ROBUST PRELIMINARY DESIGN FOR MULTIPLE GRAVITY ASSIST SPACECRAFT TRAJECTORIES

BY

DONALD HAMILTON ELLISON

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Abstract

The development of a new spacecraft trajectory design method most often occurs because a particular capability does not exist. The invention is usually considered successful so long as it is capable of producing solutions to the problem in question, and thus satisfies a particular design requirement, or is mathematically elegant. When innovation favors the latter to the exclusion of the former, the interoperability of the new method with existent techniques, and the utility of the method in the context of the overall mission design process, from concept to flight operations, is not always realized.

The concepts introduced and developed throughout this work respond to specific preliminary mission design needs, but their development is also focused on maintaining, or improving, trajectory design workflow compatibility and efficiency. The techniques described address specific contributions to the multiple gravity-assist trajectory optimization state-of-the-art, however, each one is also an important component of a modern trajectory design paradigm and is valuable for its ability to be integrated with and streamline that process as a whole.

The bounded-impulse approximation is a widely used method for early stage trajectory design for low and high thrust vehicles. Many previous studies involving this method of design have focused on developing new or improved trajectory transcriptions. This dissertation introduces analytic techniques for calculating the Jacobian matrix for two existing bounded-impulse trajectory models. The calculations allow for the use of a smooth spacecraft power model. One such model is introduced that handles multiple thruster on-off events and a variety of logic programs. A smooth spline-based ephemeris system is also discussed that is compatible with the analytic Jacobian formulae.

Mission design activities associated with the NASA New Frontiers 4 proposal call identified a particular shortcoming of the popular Sims-Flanagan bounded-impulse model, namely that its control nodes are distributed equally in time, clustering them at apoapsis for eccentric transfers, which reduces the control authority at periapsis. In response to this, a partially regularized bounded-impulse model is introduced that distributes control nodes equally at both apses. The new transcription is capable of delivering the same mass to the target as a trajectory modeled using Sims-Flanagan, but requires far fewer control segments in the trajectory discretization to do so.

A bounded-impulse trajectory is usually sufficient to generate a first order (or better) estimate of a mission's mass budget, however, these low-fidelity models can prove problematic to converge inside a flight-fidelity design tool that models finite-burn arcs. Low-thrust trajectories in particular suffer from this issue due to the extended period of time that the thruster operates. Analytic partial derivative computations are introduced in this work that enable the replacement of bounded-impulse maneuvers and Keplerian propagation with numerical integration arcs without reducing the runtime performance such that the model becomes unusable for preliminary design tasks. These calculations are also compatible with a smooth electric power model and accommodate final time free problems. The resulting trajectory is shown to be sufficiently accurate such that the flight heritage tool MIRAGE can track it within acceptable error limits placed on the spacecraft's state vector.

Finally, improvements that this thesis makes to bounded-impulse trajectory models are leveraged to solve a challenging planetary satellite tour design problem. The robustness improvements that the analytic Jacobian formulae afford the trajectory transcription, are combined with a parallelized flyby tree pathfinding algorithm to produce a design framework capable of autonomously optimizing a moon tour mission similar to Galileo from launch through tour end using two-body dynamics.

To mom and dad.

Acknowledgments

No two people have contributed more to my technical and professional development than my academic adviser Professor Bruce Conway, who has been an invaluable guide to an impressive lineage of trajectory optimists that I am humbled to be a part of, and Jacob Englander, who continues to find uses for me and has been a mentor of unparalleled influence. I would also like to acknowledge and thank the other members of the committee, I am fortunate to have five subject matter experts as the final arbiters of this work.

I had many other role models and colleagues throughout my time at the University of Illinois. Ryne Beeson put up with me tagging along for runs and desolate rides. My fellow Canadians, and friends Alex Ghosh and Erik Kroeker were a constant reminder of home and were always available for technical help, even when they were trying to become astronauts. Life would be far less interesting without Bindu, Pat, Devin, Vishwa, Josh and all of the other graduate students with an astrodynamics problem.

Just before graduating I decided to get a real job, thank you to Geoff Wawrzyniak for giving me an opportunity. I am grateful to Steve Squyres at Cornell University for trusting an unknown graduate student to work on his dream, and for a celebrity while doing it. My newest friends and colleagues Kyle Hughes and Bruno Sarli (and Chikako Suzuki (Hirose) from long-distance) promise to make the coming years both productive and enjoyable, hats off to you and the rest of the mission designers. Some of the work described in this dissertation was supported by NASA grants NNX14AD27G and NNX17AE60G. Thank you to Rich Burns of the Space Science Mission Operations project and the Internal Research & Development (IRAD) program at NASA Goddard Space Flight Center for that support as well as Jeremy Knittel for letting me work on NELLS.

My wife Nikita continues to prove everyday that being a mom requires, without question, more dedication, commitment and patience than is necessary to complete a Ph. D.. She has been the single greatest source of support throughout this process and has never asked for anything in return for putting up with me bringing work home every night. I am reminded what true curiosity is every day by Charles whose early vocabulary included the names of the Galilean moons and by Adrien who is proof that big inspiration can come in small packages.

To my family at home, thank you to my brother Michael for showing me what it means to make a difference in the world and to my grandparents who always give the best advice. Many people have helped get me to where I am today, but my parents get the final mention. It all started with you and everything I am, have done, and will do is built on your shoulders, thank you for your love and for throwing me over countless finish lines.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACO Ant Colony Optimization

AD Automatic Differentiation

AGA aerogravity assist

BB branch and bound

BVP Boundary Value Problem

CATO Computer Algorithm for Trajectory Optimization

CONOPS concept of operations

COV Calculus of Variations

DE Differential Evolution

DPTRAJ/ODP Double Precision Trajectory and Orbit Determination Program

DSN Deep Space Network

DSMPGA Dynamic-Size Multiple Population Genetic Algorithm

EB Evolutionary Branching

EMTG Evolutionary Mission Trajectory Generator

GASP Gravity Assist Space Pruning

GSFC Goddard Space Flight Center

GTOC Global Trajectory Optimization Competition

HGGA Hidden Genes Genetic Algorithm

IPOPT Interior Point OPTimizer

JHUAPL Johns Hopkins University Applied Physics Laboratory

KKT Karush-Kuhn-Tucker

LRTS lazy race tree search

MONTE Mission analysis, Operations, and Navigation Toolkit Environment

MCTS Monte Carlo tree search

MGA Multiple Gravity Assist

MIRAGE Multiple Interferometric Ranging Analysis using GPS Ensemble

MOGA Multi-Objective Genetic Algorithm

MOSES Multiple Orbit Satellite Encounter Software

MPI message passing interface

NELLS NASA Exhaustive Lambert Lattice Search

NSGA Non-Dominated Sorting Genetic Algorithm

NSGA-II Non-Dominated Sorting Genetic Algorithm II

OD orbit determination

PLATO PLAnetary Trajectory Optimization

RTBP Restricted Three Body Problem

SA Simulated Annealing

SNOPT Sparse Nonlinear OPTimizer

SOI sphere of influence

SQP sequential quadratic programming

TOF time of flight

TPBVP Two Point Boundary Value Problem

VARITOP Variational calculus Trajectory Optimization Program

VILM v-infinity leveraging maneuver

SEP solar-electric propulsion

SRP solar radiation pressure

AU Astronomical Unit

GSFC Goddard Space Flight Center

PI Principal Investigator

IAU International Astronomical Union

JPL Jet Propulsion Laboratory

NASA National Aeronautics and Space Administration

SPICE Spacecraft Planet Instrument Camera-matrix Events

STK Systems Tool Kit

NLP nonlinear program

MBH monotonic basin hopping

FBS forward-backward shooting

MGALT multiple gravity assist with low-thrust

MGALTS multiple gravity assist with low-thrust using a Sundman transformation

MGA-1DSM Multiple Gravity Assist with One Deep Space Maneuver

MGAnDSMs multiple gravity assist with n deep-space maneuvers using shooting

FBLT finite-burn low-thrust

ESA European Space Agency

ACT Advanced Concepts Team

GA genetic algorithm

GALLOP Gravity Assisted Low-thrust Local Optimization Program

MALTO Mission Analysis Low-Thrust Optimization

PaGMO Parallel Global Multiobjective Optimizer

HOC hybrid optimal control

HOCP hybrid optimal control problem

PSO particle swarm optimization

SEPTOP Solar Electric Propulsion Trajectory Optimization Program

STOUR Satellite Tour Design Program

PaGMO Parallel Global Multiobjective Optimizer

HDDP Hybrid Differential Dynamic Programming

ACT Advanced Concepts Team

GMAT General Mission Analysis Toolkit

RTG radioisotope thermal generator

ASRG advanced Stirling radiosotope generator

ARRM Asteroid Robotic Redirect Mission

PPU power processing unit

STM state transition matrix

MTM maneuver transition matrix

DSM deep-space maneuver

Necessity is the mother of invention.

- unknown

Chapter 1

Introduction

Spacecraft trajectory design is a broad concept that, in order to be carried out successfully, must draw upon specialized skill sets from a variety of technical disciplines. The specific knowledge required to perform the design tasks differs considerably depending on the particular category of mission that is being supported. As spacecraft technology progresses, the menu of mission classifications continues to expand, and the potential responsibilities of a trajectory design engineer are becoming more extensive in turn. The demands placed on such an individual might include the quantification of the long-term station keeping requirements of an Earth-observing weather satellite, designing a many-hundred revolution transfer for an electric-bus commercial satellite to geostationary orbit, investigating efficient transfers in Earth-Moon space that exploit the natural dynamics of a multi-body system, or even confronting the enormous combinatoric challenge of constructing a planetary satellite tour. The focus of this work is restricted to the preliminary design techniques associated with MGA trajectories. Many of the concepts that are discussed have been incorporated into the planetary mission design process at the National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC). With this in mind, discussion is provided throughout that illustrates how a particular characteristic of a design method lends itself well to a NASA flight project or proposal effort.

While the methods described in this work are most appropriately characterized as preliminary design techniques, it is important to emphasize that a major aspect of their utility is that the trajectory analysis products that they are capable of producing are compatible with higher fidelity design processes that are carried out in the later stages of a mission's lifecycle. In addition, a critical underlying theme shared by the algorithms and supporting frameworks discussed in this dissertation, is that of automation; not only of each individual design step, but of the process as a whole. The high cadence of a modern NASA Science Mission Directorate flight project proposal effort, be it for a strategic or Principal Investigator (PI)-led mission, requires that a trajectory design team be able to rapidly respond to a systems engineering environment that is in constant flux. This is particularly true of a concept study that is based around an electric propulsion system, as the spacecraft hardware and trajectory design processes are inseparable. In order to be successful, trajectory analysts must employ techniques that are not only thorough in their exploration of a complex design space, but are highly flexible and ideally autonomous in order to maximize productivity. With this as motivation, it is important to describe how a planetary mission proposal is supported from initial inception through to flight operations and how the contents of this document contribute to that process.

1.1 The Path to Flight

The planning of an interplanetary science mission is a complex task and an important requirement placed on the flight dynamics team is that it be able to deliver trajectory solutions during any phase of the process beginning with initial concept studies through flight operations. A successful design workflow must not only feature tools and techniques for producing trajectory solutions at any fidelity level, but ideally it will also enforce process requirements that ensure the transition of solutions from low and medium fidelity to flight-fidelity is reproducible and is achieved with minimal human in-the-loop activity.

Figure 1.1 depicts, at a high level, the mission design process in place at several organizations. The names of some example software tools typically employed at each stage of the mission design process are listed from left to right in the order that are usually leveraged. Of particular note is that the European Space Agency (ESA) Advanced Concepts Team (ACT), does not support a complete mission design infrastructure, and their inclusion in this discussion is for comparison reasons. The ACT is primarily a research-focused organization that is not generally required to support flight projects, and for this reason the majority of their contributions are in the form of leading theoretical work. Much of the research described in the following chapters owes its origins to the past and present members of this world-class organization.

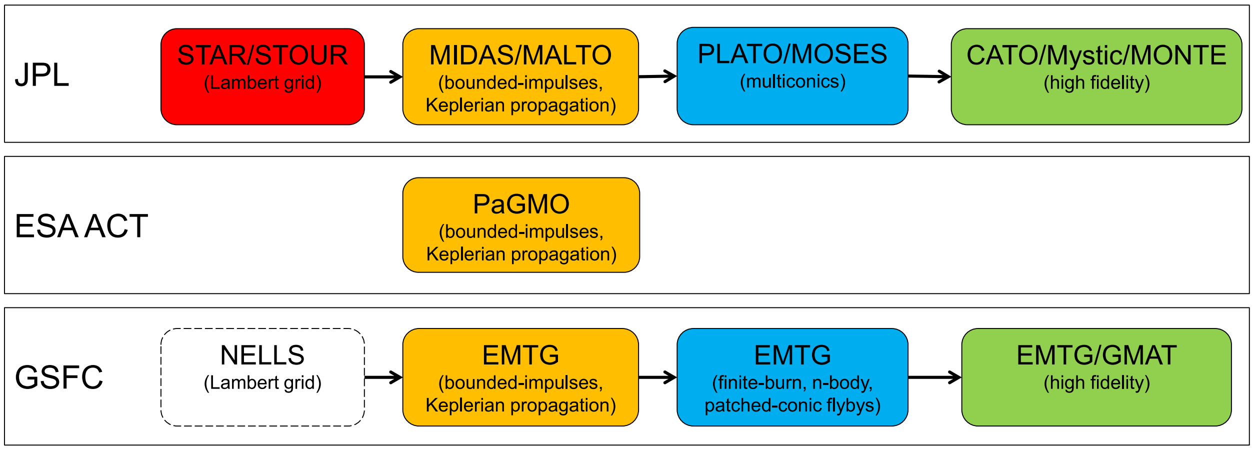

JPL
STAR/STOUR
(Lambert grid)
MIDAS/MALTO
(bounded-impulses, Keplerian propagation)
PLATO/MOSES
(multiconics)
CATO/Mystic/MONTE
(high fidelity)
ESA ACT
PaGMO
(bounded-impulses, Keplerian propagation)
GSFC
NELLS
(Lambert grid)
EMTG
(bounded-impulses, Keplerian propagation)
EMTG
(finite-burn, n-body, patched-conic flybys)
EMTG/GMAT
(high fidelity)

Figure 1.1: Comparison of mission design workflow paradigms from several institutions.

1.1.1 Preliminary Design

Traditionally, a mission planning effort begins by using frameworks that favor simplified models and techniques that can support a breadth-oriented exploration of the design space of interest. Mission design usually requires a broad search phase if it features a complex combinatoric element (e.g. a large number of flyby target possibilities). The goal of a preliminary design analysis is to assess, at a fundamental level, the basic feasibility of the mission. Trajectory solutions resulting from this type of analysis will typically determine if the mass budget is within the correct order of magnitude and which flyby sequences are possible for a given launch window that can deliver the payload to its destination. Despite the fact that many preliminary design methods make use of simplified physics and hardware models, they can be extremely resource intensive from a computational point of view due to the size of most MGA design spaces.

An example broad search design implementation is the Satellite Tour Design Program (STOUR) developed at the Jet Propulsion Laboratory (JPL) for the Galileo mission to Jupiter [8]. The STOUR algorithm approximates trajectories as two-body conic sections connected with patched-conic flybys. Modernizations of the STOUR concept have been developed in recent years, most notably the STAR algorithm by Damon Landau at JPL, Explore by Lantukh and Russell [9] as well as other contributors at the Johns Hopkins University Applied Physics Laboratory (JHUAPL), and most recently the NASA Exhaustive Lambert Lattice Search (NELLS) by Jeremy Knittel at KinetX and the author. The entirety of Chapter 6 is dedicated to the contributions that this dissertation makes to NELLS, and by extension preliminary MGA mission design in general.

Often low-fidelity design methods are employed at the preliminary design stage as they are typically more straightforward to implement and are less computationally expensive. Such methods even exist for continuous-thrust trajectory design, where the spacecraft's control and state histories are often approximated using analytic techniques. Assumed shape techniques include those by Petropoulos, who modeled low-thrust transfers using exponential sinusoids [10], Wall and Conway who employed an inverse polynomial to model rendezvous trajectories [11], De Pascale and Vasile [12] and Vasile et al. [13] who extended the concept to spatial transfers and Taheri and Abdelkhalik [14] who employed a Fourier series to model continuous thrust trajectories. The primary advantage of these methods over more sophisticated ones is their fast execution time, which makes them particularly suitable for low-fidelity design.

Many preliminary design techniques also make use of the bounded-impulse maneuver approximation for both high-thrust chemical rocket motors as well as to generate approximations of continuous-thrust transfers. The latter is usually achieved using some variation of the method due to Sims and Flanagan [15]. The bounded-impulse trajectory approximation will be introduced in section 3.

1.1.2 Medium Fidelity Mission Design

Bounded-impulse methods are often sufficiently accurate to be used for high-level mission and systems trades. The simplified dynamics that constitute most low-fidelity techniques, however, generate trajectories of insufficient accuracy for some mission types or do not meet the minimum accuracy requirements for certain mission design activities such as Discovery and New Frontiers proposal work. For proposals in particular, at least one high-fidelity point solution is typically expected at the conclusion of Pre-Phase A. The flight-fidelity tools that produce these solutions, however, require a great deal of human analyst time that may not be available or affordable in the context of a proposal. In addition, the flight-fidelity tools are not always readily available to all teams wishing to submit a proposal. An intermediate level of design fidelity that bridges the gap between medium and high fidelity that is easy and inexpensive to use and is also readily available to all potential users is therefore highly desirable. Such a “medium” fidelity technique should avoid bounded-impulse approximations and rather integrate the equations of motion, and its dynamics model should include reasonably high fidelity models of propulsion and power systems as well as significant orbit perturbations such as third-body gravity and solar radiation pressure (SRP).

The main discriminator between this trajectory model and one of flight quality is that, even at this level of accuracy, patched-conic flybys are typically retained as the incorporation of fully-integrated flybys is only usually necessary for accurate navigational analysis. An example of this medium fidelity concept, referred to as the finite-burn low-thrust (FBLT) transcription was introduced by Englander et al. [16] and is the focus of Chapter 5. It is a unique aspect of the GSFC planetary mission design paradigm and is sufficient to ensure that a proposed trajectory is actually feasible and is not an artifact of the low-fidelity models available in most preliminary design tools. Another medium fidelity implementation capable of handling low-thrust trajectories was proposed by Yam et al. [17]. This method uses Taylor integration to propagate the spacecraft state subject to continuous-thrust, however, it is not capable of incorporating an accurate power model into the optimization process, and is therefore unfortunately not suitable for most preliminary design tasks. Another method that falls under this classification is the concept of multi-conic trajectory analysis [18–20]. The multi-conic method models perturbative accelerations acting on a spacecraft by considering the conic orbital motion due to each perturbation source separately and then combines these individual influences. In this sense, it is similar to Encke’s method of perturbations [21], the differentiating characteristic being that the multi-conic method does not rely on explicit numerical integration. The PLAnetary Trajectory Optimization (PLATO) and Multiple Orbit Satellite Encounter Software (MOSES) [22] software packages, which were developed at JPL and were used to aid in the design of the Galileo interplanetary cruise and satellite tour trajectories respectively, are examples of tools that employ this method.

1.1.3 High Fidelity Mission Design

A high-fidelity spacecraft reference trajectory does not rely on any simplified model assumptions, such as the bounded-impulse and patched-conics approximations. The differential equations of motion are forward-integrated using a fine fixed step size, or error-controlled adaptive step size, and typically include the gravitational influences of the planets and many of the most prominent small bodies in the solar system. Solar pressure is accounted for and higher-order gravitational models are used for all close encounters with massive bodies (i.e. flybys and rendezvous). If a thorough medium fidelity analysis was performed, large improvements to the design using differential correctors or local optimizers should not occur at this stage, and will only be employed to close the trajectory in the context of the high accuracy physics engine at the core of a final stage design tool. These software solutions include the General Mission Analysis Toolkit (GMAT) [23], FreeFlyer [24], Systems Tool Kit (STK) [25], and Copernicus [26].

A current major technical challenge faced in the field of preliminary trajectory design is the automatic robust generation of sufficiently accurate medium fidelity initial guesses for a high-fidelity solver for missions of extreme complexity. These difficult scenarios include many-revolution continuous-thrust transfers, low-energy trajectories and of particular importance to this dissertation, tours of planetary satellite systems. While high-fidelity trajectory generation is not discussed at length in the remainder of this document, it should be noted that all of the contributions described herein serve to enable and improve the procedure for arriving at high-fidelity and flight-fidelity solutions.

1.1.4 Flight Fidelity

The final step in the mission design process is the generation of the reference trajectory for the spacecraft to track and the navigation solutions and resulting control commands that it requires to do so. There exist very few software implementations capable of generating navigation solutions for a spacecraft in flight. The Double Precision Trajectory and Orbit Determination Program (DPTRAJ/ODP) is a notable example that was used extensively to support the Voyager program at JPL. The licensed version of this software

(Multiple Interferometric Ranging Analysis using GPS Ensemble (MIRAGE)) has been used to support NEAR, Genesis, Stardust, Deep Space 1 and many other planetary missions at JPL, and more recently at KinetX, MESSENGER, New Horizons and OSIRIS-REx. Mystic, developed by Greg Whiffen and others at JPL [27] is a six degree-of-freedom optimization program that was used to design the Dawn trajectory, and is currently used to fly the spacecraft as well. Finally, JPL recently made their operational navigation software Mission analysis, Operations, and Navigation Toolkit Environment (MONTE) available to license. The MONTE software has been used for all operational navigation tasks at JPL since 2012.

Flight-validated trajectory software packages feature all of the modeling fidelity of a high-fidelity tool, but include additional capabilities that allow them to be integrated into a spacecraft's operational ground system. For example, they must include the ability to process radiometric data from the Deep Space Network (DSN), which is used to produce an orbit determination update and then appropriately command the spacecraft such that it tracks the reference path. These tasks require a close interaction between the mission design and flight navigation teams, a concept that is discussed further in Chapter 5. In particular, it is essential that the flight heritage tool be able to produce control commands capable of tracking the reference trajectory provided by the mission designers.

1.1.5 Optimization

The complexity of many modern space mission designs often requires that optimization methods be leveraged during the preliminary design process, and even for the generation of flight solutions. A variety of optimization techniques have been applied to solve trajectory optimization problems, however most can be categorized as indirect, direct or hybrid methods. Indirect methods [28–34] use necessary conditions derived using the Calculus of Variations (COV) [35,36] and solve the resulting Hamiltonian boundary value problem that results. Direct methods [37–42] cast the trajectory optimization problem as a parameter optimization problem and directly extremalize a cost function, and hybrid approaches [27,43–45] rely on Bellman's Principle of Optimality of a dynamic program.

The Evolutionary Mission Trajectory Generator (EMTG) introduced by Englander and Conway [46–48] is a direct method hybrid optimal control (HOC) automaton under development at GSFC that is capable of optimizing both high and low-thrust trajectories at several levels of fidelity. The Mission Analysis Low-Thrust Optimization (MALTO) software package created by Sims et al. at JPL [49], Gravity Assisted Low-thrust Local Optimization Program (GALLOP) [50,51] produced at Purdue University and Parallel Global Multiobjective Optimizer (PaGMO) [52] developed by the ESA ACT are other tools capable of producing low-fidelity designs that rely on direct optimization methods. Other design tools that employ optimization techniques include Variational calculus Trajectory Optimization Program (VARITOP) and the more precise Solar Electric Propulsion Trajectory Optimization Program (SEPTOP), both of which employ COV based techniques and are classified as indirect solvers. For a more comprehensive review of trajectory optimization techniques, the reader is referred to the *Journal of Guidance, Control, and Dynamics* survey paper by John Betts [53] and the book edited by Conway [54].

The incorporation of accurate hardware modeling into the trajectory design process is required for any credible mission design framework (primarily for solar electric spacecraft) and is a topic of current research [55]. Embedding realistic spacecraft power hardware models into most optimization algorithms can be challenging and is the topic of section 2.4. Despite these challenges, accurate power hardware modeling is a requirement for Discovery and New Frontiers proposals and for this reason, when possible, a trajectory design effort should account for such models at the earliest possible stage. The contributions to preliminary trajectory design and optimization methods described in chapters 3, 4, and 5 account for the requirement that accurate power models be incorporated at all stages of the mission design process.

1.2 Multiple Gravity Assist Trajectories

A gravity assist (or flyby) is the intentional execution of a close-proximity pass by a massive body for the purpose of altering a spacecraft's trajectory. Bending of the flight path is achieved via a momentum exchange between the spacecraft and the flyby target body. This maneuver provides a means of altering a spacecraft's velocity vector without the use of any on-board propellant. Who precisely first developed the concept of the gravity assist and realized that it may be particularly enabling when applied to interplanetary trajectories is the subject of a typical debate that embroils most technical fields from time to time. Suffice it to say that several individuals have contributed to the subject of gravity assists starting with Russian scientists Yuri Kondratyuk and Friedrich Zander in the early 20^{th} century. The application of the idea to modern trajectory design in a technical sense is probably most appropriately attributed to the work of JPL mathematician Michael Minovitch [56] in 1961-1963 although others had suggested its use prior to that including the science fiction writers Ray Cummings [57] and Arthur C. Clarke [58] [1].

The gravity assist maneuver was first performed by the Soviet probe Luna 3 in 1959 in order to photograph the far side of the moon. The use of a gravity assist trajectory for the express purpose of reaching another

[1]Clarke interestingly also cites the work of Lawden [59] as the earliest reference that he was able to find on the subject at the time.

planet in our solar system was first made by the Mariner 10 spacecraft as it passed by Venus on its way to Mercury in 1974. In many cases, the ability to use flyby trajectories may actually determine whether or not a certain mission profile is even feasible, as was the case with the Voyager 1 and 2, Galileo, Ulysses, Cassini, MESSENGER, and Juno spacecraft as well as, most recently, the OSIRIS-REx mission to the asteroid Bennu. For Cassini in particular, the use of Titan as a flyby target was the key to the success of the mission. Cassini launched with enough onboard propellant for about 2.4 km/s of velocity change. By the time it reached Saturn, the spacecraft had performed one 450 m/s deep space maneuver and a 626 m/s capture maneuver at Saturn, and had only enough propellant remaining for about 1 km/s of velocity change maneuvers. Titan flybys provided about 90 km/s of velocity change over the course of Cassini's main, equinox and solstice tours, thereby enabling such a lengthy and complex mission.

1.2.1 Solving the Multiple Gravity Assist Problem: Pathfinding and Pathsolving

The MGA problem is difficult to solve for several reasons. The first is that there are typically many flyby targets to consider, resulting in a complex combinatorics problem. An MGA trajectory problem also includes continuous variables including flight times and launch geometry parameters such as right ascension, declination and launch $C3$. If maneuvers are to be included, this significantly complexifies the problem, which in turn typically requires a corresponding increase in the sophistication of the chosen solution method. Furthermore, a particular method might include elements of optimization that seek to identify solutions that extremalize a particular cost function such as propellant consumed or time of flight (TOF).

A particularly well constructed summary of global MGA solution methods is provided by Lantukh in his dissertation [9], however, a brief discussion follows here for the sake of completeness. As Lantukh notes, solving the MGA problem requires that the trajectory designer address both the pathfinding (planning) as well as the pathsolving (scheduling) aspects of the problem.

Pathfinding involves determining the order of countable categorical events required to achieve the mission objectives. These discrete events include, for example, the order of flyby targets. The main challenge encountered when solving MGA trajectory pathfinding problems is the high degree of combinatoric complexity associated with most mission design spaces. Pathsolving is the act of determine a solution, or family of solutions, associated with a particular pathfinding configuration. In contrast to pathfinding, pathsolving typically involves a continous design space where the solution technique must select continuous or mixed-integer decision parameters. A particular solution method might address one or both of these aspects of the

MGA problem.

1.2.2 Graphical Methods

The oldest class of MGA solution techniques are those requiring significant manual calculations and are often paired with graphical aids that help guide the trajectory design engineer in the solution process. These methods are characterized by their heavy reliance on human-in-the-loop mathematical analysis and intuition. Strange and Longuski introduced the concept of a Tisserand plot [60], which are plots of constant v_∞ contours. These plots indicate whether a ballistic transfer from one flyby body to another is feasible from an energy perspective, but not from a phasing perspective. An example Tisserand plot is shown in Fig. 1.2. A follow-on analysis is therefore required using a solver such as STOUR or EMTG to determine whether a feasible transfer actually exists. Campagnola and Russell [61] augmented this concept to include transfers featuring impulsive maneuvers, and then extended it further for use in low-energy multi-body mission design activities [62].

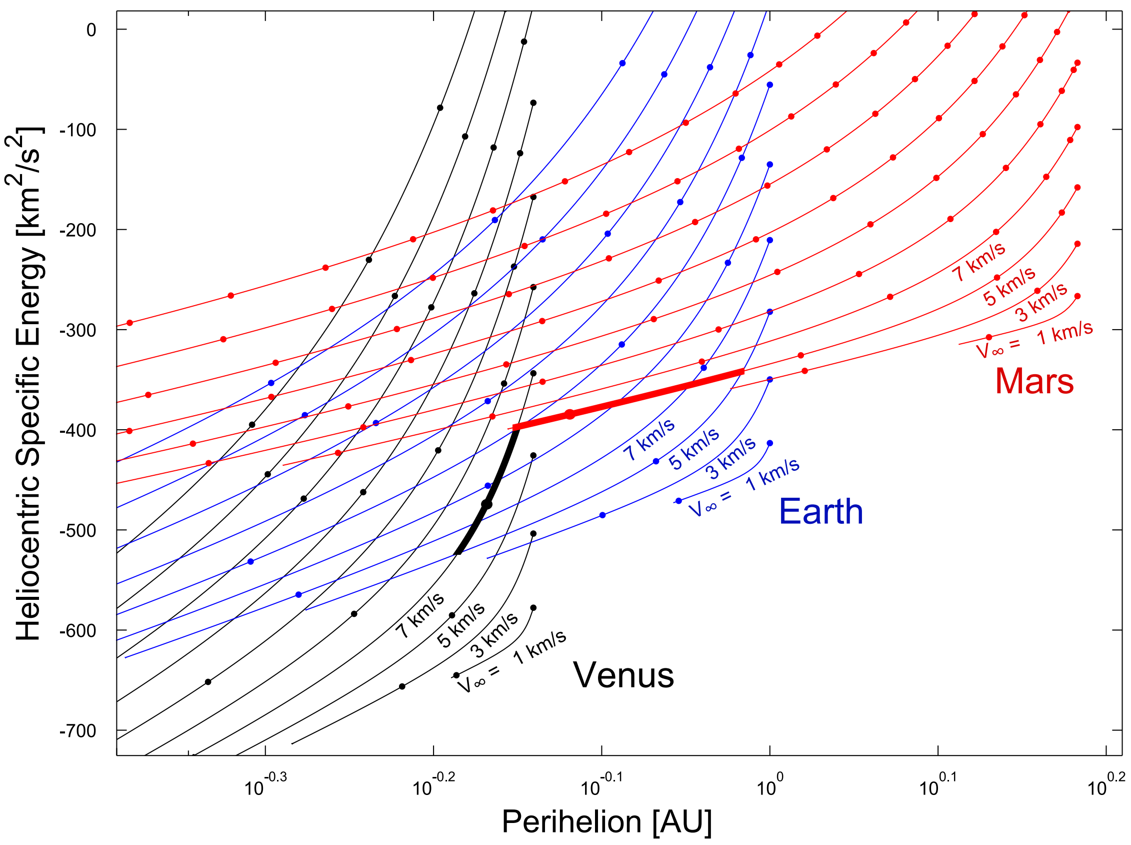


Figure 1.2: Figure from [2]. Tisserand graph for heliocentric trajectories, with example EVME trajectory (in bold). The plot shows that for an Earth launch v_∞ of 5 km/s, Venus can be reached, then Mars, and finally back to Earth with an arrival v_∞ of 5 km/s.

Another example of a graphical aid that has been incorporated into mission design workflows is the v_∞ sphere [63]. This geometric construct is a sphere with radius equal to the flyby hyperbolic excess velocity vector magnitude that is formed by the loci of all v-infinity vector tips. The sphere is a map of all possible orbits around the central body. Various studies have characterized the various flyby trajectories that result from placing the tip of the v_∞ vector at various locations on this sphere. Russell and Ocampo examined free return trajectories [64], Strange et al. performed a thorough mapping of the globe [3] and Woolley and Scheeres [65] applied the concept to single-body moon tour endgame design. Most recently, Lantukh and Russell [66] and Lantukh [9] extended the method past a pure graphical aid and incorporated the v_∞ sphere into an automated method for incorporating $n\pi$ transfers into the global MGA search problem.

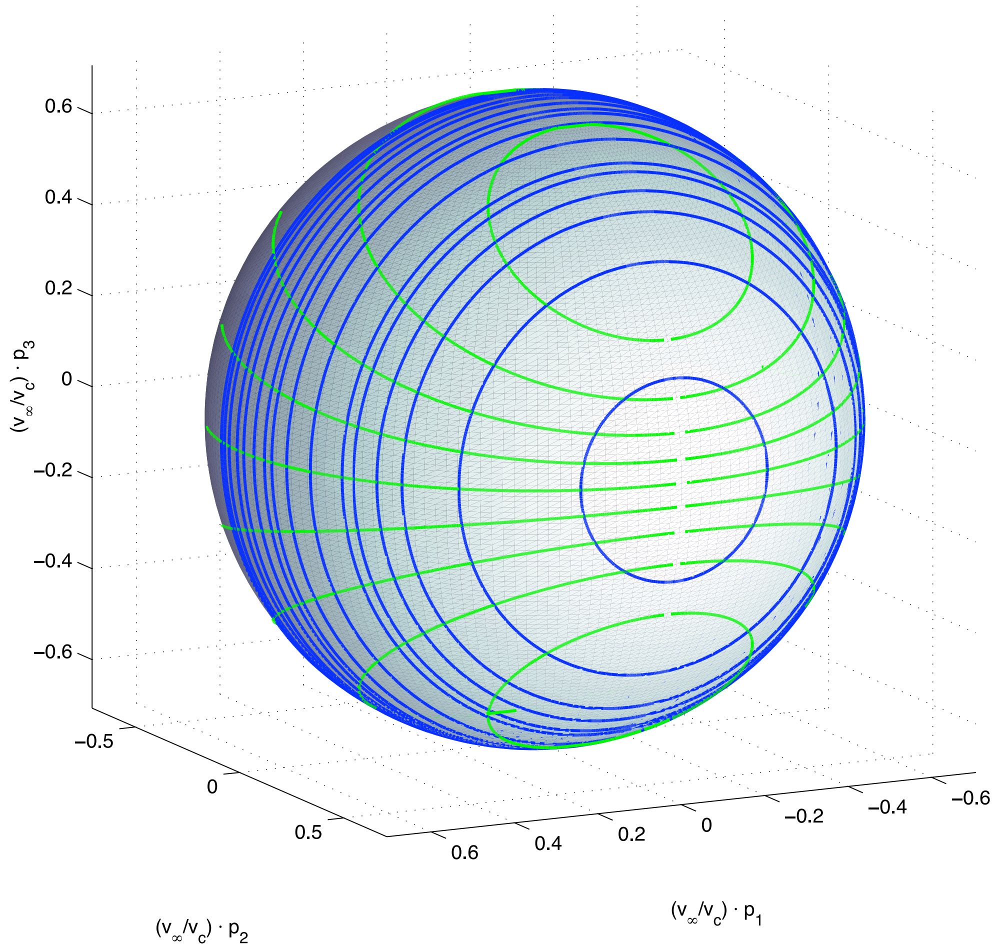


Figure 1.3: A simple v_∞ sphere, where blue contours represent orbits with the same period and green contours map orbits with the same inclination. Reproduced from [3].

1.2.3 Exhaustive Methods

The second category of MGA solution techniques are those that exhaustively explore the problem design space. These methods employ a grid to discretize the problem's continuous variables, thereby enabling a systematic search of a portion of or, for problems of very low dimensionality, the entirety of the decision space. Perhaps the most well-known of these methods is the one employed by STOUR [8], however, as mentioned previously there have been several more recently developed implementations that fall into this category including STAR, Explore [9], SOURCE [67] and NELLS. The core algorithm of STOUR requires the user to define *a priori* a sequence of flyby targets, then the program will step through a range of launch dates and compute all conic point-to-point ballistic trajectories connecting those bodies adhering to some specified launch date vs. time of flight to the first flyby target grid discretization. These two-body solutions may be obtained analytically by solving Lambert's problem and are connected using a patched-conics flyby model, a concept that is discussed further in section 2.3. Since the Galileo mission, STOUR has been automated [68] and its capabilities have been augmented to allow for the use of aerogravity assist (AGA)s [4, 69], powered flybys [70], deep-space maneuver (DSM)s [71] and shape-based approximations of low-thrust transfers [72].

It is becoming more common for parallel computing techniques to be incorporated into systematic search methods. This is often due to the large volume of local searches that must be executed in order to thoroughly explore the design space. Russell [73] employed parallel computing to solve a large number of Two Point Boundary Value Problem (TPBVP)s using primer vector theory for the Restricted Three Body Problem (RTBP). Izzo et al. [74] also used parallel computing to search the flyby sequence space for their solution to the Global Trajectory Optimization Competition (GTOC) 6 problem.

Exhaustive methods nearly always require that bounds be placed on the search domain, or a "pruning" mechanism be incorporated into the method, in order to make the problem computationally tractable in a reasonable amount of time. These strategies, commonly referred to as branch and bound (BB), can be incorporated directly into the search algorithm, such as with SOURCE, Explore and NELLS, or they can be used to limit the scope of the search, while not directly solving for MGA trajectories as is the case with the Gravity Assist Space Pruning (GASP) algorithm [75, 76]. Pruning mechanisms aside, exhaustive methods fundamentally operate by mapping the entire search space. Conversely, metaheuristic methods employ mechanisms that identify solutions without the need for total enumeration and often feature a stochastic search element.

1.2.4 Metaheuristic Methods

Metaheuristics are problem-independent, typically non-greedy, methods that search a given problem space according to some set of behavioral laws. Vinkó and Izzo [77] compare the global optimization performance of several of these methods applied to several benchmark trajectory design problems. A similar study was also carried out by Vasile et al. [78]. One interesting attribute that many metaheuristics share is that their derived search mechanism mimics the behavior of certain animal species such as particle swarm optimization (PSO) [79] and Ant Colony Optimization (ACO) [80]. Algorithms based around ACO were first applied to the MGA problem by Ceriotti and Vasile [81]. This method was then extended, by the same authors, to include a single DSM [82].

