ are

$$\boldsymbol{C} = \begin{bmatrix} q_0 \\ \vdots \\ q_{M-1} \\ S_0 \\ \vdots \\ S_{M-1} \\ -1 \leq \phi_0 \leq 1 \\ \vdots \\ -1 \leq \phi_{M-1} \leq 1 \\ 0 \geq 1.0 - n_0 \, F\left(C_3\right) \\ x_f - x_M \\ \dot{x}_f - \dot{x}_M \\ \vdots \\ z_f - z_M \\ \dot{z}_f - \dot{z}_M \end{bmatrix} \tag{5.8}$$

and

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{T}_0 \\ \vdots \\ \boldsymbol{T}_{M-1} \\ \phi_0 \\ \vdots \\ \phi_{M-1} \end{bmatrix} \tag{5.9}$$

5.2.1 Linear Subproblem

For a trajectory to be feasible it must satisfy

$$\boldsymbol{C}(\boldsymbol{U}) = 0 \tag{5.10}$$

Generally, the controls, $\boldsymbol{U}$, will not satisfy Eq. (5.10), so a method is needed that can modify $\boldsymbol{U}$ such that Eq. (5.10) will be true. The standard approach to a problem of this type is to take a first order Taylor series expansion of the constraint equations, $\boldsymbol{C}$, with respect to the controls, $\boldsymbol{U}$, and set that equation equal to zero. Taking the first order expansion gives

$$0 = \boldsymbol{C}(\boldsymbol{U}) + \frac{\partial \boldsymbol{C}(\boldsymbol{U})}{\partial \boldsymbol{U}} \boldsymbol{\delta U} \tag{5.11}$$

The control update is

$$\boldsymbol{U}_{update} = \boldsymbol{U} + \boldsymbol{\delta U} \tag{5.12}$$

If $\boldsymbol{\delta U}$ satisfies Eq. (5.11), then it linearly satisfies the constraint equations, and if the initial control vector, $\boldsymbol{U}$ is close to a feasible solution, then the linear method will converge to a feasible solution. With the constraints and control vector defined, Alg. (5.2) is used to satisfy the constraints. The trajectory that is generated will

Algorithm 5.2 Algorithm for satisfying the constraints. ϵ is a small positive number

```
while $|\boldsymbol{C}|^2 > \epsilon$ do
  solve $\boldsymbol{C} + \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{U}} \boldsymbol{\delta U}$
  update $\boldsymbol{U} = \boldsymbol{U} + \boldsymbol{\delta U}$
end while
```

depend on how Eq. (5.11) is solved.

5.2.2 Pseudoinverse Solution to the Linear Subproblem

The easiest and perhaps simplest method for solving Eq. (5.11) is to use the pseudoinverse[56] to construct $\boldsymbol{\delta U}$. The psuedoinverse minimizes the $\boldsymbol{\delta U}^T \boldsymbol{\delta U}$ while satisfying the constraint equation, Eq. (5.11). Defining $\boldsymbol{C_U} = \frac{\partial \boldsymbol{C}(\boldsymbol{U})}{\partial \boldsymbol{U}}$, the pseudoinverse, $\boldsymbol{C_U}^+$, of $\boldsymbol{C_U}$ is

$$\boldsymbol{C_U}^+ = \boldsymbol{C_U}^T \left(\boldsymbol{C_U}\boldsymbol{C_U}^T\right)^{-1} \tag{5.13}$$

Using the pseudoinverse, the control update is

$$\boldsymbol{\delta U} = -\boldsymbol{C_U}^+ \boldsymbol{C}\left(\boldsymbol{U}\right) \tag{5.14}$$

The pseudoinverse selects the "smallest" change in control such the constraint equation is satisfied. Because of the way the pseudoinverse generates the control update, it tends to converge quickly when used in Alg. (5.2).

5.2.3 Alternative Solutions to the Linear Subproblem

While the pseudoinverse approach does converge quickly and is simple to use, it is not the only approach to finding feasible trajectories. An alternative approach to the psuedoinverse method demonstrates that the solution to Eq. (5.11) can be customized to different needs. In the following examples, Eq. (5.11) is solved such that an alternative cost function is minimized for several iterations. The cost function used is

$$\sum_{i=0}^{M-1} |\boldsymbol{T}_i| + 2\pi \boldsymbol{\delta r}_i^T \boldsymbol{\delta r}_i \tag{5.15}$$

Here $\boldsymbol{\delta r}_i$ is the deviation in the position of the spacecraft. $\boldsymbol{\delta r}_i$ is included in the cost function to stabilize the algorithm and prevent the control update from causing large changes to the control. Because of the difficulty with minimizing Eq. (5.15), a multiple shooting method[2, 39] is also employed.

The multiple shooting method breaks the trajectory into $M-1$ segments. The states are now treated as control variables, and the continuity conditions are added to the constraint set. The continuity conditions require that the solution does not have any discontinuities in the mass, position, and velocity of the spacecraft. The control for the multiple shooting problem, $\boldsymbol{U_{ms}}$ is

$$\boldsymbol{U_{ms}} = \begin{bmatrix} \boldsymbol{U} \\ x_{g,0} \\ \dot{x}_{g,0} \\ \vdots \\ y_{g,i} \\ \dot{y}_{g,i} \\ \vdots \\ z_{g,M-1} \\ \dot{z}_{g,M} \end{bmatrix} \tag{5.16}$$

Here the subscript g, i denotes the presumed position and velocity of the spacecraft at the i^{th} segment. The constraint vector, $\boldsymbol{C}_{ms}$ is

$$\boldsymbol{C_{ms}} = \begin{bmatrix} \boldsymbol{C} \\ x_i - x_{g,0} \\ \dot{x}_i - \dot{x}_{g,0} \\ \vdots \\ y_i - y_{g,i} \\ \dot{y}_i - \dot{y}_{g,i} \\ \vdots \\ z_i - z_{g,M-1} \\ \dot{z}_i - \dot{z}_{g,M} \end{bmatrix} \tag{5.17}$$

The additional constraints represent the continuity conditions between the presumed position and velocity and the actual position and velocity of the spacecraft. The first order expansion of the new constraint equation is

$$0 = \boldsymbol{C}_{ms}\left(\boldsymbol{U}_{ms}\right) + \frac{\partial \boldsymbol{C}_{ms}\left(\boldsymbol{U}_{ms}\right)}{\partial \boldsymbol{U}_{ms}} \boldsymbol{\delta U}_{ms} \tag{5.18}$$

The control update $\boldsymbol{\delta U}_{ms}$ is selected such that it satisfies Eq. (5.18) and minimizes Eq. (5.15).

The initial Chebyshev trajectory is provided by Fig. 4.4 and Fig. 4.5. In these two figures, the initial thrust varies over the entire trajectory, and the thruster is always on. For this example, the specific impulse is set to 6400 seconds, and the maximum thrust is 514.7 mN. The launch mass is 3885 kg, and the initial C_3 is constrained to be 0. The power system is assumed to provide a constant power over all time; the mass of the power system is not considered in this example. The goal of these examples is to demonstrate that custom approaches to solving Eq. (5.11) can be developed depending on the need.

Fig. 5.1 and Fig. 5.2 are the trajectories after Fig. 4.4 and Fig. 4.5 have been made feasible. The major difference between the initial guess and the final feasible trajectory is the appearance of coast-arcs, which would not occur if the psuedoinverse had been used. These examples demonstrate that different methods can be used to solve Eq. (5.11).

5.2.4 Examples

An Earth to Mars mission is considered to demonstrate the utility of the pseudoinverse method, with a flight time of 500 days. The initial trajectories are provided by the Chebyshev approximation method. For the 500 day example, the launch date is July 23, 2009, and the launch vehicle is the Atlas V 501 (Table 5.1). For solar electric power sources the power output scales as

$$\frac{P\left(t\right)}{P\left(t_0\right)} = \left(\frac{r_0}{r\left(t\right)}\right)^2 \tag{5.19}$$

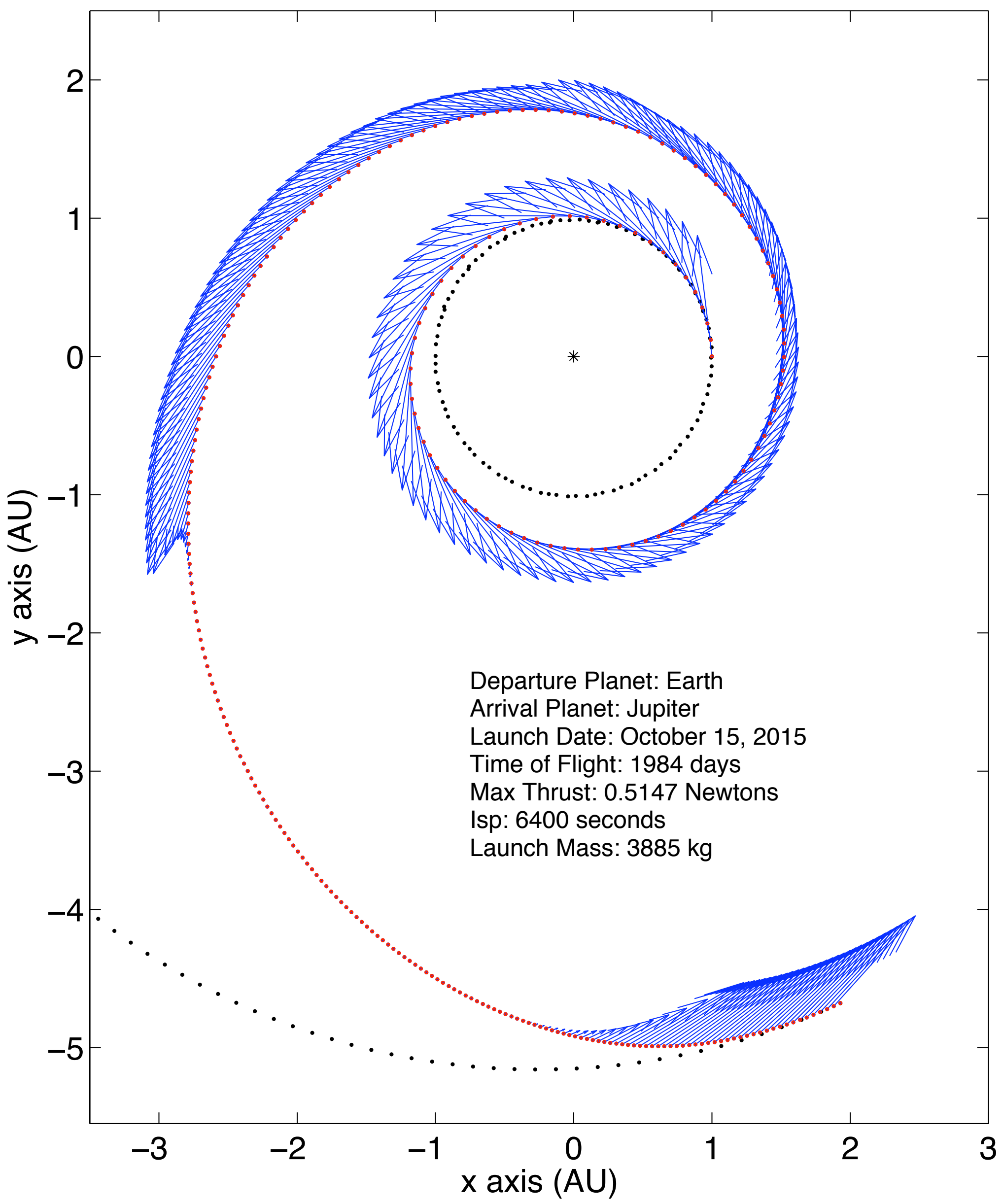


Figure 5.1: Fully integrated 5.5 year Earth to Jupiter trajectory. Initial guess is provided from Fig. 4.4. The jet power is 32.28 kW. The power system is assumed to provide a constant source of power.

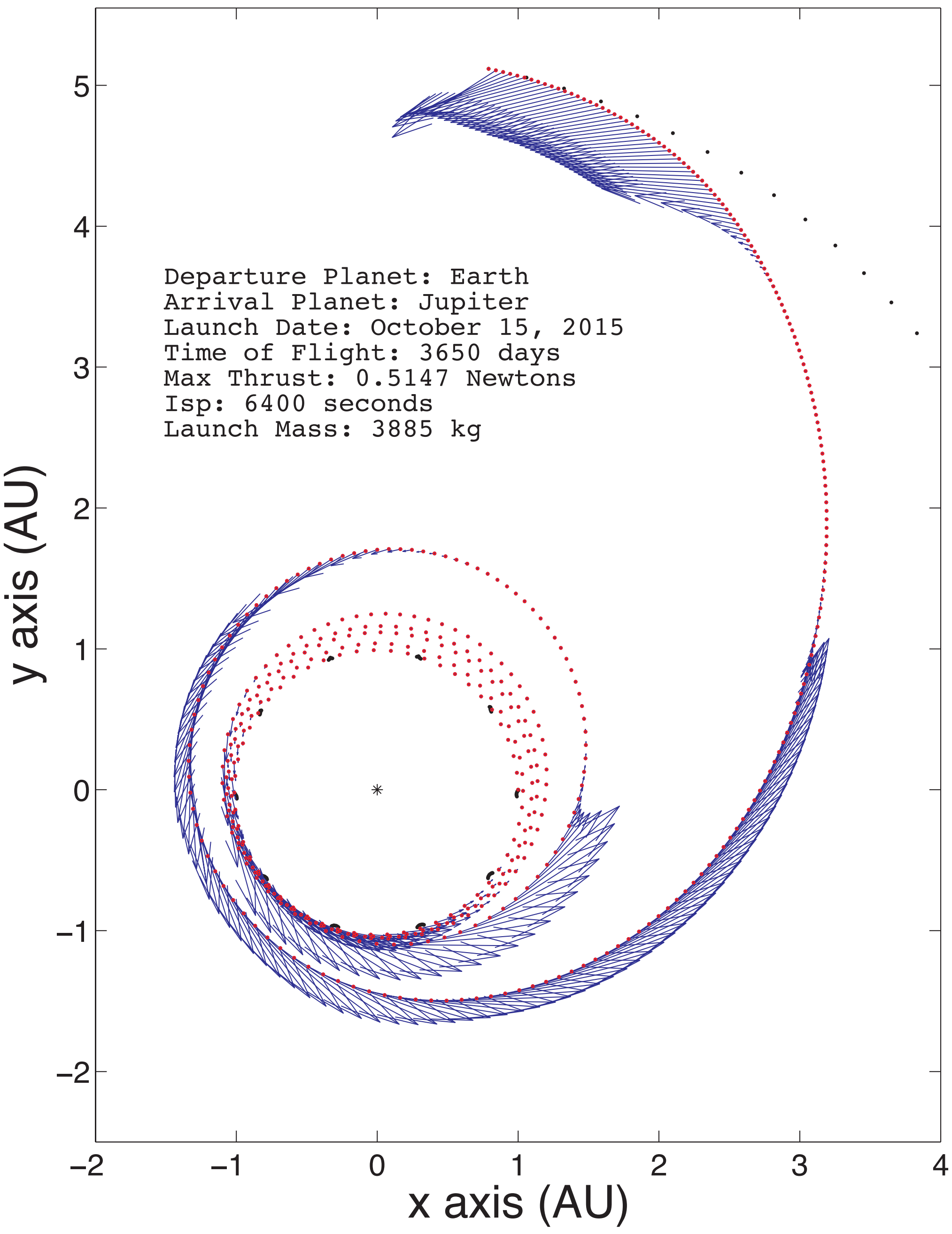


Figure 5.2: Fully integrated 10 year multi revolution Earth to Jupiter trajectory. Chebyshev approximation is provided Fig. 4.5. The jet power is 32.28 kW. The power system is assumed to provide a constant source of power.

The results are shown in Fig. 5.3, Fig. 5.4, Fig. 5.5, and Fig. 5.6. The results in the figures are not optimized, and solar arrays provide power that vary with $|\boldsymbol{r}|^{-2}$. Although the trajectories are not optimized, the figures provide some interesting results. The BPT-4000 thruster performs the best when it is fully powered over the entire trajectory. Because of the NEXT thruster's high power requirements and higher specific impulse the final mass increases as power increases, but for low powers the BPT-4000 delivers a larger mass to Mars. The final mass delivered to Mars by the NSTAR thruster rapidly decreases as power decreases because the thruster is fully utilized; resulting in a trajectory that is marginally feasible.

Table 5.1: Table of launch vehicle properties. The table shows the launch vehicle name, $C_3 = 0$ launch mass and maximum C_3

Launch Vehicle	$C_3 = 0$ mass (kg)	Maximum C_3 (km/s)
Atlas V 401	3445	90
Atlas V 501	2680	70
Atlas V 511	3765	90
Atlas V 521	4545	90
Atlas V 531	5210	110
Atlas V 541	5820	120
Atlas V 551	6330	120
Delta IV 4040-12	2735	30
Delta IV 4050-19	9305	60
Delta IV D4450-14	4580	25

The examples demonstrate the solver can reuse a single Chebyshev trajectory to generate feasible trajectories with different thrusters and power constraints.