Solution techniques in this category often feature concepts rooted in evolutionary biology to guide their search and are often population based, meaning that they maintain a collection of candidate solutions as the method iteratively searches for the globally optimal solution. The first application of an evolution-inspired algorithm to the MGA problem was the work by Gage et al. [83]. Their study displayed the superiority of a genetic algorithm (GA)'s performance over that of a grid search and how that trend grew as the number of design variables increased. This work was followed by the investigations performed by Vasile [84] and Vasile and De Pascale [85]. Their method combined a global evolutionary algorithm called Evolutionary Branching (EB) with a local sequential quadratic programming (SQP) sequence. Differential Evolution (DE) [86] is a population-based evolutionary algorithm that has been applied extensively to the MGA problem. Olds et al. applied DE to the problem of interplanetary MGA trajectory design [87]. This was followed by Izzo et al. [88] who employed DE in conjunction with an asynchronous island model and Izzo et al. [89] who applied it to a generalized island model. The team led by Izzo also successfully incorporated DE into their search method for finding the best known (at the time) solution to the GTOC 6 problem [74]. Differential evolution was also used by Englander et al. [90] to solve the Multiple Gravity Assist with One Deep Space Maneuver (MGA-1DSM) problem. Vinkó and Izzo [77] developed a hybrid solution method that combined PSO and DE and applied it to the MGA problem. Finally, Englander [48] and Englander et al. [90] made use of this technique to solve the inner-loop problem of an interplanetary trajectory HOC automaton.

Genetic algorithms [91] are one of the most popular methods for solving trajectory design problems, especially those involving gravity assists. Rauwolf and Coverstone-Carroll [92] applied a GA to solve the interplanetary low-thrust problem, which was particularly important foundational work for researchers who eventually included gravity-assist events in the problem formulation. Conway et al. examined the MGA problem as a hybrid optimal control problem (HOCP) and proposed several two-stage solution techniques including a GA + nonlinear program (NLP) solver as well as one that combined BB + GA into the two-loop HOC solution process [93]. Wall and Conway examined missions to asteroids [94] using another HOC formulation that used a binary GA outer-loop solver to choose the sequence of asteroids and a real-valued GA combined with a novel shape-based trajectory approximation method to determine the trajectory between targets. Woo et al. [95] used a GA to choose initial guesses for the low-thrust trajectory optimization package SEPTOP. Gad and Abdelkalik developed the Hidden Genes Genetic Algorithm (HGGA) for solving the MGA problem with a free number of impulsive maneuvers [96]. Abdelkalik and Gad introduced the Dynamic-Size Multiple Population Genetic Algorithm (DSMPGA) for solving the same problem [97]. Recently, Englander applied the concept of the junk gene from evolutionary biology to the interplanetary MGA problem [48].

Multi-Objective Genetic Algorithm (MOGA)s, which seek to identify Pareto optimal solutions have also been used by several researchers for spacecraft trajectory optimization problems. Coverstone et al. [98] leveraged the Non-Dominated Sorting Genetic Algorithm (NSGA) developed by Srinivas and Deb [99] to solve single-phase interplanetary trajectories to Mars and Mercury. Vavrina and Howell [51] used a MOGA to choose initial values for the GALLOP program, which uses the Sims-Flanagan transcription method along with an NLP solver to solve the multiple gravity assist with low-thrust (MGALT) problem. Later Vavrina et al. [100] and Englander et al. [101] employed the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [102] to perform systems-level optimization of high and low-thrust interplanetary MGA trajectories.

Another class of metaheuristics are augmented local searches such as Simulated Annealing (SA) [103], Tabu search [104,105] and monotonic basin hopping (MBH) (Appendix A.1) [106,107]. These search strategies often include a gradient-based local search step, which is further enabled by an exploration mechanism that reinitializes the local search in the vicinity of previously discovered solutions, thus avoiding entrapment in local minima, the primary weakness of a gradient-based search. Johnson applied Tabu search [108] to the problem of multiple target tour trajectory problems. The metaheuristic method that is most relevant to this work is MBH. This method was first applied to the MGA problem by Addis et al. [109]. Yam et al. [110,111] combined the MBH iterative local search method with the Sims-Flanagan bounded-impulse model to compute approximations to continuous-thrust transfers. Englander and Englander [112] performed an extensive analysis that examined tuning the MBH algorithm for low-thrust trajectory optimization problems. The solution method of Yam et al. served as the seed for the HOC work by, Englander and Conway [47] and Ellison et al. [113] and is the topic of the following section.

1.3 Hybrid Optimal Control Trajectory Automaton

Spacecraft trajectory optimization problems belong to a general class of problems called hybrid optimal control problems [114–116]. The defining characteristic of an HOCP is the inclusion of both discrete and continuous variables in the parameterization of the problem. Specifically, the values of the discrete variables define the configuration of the sub-problem described by the continuous variables. When applied to an MGA problem, this means that an HOCP seeks to simultaneously solve both the pathfinding as well as the pathsolving problems that were discussed in section 1.2.1. Consider the problem of transferring a spacecraft from the Earth to another body in the solar system. A typical discrete variable set that characterizes this problem is the flyby sequence (including the option to not use gravity assists) that the spacecraft will use to complete the transfer to the target body. These discrete variables describe a particular transfer structure that is then locally optimized by varying continuous variables such as the launch epoch, the engine throttle time history and the TOF of each flyby leg. The first description of an HOC solution framework for aerospace mission/trajectory design problems can be found in the pioneering by Ross and D'Souza [117]. The first application of the HOC method to the interplanetary MGA problem was by Conway et al. [93], followed shortly by an extension to multi-objective HOC by Vavrina and Howell [51]. Recently, the incorporation of additional variables such as the spacecraft's solar array size, thruster type, launch vehicle employed and other systems-level discrete optimization variables have also been successfully incorporated into the HOCP structure [100, 101].

1.3.1 Hybrid Optimal Control Solver Structure

One method for solving HOCPs, such as the notional mission described above, is to use a nested optimization loop structure. This method utilizes an *outer-loop* to select values for the discrete variables. These in turn define a particular problem space in which the real-valued variables may be varied by the *inner-loop*. The purpose of the inner-loop solver is to evaluate a particular choice of outer-loop variable values based on a cost metric (e.g. the amount of spacecraft mass that was delivered to the target) in order to compare it against other discrete variable configurations. Examples of typical outer-loop solvers include genetic algorithms GAs, BB or even total enumeration (grid searches) for small problems. The inner-loop problem has been solved using a variety of methods, several of which are discussed later in sections 3.1, 3.2, 4.2 and 5.1. The most popular solution method is to discretize the problem and convert it into a nonlinear programming problem [118]. The specific nested loop solver implementation referenced in this work is the EMTG [46, 48], however, this nested loop solver structure has enjoyed widespread success, and several research efforts have exploited it for spacecraft trajectory design [110, 111, 119].

1.3.2 Outer-Loop Solver

In the context of the MGA problem, one of the primary roles of the outer-loop solver is to perform flyby sequence pathfinding. For example, if the specified mission is to transfer a spacecraft from Earth to Jupiter, on a given iteration of the outer-loop one or more sequences will be constructed from a set of possible gravity-assist target bodies (e.g. {Venus-Earth-Mars, Earth-Venus-Venus-Earth}) for the inner-loop to evaluate. The inner-loop will then solve the continuous optimization problem, and compute the resulting cost function value. This cost function information is, in turn, used by the outer-loop to assess the efficacy of any one flyby route.

Several algorithms have been shown to be effective at solving the HOC outer-loop problem. Branch and bound is perhaps the most straight-forward to both conceptualize and implement and forms the basis for most exhaustive MGA solution techniques. The BB method casts the set of all possible event sequences as a tree structure. For a multi-stage optimization problem, the total enumeration of the states that the problem variables may occupy are represented as nodes in the tree. Possible subsequent decisions are represented as child nodes that branch from the current parent node. Since traversing the tree of a problem with high combinatorial complexity, such as the MGA problem, rapidly becomes computationally intractable, partial solutions are pruned away at each level of the tree according to some bounding law. For example, the beam search tree traversal algorithm only branches a specified number or percentage of child nodes at each level of the tree according to the cost function value of the partial solutions.

Since typical cost functions for the MGA problem do not obey the law of superposition, that is each flyby-to-flyby phase of the problem is not independent of previous and subsequent phases, BB will not guarantee the identification of the globally optimal solution, and also means that the cost function value of partial solutions cannot be meaningfully compared to known full-length solutions (solutions that have traversed the entire depth of the tree) or even other partial-length solutions. Nevertheless, BB has been employed to explore the solutions spaces of challenging mission design problems such as the GTOC 6 problem, and is often the only feasible choice when the complexity of the problem becomes too great to effectively employ other outer-loop algorithms such as GAs. This technique for solving the MGA sequencing problem used in chapter 6 where it is applied to the planetary satellite tour problem.

Genetic algorithms have become a popular choice for solving the interplanetary MGA problem as first demonstrated by Gage et al. [83]. A GA was first incorporated into an MGA HOC solution framework by Conway et al. [93]. Genetic algorithms operate by first generating a population of random decision vectors, which are referred to as chromosomes. At each generation, the fitness of each of these chromosomes is determined by evaluating the problem's cost function. After evaluation, a selection operator is applied that selects the members of the population that will become the parents for the next generation. A crossover operator is then applied that creates the next population from the selected parents, replacing a certain portion of the overall population. Finally, with some small propability, random mutation occurs amongst the population. In order to guard the highest-valued members of the population from being discarded due to random mutation, a certain number of elite individuals are guaranteed to be preserved and carried over to the next generation. The genetic algorithm will proceed for a certain number of pre-specified generations, or until some user-specifed convergence criterion is satisfied.

The most useful class of genetic algorithm, as far as preliminary mission design is concerned, are the so-called multi-objective genetic algorithms. These routines are capable of mapping Pareto surfaces in a problem's design space as opposed to optimal point-solutions, and are of more use to mission design engineers who must react to constantly evolving spacecraft hardware designs and requirements. Recent advances by Vavrina et al. [100] and Englander et al. [101] employ the NSGA-II [102] and represent the current state-of-the-art in system-level spacecraft trajectory optimization research.

1.3.3 Inner-Loop Solver

The objective of the HOC inner-loop solver is to compute the locally optimal trajectory solution for a given configuration of the outer-loop optimization problem, i.e. it performs trajectory pathsolving. The specific problem formulation varies depending on the level of fidelity required as well as whether a chemical or electric propulsion system is being employed. Several trajectory models are considered in this work.

As with the outer-loop, there are also many methods available for solving the inner-loop problem. If the trajectory problem is formulated as a Hamiltonian boundary value problem, then primer vector techniques can be applied to obtain solutions that satisfy the necessary conditions of optimality. Metaheuristic techniques such as PSO and DE are attractive in that they are relatively straightforward to implement and do not require gradient information to perform their local search. The inner-loop solution technique that is the topic of discussion for much of this work is nonlinear programming, a gradient-based technique that will be introduced in section 2.1. Regardless of the strategy that is employed to solve the trajectory optimization problem, it is essential to the successful operation of the overall HOC solver that the inner-loop operate in a robust fashion.

1.3.4 Accurate and Efficient Execution of the Inner-Loop

The goal of the inner-loop solver in an HOC framework is to return the cost function value associated with a particular selection of outer-loop decision variables. If the inner-loop fails to return as solution (assuming one exists), or even if it returns a solution of poor quality (i.e. not locally optimal), then the effectiveness of the outer-loop will be compromised. For example, if a genetic algorithm is used as the outer-loop solver, and the inner-loop returns an inaccurate cost function value for a particular flyby sequence, then that sequence will be scored lower than it otherwise would have been. As a result of this, during the selection phase, that member of the population could be removed from the population pool prematurely, thus spoiling the overall integrity of the genetic pool.

In addition to robustness concerns, the second most important characteristic of a high-quality inner-loop solver is fast execution. The vast majority of the execution time associated with an HOC solver is attributed to the inner-loop. Whereas the outer-loop typically only needs to perform a few basic initialization operations during each of its iterations, the costliest typically being sorting routines, the inner-loop must evaluate the trajectory cost function. This routine typically includes the evaluation of the equations of motion, calls to hardware modeling routines, ephemerides and even costly numerical integration for higher fidelity models. In order to ensure that the local decision space is thoroughly explored, and the local minimum identified, many solution paradigms evaluate the inner-loop multiple times for each iteration of the outer-loop. For all of these reasons, inner-loop execution usually dominates the runtime of the overall optimization framework, and any improvements to the efficiency of its operation are important for reducing the time-to-solution.

One of the most significant applications of the contributions described in this work to a practical problem was the optimization of the Phase-A trajectory for the Lucy mission. Flight dynamics analysis tasks supporting this mission heavily rely on the inner loop of the optimization framework that is a focal point of this work. Lucy is a NASA Discovery program project that will perform flybys of one main belt asteroid and six Jupiter Trojan small bodies. It is scheduled to launch in October of 2021. It was selected in January 2017 to proceed to Phase-B study. The original design was performed by Brian Sutter and Chelsea Welch from Lockheed Martin, however, the final optimization was performed by Jacob Englander using EMTG. The optimization of the trajectory decreased the propellant cost function, reduced the small body encounter relative velocities by a significant amount and made the Polymele and Leucus encounters possible. The Lucy mission itinerary is summarized in Table 1.1 and the trajectory is shown in Fig. 1.4 and 1.5 [120].

Table 1.1: Lucy small body targets and nominal encounter dates.

Event/Encounter	Encounter Date
Launch	October 16 2021
DSM 1	April 19 2022
EGA 1	October 16 2022
DSM 2	February 2 2024
EGA 2	December 13 2024
52246 Donaldjohanson	April 20 2025
DSM 3	April 3 2027
3548 Eurybates	August 12 2027
15094 Polymele	September 15 2027
DSM 4	September 29 2027
11351 Leucus	April 18 2028
DSM 5	July 23 2028
21900 Orus	November 11 2028
EGA 3	December 26 2030
617 Patroclus + Menoetius	March 2 2033

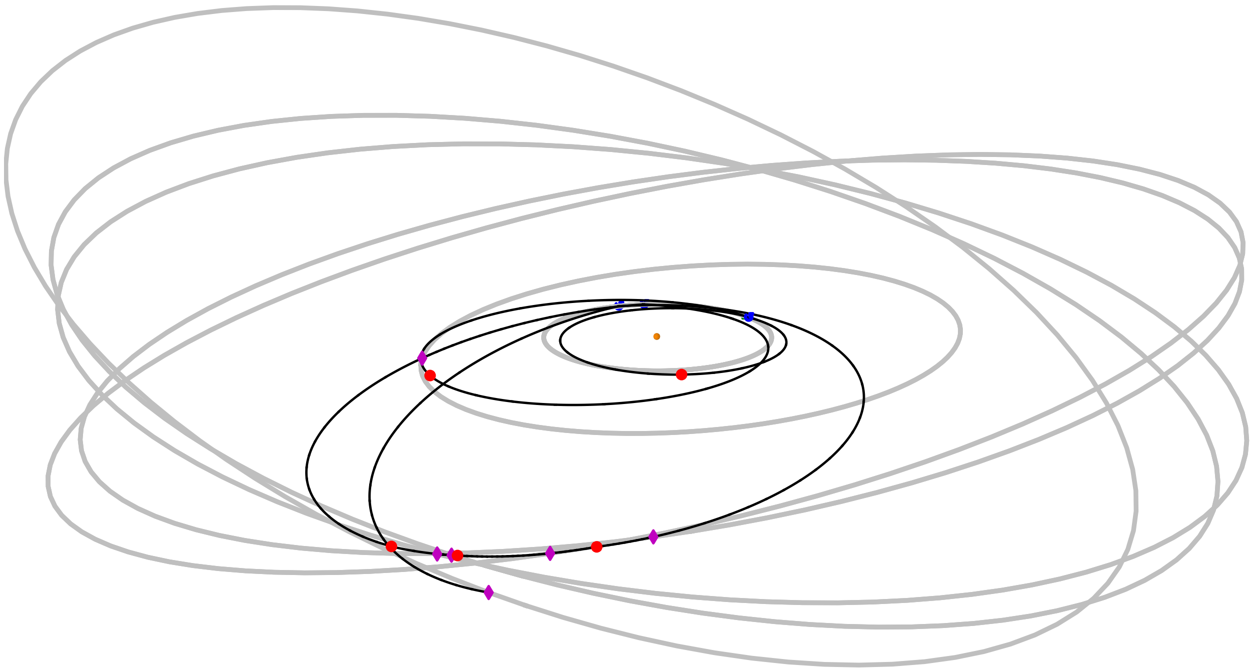


Figure 1.4: Lucy optimized trajectory showing the inclined orbits of the small body targets.

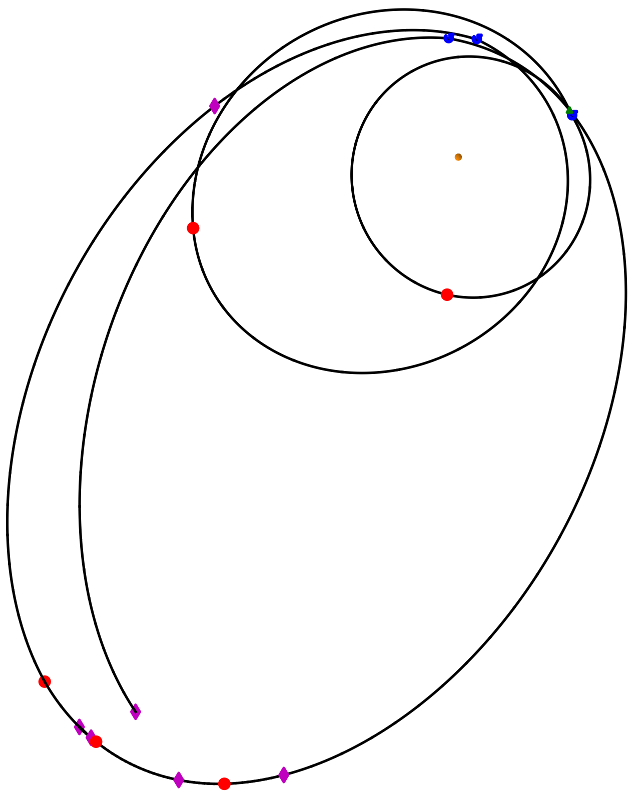


Figure 1.5: Lucy optimized trajectory. Red dots are bipropellant maneuvers, purple diamonds are small body intercepts and blue dots are Earth flybys.

The analytic derivative methods described in chapters 2 and 3, which significantly increase the robustness of the local optimizer, directly contributed to the improved Lucy solution. The development of the high-thrust optimization transcription described in chapter 3, as well as its analyic gradients, have also been directly applied to Lucy missed-maneuver analysis activities by the author and other members of the Lucy flight dynamics team.

1.3.5 Trajectory Design Heritage of Hybrid Optimal Control

Trajectory design and optimization using HOC is a relatively new practice. Despite this, HOC has already accrued design heritage on several mission proposals as well as a phase-B flight project. This was accomplished with the EMTG mission design software, which represents the cumulative efforts of several researchers including Dr. Jacob Englander, Mr. Matthew Vavrina, Dr. Alexander Ghosh, Mr. Ryne Beeson as well as the author.

To the author's knowledge, the first example of a HOC automaton being successfully applied to a mission proposal effort is Asteroid Robotic Redirect Mission (ARRM) [100]. This effort was particularly noteworthy as the use of the HOC method was instrumental in ARRM Option B being selected in favor of Option A before the entire project was ultimately canceled by the United States congress. Since the work on ARRM, EMTG has been used on several other proposals, including several in the New Frontiers 4 round of selection. The most successful of these was the design of the trajectory for the CAESAR mission, which was selected to advance to Phase A in December 2017.

1.4 Satellite Tour Design

Although the idea of a planetary satellite tour had been the subject of technical consideration since the early 1970s, the Galileo mission to Jupiter marked the first time that gravity assists were used to enable a scientific tour of a planetary satellite system. From December 1995 to September 2003, Galileo performed 33 flybys of Io, Europa, Ganymede and Callisto during its survey of the Jovian system. The Galileo tour design was a manual process where STOUR was employed to perform the initial ballistic patched-conic design [2]. This solution was then input to MOSES, which used the multiconic method to produce a trajectory that approximated the influence of other gravitating bodies, SRP etc [22], before finally being forward integrated in high-fidelity.

The Cassini spacecraft is currently completing the final week of its twenty year mission, most of which has been spent orbiting the planet Saturn. Cassini's tour of Saturn's moons is significantly more sophisticated than Galileo's. To respond to this, the Computer Algorithm for Trajectory Optimization (CATO) software was developed, which integrated the spacecraft's equations of motion subject to the point mass gravitational effects of other bodies, SRP and the oblateness of the central body [122, 123].

1.4.1 Flyby Combinatorics

The human lifespan, combined with a general unwillingness to wait large spans of time before receiving scientific returns from a mission, limits the number of gravity-assists that interplanetary spacecraft missions can reasonably use (to perhaps not greater than ten) [3]. By contrast, a planetary satellite tour can include hundreds of gravity-assists, as was the case with the Cassini mission where the orbiter performed 162 targeted moon flybys over the course of its 13 year exploration of the Saturnian system. As a result of this disparity,

[2]Prior to its automation by Steve Williams [121], the STOUR search process was entirely user-interactive

[3]The most number of GAs used by a spacecraft during its interplanetary cruise, to date, is six by the MESSENGER spacecraft.

many of the search techniques that have been shown to be effective for interplanetary MGA design encounter severe limitations when applied to satellite tour problems.

Heuristic methods such as genetic algorithms were first applied to the interplanetary MGA problem in order to help cope with the large dimension of the flyby pathfinding problem. Even for relatively modest problems where the chromosome length is perhaps only 10-15 bits in length, a genetic algorithm run can last days or weeks in order to ensure that the design space has been properly explored and that the population has maturely evolved. Tour problems represent a significantly more complex pathfinding problem, and some variation of the branch and bound method remains the only practical approach to contend with such a large design space in an acceptable amount of time. Of course, as discussed in section 1.2.3, the branch and bound method does not consider the entire search space and so any advances made in methods that more thoroughly explore the design space of extremely high dimensional pathfinding problems, such as planetary satellite tours, are valuable.

In order to illustrate the computational complexity associated with different MGA problems, first consider a mission from Earth to Jupiter, where we limit valid intermediate flyby targets to Earth, Venus or Mars. If the choice during each phase of the problem is to just flyby either Earth, Venus or Mars, and we let the spacecraft perform up to five flybys on its way to Jupiter, there exist 243 different flyby combinations of Earth, Venus and Mars (e.g E-EVEEM-J). For comparison, let's assume that the Galileo spacecraft had exactly four flyby options for each phase of its Jupiter tour: perform a flyby of either Io, Europa, Ganymede or Callisto. For its 33 flyby tour, there exists 7.3786976×10^{19} unique flyby combinations of the four Galilean moons.

1.4.2 Automation

Automation of the flyby sequence search for gravity-assist trajectories is a highly desirable capability, and is the subject of several ongoing research efforts. Regarding interplanetary trajectories, the work of Englander and Conway is arguably the most advanced. Certainly it has enjoyed the most success to date, with their methods being applied to produce the final optimized trajectory for the recently selected NASA Discovery class Lucy mission. That body of research has shown that a HOC automaton is capable of automating the interplanetary mission design process. These methods have not yet been extended to the design of planetary satellite tours.

Significant work has been performed that has enabled automated pathfinding and pathsolving of the satellite tour problem. The most conceptually straightforward methods are based on exhaustive grids. The

STOUR software was used during the preliminary design phase of the Galileo tour [22] albeit as a user-in-the-loop application. As previously mentioned, STOUR was eventually automated and served as one of the many inspirations for what is probably the most advanced pathsolving tools capable of preliminary design of a gas giant moon tour, Explore [9]. Explore is similar to STOUR in that it is an exhaustive grid-based MGA search method. In addition to patched Lambert C3 matching routines, Explore incorporates more sophisticated trajectory elements such as v-infinity leveraging maneuver (VILM)s [124] and $n\pi$ transfers [66] as well as advanced search logic such as Pareto pruning algorithms. Explore was employed to produce the third place solution for the GTOC 6 problem. That competition also saw other groundbreaking work in satellite tour design. Izzo et al. [74] developed a completely automated tree search algorithm that employed a parallel asynchronous island optimization algorithm.

Where the previously mentioned studies have focused on the design of the entire tour, there have also been studies that focused on the automation of specific aspects of a moon tour mission. The work of Lynam [125,126] and Didion and Lynam [127] was specifically concerned with exhaustive searches for double and triple flyby capture sequences at the beginning of a tour. These works produced an extensive library of precomputed capture sequences spanning several decades. Valid interplanetary trajectories were then identified for a particular capture solution via backwards propagation. These studies relied on analytic derivations of the required satellite phasing constraints and incoming interplanetary velocity asymptotes required to produce a feasible capture. While interesting, this body of work is not particularly conducive to a rapid preliminary design cadence.

The state-of-the-art for automated capture design was recently advanced by Scott et al. [128], who developed a strategy for designing a capture sequence that is subject to practical constraints that are placed on the interplanetary trajectory. That is to say, their method is flexible enough to design a capture trajectory based deviations from an already-established nominal interplanetary arrival asymptote. In that respect, their work represents a significant increase in utility from a practical preliminary mission design standpoint over Lynam's exhaustive solution database methods.

Automatic design of satellite capture and tour trajectories is one of the most challenging problems in modern astrodynamics, and remains an active research area. This work revisits the automated tour pathsolving problem in part where Lantukh's work on Explore left off, by investigating parallelization of the underlying flyby tree search algorithm as well as the transition from a C3 matching Lambert grid solution to a direct optimization transcription that is capable of locally optimizing an entire end-to-end moon tour problem.

1.4.3 Path Constraints and Hazard Avoidance

The development of an automated method for designing planetary satellite tours must be flexible enough to accommodate practical engineering considerations, such as environmental hazards. Formulating a useful tour solution is not as simple as stringing together as many flybys as possible given a certain quantity of propellant. The environment around the giant planets Jupiter and Saturn are hostile regions for spacecraft to operate in. The Jupiter-Io plasma torus is one of the most radioactively hazardous areas for a spacecraft to fly through. Extended periods of time spent in this region close to the planet will rapidly degrade all but the most radiation-hardened electronics, and practical tour design in the Jovian system must take this into account [129]. High relative-velocity collisions with the debris and dust that can be found in planetary ring structures can be catastrophic to spacecraft and present another navigational hazard when designing and flying trajectories in close proximity to the giant planets. A particularly interesting example of an environmental factor that will impact future mission design is the high-inclination of the Uranian system caused by the planet's 97 degree axial tilt relative to the ecliptic plane. Any spacecraft that will go into orbit around Uranus in the future will have to avoid the planet's rings and also contend with inclination reduction should one of its goals be close-proximity exploration of Uranus's moons [130].

In addition to natural phenomena that require the imposition of path constraints on a trajectory, other mission requirements also place constraints on a spacecraft performing a satellite tour. One of the most common of these is line-of-sight communications constraints. In order to communicate wih the ground, a spacecraft's high gain antenna must point towards Earth. Since most interplanetary spacecraft buses do not have articulating high-gain mounts, this means slewing the entire spacecraft, which then places restrictions on when maneuvers can occur. Perhaps even more consequential is whether a particular obtained tour solution is viable from a navigational perspective. The short encounter to encounter times during a tour can become a serious concern when assessing the risk associated with the navigation of the spacecraft. This is one of the major concerns with multiple satellite-aided captures at a gas giant, as studied by Lynam [126].

The automated methods described in chapter 6 do not directly address these concerns, however, the general automated tour solving framework described in that chapter does not preclude the future inclusion of environmental and operational constraints into the search process.

1.5 Summary of Contributions and Dissertation Structure

The contributions that will be described in the remaining chapters of this work impact multiple areas of mission design and trajectory optimization.

Chapter 2 describes the general direct method multiple shooting formulation that will be the subject of discussion for much of this dissertation. The forward-backward shooting (FBS) transcription is a generic trajectory optimization concept that may be specialized to both high and low-thrust spacecraft at varying levels of fidelity. The specific contributions described in this chapter are:

1. The description of a solar electric power model that is of sufficient accuracy such that it may be used for NASA Discovery and New Frontiers proposal efforts. A smoothing technique is discussed that makes the power model well-behaved when used inside a gradient-based optimizer.

2. A rapid-retrieval RAM-based ephemeris system is described that is compatible with a gradient-based optimizer and that represents an improvement over the state-of-the-art in that the smooth ephemeris interpolation and data storage processes that are presented are superior to similar existing ephemeris systems.

Chapter 3 provides a complete description of the algorithms necessary to compute analytic defect constraints for two bounded-impulse trajectory models.

1. The partial derivative mathematics required to analytically compute match-point defect constraints for two low-fidelity transcriptions are described in the form of linear mapping matrices.

2. The recursive sweep algorithm for computing the defect constraints using these matrices is described.

3. Results of an investigation of the robustness properties of a numerical optimizer using the analytic derivatives described in this chapter versus using finite-differencing are presented.

4. Gradient accuracy is verified using automatic differentiation and timing data relative to Jacobian calculation using finite differencing is provided.

The material in chapter 4 describes a new medium-fidelity low-thrust trajectory transcription, which is a time-regularized version of the well-known Sims-Flanagan transcription. This trajectory model corrects a shortcoming of the time-discretization employed by the original Sims-Flanagan transcription, namely the lack of control nodes located at the periapse of eccentric transfers. The contributions described are:

1. An algorithm that regularizes the Sims-Flanagan transcription and redistributes the control nodes

equally in eccentric anomaly.

2. An exact method for propagating a spacecraft's state using a modified universal Kepler propagator in conjunction with the new transcription.

3. The analytic calculations required to compute match-point derivatives for the new trajectory transcription.

4. A method for extending the new regularization algorithm to distribute control nodes with equal true anomaly spacing.

The FBLT transcription is a straightforward extension of the MGALT model, however, the techniques required to compute the match-point defect gradients for this problem is significantly different. Chapter 5 describes the improvements to the FBLT physics model, gradient computations and efforts made to verify this trajectory transcription against heritage navigation tools. These advancements include:

1. Extensions of the FBLT model to include modeling of n-body gravity, SRP and planetary oblateness perturbations.

2. A method for computing time-of-flight derivatives for time free trajectory optimization problems modeled with the FBLT transcription that employ fixed-step explicit numerical integration.

3. A description of the method used to verify the accuracy of the FBLT model against a heritage spacecraft navigation tool. This work was carried out in cooperation with engineers at KinetX Aerospace.

Chapter 6 discusses advancements made towards fully automating the planetary satellite tour design process. These include:

1. A tree data structure and a search algorithm, similar to the one developed by Lantukh for Explore, capable of performing Lambert C3 matching for the interplanetary and satellite tour MGA problems.

2. A process for parallelizing the Lambert tree search using OpenMP multithreading

3. The incorporation of a tree node object pool to increase the efficiency of the tree search program.

4. A demonstration of the ability of a direct method optimization framework to locally optimize an end-to-end launch, interplanetary cruise, and planetary satellite tour using an initial guess from the C3 matching tree search. This is done using the MGAnDSMs transcription described in chapter 3, and is enabled by the gradient computations described therein.

The final chapter summarizes the contributions described in this dissertation and outlines avenues for future research and plans for pursuing them.

Chapter 2

Trajectory Optimization Using Nonlinear Programming and Forward-Backward Shooting

Figure 2.1: NASA Pioneer at Jupiter. By Rick Guidice.

The various MGA trajectory optimization techniques described in this work are all examples of direct method trajectory transcriptions and rely on the concept of parameter optimization using nonlinear programming. This chapter will introduce the concept of an NLP and discuss one of the main challenges associated with achieving a robust problem formulation, namely the accurate specification of constraint and objective function gradients.

The family of trajectory transcriptions used throughout this work is described in section 2.2. A smooth solar electric spacecraft power model and a smooth ephemeris model are discussed, both of which are compatible for use with a gradient-based optimizer. The power model is suitable for preliminary mission design and meets the requirements set forth for NASA Discovery program proposal work [131].

2.1 Nonlinear Programming

An NLP is an optimization problem whereby the minimization (or maximization) of an objective function $J(\mathbf{x})$ is achieved by varying a set of real variables $\mathbf{x}$, subject to a set of equalities and inequalities, commonly referred to as constraints. Examples of numerical NLP solvers include the Interior Point OPTimizer (IPOPT) [132] or the Sparse Nonlinear OPTimizer (SNOPT) [133], which is the solver that was used for this work. These algorithms are designed to solve mathematical programming problems of the following form:

$$(NLP) = \begin{cases} \min\limits_{x} \quad f(x) \quad \mathbb{R}^n \mapsto \mathbb{R} \\ s.t. \;\; l_b \leq \begin{pmatrix} x \\ c(x) \end{pmatrix} \leq u_b \end{cases} \tag{2.1}$$

$$x \in \Omega \subseteq \mathbb{R}^n, \quad c \in \mathbb{R}^m \quad l_b, u_b \in \mathbb{R}^{n+m}$$

where $f(x)$ is a smooth scalar function, $c(x)$ is a vector of smooth nonlinear and linear constraint functions, $\{l_b, u_b\}$ are vectors of constant lower and upper bounds on x and c and Ω is the feasible region for (NLP). A feasible point $\mathbf{x}_f$ for the problem posed in Equation (2.1) is one that satisfies all of the problem constraints [118].

The core algorithm implemented in SNOPT is a sparse SQP solver, which like many NLP solution methods relies on the Karush-Kuhn-Tucker (KKT) first order necessary conditions [134, 135] to determine whether or not $\mathbf{x}_f$ is in fact the optimal solution $\mathbf{x}^*$ to the problem described in Eq. (2.1):

1. $\lambda_i^* \geq 0 \quad i = 1, 2, ..., m$ dual feasibility
2. $\lambda_i^* c_i(\mathbf{x}^*) = 0 \quad i = 1, 2, ..., m$ complementary slackness
3. $\nabla \mathbf{f}(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla c_i(\mathbf{x}^*) = 0$ stationarity

One will note that the stationarity necessary condition requires the evaluation of the gradients of the objective function **f** and the constraint vector **c** with respect to the problem decision vector **x**. The matrix containing the partial derivatives of the constraints with respect to the decision vector is known as the Jacobian. Degraded Jacobian accuracy as a result of using approximation techniques such as finite differences in lieu of specifying analytically obtained values for the dense entries of the Jacobian, can increase the number of iterations required to solve the problem or could even result in the solver encountering numerical difficulties resulting in more severe exit conditions (e.g. singular entries in or rank-deficiency of the Jacobian).

2.1.1 Jacobian Calculation Techniques

Nonlinear programming solvers typically include a mode of operation where the software can compute the Jacobian matrix via a finite differencing approximation. There are many possible finite difference algorithms. The forward and central differencing methods are shown in Eq. (2.2) and (2.3). These algorithms exhibit first and second order Taylor series truncation error respectively, however, higher-order accuracy methods are also possible [136].

$$\frac{\partial c_j(\mathbf{x})}{\partial x_i} = \frac{c_j(x_i + h) - c_j(x_i)}{h} + \mathcal{O}(h) \tag{2.2}$$

$$\frac{\partial c_j(\mathbf{x})}{\partial x_i} = \frac{c_j(x_i + h) - c_j(x_i - h)}{2h} + \mathcal{O}(h^2) \tag{2.3}$$

While divided differences are straightforward to implement, when the problem constraints are complex functions, the accuracy limits of this technique make its use inside an NLP solver undesirable for several reasons. First, finite differencing requires multiple calls to the NLP cost function in order to compute a single entry in the Jacobian. When the problem cost function is computationally expensive to evaluate, this can seriously hinder the progress of the optimizer from a pure runtime point-of-view. The more serious problem with employing divided differences is the inherent inaccuracy of the method. As Eq. (2.2) and (2.3) indicate, reducing the size of the perturbation step h will reduce the truncation error of the method. Doing so beyond a certain point, however, will begin to increase the floating point round-off error when the algorithm is implemented on a digital computer. For these reasons, finite-differencing is best avoided unless the use of other techniques is not possible, such as when using black-box software packages.

The concept of generating derivatives using complex numbers was introduced by Lyness and Moler [137, 138]. Perhaps the most advanced derivative computation technique in the category of complex step methods is the work by Lantoine et al. [139] and Pellegrini and Russell [136] with their introduction of multicomplex step differentiation. Multicomplex numbers are an extension of the complex numbers, and this method uses minute perturbations along the appropriate multicomplex direction to obtain partial derivatives of arbitrary order for any holomorphic function. While this technique has been shown to be both accurate and robust, its implementation is non-trivial and typically requires the development of auxiliary software that implements multicomplex mathematical floating point operations.

Still yet another class of methods for numerically computing partial derivatives is automatic differentiation. Automatic differentiation, also known as algorithmic differentiation or computational differentiation, encompasses a class of techniques that seeks to numerically compute the derivatives of a function expressed in a computer program at accuracies matching machine epsilon precision and can be realized using a wide variety of implementation strategies. In general, Automatic Differentiation (AD) techniques fundamentally rely on the fact that the chain rule can be successively applied to the mathematical expressions contained in a computer program in order to obtain derivatives of those expressions to arbitrary order. Automatic differentiation was first introduced in 1964, with the pioneering work on forward mode derivative accumulation by Wengert [140]. Reverse mode accumulation was introduced shortly after by Linnainmaa [141,142] and is the discrete analogue to continuous adjoints of differential equations. Automatic differentiation implementations can be roughly partitioned into these two main operational families, namely techniques that use forward mode derivative accumulation, and those that use reverse mode accumulation. Additional details regarding forward mode AD will be discussed in section 3.6.1.

Often the most computationally efficient means for computing the Jacobian is to provide analytic expressions for the dense entries. Depending on the complexity of the NLP problem being solved, this can be a tedious and error-prone process. Despite this, a large portion of the contributions described in this work relate to the analytic calculation of partial derivatives for trajectory transcriptions. Chapter 3 will describe algorithms for computing the Jacobian for two bounded-impulse trajectory models and chapter 4 will extend these methods to a time-regularized bounded-impulse model. Chapter 5 will discuss techniques for computing the Jacobian of a finite-burn trajectory model that employs numeric integration.

2.2 Forward-Backward Multiple Shooting

The MGA trajectory optimization problem is a multi-phase optimal control problem and a mission trajectory may be organized into N_p phases. In a forward-backward shooting phase, the spacecraft state is propagated from both phase boundaries inwards to an interior defect point in contrast to a single-shooting algorithm where the spacecraft state is propagated from one of the boundaries directly to the other. There are a number of advantages to using a multiple shooting transcription over one with a single forward shooter. The most obvious benefit is that the length of the shooting arcs are reduced, which leads to a reduction in numerical sensitivity and improved convergence. Moving the terminal manifold constraint to the mid-point of the phase prevents the optimizer from having to satisfy equality constraints with respect to an ephemeris-pegged object. Furthermore, from a practical standpoint, defects in the trajectory are easier to correct farther away from targeted flybys.

The boundaries of an FBS phase are called control points and can be massive bodies (flyby targets) such as planets, their satellites, asteroids, etc., or even a free point in space. The optimizer encodes the mass and the relative velocity vector of the spacecraft with respect to the control points ($\mathbf{v}_\infty$) at either side of the phase as decision variables. These decision parameters are combined with ephemeris data to form the complete spacecraft state at either side of a phase, which is then propagated forwards and backwards from the control points at the phase boundaries. This two-sided shooting, in general results in a discontinuity of the spacecraft's state vector ($\mathbf{X}$) at some location along the phase, which is closed to within some tolerance by the numerical optimizer as shown in Figure 2.2.

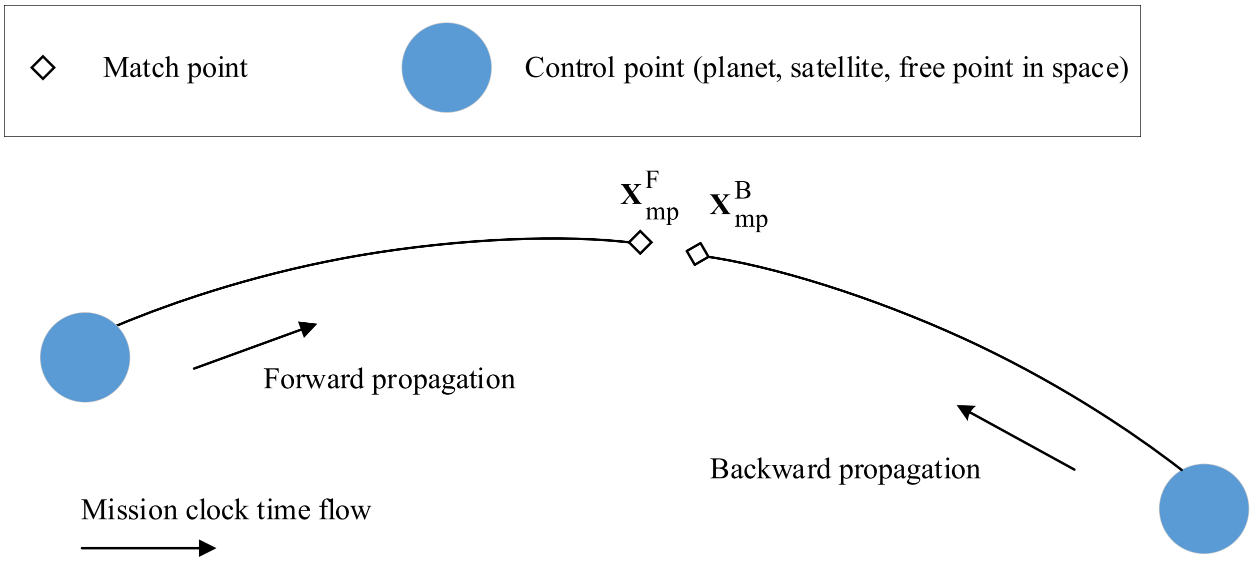


Figure 2.2: A single FBS phase.

The FBS formulation is popular, especially among interplanetary trajectory designers and there are several examples of spacecraft trajectory optimization tools and design infrastructures that make use of the FBS concept, including CATO [122,123], MALTO [15,49] and EMTG [47,48,90,113,143]

The hallmark constraint of an FBS phase is the *match point* continuity constraint. In order to enforce continuity of the state vector across the match point, a vector of "defect" constraints is applied:

$$\mathbf{c}_{\text{mp}} = \mathbf{X}^{\text{B}}_{\text{mp}} - \mathbf{X}^{\text{F}}_{\text{mp}} = \begin{bmatrix} \mathbf{r}^{\text{B}} - \mathbf{r}^{\text{F}} \\ \mathbf{v}^{\text{B}} - \mathbf{v}^{\text{F}} \\ m^{\text{B}} - m^{\text{F}} \end{bmatrix} = \mathbf{0} \tag{2.4}$$

For a multi-phase mission, the sum of the individual phase flight times (Δt_p) is constrained such that it fall within the upper and lower bounds (Δt_{min} and Δt_{max}) placed on the total mission time-of-flight Δt_{flight} by the trajectory analyst:

$$c_{\text{TOF}} = \Delta t_{\text{min}} \leq \Delta t_{\text{flight}} = \sum_{i=1}^{N_p} \Delta t_{p_i} \leq \Delta t_{\text{max}} \tag{2.5}$$

A list of typical decision variables that characterize an FBS mission are described in Table 2.1.

Table 2.1: Typical decision variables for an FBS mission.

x_i	Description	Count
t_{launch}	Launch epoch	1
v_∞	Launch impulse magnitude	1
RA	Right ascension of launch asymptote	1
DEC	Declination of launch asymptote	1
Δt_p	Phase time of flight	N_p
m_{f_p}	Phase final mass	N_p
$\mathbf{v}_{\infty_p}(t_0)$	Phase initial excess velocity vector	$3(N_p - 1)$
$\mathbf{v}_{\infty_p}(t_f)$	Phase final excess velocity vector	$3N_p$ [a]

[a] For a rendezvous this becomes $3(N_p - 1)$ because the rendezvous maneuver is deterministic.

2.3 The Patched-Conics Model

By their nature, preliminary trajectory design methods incorporate approximations that aid in accelerating the exploration of the design space. One of the most widely utilized approximation methods is the patched-conics model. This technique allows for the calculation of MGA trajectories using a two-body Keplerian propagation. In this framework, flyby maneuvers are modeled as instantaneous elastic collisions in the frame of the central body. The velocity of the spacecraft with respect to the central body immediately prior to the flyby is denoted by $\mathbf{v}^-_{\mathrm{s/c}}$. Then, interaction with the flyby target moving with velocity $\mathbf{v}_{\mathrm{planet}}$ causes an instantaneous change in the spacecraft velocity $\Delta\mathbf{v}_{\mathrm{flyby}}$. The magnitude of the velocity of the spacecraft immediately after the flyby $\mathbf{v}^+_{\mathrm{s/c}}$ may be larger or smaller than the velocity prior to the event, in the frame of the central body. The magnitude of the hyperbolic excess velocity vectors with respect to the GA target $\mathbf{v}^-_{\infty}, \mathbf{v}^+_{\infty}$, however, remain unchanged. The body frame conservation of energy, must be formulated as a constraint when the patched-conic model is employed by a numerical optimizer:

$$c_{v_\infty} = v^+_\infty - v^-_\infty = 0 \tag{2.6}$$

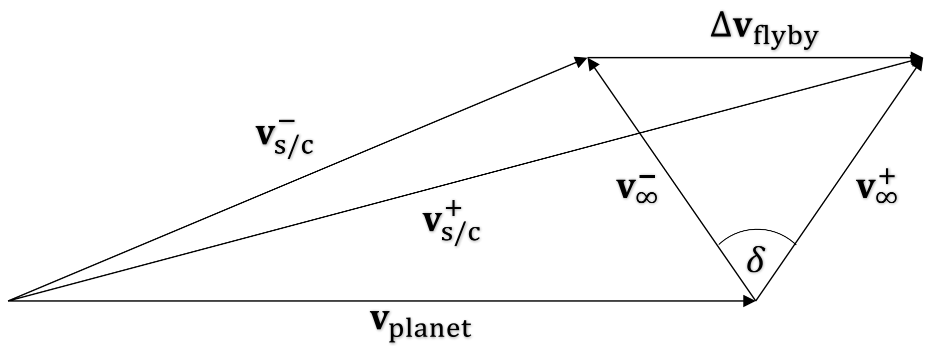
The flyby geometry must be further constrained such that the altitude of the spacecraft does not drop below a minimum safe altitude h_{safe} above the body's radius r_{body}, and that the turn angle of the flyby is physically realizable:

$$\begin{aligned} c_{\text{flyby-altitude}} &= r_{\mathrm{periapse}} - (r_{\mathrm{body}} + h_{\mathrm{safe}}) \geq 0 \qquad (2.7)\\ &= \frac{\mu_{\mathrm{body}}}{{v^+_\infty}^2}\left[\frac{1}{\sin(\delta/2)} - 1\right] - (r_{\mathrm{body}} + h_{\mathrm{safe}}) \geq 0 \end{aligned}$$

where δ is defined in Eq. (2.8).

$$\delta = \mathrm{acos}\left(\frac{\mathbf{v}^-_\infty \cdot \mathbf{v}^+_\infty}{v^-_\infty \;\; v^+_\infty}\right) \tag{2.8}$$

and r_{body} is the radius of the target flyby body.



$\Delta\mathbf{v}_{\text{flyby}}$
$\mathbf{v}^{-}_{s/c}$
$\mathbf{v}^{+}_{s/c}$
$\mathbf{v}^{-}_{\infty}$
$\mathbf{v}^{+}_{\infty}$
δ
$\mathbf{v}_{\text{planet}}$

Figure 2.3: Patched-conic model flyby vector geometry.

Gradients for Equations (2.6) and (2.7) with respect to the flyby velocity asymptote vectors are computed as follows:

$$\frac{\partial c_{\text{flyby-altitude}}}{\partial v^{-}_{\infty_x}} = -\frac{\mu_{\text{body}}\ \cos\left(\frac{\text{acos}(\alpha)}{2}\right)\left(v^{+}_{\infty_x}{v^{-}_{\infty_y}}^{2} - v^{-}_{\infty_x}v^{+}_{\infty_y}v^{-}_{\infty_y} + v^{+}_{\infty_x}{v^{-}_{\infty_z}}^{2} - v^{-}_{\infty_x}v^{+}_{\infty_z}v^{-}_{\infty_z}\right)}{r_{\text{periapse}}(\alpha-1)\left[1-\frac{(v^{-}_{\infty_x}v^{+}_{\infty_x}+v^{-}_{\infty_y}v^{+}_{\infty_y}+v^{-}_{\infty_z}v^{+}_{\infty_z})^2}{\gamma\beta}\right]^{1/2}\gamma^{3/2}\beta^{3/2}} \tag{2.9}$$

$$\frac{\partial c_{\text{flyby-altitude}}}{\partial v^{-}_{\infty_y}} = -\frac{\mu_{\text{body}}\ \cos\left(\frac{\text{acos}(\alpha)}{2}\right)\left(v^{+}_{\infty_y}{v^{-}_{\infty_x}}^{2} - v^{-}_{\infty_y}v^{+}_{\infty_x}v^{-}_{\infty_x} + v^{+}_{\infty_y}{v^{-}_{\infty_z}}^{2} - v^{-}_{\infty_y}v^{+}_{\infty_z}v^{-}_{\infty_z}\right)}{r_{\text{periapse}}(\alpha-1)\left[1-\frac{(v^{-}_{\infty_x}v^{+}_{\infty_x}+v^{-}_{\infty_y}v^{+}_{\infty_y}+v^{-}_{\infty_z}v^{+}_{\infty_z})^2}{\gamma\beta}\right]^{1/2}\gamma^{3/2}\beta^{3/2}} \tag{2.10}$$

$$\frac{\partial c_{\text{flyby-altitude}}}{\partial v^{-}_{\infty_z}} = -\frac{\mu_{\text{body}}\ \cos\left(\frac{\text{acos}(\alpha)}{2}\right)\left(v^{+}_{\infty_z}{v^{-}_{\infty_x}}^{2} - v^{-}_{\infty_z}v^{+}_{\infty_x}v^{-}_{\infty_x} + v^{+}_{\infty_z}{v^{-}_{\infty_y}}^{2} - v^{-}_{\infty_z}v^{+}_{\infty_y}v^{-}_{\infty_y}\right)}{r_{\text{periapse}}(\alpha-1)\left[1-\frac{(v^{-}_{\infty_x}v^{+}_{\infty_x}+v^{-}_{\infty_y}v^{+}_{\infty_y}+v^{-}_{\infty_z}v^{+}_{\infty_z})^2}{\gamma\beta}\right]^{1/2}\gamma^{3/2}\beta^{3/2}} \tag{2.11}$$

where

$$
\begin{aligned}
\alpha &= \frac{v_{\infty_x}^{-} v_{\infty_x}^{+} + v_{\infty_y}^{-} v_{\infty_y}^{+} + v_{\infty_z}^{-} v_{\infty_z}^{+}}{\beta^{1/2}\gamma^{1/2}} \\
\beta &= {v_{\infty_x}^{+}}^{2} + {v_{\infty_y}^{+}}^{2} + {v_{\infty_z}^{+}}^{2} \\
\gamma &= {v_{\infty_x}^{-}}^{2} + {v_{\infty_y}^{-}}^{2} + {v_{\infty_z}^{-}}^{2}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial c_{\text{flyby-altitude}}}{\partial v_{\infty_x}^{+}} &= \frac{2v_{\infty_x}^{+}\ \mu_{\text{body}}}{r_{\text{periapse}}\xi^{2}} - \frac{2v_{\infty_x}^{+}\ \mu_{\text{body}}}{r_{\text{periapse}}\ \sin\left(\frac{\text{acos}(\epsilon)}{2}\right)\xi^{2}} \\
&- \frac{\mu_{\text{body}}\ \cos\left(\frac{\text{acos}(\epsilon)}{2}\right)\left(v_{\infty_x}^{-}{v_{\infty_y}^{+}}^{2} - v_{\infty_y}^{-}v_{\infty_x}^{+}v_{\infty_y}^{+} + v_{\infty_x}^{-}{v_{\infty_z}^{+}}^{2} - v_{\infty_z}^{-}v_{\infty_x}^{+}v_{\infty_z}^{+}\right)}{r_{\text{flyby}}(\epsilon-1)\left[1-\frac{\phi^{2}}{\psi\xi}\right]^{1/2}\psi^{1/2}\xi^{5/2}}
\end{aligned}
\tag{2.12}
$$

$$
\begin{aligned}
\frac{\partial c_{\text{flyby-altitude}}}{\partial v_{\infty_y}^{+}} &= \frac{2v_{\infty_y}^{+}\ \mu_{\text{body}}}{r_{\text{periapse}}\xi^{2}} - \frac{2v_{\infty_y}^{+}\ \mu_{\text{body}}}{r_{\text{periapse}}\ \sin\left(\frac{\text{acos}(\epsilon)}{2}\right)\xi^{2}} \\
&- \frac{\mu_{\text{body}}\ \cos\left(\frac{\text{acos}(\epsilon)}{2}\right)\left(v_{\infty_y}^{-}{v_{\infty_x}^{+}}^{2} - v_{\infty_x}^{-}v_{\infty_y}^{+}v_{\infty_x}^{+} + v_{\infty_y}^{-}{v_{\infty_z}^{+}}^{2} - v_{\infty_z}^{-}v_{\infty_y}^{+}v_{\infty_z}^{+}\right)}{r_{\text{flyby}}(\epsilon-1)\left[1-\frac{\phi^{2}}{\psi\xi}\right]^{1/2}\psi^{1/2}\xi^{5/2}}
\end{aligned}
\tag{2.13}
$$

$$
\begin{aligned}
\frac{\partial c_{\text{flyby-altitude}}}{\partial v_{\infty_z}^{+}} &= \frac{2v_{\infty_z}^{+}\ \mu_{\text{body}}}{r_{\text{periapse}}\xi^{2}} - \frac{2v_{\infty_z}^{+}\ \mu_{\text{body}}}{r_{\text{periapse}}\ \sin\left(\frac{\text{acos}(\epsilon)}{2}\right)\xi^{2}} \\
&- \frac{\mu_{\text{body}}\ \cos\left(\frac{\text{acos}(\epsilon)}{2}\right)\left(v_{\infty_z}^{-}{v_{\infty_x}^{+}}^{2} - v_{\infty_x}^{-}v_{\infty_z}^{+}v_{\infty_x}^{+} + v_{\infty_z}^{-}{v_{\infty_y}^{+}}^{2} - v_{\infty_y}^{-}v_{\infty_z}^{+}v_{\infty_y}^{+}\right)}{r_{\text{flyby}}(\epsilon-1)\left[1-\frac{\phi^{2}}{\psi\xi}\right]^{1/2}\psi^{1/2}\xi^{5/2}}
\end{aligned}
\tag{2.14}
$$

where

$$\begin{aligned}
\epsilon &= \frac{\phi}{\psi^{1/2}\xi^{1/2}} \\
\xi &= {v_{\infty_x}^{+}}^2 + {v_{\infty_y}^{+}}^2 + {v_{\infty_z}^{+}}^2 \\
\phi &= v_{\infty_x}^{-} v_{\infty_x}^{+} + v_{\infty_y}^{-} v_{\infty_y}^{+} + v_{\infty_z}^{-} v_{\infty_z}^{+} \\
\psi &= {v_{\infty_x}^{-}}^2 + {v_{\infty_y}^{-}}^2 + {v_{\infty_z}^{-}}^2
\end{aligned}$$

2.4 Spacecraft Power System Modeling

The solar electric power model introduced in this section is compatible with gradient-based optimizers that require differentiability of the problem constraints and objective function. For a spacecraft using a solar electric propulsion (SEP) system, the maximum amount of thrust that the system can produce depends on the power available, which in turn depends on its distance from the Sun. The relationship between power generated and thrust produced must be modeled in the trajectory optimization process even at the preliminary design stage. The trajectory design analyses presented in this work incorporate models of real thruster hardware. Typically, trajectory design engineers are supplied with polynomial representations of the thrust and mass flow rate of a given thruster, *e.g.*,

$$T_{\max} = e_T P_{\text{eff}}^4 + d_T P_{\text{eff}}^3 + c_T P_{\text{eff}}^2 + b_T P_{\text{eff}} + a_T \tag{2.15}$$

$$\dot{m}_{\max} = e_m P_{\text{eff}}^4 + d_m P_{\text{eff}}^3 + c_m P_{\text{eff}}^2 + b_m P_{\text{eff}} + a_m \tag{2.16}$$

where P_{eff} is the power available to each thruster,

$$P_{\text{eff}} = P / N_{\text{active}} \tag{2.17}$$

and N_{active} is the number of thrusters firing at any point in time. The polynomial coefficients used in Eq. (2.15) and (2.16) are provided in Table A.1 in Appendix A.2.

Equations (2.15) and (2.16) are valid over a range $[P_{\min}, P_{\max}]$ where $P_{\min}$ represents the minimum amount of power necessary to turn on the thruster's power processing unit (PPU) at the lowest setting and $P_{\max}$ represents the maximum amount of power that the PPU can safely accommodate. The total power available to the electric propulsion system P is the difference between the power generated by the spacecraft

$P_{\text{generated}}$ and the power required to operate the spacecraft bus $P_{\text{s/c}}$,

$$P = (1 - \delta_{\text{power}}) \left(P_{\text{generated}} - P_{\text{s/c}}\right) \tag{2.18}$$

where δ_{power} is the propulsion power margin. In this work, the power delivered by a solar array is given by [51]:

$$P_{\text{generated}} = \frac{P_0}{r_{s/\odot}^2} \left(\frac{\gamma_0 + \gamma_1/r_{s/\odot} + \gamma_2/r_{s/\odot}^2}{1 + \gamma_3 r_{s/\odot} + \gamma_4 r_{s/\odot}^2} \right) \tag{2.19}$$

where the γ_i are solar array coefficients typically determined from experimental data provided by the manufacturer, $r_{s/\odot}$ is the distance between the Sun and the spacecraft in Astronomical Unit (AU) and P_0 is the "base power" delivered by the array at 1 AU. P_0 is in turn a function of the time since launch,

$$P_0 = P_{0\text{-BOL}} \left(1 - \tau\right)^t \tag{2.20}$$

where $P_{0\text{-BOL}}$ is the base power delivered by the array at 1 AU on the day of launch, τ is the decay rate of the solar arrays measured as a percentage per year, and t is the time since launch in years. Equation (2.20) may also be used to model the decay of a radioisotope thermal generator (RTG) or advanced Stirling radiosotope generator (ASRG) power system. The power required by the spacecraft bus $P_{\text{s/c}}$ is also modeled as a polynomial,

$$P_{\text{s/c}} = a_{\text{s/c}} + b_{\text{s/c}}/r_{s/\odot} + c_{\text{s/c}}/r_{s/\odot}^2 \tag{2.21}$$

where $a_{\text{s/c}}$, $b_{\text{s/c}}$, and $c_{\text{s/c}}$ are specified by the mission designer.

The most interesting case is when $P_{\text{eff}} > P_{\text{max}}$ or $P_{\text{eff}} < P_{\text{min}}$, and therefore thrusters must be switched on or off. When $P_{\text{eff}} > P_{\text{max}}$ then either an additional thruster must be switched on, or if no other thrusters are available, P_{eff} must be clipped to P_{max}. Finally, when $P_{\text{eff}} < P_{\text{min}}$ then a thruster must be switched off. A discontinuity exists in Equations (2.15) and (2.16) at the boundaries where $P_{\text{eff}} = P_{\text{max}}$ or $P_{\text{eff}} = P_{\text{min}}$. A discontinuity such as this results in the gradients of Equations (2.15) and (2.16) to become undefined, which will negatively impact the performance of gradient-based optimizers, especially in the case where there is not enough power to turn on any thrusters at all. It is desirable to smooth the power and propulsion models and remove the discontinuity. McConaghy [144] proposed smoothing the propulsion model using the "smoothstep" technique from the field of computer graphics. A different approach has been developed for this work.

Heaviside [145] defined the unit step function as instantaneously taking half value at the point of transition. It is then possible to approximate the step function using the logistics function,

$$H(x) = \lim_{\alpha\to\infty} \frac{1}{1+\exp(-2\alpha x)}. \tag{2.22}$$

Equation (2.22) is continuously differentiable and therefore eliminates the problems that a gradient-based solver would have with a step function. In the context of multi-thruster switching, a set of Heaviside step functions $\{H_i(P)\}$ is introduced,

$$H_i(P) = \frac{1}{1+\exp(-2\alpha(P-P_i^*))} \tag{2.23}$$

where each Heaviside step function $H_i(P)$ defines the switch state of the i^{th} thruster and α defines the sharpness of the transition. The larger the value of α, the closer $H_i(P)$ approximates the Heaviside step function. However, while the derivatives $H_i'(P)$ increase as α increases, they remain finite and $H_i(P)$ remains continuous. Numerical experimentation indicates that the optimizer behaves best when the derivatives are reasonably small (*i.e.* small α), however, if α is too small then the thruster model will frequently report a fractional non-integer $H_i(P)$. N_{active} may then be defined as,

$$N_{\text{active}} = \sum_{i=1}^{N} H_i(P) \tag{2.24}$$

To fully define N_{active}, it is necessary to define the transition powers P_i^* at which a thruster would be switched on and off. For propulsion modules featuring multiple thrusters, several different thruster logic programs may be followed. When as few thrusters as possible are activated for a given available power each P_i^* is an integer multiple of $P_{\max}$ except for P_1, which is equal to $P_{\min}$. Alternatively if as many thrusters as possible are activated, then each P_i^* would be an integer multiple of $P_{\min}$. It is also possible to define other switching laws such as maximum thrust or maximum total specific impulse, or even maximize the propulsion system efficiency for a given power level, in which case one would need to compute the transition P_i^* where those merit functions change as a function of the number of thrusters. Two of these thruster logic programs are provided in Algorithms 1 and 2:

Figure 2.4 shows the available thrust for a notional system with four XR-5 (BPT-4000) thrusters, each with $P_{\min} = 0.302$ kW. The plot is focused on the region between 2 and 4 AU where the thruster transitions occur. The spacecraft's solar array can provide 10 kW at 1 AU and the spacecraft bus requires 0.5 kW at all times. Several values of α are shown, each a different compromise between smoothness and accuracy. One

Algorithm 1 Multi-Thruster Logic:
Maximum Number of Thrusters

```
$n_{\text{active}} = n_{\text{available}}$
**if** $P \geq P_{\min}$ **then**
    **for** $n = n_{\text{available}}; n > 0; --n$ **do**
        **if** $P \geq nP_{\min}$ **then**
            $n_{\text{active}} = n$
            break
        **end if**
    **end for**
**else**
    $n_{\text{active}} = 0$
**end if**
```

Algorithm 2 Multi-Thruster Logic:
Minimum Number of Thrusters

```
$n_{\text{active}} = n_{\text{available}}$
**if** $P \geq P_{\min}$ **then**
    **for** $n = n_{\text{available}}; n > 0; --n$ **do**
        **if** $P \leq nP_{\max}$ **then**
            $n_{\text{active}} = n$
            break
        **end if**
    **end for**
**else**
    $n_{\text{active}} = 0$
**end if**
```

can see that the line marked with + symbols, representing $\alpha = 1000$, closely approximates the unsmoothed solid red line that does not have continuous derivatives. The smoothing method described here removes these discontinuities and improves the robustness of the solver. A value of $\alpha = 100$ is recommended as a good compromise between well-behaved derivatives and accuracy.

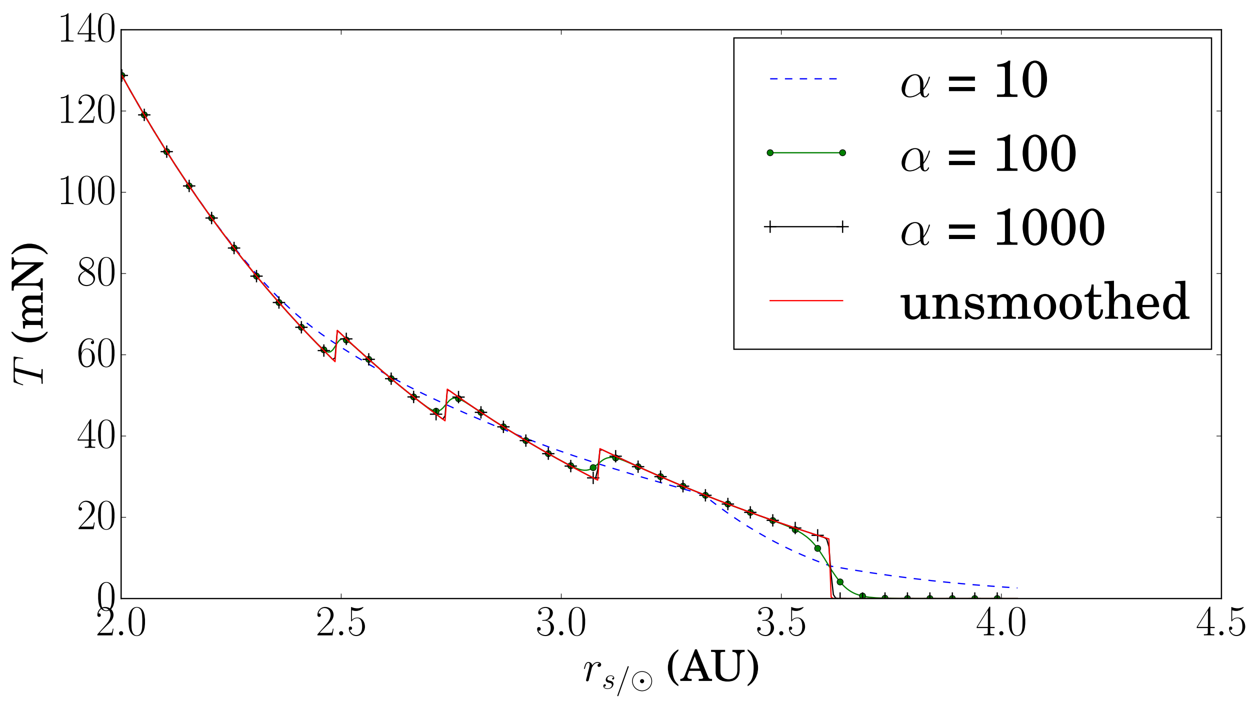


Figure 2.4: Thrust vs. distance from sun for various values of α. The propulsion system consists of 4xBPT-4000 thrusters with $P_{BOL} = 10$ kW.

2.5 Partial Derivatives of the Ephemeris

In most trajectory design applications, ephemeris data for solar system bodies is provided using the Spacecraft Planet Instrument C-matrix Events (SPICE) ephemeris toolkit [146]. SPICE provides a compact, high-precision approximation to the actual integrated ephemerides by using Chebychev polynomial fits and is considered industry standard for astronomers and planetary scientists, and also for most trajectory design work. However, SPICE suffers from several major limitations that make it less than ideal for optimization applications. First, in the interest of maintaining a small memory footprint, SPICE reads ephemeris data directly from the hard drive for each query and is therefore quite slow. While this was a requirement of software in the past when computer memory was significantly more limited, it is also a major execution speed handicap for modern trajectory optimization frameworks. Second, and of particular importance to this work, SPICE does not provide derivatives of the Chebychev functions used to model the body states, which are required to compute time derivatives of the ephemeris. Furthermore, in the interest of maintaining accuracy, each of the six states (x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$) is fit to a separate polynomial for most SPICE ephemeris types. A body's velocity in SPICE is therefore not guaranteed to be the derivative of its position. Finally, the Chebychev state vector fits are performed over fairly short intervals of time. The polynomial fitting functions are C^0 continuous at the boundaries of these intervals, and are, therefore, not differentiable at these points. As a result of these limitations, SPICE is not the ideal ephemeris package for trajectory optimization because it does not work well with an analytic derivative formulation.

Several attempts were made to circumvent the issue of a non-smooth ephemeris. The first approach was to simply compute all derivatives with respect to time variables (e.g. launch epoch and phase flight times) using finite differencing, however, this was slow and resulted in poor convergence properties. The second approach was to finite difference just the ephemeris itself and then chain the resulting boundary state derivative with the analytical derivatives derived in a previous work [143]. The second approach was much faster but delivered a poor approximation to the actual derivatives and also resulted in poor convergence. The third approach was to fit a cubic spline to the SPICE ephemeris data and cache the spline coefficients in the computer's random access memory (RAM), a strategy first proposed by Arora and Russell [147]. The approach used in this work is similar, except that where Arora and Russell created a complex ephemeris archives and saved them to the hard drive such that they could be reused, the package created for this work (SplineEphem) performs the spline fitting at the start of the optimization program's execution. The resulting ephemeris system is simpler to use at the cost of a few seconds of load time prior to each optimization run.

2.5.1 SplineEphem

The creation of a SplineEphem fit for a given solar system body consists of two steps. First, SPICE is used to generate the set of ephemeris points over which the spline will be fit. The spline fits themselves are then computed using the GNU Scientific Library (GSL) spline package [148]. Figures 2.5 and 2.6 show the error in position and velocity for a SplineEphem fit to Jupiter over 1000 days, with 100 ephemeris points per orbital period of Jupiter. The maximum position error in this example is 414 km and the maximum velocity error is 4.3 m/s, which is more than sufficient for the preliminary trajectory design studies presented in this work. Should better accuracy be required for high-fidelity planning, the number of sample points may be increased at the cost of a larger RAM footprint.

In addition to providing analytic derivatives that themselves lead to improved speed and convergence, SplineEphem is much faster than SPICE. SplineEphem's access speed was tested using 1e+7 ephemeris access events. When the accesses are sequential in time then SplineEphem is 12 times faster than SPICE, and if the sample epochs are randomized, then SplineEphem is 82 times faster.

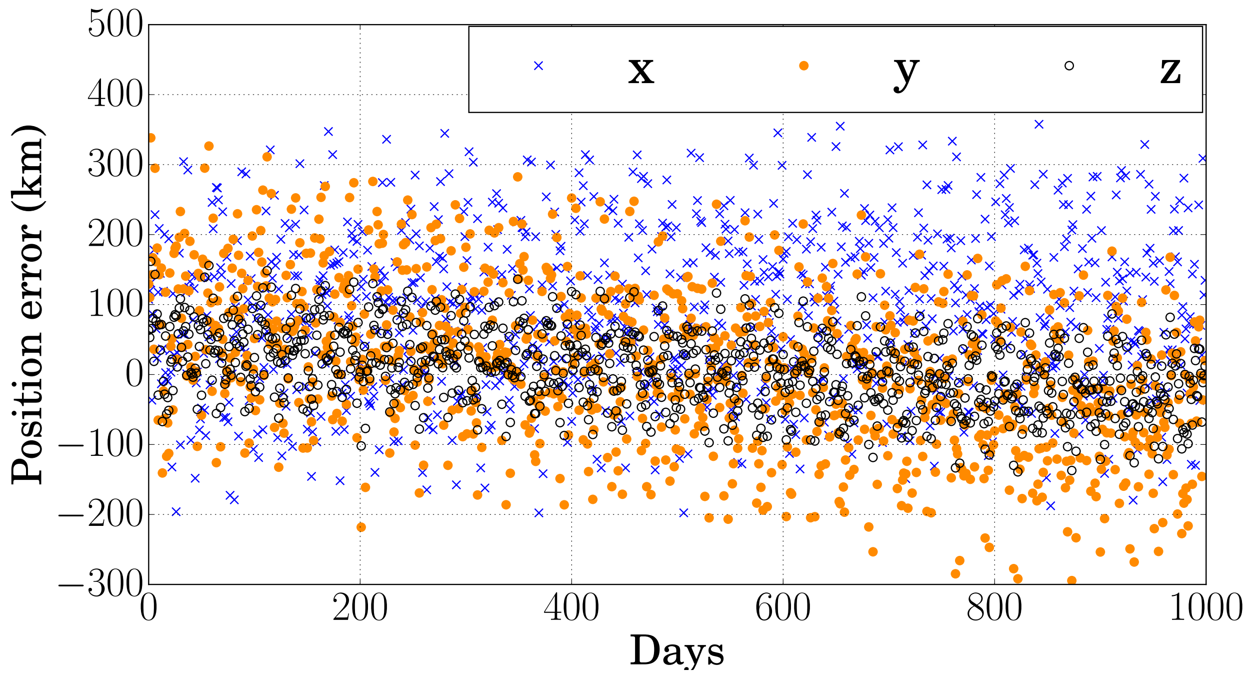


Figure 2.5: SplineEphem error relative to SPICE for the position of Jupiter relative to the Sun, over a 1000-day period.

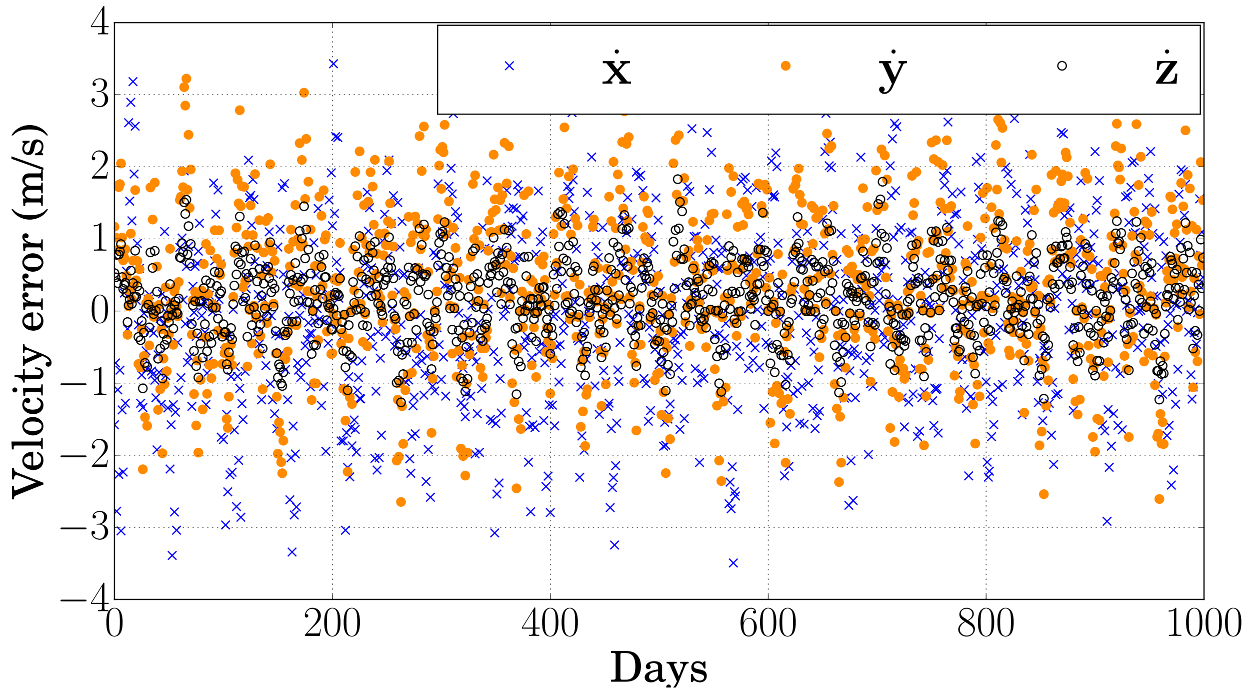


Figure 2.6: SplineEphem error relative to SPICE for the velocity of Jupiter relative to the Sun, over a 1000-day period.

Chapter 3

Analytic Gradient Computation for Bounded-Impulse Trajectory Models

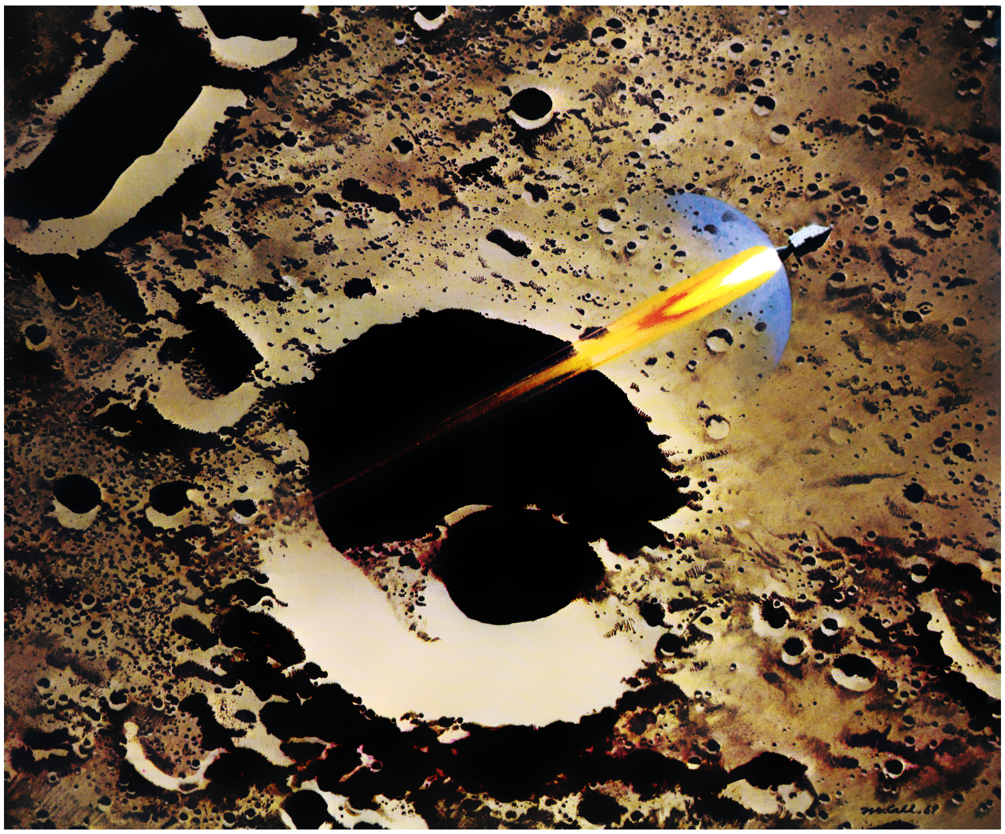


Figure 3.1: Apollo 8 Coming Home by Robert McCall.

The modeling of spacecraft maneuvers as impulsive events is a common preliminary design technique. The impulsive approximation treats a finite-burn maneuver as an instantaneous change in a spacecraft's velocity vector (i.e. an elastic collision). For nearly all preliminary maneuver design activities, this approximation is acceptable, and the additional propellant cost of modeling a burn of finite duration is of little importance to early stage design.

Traditionally, most impulsive analysis was performed in the context of determining a Δv budget for a particular trajectory. That is to say, the impulsive maneuver could be of infinite magnitude, and whether or not a particular required maneuver Δv was achievable or not from a hardware/propellant perspective was only considered *a posteriori*. For the preliminary design analysis, and corresponding trajectory transcription techniques, described in this work, all impulsive maneuvering is performed assuming a bounded-impulse model. That is to say, mass is propagated alongside position and velocity, and unachievable maneuvers from a propellant consumption perspective are discarded during the optimization process.