5.2.5 A Method for Handling Over Constrained Problems

Eq. (5.11) is a linear equation and will always have a solution, however $\boldsymbol{C}$ may be over constrained such that no solution exists. For the over constrained case it is beneficial to know how much thrust is required to make the solution feasible. When the problem is over constrained, no solution exits and a metric to quantify

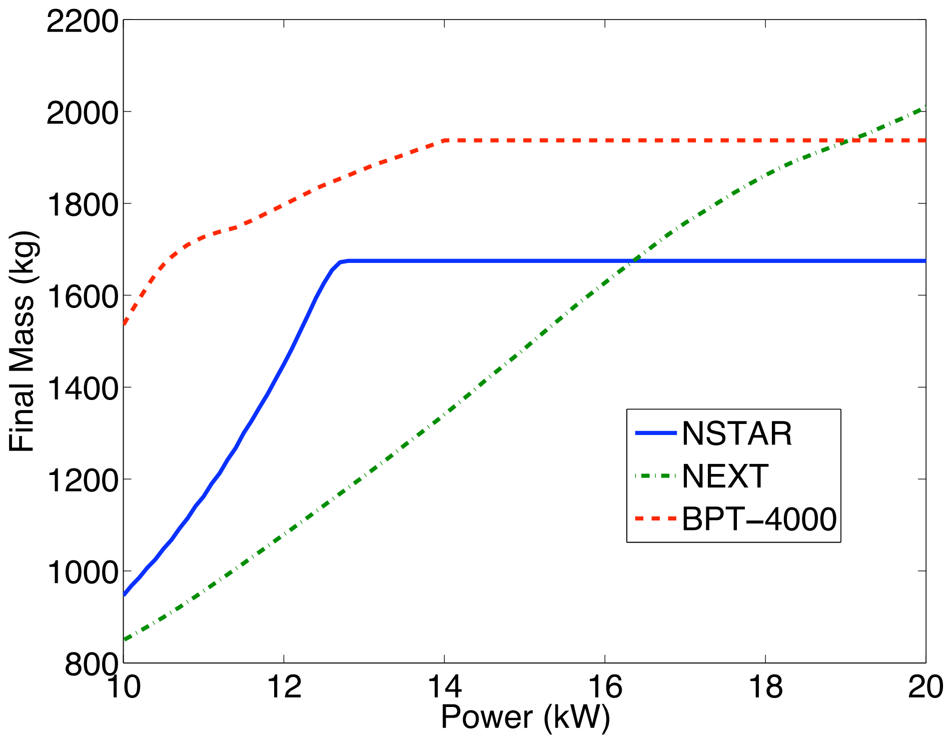


Figure 5.3: The final mass for a series of feasible trajectories for a 500 Earth to Mars mission. The x axis is the initial power at 1 AU. Solar arrays are used as the power source. Two thrusters are used.

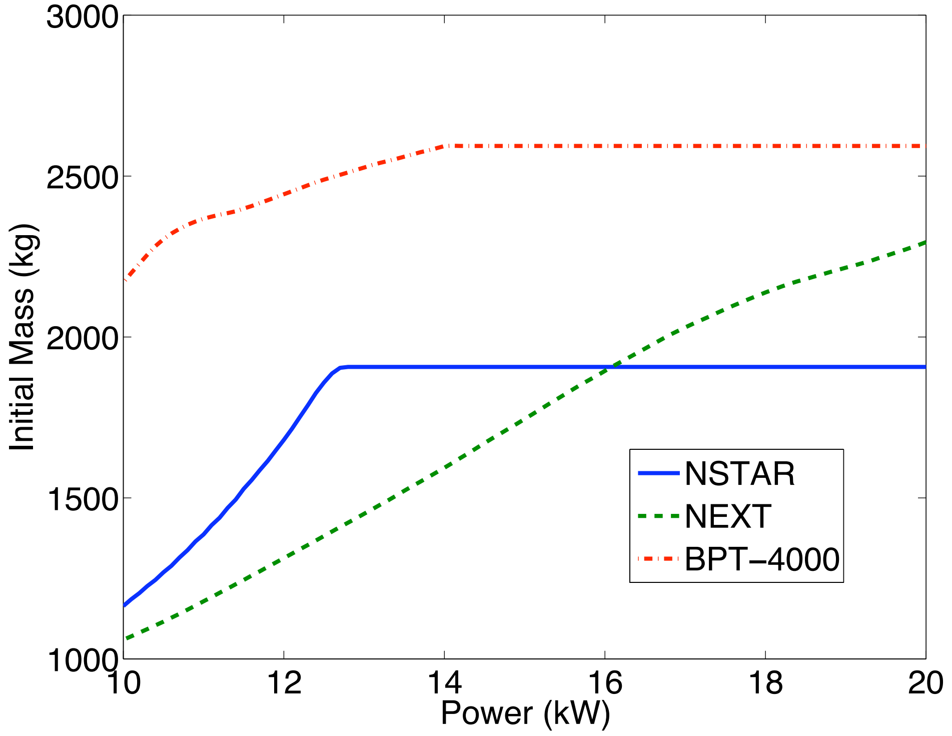


Figure 5.4: Initial mass for a 500 day Earth to Mars trajectories. The x axis is the initial power at 1 AU. Solar arrays are the power source. The example utilizes 2 thrusters.

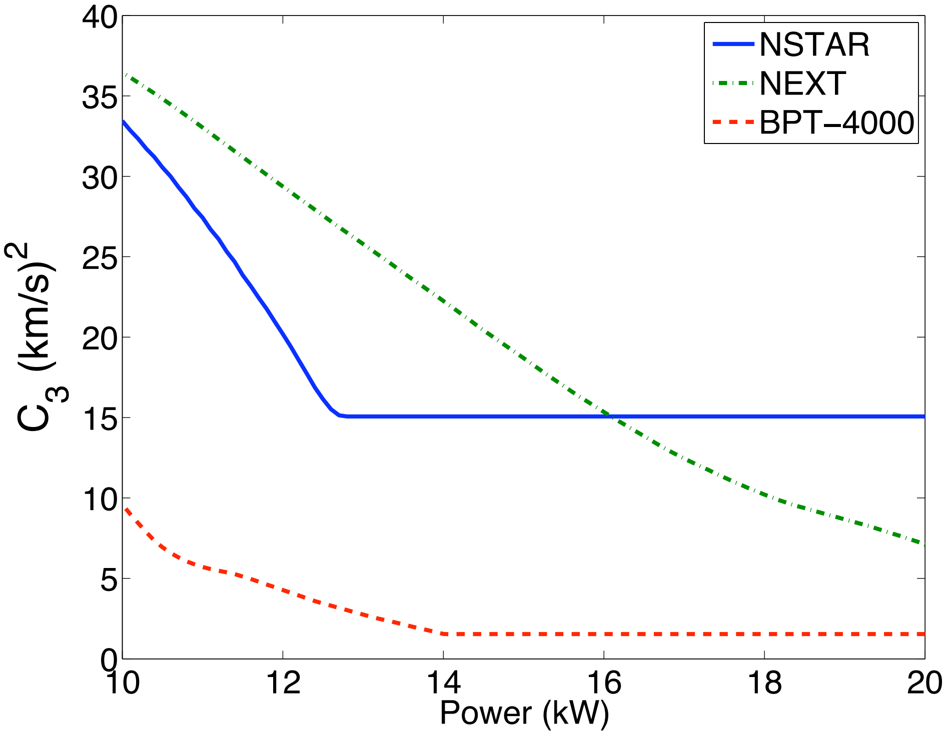


Figure 5.5: The C_3 as a function of the initial power for an Earth to Mars 500 day mission. The example utilizes 2 thrusters.

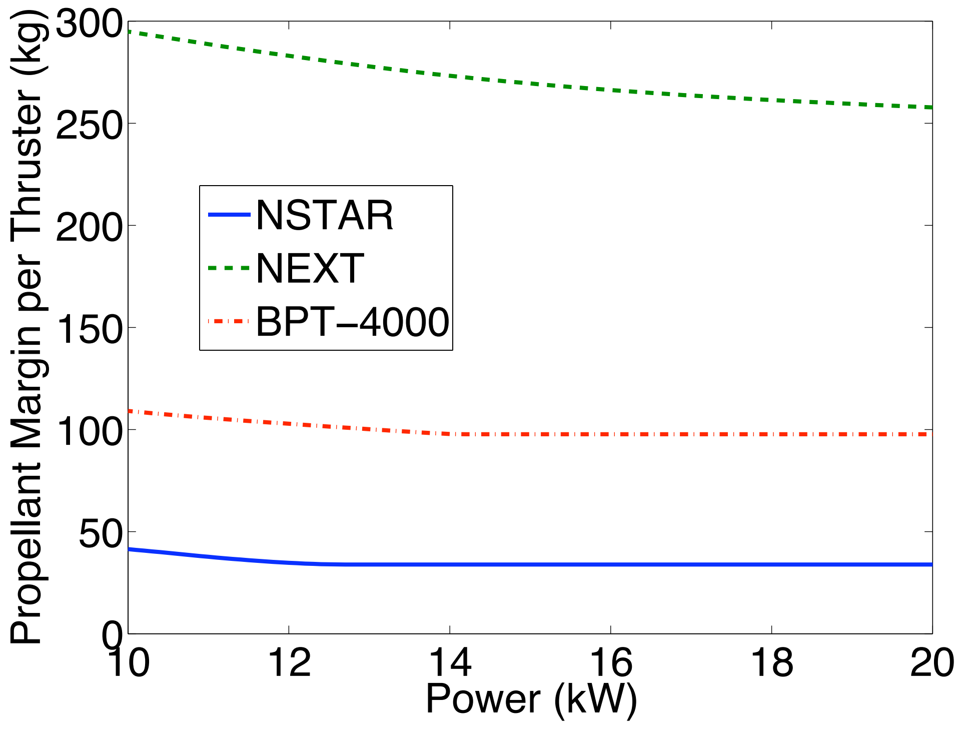


Figure 5.6: Propellant mass throughput margin remaining per thruster. Negative mass values would indicate that the throughput limit on the thruster has been violated.

the infeasibility is needed. Normally, the infeasibility is quantified as $\boldsymbol{C}^T\boldsymbol{C}$[16, 35]. Because $\boldsymbol{C}$ is a mixture of constraints, this does not provide any real insight as to how far the infeasible solution is from being feasible.

Instead of returning the norm of the constraint violations, $\boldsymbol{C}$, when the solver fails, it attempts to answer the question, "how much thrust is required for a successful mission?" The solver uses a homotopy method to decrease the maximum thrust level until an infeasible solution found. When an infeasible solution is found the last feasible solution is the minimum thrust required for the mission.

In this example, a solar power source is utilized, and the maximum thrust is 92 mN ,and the specific impulse is 3100 seconds. The initial jet power is 2.7 kW and scales with Eq. (5.19). The time of flight for the Earth to Mars trip is 600 days, the C_3 is zero, and the launch mass is 500 kg. The C_3 and launch mass are fixed to prevent the solver from reducing the launch mass or increasing the C_3 to satisfy the constraints. The requested thrust level is 75 mN. The solver returns Fig. 5.7. Fig. 5.7 shows that the thrust level for a feasible trajectory is about 90 mN, so the thruster needs to provide 15 mN of extra thrust to make the mission feasible. In this case, the solver starts with a maximum thrust level of 119 mN then decreases the thrust in 1 mN increments. The solver fails at 84 mN so the 85 mN trajectory is the minimum thrust solution.

5.3 Conclusions

In this chapter, automated methods for finding feasible trajectories have been described. The algorithms use simple numerical methods, but generate feasible trajectories that can either be used for analysis or fed to optimization routines. The algorithm uses the Chebyshev approximation to generate a thrust profile, which is

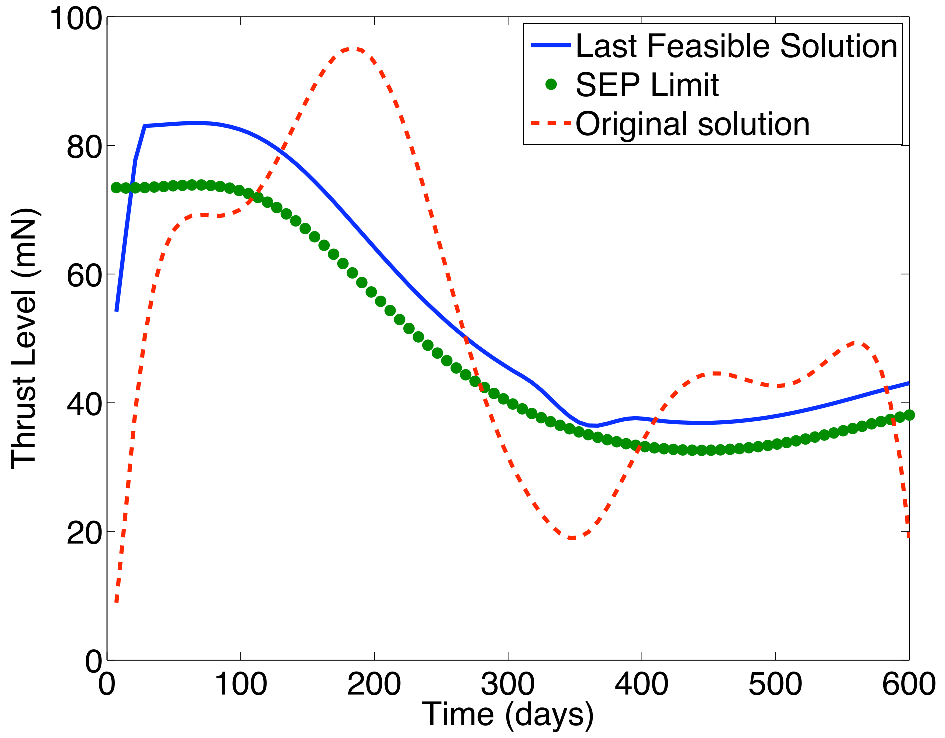


Figure 5.7: The thrust profile for a 600 day Earth to Mars mission. The thrust and C_3 are fixed at 500 kg and zero respectively. The power is supplied by solar arrays. Using the psuedoinverse approach the solver attempts to converge on the closest thrust profile that satisfies the constraints. Because the constraint cannot be satisfied the solver returns the last feasible solution.

then converted into a feasible trajectory. An infeasible to feasible solver is also described that detects infeasible problems and returns the minimum thrust level. While the trajectories generated in this section are feasible, they are not optimal. For low level trade studies optimal trajectories are needed to quantify the benefits of different thrusters.

CHAPTER VI

SECOND ORDER OPTIMIZATION ALGORITHM

6.1 Introduction

The previous chapter described how feasible trajectories could be generated and an algorithm for measuring the distance, in thrust, between feasible and infeasible trajectories. While generating feasible trajectories is useful, ultimately, the final goal is optimization. Once a feasible trajectory is found, the next stage is optimizing the trajectory. The feasible trajectory acts as a good initial guess to the optimizer since the boundary constraints are satisfied.

In this chapter a new optimization algorithm is described that approximates the cost function and constraints to the second order. The algorithm differs from previous methods because it generates a second order control update that satisfies the constraints to the second order, which maintains feasibility. Current methods use a linear control update, which satisfies the constraints to the first order[38, 22, 21, 62, 59, 60, 31, 16, 2, 39].

The two different formulations[7] for trajectory optimization problems are a multi-stage formulation and a continuos formulation. The algorithm derived in this chapter optimizes multi-stage problems. Fig. 6.1 graphically shows the multistage process. In the multistage process, the states carry information from one stage to the next, while the controls act on a single stage. Variables that are constant for a given trajectory but affect multiple stages are called parameters. In the multi-stage

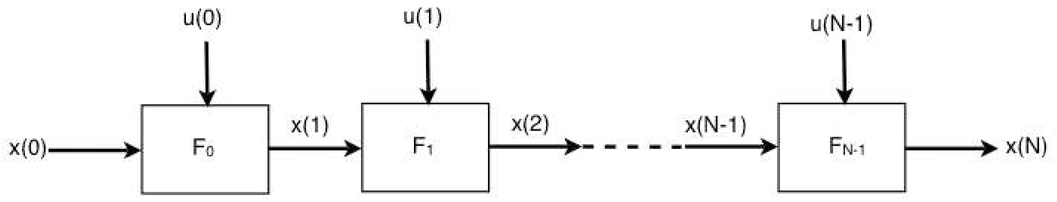


Figure 6.1: Shows how the states and controls are applied in a multistage process. x(i) and u(i) represent the state and control for a given stage. F(i) takes in x(i) and u(i) and outputs x(i+1).

process, the dynamics are governed by

$$\boldsymbol{x}\left(i+1\right)=\boldsymbol{F}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},i\right) \tag{6.1}$$

where $\boldsymbol{x}$ is the vector of the states, $\boldsymbol{u}$ is a vector of the controls, $\boldsymbol{p}$ is a vector of the parameters, and i is the stage. The cost function to be minimized is

$$J=\sum_{i=0}^{N-1}L\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},i\right)+\phi\left(\boldsymbol{x}^{N},\boldsymbol{p}\right) \tag{6.2}$$

The constraints are given by

$$0=q\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},i,j\right) \tag{6.3}$$

where j is an index enumerating the active constraints on the i^{th} segment.

In this chapter i is exclusively used to indicate the segment number, and j and k are used to index variables and functions on a single segment. For example, q^{ij} and q^{j} are read as the j^{th} constraint for the i^{th} segment. The i is dropped when it is understood that a particular equation or variable is acting on single stage.

The algorithm is designed to satisfy the Karush-Kuhn-Tucker (KKT) conditions[39] and uses the first and second order partials of the cost function, dynamics, and constraints. This allows it to construct a second order control update that satisfies the KKT conditions to the second order. This algorithm is an extension of sequential quadratic programming[39] method, however it differs from sequential quadratic programming because is requires second order dynamical and constraint information. The algorithm uses techniques that are similar to those used in differential dynamic programming[21, 22, 31, 38]. The algorithm approximates the constraints and cost function to the second order and generates a control update which satisfies the local KKT conditions and the constraints the second order. The control update on a particular segment has the form

$$\delta u^j = \delta u_p^j + \begin{bmatrix} U_{\boldsymbol{x}}^j & U_{\boldsymbol{p}}^j \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta x} \\ \boldsymbol{\delta p} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \boldsymbol{\delta x}^T & \boldsymbol{\delta p}^T \end{bmatrix} \begin{bmatrix} U_{\boldsymbol{xx}}^j & U_{\boldsymbol{xp}}^j \\ U_{\boldsymbol{px}}^j & U_{\boldsymbol{pp}}^j \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta x} \\ \boldsymbol{\delta p} \end{bmatrix} \tag{6.4}$$

The control update in Eq. (6.4) differs from other algorithms because it includes a second order term.

This chapter begins by outlining the algorithm and then discussing the control update. Next, several example problems are shown that demonstrate the benefits of the algorithm. Finally, current issues with the algorithm are highlighted.

6.2 Optimization Algorithm

This algorithm is a direct method that uses the principle of optimality[7]. The principal of optimality requires that every subarc on an optimal trajectory is also optimal. First, the initial guess is integrated, and the partials of the dynamics, constraints, and cost function are stored. The partials supplied by the user are

$$\begin{bmatrix} q^{ij}_{\boldsymbol{x}^i} \\ q^{ij}_{\boldsymbol{u}^i} \\ q^{ij}_{\boldsymbol{p}} \end{bmatrix} \begin{bmatrix} L^{i}_{\boldsymbol{x}^i} \\ L^{i}_{\boldsymbol{u}^i} \\ L^{i}_{\boldsymbol{p}} \end{bmatrix} \boldsymbol{F}^i_{\boldsymbol{x}^i} \;\; \boldsymbol{F}^i_{\boldsymbol{u}^i} \;\; \boldsymbol{F}^i_{\boldsymbol{p}} \tag{6.5}$$

$$\begin{bmatrix} q^{ij}_{\boldsymbol{x}^i\boldsymbol{x}^i} & q^{ij}_{\boldsymbol{x}^i\boldsymbol{u}^i} & q^{ij}_{\boldsymbol{x}^i\boldsymbol{p}} \\ q^{ij}_{\boldsymbol{u}^i\boldsymbol{x}^i} & q^{ij}_{\boldsymbol{u}^i\boldsymbol{u}^i} & q^{ij}_{\boldsymbol{u}^i\boldsymbol{p}} \\ q^{ij}_{\boldsymbol{p}\boldsymbol{x}^i} & q^{ij}_{\boldsymbol{p}\boldsymbol{u}^i} & q^{ij}_{\boldsymbol{p}\boldsymbol{p}} \end{bmatrix} \begin{bmatrix} L^{i}_{\boldsymbol{x}^i\boldsymbol{x}^i} & L^{i}_{\boldsymbol{x}^i\boldsymbol{u}^i} & L^{i}_{\boldsymbol{x}^i\boldsymbol{p}} \\ L^{i}_{\boldsymbol{u}^i\boldsymbol{x}^i} & L^{i}_{\boldsymbol{u}^i\boldsymbol{u}^i} & L^{i}_{\boldsymbol{u}^i\boldsymbol{p}} \\ L^{i}_{\boldsymbol{p}\boldsymbol{x}^i} & L^{i}_{\boldsymbol{p}\boldsymbol{u}^i} & L^{i}_{\boldsymbol{p}\boldsymbol{p}} \end{bmatrix} \tag{6.6}$$

$$\boldsymbol{F}^i_{\boldsymbol{x}^i\boldsymbol{x}^i} \;\; \boldsymbol{F}^i_{\boldsymbol{x}^i\boldsymbol{u}^i} \;\; \boldsymbol{F}^i_{\boldsymbol{x}^i\boldsymbol{p}} \;\; \boldsymbol{F}^i_{\boldsymbol{u}^i\boldsymbol{x}^i} \;\; \boldsymbol{F}^i_{\boldsymbol{u}^i\boldsymbol{u}^i} \;\; \boldsymbol{F}^i_{\boldsymbol{u}^i\boldsymbol{p}} \;\; \boldsymbol{F}^i_{\boldsymbol{p}\boldsymbol{x}^i} \;\; \boldsymbol{F}^i_{\boldsymbol{p}\boldsymbol{u}^i} \;\; \boldsymbol{F}^i_{\boldsymbol{p}\boldsymbol{p}} \tag{6.7}$$

The algorithm is designed to minimize the cost function, Eq. (6.3), but instead of using the cost function, the algorithm uses the cost-to-go given in Eq. (6.8). The cost-to-go function measures the cost associated with traveling from the current segment, i, to the final node N.

$$V^i = \sum_{n=i}^{N-1} L\left(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, n\right) + \phi\left(\boldsymbol{x}^N, \boldsymbol{p}\right) \tag{6.8}$$

The partials of the cost-to-go function are represented in Eq. (6.9), and they are generated by the algorithm.

$$\begin{bmatrix} V^{i}_{\boldsymbol{x}^i} \\ V^{i}_{\boldsymbol{u}^i} \\ V^{i}_{\boldsymbol{p}} \end{bmatrix}, \begin{bmatrix} V^{i}_{\boldsymbol{x}^i\boldsymbol{x}^i} & V^{i}_{\boldsymbol{x}^i\boldsymbol{u}^i} & V^{i}_{\boldsymbol{x}^i\boldsymbol{p}} \\ V^{i}_{\boldsymbol{u}^i\boldsymbol{x}^i} & V^{i}_{\boldsymbol{u}^i\boldsymbol{u}^i} & V^{i}_{\boldsymbol{u}^i\boldsymbol{p}} \\ V^{i}_{\boldsymbol{p}\boldsymbol{x}^i} & V^{i}_{\boldsymbol{p}\boldsymbol{u}^i} & V^{i}_{\boldsymbol{p}\boldsymbol{p}} \end{bmatrix} \tag{6.9}$$

An overview of the algorithm is outlined in Alg. (6.1). Before Alg. (6.1) can be implemented, several issues need to be explained. The issues are

1. How are the partials of V^i computed?

2. How is the control update generated?

Algorithm 6.1 High level overview of optimization algorithm

```
Use initial control
Set control update=0
while Trajectory not optimized do
  Apply control update and compute trajectory
  Store partials of $L$, $\boldsymbol{f}$, and $q^j$
  for $i = N - 1$ to 1 do
    Compute partials of $V^i$
    Compute control update
  end for
end while
```

3. What is the optimality condition?

4. How is the update applied?

The following sections will address these four issues.

6.3 Cost-To-Go Function

The cost-to-go function, Eq. (6.3), is an important concept in optimization[21, 22, 31, 38, 10]. The cost-to-go defines the cost incurred in travel from the current point to the final destination. This section addresses how to compute the partials of V, which will be used construct the second order control update. For multi-stage problems that use the cost-to-go function, the problem is worked backwards so that the final stage is handled first and the first stage is dealt with last.

Before a recursive method for constructing the partials of V can be provided, an auxiliary function, W, has to be introduced. It represents the cost-to-go function once the control update has been applied. In this sense, W is only a function of the states and parameters. Primarily, W is used to discriminate between partials that have been constructed using the control update and those that have not. At this point, it is assumed that the second order control update can be computed, however the next section describes a method for constructing the control update.

Alg. (6.2) is the recursive method for constructing the partials of V and W, which are required to construct the control update. The procedure begins by ini-

Algorithm 6.2 Expansion of cost-to-go

```
Initialize $W^N(\boldsymbol{x}, \boldsymbol{p})$
for i=N-1 to 1 do
    Expand $W^{i+1}(\boldsymbol{x}, \boldsymbol{p})$ to $V^i(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{u})$
    Apply control control and reduce $V^i(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{u})$ to $W^i(\boldsymbol{x}, \boldsymbol{p})$
end for
```

tializing the partials of W^N and its partials.

$$\begin{bmatrix} W^N_{\boldsymbol{x}^N} \\ W^N_{\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \phi^N_{\boldsymbol{x}^N} \\ \phi^N_{\boldsymbol{p}} \end{bmatrix}, \begin{bmatrix} W^N_{\boldsymbol{x}^N\boldsymbol{x}^N} & W^N_{\boldsymbol{x}^N\boldsymbol{p}} \\ W^N_{\boldsymbol{p}\boldsymbol{x}^N} & W^N_{\boldsymbol{p}\boldsymbol{p}} \end{bmatrix} = \begin{bmatrix} \phi^N_{\boldsymbol{x}^N\boldsymbol{x}^N} & \phi^N_{\boldsymbol{x}^N\boldsymbol{p}} \\ \phi^N_{\boldsymbol{p}\boldsymbol{x}^N} & \phi^N_{\boldsymbol{p}\boldsymbol{p}} \end{bmatrix} \tag{6.10}$$

Next, the partials of W^{i+1} need to be converted into the partials of V^i. In the following equations $W = W^{i+1}$, $V = V^i$, $\boldsymbol{F} = \boldsymbol{F}^i$. Derivatives are also taken with respect to the appropriate index, so $\boldsymbol{F}_{\boldsymbol{x}} = \boldsymbol{F}^i_{\boldsymbol{x}^i}$. Eq. (6.11) and Eq. (6.12) map the partials of W^{i+1} to V^i.

$$\begin{bmatrix} V_{\boldsymbol{x}} \\ V_{\boldsymbol{p}} \\ V_{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}^T_{\boldsymbol{x}} & 0 \\ \boldsymbol{F}^T_{\boldsymbol{p}} & I \\ \boldsymbol{F}^T_{\boldsymbol{u}} & 0 \end{bmatrix} \begin{bmatrix} W_{\boldsymbol{x}} \\ W_{\boldsymbol{p}} \end{bmatrix} + \begin{bmatrix} L_{\boldsymbol{x}} \\ L_{\boldsymbol{p}} \\ L_{\boldsymbol{u}} \end{bmatrix} \tag{6.11}$$

$$\begin{bmatrix} V_{\boldsymbol{xx}} & V_{\boldsymbol{xp}} & V_{\boldsymbol{xu}} \\ V_{\boldsymbol{px}} & V_{\boldsymbol{pp}} & V_{\boldsymbol{pu}} \\ V_{\boldsymbol{ux}} & V_{\boldsymbol{up}} & V_{\boldsymbol{uu}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{x}}^T & 0 \\ \boldsymbol{F}_{\boldsymbol{p}}^T & I \\ \boldsymbol{F}_{\boldsymbol{u}}^T & 0 \end{bmatrix} \begin{bmatrix} W_{\boldsymbol{xx}} & W_{\boldsymbol{xp}} \\ W_{\boldsymbol{px}} & W_{\boldsymbol{pp}} \end{bmatrix} \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{x}}^T & \boldsymbol{F}_{\boldsymbol{p}}^T & \boldsymbol{F}_{\boldsymbol{u}}^T \\ 0 & I & 0 \end{bmatrix} + \begin{bmatrix} L_{\boldsymbol{xx}} & L_{\boldsymbol{xp}} & L_{\boldsymbol{xu}} \\ L_{\boldsymbol{px}} & L_{\boldsymbol{pp}} & L_{\boldsymbol{pu}} \\ L_{\boldsymbol{ux}} & L_{\boldsymbol{up}} & L_{\boldsymbol{uu}} \end{bmatrix} + \sum_j W_{\boldsymbol{x}^j} \cdot \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{xx}}^j & \boldsymbol{F}_{\boldsymbol{xp}}^j & \boldsymbol{F}_{\boldsymbol{xu}}^j \\ \boldsymbol{F}_{\boldsymbol{px}}^j & \boldsymbol{F}_{\boldsymbol{pp}}^j & \boldsymbol{F}_{\boldsymbol{pu}}^j \\ \boldsymbol{F}_{\boldsymbol{ux}}^j & \boldsymbol{F}_{\boldsymbol{up}}^j & \boldsymbol{F}_{\boldsymbol{uu}}^j \end{bmatrix} \tag{6.12}$$

Eq. (6.11) and Eq. (6.12) use W^{i+1} to construct the expansion of the cost-to-go function. The next step in Alg. (6.2) is to use the control update and V^i to construct the partials of W^i.

Applying the control update from Eq. (6.4) to the partials of the cost-to-go function reduces it to a function of the states and parameters. In the following set of equations $W = W^i$, $V = V^i$, $\boldsymbol{\delta u}_p = \begin{bmatrix} \delta u_p^1 \; \delta u_p^2 \; \dots \; \delta u_p^m \end{bmatrix}$,

$$\boldsymbol{U}_{\boldsymbol{x}} = \begin{bmatrix} U_{\boldsymbol{x}}^1 \\ U_{\boldsymbol{x}}^2 \\ \vdots \\ U_{\boldsymbol{x}}^m \end{bmatrix} \tag{6.13}$$

$$\boldsymbol{U}_{\boldsymbol{p}} = \begin{bmatrix} U_{\boldsymbol{p}}^1 \\ U_{\boldsymbol{p}}^2 \\ \vdots \\ U_{\boldsymbol{p}}^m \end{bmatrix} \tag{6.14}$$

Applying the control updates to the partials of V gives

$$W_{\boldsymbol{x}} = V_{\boldsymbol{x}} + \boldsymbol{U_x}^T V_{\boldsymbol{u}} + V_{\boldsymbol{xu}} \boldsymbol{\delta u}_p + \boldsymbol{U_x}^T V_{\boldsymbol{uu}} \boldsymbol{\delta u}_p \tag{6.15}$$

$$W_{\boldsymbol{p}} = V_{\boldsymbol{p}} + \boldsymbol{U_p}^T V_{\boldsymbol{u}} + V_{\boldsymbol{pu}} \boldsymbol{\delta u}_p + \boldsymbol{U_p}^T V_{\boldsymbol{uu}} \boldsymbol{\delta u}_p \tag{6.16}$$

$$W_{\boldsymbol{xx}} = V_{\boldsymbol{xx}} + V_{\boldsymbol{xu}} \boldsymbol{U_x} + \boldsymbol{U_x}^T V_{\boldsymbol{uu}} \boldsymbol{U_x} + \sum_j U^j_{\boldsymbol{xx}} \left(V_{\boldsymbol{u}} + V_{\boldsymbol{uu}} \boldsymbol{\delta u}_p \right)^j \tag{6.17}$$

$$W_{\boldsymbol{pp}} = V_{\boldsymbol{pp}} + V_{\boldsymbol{pu}} \boldsymbol{U_p} + \boldsymbol{U_p}^T V_{\boldsymbol{uu}} \boldsymbol{U_p} + \sum_j U^j_{\boldsymbol{pp}} \left(V_{\boldsymbol{u}} + V_{\boldsymbol{uu}} \boldsymbol{\delta u}_p \right)^j \tag{6.18}$$

$$W_{\boldsymbol{xp}} = V_{\boldsymbol{xp}} + V_{\boldsymbol{xu}} \boldsymbol{U_p} + \frac{1}{2} \boldsymbol{U_x}^T V_{\boldsymbol{uu}} \boldsymbol{U_p} + \sum_j U^j_{\boldsymbol{xp}} \left(V_{\boldsymbol{u}} + V_{\boldsymbol{uu}} \boldsymbol{\delta u}_p \right)^j \tag{6.19}$$

$$W_{\boldsymbol{px}} = V_{\boldsymbol{px}} + V_{\boldsymbol{pu}} \boldsymbol{U_x} + \frac{1}{2} \boldsymbol{U_p}^T V_{\boldsymbol{uu}} \boldsymbol{U_x} + \sum_j U^j_{\boldsymbol{px}} \left(V_{\boldsymbol{u}} + V_{\boldsymbol{uu}} \boldsymbol{\delta u}_p \right)^j \tag{6.20}$$

This section has outlined a recursive procedure for expanding and reducing the partials of the cost-to-go function. This expansion is used in the next section to construct the control update.

6.4 Second Order Control Update

This section constructs the control update for the i^{th} segment. The formulation begins by appending the constraints to the cost-to-go function, Eq. (6.8), giving

$$V^{i*} = V^i + \sum_j \lambda^j q^j \tag{6.21}$$

First, a second order Taylor series expansion of Eq. (6.21) is constructed with respect to $\boldsymbol{x}$, $\boldsymbol{u}$, and $\boldsymbol{p}$. Then, the Taylor series expansion is differentiated with respect to $\boldsymbol{\delta u}$ and λ^j, which gives the necessary conditions, Eq. (6.23) and Eq. (6.24).

$$0 = V_{\boldsymbol{u}} + 0.5 \left(V_{\boldsymbol{ux}} + V^T_{\boldsymbol{xu}} \right) \boldsymbol{\delta x} + 0.5 \left(V_{\boldsymbol{up}} + V^T_{\boldsymbol{pu}} \right) \boldsymbol{\delta p} + V_{\boldsymbol{uu}} \boldsymbol{\delta u} + \tag{6.22}$$
$$\sum_j \lambda^j \left(q^j_{\boldsymbol{u}} + 0.5 \left(q^j_{\boldsymbol{ux}} + q^{jT}_{\boldsymbol{xu}} \right) \boldsymbol{\delta x} + 0.5 \left(q^j_{\boldsymbol{up}} + q^{jT}_{\boldsymbol{pu}} \right) \boldsymbol{\delta p} + q^j_{\boldsymbol{uu}} \boldsymbol{\delta u} \right)$$

and

$$0 = q^j + q_{\boldsymbol{u}}^{jT}\boldsymbol{\delta u} + 0.5\boldsymbol{\delta u}^T q_{\boldsymbol{uu}}^j \boldsymbol{\delta u} + 0.5\boldsymbol{\delta x}^T q_{\boldsymbol{xu}}^j \boldsymbol{\delta u} + 0.5\boldsymbol{\delta u}^T q_{\boldsymbol{ux}}^j \boldsymbol{\delta x} + \quad (6.23)$$
$$0.5\boldsymbol{\delta x}^T q_{\boldsymbol{xx}}^j \boldsymbol{\delta x} + q_{\boldsymbol{p}}^{jT}\boldsymbol{\delta p} + 0.5\boldsymbol{\delta p}^T q_{\boldsymbol{pu}}^j \boldsymbol{\delta u} + 0.5\delta x^T q_{\boldsymbol{xp}}^j \boldsymbol{\delta p} + 0.5\boldsymbol{\delta p}^T q_{\boldsymbol{pp}}^j \boldsymbol{\delta p} +$$
$$q_{\boldsymbol{x}}^{jT}\boldsymbol{\delta x} + 0.5\boldsymbol{\delta x}^T q_{\boldsymbol{xp}}^j \boldsymbol{\delta p} + 0.5\boldsymbol{\delta p}^T q_{\boldsymbol{px}}^j \boldsymbol{\delta x}$$

The solution to Eq. (6.23) and Eq. (6.24) is control update that will satisfy the necessary conditions to the second order. Since $\boldsymbol{\delta x}$ and $\boldsymbol{\delta p}$ are unknown for a particular stage, the control update needs to be a function of $\boldsymbol{\delta x}$ and $\boldsymbol{\delta p}$. For this reason, the control update takes the form in Eq. (6.4), which is a truncated series expansion for the true solution.

The control update is constructed term by term. First the affine term, $\boldsymbol{\delta u}_p$, is constructed by replacing $\boldsymbol{\delta u}$ and $\boldsymbol{\lambda}$ with $\boldsymbol{\delta u}_p$ and $\boldsymbol{\lambda}_p$. The resulting equations are

$$0 = \begin{bmatrix} V_{\boldsymbol{u}} \\ q^1 \\ \vdots \\ q^m \end{bmatrix} + (M + 0.5H)\begin{bmatrix} \boldsymbol{\delta u_p} \\ \boldsymbol{\lambda}_p \end{bmatrix} \quad (6.24)$$

where

$$M = \begin{bmatrix} V_{\boldsymbol{uu}} & q_{\boldsymbol{u}}^1 & \dots & q_{\boldsymbol{u}}^m \\ q_{\boldsymbol{u}}^{1T} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \\ q_{\boldsymbol{u}}^{mT} & 0 & \dots & 0 \end{bmatrix} \quad (6.25)$$

and

$$H = \begin{bmatrix} \sum \lambda_p^j q_{\boldsymbol{uu}}^j & q_{\boldsymbol{uu}}^1 \boldsymbol{\delta u_p} & \dots & q_{\boldsymbol{uu}}^m \boldsymbol{\delta u_p} \\ \boldsymbol{\delta u_p}^T q_{\boldsymbol{uu}}^1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\delta u_p}^T q_{\boldsymbol{uu}}^m & 0 & \dots & 0 \end{bmatrix} \quad (6.26)$$

Here m represents the number of constraints on a particular segment.