The practice of incorporating impulsive maneuvers into preliminary design is not limited to high-thrust regimes. The pioneering work by Sims and Flanagan [15] introduced a method for approximating low-thrust trajectory arcs using a series of bounded-impulses that laid the groundwork for how most preliminary low-thrust mission design has been carried out since that paper was published. In particular, low-fidelity low-thrust interplanetary mission design at NASA GSFC, JPL and JHUAPL is all based on this concept.

The bounded-impulse approximation for spacecraft trajectories, in both high and low-thrust regimes, has an important role in many preliminary trajectory design frameworks [15,46,49,111,149]. Mission design efforts have benefited greatly from the execution speed of software implementations of these models, and they are valued for their ability to rapidly generate preliminary design quality solutions capable of steering trade studies and proposal efforts [100, 101]. Efficiency is important when exploring problems with large design spaces, therefore, any measures that can be taken to improve a preliminary design method's execution speed should be taken.

The primary advantage of computing maneuvers impulsively is that Keplerian propagation of the spacecraft is possible at all times if two-body motion is assumed. That is to say, the spacecraft's state can be advanced according to the following matrix equation:

$$\begin{bmatrix} \mathbf{r}_{k+1} \\ \mathbf{v}_{k+1} \end{bmatrix} = \begin{bmatrix} F & G \\ \dot{F} & \dot{G} \end{bmatrix} \begin{bmatrix} \mathbf{r}_{k} \\ \mathbf{v}_{k} \end{bmatrix}, \tag{3.1}$$

where expressions for $F, \dot{F}, G$ and $\dot{G}$ are provided in Eq. (3.18), (3.19), (3.21) and (3.22). When modeling a true finite-burn, even if only central body gravity is being considered, numerical integration of a thrust term in the differential equations of motion is required.

The accurate and efficient computation of constraint gradients has been shown to be of critical importance to a preliminary design optimization framework [73,150,151]. Multi-objective HOC solution frameworks that seek to generate Pareto surfaces [100,101] in particular are enabled by the fast computation of accurate NLP gradients.

The multiple gravity-assist low-thrust (MGALT) [47] and multiple gravity-assist with n deep space maneuvers using shooting (MGAnDSMs) [152] are two examples of bounded-impulse trajectory models and are described in the next two sections. The remainder of the Chapter is then dedicated to analytic techniques for computing the Jacobian matrices for both of these transcriptions. The methods presented allow for the incorporation of an accurate solar-electric power model [131] into the trajectory model and are applicable to trajectories around any central body.

Despite the utility of providing analytic gradient information to an NLP optimizer, an algorithm specification for computing partial derivatives of one of the most important trajectory models to preliminary mission design, the Sims-Flanagan transcription, has been notably absent from the literature to date. While several variations of the Sims-Flanagan transcription exist, due to improvements made to the algorithm since it was introduced, the version presented here has been successfully incorporated into an operational preliminary design framework [47] that has supported several Discovery and New Frontiers class mission concept studies and implementations, and is recommended for the general practitioner. Particular emphasis is placed on the computation of the Sims-Flanagan phase defect constraint gradients, which are characterized by complex functional dependencies.

Application problems that underscore the benefits gained from utilizing an analytically computed Jacobian in the optimizer are presented at the end of this chapter. Several of these examples also investigate the effect that using these analytic techniques has on a solver scheme that incorporates both local gradient searches using an NLP solver as well as a stochastic search component realized with the monotonic basin hopping (MBH) heuristic [48, 153–155]. The algorithms described here have been verified for accuracy by using an automatic differentiation package that was developed by Ghosh [156].

3.1 Multiple Gravity-Assist Low-Thrust Trajectory Model

The multiple gravity assist with low-thrust (MGALT) transcription is a simplified model that combines the well-known Sims-Flanagan transcription [15] with a two-body patched-conic flyby model [157]. Each phase of a Sims-Flanagan trajectory is itself a multistage optimal control problem, and is discretized into N equal time segments. Fig. 3.2 depicts a single phase and shows the match point discontinuity.

The continuous thrust that may be applied over the course of the k^{th} segment in a phase is approximated by a bounded-impulse at the center of the segment whose maximum magnitude is equal to the change in velocity that would have been provided had the spacecraft's thruster been operated continuously at its

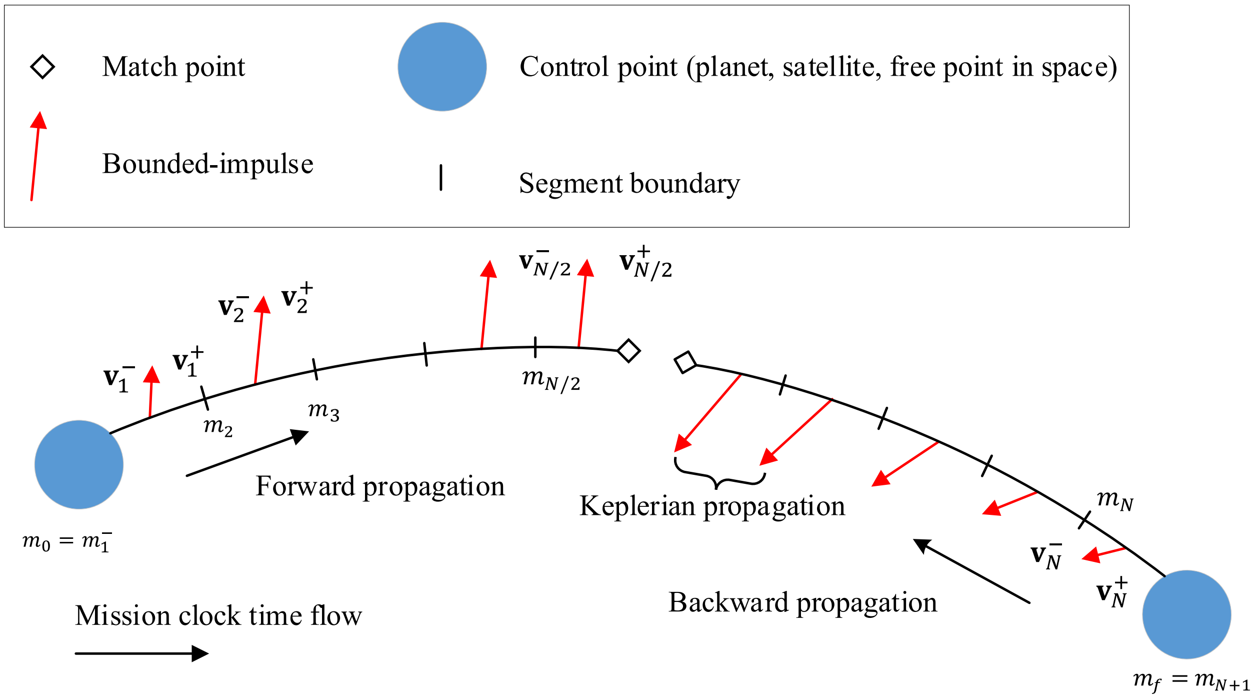


Figure 3.2: A single MGALT phase. The velocity symbols $\mathbf{v}_k^-$ and $\mathbf{v}_k^+$ are present to indicate the velocity of the spacecraft immediately prior to and immediately after the kth impulsive maneuver.

maximum throttle setting for the full duration of the segment:

$$\Delta v_{\max_k} = \frac{N_{\text{active}} D \; T_{\max_k} \left(t_f - t_0\right)}{m_k N} \tag{3.2}$$

In Eq. (3.2), N_{active} is the number of active thrusters, D is the thruster duty cycle, $T_{\max_k}$ is the maximum available thrust for the current maneuver, t_0 and t_f are the beginning and ending epochs of the current phase and m_k is the mass of the spacecraft at the center of the segment, just prior to the applied impulse.

Since the applied thrust is approximated as a discontinuous $\Delta\mathbf{v}$ vector, it is possible to propagate the spacecraft's position and velocity components using Kepler's equation between applied impulses. Furthermore, due to this discontinuity, the following notation is adopted to distinguish between the spacecraft's velocity just before, and immediately following, the applied impulse, i.e.

$$\mathbf{v}_k^+ = \mathbf{v}_k^- + \Delta v_{\max_k} \mathbf{u}_k \tag{3.3}$$

for forward propagation, and

$$\mathbf{v}_k^- = \mathbf{v}_k^+ - \Delta v_{\max_k} \mathbf{u}_k \tag{3.4}$$

for backward propagation. Note that in the backwards propagation half phase, the $-$ and $+$ superscripts on the pre- and post-impulse spacecraft velocities preserve the manner by which a trajectory will be physically flown and not the way it is numerically propagated in the solver. The 3x1 vector $\mathbf{u}_k$ contains the control parameters associated with the k^{th} maneuver, for which the norm is bounded by one (the "up-to-unit vector control" stage constraint [158]), i.e.

$$\mathbf{u}_k = \begin{bmatrix} u_{x_k} & u_{y_k} & u_{z_k} \end{bmatrix}^T; \qquad \|\mathbf{u}_k\| \leq 1 \tag{3.5}$$

Each MGALT phase contains N control vectors. The spacecraft's mass across the k^{th} bounded impulse is computed using the following equation:

$$m_k = \begin{cases} m_{k-1} - \|\mathbf{u}_{k-1}\| \, \Delta m_{\max_{k-1}} & \text{forward propagation} \\ m_{k+1} + \|\mathbf{u}_k\| \, \Delta m_{\max_k} & \text{backward propagation} \end{cases} \tag{3.6}$$

where $\Delta m_{\max_k} = D \; \Delta t_k \; \dot{m}_{\max_k}$ and $\Delta t_k = \frac{(t_f - t_0)}{N}$. Note that the impulsive thrust approximation implies $m_{k-1}^+ = m_k^-$ and for notational convenience, we set $m_k^- = m_k$.

An MGALT phase is a specific variety of the general FBS phase. As such, it inherits the constraints and decision variables as described in Table 2.1 and section 2.2, and adds to them the components of the control vectors $\mathbf{u}_k$.

The maximum available thrust $T_{\max}$ and the maximum mass-flow rate $\dot{m}_{\max}$ are computed using a propulsion system power model [113, 159]. These quantities are typically functions of their input power P, which is in turn a function of the spacecraft's distance from the sun $r_{s/\odot}$ for the case of a solar electric power system. In advanced power system models that model hardware degradation, the power available may also be dependent on the time since launch, and will therefore have a direct dependency on the current epoch t i.e.

$$T_{\max_k}(P_k(r_{s/\odot}, t)); \quad \dot{m}_{\max_k}(P_k(r_{s/\odot}, t)) \tag{3.7}$$

3.2 Multiple Gravity-Assist with n Deep-Space Maneuvers Trajectory Model

The multiple gravity assists with n deep space maneuvers per phase, using a shooting technique (MGAnDSMs), is an FBS phase that models the trajectory of a spacecraft using a chemical engine [152]. A typical single MGAnDSMs phase is depicted in Fig. 3.3. This phase type allows for the number n of mid-course maneuvers to be selected a priori or by an outer-loop optimizer. The maneuvers are separated in time by a Δt_k optimization variable that determines their location in the phase. Note that maneuvers at the phase end points are also possible. Alternatively, the phase endpoint may be a gravity-assist body. If fewer than n maneuvers are optimal for the transfer, the formulation is structured such that one or more of the potential maneuvers will have a magnitude of zero.

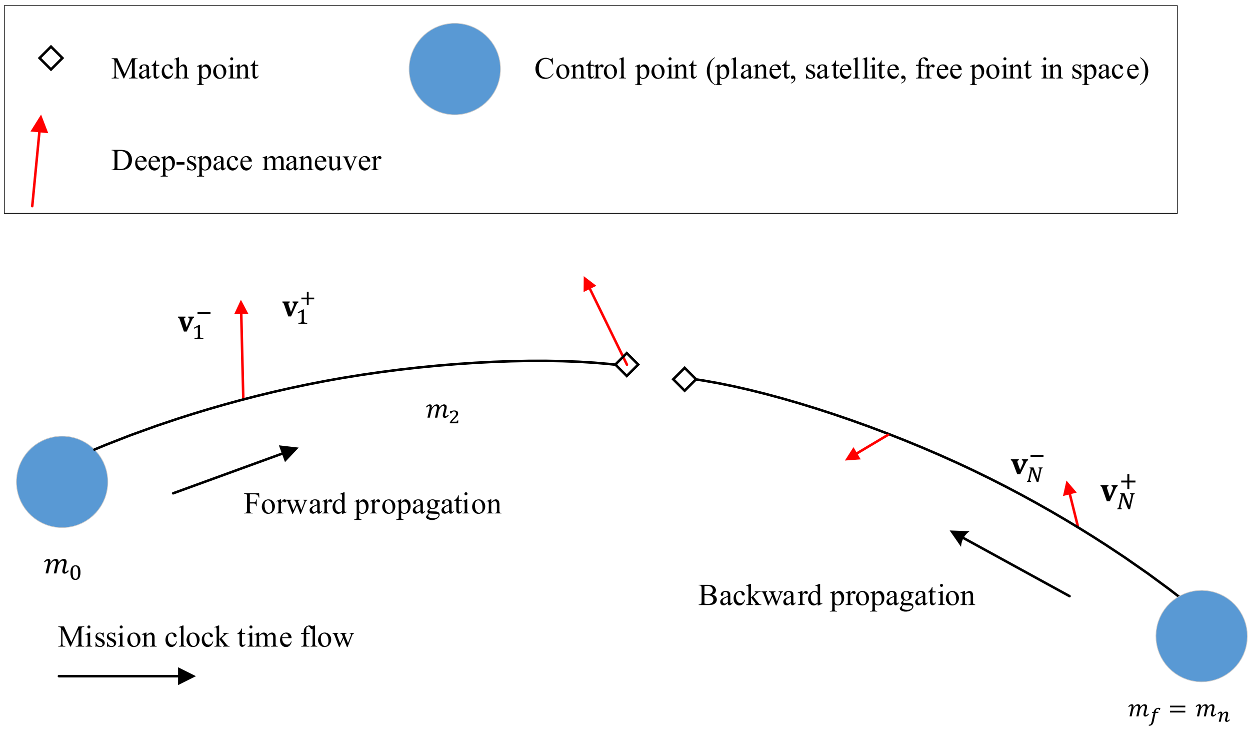


Figure 3.3: A single MGAnDSMs phase.

Unlike MGALT, the applied Δv at each maneuver is encoded directly as a decision variable:

$$\mathbf{v}_k^+ = \mathbf{v}_k^- + \Delta\mathbf{v}_k \tag{3.8}$$

Furthermore, mass is propagated using the Tsiolkovsky rocket equation:

$$m_{k+1} = m_k e^{-\Delta v_k / v_\mathrm{e}} \tag{3.9}$$

where v_e is the rocket's exhaust velocity measured relative to the vehicle.

As with MGALT, the specialization of an FBS phase to an MGAnDSMs phase introduces several additional optimization parameters and constraints. As summarized in Table 3.1, for this phase type, the mid-course maneuver vectors (if any), and the relative time variables Δt_k that separate them must be selected by the NLP solver.

Table 3.1: Additional decision variables for an MGAnDSMs phase.

x_i	Description	Number
Δt_1	time to first maneuver	1
$\Delta t_2, ..., \Delta t_n$	inter-maneuver times	$n-1$
Δt_{n+1}	time from last maneuver to phase end	1
$\Delta \mathbf{v}_1, ..., \Delta \mathbf{v}_n$	DSM vectors	$3n$

The inter-maneuver times $\Delta t_2, ..., \Delta t_n$ are only present for $n > 1$ and, for any single phase, the inter-maneuver times Δt_k are constrained such that their sum does not exceed the phase flight time:

$$\sum_{k=1}^{n+1} \Delta t_k - \Delta t_p = 0 \tag{3.10}$$

In practice, the true decision variable selected by the NLP solver, is $\alpha_k \in [0,1]$, with $\Delta t_k = \alpha_k \Delta t_p$. Then, the elements in the set $\{\alpha_i\}$ are constrained such that their sum does not exceed one.

3.3 Two-Body Propagation Using a Universal Variable Formulation

The transcriptions described in this paper utilize Keplerian two-body propagation using the Lagrange coefficients [160], Eq. (3.1). It should be noted that this could be replaced with numerical integration in the case of MGAnDSMs for the purpose of including additional dynamics thereby increasing the fidelity of the solution. For the case of MGALT, if the two-body propagation is replaced with numerical integration, the impulsive approximation should be discarded in favor of including the thrust term directly into the differential equations of motion [159].

It is generally beneficial to use a two-body propagation method capable of robustly propagating any orbit initial condition (i.e. elliptical, parabolic or hyperbolic) initialized by a search method. For this reason, a propagation method based on universal orbit variables is employed. Here we extend the propagator described by Der [161] to any conic orbit. The universal variables are defined according to the energy regime of the orbit. For elliptic trajectories:

$$\alpha = \frac{1}{a} = \frac{2}{r_k} - \frac{v_k^2}{\mu} > 0 \tag{3.11}$$

then defining

$$y = \alpha\chi^2 \tag{3.12}$$

$$C = \frac{1}{y}\left(1 - \cos(\sqrt{y})\right) \tag{3.13}$$

$$S = \frac{1}{y^3}\left(\sqrt{y} - \sin(\sqrt{y})\right) \tag{3.14}$$

the universal variables become:

$$\begin{cases} U_1 = \chi(1 - yS) \\ U_2 = \chi^2 C \\ U_3 = \chi^3 S \\ U_0 = 1 - \alpha U_2 \end{cases} \tag{3.15}$$

For a hyperbolic trajectory, $\alpha < 0$ and then

$$\begin{cases} U_0 = \cosh(\sqrt{-\alpha\chi}) \\ U_1 = \frac{1}{\sqrt{-\alpha\chi}}\sinh(\sqrt{-\alpha\chi}) \\ U_2 = \frac{1}{\alpha}(1 - U_0) \\ U_3 = \frac{1}{\alpha}(\chi - U_1) \end{cases} \tag{3.16}$$

Parabolic trajectories are unlikely to be initialized by a search method. However, they can occur when an optimizer transitions a hyperbolic trajectory to an elliptic one, or vice versa. In this case, $\alpha = 0$ and

$$\begin{cases} U_0 = 1 \\ U_1 = \chi \\ U_2 = \frac{1}{2}U_1\chi \\ U_3 = \frac{1}{3}U_2\chi \end{cases} \tag{3.17}$$

Since it will never be the case that α will be exactly equal to zero (to machine precision), we find that a tolerance of $|\alpha| < 1 \times 10^{-12}$ works well.

The Lagrange coefficients and their time derivatives are given by:

$$F = 1 - \frac{U_2}{r_k} \tag{3.18}$$

$$\dot{F} = -\frac{\sqrt{\mu}}{r_{k+1}r_k}U_1 \tag{3.19}$$

$$\ddot{F} = -\frac{\sqrt{\mu}}{r_k}\left(\frac{\dot{U}_1}{r_{k+1}} - U_1\frac{\dot{r}_{k+1}}{r_{k+1}^2}\right) \tag{3.20}$$

$$G = \frac{1}{\sqrt{\mu}}\left(r_k U_1 + \sigma_k U_2\right) \tag{3.21}$$

$$\dot{G} = 1 - \frac{U_2}{r_{k+1}} \tag{3.22}$$

$$\ddot{G} = -\left(\frac{\dot{U}_2}{r_{k+1}} - U_2\frac{\dot{r}_{k+1}}{r_{k+1}^2}\right) \tag{3.23}$$

where,

$$r_{k+1} = r_k U_0 + \sigma_k U_1 + U_2 \tag{3.24}$$

$$\dot{r}_{k+1} = r_k \dot{U}_0 + \sigma_k \dot{U}_1 + \dot{U}_2 \tag{3.25}$$

$$\sigma_{k+1} = \sigma_k U_0 + (1 - \alpha r_k) U_1 \tag{3.26}$$

$$\sigma_k = \frac{\mathbf{r}_k \cdot \mathbf{v}_k}{\sqrt{\mu}} \tag{3.27}$$

$$\frac{\partial U_0}{\partial \chi} = -\alpha U_1; \quad \frac{\partial U_n}{\partial \chi} = U_{n-1} \quad n = 1, 2, ... \tag{3.28}$$

$$\frac{\partial \chi}{\partial t} = \frac{\sqrt{\mu}}{r} \tag{3.29}$$

Kepler's equation in terms of the universal variable may be written as follows:

$$f = r_k U_1 + \sigma_k U_2 + U_3 - \sqrt{\mu} \Delta t \tag{3.30}$$

where,

$$\frac{\partial f}{\partial \chi} = r_k \tag{3.31}$$

$$\frac{\partial^2 f}{\partial \chi^2} = \sigma_k \tag{3.32}$$

Solution of Eq. (3.30) for the root χ is achieved by iteration using the Laguerre-Conway method [162], using the following update scheme:

$$\chi_{i+1} = \chi_i - \delta\chi; \quad \chi_0 = \begin{cases} \alpha\sqrt{\mu}\Delta t, & \text{if } \alpha > 0 \\ \frac{\sqrt{\mu}}{10 r_k}\Delta t, & \text{otherwise} \end{cases} \tag{3.33}$$

where,

$$\delta\chi = \begin{cases} \frac{Nf}{\frac{\partial f}{\partial \chi} \pm \sqrt{\eta}}, & \text{if } \eta > 0 \\ \frac{f}{\frac{\partial f}{\partial \chi}}, & \text{otherwise} \end{cases} \tag{3.34}$$

and,

$$\eta = \left| (N-1)^2 \left(\frac{\partial f}{\partial \chi} \right)^2 - N(N-1) f \frac{\partial^2 f}{\partial \chi^2} \right| \tag{3.35}$$

Laguerre's root finding method, when applied to a general smooth function f, requires that the sign in the denominator of Eq. (3.34) be positive if $f^{'}$ (Eq. (3.31) in this case) is non-negative and negative otherwise. Since Eq. (3.31) can never be negative for the particular case of Eq. (3.30), the sign will always be positive. The order N is typically set to 5, but can be increased if numerical instabilities are encountered.

3.4 Match Point Gradient Computation

A large majority of the dense entries of the Jacobian matrix for the MGALT and MGAnDSMs transcriptions are comprised of partial derivatives of the phase match points; this is because the partials of the match point constraint vector $\mathbf{c}_{\text{mp}}$ are dense with respect to nearly all entries in the decision vector $\mathbf{x}$. These derivatives are also the most complicated Jacobian entries to calculate. At the most general level, their computation can be summarized by the following expression:

$$\frac{\partial \mathbf{c}_{\text{mp}}}{\partial \mathbf{x}} = \frac{\partial \mathbf{X}^{\text{B}}_{\text{mp}}}{\partial \mathbf{x}} - \frac{\mathbf{X}^{\text{F}}_{\text{mp}}}{\partial \mathbf{x}} \tag{3.36}$$

Calculating the matrix Eq. (3.36) requires the propagation of sensitivity information from various intermediate points along a phase onwards to the match point. STMs can be used to map derivatives across Keplerian arcs. The bounded impulse approximation introduces a velocity discontinuity at the location of each maneuver (i.e. $\mathbf{r}(t) \in C^0$ and $\mathbf{v}(t) \in C^{-1}$). For this reason, in addition to the STMs, a method is also required for mapping derivative information across these discontinuities. This mapping can also be expressed as a matrix and is hereafter referred to as the maneuver transition matrix (MTM). The alternating STM/MTM derivative mapping technique is illustrated in Fig. 3.4. It is useful, to associate the maneuver index k with the STMs and MTMs in order to keep track of the large number of calculations required to compute the match point partials.

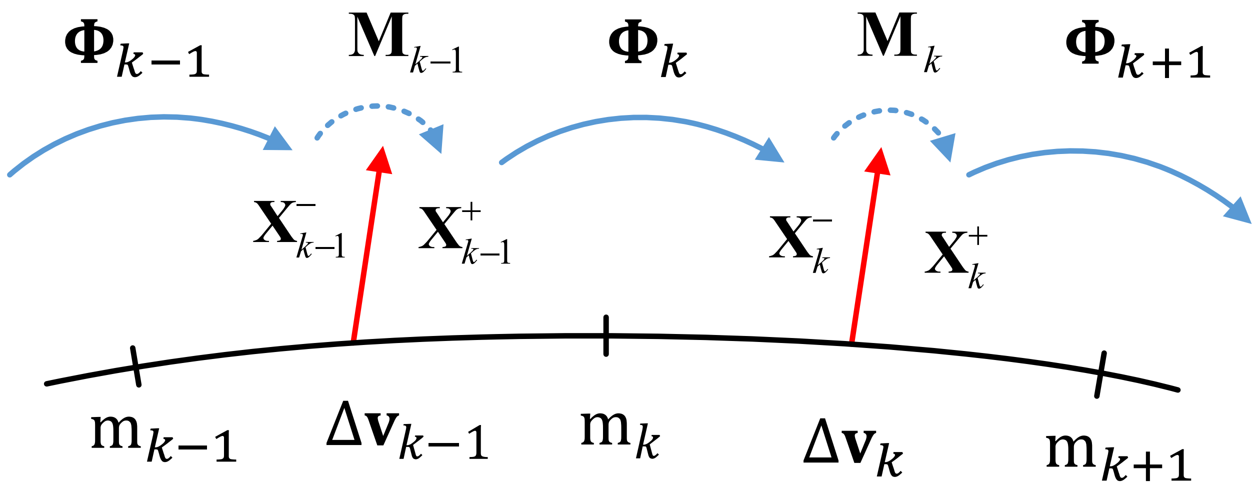


Figure 3.4: Derivative mapping matrices.

Computation of the match point partial derivatives is complicated by the complex functional dependencies created during the propagation of the spacecraft state. The dependency map for the MGALT transcription, especially when it is coupled with a solar electric power model, is particularly intricate as shown in Fig. 3.5, where the + and - superscripts have been dropped from the position vector at the time of the applied maneuver, i.e. $\mathbf{r}_k^+ = \mathbf{r}_k^- = \mathbf{r}_k$ due to the impulsive maneuver approximation.

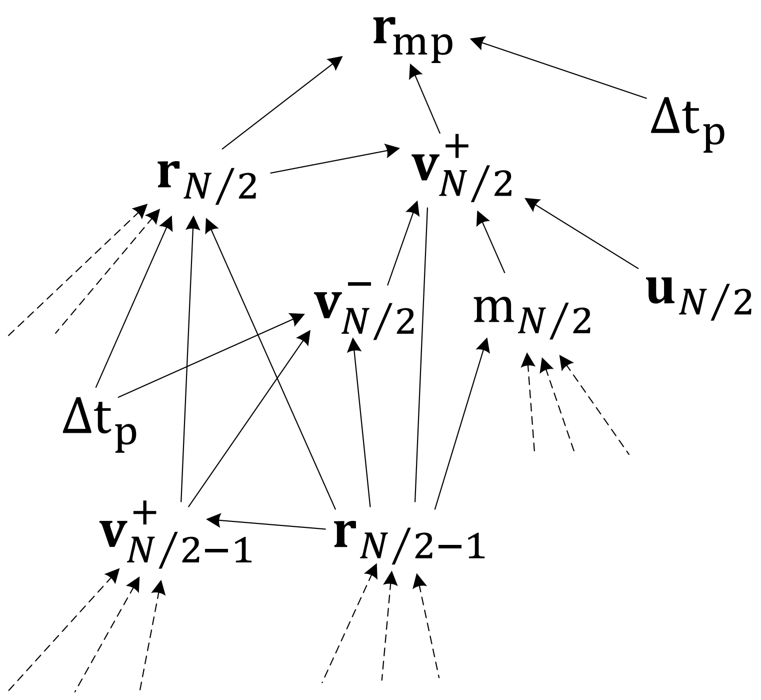


Figure 3.5: Forward half phase MGALT functional dependency tree.

3.4.1 State and Derivative Propagation Sweeps

Multiplication of these matrices is transitive, allowing chains of STM-MTM pairs to propagate derivative information from any point along the phase to the match point, as shown in Fig. 3.6.

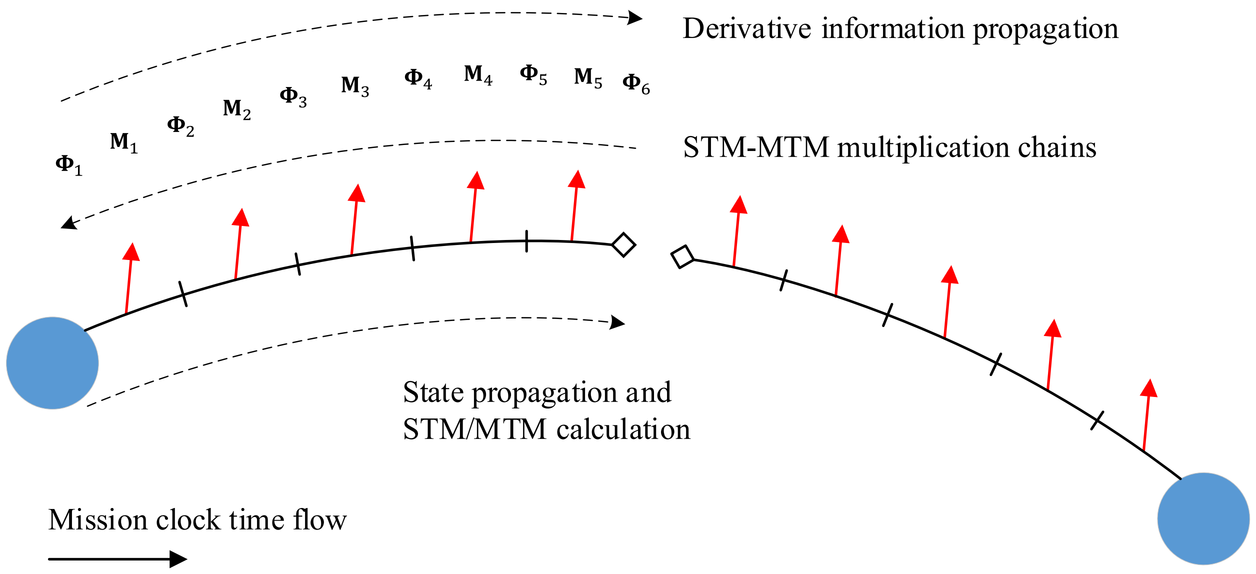

Derivative information propagation
Φ_1 M_1 Φ_2 M_2 Φ_3 M_3 Φ_4 M_4 Φ_5 M_5 Φ_6
STM-MTM multiplication chains
State propagation and
STM/MTM calculation
Mission clock time flow

Figure 3.6: Example of a 10 segment MGALT phase. Match point derivatives are calculated in three steps: 1) the STMs and MTMs are computed as the state is propagated to the match point, 2) the STM-MTM multiplication chains are assembled from the match point outwards to the phase boundaries, 3) derivative information is propagated from points along the phase to the match point.

First, the spacecraft state ($\mathbf{X}$) is propagated from both phase boundaries inward to the defect in the center of the phase. As this propagation proceeds, the STMs and MTMs are calculated and stored. Then STM-MTM matrix chains are constructed in a backwards sweep that opposes the direction of state propagation. It is most efficient to perform this sweep beginning at the match point and then proceed towards the phase boundary as the STM-MTM sub-chains can be successively constructed by recursively combining previously computed sub-chains, as illustrated in Fig. 3.7 for a forward half-phase. The same holds true for backward propagated half-phases.

Finally, the STM-MTM chains can be used to propagate derivative information from any point along the phase to the match point in order to compute the defect constraint $\mathbf{c}_{\mathrm{mp}}$ there. The calculation of STMs and MTMs and the construction of the matrix chains represents the majority of the computational effort required to evaluate a two-point shooting phase as depicted in Fig. 3.6.

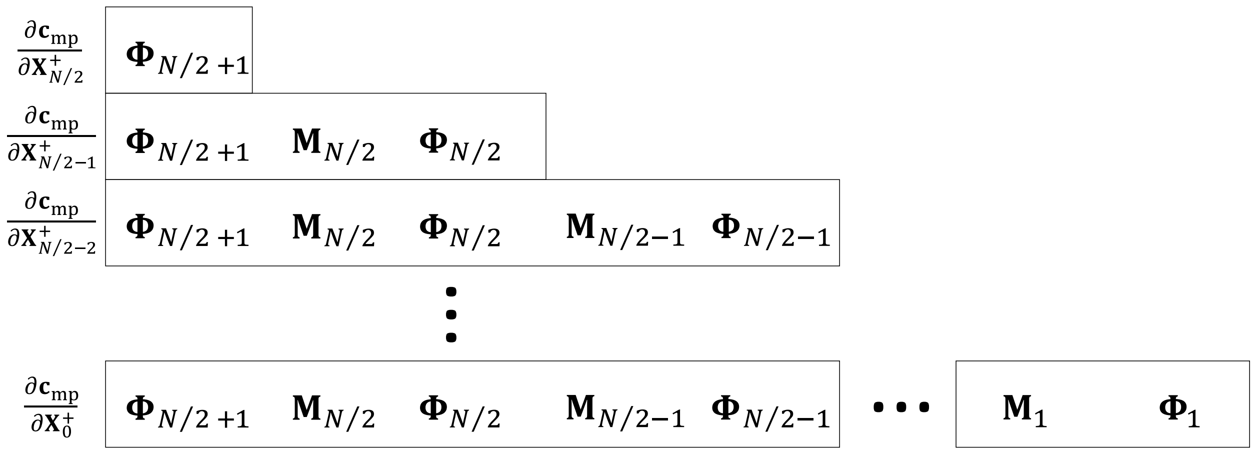

$\frac{\partial \mathbf{c}_{\mathrm{mp}}}{\partial \mathbf{X}^+_{N/2}}$
$\mathbf{\Phi}_{N/2+1}$
$\frac{\partial \mathbf{c}_{\mathrm{mp}}}{\partial \mathbf{X}^+_{N/2-1}}$
$\mathbf{\Phi}_{N/2+1}$ $\mathbf{M}_{N/2}$ $\mathbf{\Phi}_{N/2}$
$\frac{\partial \mathbf{c}_{\mathrm{mp}}}{\partial \mathbf{X}^+_{N/2-2}}$
$\mathbf{\Phi}_{N/2+1}$ $\mathbf{M}_{N/2}$ $\mathbf{\Phi}_{N/2}$ $\mathbf{M}_{N/2-1}$ $\mathbf{\Phi}_{N/2-1}$
$\frac{\partial \mathbf{c}_{\mathrm{mp}}}{\partial \mathbf{X}^+_{0}}$
$\mathbf{\Phi}_{N/2+1}$ $\mathbf{M}_{N/2}$ $\mathbf{\Phi}_{N/2}$ $\mathbf{M}_{N/2-1}$ $\mathbf{\Phi}_{N/2-1}$ ••• $\mathbf{M}_1$ $\mathbf{\Phi}_1$

Figure 3.7: Recursive construction of the STM-MTM chain for a forward propagated half-phase.

Two-Body State Transition Matrix

The two-body perturbation state transition matrix contains the first-order sensitivities of the spacecraft's position and velocity vectors at arbitrary time t_k with respect to variations in the position/velocity vector at a previous time t_{k-1}.

$$\mathbf{\Phi}(t_k, t_{k-1}) = \begin{bmatrix} \tilde{\mathbf{R}}(t) & \mathbf{R}(t) \\ \tilde{\mathbf{V}}(t) & \mathbf{V}(t) \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{r}_k}{\partial \mathbf{r}_{k-1}} & \frac{\partial \mathbf{r}_k}{\partial \mathbf{v}_{k-1}} \\ \frac{\partial \mathbf{v}_k}{\partial \mathbf{r}_{k-1}} & \frac{\partial \mathbf{v}_k}{\partial \mathbf{v}_{k-1}} \end{bmatrix} \tag{3.37}$$

The 3x3 quadrants of the perturbation STM can be computed analytically and are available from several sources [160, 163–167]. The expressions for these submatrices developed by Battin [160] are provided in the Appendix (Equations (A.1) - (A.4)).

Augmented State Transition Matrix

In order to account for first order sensitivities of mass, time-of-flight and launch epoch/previous phase flight times, the spacecraft state vector is augmented with two slots, one to hold the current phase flight time and one to hold a previous phase's flight time.

$$\mathbf{X} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ m \\ \Delta t_{\text{p}} \\ \Delta t_{\text{previous}} \end{bmatrix} \tag{3.38}$$

Going forward, the vector **X** will refer to this augmented state vector. The two-body STM is expanded with additional rows and columns to accommodate the augmented state vector:

$$\begin{aligned}
\mathbf{\Phi}_k &= \frac{\partial \mathbf{X}_k^-}{\partial \mathbf{X}_{k-1}^+} \\
&= \left[\begin{array}{cc|c|c|c} \tilde{\mathbf{R}}_k(t) & \mathbf{R}_k(t) & \mathbf{0}_{6\times1} & \frac{\partial \mathbf{r}_k^-}{\partial \Delta t_p} & \mathbf{0}_{6\times1} \\ \tilde{\mathbf{V}}_k(t) & \mathbf{V}_k(t) & & \frac{\partial \mathbf{v}_k^-}{\partial \Delta t_p} & \\ \hline \multicolumn{2}{c|}{\mathbf{0}_{3\times6}} & \multicolumn{3}{c}{\mathbb{I}_{3\times3}} \end{array}\right] \\
&= \left[\begin{array}{cc|c|c|c} \frac{\partial \mathbf{r}_k^-}{\partial \mathbf{r}_{k-1}^+} & \frac{\partial \mathbf{r}_k^-}{\partial \mathbf{v}_{k-1}^+} & \mathbf{0}_{6\times1} & \frac{\partial \mathbf{r}_k^-}{\partial \Delta t_p} & \mathbf{0}_{6\times1} \\ \frac{\partial \mathbf{v}_k^-}{\partial \mathbf{r}_{k-1}^+} & \frac{\partial \mathbf{v}_k^-}{\partial \mathbf{v}_{k-1}^+} & & \frac{\partial \mathbf{v}_k^-}{\partial \Delta t_p} & \\ \hline \multicolumn{2}{c|}{\mathbf{0}_{3\times6}} & \multicolumn{3}{c}{\mathbb{I}_{3\times3}} \end{array}\right]
\end{aligned} \tag{3.39}$$

Proceeding from the left, the first additional row/column in Eq. (3.39) is for the spacecraft's mass, then current phase time-of-flight and previous phase flight times. This augmented STM may be used for both the MGALT and MGAnDSMs transcriptions as both transcriptions use Keplerian two-body propagation methods. Note that Eq. (3.39) depicts the STM from a forward half phase, the index and impulsive burn notational convention changes slightly for a backwards half phase:

$$\mathbf{\Phi}_k = \frac{\partial \mathbf{X}_k^+}{\partial \mathbf{X}_{k+1}^-} \quad \text{(backwards half phase STM)} \tag{3.40}$$

It is important to note that the computations associated with previous phase flight times are valid for *all* previous phase flight times as well as the launch/departure epoch. A perturbation in any one of these past epoch variables impacts the current state in the same manner. The inclusion of the two additional variables in the state vector is for the convenience of expressing the calculations in the following sections as matrix operations and does not result in a corresponding augmentation of the match point constraint (Eq. 2.4), which remains a defect in position, velocity and mass only.

MGALT Maneuver Transition Matrix

Unlike the STM, the MTM is transcription dependent due to the presence of thruster hardware modeling for MGALT and because mass is propagated differently for the two trajectory models. The MGALT MTM is calculated as follows:

$$
\begin{aligned}
\mathbf{M}_k &= \frac{\partial \mathbf{X}_k^+}{\partial \mathbf{X}_k^-} \\
&= \left[\begin{array}{cc|c|c|c}
\frac{\partial \mathbf{r}_k^+}{\partial \mathbf{r}_k^-} & \frac{\partial \mathbf{r}_k^+}{\partial \mathbf{v}_k^-} & \frac{\partial \mathbf{r}_k^+}{\partial m_k} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\
\frac{\partial \mathbf{v}_k^+}{\partial \mathbf{r}_k^-} & \frac{\partial \mathbf{v}_k^+}{\partial \mathbf{v}_k^-} & \frac{\partial \mathbf{v}_k^+}{\partial m_k} & \mathbf{M}_{k_{24}} & \mathbf{M}_{k_{25}} \\
\hline
\frac{\partial m_{k+1}}{\partial \mathbf{r}_k^-} & \mathbf{0}_{1\times3} & 1 & \mathbf{M}_{k_{34}} & \mathbf{M}_{k_{35}} \\
\hline
\multicolumn{3}{c|}{\mathbf{0}_{2\times7}} & \multicolumn{2}{c}{\mathbb{I}_{2\times2}}
\end{array}\right] \\
&= \left[\begin{array}{cc|c|c|c}
\mathbb{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\
\frac{\partial \mathbf{v}_k^+}{\partial \mathbf{r}_k^-} & \mathbb{I}_{3\times3} & \frac{\partial \mathbf{v}_k^+}{\partial m_k} & \mathbf{M}_{k_{24}} & \mathbf{M}_{k_{25}} \\
\hline
\frac{\partial m_{k+1}}{\partial \mathbf{r}_k^-} & \mathbf{0}_{1\times3} & 1 & \mathbf{M}_{k_{34}} & \mathbf{M}_{k_{35}} \\
\hline
\multicolumn{3}{c|}{\mathbf{0}_{2\times7}} & \multicolumn{2}{c}{\mathbb{I}_{2\times2}}
\end{array}\right]
\end{aligned}
\tag{3.41}
$$

For forward propagated half phases:

$$
\frac{\partial \mathbf{v}_k^+}{\partial \mathbf{r}_k^-} = \frac{\partial \Delta v_{\max_k}}{\partial \mathbf{r}_k^-} = \mathbf{u}_k \frac{D \Delta t_k}{m_k} \cdot \frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial r_{s/\odot_k}} \cdot \frac{\partial r_{s/\odot_k}}{\partial \mathbf{r}_k} \tag{3.42}
$$

The last term in Eq. (3.42) implies that these calculations are valid for any central body, i.e.,

$$\mathbf{r}_{s/\odot} = \mathbf{r} - \mathbf{r}_{\odot} \tag{3.43}$$

where $\mathbf{r}_{\odot}$ is the position of the sun with respect to the central body. This sub-matrix is similarly calculated for backwards propagated half phases, with the addition of an extra term that accounts for the fact that in Eq. (3.6), m_k has a direct dependence on $\dot{m}_{\max_k}$:

$$\begin{aligned}\frac{\partial \mathbf{v}_k^-}{\partial \mathbf{r}_k^+} = \mathbf{u}_k \frac{D\Delta t_k}{m_k} \cdot \frac{\partial P_k}{\partial r_{s/\odot_k}} \cdot \frac{\partial r_{s/\odot_k}}{\partial \mathbf{r}_k} \cdot \Bigg[\frac{\partial T_{\max_k}}{\partial P_k} \\ - \|\mathbf{u}_k\| \frac{D\Delta t_k T_{\max_k}}{m_k} \cdot \frac{\partial \dot{m}_k}{\partial P_k} \Bigg]\end{aligned} \tag{3.44}$$

From Eq. (3.3) and (3.4), it follows that:

$$\frac{\partial \mathbf{v}_k^+}{\partial m_k} = \frac{\partial \Delta v_{\max_k}}{\partial m_k} = -\mathbf{u}_k \frac{D\Delta t_k T_{\max_k}}{m_k^2} = -\mathbf{u}_k \frac{\Delta v_{\max_k}}{m_k} \tag{3.45}$$

$$\frac{\partial \mathbf{v}_k^-}{\partial m_{k+1}} = \frac{\partial \Delta v_{\max_k}}{\partial m_k} \cdot \frac{\partial m_k}{\partial m_{k+1}} = \mathbf{u}_k \frac{D\Delta t_k T_{\max_k}}{m_k^2} = \mathbf{u}_k \frac{\Delta v_{\max_k}}{m_k} \tag{3.46}$$

The sensitivity of the post/pre-burn mass to changes in the spacecraft's current position is calculated as follows:

$$\frac{\partial m_{k+1}}{\partial \mathbf{r}_k^-} = -\|\mathbf{u}_k\| D\Delta t_k \, \frac{\partial \dot{m}_k}{\partial P_k} \cdot \frac{\partial P_k}{\partial r_{s/\odot_k}} \cdot \frac{\partial r_{s/\odot_k}}{\partial \mathbf{r}_k} \tag{3.47}$$

$$\frac{\partial m_k}{\partial \mathbf{r}_k^+} = \|\mathbf{u}_k\| D\Delta t_k \, \frac{\partial \dot{m}_k}{\partial P_k} \cdot \frac{\partial P_k}{\partial r_{s/\odot_k}} \cdot \frac{\partial r_{s/\odot_k}}{\partial \mathbf{r}_k} \tag{3.48}$$

The submatrices $\mathbf{M}_{k_{24}}$ and $\mathbf{M}_{k_{34}}$ will be discussed in section 3.4.3 and $\mathbf{M}_{k_{25}}$ and $\mathbf{M}_{k_{35}}$ in section 3.4.4.

MGAnDSMs Maneuver Transition Matrix

The MTM changes slightly for the high-thrust case as previously mentioned since mass is propagated using the rocket equation. The MTM becomes:

$$\mathbf{M}_k = \frac{\partial \mathbf{X}_k^+}{\partial \mathbf{X}_k^-} = \left[\begin{array}{ccc|c} \mathbb{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \\ \mathbf{0}_{3\times3} & \mathbb{I}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{7\times2} \\ \cline{1-3} \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{M}_{k_{33}} & \\ \hline & \mathbf{0}_{2\times7} & & \mathbb{I}_{2\times2} \end{array}\right] \tag{3.49}$$

where,

$$\mathbf{M}_{k_{33}} = \frac{\partial m_{k+1}}{\partial m_k} = e^{-\Delta v_k / v_e}. \tag{3.50}$$

STM-MTM Chain

Once $\mathbf{\Phi}_k$ and $\mathbf{M}_k$ have been calculated for each segment, propagation of derivative information from any point along the trajectory onward to the match point is achieved through sequential multiplication of these matrices:

$$\frac{\partial \mathbf{X}_F^\dagger}{\partial \mathbf{X}_k^+} = \mathbf{\Phi}_{N/2+1}\mathbf{M}_{N/2}\mathbf{\Phi}_{N/2} \cdot \ldots \cdot \mathbf{M}_{k+1}\mathbf{\Phi}_{k+1} \tag{3.51}$$

The only remaining step towards obtaining the actual derivatives of interest in Eq. (3.36) is to compute the derivative $\frac{\partial \mathbf{X}_k^+}{\partial x_i}$, that is the sensitivity of the spacecraft's state immediately following the k^{th} impulse to changes in the i^{th} element of the decision vector. One or more of these derivatives may be contained in the derivative connection matrix $\mathbf{\Xi}_k$. Details on the calculation of $\mathbf{\Xi}_k$ vary depending on the particular p_i being considered and are provided in subsequent sections for several decision vector entries. After computing the derivative connection matrix, the match point Jacobian entries may be calculated as follows:

$$\mathbf{\Phi}_{N/2+1}\mathbf{M}_{N/2}\mathbf{\Phi}_{N/2} \cdot \ldots \cdot \mathbf{M}_{k+1}\mathbf{\Phi}_{k+1}\mathbf{\Xi}_k \tag{3.52}$$

3.4.2 Partials With Respect to Segment Control Variables

For both bounded impulse models, the majority of the problem decision variables are those defining the magnitude and the direction of thrust applied over the course of each segment. The MTM submatrices corresponding to these variables are computed differently for the two models.

MGALT Control Variables

For the low-thrust case, the actual decision variable is the "up-to-unit" throttle vector that scales the maximum allowed Δv for the segment. For forward propagated half phases, from Eq. (3.3):

$$\frac{\partial \mathbf{v}_k^+}{\partial \mathbf{u}_k} = \Delta v_{\max_k} \mathbb{I}_{3\times 3} \tag{3.53}$$

and from Eq. (3.6):

$$\frac{\partial m_{k+1}}{\partial \mathbf{u}_k} = -\frac{\mathbf{u}_k^T}{\|\mathbf{u}_k\|} D\ \Delta t_k\ \dot{m}_{\max_k} \tag{3.54}$$

therefore,

$$\boldsymbol{\Xi}_k = \frac{\partial \mathbf{X}_k^+}{\partial \mathbf{u}_k} = \left[\begin{array}{c} \mathbf{0}_{3\times 3} \\ \hline \frac{\partial \mathbf{v}_k^+}{\partial \mathbf{u}_k} \\ \hline \frac{\partial m_{k+1}}{\partial \mathbf{u}_k} \\ \hline \mathbf{0}_{2\times 3} \end{array}\right] \tag{3.55}$$

and hence,

$$\frac{\partial \mathbf{X}_{\text{mp}}^{\text{F}}}{\partial \mathbf{u}_k} = \begin{cases} \boldsymbol{\Phi}_{k+1}\boldsymbol{\Xi}_k & k = N/2 \\ \prod_{i=1}^{N/2-k} \left(\boldsymbol{\Phi}_{N/2+2-i}\mathbf{M}_{N/2+1-i}\right) \boldsymbol{\Phi}_{k+1}\boldsymbol{\Xi}_k & 1 \leq k < N/2 \end{cases} \tag{3.56}$$

When considering derivatives of the match point constraint with respect to control parameters in the backwards propagated half phase, the method changes slightly and $\frac{\partial \mathbf{X}_k^-}{\partial \mathbf{u}_k}$ must now be calculated:

$$\frac{\partial \mathbf{v}_k^-}{\partial \mathbf{u}_k} = \frac{D\Delta t_k\ \dot{m}_{\max_k}}{\|\mathbf{u}_k\|} \Delta v_{\max_k} \mathbf{u}_k \mathbf{u}_k^T - \Delta v_{\max_k} \mathbb{I}_{3\times 3} \tag{3.57}$$

and from Eq. (3.6):

$$\frac{\partial m_{k+1}}{\partial \mathbf{u}_k} = \frac{\mathbf{u}_k^T}{\|\mathbf{u}_k\|} D \; \Delta t_k \; \dot{m}_{\max_k} \tag{3.58}$$

therefore,

$$\boldsymbol{\Xi}_k = \frac{\partial \mathbf{X}_k^-}{\partial \mathbf{u}_k} = \left[\begin{array}{c} \mathbf{0}_{3\times3} \\ \hline \frac{\partial \mathbf{v}_k^-}{\partial \mathbf{u}_k} \\ \hline \frac{\partial m_{k+1}}{\partial \mathbf{u}_k} \\ \hline \mathbf{0}_{2\times3} \end{array}\right] \tag{3.59}$$

and hence,

$$\frac{\partial \mathbf{X}_{\text{mp}}^{\text{B}}}{\partial \mathbf{u}_k} = \begin{cases} \boldsymbol{\Phi}_{k+1}\boldsymbol{\Xi}_k & k = N/2+1 \\ \prod_{i=k}^{N} \left(\boldsymbol{\Phi}_i \mathbf{M}_{i-1}\right) \boldsymbol{\Phi}_{k+1}\boldsymbol{\Xi}_k & N/2+1 < k \leq N \end{cases} \tag{3.60}$$

MGAnDSMs Control Variables

Computing match point control sensitivities for MGAnDSMs is done differently from the procedure for the MGALT case in that the applied Δv at each mid-course maneuver is directly encoded as a decision variable (in lieu of the MGALT throttle parameter) and mass is propagated with the rocket equation. Therefore, for forward propagation:

$$\frac{\partial m_{k+1}}{\partial \Delta \mathbf{v}_k} = -\frac{m_k}{v_e}\frac{\Delta \mathbf{v}_k^T}{\Delta v_k} e^{-\Delta v_k / v_e} \tag{3.61}$$

and,

$$\boldsymbol{\Xi}_k = \frac{\partial \mathbf{X}_k^+}{\partial \Delta \mathbf{v}_k} = \left[\begin{array}{c} \mathbf{0}_{3\times3} \\ \hline \mathbb{I}_{3\times3} \\ \hline \frac{\partial m_{k+1}}{\partial \mathbf{u}_k} \\ \hline \mathbf{0}_{3\times3} \end{array}\right] \tag{3.62}$$

The procedure is similar for backwards propagation. The match point derivative calculations for phase final mass and the components of $\mathbf{v}_\infty$ are not discussed here, but their computation is analogous to the control sensitivity calculations for both MGALT and MGAnDSMs.

3.4.3 Partials with Respect to the Current Phase Flight Time

MGALT Flight Time Variables

The current phase flight time variable enters into the match point derivative computations via the STM in addition to the MTM. The STM contains explicit time dependencies of the pre-impulse position and velocity because the Kepler propagation length of each segment is computed from the current phase flight time. The method for computing these derivatives is due to Pitkin [168] and is also used by Lantoine and Russell for computing state sensitivities with respect to time-of-flight variables across one Keplerian arc [169]. The plus and minus signs correspond to forward and backward propagated half phases respectively. Expressions for computing the Lagrange coefficients F and G are provided by Battin [160] and their time derivatives are discussed in the article by Pitkin [168].

$$\frac{\partial \mathbf{r}_k^-}{\partial \Delta t_p} = \frac{\partial \mathbf{r}_k^-}{\partial \Delta t_k} \cdot \frac{\partial \Delta t_k}{\partial \Delta t_p} = \pm \left[\dot{F}\mathbf{r}_{k-1}^+ + \dot{G}\mathbf{v}_{k-1}^+\right] \frac{\partial \Delta t_k}{\partial \Delta t_p} \tag{3.63}$$

$$\frac{\partial \mathbf{v}_k^-}{\partial \Delta t_p} = \frac{\partial \mathbf{v}_k^-}{\partial \Delta t_k} \cdot \frac{\partial \Delta t_k}{\partial \Delta t_p} = \pm \left[\ddot{F}\mathbf{r}_{k-1}^+ + \ddot{G}\mathbf{v}_{k-1}^+\right] \frac{\partial \Delta t_k}{\partial \Delta t_p} \tag{3.64}$$

The derivative of segment propagation time with respect to Δt_p is calculated differently for the half-segments than it is for full segments. In the MGALT model, half-segments occur at the left and right phase boundaries, and on either side of the match point:

$$\frac{\partial \Delta t_k}{\partial \Delta t_p} = \begin{cases} \frac{1}{N} & \text{for full segments} \\ \frac{1}{2N} & \text{for half-segments} \end{cases} \tag{3.65}$$

The MTM also contains entries that facilitate the computation of partial derivatives with respect to Δt_p. Specifically, changes in $\Delta v_{\max_k}$ and $\Delta m_{\max_k}$ due to variations in Δt_p are encoded in each MTM. The submatrices $\mathbf{M}_{k_{24}}$ and $\mathbf{M}_{k_{34}}$ are comprised of Δt_p derivative directions of P_k, Δt and t_k that do not involve the position of the spacecraft at the point of the applied maneuver $\mathbf{r}_k^-$. To see what is meant by this, it is helpful to examine the partial derivative of P_k with respect to Δt_p:

$$\frac{\partial P_k}{\partial \Delta t_p} = \frac{\partial P_k}{\partial r_{s/\odot_k}} \cdot \frac{\partial r_{s/\odot_k}}{\partial \mathbf{r}_k^-} \cdot \frac{\partial \mathbf{r}_k^-}{\partial \Delta t_p} + \frac{\partial P_k}{\partial t_k} \cdot \frac{\partial t_k}{\partial \Delta t_p} \tag{3.66}$$

The first term in Eq. (5.43) is calculated by Eq.'s (3.42), (3.44) and (3.63). The second term must be accounted for by $\mathbf{M}_{k_{24}}$ and $\mathbf{M}_{k_{34}}$.

The sub-matrix $\mathbf{M}_{k_{24}}$ is calculated as follows for forward propagation, where t_k is the current mission epoch (measured from launch):

$$\mathbf{M}_{k_{24}} = \mathbf{u}_k \frac{D}{m_k} \left[\Delta t_k \frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \cdot \frac{\partial t_k}{\partial \Delta t_p} + \frac{\partial \Delta t_k}{\partial \Delta t_p} T_{\max_k} \right] \tag{3.67}$$

and with the added terms accounting for the dependence on $\dot{m}_{\max_k}$by $\Delta v_{\max_k}$ for backwards propagation:

$$\begin{aligned} \mathbf{M}_{k_{24}} =& - \mathbf{u}_k \frac{D}{m_k} \left[\Delta t_k \frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \cdot \frac{\partial t_k}{\partial \Delta t_p} + \frac{\partial \Delta t_k}{\partial \Delta t_p} T_{\max_k} \right. \\ & \left. - \|\mathbf{u}_k\| \frac{D \Delta t_k \; T_{\max_k}}{m_k} \left(\dot{m}_k \frac{\partial \Delta t_k}{\partial \Delta t_p} + \Delta t_k \; \frac{\partial \dot{m}_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \cdot \frac{\partial t_k}{\partial \Delta t_p} \right) \right] \end{aligned} \tag{3.68}$$

The derivative of the current epoch t_k with respect to Δt_p is readily calculated once one considers how the current epoch is computed for a forward propagated half phase (where t_0 is the epoch at the start of the phase):

$$\begin{aligned} t_k &= t_0 + \Delta t_1 + \Delta t_2 + \Delta t_3 + \ldots + \Delta t_k \\ &= t_0 + \frac{\Delta t_p}{2N} + \frac{\Delta t_p}{N} + \frac{\Delta t_p}{N} + \ldots + \frac{\Delta t_p}{N} \\ &= t_0 + \frac{(k - 0.5)}{N} \Delta t_p \end{aligned} \tag{3.69}$$

thus,

$$\frac{\partial t_k}{\partial \Delta t_p} = \left(\frac{k - 0.5}{N} \right) \tag{3.70}$$

For a backwards propagated half phase, the calculation is similar and the derivative is:

$$\frac{\partial t_k}{\partial \Delta t_p} = \left(\frac{0.5 - k}{N}\right) \tag{3.71}$$

The final sub-matrix in the augmented MTM is computed as follows, again with the negative sign corresponding with forward propagation, the positive sign with backward:

$$\mathbf{M}_{k_{34}} = \mp\|\mathbf{u}_k\| D \left(\dot{m}_{\max_k} \frac{\partial \Delta t_k}{\partial \Delta t_p} + \Delta t_k \frac{\partial \dot{m}_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \cdot \frac{\partial t_k}{\partial \Delta t_p}\right) \tag{3.72}$$

Computing the match point Δt_p gradients requires using the following matrix:

$$\frac{\partial \mathbf{X}_{\text{mp}}^{\text{F}}}{\partial \Delta t_p} = \mathbf{\Phi}_{N/2+1}\mathbf{M}_{N/2}\mathbf{\Phi}_{N/2} \cdot \ldots \cdot \mathbf{\Phi}_2\mathbf{M}_1\mathbf{\Xi}_1 \tag{3.73}$$

where

$$\mathbf{\Xi}_1 = \begin{bmatrix} \frac{\partial \mathbf{r}_1^-}{\partial \Delta t_p} \\ \frac{\partial \mathbf{v}_1^-}{\partial \Delta t_p} \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{3.74}$$

and

$$\frac{\partial \mathbf{X}_{\text{mp}}^{\text{B}}}{\partial \Delta t_p} = \mathbf{\Phi}_{N/2+2}\mathbf{M}_{N/2+1}\mathbf{\Phi}_{N/2+3} \cdot \ldots \cdot \mathbf{\Phi}_{N+1}\mathbf{M}_N\mathbf{\Xi}_N \tag{3.75}$$

where

$$\mathbf{\Xi}_N = \begin{bmatrix} \phi^{\text{B}} \\ \hline 0 \\ 1 \\ 0 \end{bmatrix} \tag{3.76}$$

In Eq. (3.76), ϕ accounts for the fact that the state at the right hand boundary of a phase has non-zero Δt_p

gradients:

$$\phi^{\mathrm{B}} = \begin{bmatrix} \tilde{\mathbf{R}}_{N+2} & \mathbf{R}_{N+2} \\ \tilde{\mathbf{V}}_{N+2} & \mathbf{V}_{N+2} \end{bmatrix} \frac{\partial \mathbf{X}_f}{\partial \Delta t_p} + \begin{bmatrix} \frac{\partial \mathbf{r}_N^+}{\partial \Delta t_p} \\ \frac{\partial \mathbf{v}_N^+}{\partial \Delta t_p} \end{bmatrix} \tag{3.77}$$

MGAnDSMs Flight Time Variables

Computing current phase flight time derivatives for the high thrust transcription requires that Eq. (3.65) be replaced by:

$$\frac{\partial \Delta t_k}{\partial \Delta t_p} = \alpha_k \tag{3.78}$$

since, for MGAnDSMs, the inter-maneuver Keplerian propagation times Δt_k are not uniform as they are for MGALT, rather each is constructed out of two decision variables the α_k and Δt_{p}:

3.4.4 Partials with Respect to Previous Phase Flight Times and Launch Epoch

Flight time variables from previous phases as well as the launch epoch variable affect the state at the left and right boundaries of the current phase. As a result, Eq. (3.74) must be modified to become:

$$\boldsymbol{\Xi}_1 = \begin{bmatrix} \phi^{\mathrm{F}} \\ \hline 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.79}$$

$$\phi^{\mathrm{F}} = \begin{bmatrix} \tilde{\mathbf{R}}_1 & \mathbf{R}_1 \\ \tilde{\mathbf{V}}_1 & \mathbf{V}_1 \end{bmatrix} \frac{\partial \mathbf{X}_0}{\partial \Delta t_{\mathrm{previous}}} + \begin{bmatrix} \frac{\partial \mathbf{r}_1^-}{\partial \Delta t_{\mathrm{previous}}} \\ \frac{\partial \mathbf{v}_1^-}{\partial \Delta t_{\mathrm{previous}}} \end{bmatrix} \tag{3.80}$$

Previous phase flight times do not impact the Keplerian propagation time between maneuvers in the current phase. For this reason, the STM does not contain explicit partial derivatives of the state with respect to previous time-of-flight variables. Changes in a previous phase's flight time will, however, result in a change to the current maneuver's epoch. Specifically, the current epoch's sensitivity is unity, i.e.

$$\frac{\partial t_k}{\partial \Delta t_{\text{previous}}} = 1, \tag{3.81}$$

therefore,

$$\mathbf{M}_{k_{25}} = \mathbf{u}_k \frac{D}{m_k} \left(\Delta t_k \frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \right), \tag{3.82}$$

for forward half phases,

$$\mathbf{M}_{k_{25}} = -\mathbf{u}_k \frac{D \Delta t_k}{m_k} \left(\frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} - \|\mathbf{u}_k\| \frac{D \Delta t_k \; T_{\max_k}}{m_k} \frac{\partial \dot{m}_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \right) \tag{3.83}$$

for backward half phases and

$$\mathbf{M}_{k_{35}} = \mp \|\mathbf{u}_k\| D \Delta t_k \; \frac{\partial \dot{m}_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \tag{3.84}$$

with the negative (positive) sign being used for forward (backward) half phases.

3.4.5 Partials with Respect to the MGAnDSMs Burn Indices

The match point derivatives with respect to the MGAnDSMs burn indices $\{\alpha_i\}$ are computed similarly to the phase flight time variables, except that each one influences the size of only one propagation arc in a half phase.

$$\frac{\partial \mathbf{X}_{\text{mp}}^{\text{F}}}{\partial \Delta t_k} = \mathbf{M}_{N/2} \mathbf{\Phi}_{N/2} \cdot \ldots \cdot \mathbf{\Phi}_{k+1} \mathbf{M}_k \mathbf{\Xi}_k \tag{3.85}$$

where

$$\mathbf{\Xi}_k = \begin{bmatrix} \frac{\partial \mathbf{r}_k^-}{\partial \alpha_k} \\ \frac{\partial \mathbf{v}_k^-}{\partial \alpha_k} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.86}$$

$$\frac{\partial \mathbf{X}_{\text{mp}}^{\text{B}}}{\partial \Delta t_k} = \mathbf{\Phi}_{N/2+1}\mathbf{M}_{N/2+1}\mathbf{\Phi}_{N/2+2} \cdot \ldots \cdot \mathbf{\Phi}_k \mathbf{M}_k \mathbf{\Xi}_k \tag{3.87}$$

where

$$\mathbf{\Xi}_k = \begin{bmatrix} \frac{\partial \mathbf{r}_k^+}{\partial \alpha_k} \\ \frac{\partial \mathbf{v}_k^+}{\partial \alpha_k} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.88}$$

3.5 Distance Constraint Derivative Computation

This section will discuss an operational constraint that imposes a minimum and/or maximum value on the distance between the spacecraft and bodies in the solar system, *i.e.*,

$$d_{\text{LB}} \leq r_{\text{s/c-body}} \leq d_{\text{UB}} \tag{3.89}$$

where d_{LB} and d_{UB} are defined by the analyst for each body of interest. For example, the spacecraft may be constrained to never get too close to the sun due to thermal loading, or may not be allowed to fly farther from the earth than some maximum distance for communications hardware limitations. These constraints occur often in real-world mission design, especially in low-thrust applications where the desire to prevent the spacecraft from becoming too hot is in conflict with the availability of more power and therefore more efficient propulsive capabilities closer to the sun.

The distance constraint is straightforward to pose in the optimization problem because it requires only looking up the position of the relevant solar system bodies at each segment in the trajectory. However it is quite computationally expensive for two reasons. The first is that each ephemeris lookup requires a database query, which is quite slow. The second reason is that computing analytical derivatives of the distance constraint requires recursive multiplications of the STMs and MTMs along the trajectory, similar to the STM-MTM chains used to compute the match point derivatives (e.g. Eq. (3.56)). In a large problem with many segments, over 50% of the execution time for the trajectory optimization can be accounted for by the derivative calculation code for the distance constraint.

In particular, the distance constraint and its derivatives must be computed at each maneuver point in the phase. The STM-MTM chains that must be computed in order to facilitate this are as follows:

$$\frac{\partial \mathbf{X}_j^-}{\partial \mathbf{X}_k^+} = \begin{cases} \mathbf{\Phi}_{k+1} & j = k+1 \\ \prod_{i=1}^{j-k} \left(\mathbf{\Phi}_{j+2-i}\mathbf{M}_{j+1-i}\right)\mathbf{\Phi}_{k+1} & 2 \leq k < j \leq N/2 \end{cases} \tag{3.90}$$

for forward half phases, and

$$\frac{\partial \mathbf{X}_j^+}{\partial \mathbf{X}_k^-} = \begin{cases} \mathbf{\Phi}_{k+1} & j = k-1 \\ \prod_{i=j+2}^{k} \left(\mathbf{\Phi}_{i}\mathbf{M}_{i-1}\right)\mathbf{\Phi}_{k+1} & N/2+2 \leq j < k \leq N \end{cases} \tag{3.91}$$

for backward half phases. Then, as an example of how these chains are used to compute a path constraint sensitivity, to compute the gradient of the distance of the spacecraft from some body of interest at the center of the j^{th} segment $d_j = \|\mathbf{r}_j - \mathbf{r}_{\text{body}}\|$, with respect to a previous control variable $\mathbf{u}_k$, Eq. (3.90) may be combined with Eq. (3.55):

$$\frac{\partial d_j}{\partial \mathbf{u}_k} = \frac{\partial d_j}{\partial \mathbf{X}_j^-}\frac{\partial \mathbf{X}_j^-}{\partial \mathbf{X}_k^+}\mathbf{\Xi}_k \tag{3.92}$$

It should be noted that since the distance constraint is only enforced at maneuver locations along the phase, it is really only useful in conjunction with the MGALT model, which features a large number of impulses. In order to impose a similar constraint for the MGAnDSMs transcription, additional "non-maneuver" nodes can be introduced at regular intervals along the phase that are used exclusively to enforce the distance constraint. Furthermore, this method for computing gradients of a distance constraint may actually be applied to any path constraint that the designer might be required to place on the spacecraft's trajectory.

3.6 Gradient Accuracy Verification

3.6.1 Forward Mode Automatic Differentiation Using Operator Overloading

The mathematical foundations of forward mode AD are often described using the concept of a dual number. A dual number extends the real numbers and forms a one-dimensional Grassmann algebra. This is done by extending each real number $x \in \mathbb{R}$ with a nilpotent component ϵ. An example of dual numbers being used to accumulate derivative information for the function $f(x_1, x_2) = x_1 x_2 + \ln(x_1)$ is shown in Table 3.2.

Table 3.2: Automatic differentiation performed with dual numbers.

Expression	Derivative
$f_1 = x_1$	$f_1' = x_1 + \epsilon_1$
$f_2 = x_2$	$f_2' = x_2 + \epsilon_2$
$f_3 = \ln(f_1)$	$f_3' = \ln(x_1 + \epsilon_1)$ $= \ln(x_1) + \ln(1 + \frac{\epsilon_1}{x_1})$ $= \ln(x_1) + \frac{1}{x_1}\epsilon_1 + \mathcal{O}(\epsilon_1^2)$ $= \ln(x_1) + \frac{1}{x_1}\epsilon_1$
$f_4 = f_1 f_2$	$f_4' = f_1' f_2 + f_1 f_2' = 2x_1 x_2 + x_2\epsilon_1 + x_1\epsilon_2$
$f_5 = f_3 + f_4$	$f_5' = f_3' + f_4' = 2x_1 x_2 + \ln(x_1) + \left(x_2 + \frac{1}{x_1}\right)\epsilon_1 + x_1\epsilon_2$

In the above procedure, we make use of the fact that higher order expressions involving the nilpotent ϵ will annihilate. Now, derivative information is obtained by examining the coefficients of ϵ_1 and ϵ_2 in the final row of Table 3.2, i.e.

$$\frac{\partial f}{\partial x_1} = \left(x_2 + \frac{1}{x_1}\right) \tag{3.93}$$

$$\frac{\partial f}{\partial x_2} = x_1 \tag{3.94}$$

This procedure can be reflected in a software implementation by exploiting operator overloading. The dual number algebra can be replicated by overloading the standard floating point mathematical operators such that when they are used in an elementary mathematical expression, not only is the result of the expression computed and stored in memory, but each derivative direction is propagated and stored as well. This concept is illustrated in Table 3.3. Although analytical expressions appear in the table it is emphasized that, in practice, all derivative information is stored in floating point memory without the creation or storage of symbolic expressions.

3.6.2 Accuracy Verification

The analytic techniques described in this work are not the only methods available for computing the constraint partials of a nonlinear program. Finite differencing methods are probably the simplest to implement, although this is typically accompanied by a decrease in program execution speed and worse a decrease in

Table 3.3: Forward AD using operator overloading.

Original Line of Code (Stored Value)	Overloaded Derivative Operations	Accumulated Derivative Information
$f_1 = x_1$	1 (seeded)	$[1, 0]$
$f_2 = x_2$	1 (seeded)	$[0, 1]$
$f_3 = \ln(f_1)$	$(1/f_1)f_1' = \frac{1}{x_1}$	$\left[\frac{1}{x_1}, 0\right]$
$f_4 = f_1 f_2$	$f_1' f_2 = f_2$	$[x_2, 0]$
	$f_1 f_2' = f_1$	$[x_2, x_1]$
$f_5 = f_3 + f_4$	$f_3' + f_4'$	$\left[\left(x_2 + \frac{1}{x_1}\right), x_1\right]$

the accuracy of the partials. Other techniques such as complex step differentiation and algorithmic differentiation (AD) can produce near-machine precision derivative information, and are general techniques that do not require any *a priori* information about the problem being solved. Both of these techniques typically require a longer runtime than specialized analytic formulae, if they are used to compute derivatives during a typical cost function evaluation. If they are used to compute only individual STMs and MTMs, which are then incorporated into derivative computations such as Eq. (3.51), then one would expect the runtimes to decrease. The interested reader is directed to a detailed study by Pellegrini and Russell, that examines finite differencing techniques of varying order, the complex step method as well as analytic methods as applied to STM computation for integrated trajectories [136]. This section will examine the accuracy of the partial derivative calculation methods described in this paper compared with the same derivatives obtained using central differencing as well as ones calculated using the AD library developed by Ghosh [170]. This AD software package uses C++ operator overloading in a tapeless forward-mode configuration.

An example 20 segment Earth to Mars MGALT trajectory phase was evaluated, and the Jacobian was computed using central differencing, the formulae presented in this paper as well as AD. The partial derivatives calculated with AD are assumed to be truth. An example column from the Jacobian is shown in Tables 3.4 and 3.5 for the finite differencing and analytic cases respectively. This column contains the partial derivatives the match-point continuity constraint $\mathbf{c}_{\mathrm{mp}}$ with respect to the x component of the control vector from the first segment in the phase u_{x_1} (i.e. the segment furthest from the match-point in the forward half-phase). The error of the partial derivative values relative to the values computed using AD is shown for both methods.

Table 3.4: Example match-point derivatives computed using central differencing compared with algorithmic differentiation. A central differencing step size of 1.0e-7 was used.

Derivative	Value computed with FD	Error Relative to AD value
$\frac{\partial x^\dagger}{\partial u_{x_1}}$	-2020560.390983700	0.002538938555645e-5
$\frac{\partial y^\dagger}{\partial u_{x_1}}$	-166778.4769555778	0.136749253496038e-5
$\frac{\partial z^\dagger}{\partial u_{x_1}}$	-81633.22222039030	0.084320438683687e-5
$\frac{\partial \dot{x}^\dagger}{\partial u_{x_1}}$	-0.1006144971940313	0.016670205267864e-5
$\frac{\partial \dot{y}^\dagger}{\partial u_{x_1}}$	0.07758976470341465	0.029646679151652e-5
$\frac{\partial \dot{z}^\dagger}{\partial u_{x_1}}$	0.03673856948309212	0.050559366283141e-5
$\frac{\partial m^\dagger}{\partial u_{x_1}}$	2.346964682917494	0.027179713285220e-5

Table 3.5: Example match-point derivatives computed analytically compared with algorithmic differentiation.

Derivative	Value computed analytically	Error relative to AD value
$\frac{\partial x^\dagger}{\partial u_{x_1}}$	-2020560.339682915	0.0e-14
$\frac{\partial y^\dagger}{\partial u_{x_1}}$	-166778.7050242114	0.296659647101017e-14
$\frac{\partial z^\dagger}{\partial u_{x_1}}$	-81633.29105393921	0.267389352563618e-14
$\frac{\partial \dot{x}^\dagger}{\partial u_{x_1}}$	-0.1006144804213910	0.082758193947980e-14
$\frac{\partial \dot{y}^\dagger}{\partial u_{x_1}}$	0.07758974170063281	0.035772223244046e-14
$\frac{\partial \dot{z}^\dagger}{\partial u_{x_1}}$	0.03673855090831362	0.037774456163087e-14
$\frac{\partial m^\dagger}{\partial u_{x_1}}$	2.346964045019395	0.018921858253112e-14

As one would expect, the Jacobian values obtained with central differencing differ from the AD values with a relative error on the order of 1.0e-6 to 1.0e-8. With the analytic methods presented in this paper precision at, or very close to, machine precision is achieved.

3.7 Example: Ice Giant Intercept

This problem is inspired by the Ice Giants Decadal Mission Study Final Report from the NASA Planetary Science Decadal Survey carried out by the Applied Physics Laboratory [171]. The thruster mass flow rate and thrust polynomial coefficients, sufficiently accurate for preliminary design purposes, are from the 2014 Discovery proposal NEXT thruster guide [131] and are provided in the Appendix (Table A.1). The problem was first solved assuming all parameters having the values in Table 3.6 [172] except for the following: a minimum earth flyby altitude of 300 km, a maximum time of flight of 13.5 years, a simplified solar array model $\{\gamma_0 = 1.0, \gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = 0.0\}$ and no forced coasts. This solution was then used as an initial guess for the problem as described in Table 3.6. The NLP solver was run using a finite differenced Jacobian and then using an analytic Jacobian.

Table 3.6: Uranus Probe mission parameters

Parameter	Value
Maximum allowed initial mass	4200 kg
Earliest allowed launch date	January 1st, 2020
Launch window	365.25 days
Maximum flight time	13 years
Launch vehicle	Atlas V (531) NLS-1
Launch C3	11.834 km^2/s^2
Launch declination bounds	$[-28.5°, 28.5°]$
Arrival type	intercept
Maximum arrival v_∞	7.21 km/s
Thruster	2 x NEXT TT10 high I_{sp}
Throttle logic	minimum number of thrusters
Thruster duty cycle	0.9
Solar power coefficients	$\gamma_0 = 1.1705$ $\gamma_1 = 0.0289$
	$\gamma_2 = -0.2197$ $\gamma_3 = -0.0202$
	$\gamma_4 = -0.0001$
Spacecraft bus power requirement coefficients	$a_{s/c} = 0.8$
	$b_{s/c} = 0.0$
	$c_{s/c} = 0.0$
Solar array BOL power (launch at 1 A.U.)	26.0 kW
Power margin (δ_{power})	15%
Flyby sequence	E-E-U
Minimum earth flyby altitude	1000 km
Post-launch checkout coast	30 days
Pre-Earth flyby coast	42 days
Post-Earth flyby coast	7 days
Number of segments per phase	100
Ephemeris	SplineEphem
SNOPT feasibility tolerance	1.0e-5
SNOPT optimality tolerance	1.0e-6
Objective function	maximize final mass
SNOPT max. runtime	3600 s
SNOPT major iteration limit	8000

Table 3.7: SNOPT convergence statistics for the Uranus Probe mission

Metric	Analytic	Finite Differences
Optimal normalized cost	0.80964301784	0.80964033542
SNOPT major iterations	755	1162
Feasibility achieved	1.07094e-13	2.62403e-09
SNOPT execution time (s)	15	112
Optimality tolerance met?	Yes	No

The results summarized in Table 3.7 indicate that while both Jacobian calculation methods resulted in essentially equal final cost values, the NLP solver was able to achieve a significantly tighter feasibility tolerance in fewer iterations using analytic gradients as opposed to finite differenced ones. The analytic Jacobian also led to an optimal exit condition for the NLP solver. The locally optimal trajectory found for this problem is shown in Fig. 3.8 and 3.9.