Next, the linear terms are collected. $\boldsymbol{\Lambda}$ is defined as the set of λ values on the optimal trajectory. So, like $\boldsymbol{U}$, the appropriate partials to $\boldsymbol{\Lambda}$ must be constructed. Collecting the linear terms gives

$$0 = \begin{bmatrix} 0.5V_{\boldsymbol{ux}} + 0.5V_{\boldsymbol{xu}}^T \\ q_{\boldsymbol{x}}^{1T} \\ \vdots \\ q_{\boldsymbol{x}}^{mT} \end{bmatrix} + \begin{bmatrix} 0.5\sum \lambda_p^j \left(q_{\boldsymbol{ux}}^j + q_{\boldsymbol{xu}}^{jT}\right) \\ 0.5\boldsymbol{\delta u}_{\boldsymbol{p}}^T \left(q_{\boldsymbol{ux}}^1 + q_{\boldsymbol{xu}}^{1T}\right) \\ \vdots \\ 0.5\boldsymbol{\delta u}_{\boldsymbol{p}}^T \left(q_{\boldsymbol{ux}}^m + q_{\boldsymbol{xu}}^{mT}\right) \end{bmatrix} + (M+H) \begin{bmatrix} \boldsymbol{U}_{\boldsymbol{x}} \\ \boldsymbol{\Lambda}_{\boldsymbol{x}} \end{bmatrix} \tag{6.27}$$

$$0 = \begin{bmatrix} 0.5V_{\boldsymbol{up}} + 0.5V_{\boldsymbol{pu}}^T \\ q_{\boldsymbol{p}}^{1T} \\ \vdots \\ q_{\boldsymbol{p}}^{mT} \end{bmatrix} + \begin{bmatrix} 0.5\sum \lambda_p^j \left(q_{\boldsymbol{up}}^j + q_{\boldsymbol{pu}}^{jT}\right) \\ 0.5\boldsymbol{\delta u}_{\boldsymbol{p}}^T \left(q_{\boldsymbol{up}}^1 + q_{\boldsymbol{pu}}^{1T}\right) \\ \vdots \\ 0.5\boldsymbol{\delta u}_{\boldsymbol{p}}^T \left(q_{\boldsymbol{up}}^m + q_{\boldsymbol{pu}}^{mT}\right) \end{bmatrix} + (M+H) \begin{bmatrix} \boldsymbol{U}_p \\ \boldsymbol{\Lambda}_p \end{bmatrix} \tag{6.28}$$

With the linear terms found the second order terms can be constructed. In order to construct the second order partials the following definitions are needed

$$\boldsymbol{\Lambda}_{\boldsymbol{x}^j}^k = \frac{\partial \lambda^k}{\partial \boldsymbol{x}^j} \tag{6.29}$$

and

$$\boldsymbol{U}_{\boldsymbol{x}^j \boldsymbol{x}} = \frac{\partial^2 \boldsymbol{U}}{\partial \boldsymbol{x}^j \partial \boldsymbol{x}} \tag{6.30}$$

Using this notation, $\boldsymbol{\Lambda}_{\boldsymbol{x}^j}^k$ is the partial derivative of k^{th} element of the vector $\boldsymbol{\lambda}$ with respect to the j^{th} element of the vector $\boldsymbol{x}$. Similarly, $\boldsymbol{U}_{\boldsymbol{x}^j \boldsymbol{x}}$ is the second partial derivative of the control vector $\boldsymbol{U}$ with respect to the vector $\boldsymbol{x}$ and the j^{th} element of the vector $\boldsymbol{x}$. Applying this notation, the second order control update is

$$0 = \begin{bmatrix} \sum_k \Lambda^k_{\boldsymbol{x}^j}\left(q^k_{\boldsymbol{ux}} + q^{kT}_{\boldsymbol{ux}} + 2q^k_{\boldsymbol{uu}}\boldsymbol{U_x}\right) \\ \boldsymbol{U}^T_{\boldsymbol{x}^j} q^1_{\boldsymbol{ux}} + q^1_{\boldsymbol{x}^j\boldsymbol{u}}\boldsymbol{U_x} + q^1_{\boldsymbol{x}^j\boldsymbol{x}} + \boldsymbol{U_{x^j}}^T q^1_{\boldsymbol{uu}}\boldsymbol{U_x} \\ \boldsymbol{U}^T_{\boldsymbol{x}^j} q^2_{\boldsymbol{ux}} + q^2_{\boldsymbol{x}^j\boldsymbol{u}}\boldsymbol{U_x} + q^2_{\boldsymbol{x}^j\boldsymbol{x}} + \boldsymbol{U_{x^j}}^T q^2_{\boldsymbol{uu}}\boldsymbol{U_x} \\ \vdots \\ \boldsymbol{U}^T_{\boldsymbol{x}^j} q^m_{\boldsymbol{ux}} + q^m_{\boldsymbol{x}^j\boldsymbol{u}}\boldsymbol{U_x} + q^m_{\boldsymbol{x}^j\boldsymbol{x}} + \boldsymbol{U_{x^j}}^T q^m_{\boldsymbol{uu}}\boldsymbol{U_x} \end{bmatrix} + (M+H)\begin{bmatrix} \boldsymbol{U}_{\boldsymbol{x}^j\boldsymbol{x}} \\ \boldsymbol{\Lambda}_{\boldsymbol{x}^j\boldsymbol{x}} \end{bmatrix} \tag{6.31}$$

$$0 = \begin{bmatrix} \sum_k \Lambda^k_{\boldsymbol{p}^j}\left(q^k_{\boldsymbol{up}} + q^{kT}_{\boldsymbol{up}} + 2q^k_{\boldsymbol{uu}}\boldsymbol{U_p}\right) \\ \boldsymbol{U_{p^j}}^T q^1_{\boldsymbol{up}} + q^1_{\boldsymbol{p}^j\boldsymbol{u}}\boldsymbol{U_p} + q^1_{\boldsymbol{p}^j\boldsymbol{p}} + \boldsymbol{U_{p^j}}^T q^1_{\boldsymbol{uu}}\boldsymbol{U_p} \\ \boldsymbol{U_{p^j}}^T q^2_{\boldsymbol{up}} + q^2_{\boldsymbol{p}^j\boldsymbol{u}}\boldsymbol{U_p} + q^2_{\boldsymbol{p}^j\boldsymbol{p}} + \boldsymbol{U_{p^j}}^T q^2_{\boldsymbol{uu}}\boldsymbol{U_p} \\ \vdots \\ \boldsymbol{U_{p^j}}^T q^m_{\boldsymbol{up}} + q^m_{\boldsymbol{p}^j\boldsymbol{u}}\boldsymbol{U_p} + q^m_{\boldsymbol{p}^j\boldsymbol{p}} + \boldsymbol{U_{p^j}}^T q^m_{\boldsymbol{uu}}\boldsymbol{U_p} \end{bmatrix} + (M+H)\begin{bmatrix} \boldsymbol{U}_{\boldsymbol{p}^j\boldsymbol{p}} \\ \boldsymbol{\Lambda}_{\boldsymbol{p}^j\boldsymbol{p}} \end{bmatrix} \tag{6.32}$$

$$0 = \begin{bmatrix} \sum_k \Lambda^k_{\boldsymbol{p}^j}\left(q^k_{\boldsymbol{ux}} + q^{kT}_{\boldsymbol{ux}} + 2q^k_{\boldsymbol{uu}}\boldsymbol{U_x}\right) \\ \boldsymbol{U_{p^j}}^T q^1_{\boldsymbol{ux}} + q^1_{\boldsymbol{p}^j\boldsymbol{u}}\boldsymbol{U_x} + q^1_{\boldsymbol{p}^j\boldsymbol{x}} + \boldsymbol{U_{p^j}}^T q^1_{\boldsymbol{uu}}\boldsymbol{U_x} \\ \boldsymbol{U_{p^j}}^T q^2_{\boldsymbol{ux}} + q^2_{\boldsymbol{p}^j\boldsymbol{u}}\boldsymbol{U_x} + q^2_{\boldsymbol{p}^j\boldsymbol{x}} + \boldsymbol{U_{p^j}}^T q^2_{\boldsymbol{uu}}\boldsymbol{U_x} \\ \vdots \\ \boldsymbol{U_{p^j}}^T q^m_{\boldsymbol{ux}} + q^m_{\boldsymbol{p}^j\boldsymbol{u}}\boldsymbol{U_x} + q^m_{\boldsymbol{p}^j\boldsymbol{x}} + \boldsymbol{U_{p^j}}^T q^m_{\boldsymbol{uu}}\boldsymbol{U_x} \end{bmatrix} + (M+H)\begin{bmatrix} \boldsymbol{U}_{\boldsymbol{p}^j\boldsymbol{x}} \\ \boldsymbol{\Lambda}_{\boldsymbol{p}^j\boldsymbol{x}} \end{bmatrix} \tag{6.33}$$

$$0 = \begin{bmatrix} \sum_k \Lambda^k_{\boldsymbol{x}^j}\left(q^k_{\boldsymbol{up}} + q^{kT}_{\boldsymbol{up}} + 2q^k_{\boldsymbol{uu}}\boldsymbol{U_p}\right) \\ \boldsymbol{U}^T_{\boldsymbol{x}^j} q^1_{\boldsymbol{up}} + q^1_{\boldsymbol{x}^j\boldsymbol{u}}\boldsymbol{U_p} + q^1_{\boldsymbol{x}^j\boldsymbol{p}} + \boldsymbol{U_{x^j}}^T q^1_{\boldsymbol{uu}}\boldsymbol{U_p} \\ \boldsymbol{U}^T_{\boldsymbol{x}^j} q^2_{\boldsymbol{up}} + q^2_{\boldsymbol{x}^j\boldsymbol{u}}\boldsymbol{U_p} + q^2_{\boldsymbol{x}^j\boldsymbol{p}} + \boldsymbol{U_{x^j}}^T q^2_{\boldsymbol{uu}}\boldsymbol{U_p} \\ \vdots \\ \boldsymbol{U}^T_{\boldsymbol{x}^j} q^m_{\boldsymbol{up}} + q^m_{\boldsymbol{x}^j\boldsymbol{u}}\boldsymbol{U_p} + q^m_{\boldsymbol{x}^j\boldsymbol{p}} + \boldsymbol{U_{x^j}}^T q^m_{\boldsymbol{uu}}\boldsymbol{U_p} \end{bmatrix} + (M+H)\begin{bmatrix} \boldsymbol{U}_{\boldsymbol{x}^j\boldsymbol{p}} \\ \boldsymbol{\Lambda}_{\boldsymbol{x}^j\boldsymbol{p}} \end{bmatrix} \tag{6.34}$$

The solution to equations Eq. (6.24)-Eq. (6.34) is the control update. Solving Eq. (6.24) requires an iterative numerical method because it is a set of nonlinear coupled equations. However, the linear and second order terms are linear and can be solved using a linear algebra package.

6.4.1 Optimality Condition

The optimality condition for a segment, $\boldsymbol{O}_c^i$, is given

$$\boldsymbol{O}_c^i = \begin{bmatrix} \boldsymbol{V_u^i} \\ q^1 \\ \vdots \\ q^m \end{bmatrix} + \begin{bmatrix} q_{\boldsymbol{u}}^{1T} & \dots & q_{\boldsymbol{u}}^{mT} \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \boldsymbol{\lambda}_p \tag{6.35}$$

The trajectory is considered optimal when for all i,

$$\left|\boldsymbol{O}_c^i\right| \leq \epsilon \tag{6.36}$$

where ϵ is some small positive number. The constraint violation is simply

$$\begin{bmatrix} q^1 \\ q^2 \\ \vdots \\ q^m \end{bmatrix} \tag{6.37}$$

6.5 Application of the Control Update

Once the control updates have been computed, there are two possible methods of applying them. The first method is to simply compute the state and parameter deviations and then compute the control update from Eq. (6.4). This is the simplest and easiest method to use because it requires little computational effort and has no possibility of failure.

Another approach is to compute the state and parameter deviations then generate the control update by iteratively solving Eq. (6.23) and Eq. (6.24). Solving these two equations results in a more accurate control update since Eq. (6.4) is a truncated series solution to Eq. (6.23) and Eq. (6.24). However, because Eq. (6.23) and Eq. (6.24) must be solved iteratively, there is a possibility that the solver may not converge.

6.6 Single Stage Variant

The algorithm above outlines a recursive process for the multi-stage process, however the multi-stage optimization problem can be converted into an equivalent single stage process. In the single stage variant the entire process is treated as one stage; this is similar to the sequential quadratic programming method[2, 39]. A multi-stage process can be converted into a single stage process. The cost-to-go function is expanded with the following equations.

$$V^i_{\boldsymbol{x}} = \boldsymbol{F}^{iT}_{\boldsymbol{x}} V^{i+1}_{\boldsymbol{x}} + L^i_{\boldsymbol{x}} \tag{6.38}$$

$$V^i_{\boldsymbol{xx}} = \boldsymbol{F}^{iT}_{\boldsymbol{x}} V^{i+1}_{\boldsymbol{xx}} \boldsymbol{F}^i_{\boldsymbol{x}} + L^i_{\boldsymbol{xx}} + \sum_j V^{i+1}_{\boldsymbol{x}^j} \boldsymbol{F}^{ij}_{\boldsymbol{xx}} \tag{6.39}$$

$$V^i_{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{F}^{iT}_{\boldsymbol{u}} V^{i+1}_{\boldsymbol{x}} + L^i_{\boldsymbol{u}} \\ V^{i+1}_{\boldsymbol{u}} \end{bmatrix} \tag{6.40}$$

$$V^i_{\boldsymbol{uu}} = \begin{bmatrix} {\boldsymbol{F}_{\boldsymbol{u}}}^{iT} V^{i+1}_{\boldsymbol{xx}} {\boldsymbol{F}_{\boldsymbol{u}}}^i + L^i_{\boldsymbol{uu}} + \sum_j V^{i+1}_{\boldsymbol{x}^j} \boldsymbol{F}^{ij}_{\boldsymbol{uu}} & {\boldsymbol{F}_{\boldsymbol{u}}}^{iT} V^{i+1}_{\boldsymbol{xu}} \\ V^{i+1}_{\boldsymbol{ux}} {\boldsymbol{F}_{\boldsymbol{u}}}^i & V^{i+1}_{\boldsymbol{uu}} \end{bmatrix} \tag{6.41}$$

$$V^i_{\boldsymbol{ux}} = \begin{bmatrix} {\boldsymbol{F}_{\boldsymbol{u}}}^{iT} V^{i+1}_{\boldsymbol{xx}} {\boldsymbol{F}_{\boldsymbol{x}}}^i + L^i_{\boldsymbol{ux}} + \sum_j V^{i+1}_{\boldsymbol{x}^j} \boldsymbol{F}^{ij}_{\boldsymbol{ux}} \\ V^{i+1}_{\boldsymbol{ux}} {\boldsymbol{F}_{\boldsymbol{x}}}^i \end{bmatrix} \tag{6.42}$$

$$V^i_{\boldsymbol{xu}} = \begin{bmatrix} {\boldsymbol{F}_{\boldsymbol{x}}}^{iT} V^{i+1}_{\boldsymbol{xx}} {\boldsymbol{F}_{\boldsymbol{u}}}^i + L^i_{\boldsymbol{xu}} + \sum_j V^{i+1}_{\boldsymbol{x}^j} \boldsymbol{F}^{ij}_{\boldsymbol{xu}} & {\boldsymbol{F}_{\boldsymbol{x}}}^{iT} V^{i+1}_{\boldsymbol{xu}} \end{bmatrix} \tag{6.43}$$

Similarly, each constraint is expanded as

$$q^i_{\boldsymbol{x}} = \boldsymbol{F}^{iT}_{\boldsymbol{x}} q^{i+1}_{\boldsymbol{x}} \tag{6.44}$$

$$q^i_{\boldsymbol{xx}} = \boldsymbol{F}^{iT}_{\boldsymbol{x}} q^{i+1}_{\boldsymbol{xx}} \boldsymbol{F}^i_{\boldsymbol{x}} + \sum_j q^{i+1}_{\boldsymbol{x}^j} \boldsymbol{F}^{ij}_{\boldsymbol{xx}} \tag{6.45}$$

$$q^i_{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{F}^{iT}_{\boldsymbol{u}} q^{i+1}_{\boldsymbol{x}} \\ \\ q^{i+1}_{\boldsymbol{u}} \end{bmatrix} \tag{6.46}$$

$$q^i_{\boldsymbol{uu}} = \begin{bmatrix} \boldsymbol{F_u}^{iT} q^{i+1}_{\boldsymbol{xx}} \boldsymbol{F_u}^i + \sum_j q^{i+1}_{\boldsymbol{x}^j} \boldsymbol{F}^{ij}_{\boldsymbol{uu}} & \boldsymbol{F_u}^{iT} q^{i+1}_{\boldsymbol{xu}} \\ \\ q^{i+1}_{\boldsymbol{ux}} \boldsymbol{F_u}^i & q^{i+1}_{\boldsymbol{uu}} \end{bmatrix} \tag{6.47}$$

$$q^i_{\boldsymbol{ux}} = \begin{bmatrix} \boldsymbol{F_u}^{iT} q^{i+1}_{\boldsymbol{xx}} \boldsymbol{F_x}^i + \sum_j q^{i+1}_{\boldsymbol{x}^j} \boldsymbol{F}^{ij}_{\boldsymbol{ux}} \\ \\ q^{i+1}_{\boldsymbol{ux}} \boldsymbol{F_x}^i \end{bmatrix} \tag{6.48}$$

$$q^i_{\boldsymbol{xu}} = \begin{bmatrix} \boldsymbol{F_x}^{iT} q^{i+1}_{\boldsymbol{xx}} \boldsymbol{F_u}^i + \sum_j q^{i+1}_{\boldsymbol{x}^j} \boldsymbol{F}^{ij}_{\boldsymbol{xu}} & \boldsymbol{F_x}^{iT} q^{i+1}_{\boldsymbol{xu}} \end{bmatrix} \tag{6.49}$$

Using these equations, the constraint and cost-to-go function can be collapsed into a single stage.

6.7 Proof of Concept Example

The previous sections outlined the mathematical equations that govern the algorithm. The current section applies the algorithm to several test examples, which demonstrates the potential benefits of the algorithm. The algorithm is coded in Matlab. For the proof of concept test uses Matlab's *fmincon* optimization routine as a reference. *fmincon* is a routine that solves optimization problems subject to linear and nonlinear constraints. For the proof of concept test *fmincon* uses a *'medium-scale: SQP, Quasi-Newton, line-search'* algorithm. The Matlab version used is 7.4.0.287 (R2007a) on an Apple computer running OS 10.5.3.

The examples utilize a linear dynamical problem with a quadratic cost function and linear and nonlinear constraints. The dynamics for the problem are governed by

$$\boldsymbol{x}^{i+1} = \begin{bmatrix} \boldsymbol{r}^{i+1} \\ \boldsymbol{v}^{i+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}^{i} \\ \boldsymbol{v}^{i} \end{bmatrix} + \begin{bmatrix} \int_{t^i}^{t^{i+1}} \int_{t^i}^{t^{i+1}} \boldsymbol{A}^i d\tau d\psi \\ \int_{t^i}^{t^{i+1}} \boldsymbol{A}^i d\tau \end{bmatrix} \tag{6.50}$$

Here $\boldsymbol{r}$ is the position of the body, $\boldsymbol{v}$ is the velocity, and $\boldsymbol{A}$ is the acceleration. In these problems $\boldsymbol{A}$ is the control variable. The cost function is

$$J = \sum_i 0.5 \left|\boldsymbol{A}^i\right|^2 \left(t^{i+1} - t^i\right) \tag{6.51}$$

The problem is set up with 10 segments each spanning a time period equal to $(t^{i+1} - t^i) = \Delta t = 1$. The initial guess is

$$x^0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{6.52}$$

and

$$\boldsymbol{A}^i = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, & \text{if } i = 0 \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T, & \text{if } i \geq 1 \end{cases} \tag{6.53}$$

To ensure that the various subroutines have been written correctly, the first test attempts to solve a problem with linear constraints. The constraints for the linear test are

$$0 = \boldsymbol{x}^0 - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{6.54}$$

and

$$0 = \boldsymbol{x}^{10} - \begin{bmatrix} 25 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{6.55}$$

Both *fmincon* and the second order update converged to the optimal solution in a single step.

For the second proof of concept test the constraints are

$$0 = \boldsymbol{x}^0 - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{6.56}$$

$$\boldsymbol{r}^{10} - \begin{bmatrix} 25 & 0 & 0 \end{bmatrix}^T \tag{6.57}$$

and

$$0 = 0.5\boldsymbol{v}^{10} \cdot \boldsymbol{v}^{10} - 0.5 \tag{6.58}$$

The quadratic constraint in Eq. (6.58) makes the problem more difficult and tests the effectiveness of the various algorithms. The results of the optimization are shown in Fig. 6.2. The figure shows that the single stage method converges in one step. This is expected because in the single stage case the problem is exactly quadratic, which is the type of problem the algorithm is designed to solve. The multi-stage version of the algorithm takes several steps because the multi-stage version truncates the control law at every stage, which results in the loss of information at every stage. *fmincon* takes almost 20 iterations to converge, which is about five times as many iterations as the multi-stage method required. In this example, with linear dynamics and quadratic constraints, the single and multi-stage algorithm outperformed *fmincon*.

For the final proof of concept test, a higher order a nonlinear terminal constraints is used. The initial and terminal constraints are

$$0 = \boldsymbol{x}^0 - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{6.59}$$

$$\boldsymbol{r}^{10} = \begin{bmatrix} 25 & 0 & 0 \end{bmatrix}^T \tag{6.60}$$

and

$$0 = 0.5\boldsymbol{v}^{10} \cdot \boldsymbol{v}^{10} + \left(\boldsymbol{v}^{10} \cdot \boldsymbol{v}^{10}\right)^2 - 2 \tag{6.61}$$

This test cases stresses the algorithm because the constraints are highly nonlinear and the initial guess is poor since it does not satisfy any of the constraints. Fig. 6.3 shows the that the single and multi-stage versions of the algorithm converged to

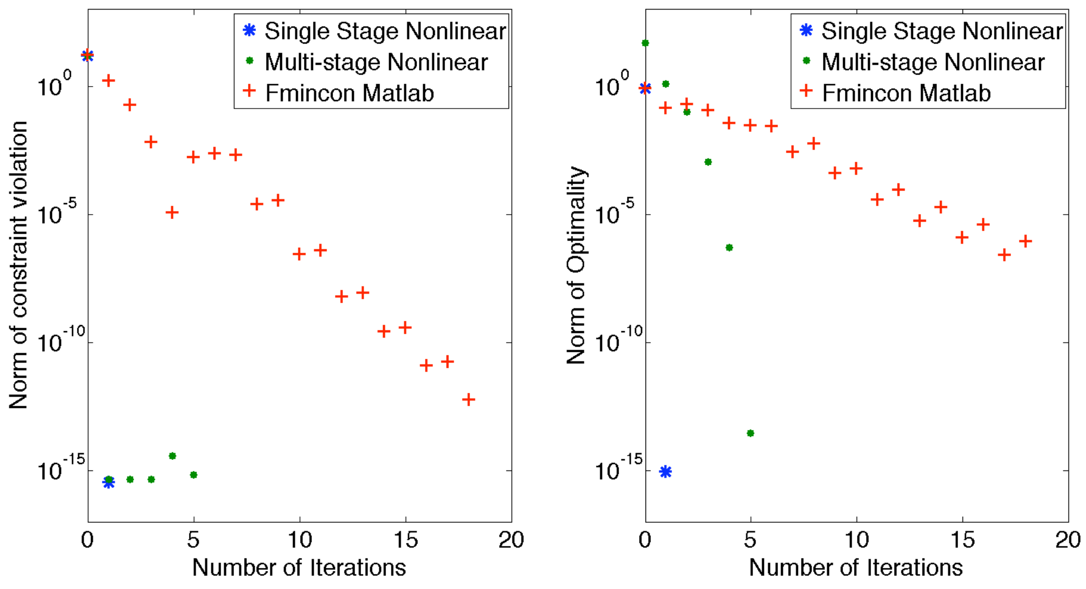


Figure 6.2: Rate of convergence for single and multi stage versions of the algorithm vs. *fmincon*. For the quadratic constraint case, the single stage method converges in one step and while the multi-stage method took several steps to converge. *fmincon* had the slowest rate of convergence and took almost 20 iterations to converge.

the optimal solution within about five iterations while *fmincon* does not converge within 500 iterations. This shows that this algorithm can be more robust than linear methods.

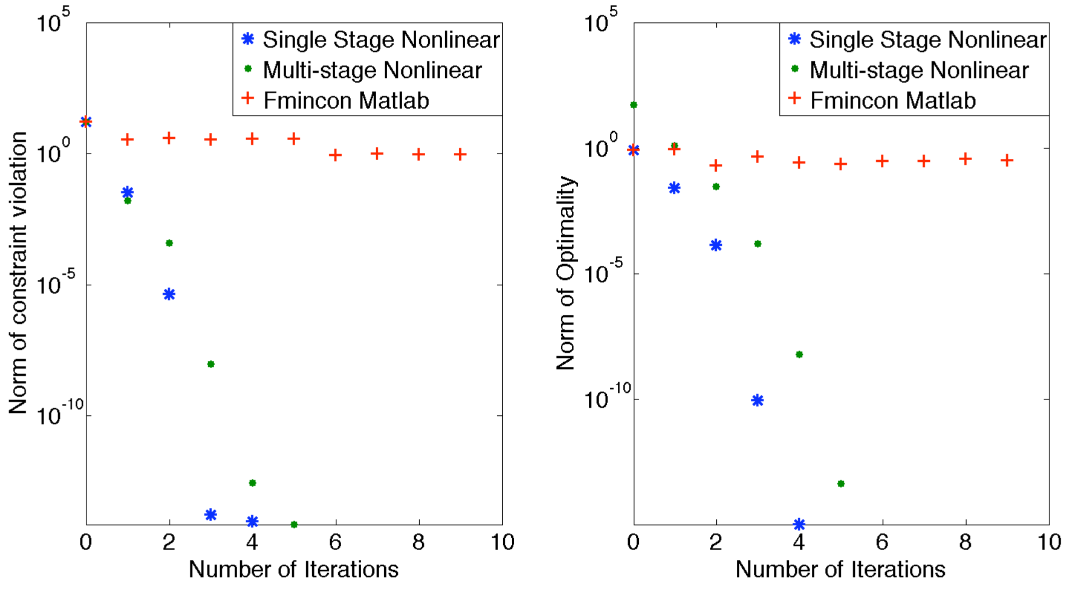


Figure 6.3: Rate of convergence for single and multi stage versions of the algorithm vs. *fmincon*. For the nonlinear constraint case, the single and multi-stage method converge in several steps and *fmincon* fails to converge within 500 iterations.

6.7.1 Inequality Constraint Example

In the previous examples only the equality constraints are handled. In this example, control constraints are considered and are handled by the slack variable method[39]. The dynamics and cost function remain the same. The flight time is 8 and the total number of segments is 20. The constraints are

$$\frac{1}{2}\left|\boldsymbol{A}^i\right|^2 \leq 0.5 \tag{6.62}$$

and the initial and final constraints are

$$\boldsymbol{x}^0 - \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{6.63}$$

$$\boldsymbol{r}^{20} - \begin{bmatrix} 25 & 0 & 0 \end{bmatrix}^T \tag{6.64}$$

The initial guess is

$$\boldsymbol{x}^0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{6.65}$$

and

$$\boldsymbol{A}^i = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \tag{6.66}$$

Using the multi-stage method, the algorithm converges to the optimal solution in three iterations. The thrust profile is shown in Fig. 6.4. As the algorithm iterates, it stays within the constraint boundaries and is able to converge to the correct solution. This is a valuable property because the solution at every iteration may not be optimal, but it is a valid solution to the original constraint problem.

6.8 Outstanding Issues

While the algorithm takes fewer iterations to converge and can handle inequality and equality constraints well, there are several outstanding issues that prevent its use in more difficult problems.

6.8.1 Particular Solution

Unlike most optimization algorithms that generate a linear control update, this algorithm requires solving a nonlinear coupled equation given by Eq. (6.24). Unlike linear problems, which can be solved relatively easily, Eq. (6.24) must be solved using iterative numerical methods, and it can have multiple solutions. In order to reliably minimize the solutions, a method is needed that can find the solve Eq. (6.24) while minimizing the cost-to-go.

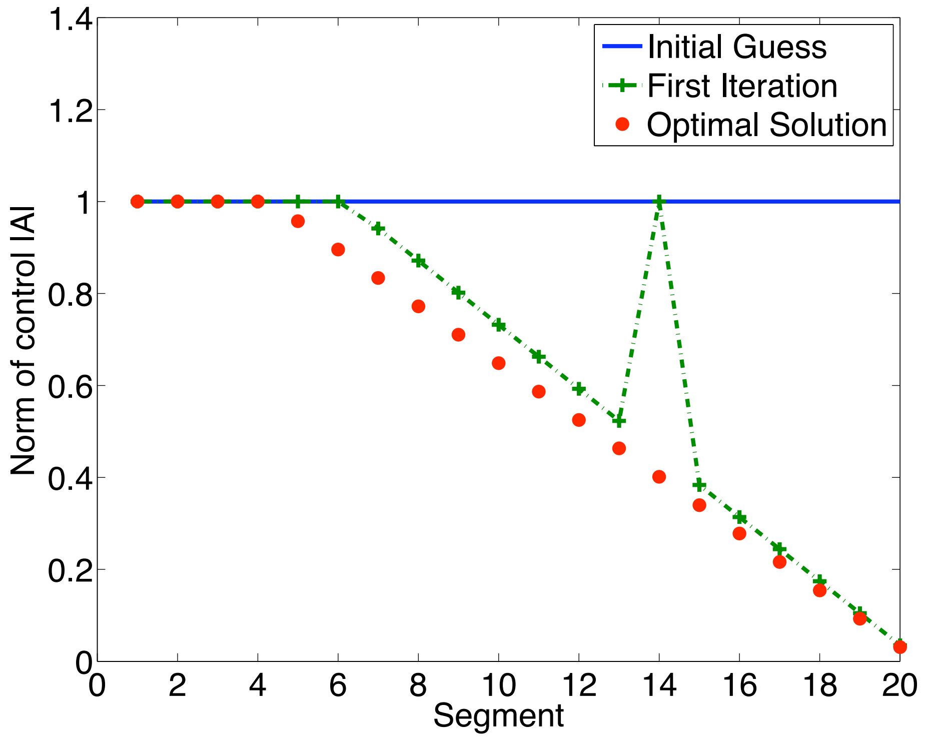


Figure 6.4: Acceleration profile over 3 iterations. The second order algorithm has the ability to automatically turn on and off inequality constraints. The intermediate thrust profile is locked onto non optimal constraints, which are then automatically turned off in the next iteration.

6.8.2 Nonquadratic Cost Functions

The algorithm uses second order information to model the constraints and cost function. However, if the cost function or constraints are not modeled well by a second order model then Eq. (6.24) will be difficult to solve. In testing, it was found that algorithm had trouble with L_1, specifically after several iterations the algorithm would either diverge or fail to solve Eq. (6.24).

6.9 Conclusion

In this chapter, a new second order optimization algorithm that uses a nonlinear (second order) control update is derived. The algorithm is implement and tested on several examples with non linear constraints. The algorithm converges quickly, and for quadratic constraints, the algorithm maintains feasibility while optimizing. This ensures that at each iteration the solution still solves the original constraint problem. For inequality constraints, the algorithm is able to maintain the solution within the feasible set, and it automatically finds the appropriate active control set.

CHAPTER VII

CONCLUSIONS

The central aim of this work has been to create methods for automating the generation of feasible trajectories for use in trade studies. Two different methods are designed for high and low level trade studies. Both methods utilize algorithms that generate their own initial guesses for the optimization process, giving the algorithms a self starting capability, which removes the burden of generating an initial guess from the user. **These methods act as front ends to other software routines to auto generate feasible trajectories making optimization routines simpler and easier to use**.

7.1 High Level Trades

High level trades use reduced order models to quickly identify the feasible space of a mission. They are designed to provide an upper bound on the performance as well as to quantify the key parameters of interest. The work in Chapter III expanded on the work by Kluever[26, 25], Lorenzo[32, 8], Marec[34], and Jahn[23]. The analysis of the optimal net payload mass for the constant specific impulse case generalized Jahn's analysis. This analysis removed the specific impulse's dependance on the power level and mass. Instead it was found that the specific impulse is a function of the ionization cost, trip time, power system's power to mass ratio, and the time averaged power used to initial power ratio. This result shows that the optimal specific impulse is not dependent on the mass of the spacecraft but is dependent on the scaling of the power system and mission duration.

Nondimensionalizing the equations and mass fractions showed that the power system and propellant mass should be treated as one system and that the mass optimal mass distributed to power system and propellant system is dependent on the trip time, propellant ionization cost, and power system scaling. From the results in Chapter III, it is clear that the optimal distribution favors more massive power systems for short missions, but for long mission durations, the optimal solution distributes the mass equally between the power system and propellant.

Using these reductions, an optimization program is developed that automates the high level trades over the power system specific power. The algorithm employs a two stage homotopy process to autonomously generate the initial guess and solve the full optimization process. Once a single solution is found, the power system's α is varied to see how it effects the net payload mass, specific impulse, and C_3. In order to ensure that the solution is optimal, the program checks the current switching structure with the optimal switching structure, which allows the program to add and remove coast arcs without user intervention.

Using these reductions, the optimal specific impulse can be estimated because it is coupled to the flight time, but independent of the spacecraft mass. Furthermore, the automated methods used allow for a whole continuum of solutions to be found, which differs from the results of Kluever and Lorenzo, where only several solutions are provided.

7.2 Feasible Trajectory Generation for Low Level Trades

Low level trades are critical to evaluating the benefits of new technologies. For electric propulsion, the trajectory has to be found such that it satisfies the propulsion and power system constraints. In order to automate the generation of feasible trajectories, a two stage process is used. The Chebyshev approximation method outlined in Chapter IV expands on the shape based method[48, 20, 45, 50]. Specifically, the Chebyshev method address two of the major limitation in the exponential sinusoid[48, 49, 47, 50], its inability to handle rendezvous problems, and the lack of freedom in optimizing the trajectory.

The Chebyshev approximation method uses a different parameterization that allows the method to scale to an arbitrary order. This ensures that enough degrees of freedom exist so that the trajectory can be optimized. In order to simplify the search for trajectories, an automated algorithm is developed that can generate trajectories between two bodies of interest using an ephemeris. The algorithm is designed to be self contained so that it can be implemented in a parallel or distributed environment, which reduces the computational time. A proof of concept tool is then constructed which showcases the automated algorithm and utilizes the parallel capability of the method. The Chebyshev method has the ability to autonomously find and optimize trajectories.

Chapter V uses the trajectories generated by the Chebyshev method to generate initial thrust laws. Thruster, launch vehicle parameterizations, and constraints are then used as constraints on the trajectory. An algorithm is developed that iteratively updates the thrust control law until a trajectory is found that satisfies the constraints. Because the process is tied to the Chebyshev method, a new initial guess does not have to be generated for each thruster models, which simplifies the process and allows the same Chebyshev trajectory to be used to find feasible trajectories that utilize different thrusters.

The Chebyshev and the feasible trajectory generation method provide a framework that can be used as a front end to other optimization routines such as MALTO or Mystic. The framework provided here is illustrated in Fig. 7.1. The power of these methods is that they are autonomous and that they can be used with other programs to generate initial guesses for trajectory optimization programs.