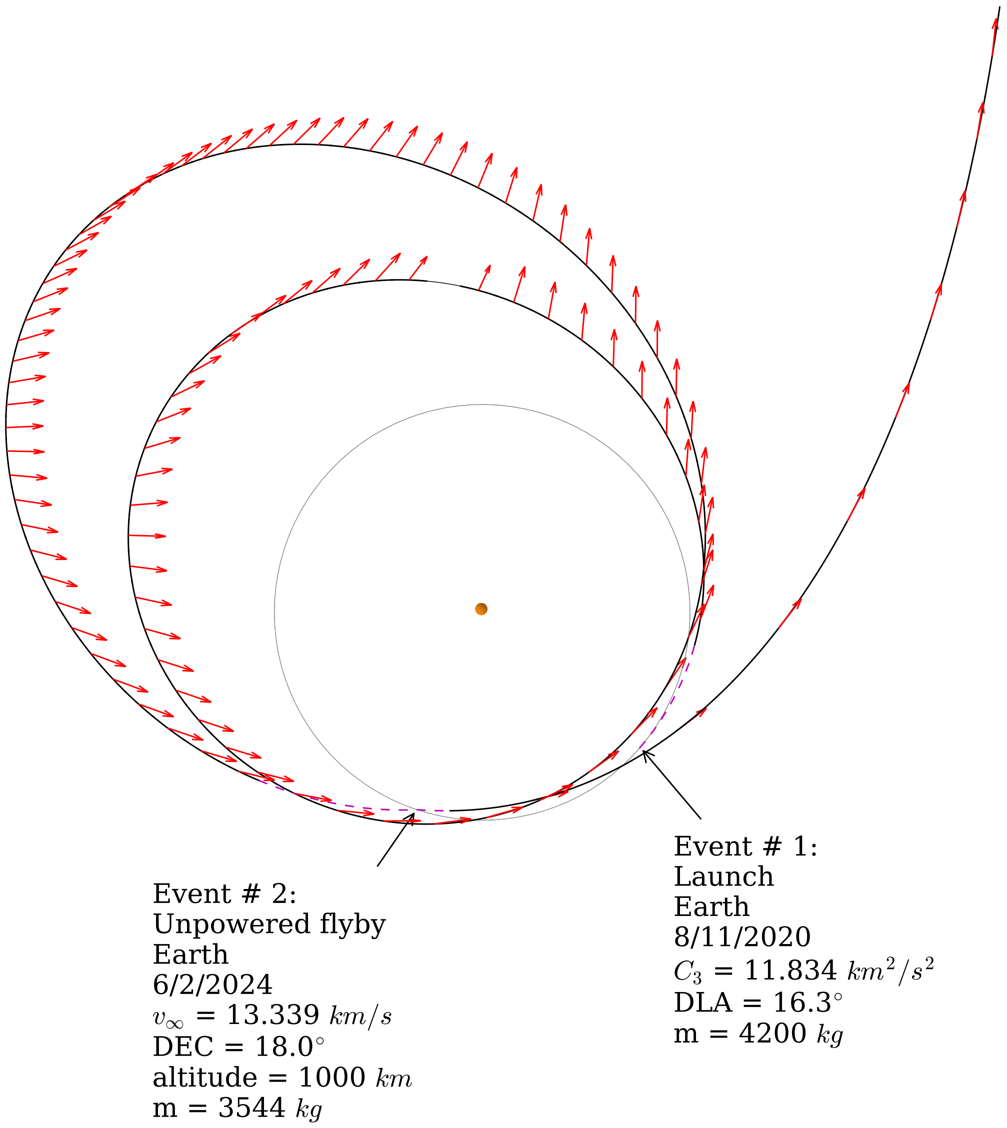


Figure 3.8: Optimal Uranus Probe trajectory.

Figure 3.10 shows the throttle setting used as well as the number of active thrusters throughout the mission. The thruster count transitions were smoothed using the technique described in section 2.4. Figure 3.11 depicts the power generated by the spacecraft's solar array, the power utilized by the thrusters and the thrust achieved. It is important to note that while a bang-bang control structure is exhibited in Figure

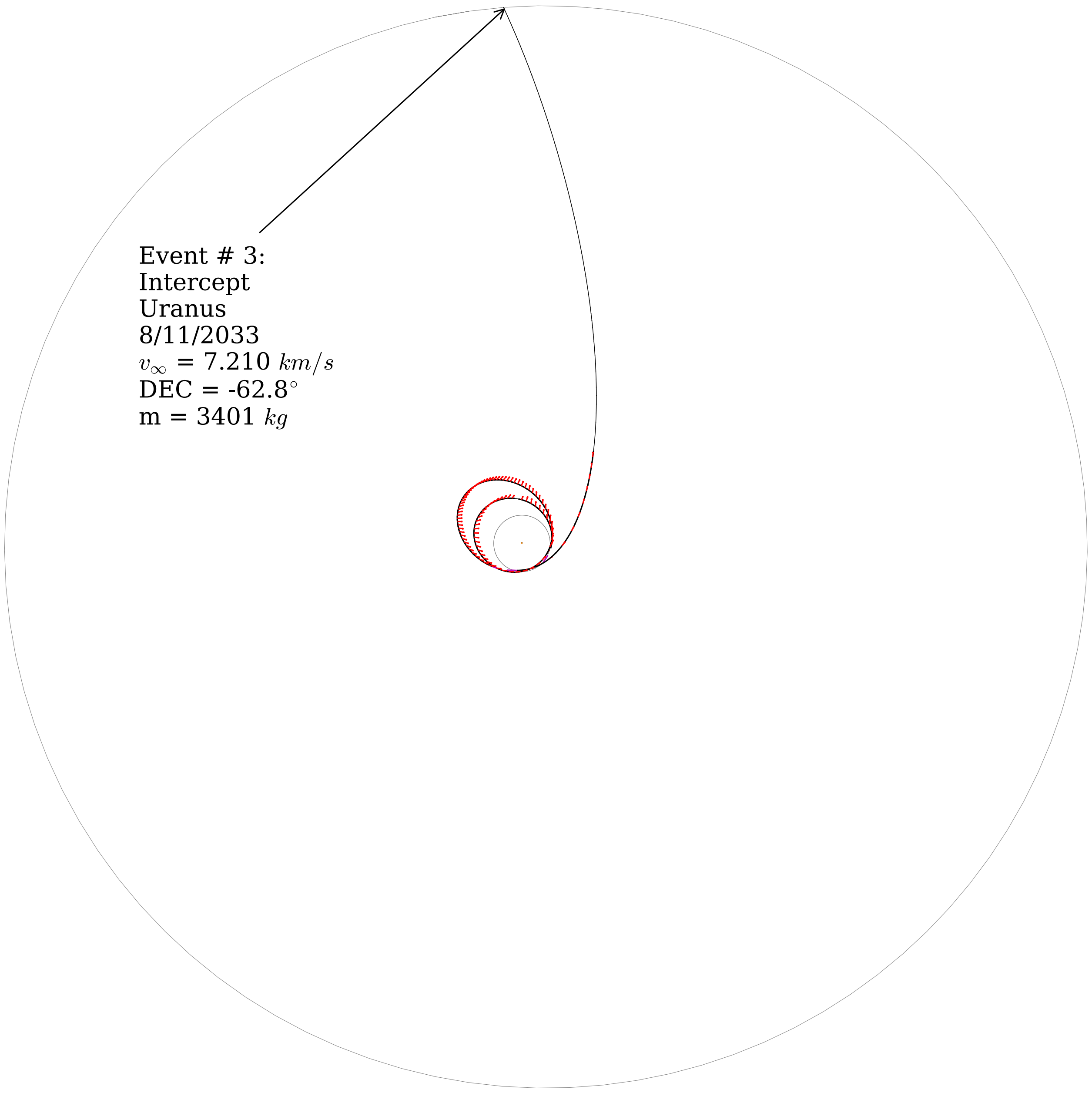


Figure 3.9: Optimal Uranus Probe trajectory.

3.10, the power model does not allow for the thrusters to operate at their maximum input voltage setting at all times. The gap present between the two power curves in Figure 3.11 is due to the power margin and spacecraft bus power requirements.

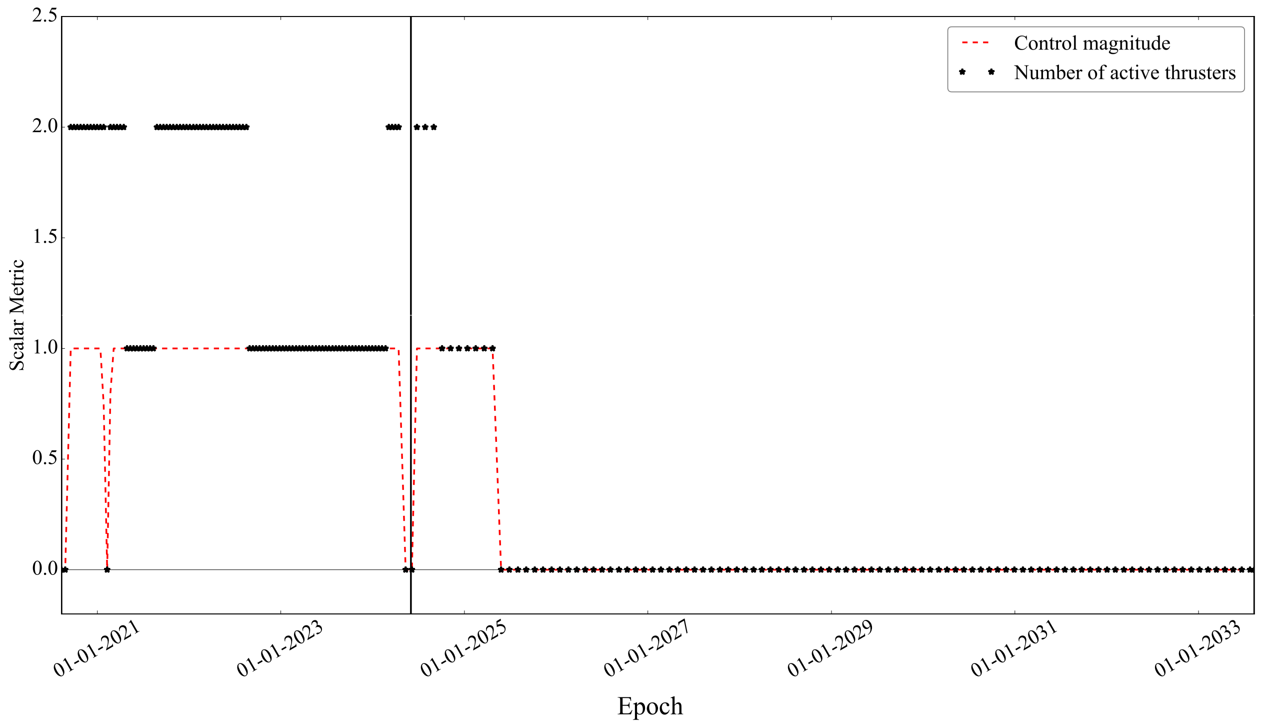


Figure 3.10: Optimal Uranus Probe trajectory throttle and active thruster histories.

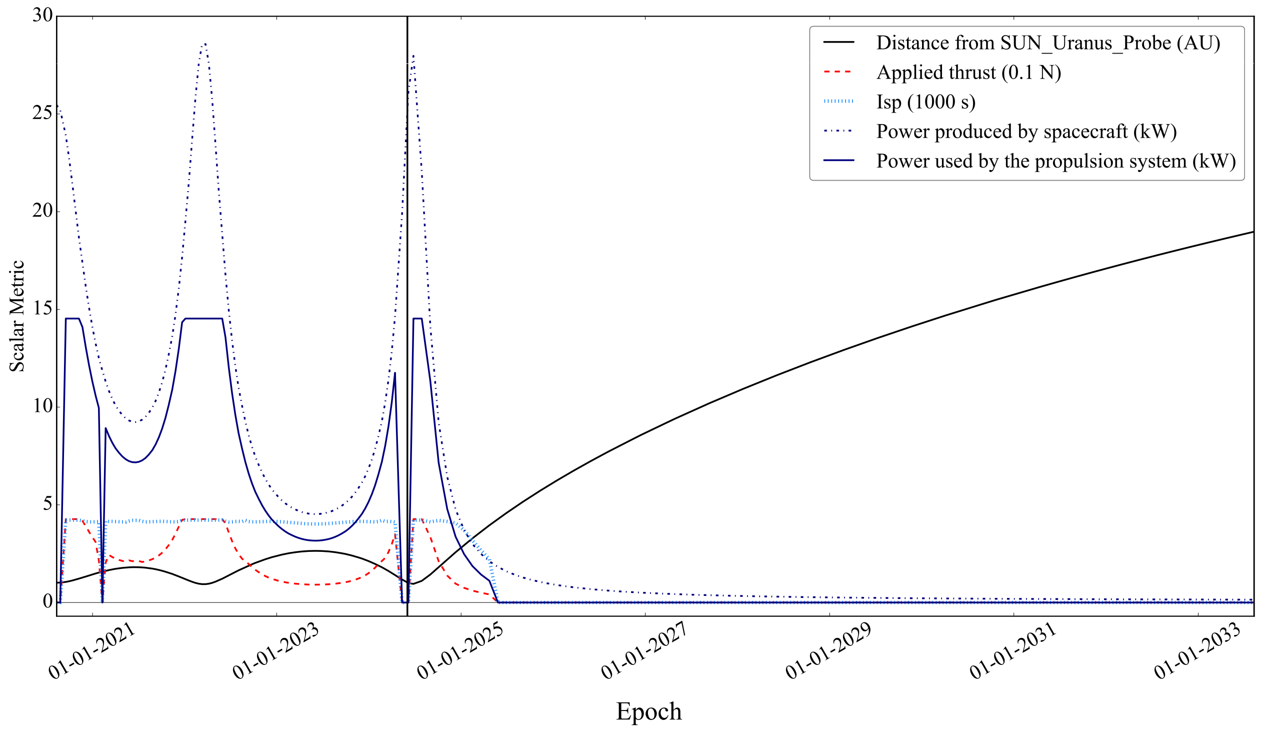


Figure 3.11: Optimal Uranus Probe trajectory thrust and power level histories.

3.8 Example: Mercury Spiral-Down

For the next example, a more thorough investigation of the convergence properties of the NLP solver SNOPT was carried out. The mission considered is a low-thrust rendezvous with Mercury with a single intermediate flyby of Venus, and was previously solved by Sauer [173] and Debban et al. [174]. The objective function to be maximized is the mass delivered to Mercury. The problem was first solved using the MBH heuristic in conjunction with SNOPT, which provided the local search method. The MBH algorithm is described in Appendix A.1. The problem assumptions are provided in Table 3.8 and the locally optimal solution that was found is shown in Figure 3.12 and has a normalized cost function of 0.701429.

Table 3.8: Mercury rendezvous mission parameters.

Parameter	Value
Maximum allowed initial mass	603 kg
Earliest allowed launch date	January 1st, 2002
Maximum flight time	3 years
Launch vehicle	Delta 7326
Launch C3	2.0 km^2/s^2
Launch declination bounds	$[-90.0°, 90.0°]$
Arrival type	rendezvous
Thruster	1 x NSTAR
Thruster duty cycle	1.0
Solar power coefficients	$\gamma_0 = 1.0$ $\gamma_1 = 0.0$
	$\gamma_2 = 0.0$ $\gamma_3 = 0.0$
	$\gamma_4 = 0.0$
Spacecraft bus power requirement coefficients	$a_{s/c} = 0.0$
	$b_{s/c} = 0.0$
	$c_{s/c} = 0.0$
Solar array BOL power (launch at 1 A.U.)	2.6 kW
Flyby sequence	E-V-Y
Minimum earth flyby altitude	300 km
Pre-Venus flyby coast	45 days
Post-Venus flyby coast	15 days
Number of segments per phase	200
Ephemeris	SplineEphem
SNOPT feasibility tolerance	1.0e-5
SNOPT optimality tolerance	1.0e-3
Objective function	maximize final mass
SNOPT max. runtime	1800 s
SNOPT major iteration limit	8000

Once the solution shown in Figure 3.12 was obtained, a series of convergence tests were performed by perturbing the optimal decision vector $\mathbf{x}^*$ with a series of random vectors drawn from a normal distribution $\mathcal{N}$ with a mean μ of zero:

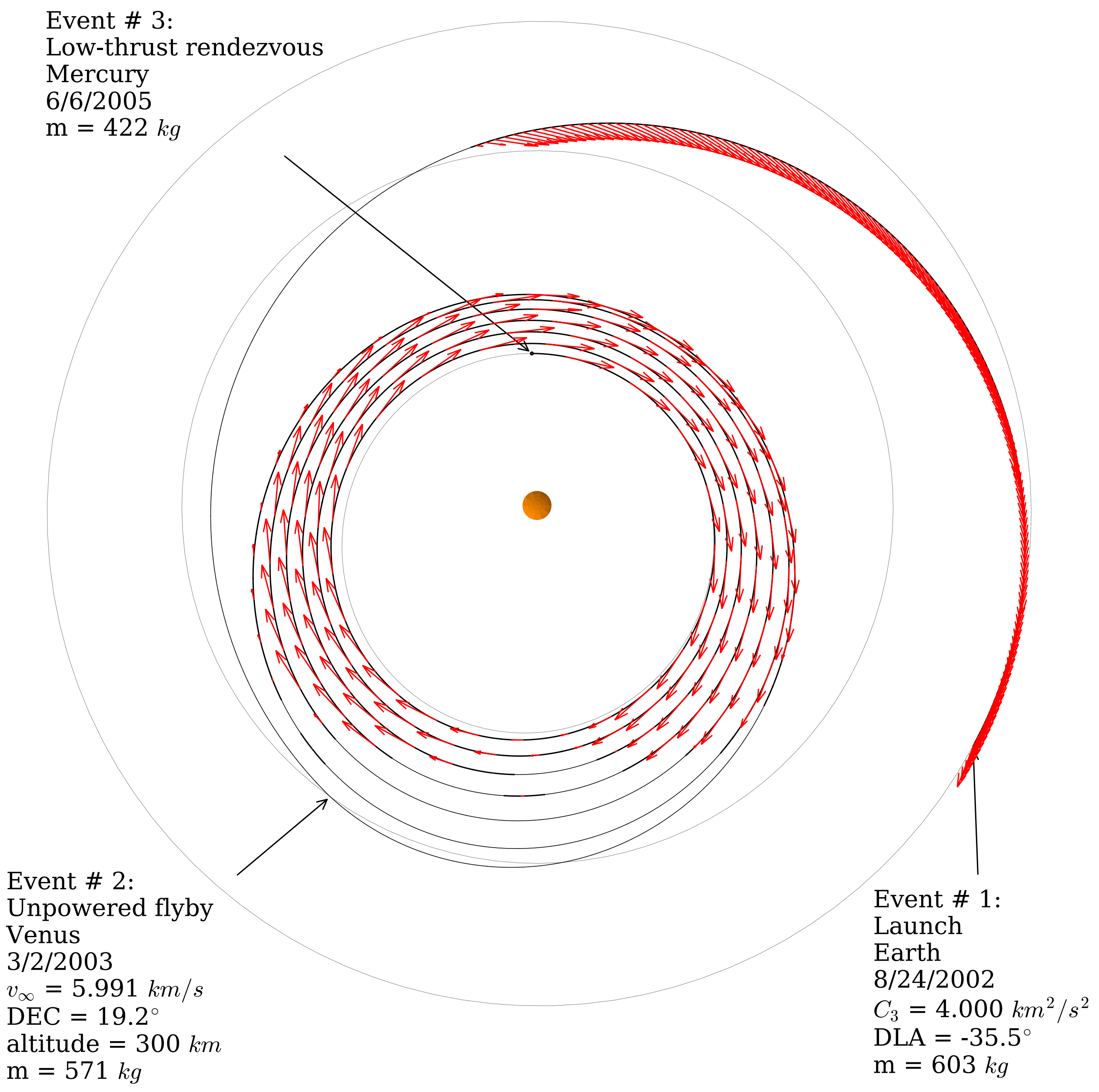


Figure 3.12: Optimal Mercury rendezvous trajectory.

$$x_i = x_i^* + \mathcal{N}(\mu = 0, \sigma_i) \quad x_i \in \mathbf{x} \quad \text{and} \quad \sigma_i \in \mathbb{R} \tag{3.95}$$

The standard deviation used to perturb the i^{th} decision variable (σ_i) was selected differently for each type of variable. A summary of these standard deviations is shown in Table 3.9.

The problem was then re-solved once using an analytic Jacobian and once with the partials provided by finite differences in order to draw a comparison between the two methods. This process was repeated for

Table 3.9: Standard deviations used to perturb each type of decision variable in the Mercury rendezvous problem.

Decision variable	σ_i
launch epoch	1.0 days
launch v_∞	1.0 km/s
right ascension of launch asymptote	0.5 radians
declination of launch asymptote	0.5 radians
Earth-Venus flight time	1.0 days
Earth-Venus terminal $\mathbf{v}_\infty$	1.0 km/s
Earth-Venus final mass	1.0 kg
Venus-Mercury flight time	1.0 days
Venus-Mercury initial $\mathbf{v}_\infty$	1.0 km/s
Venus-Mercury terminal $\mathbf{v}_\infty$	1.0 km/s
Venus-Mercury final mass	1.0 kg
control	0.1

100 trials. In each case, SNOPT was allowed to run for a maximum of 30 minutes. If the solver did not converge, exit stating infeasibility or otherwise terminate prior to the 30 minute limit, its achieved objective value was recorded as zero and its execution time was recorded as 30 minutes. The objective function values achieved in each of these 200 trials are shown in Figure 3.13. The time-to-conclusion for each trial is shown in Figure 3.14.

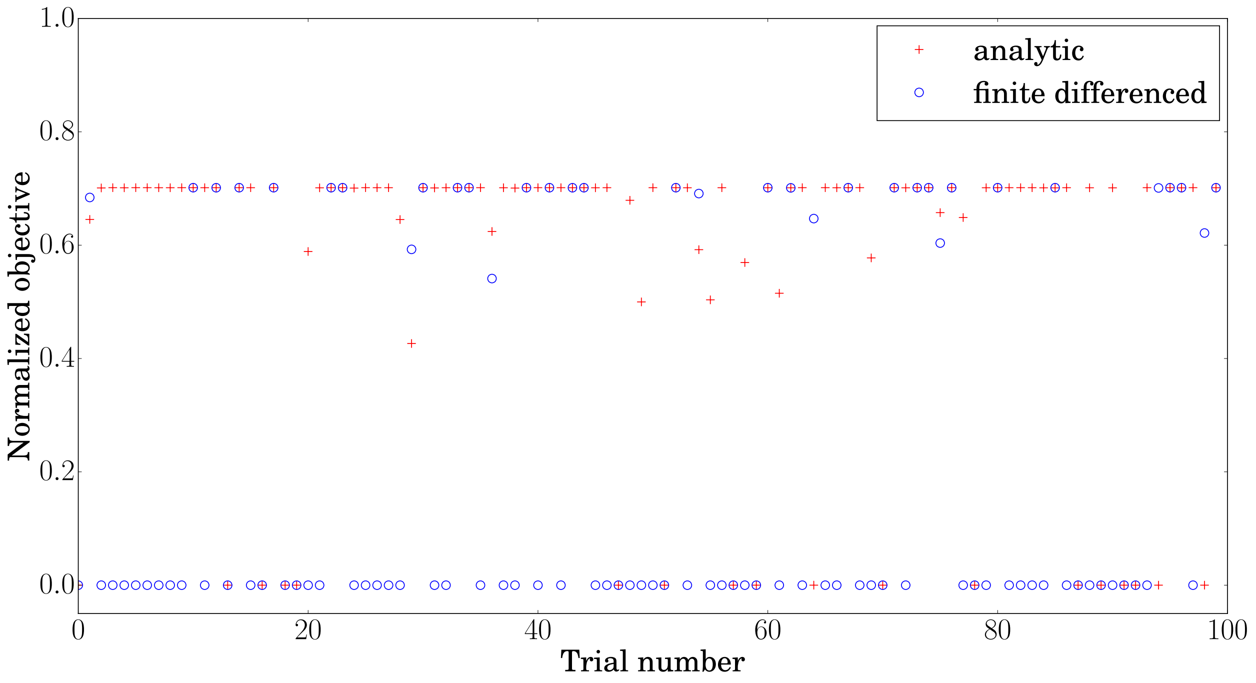


Figure 3.13: Final normalized objective values achieved by SNOPT for 100 perturbed initial conditions around the solution shown in Figure 3.12.

The times-to-conclusion in Figure 3.14 include instances where the solver successfully converged (feasi-

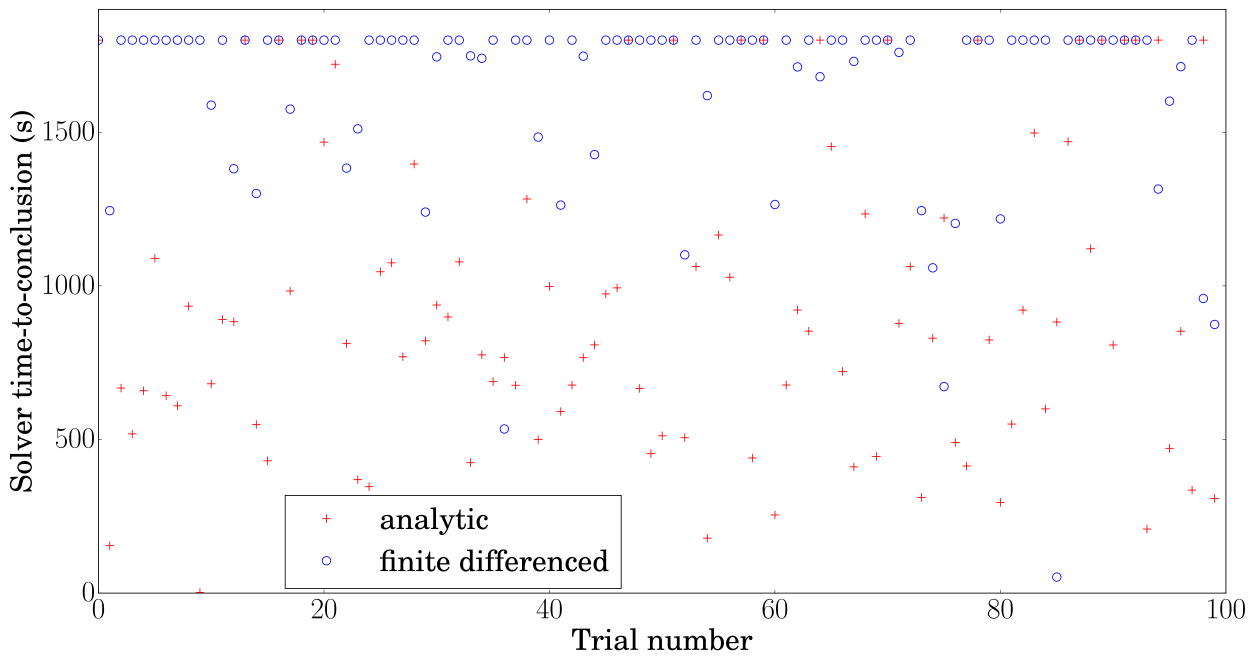


Figure 3.14: SNOPT time-to-conclusion for 100 perturbed initial conditions around the solution shown in Figure 3.12.

bility and optimality tolerances satisfied), partially converged (feasible only), or successfully exited but did not converge (e.g. claimed problem was infeasible, encountered numerical difficulties during linear system decomposition etc.). Given a wall clock time limit of 30 minutes, using analytic derivatives SNOPT successfully reached a conclusion for 82/100 trials. Using finite differences it did so in 27/100 trials. Of these "successful conclusion" cases, 66 converged using an analytic Jacobian and 27 did so using finite differences. These statistics reveal an interesting finding. They suggest that when using analytic derivatives, not only is the solver able to converge more robustly, but also that its SQP algorithm is able to more effectively determine when it will *not* converge.

The next series of experiments that were performed sought to determine which decision variables were the most sensitive to perturbation for this problem, and how effective analytic derivatives are at re-converging a perturbed solution compared with finite differencing. The decision variables were divided into five categories: **time** (launch epoch, E-V flight time, V-Y flight time), **mass** (mass at V, mass at Y), **hyperbolic excess vectors** (launch v_∞, E-V final $\mathbf{v}_\infty$, V-Y initial $\mathbf{v}_\infty$, V-Y final $\mathbf{v}_\infty$), **control** and **launch asymptote orientation** (RA and DEC of launch asymptote). Using the standard deviations in Table 3.9, 20 perturbed initial decision vectors were created for each decision variable category. For example, for the **time** category only the launch epoch and phase flight time variables were perturbed; all other decision variables retained their optimal values. Each perturbed decision vector was used to initialize one SNOPT run once using analytic partials and one using finite differencing. A summary of convergence statistics for these experiments is provided in Table 3.10.

Table 3.10: Convergence rates for each decision variable category.

Decision variable category	Analytic convergence rate	Finite differencing convergence rate
time	19/20	16/20
mass	19/20	12/20
hyperbolic excess	16/20	5/20
control	19/20	11/20
launch RA/DEC	19/20	16/20

The results in Table 3.10 quantify the superiority of employing analytic gradients in the vicinity of an NLP solution. When employing an analytically defined Jacobian, SNOPT converged more frequently for all decision variable types.

3.9 Example: Odysseus Direct

The next mission design application considered is a flight to the Greek camp Jupiter Trojans that orbit the Sun-Jupiter L_4 point. A notional mission to 1184 Odysseus is considered where two proximity constraints are placed on the spacecraft: the spacecraft must not get closer to the Sun than 0.9 AU and it must not be farther than 5.75 AU from Earth at any point during its transfer to Odysseus. The solar proximity constraint is a common constraint that is usually applied to avoid excessive thermal loading of the spacecraft. The latter path constraint could be due to a communication range limitation of a high gain antenna used during the mission's interplanetary cruise. The value of 5.75 AU is somewhat contrived in that it is not based on any actual communication hardware). The mission parameters are provided in Table 3.11.

Four hundred MBH solver instances were run for this problem: 100 trials using the analytic methods introduced in this paper along with SplineEphem, 100 trials using the analytic gradients as well as raw SPICE calls, 100 trials using finite differencing and SplineEphem, and finally 100 trials using finite differencing with direct calls to SPICE. For all search strategies, SNOPT was allowed to iterate for a maximum of 20 minutes, although it typically converged or otherwise terminated before that time limit. The statistics for the 400 trials (100 for each Jacobian calculation strategy) are shown in Table 3.12 and the optimal solution found is shown in Figure 3.15.

The results in Table 3.12 indicate that analytic gradients significantly improve the performance of the solver. The more accurate derivatives increase the robustness of the SQP method, and the step direction

Table 3.11: Odysseus direct mission parameters

Parameter	Value
Maximum allowed initial mass	3500 kg
Earliest allowed launch date	January 1st, 2024
Launch window	365.25 days
Maximum flight time	14 years
Launch vehicle	Atlas V (551) NLS-2
Maximum launch C3	28.0 km^2/s^2
Launch declination bounds	$[-28.5^\circ, 28.5^\circ]$
Arrival type	low-thrust rendezvous
Thruster	2 x NEXT TT11, high thrust
Throttle logic	minimum number of thrusters
Thruster duty cycle	0.9
Solar power coefficients	$\gamma_0 = 1.0$ $\gamma_1 = 0.0$
	$\gamma_2 = 0.0$ $\gamma_3 = 0.0$
	$\gamma_4 = 0.0$
Spacecraft bus power requirement coefficients	$a_{s/c} = 1.0$
	$b_{s/c} = 0.0$
	$c_{s/c} = 0.0$
Solar array BOL power (launch at 1 A.U.)	40.0 kW
Power margin (δ_{power})	15%
Flyby sequence	E-O
Post-launch checkout coast	60 days
Number of segments per phase	200
Ephemeris	SplineEphem
SNOPT feasibility tolerance	1.0e-5
Objective function	maximize final mass
MBH runtime	8 hrs.
SNOPT max. runtime	20 min
SNOPT major iteration limit	8000

Table 3.12: Solution statistics for Odysseus Direct 8-hour MBH runs.

Metric	Analytic + SplineEphem	Analytic + SPICE	FD + SplineEphem	FD + SPICE
Best mass delivered (kg)	2304	2297	2287	2228
Mean # feasible solutions	300	283	338	318
Mean-max mass delivered (kg)	1015	631	862	450
Mean-mean mass delivered (kg)	778	316	503	381
Success rate (%)	78	63	56	51
Mean time to best solution	5h 21 min.	5 h 25 min.	5 h 30 min.	5 h 47 min.

determined by the minor level iterations is more likely to be accepted by the major level. The success percentage indicates the number of MBH+NLP solver runs that found at least one feasible solution in the eight hour search period. The runs that featured analytic gradients, in general, had a higher success rate than the finite difference trials. The switch from SplineEphem to SPICE had a negative impact on the performance of the optimizer. In general, the cost function values significantly decreased and the ability of the solver to find feasible solutions was hindered.

The same mission was also optimized without imposing the distance constraint. The optimal unconstrained trajectory is shown in Figure 3.17. Without the requirement that it remain within 5.75 AU of Earth, the spacecraft passes well beyond the 5.75 AU limit prior to rendezvous with the asteroid, as indicated by the horizontal red dashed line in Fig. 3.18. For this particular problem, the unconstrained and constrained problems are in the same family, and it can be concluded *a posteriori* that the proximity constraint could also have been enforced indirectly by constraining the arrival date at Odysseus.

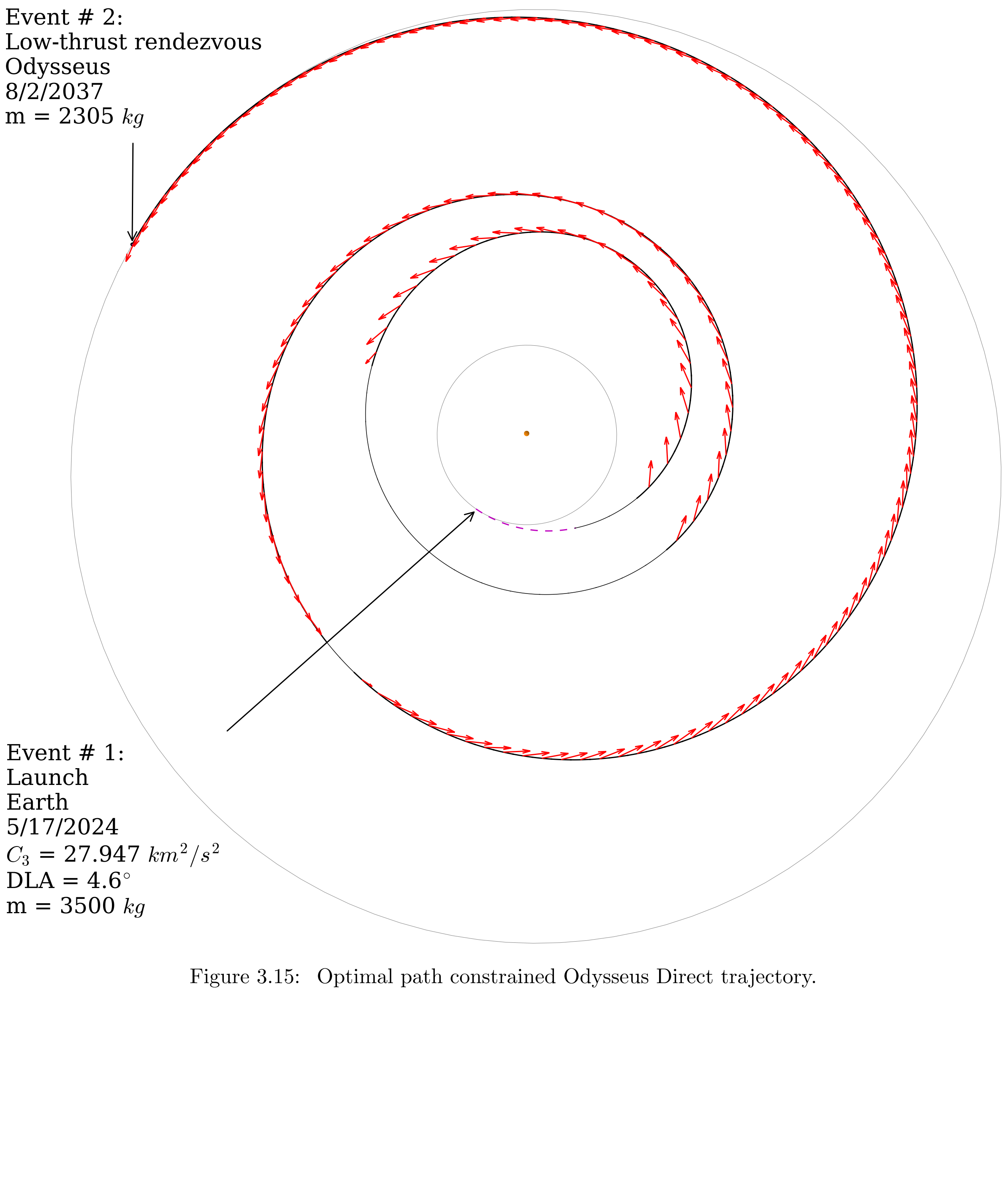


Figure 3.15: Optimal path constrained Odysseus Direct trajectory.

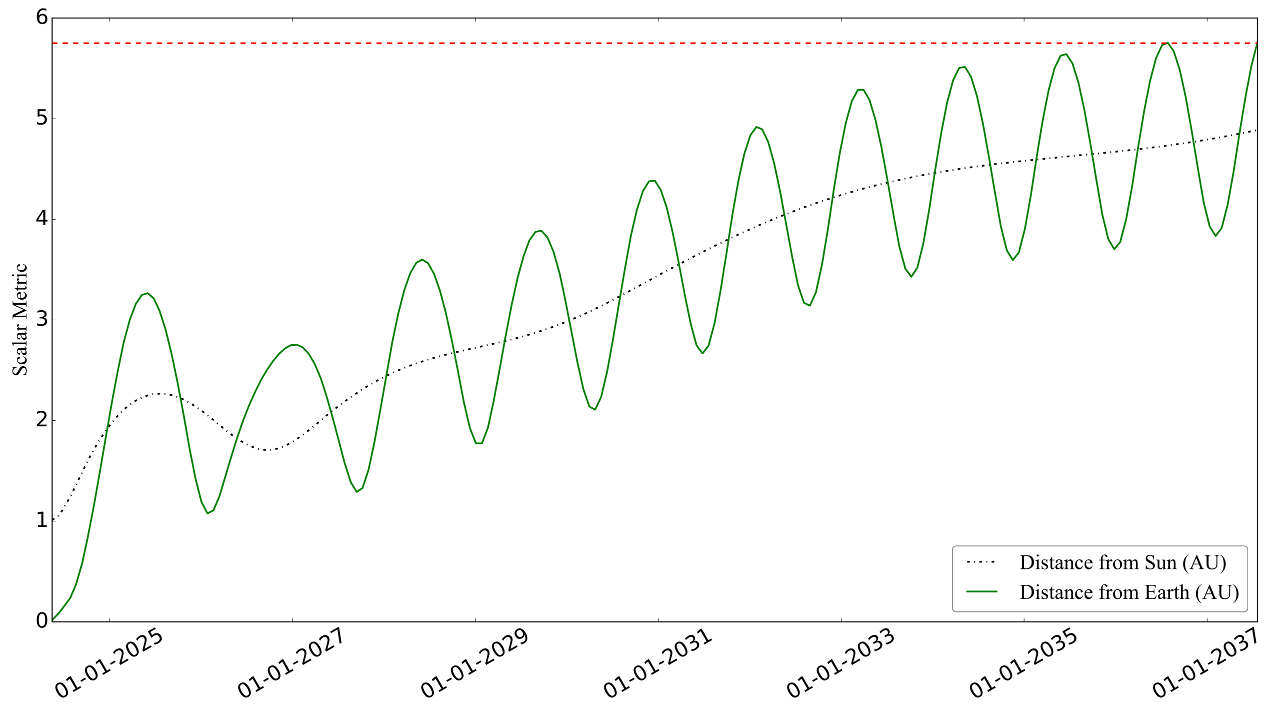


Figure 3.16: Constrained Odysseus Direct distance from the Earth and Sun as a function of time. The horizontal line placed at 5.75 A.U. shows that the proximity constraint was successfully enforced.