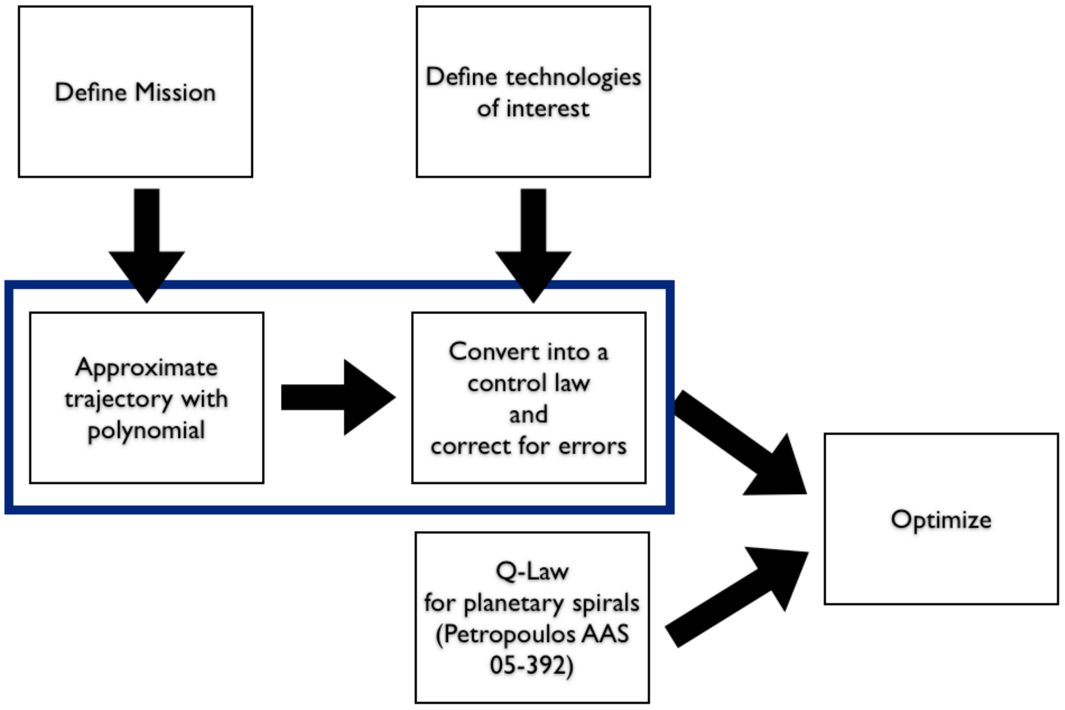

Define Mission
Define technologies of interest
Approximate trajectory with polynomial
Convert into a control law and correct for errors
Optimize
Q-Law for planetary spirals (Petropoulos AAS 05-392)

Figure 7.1: The framework in which the Chebyshev method and feasible trajectory generation routine fits in with previously conducted research. The blue box represents the work contained in this thesis.

7.3 Nonlinear Optimization Algorithm

Most optimization methods use first and sometimes second order information to generate a linear control update[38, 22, 21, 60, 62, 59, 31, 2, 39, 16, 35]. In-Chapter VI an optimization algorithm is derived that uses a second order update and satisfies the constraints and optimality conditions to the second order. The method is designed to begin with a feasible trajectory and maintain feasibility as it optimizes, which differs from current methods. The algorithm is implemented and tested against a commonly available optimization routine. Because of the nonlinear nature of the algorithm, the algorithm has to ability to automatically turn inequality constraints on and off and it can converge faster than current linear methods.

7.4 Future Work

The research presented has shown that the trajectory generation process for high and low level trades can be automated and implemented using simple numerical techniques. The second order optimization algorithm demonstrates that higher order methods can converge more quickly and can have the ability to identify the active set of constraint. While the methods have been successfully tested, several issues remain, which should be investigated further.

7.4.1 Multi-Dimensional High Level Trades

In the current implementation of the high level trade algorithm, the method only iterates over the power system's mass to power ratio. In practice, it would be useful to be able to conduct multidimensional trades involving different variables, like the power system, propellant, and flight times. Generalizing the method and using parallel processing would allow rapid trade studies to be conducted over a wide range of conditions.

7.4.2 Multiple Leg Missions

The Chebyshev method is designed to solve the rendezvous problem between two different bodies. Current missions are looking to take advantage of electric propulsion by visiting multiple bodies. Extending the Chebyshev method to handle multiple leg missions would allow entire missions to be prototyped.

7.4.3 L_1 Cost Functions

The optimization algorithm is designed to approximate the cost function and constraints to the second order. For problems with that structure the algorithm performs well, however for L_1 cost functions, the algorithm has some difficulty and can get stuck around an infeasible solution. In order to solve this problem, a method is required that can properly bias the solution so the solver converges onto the feasible solution. Biasing the initial guess appropriately is critical to ensuring the algorithm can handle a wide variety of problems.

APPENDICES

APPENDIX A

Chebyshev Trajectory Approximation Program

A.1 Setup

The first step in installing the program is to ensure that the operating system is OS 10.5.x, where x is any number. In the“Chebyshev program” folder on the CD there are two items, The “Chebyshev Approximation” program and the data files, stored in the “TrajOptimization” folder. The “TrajOptimization/Data” folder contains the necessary SPICE files. The ephemeris can be updated as required by simply adding the appropriate SPICE ephemeris files.

The “TrajOptimization” folder needs to be located in the ”/Library/Application Support/” folder. Once placed there, double click on “Chebyshev Approximation” to launch the program. Once launched the program interface should look like Fig. A.1.

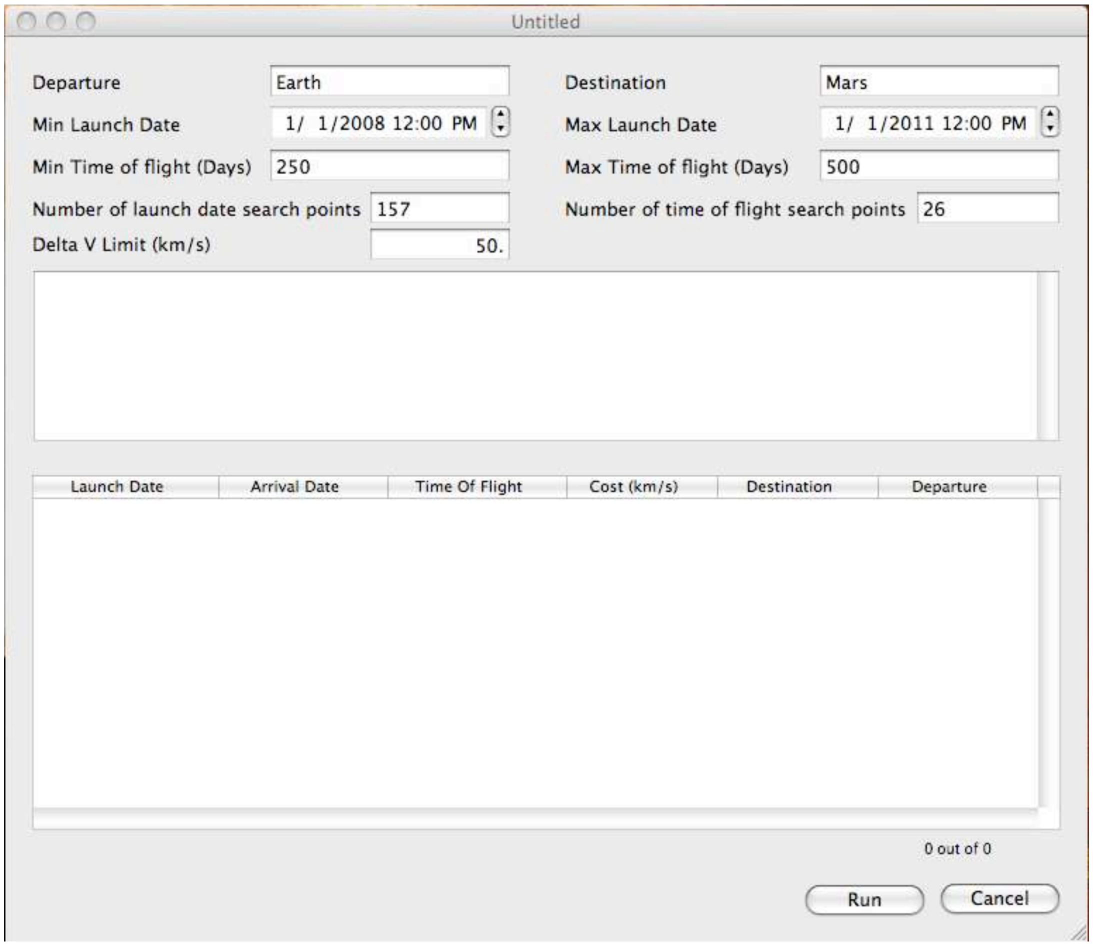
A.2 Earth to Mars Walk Through

In this section a quick tutorial is provided on how to setup an Earth to Mars search. First launch the program and an interface similar to Fig. A.1 should pop up. The first step is to set the departure and destination bodies. Change the departure field to **Earth** and the destination field to **Mars**. Next the min launch date and max launch date need to be set. These are a lower and upper bound, respectively, on the departure date of the spacecraft. The min launch date should be changed to **1/1/2008** and the max launch date should be set to **1/1/2011**. A three year search period is set because the synodic period between Earth and Mars is approximately 3 years. The min time of flight and max time of flight are the lower and upper bound, respectively, on the flight time of the spacecraft. The min time of flight should be changed to **250** and the max time of flight should be set to **500**. This means that the program will search for flight times between 250 and 500 days. With the upper and lower bounds set, the search grid needs to be set. The two values that determine the number of grid points in the search space are "Number of launch date search points" and "Number of time of flight search points". "Number of launch date search points" determines how many search grid points exist between the minimum and maximum launch date. "Number of time of flight search points" determines how many search grid points exist between the minimum and maximum flight times. "Number of launch date search points" needs to be set to **157** which means that the program will vary the launch date by one week increments. "Number of time of flight search points" should be set to **26** which means that the program will search over the flight times in 10 day increments. Finally, the "Delta V Limit" is the maximum ΔV allowed. If a particular solution has a ΔV greater than "Delta V Limit" then the solution will NOT be recorded. For this example the "Delta V Limit" should be set to **50**.

Now that the interface is set up go ahead and click on the "Run" button. At this point several things will happen. A few text messages should appear in the message window, Fig. A.2. Fig. A.2 simply passes messages from the program to the user.

Once several trajectories are found the trajectory listing, Fig. A.3 should begin to populate with information. The trajectory listing represents all the trajectories that have been computed. The trajectory listing is sortable by clicking on the appropriate heading name.

Finally, a progress bar should also appear. The progress bar has three colors, red, Fig. A.4, yellow, Fig. A.5, and green Fig. A.6. Red indicates that the less then half the grid points have been searched. Yellow indicates that less than three quarters of the grid points have been searched and green indicates that over three quarters of the grid points have been searched. When the progress bar disappears the program has completed searching the grid points. At this point the data can be saved by going to "**File**− >**Save**".

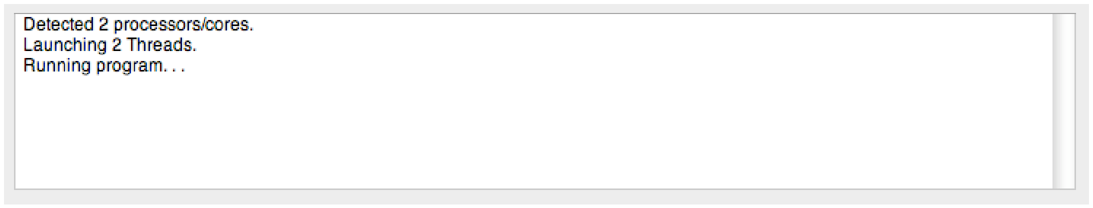


Departure: Earth
Destination: Mars
Min Launch Date: 1/ 1/2008 12:00 PM
Max Launch Date: 1/ 1/2011 12:00 PM
Min Time of flight (Days): 250
Max Time of flight (Days): 500
Number of launch date search points: 157
Number of time of flight search points: 26
Delta V Limit (km/s): 50.

Launch Date	Arrival Date	Time Of Flight	Cost (km/s)	Destination	Departure

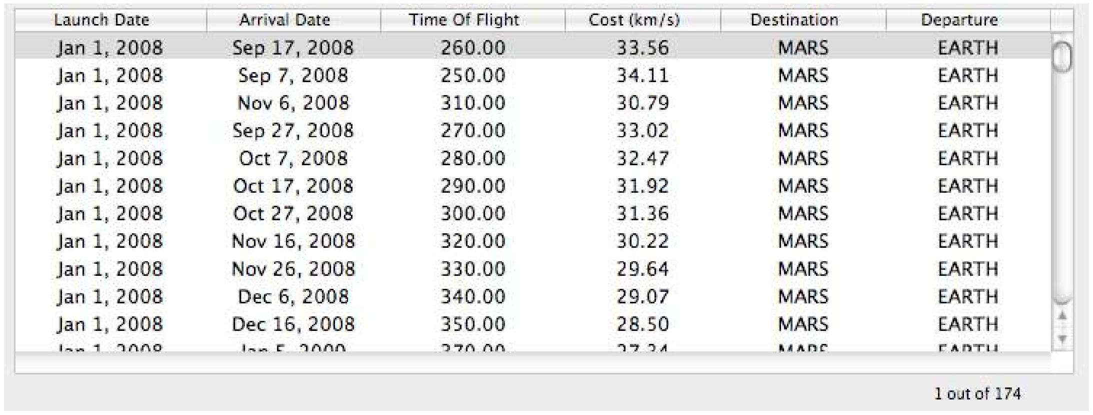
0 out of 0

Figure A.1: The interface of the Chebyshev approximation program. The program takes several limited inputs then searches over the launch dates and arrival dates to find trajectories with low ΔV costs.



```
Detected 2 processors/cores.
Launching 2 Threads.
Running program. . .
```

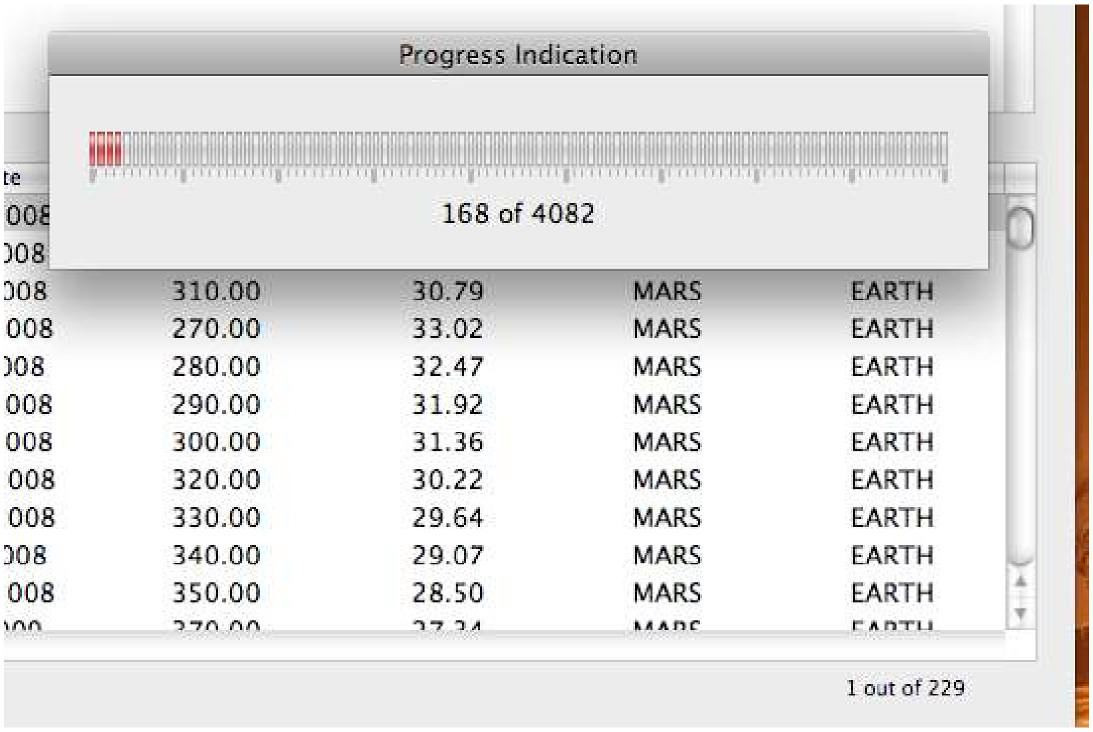
Figure A.2: Message window that displays messages to the user. The message window should indicate the number of processors/cores being utilized by the program.



Launch Date	Arrival Date	Time Of Flight	Cost (km/s)	Destination	Departure
Jan 1, 2008	Sep 17, 2008	260.00	33.56	MARS	EARTH
Jan 1, 2008	Sep 7, 2008	250.00	34.11	MARS	EARTH
Jan 1, 2008	Nov 6, 2008	310.00	30.79	MARS	EARTH
Jan 1, 2008	Sep 27, 2008	270.00	33.02	MARS	EARTH
Jan 1, 2008	Oct 7, 2008	280.00	32.47	MARS	EARTH
Jan 1, 2008	Oct 17, 2008	290.00	31.92	MARS	EARTH
Jan 1, 2008	Oct 27, 2008	300.00	31.36	MARS	EARTH
Jan 1, 2008	Nov 16, 2008	320.00	30.22	MARS	EARTH
Jan 1, 2008	Nov 26, 2008	330.00	29.64	MARS	EARTH
Jan 1, 2008	Dec 6, 2008	340.00	29.07	MARS	EARTH
Jan 1, 2008	Dec 16, 2008	350.00	28.50	MARS	EARTH

1 out of 174

Figure A.3: Trajectory listing that displays pertinent information on the computed trajectories. The trajectory listing shows the launch date, arrival date, flight time, and ΔV cost. The trajectory listing is sortable by clicking on the appropriate heading.



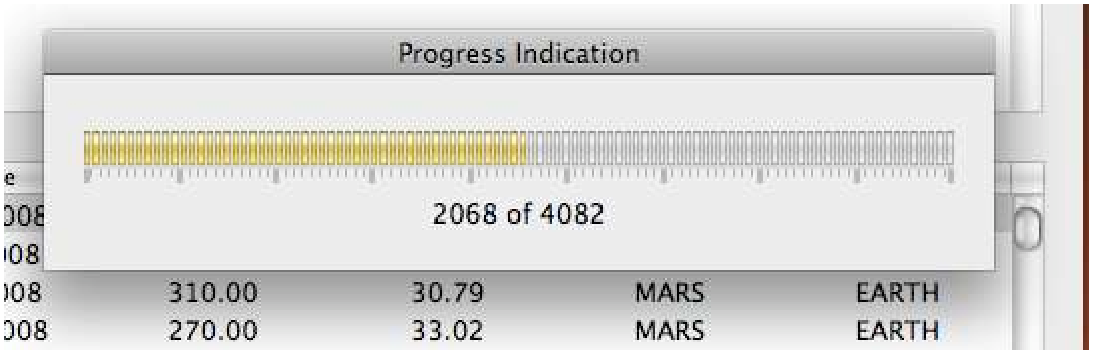
Progress Indication

168 of 4082

008	310.00	30.79	MARS	EARTH
008	270.00	33.02	MARS	EARTH
008	280.00	32.47	MARS	EARTH
008	290.00	31.92	MARS	EARTH
008	300.00	31.36	MARS	EARTH
008	320.00	30.22	MARS	EARTH
008	330.00	29.64	MARS	EARTH
008	340.00	29.07	MARS	EARTH
008	350.00	28.50	MARS	EARTH

1 out of 229

Figure A.4: Red progress bar indicates that less than half of the grid points have been searched.