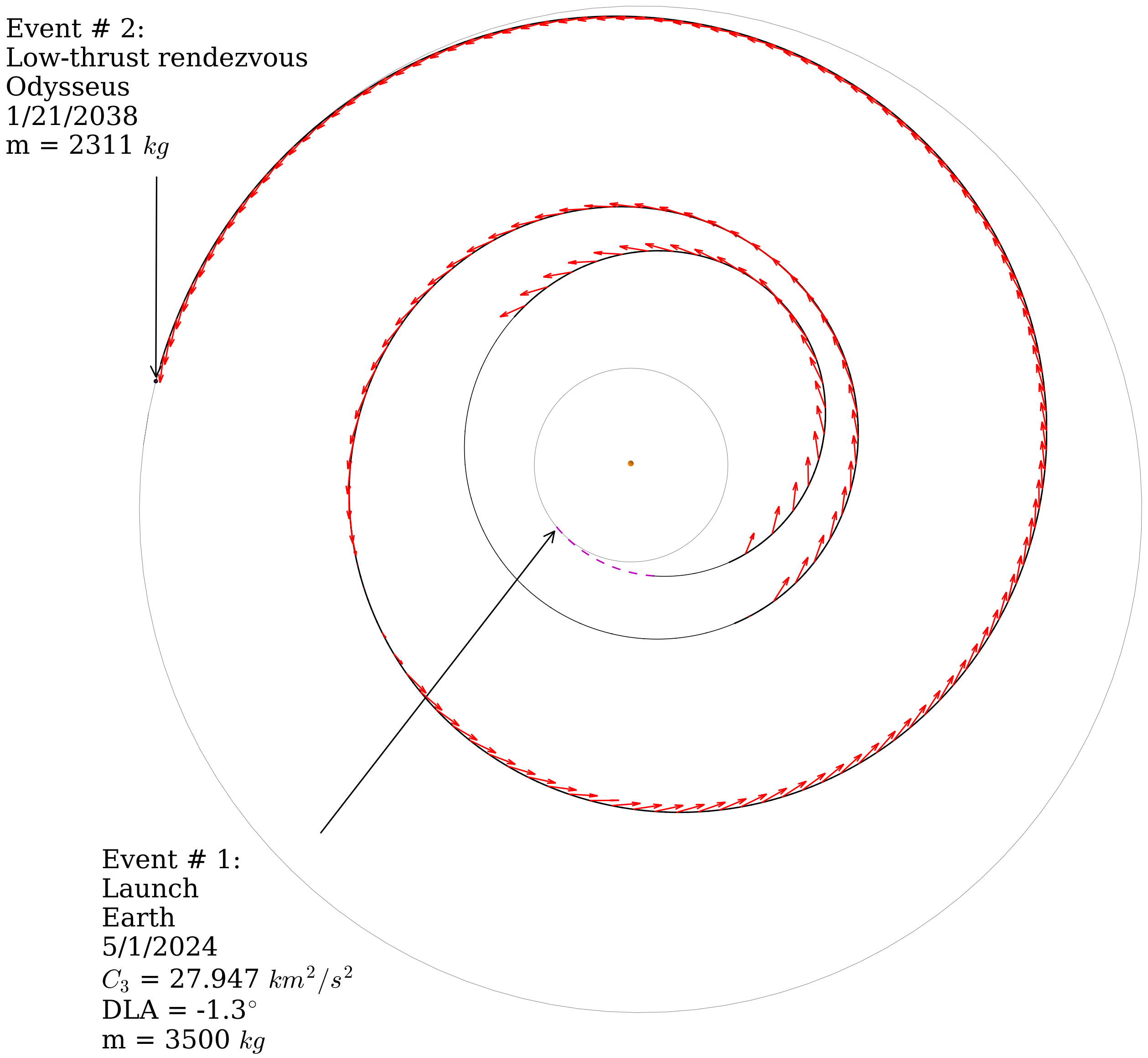


Figure 3.17: Optimal Odysseus Direct trajectory without proximity constraint.

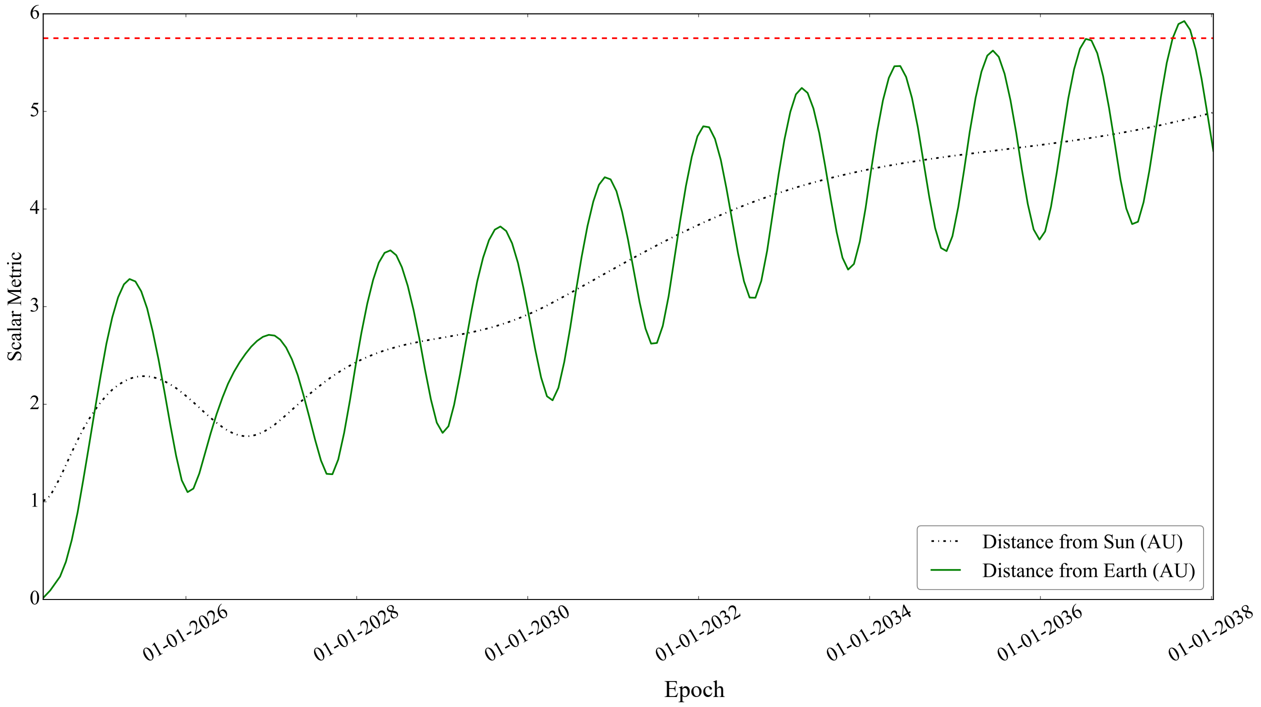


Figure 3.18: Unconstrained Odysseus Direct distance from the Earth and Sun as a function of time.

3.10 Example: Cassini

The following example demonstrates the relative performance of analytic derivatives and finite differencing for the MGAnDSMs transcription. The Cassini problem solved here uses the same flyby sequence that the actual orbiter used on its way to Saturn (E-VVEJ-S). Due to the unavailability of a Titan IV (401) B launch vehicle model, it was assumed for this example that the spacecraft was launched using an Atlas V (551). The solver was allowed to place a single deep space maneuver at any point along each phase. An orbital insertion maneuver was included in the total mission Δv and thus influenced the final delivered mass. The mission parameters and assumptions are described in Table 3.13.

Table 3.13: Cassini mission parameters.

Parameter	Value
Earliest allowed launch date	January 1st, 1997
Saturn arrival date	unbounded
Launch window	365.25 days
Maximum flight time	7 years
Maximum allowed initial mass	10000 kg
Launch vehicle	Atlas V (551) NLS-2
Maximum launch C3	18.069 km^2/s^2
Launch declination bounds	$[-28.5°, 28.5°]$
Thruster I_{sp}	312 s
Flyby sequence	E-VVEJ-S
Ephemeris model	SPICE
Arrival type	orbital insertion
Insertion semimajor axis	5 447 500 km
Insertion eccentricity	0.98
SNOPT feasibility tolerance	1.0e-5
Max SNOPT runtime	5 s
Objective function	maximize $\log_{10}$ final mass

One hundred instances of the MBH solver were run for four hours. No initial guess was provided. All MBH runs converged to the known optimal solution. The best known optimal solution found delivers 3195 kg to orbit around Saturn, and requires only a single 0.428 km/s maneuver at aphelion between the two Venus flybys and a 0.622 km/s orbital insertion maneuver, totaling 1.05 km/s. This trajectory is shown in Figures 3.19 and 3.20.

Table 3.14: Solution statistics for 100 4-hour MBH runs.

Metric	Analytic	Finite Differences
Best mass delivered (kg)	3195	3195
Mean number of MBH hops taken	1432	591
Mean time to best solution	1 hr. 48 min.	2 hr. 10 min.

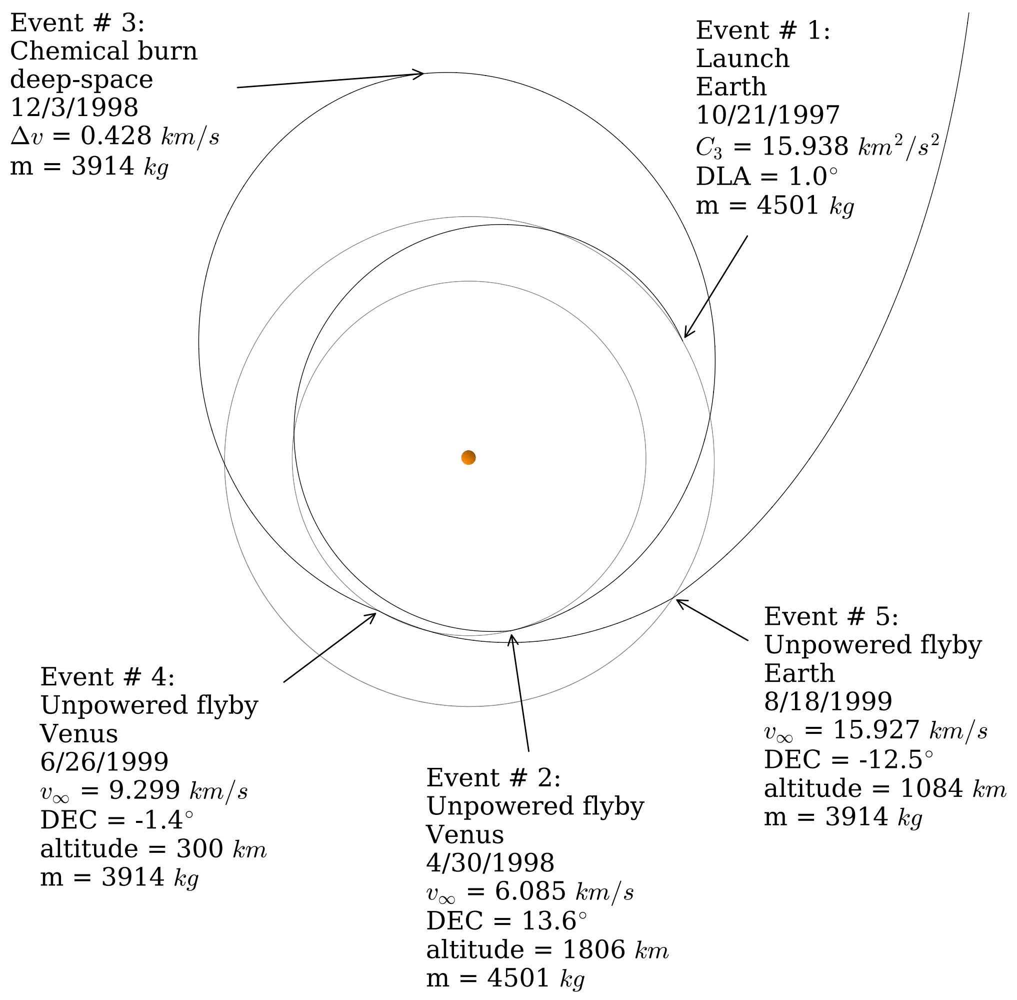


Figure 3.19: Optimal Cassini trajectory.

The relative performance between the runs using analytic derivatives and the those using finite differencing is more comparable for this transcription, as shown in Table 3.14, in part due to the fact that the Cassini problem is not particularly challenging for the MBH + SNOPT solver. As a result, both methods display considerable robustness, with none of the runs failing to identify the known optimal solution. The analytic derivatives arrived at their best found solution faster than the runs using finite differencing. This is due to the increased execution speed that employing analytic partials allows for in addition to the robustness benefits discussed in section 3.8.

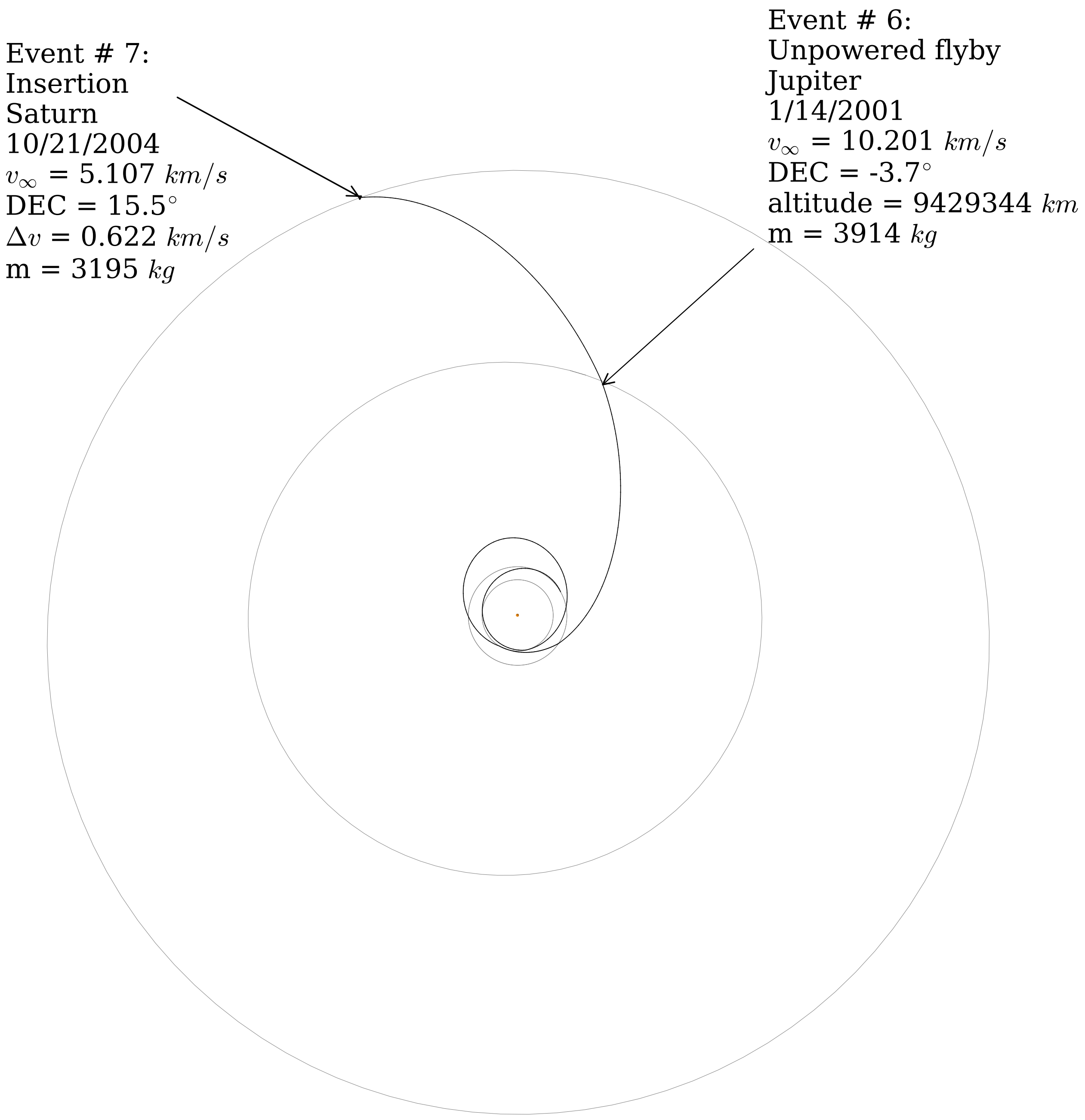


Figure 3.20: Optimal Cassini trajectory.

3.11 Example: Comet Sample Return Mission

The benefits of implementing the analytic methods presented in this work are demonstrated by solving a relatively complex interplanetary mission design problem with the goal of recovering a surface sample from the comet 67P/ChuryumovGerasimenko. Comet Surface Sample Return (CSSR) has been of ongoing interest to the planetary science community [172]. Prior investigation has demonstrated that comet rendezvous, and in particular roundtrip missions, are enabled by solar electric propulsion [175–177]. For a more recent and relevant mission example, the authors selected the New Frontiers 4 comet sample return concept proposed by Choukroun et al. [178].

The solution technique introduced by Yam et al., which combines the metaheuristic monotonic basin hopping (MBH) with a gradient-based local optimization step, is used to solve this problem. The local optimizer selected for this example is SNOPT. The MBH solution method begins by initializing each decision variable randomly within within some bounded interval. Then, the NLP is optimized until the problem is determined to be locally optimal, or the solver terminates due it exceeding user-specified resource limitations (e.g. run time, maximum iterations etc.) or some other solver exit condition. The decision vector is then perturbed with new values being selected from a statistical distribution before the local solver is executed again. The process repeats until a stagnation condition is met or a specified time limit is exceeded. For further details on the MBH algorithm, the reader is directed to the initial study by Yam et al. or the more recent work by Englander et al. [46], Englander and Englander [112] and Englander and Conway [47].

The parameters for the example comet sample return problem are summarized in Table 3.15. The launch vehicle performance data for the Atlas V (431) was obtained from the NASA Launch Services Program Launch Vehicle Performance website [179]. The paper by Choukroun et al. did not specify which thrusters were used in their design, however, the paper does mention that the mission was designed around a Lockheed-Martin A2100 bus. The A2100 is a commercial satellite bus that flies XR-5 hall current thrusters [180]. Performance models for the BPT-4000 (now designated the XR-5) are from the experimental study conducted by Hofer [181]. For this problem, the BPT-4000 Ext-high-$\mathrm{I_{sp}}$ throttle curve was used. The paper by Choukroun [178] also does not specify any information about the solar array hardware used, so for this problem all γ_i in the solar array model are set to zero except $\gamma_0 = 1$, corresponding to a $1/r^2$ electric power model. Additional details about the solar electric power modeling, including multiple thruster switching conditions are provided by Englander and Conway [47].

In order to quantify the solver's speed and robustness improvements, the comet sample return problem

Table 3.15: Parameters for the sample return mission to comet 67/P.

Parameter	Value
Maximum allowed initial mass	10000 kg
Earliest allowed launch date	June 16^{th}, 2024
Launch window	1 day
67/P arrival date bounds	April 1^{st} 2029 - April 30^{th} 2029
Fixed 67/P departure date	December 8^{th} 2033
Earth return date bounds	November 8^{th} 2034 - November 9^{th} 2036
Maximum flight time	12.5 years
Launch vehicle	Atlas V (431) NLSII
Maximum launch C3	25.0 km^2/s^2
Launch declination bounds	$[-28.5°, 28.5°]$
Arrival type	low-thrust rendezvous
Thruster	3 x BPT-4000, extra high I_{sp}
Throttle logic	minimum number of thrusters
Thruster duty cycle	0.9
Solar power coefficients	$\gamma_0 = 1.0$ $\gamma_1 = 0.0$
	$\gamma_2 = 0.0$ $\gamma_3 = 0.0$
	$\gamma_4 = 0.0$
Spacecraft bus power requirement coefficients	$a_{s/c} = 1.0$
	$b_{s/c} = 0.0$
	$c_{s/c} = 0.0$
Solar array BOL power (launch at 1 A.U.)	30.0 kW
Flyby sequence	E-67/P-E
Forced post-Launch checkout coast	60 days
Number of segments per phase	80
SNOPT feasibility tolerance	1.0e-5
Objective function	maximize returned mass to Earth
MBH runtime	2 hrs.
SNOPT max. runtime	120 seconds
SNOPT major iteration limit	8000

was solved 100 times with SNOPT being provided a Jacobian calculated using the methods presented in this work and 100 times where it was provided with a Jacobian computed with a central differencing routine. Both the analytic and finite differenced runs were provided with the exact Jacobian sparsity pattern. The results of the runs using analytic and finite differenced derivatives are compared in Table 3.16. The best locally optimal solution found is shown in Fig. 3.21. None of the runs using finite differencing identified any feasible solutions. Furthermore, even when the solver employing finite differencing to compute its Jacobian is initialized with the optimal answer identified by the solver using analytic partial derivatives, the former identified the feasibility of the seeded solution and then diverged almost immediately after it began iterating. By contrast, out of 100 trials where our analytic techniques were used to compute the Jacobian, 94 identified a feasible solution to the problem and 60 cases identified the best known cost function value to within 1 kg. Note that the best known value cost function value is the one reported in Fig. 3.21 as no other literature source has been found that has solved this problem. The average MBH+NLP run identified 55 feasible solutions in its two hour search and on average identified its best solution after 16.6 minutes. The comet sample return problem is not only an example of how providing analytic partial derivatives to the MBH+NLP solver is beneficial, it is an example of how doing so enables the solver.

Information about the state of the solar electric power system, as well as other pertinent spacecraft information is shown in Fig. 3.22. These details reveal how the methods developed in this work allow for the use

Table 3.16: Solution statistics for 100 2-hour MBH+NLP runs.

Metric	Analytic	Finite differenced
Best mass delivered (kg)	2104	-
Mean # feasible solutions	55	-
Mean-max mass delivered (kg)	1839	-
Mean-mean mass delivered (kg)	1440	-
Success rate (%)	94	-
Mean time to best solution (min.)	16.6	-

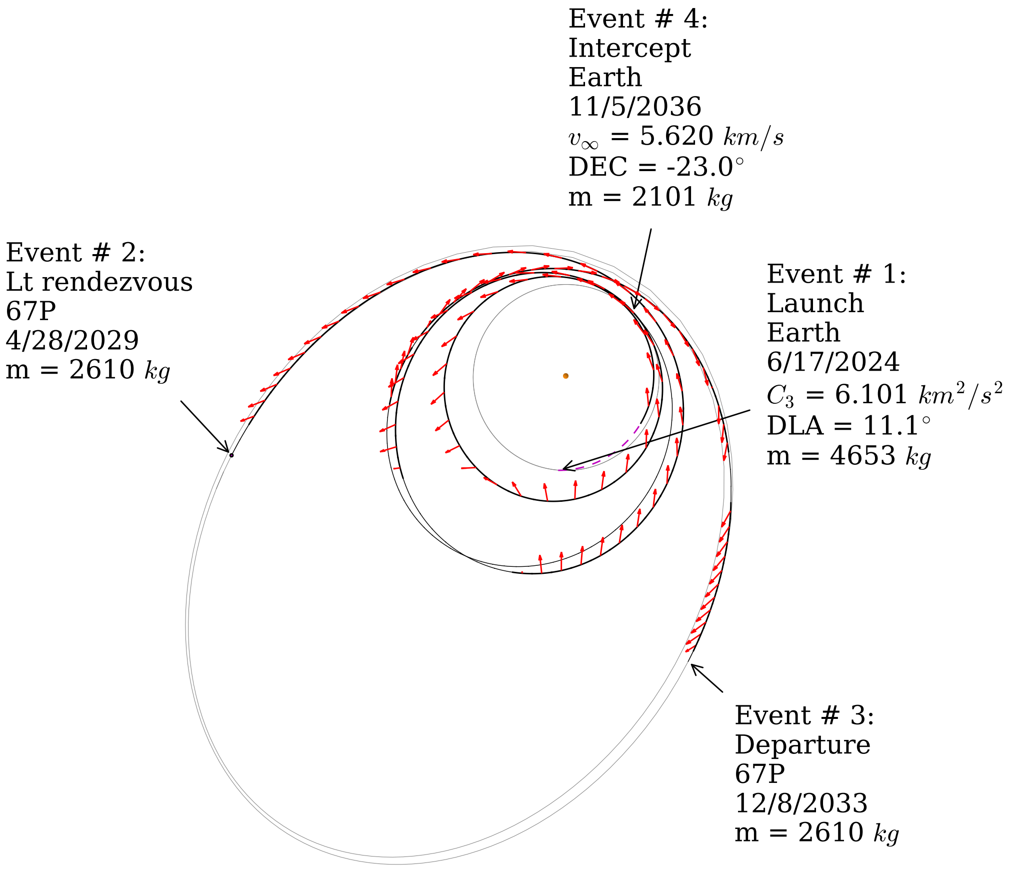


Figure 3.21: Sample return trajectory to comet 67/P.

of a complex power system model, which is necessary for most mission proposal design efforts.

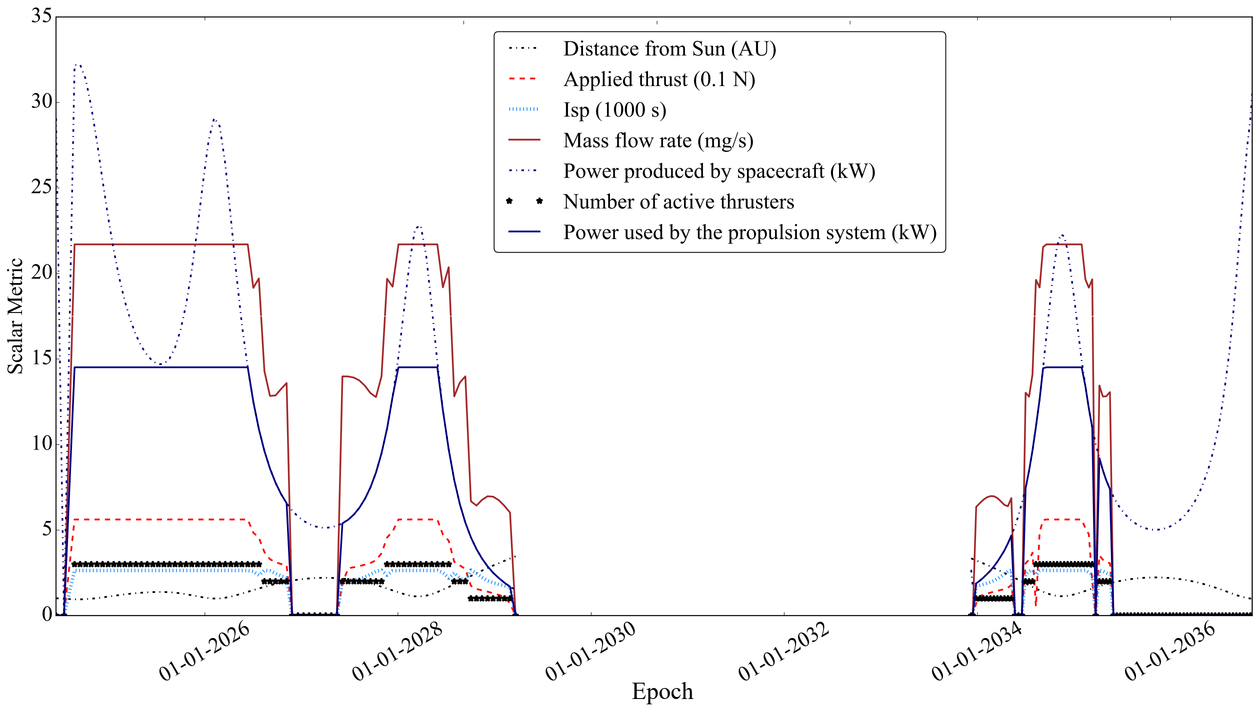


Figure 3.22: Spacecraft metrics for the 67/P sample return mission.

3.12 Computational Efficiency and Accuracy

The analytic techniques described in this work are not the only methods available for computing the constraint partials of a nonlinear program, as was discussed in section 2.1.1. This section will examine the computational efficiency and accuracy of the partial derivative calculation methods described in this paper compared with the same derivatives obtained using central differencing as well as ones calculated using the AD library developed by Ghosh [170]. The AD software package uses C++ operator overloading in a tapeless forward-mode configuration.

The relative computational efficiencies of the analytic techniques presented in this paper, AD and central differencing are quantified by examining the average time required to compute the Jacobian for the comet sample return problem described in section 3.11. The Jacobian matrix for the example problem was computed 1000 times using each method, and the average required calculation time ratios of central differencing and AD to analytic are summarized in Fig. 3.23. As with the example problem solved in the previous section, the Jacobian sparsity pattern was provided to the solver when using the analytic formulae or finite differencing. The automatic differentiation library used also uses a sparse formulation that avoids the calculation of dense zero entries in the Jacobian.

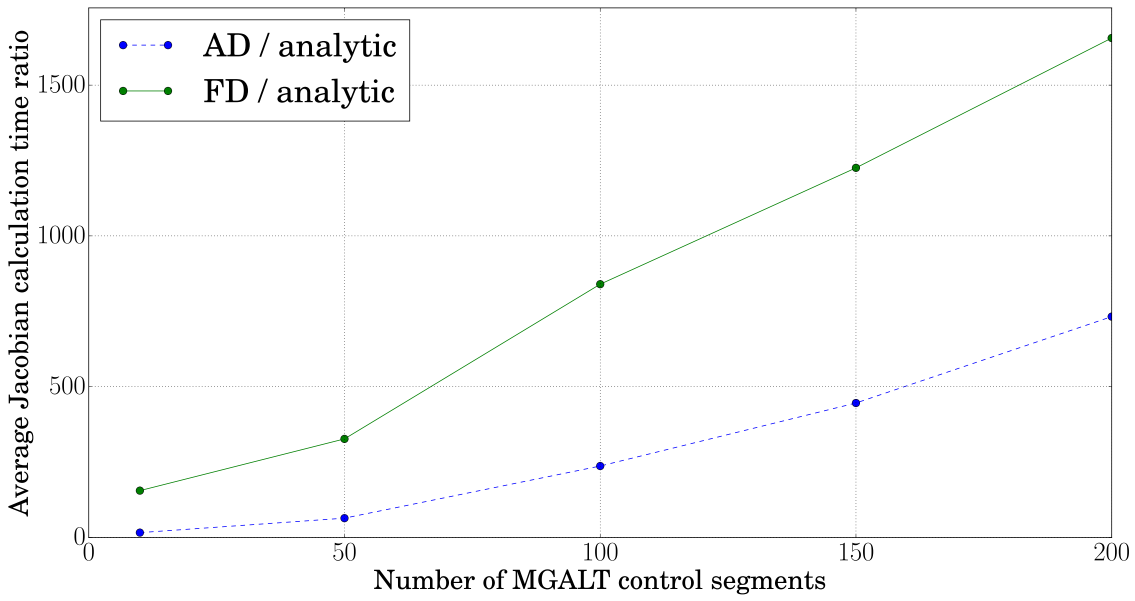


Figure 3.23: Run time ratios for computing the Jacobian of the comet sample return example problem using AD and central differencing as a function of the number of control segments used.

It is important to note that while the central differencing scheme employed in this work avoids dense zero calculations, it does not exploit the specific sparsity pattern of the MGALT Jacobian. That is to say, the example problem presented is comprised of two MGALT phases, and a large majority of each phase's decision variables are decoupled from the other phase. A "smart" finite differencing routine would exploit this problem structure and see an execution speed increase. For this problem, one would expect to see a reduction in Jacobian computation time of about half since every time a cost function call is made to compute a Jacobian entry associated with phase 1, a corresponding entry associated with phase 2 may also be computed. Furthermore, other finite differencing algorithms might be considered, and a forward or backward differencing routine would also lead to an execution speed increase, at the cost of derivative accuracy.

Finite differencing not only suffers from an execution speed perspective, it is also the least accurate of the three methods discussed in this section. A central differencing routine can be expected to produce gradient information that has a relative error of usually no better than 1.0e-08, although higher order algorithms have been shown to exceed the accuracy of central differencing [136]. Analytic gradients by contrast are generally as accurate as those produced by AD or complex step differentiation. The primary disadvantages of analytic calculations, as noted by Pellegrini and Russell [136], is the complexity of their implementation, and the potential unavailability of gradient information from third party libraries, a problem which is only overcome by finite differences or a custom re-implementation of the desired library.

Chapter 4

A Time-Regularized Bounded-Impulse Trajectory Transcription

The most recent Decadal Survey published by the National Research Council in 2013 included Comet Surface Sample Return as the first of five possible mission variants that the New Frontiers 4 selection should be made from. Preliminary mission design efforts that supported concept studies that targeted comets inevitably encountered a fundamental shortcoming of the MGALT transcription (if the mission was low-thrust). The equal temporal spacing of the control nodes in the MGALT model results in the clustering of nodes at apoapsis, leaving few nodes at periapsis. The faster orbital velocities near periapsis require more control nodes in order to more accurately capture the rapidly changing dynamics there. An analogous effect can be seen by employing an adaptive-step integrator to propagate a highly eccentric orbit. The adaptive-step algorithm will lengthen the step size at apoapsis in order to reduce computational time, but reduce it at periapsis where error control necessitates a smaller time step.

In this chapter, a new time-regularized low-thrust transcription is introduced based on the MGALT model. The new transcription retains the ability to use Keplerian propagation arcs, and does not rely on a series solution to do so. Furthermore, due to the way in which the Keplerian propagator is employed to translate angular propagation arcs into flight times, the state propagation aspect of the algorithm is no longer iterative in the sense that a traditional Kepler solver is. These improvements are made at the cost of having to introduce only one decision variable and constraint per trajectory phase. A motivating notional comet rendezvous problem is solved that illustrates the benefits (i.e. computational savings) of adopting the new transcription in favor of the original Sims-Flanagan based models.

4.1 Regularization of the Bounded-Impulse Model

Regularization, as it pertains to celestial mechanics and astrodynamics, involves using a mathematical transformation to express the equations of motion in a more convenient form [182]. The concept of regularization was born from the studies pertaining to the three body problem by Tullio Levi-Cevita [183] and the Danish

German astronomer Peter Hansen. The primary issue that regularization transformations seek to mitigate is the singularity that arises in the n-body equations of motion :

$$\ddot{\mathbf{r}}_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} G m_j \frac{(\mathbf{r}_j - \mathbf{r}_i)}{\|\mathbf{r}_j - \mathbf{r}_i\|^3} \tag{4.1}$$

when two particles collide.

The problem of the n-body singularity came to light in a practical sense with the advent of Cowell's method of perturbations, where the equations of motion are integrated directly with quadratures [160]. This method was first used by its inventor Phillip Herbert Cowell, along with Andrew Claude de la Cherois Crommelin to predict the return of comet 1P/Halley in 1910 [184]. The ability of regularization techniques to recast the equations of motion in a more favorable form again became of interest in the latter half of the twentieth century when digital computers became capable enough to employ Cowell's method for day-to-day astrodynamics analysis.

4.1.1 The Sundman Transformation

Karl F. Sundman was a Finnish astronomer who worked on the three-body problem, and developed the following regularization transformation:

$$dt = r ds, \tag{4.2}$$

where s replaces time t as the independent variable of integration. The new variable is sometimes referred to as "ficticious time", or "s-time", with the corresponding integration referred to as "s-domain" integration. In many applications, this transformation is expressed in terms of the universal anomaly as follows:

$$dt = \frac{r}{\sqrt{\mu}} d\chi \tag{4.3}$$

While the Sundman transformation does not altogether remove the singularity in Eq. (4.1), as Roa points out it does reduce the power in the denominator from three to one [182]. More importantly, the transformation rescales the integration step size around an orbit such that smaller steps are taken near periapsis. By contrast, when the orbital equations of motion are integrated with respect to time, discrete points on the orbit cluster at apoapsis, and are sparser at periapsis.

4.1.2 Regularization of the Bounded-Impulse Model

While bounded-impulse models are capable of accurately predicting mass budgets for many types of low-thrust trajectories, certain classes of missions, such as those requiring multiple-revolutions of the central body with no intermediate flybys as well as those requiring large eccentricity changes, present more of a challenge. The more revolutions that the spacecraft requires to complete a transfer, the greater the number of control nodes that are required to ensure that the bounded-impulse trajectory will be suitable as a seed for a finite-burn model. For trajectories requiring large eccentricity changes the original Sims-Flanagan model, which evenly spaces control nodes in time, places few control nodes near periapse, where energy change is most efficiently carried out. These problems compound one another, and when using the MGALT model, the only way to mitigate this shortcoming is to add more control points, which increases the computational complexity of the problem. For this reason, it is desirable to find a method for redistributing the control nodes for these higher eccentricity trajectories, thus eliminating the need to increase the number of segments.

Redistribution of the control nodes along a trajectory can be achieved using a time-regularization technique such as a Sundman transformation. Pellegrini et al. [185] recently applied generalized Sundman transformations to the Stark and Kepler (sub-case of Stark) problems, but transcription regularization has been of interest for some time, especially concerning propagation of highly-elliptic trajectories using numerical integration [186, 187]. The Stark propagation method explored by Pellegrini et al. is based on a series solution, where the length of the series determines the accuracy of the solution. The Stark model was also extended for use with a Taylor integrator similar to the work by Yam et al. [17]. In this chapter, an exact method for solving the time-regularized Kepler problem is introduced that does not rely on a series solution. Solving Kepler's equation typically requires iterative techniques such as Newton's method, the Laguerre-Conway method or bisection. Instead, we introduce a strategy that transforms a universal Kepler propagator into an iteration-free method at the expense of the introduction of one additional decision variable and constraint per trajectory phase to be selected and enforced respectively by an NLP optimizer. In addition, two different control node distribution methods will be discussed, one that aligns the nodes with the universal anomaly and one that does so with the true anomaly.

4.2 Multiple Gravity-Assist Low-Thrust Using a Sundman Transformation

In the original MGALT model, a numerical optimizer selects the phase flight time variable (Δt_{p}), and a Kepler solver then iteratively determines the corresponding change in eccentric anomaly (ΔE_{p}) by solving Kepler's equation. Analogously, a Kepler solver using a universal formulation determines the change in generalized anomaly ($\Delta \chi_{\mathrm{p}}$) given a particular flight time. The generalized anomaly arises from applying a Sundman transformation to the Kepler solver's independent time variable (Eq. (4.3)).