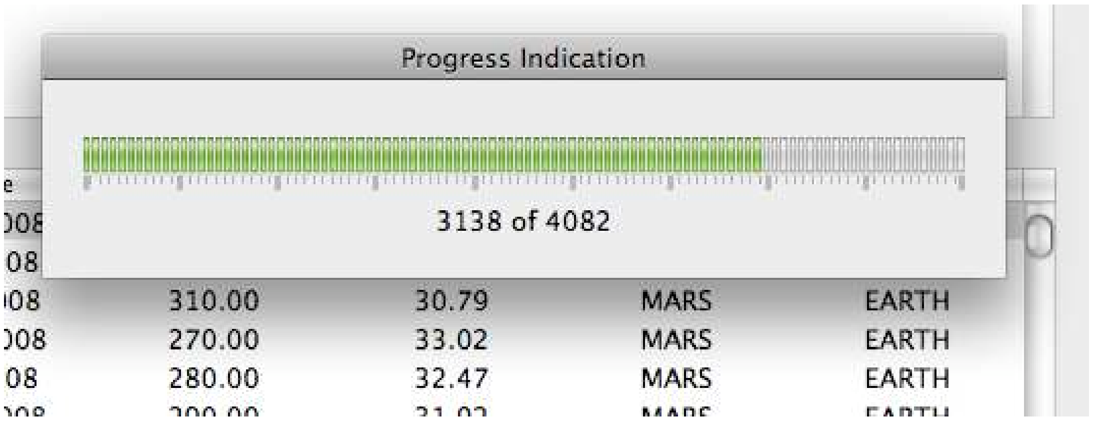


Progress Indication

2068 of 4082

```
008
008
008    310.00    30.79    MARS    EARTH
008    270.00    33.02    MARS    EARTH
```

Figure A.5: Yellow indicates that less than three quarters of the grid points have been searched.



Progress Indication

3138 of 4082

```
008
08
008    310.00    30.79    MARS    EARTH
008    270.00    33.02    MARS    EARTH
08     280.00    32.47    MARS    EARTH
```

Figure A.6: Green indicates that over three quarters of the grid points have been searched.

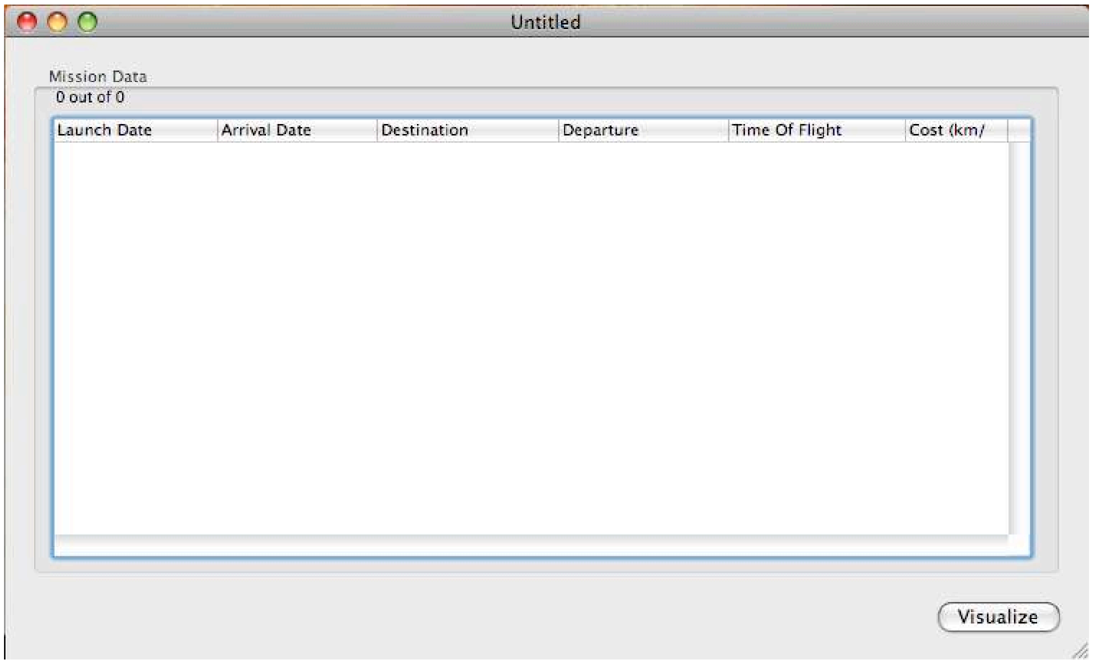
APPENDIX B

Chebyshev to Matlab Conversion

To convert the data generated by the ‘Chebyshev Approximation’ into a usable Matlab data file requires the ‘ChebyshevToMat’ program located in the same folder as ‘Chebyshev Approximation’ program.

The first step is to launch the ‘ChebyshevToMat’ program. The program interface will look like Fig. B.1. In Appendix A a save file was created that stored the relevant trajectories generated by the ‘Chebyshev Approximation’ program. Now go to **File**$-$ >**Open** and load the save file from Appendix A. Once the file is loaded the interface will look like Fig. B.2. Now select the ‘Cost’ field to sort the trajectories by the ΔV. With the save file loaded and sorted the data can be exported into a Matlab file. To do this simply highlight the trajectories that need to be exported. This is done by selected a trajectory then holding shift and the up or down arrows. Another method is to select a trajectory hold the mouse down while dragging the mouse over the trajectories of interest, see Fig. B.3.

Once the trajectories that need to be exported are selected press the ‘Visualize’ button. A file called ‘Report.m’ will have been created on the hard drive. This file contains the estimated acceleration profile for the trajectories. This file will be

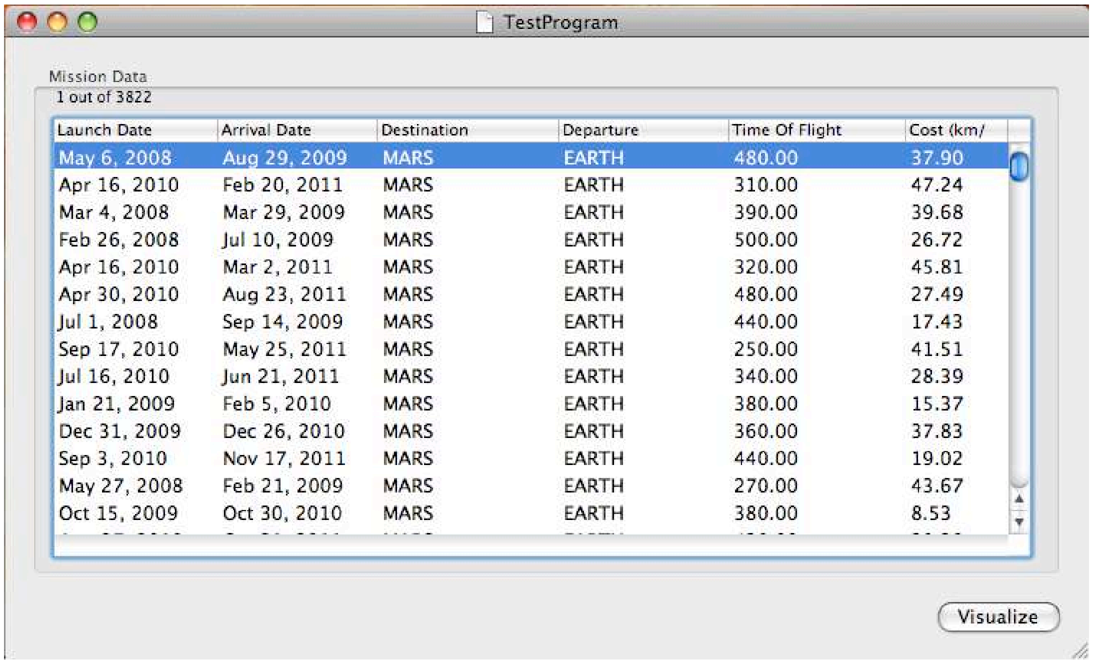


Mission Data

0 out of 0

Launch Date	Arrival Date	Destination	Departure	Time Of Flight	Cost (km/

Figure B.1: Initial interface to ChebyshevToMat program. Interface displays and allows sorting of generated trajectories.



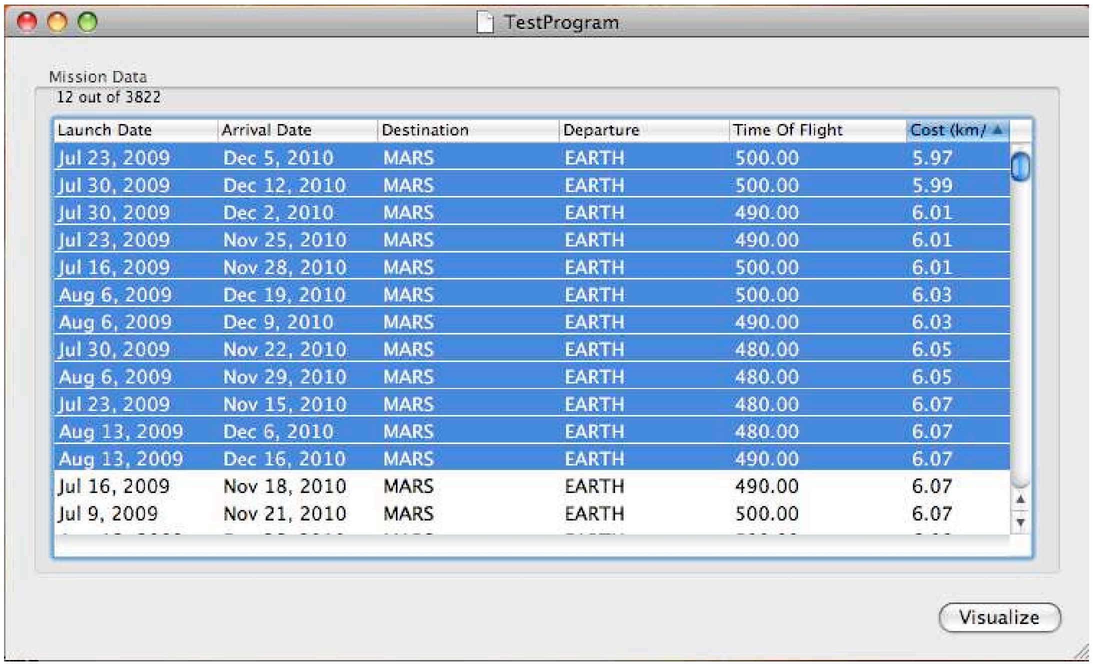
Mission Data

1 out of 3822

Launch Date	Arrival Date	Destination	Departure	Time Of Flight	Cost (km/
May 6, 2008	Aug 29, 2009	MARS	EARTH	480.00	37.90
Apr 16, 2010	Feb 20, 2011	MARS	EARTH	310.00	47.24
Mar 4, 2008	Mar 29, 2009	MARS	EARTH	390.00	39.68
Feb 26, 2008	Jul 10, 2009	MARS	EARTH	500.00	26.72
Apr 16, 2010	Mar 2, 2011	MARS	EARTH	320.00	45.81
Apr 30, 2010	Aug 23, 2011	MARS	EARTH	480.00	27.49
Jul 1, 2008	Sep 14, 2009	MARS	EARTH	440.00	17.43
Sep 17, 2010	May 25, 2011	MARS	EARTH	250.00	41.51
Jul 16, 2010	Jun 21, 2011	MARS	EARTH	340.00	28.39
Jan 21, 2009	Feb 5, 2010	MARS	EARTH	380.00	15.37
Dec 31, 2009	Dec 26, 2010	MARS	EARTH	360.00	37.83
Sep 3, 2010	Nov 17, 2011	MARS	EARTH	440.00	19.02
May 27, 2008	Feb 21, 2009	MARS	EARTH	270.00	43.67
Oct 15, 2009	Oct 30, 2010	MARS	EARTH	380.00	8.53

Figure B.2: Interface with Chebyshev approximated trajectories loaded. Each instance represents a unique trajectory.

needed to make the trajectories feasible. Now go to **File**− >**Quit** to terminate the program.



12 out of 3822

Launch Date	Arrival Date	Destination	Departure	Time Of Flight	Cost (km/
Jul 23, 2009	Dec 5, 2010	MARS	EARTH	500.00	5.97
Jul 30, 2009	Dec 12, 2010	MARS	EARTH	500.00	5.99
Jul 30, 2009	Dec 2, 2010	MARS	EARTH	490.00	6.01
Jul 23, 2009	Nov 25, 2010	MARS	EARTH	490.00	6.01
Jul 16, 2009	Nov 28, 2010	MARS	EARTH	500.00	6.01
Aug 6, 2009	Dec 19, 2010	MARS	EARTH	500.00	6.03
Aug 6, 2009	Dec 9, 2010	MARS	EARTH	490.00	6.03
Jul 30, 2009	Nov 22, 2010	MARS	EARTH	480.00	6.05
Aug 6, 2009	Nov 29, 2010	MARS	EARTH	480.00	6.05
Jul 23, 2009	Nov 15, 2010	MARS	EARTH	480.00	6.07
Aug 13, 2009	Dec 6, 2010	MARS	EARTH	480.00	6.07
Aug 13, 2009	Dec 16, 2010	MARS	EARTH	490.00	6.07
Jul 16, 2009	Nov 18, 2010	MARS	EARTH	490.00	6.07
Jul 9, 2009	Nov 21, 2010	MARS	EARTH	500.00	6.07

Figure B.3: Chebyshev Trajectories that have been sorted and selected. They are now ready for exporting which will allow them to be made into feasible trajectories.

APPENDIX C

Matlab Program to Generate Feasible Trajectories

Previously, a C++ program was used to create the 'Report.m' file which contains the acceleration profile for the selected trajectories. The 'Report.m' needs to be copied to the 'FeasibleTraj' directory. The directory already contains a 'Report.m' file.

Now open up 'MainFile.m' in Matlab. 'MainFile.m' is a Matlab program that generates a set of feasible trajectories for three thrusters, the NEXT, NSTAR, and BPT-4000. The program generates the feasible trajectories by using the Chebyshev trajectories as an initial guess then incorporating the power system model and constraints into the problem.

Near the top of the file under the %Modify These comment are 7 parameters that determine the trajectory.

dutycycle is the faction of time that the thruster is on per each thrust period.

LV is the identification number of the launch vehicle. An integer value corresponds to a particular launch vehicle, see 'LaunchVehicle.m' for the list of available launch vehicles and the ID to name correspondence.

NumEngines is the number of thrusters that can be used by the spacecraft.

Trajnum is the identification number of the Chebyshev trajectory that is used as an initial guess. The ID value is used to select the appropriate trajectory from 'Report.m'. The first trajectory exported by the program in Appendix B corresponds to the first ID in 'Report.m'.

PMIN is the lower bound on the power trade in kilowatts.

PMAX is the upper bound on the power trade in kilowatts.

PowerSystem is the type of power system being used by the spacecraft. 0 corresponds to a thrust limited power system. 1 is a constant power source. 2 is a solar array based power source.

Once the appropriate values have been selected simply press run. The current default values and 'Report.m' file will produce a set of feasible trajectories and plots.

APPENDIX D

NSTAR, NEXT, and BPT-4000 Parameterization

D.1 Thruster Model

The thruster model used here is similar to the model used in references [19, 41, 42]. A fourth order polynomial is used to model the thrust and mass flow rate as a function of the throttle, ϕ. The throttle range is $-1 \leq \phi \leq 1$. The mass flow rate parameterization is

$$\dot{m}\,[mg/s] = \sum_{i=0}^{4} C_i \phi^i \tag{D.1}$$

The thrust is not modeled directly, instead one half the thrust squared is parameterized.

$$0.5T^2\,[N * N] = \sum_{i=0}^{4} C_i \phi^i \tag{D.2}$$

Unlike the thrust and mass flow rate parameterization the power is a linear function of the throttle. The power used is given by

$$P\,[kW] = \frac{P_{max} - P_{min}}{2}(\phi + 1) + P_{min} \tag{D.3}$$

The coefficients used to parameterize the NSTAR, NEXT, and BPT-4000 thruster are given in Table D.1 and Table D.3. The coefficients are derived from references [19, 41, 42].

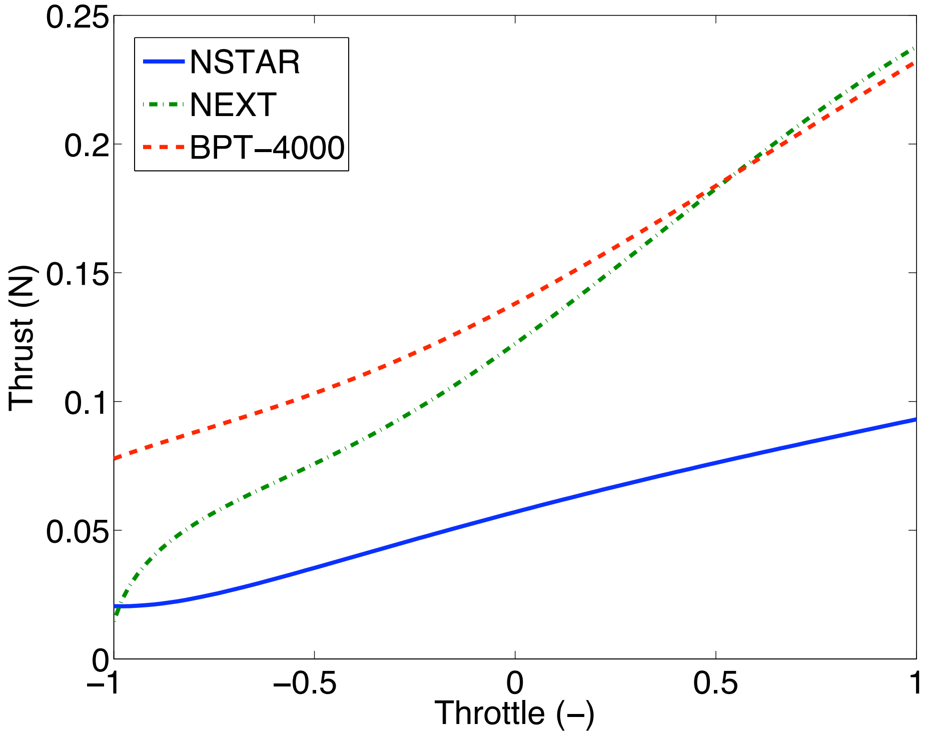


Figure D.1: The thrust for the NSTAR, NEXT, and BPT-4000 thrusters.

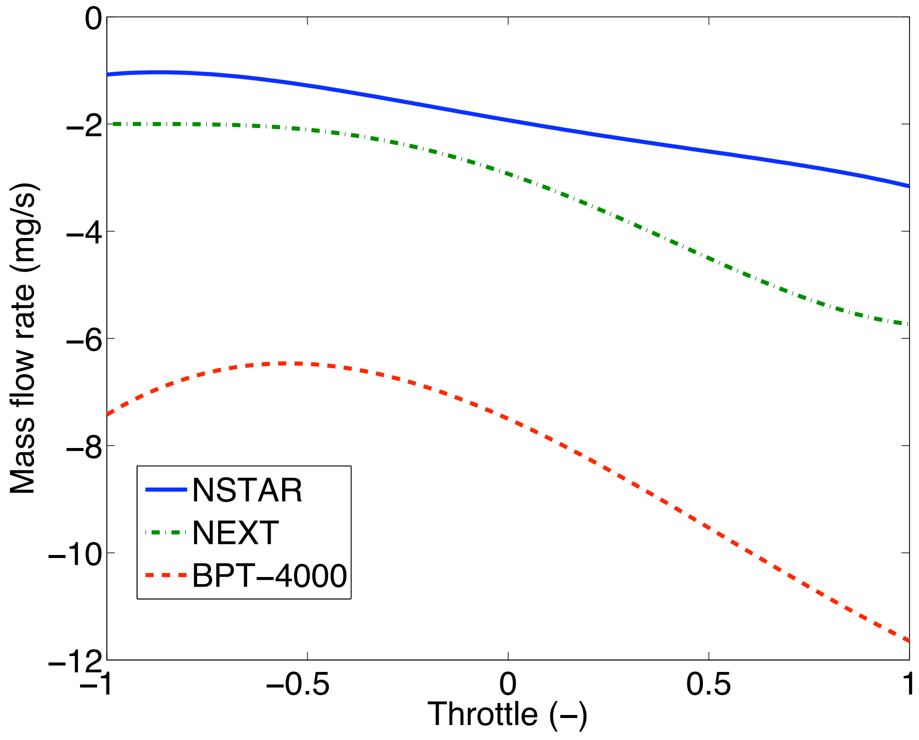


Figure D.2: The mass flow rate as function of the throttle parameter.

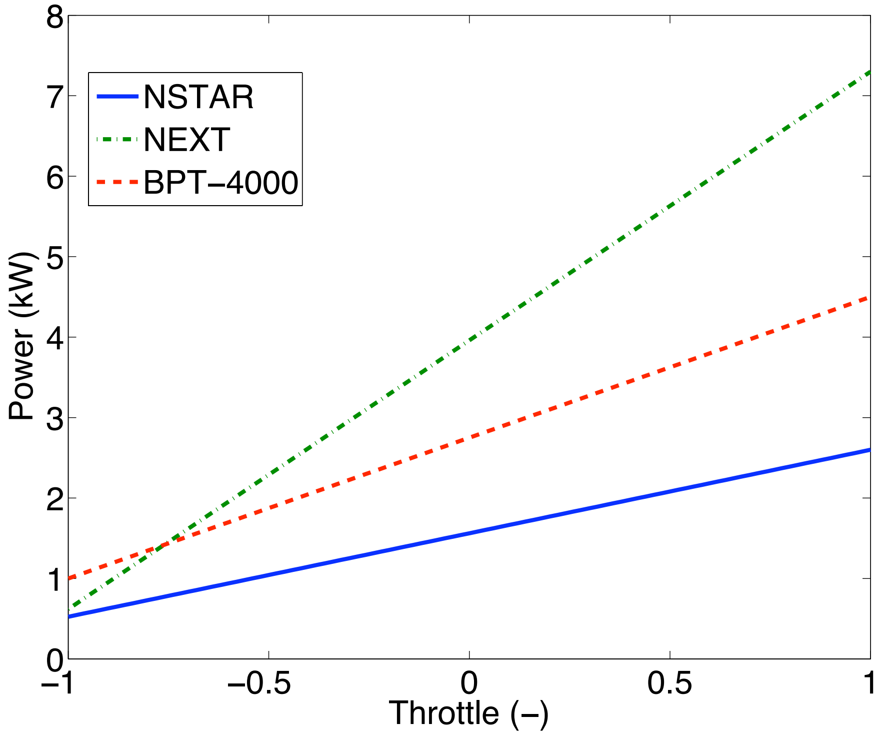


Figure D.3: The power utilized as a function of the throttle.

Table D.1: Mass flow rate parameterization coefficients

Thruster	Note	C_0	C_1	C_2	C_3	C_4
NSTAR	Q-mod	-1.93	-1.29	0.23	0.25	-0.42
NEXT	High specific impulse	-2.92	-2.57	-1.68	0.70	0.74
BPT-4000		-7.50	-3.37	-2.01	1.26	-0.02

Table D.2: Thrust constraint parameterization coefficients

Thruster	Note	C_0	C_1	C_2	C_3	C_4
NSTAR	Q-mod	0.00162	0.00234	0.00050	-0.00028	0.00013
NEXT	High specific impulse	0.00749	0.01372	0.00994	0.00036	-0.00325
BPT-4000		0.00952	0.01143	0.00660	0.00051	-0.00115

Table D.3: Minimum and maximum power for various thrusters

Thruster	Note	P_{min} (kW)	P_{max} (kW)
NSTAR	Q-mod	0.525	2.6
NEXT	High specific impulse	0.620	7.3
BPT-4000		1.0	4.5

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Monika Auweter-Kurtz. Optimization of electric thrusters for primary propulsion. In *37th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, number 2001-3347. AIAA, July 2001.

[2] John T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[3] John R. Brophy. Ion propulsion system design for the comet nucleus sample return mission. In *36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, number AIAA 200-3414, 2000.

[4] John R. Brophy. Nasa's deep space 1 ion engine. *Review of Scientific Instruments*, 73(2):1071–1078, February 2002.

[5] John R. Brophy and Muriel Noca. Electric propulsion for solar system exploration. *Journal of Propulsion and Power*, 14(5):700–707, September-October 1998.

[6] John R. Brophy, Marc D. Rayman, and Betina Pavri. Dawn: An ion-propelled journey to the beginning of the solar system. In *2008 IEEE Aerospace Conference Digest*. IEEE, 2008.

[7] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control*. Hemisphere Publishing Corp., 1975.

[8] Lorenzo Casalino and Guido Colasurdo. Optimization of variable-specific-impulse interplanetary trajectories. *Journal of Guidance, Control, and Dynamics*, 27(4):678–684, 2004.

[9] Lorenzo Casalino and Guido Colasurdo. Trade-off between payload and trip-time for ep interplanetary trajectories. Number AIAA Paper 2004-3805, July 2004.

[10] John L. Crassidis and John L. Junkins. *Optimal Estimation of Dynamic Systems*. Applied Mathematic and nonlinear science. Chapman and Hall, 2004.

[11] Michael L. Cupples and Shaun E. Green. Solar electric and chemical propulsion for a titan mission. *Journal of Spacecraft and Rockets*, 43(5):1077–1083, September-October 2006.

[12] J. M. A. Danby. *Fundamentals of Celestial Mechanics*. Willmann-Bell, 1988.

[13] Douglas I. Fiehler, Ryan Dougherty, and David Manzell. Electric propulsion system modeling for the proposed prometheus 1 mission. In *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 2005.

[14] Douglas I. Fiehler and Ralph L. McNutt Jr. Mission design for the innovative interstellar explorer vision mission. *Journal of Spacecraft and Rockets*, 43(6):1239–1247, November-December 2006.

[15] D. Milligan D. Gestal and O Camino. Smart-1 electric propulsion: An operational perspective. In *SpaceOps 2006 Conference*, number 2006-5767. AIAA, 2006.

[16] Philip E. Gill, Walter Murray, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[17] Dan M. Goebel and Ira Katz. *Fundamental of Electric Propulsion: Ion and Hall Thrusters*. JPL SPACE SCIENCE AND TECHNOLOGY SERIES. John Wiley and Sons, 2008.

[18] Curt A. Henry. An introduction to the design of the cassini spacecraft. *Space Science Reviews*, 104(1-4):129–153, July 2002.

[19] Richard R. Hofer, Thomas M. Randolph, David Y. Oh, and John Steven Snyder. Evaluation of a 4.5 kw commercial hall thruster system for nasa science missions. In *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, number AIAA 2006-4469, 2006.

[20] Dario Izzo. Lambert's problem for exponential sinusoids. *Journal of Guidance, Control, and Dynamics*, 29(5):1242–1245, 2006.

[21] David H. Jacobson. Differential dynamic programming methods for solving bang-bang control problems. *IEEE Transactions on Automatic Control*, AC-13(6):661–675, December 1968.

[22] David H. Jacobson. New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach. *Journal of Optimization Theory and Applications*, 2(6):411–440, 1968.

[23] Robert G. Jahn. *Physics of Electric Propulsion*. Dover Publications, Inc., 2006.

[24] Carl G Sauer Jr. Optimization of multiple target electric propulsion trajectories. Number AIAA-1973-205. AIAA, January 1973.

[25] Craig A Kluever. Heliospheric boundary exploration using ion propulsion spacecraft. *Journal of Spacecraft and Rockets*, 34(3):365–371, May-June 1997.

[26] Craig A Kluever and Kun-Rong Chang. Electric-propulsion spacecraft optimization for lunar missions. *Journal of Spacecraft and Rockets*, 33(2):235–239, 1996.

[27] Christophe R. Koppel. The smart-1 hall effect thruster around the moon: In flight experience. In *Presented at the 29th International Electric Propulsion Conference*, 2005.

[28] Hitoshi Kuninaka, Kazutaka Nishiyama, Ikko Funaki, Tetsuya Yamada, Yukio Shimizu, and Jun'ichiro Kawaguchi. Asteroid rendezvous of hayabusa explorer using microwave discharge ion engines. In *29th International Electric Propulsion Conference*, number IEPC-2005-10, 2005.

[29] Hitoshi Kuninaka, Kazutaka Nishiyama, Ikko Funaki, Tetsuya Yamada, Yukio Shimizu, and Jun'ichiro Kawaguchi. Powered flight of electron cyclotron resonance ion engines on hayabusa explorer. *Journal of Propulsion and Power*, 23(3):544–551, May-June 2007.

[30] Hitoshi Kuninaka, Kazutaka Nishiyama, Yukio Shimizu, Tetsuya Yamada, Ikko Funaki, and Satoshi Hosoda. Re-ignition of microwave discharge ion engines on hayabusa for homeward journey. In *30th International Electric Propulsion Conference*, number IEPC-2007-9, 2007.

[31] Li Liao and Christine A. Shoemaker. Advantages of differential dynamic programming over newton"s method for discrete-time optimal control problems. Technical report, Ithaca, NY, USA, 1992.

[32] Guido Colasurdo Lorenzo Casalino and Dario Pastrone. Optimal low-thrust escape trajectories using gravity assist. *Journal of Guidance, Control, and Dynamics*, 22(5):637–642, 1999.

[33] M. G. Marcucci and J. E. Polk. Nstar xenon ion thruster on deep space 1: Ground and flight tests. *Review of Scientific Instruments*, 71(3):1389–1400, March 2000.

[34] Jean Pierre Marec. *Optimal Space Trajectories*, pages 15–17. Elsevier Scientific Pub. Co. New York, 1979.

[35] T. Troy McConaghy. *GALLOP Version 4.5 User's Guide*. Purdue University, West Lafayette, Indiana, September 2005.

[36] T. Troy McConaghy, Theresa J. Debban, Anastassios E. Petropoulos, and James M. Longuski. Design and optimization of low-thrust trajectories with gravity assists. *Journal of Spacecraft and Rockets*, 40(3):380–387, 2003.

[37] W G Melbourne and Carl G Sauer Jr. Optimum interplanetary rendezvous with power-limited vehicles. *AIAA Journal*, 1(1):54–60, 1963.

[38] D. M. Murray and S. J. Yakowitz. Differential dynamic programming and newton's method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, 43(3):395–414, July 1984.

[39] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Science+Business Media, Inc. New York, 1999.

[40] Cesar Ocampo. Finite burn maneuver modeling for a generalized spacecraft trajectory designa and optimization system. *Annuls New York Academy of Sciences*, pages 210–233, 2004.

[41] David Y. Oh. Evaluation of solar electric propulsion technologies for discovery-class missions. *Journal of Spacecraft and Rockets*, 44(2):399–411, March-April 2007.

[42] David Y. Oh and Dan M. Goebel. Performance evaluation of an expanded range xips ion thruster system for nasa science missions. In *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, number AIAA 2006-4466. AIAA, 2006.

[43] Prashant Patel, Daniel Scheeres, and Alec Gallimore. Maximizing payload mass fractions of spacecraft for interplanetary electric propulsion missions. *Journal of Spacecraft and Rockets*, 43(4):822–827, 2006.

[44] Prashant Patel, Daniel Scheeres, Alec Gallimore, and Thomas Zurbuchen. Automating trade studies for electric propulsion mission studies. Number AAS 06-152, January 2006.

[45] Prashant Patel, Daniel Scheeres, and Thomas Zurbuchen. Spacecraft trajectories a shape based approach: Analysis and optimization. Number AAS 05-130, January 2005.

[46] Michael J. Patterson and Scott W. Benson. Next ion propulsion system development status and performance. In *43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 2007.

[47] Anastassios Petropoulos, Theresa Kowalkowski, Daniel Parcher, Paul Finlayson, Ed Rinderle, Matthew Vavrina, Jon Sims, Ryan Russell, Try Lam, Powtawche Williams, Gregory Whiffen, Nathan Strange, Jennie Johannsen, Chen-Wan Yen, Carl Sauer, Seungwon Lee, and Steven Williams. Response to the first act competition on global trajectory optimisation. Pasadena, CA : Jet

Propulsion Laboratory, National Aeronautics and Space Administration, 2006, February 2006.

[48] Anastassios Petropoulos and James Longuski. Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories. *Journal of Spacecraft and Rockets*, 41(5):787–796, 2004.

[49] Anastassios E. Petropoulos and Jon A. Sims. A reveiw of some exact solutions to the planer equations of motion of a thrusting spacecraft. 2002.

[50] Anastassios Evangelos Petropoulos. *A Shape-Based Approach to Automated, Low-Thrust Gravity-Assist Trajectory Design*. PhD thesis, Purdue University, 2001.

[51] Tara Polsgrove, Larry Kos, and Randall Hopkins. Comparison of performace predictions for new low-thrust trajectory tools. Number AIAA 2006-6742, August 2006.

[52] Marc D. Rayman. The deep space 1 extended mission: Challanges in preparing for an encounter with comet borrelly. *Acta Astronautica*, 51(1-9):507–516, 2002.

[53] Marc D. Rayman and Steven N. Williams. Design of the first interplanetary solar electric propulsion mission. *Journal of Spacecraft and Rockets*, 39(4):589–595, 2002.

[54] Ryan Russell. Primer vector theory applied to global low-thrust trade studies. *Journal of Guidance, Control, and Dynamics*, 30(2):460–472, March-April 2007.

[55] Jon A. Sims, Paul A. Finlayson, Edward A. Rinderle, Matthew A. Vavrina, and Theresa D. Kowalkowski. Implementation of a low-thrust trajectory optimization algorithm for preliminary design. Number AIAA 2006-6746, August 2006.

[56] Gilbert Strang. *Linear Algebra and its Applications*. Harcourt College Publishing, Orlando FL, 1998.

[57] David A Vallado. *Fundamentals of Astrodynamics and Applications*. Microcosm, Inc, 2 edition, May 2001.

[58] Eric W. Weisstein. Chebyshev polynomial of the first kind, May 2008.

[59] Greg Whiffen and Jon Sims. Application of a novel optimal control algorithm to low-thrust trajectory optimization. Number AAS 01-209, February 2001.

[60] Greg Whiffen and Jon Sims. Application of the sdc optimal control algorithm to low-thrust escape and capture trajectory optimization. Number AAS 02-208, January 2002.

[61] Greg J. Whiffen. Static/dynamic control for optimizing a useful objective. Number Patent 6496741, December 2002.

[62] Greg J. Whiffen. Mystic: Implementation of the static dynamic optimal control algorithm for high-fidelity, low-thrust trajectory design. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, number 2006-6741. AIAA, August 2006.

[63] Byoungsam Woo and Victoria L. Coverstone. Application of solar electric propulsion to a comet surface sample return mission. *Journal of Spacecraft and Rockets*, 43(6):1225–1230, November-December 2006.

[64] C Yam, T McConaghy, K Chen, and J Longuski. Preliminary design of nuclear electric propulsion missions to the outer planets. Number AIAA Paper 2004-5393, August 2004.

[65] Thomas Zurbuchen, Prashant Patel, Len Fisk, G. Zank, R. Malhotra, Herb Funsten, R. A. Mewaldt, and NASA Interstellar Probe Vision Mission Team. *NASA Space Science Vision Missions*, volume 224 of *Progress in Astronautics and Aeronautics*, chapter 5, pages 155–190. AIAA, 2008.

[66] Thomas Zurbuchen, Prashant Patel, Alec Gallimore, Daniel Scheeres, N. Murphey, G. Zank, R. Malhotra, Herb Funsten, and NASA Interstellar Probe Vision Mission Team. Interstellar probe: Breakthrough science enabled by nuclear propulsion. *Space Technology: Space Engineering, Telecommunication, Systems Engineering and Control*, 25(3-4), 2005.