For the original MGALT transcription, the propagation time input to the Kepler propagator Δt is simply determined by dividing the phase flight time by N, the number of discretization segments. The goal of this new algorithm is to instead place the control nodes at equal intervals of anomaly around the orbit, instead of positions separated by equal propagation times. The optimizer will now select a total propagation anomaly variable for the phase $\Delta \chi_{\mathrm{p}}$, that will be divided into N equal segments $\left(\Delta \chi = \frac{\Delta \chi_{\mathrm{p}}}{N}\right)$. It is not intuitive for a mission designer to place bounds on this new independent angle variable. It is much more natural to think in terms of flight times, therefore, the MGALT flight time constraints are retained in this new method for the mission designer to specify, with the task of selecting the total displacement in generalized anomaly being left to the NLP solver. The universal Kepler solver is used in an inverse sense to compute the time-of-flight that results from the angular displacement in each low-thrust control segment, in addition to propagating the state without the need for iteration. An epoch constraint is then imposed at the match point ensuring that the accumulated propagation time arcs $\left(\sum_{k=1}^{N} \Delta t_k\right)$ sum to the total phase flight time variable Δt_{p}. The procedure for evaluating a trajectory phase, and for obtaining all the necessary partials to compute match point derivatives, is as follows (definitions for the symbols below are provided in the Appendix):

1. $\Delta \chi_{\mathrm{p}}$ is selected by the NLP solver and is the total change in generalized anomaly required to propagate one trajectory phase. The size of the propagation segment, between each control node, is $\Delta \chi = \frac{\Delta \chi_{\mathrm{p}}}{N}$
2. Starting at the left boundary, propagate inwards to the phase center for $\frac{N}{2}$ segments of generalized anomaly
3. For each propagation segment in the forward half-phase:
   (a) beginning with some post-maneuver state $\mathbf{X}_{k-1}^{+} = \begin{bmatrix}\mathbf{r}_{k-1}^{+} & \mathbf{v}_{k-1}^{+}\end{bmatrix}^T$, compute α (Eq. (3.11)), the reciprocal of the semimajor axis, to determine if the current conic orbit is elliptic, hyperbolic or parabolic (this affects the computation of the universal functions $\{U_n\}$ and their derivatives)

(b) compute the universal functions $U_n \quad n = 0, 1, 2, 3$ (Eq. (3.15) - (3.17)).

(c) compute $\frac{\partial U_n}{\partial \alpha}$ and $\frac{\partial U_n}{\partial \chi}$ (Eq. (4.25)) for $n = 0, 1, 2, 3$ (for use in computing step g).

(d) compute the segment propagation time Δt_k (Eq. (4.7)) resulting from a propagation through $\Delta\chi$ of generalized anomaly as well as $\frac{\partial \Delta t_k}{\partial \chi}$ (Eq. (4.22))

(e) compute the Lagrange coefficients and their derivatives F, $\frac{\partial F}{\partial \chi}$, $\dot{F}$, $\frac{\partial \dot{F}}{\partial \chi}$, G, $\frac{\partial G}{\partial \chi}$, $\dot{G}$ and $\frac{\partial \dot{G}}{\partial \chi}$ (Eq. (3.18) - (4.30))

(f) compute the propagated state $\mathbf{X}_k^- = \begin{bmatrix}\mathbf{r}_k^- & \mathbf{v}_k^-\end{bmatrix}^T$ (Eq. (3.1))

(g) compute $\frac{\partial \mathbf{X}_k^-}{\partial \mathbf{X}_{k-1}^+}$ [1] and $\frac{\partial \Delta t_k}{\partial \mathbf{X}_{k-1}^+}$ (Eqs. (4.23) and (4.24)), which form the basis for computing match point derivatives

(h) compute the maximum available thrust $T_{\max_k}$ using Eq. (2.15)

(i) compute $\Delta v_{\max_k}$ with Eq. (3.2)

(j) compute the throttle magnitude constraint $\|\mathbf{u}_k\| < 1$

(k) apply the impulsive maneuver $\mathbf{v}_k^+ = \mathbf{v}_k^- + \mathbf{u}_k \Delta v_{\max_k}$

(l) compute the spacecraft's mass after the applied impulse m_{k+1}^+ with Eq. (3.6)

4. Locate the right hand phase boundary using the phase flight time NLP decision variable Δt_{p}

5. Repeat step 3 for the backwards half-phase

6. compute the match point constraint $\mathbf{c}_{\mathrm{mp}}$ and its partials $\frac{\partial \mathbf{c}_{\mathrm{mp}}}{\partial \mathbf{p}}$

$$\mathbf{c}_{\mathrm{mp}} = \begin{bmatrix} \mathbf{r}^B - \mathbf{r}^F \\ \mathbf{v}^B - \mathbf{v}^F \\ m^B - m^F \\ t^B - t^F \end{bmatrix} = \mathbf{0} \tag{4.4}$$

where the F and B superscripts denote quantities at the end of the forward and backward propagated half-phases respectively. The epochs on either side of the match point are computed as follows, with t_0 referring to the epoch at the beginning of the phase, and t_f referring to the epoch at the end of the phase:

[1]This matrix is calculated using Eqs. (9.84)-(9.87) in Battin [160] or by differentiating Eq. (3.1) and using the additional derivative information provided in the Appendix. This also provides a means for deriving the Battin equations.

$$t^F = t_0 + \sum_{k=1}^{N/2} \Delta t_k \tag{4.5}$$

$$t^B = t_f - \sum_{k=N/2+1}^{N} \Delta t_k \tag{4.6}$$

In the procedure above, the - and + superscripts indicate a quantity immediately prior to and after an applied impulse respectively. The 3x1 vector $\mathbf{u}_k$ contains the control parameters of the k^{th} control node. The scalar quantity $\Delta v_{\max}$ represents the maximum Δv achievable by the spacecraft were it to operate its thruster continuously over the course of the propagation segment, and is computed using Eq. (3.2). The spacecraft's mass across the k^{th} bounded impulse is computed using Eq. (3.6).

One peculiarity of this method is that the maximum size of a maneuver $\Delta v_{\max_k}$ can only be determined after the propagation time Δt has been computed. This is in contrast to using an equivalent time spacing of the nodes (MGALT) where segment propagation time is known prior to propagation and hence the trajectory may be computed in half-segments. For this reason, maneuvers must occur at the the right-hand boundary of a segment instead of in the center as they would for the original MGALT transcription. This is immaterial, as the location of the impulse is more or less arbitrary for an impulsive approximation of continuous-thrust, save for the case of the last segment in any half-phase where placing the maneuver on the right-hand boundary results in it occurring exactly at the match point. When both half-phases are considered, this results in the burn at the match point having a maximum *potential* size that is twice as large as the others. This is not a major concern, however, as the purpose of bounded-impulse methods is to produce an accurate estimation of the total mass requirement of a mission, and not necessarily the exact topology of its trajectory, and this method does not violate the total possible Δv that the low-thrust engine is capable of imparting. This issue also reduces asymptotically with the use of more control nodes.

4.3 MGALTS Iteration-Free Propagator

When a two-body trajectory is propagated in the time domain (i.e. when time is used as the independent variable of propagation) using Kepler's equation, an iterative root finding algorithm must be applied to solve for the corresponding change in eccentric anomaly. The need for iteration is due to Kepler's equation being transcendental in the anomaly variable (χ in Eq. (3.30)).

The original MGALT transcription relies on two-body propagation by solving Kepler's equation in the time domain. By contrast, the multiple gravity assist with low-thrust using a Sundman transformation (MGALTS) model employs Kepler's equation in an inverse sense, as the independent variable becomes the angular anomaly χ. The propagation time Δt associated with a corresponding change in generalized anomaly χ is computed as follows:

$$\Delta t_k = \frac{1}{\sqrt{\mu}} \left(r_{k-1} U_1 + \sigma_{k-1} U_2 + U_3 \right) \tag{4.7}$$

where σ_{k-1} is computed using Eq. (3.27) and the universal variables U_n are computed using Eq. (3.15) - (3.17) or alternatively, using the recursion relation in Eq. (A.6). Equation (4.7) is required to compute the epoch match point constraint in Eq. (4.4) and is not transcendental in Δt. By switching to an angular independent propagation variable, the MGALTS transcription removes the iteration that is required to propagate the spacecraft state vector for the MGALT model.

4.4 MGALTS True Anomaly Node Spacing

The selection of the generalized anomaly as the independent variable is natural for this new algorithm as the universal Kepler propagator makes direct use of it. The use of this variable instead of time is convenient as the two-body nonlinear differential equations of motion can be transformed into linear differential equations with constant coefficients. The solution to these differential equations can be obtained via a power series expansion that gives rise to the universal functions U_n. The transformation that allows for this is given in Eq. (4.3), with its generalization commonly expressed as:

$$dt = c\, r^{\gamma} ds \tag{4.8}$$

where $c \in \mathbb{R}$ and $\gamma \in \{\mathbb{R} : \gamma \geq 0\}$. The case of $\gamma = 0$ results in an equal temporal spacing of the control nodes across a trajectory arc, i.e. the original Sims-Flanagan transcription. An equal spacing in generalized or eccentric anomaly corresponds to $\gamma = 1$. For highly eccentric orbits, it is beneficial to cluster more control nodes in the vicinity of periapse, which can be achieved by increasing γ. For $\gamma = 3/2$ the control nodes align with the elliptic (intermediate) anomaly and $\gamma = 2$ results in a distribution corresponding to true anomaly f.

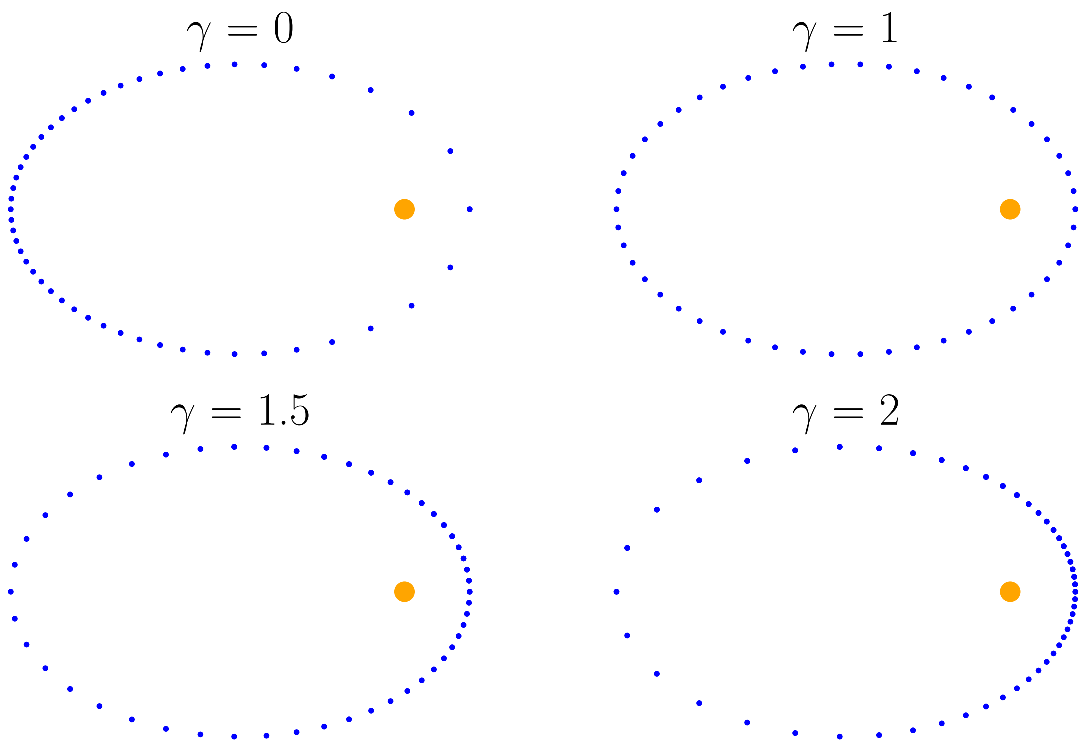


Figure 4.1: Generalized Sundman transformation applied to a discretized orbit with a semimajor axis of 1.0 and eccentricity of 0.716 where $c = 1.0$.

Arbitrary selection of γ in the context of a numerical integration scheme like Runge-Kutta or Taylor is possible as Eq. (4.8) is easily integrated using these numerical techniques. The set of universal functions $\{U_n\}$ in the analytic Kepler solver, however, comes about for the special case of $\gamma = 1$. For other powers of r, the convenient transformation of the nonlinear differential equations of motion to linear ones is destroyed. If, however, a different choice of γ (anomaly type) can be related to the generalized anomaly, the generalized anomaly-based propagation technique employing $\{U_n\}$ can be retained.

To illustrate this concept, consider the case where the independent variable chosen by the NLP solver is changed to the total change in true anomaly over one phase Δf_{p}, with the total change in true anomaly over one segment being $\Delta f = \frac{\Delta f_{\text{p}}}{N}$. In order to utilize the same universal propagation technique, the change in the generalized anomaly $\Delta\chi$ must be computed from the change in true anomaly Δf, and several additional derivatives must be computed for the Jacobian matrix. The following relationships between the change in true anomaly and the first two universal functions are due to Battin [160] [2].

$$U_0\left(\frac{\chi}{2};\alpha\right) = \sqrt{\frac{r_{k+1}}{r_k\ p}}\left(\sqrt{p}\ \cos\left(\frac{1}{2}\Delta f\right) - \sigma_k\ \sin\left(\frac{1}{2}\Delta f\right)\right) \tag{4.9}$$

$$U_1\left(\frac{\chi}{2};\alpha\right) = \sqrt{\frac{r_{k+1}r_k}{p}}\sin\left(\frac{1}{2}\Delta f\right) \tag{4.10}$$

The orbital parameter is defined as $p = a(1-e^2)$. Using Equations (4.9) and (4.10) as well as Eqs. (3.15) - (3.17), the following relationships between the change in true anomaly and the change in generalized anomaly are obtained:

$$\Delta\chi = \begin{cases} \frac{2r_k\sin\left(\frac{1}{2}\Delta f\right)}{\sqrt{p}\ \cos\left(\frac{1}{2}\Delta f\right)-\sigma_k\sin\left(\frac{1}{2}\Delta f\right)} & ;\alpha = 0 \\ \frac{2}{\sqrt{\alpha}}\tan^{-1}\left[\frac{\sqrt{\alpha}r_k\sin\left(\frac{1}{2}\Delta f\right)}{\sqrt{p}\ \cos\left(\frac{1}{2}\Delta f\right)-\sigma_k\sin\left(\frac{1}{2}\Delta f\right)}\right] & ;\alpha > 0 \\ \frac{2}{\sqrt{-\alpha}}\tanh^{-1}\left[\frac{\sqrt{-\alpha}r_k\sin\left(\frac{1}{2}\Delta f\right)}{\sqrt{p}\ \cos\left(\frac{1}{2}\Delta f\right)-\sigma_k\sin\left(\frac{1}{2}\Delta f\right)}\right] & ;\alpha < 0 \end{cases} \tag{4.11}$$

4.5 MGALTS Analytic Match Point Gradients

Computation of match point gradients for MGALTS requires slight alterations to be made to the algorithms from chapter 3, as we must now account for the addition of the χ_p decision variables as well as the fact that the segment propagation times (Δt_k variables) are now dependent on the spacecraft's state vector at the beginning of the segment in contrast to MGALT where the spacecraft's state at the end of a segment is dependent on the propagation time. To accommodate these changes, the STMs and MTMs must each be augmented with an additional row and column to hold partials required to compute sensitivities with respect to χ. It should also be noted that the STMs must now hold $\frac{\partial \Delta t_k}{\partial \mathbf{X}_k^+}$ derivative information.

[2]These equations are obtained by equating two expressions for the F and G Lagrange coefficients. The first set describes F and G in terms of change in true anomaly and the second set in terms of the universal functions (Battin [160] Eqs. (3.42) and (4.84) respectively)

$$\boldsymbol{\Phi}_k = \frac{\partial \mathbf{X}_k^-}{\partial \mathbf{X}_{k-1}^+} = \left[\begin{array}{cc|c|c|c|c}
\frac{\partial \mathbf{r}_k^-}{\partial \mathbf{r}_{k-1}^+} & \frac{\partial \mathbf{r}_k^-}{\partial \mathbf{v}_{k-1}^+} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \frac{\partial \mathbf{r}_k^-}{\partial \chi} \\
\frac{\partial \mathbf{v}_k^-}{\partial \mathbf{r}_{k-1}^+} & \frac{\partial \mathbf{v}_k^-}{\partial \mathbf{v}_{k-1}^+} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \frac{\partial \mathbf{v}_k^-}{\partial \chi} \\
\cline{1-2}
\mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 1 & 0 & 0 & 0 \\
\cline{1-3}
\frac{\partial t}{\partial \mathbf{r}_{k-1}} & \frac{\partial t}{\partial \mathbf{v}_{k-1}} & 0 & 1 & 0 & \frac{\partial t}{\partial \chi} \\
\cline{1-4}
\frac{\partial \Delta t_k}{\partial \mathbf{r}_{k-1}} & \frac{\partial \Delta t_k}{\partial \mathbf{v}_{k-1}} & 0 & 0 & 0 & \frac{\partial \Delta t_k}{\partial \chi} \\
\cline{1-5}
0 & 0 & 0 & 0 & 0 & 1
\end{array}\right] \tag{4.12}$$

$$\mathbf{M}_k = \frac{\partial \mathbf{X}_k^+}{\partial \mathbf{X}_k^-} = \left[\begin{array}{cc|c|c|c|c}
\mathbb{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\
\frac{\partial \mathbf{v}_k^+}{\partial \mathbf{r}_k^-} & \mathbb{I}_{3\times3} & \frac{\partial \mathbf{v}_k^+}{\partial m_k} & \mathbf{M}_{k_{24}} & \mathbf{M}_{k_{25}} & \mathbf{0}_{3\times1} \\
\hline
\frac{\partial m_{k+1}}{\partial \mathbf{r}_k^-} & \mathbf{0}_{1\times3} & 1 & \mathbf{M}_{k_{34}} & \mathbf{M}_{k_{35}} & 0 \\
\hline
\multicolumn{3}{c|}{\mathbf{0}_{3\times7}} & \multicolumn{3}{c}{\mathbb{I}_{3\times3}}
\end{array}\right] \tag{4.13}$$

The spacecraft state, at any point in time, does not have a dependence on the phase flight time Δt_p for the Sundman regularized version of MGALT, therefore, expressions for the maneuver transition matrix (MTM) sub-matrices differ slightly than what was presented in Chapter 3:

$$\mathbf{M}_{k_{24}} = \mathbf{u}_k \frac{D\Delta t}{m_k} \cdot \frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \tag{4.14}$$

$$\mathbf{M}_{k_{24}} = -\mathbf{u}_k \frac{D\Delta t}{m_k} \left(\frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} - \|\mathbf{u}_k\| \frac{D\Delta t \; T_{\max_k}}{m_k} \frac{\partial \dot{m}_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial t_k} \right) \tag{4.15}$$

$$\mathbf{M}_{k_{25}} = \mathbf{u}_k \frac{D}{m_k} \left(\Delta t \frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial \Delta t_k} + T_{\max_k} \right) \tag{4.16}$$

$$\begin{aligned}\mathbf{M}_{k_{25}} =& - \mathbf{u}_k \frac{D}{m_k} \left[\Delta t \frac{\partial T_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial \Delta t_k} + T_{\max_k} \right. \\ & \left. - \|\mathbf{u}_k\| \frac{D\Delta t \; T_{\max_k}}{m_k} \left(\dot{m}_{\max_k} + \Delta t \; \frac{\partial \dot{m}_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial \Delta t_k} \right) \right]\end{aligned} \tag{4.17}$$

$$\mathbf{M}_{k_{34}} = \mp \|\mathbf{u}_k\| D \Delta t \, \frac{\partial \dot{m}_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial \Delta t_k} \tag{4.18}$$

$$\mathbf{M}_{k_{35}} = \mp \|\mathbf{u}_k\| D \left(\dot{m}_{\max_k} + \Delta t \, \frac{\partial \dot{m}_{\max_k}}{\partial P_k} \cdot \frac{\partial P_k}{\partial \Delta t_k} \right) \tag{4.19}$$

For most applications $\frac{\partial P_k}{\partial \Delta t_k}$ will be zero, since the available power would most likely not have an explicit dependency on the segment propagation time (note that this does not include the non-explicit dependency on propagation time via distance from the Sun $\mathbf{r}_k$, or the explicit dependency on current epoch t_k, which is captured in Eq. (4.14)).

The spacecraft's state partials with respect to the generalized anomaly are calculated by differentiating the Lagrange coefficients:

$$\frac{\partial \mathbf{r}_{k+1}}{\partial \chi} = \frac{\partial F}{\partial \chi} \mathbf{r}_k + \frac{\partial G}{\partial \chi} \mathbf{v}_k \tag{4.20}$$

$$\frac{\partial \mathbf{v}_{k+1}}{\partial \chi} = \frac{\partial \dot{F}}{\partial \chi} \mathbf{r}_k + \frac{\partial \dot{G}}{\partial \chi} \mathbf{v}_k \tag{4.21}$$

There are several required partials of the segment propagation time:

$$\frac{\partial \Delta t}{\partial \chi} = \frac{1}{\sqrt{\mu}} \left(r_k \frac{\partial U_1}{\partial \chi} + \sigma_k \frac{\partial U_2}{\partial \chi} + \frac{\partial U_3}{\partial \chi} \right) \tag{4.22}$$

$$\frac{\partial \Delta t}{\partial \mathbf{r}_k} = \frac{1}{\sqrt{\mu}} \left(\frac{\partial r_k}{\partial \mathbf{r}_k} U_1 + r_k \frac{\partial U_1}{\partial \mathbf{r}_k} + \frac{\partial \sigma_k}{\partial \mathbf{r}_k} U_2 + \sigma_k \frac{\partial U_2}{\partial \mathbf{r}_k} + \frac{\partial U_3}{\partial \mathbf{r}_k} \right) \tag{4.23}$$

$$\frac{\partial \Delta t}{\partial \mathbf{v}_k} = \frac{1}{\sqrt{\mu}} \left(r_k \frac{\partial U_1}{\partial \mathbf{v}_k} + \frac{\partial \sigma_k}{\partial \mathbf{v}_k} U_2 + \sigma_k \frac{\partial U_2}{\partial \mathbf{v}_k} + \frac{\partial U_3}{\partial \mathbf{v}_k} \right) \tag{4.24}$$

Partials of the universal functions with respect to the generalized anomaly are calculated using the following recursion:

$$\frac{\partial U_0}{\partial \chi} = -\alpha U_1; \quad \frac{\partial U_n}{\partial \chi} = U_{n-1} \quad n = 1, 2, ... \tag{4.25}$$

$$\frac{\partial U_n}{\partial \mathbf{X}_k^-} = \frac{\partial U_n}{\partial \alpha}\frac{\partial \alpha}{\partial \mathbf{X}_k^-} \quad n = 0, 1, 2, ... \tag{4.26}$$

The derivatives of the Lagrange coefficients are given by:

$$\frac{\partial F}{\partial \chi} = -\frac{1}{r_k}\frac{\partial U_2}{\partial \chi} \tag{4.27}$$

$$\frac{\partial \dot{F}}{\partial \chi} = \frac{\sqrt{\mu}}{r_k}\left(\frac{U_1}{r_{k+1}^2}\frac{\partial r_{k+1}}{\partial \chi} - \frac{1}{r_{k+1}}\frac{\partial U_1}{\partial \chi}\right) \tag{4.28}$$

$$\frac{\partial G}{\partial \chi} = \frac{1}{\sqrt{\mu}}\left(r_k\frac{\partial U_1}{\partial \chi} + \sigma_k\frac{\partial U_2}{\partial \chi}\right) \tag{4.29}$$

$$\frac{\partial \dot{G}}{\partial \chi} = \frac{U_2}{r_{k+1}^2}\frac{\partial r_{k+1}}{\partial \chi} - \frac{1}{r_{k+1}}\frac{\partial U_2}{\partial \chi} \tag{4.30}$$

Expressions for the Lagrange coefficients themselves are provided in Eq. (3.18)-(3.23). The distance from the central body at the end of the propagation segment is given by Eq. (3.24) and its partial with respect to χ is:

$$\frac{\partial r_{k+1}}{\partial \chi} = r_k\frac{\partial U_0}{\partial \chi} + \sigma_k\frac{\partial U_1}{\partial \chi} + \frac{\partial U_2}{\partial \chi} \tag{4.31}$$

Once the derivatives for the generalized anomaly variables have been calculated, the chain rule allows one to easily extend the derivative calculation if true anomaly is used instead by way of Eq. (4.11):

$$\frac{\partial \Delta\chi}{\partial \Delta f_{\mathrm{p}}} = \frac{\partial \Delta\chi}{\partial \Delta f}\frac{\partial \Delta f}{\partial \Delta f_{\mathrm{p}}} \tag{4.32}$$

$$\frac{\partial \Delta\chi}{\partial \Delta f} = \begin{cases} \frac{r_k\sqrt{p}}{\left[\sigma_k\ \sin\left(\frac{1}{2}\Delta f\right)-\sqrt{p}\ \cos\left(\frac{1}{2}\Delta f\right)\right]^2} & ;\alpha = 0 \\ \frac{2\sqrt{p}r_k}{p+\alpha r_k^2+\sigma_k^2+\left(p-\sigma_k^2-\alpha r_k^2\right)\cos(\Delta f)-2\sqrt{p}\sigma_k\sin(\Delta f)} & ;\alpha \neq 0 \end{cases} \tag{4.33}$$

4.6 Application: Rendezvous with Comet 45P/Honda-Mrkos-Pajdušáková

The problem selected to illustrate the benefits of time regularization is a notional rendezvous with the comet 45P/Honda-Mrkos-Pajdušáková. This is a short-period comet with a period of approximately 5.25 years. We consider a direct flight to the comet, with no intermediate flybys. The rendezvous problem was solved once using the MGALT transcription (Figure 4.2), then a second time using the MGALTS transcription with generalized anomaly nodal spacing (Figure 4.3), and then finally using MGALTS with true anomaly nodal spacing (Figures 4.7 and 4.8). All three optimizations were performed using the same configuration parameters shown in Table 4.1. The NLP objective was the minimization of the propellant consumed.

Table 4.1: 45P/Honda-Mrkos-Pajdušáková rendezvous mission parameters

Parameter	Value
Maximum allowed initial mass	4000 kg
Earliest allowed launch date	August 27th, 2016
Launch window	5 years
Maximum flight time	unbounded
Latest allowed arrival date	January 1st, 2030
Launch vehicle	Atlas V (555) NLS-2
Maximum launch C3	21.7156 km^2/s^2
Launch declination bounds	$[-28.5°, 28.5°]$
Arrival type	rendezvous
Thruster	2 x NEXT TT11 high thrust
Throttle logic	minimum number of thrusters
Thruster duty cycle	0.95
Solar power coefficients	$\gamma_0 = 1.32077$ $\gamma_1 = -0.10848$
	$\gamma_2 = -0.11665$ $\gamma_3 = 0.10843$
	$\gamma_4 = -0.01279$
Spacecraft bus power requirement coefficients	$a_{s/c} = 1.0$
	$b_{s/c} = 0.0$
	$c_{s/c} = 0.0$
Solar array BOL power (launch at 1 A.U.)	40.0 kW
Power margin (δ_{power})	15%
Post-launch checkout coast	60 days
Number of segments per phase	200
Ephemeris	SPICE
SNOPT feasibility tolerance	1.0e-5
SNOPT optimality tolerance	1.0e-4
Objective function	maximize final mass

Specific information about power modeling and throttle logic used is omitted here but is available, and adheres to Discovery class mission proposal requirements [113, 131].

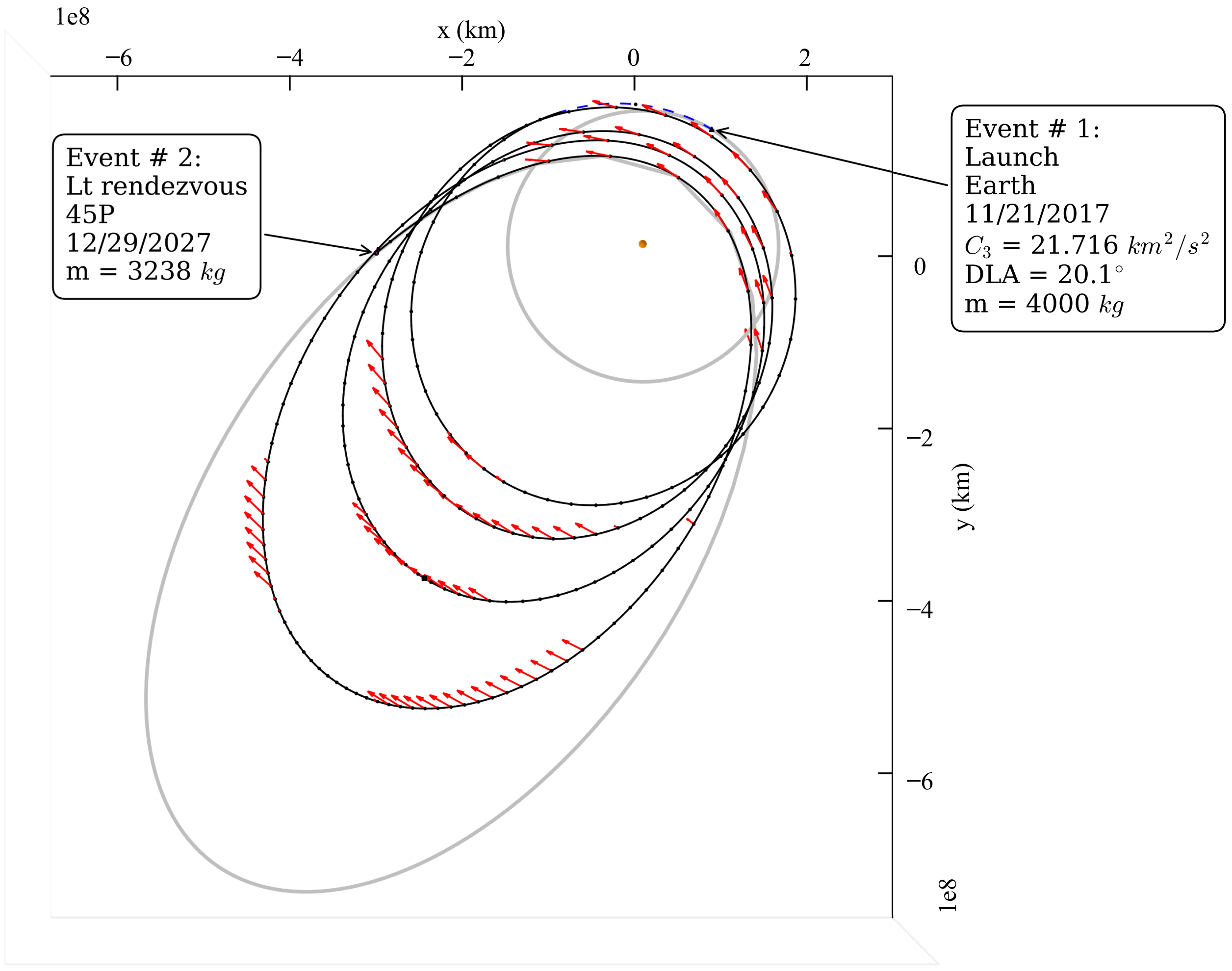


Figure 4.2: Rendezvous with 45P using MGALT with 200 segments, $\Delta t_{\text{mission}} = 3690$ days

Figures 4.2 and 4.3 show that the optimizer arrives at two solutions that differ by less than one percent in terms of mass delivered to the comet. The major difference between the two solutions is the flight time required for the transfer. The MGALT trajectory requires 10.10 years, whereas the MGALTS completes the rendezvous in only 7.96 years. The throttle histories depicted in Figures 4.4 and 4.5 shed light on why this is the case. The MGALTS trajectory features a much later launch date, yet arrives earlier at the target. It is able to achieve this due to the greater number of control points located near periapsis for that transcription. The clustering of control nodes there allows the pump-up maneuver to happen more efficiently, meaning the spacecraft can match the energy of the comet's orbit in a shorter time. The MGALT solution on the other hand is not able to achieve the same energy change per periapsis passage. This works out in a particularly unfortunate way for this mission as it results in the spacecraft having to make an entire extra revolution of the sun in order to carry out the rendezvous.

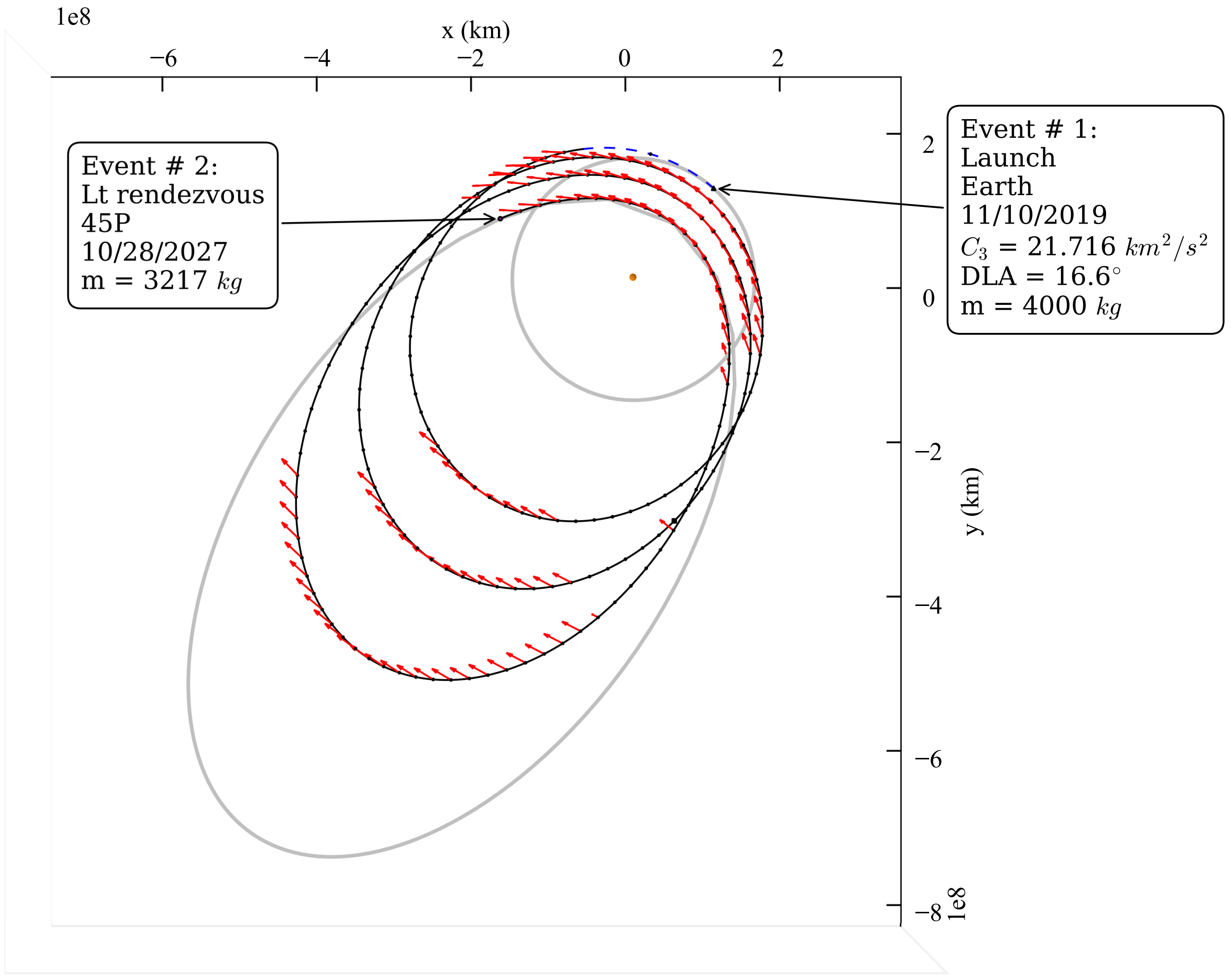


Figure 4.3: Rendezvous with 45P using time-regularized MGALTS with 200 segments, $\Delta t_{\text{mission}} = 2909$ days

By increasing the number of control nodes for the MGALT model, a solution qualitatively similar to the MGALTS solution can be obtained due to the increased number of nodes at periapsis. Figure 4.6 shows the new MGALT solution, this time using 400 segments instead of 200. Increasing the segment count by degrees will eventually yield a trajectory with a more comparable flight time to the MGALTS solution.

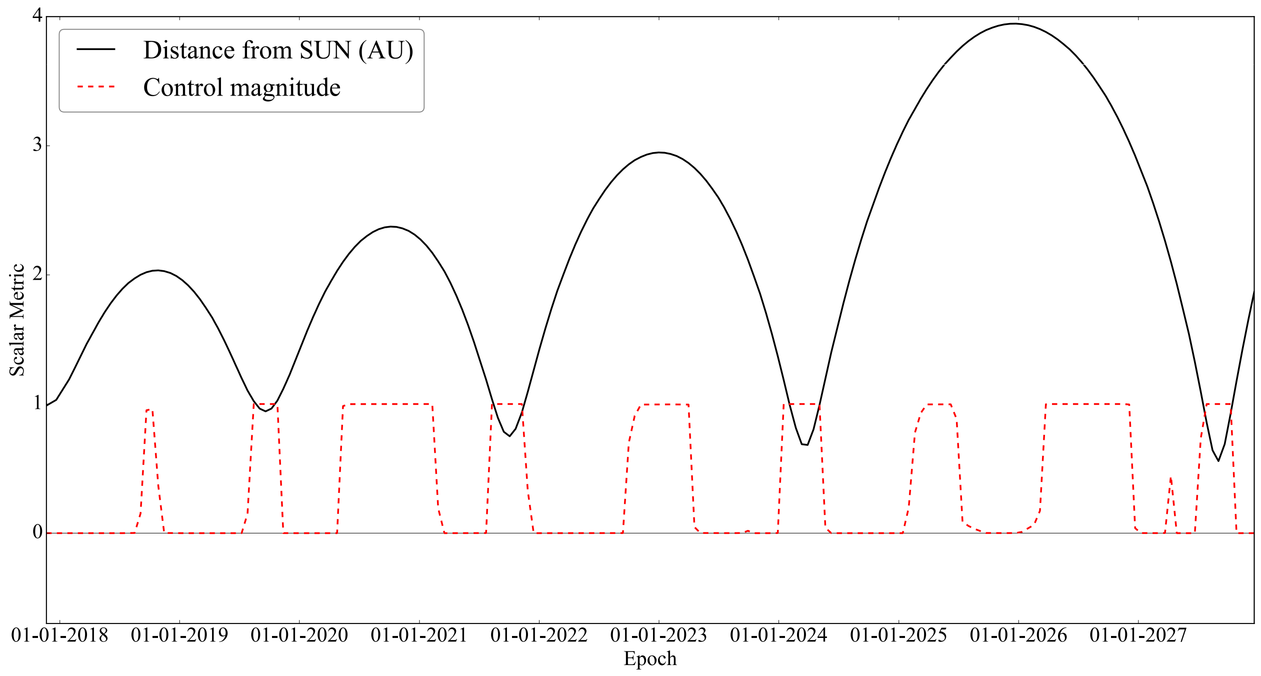


Figure 4.4: MGALT throttle magnitude and distance from the sun as a function of time.

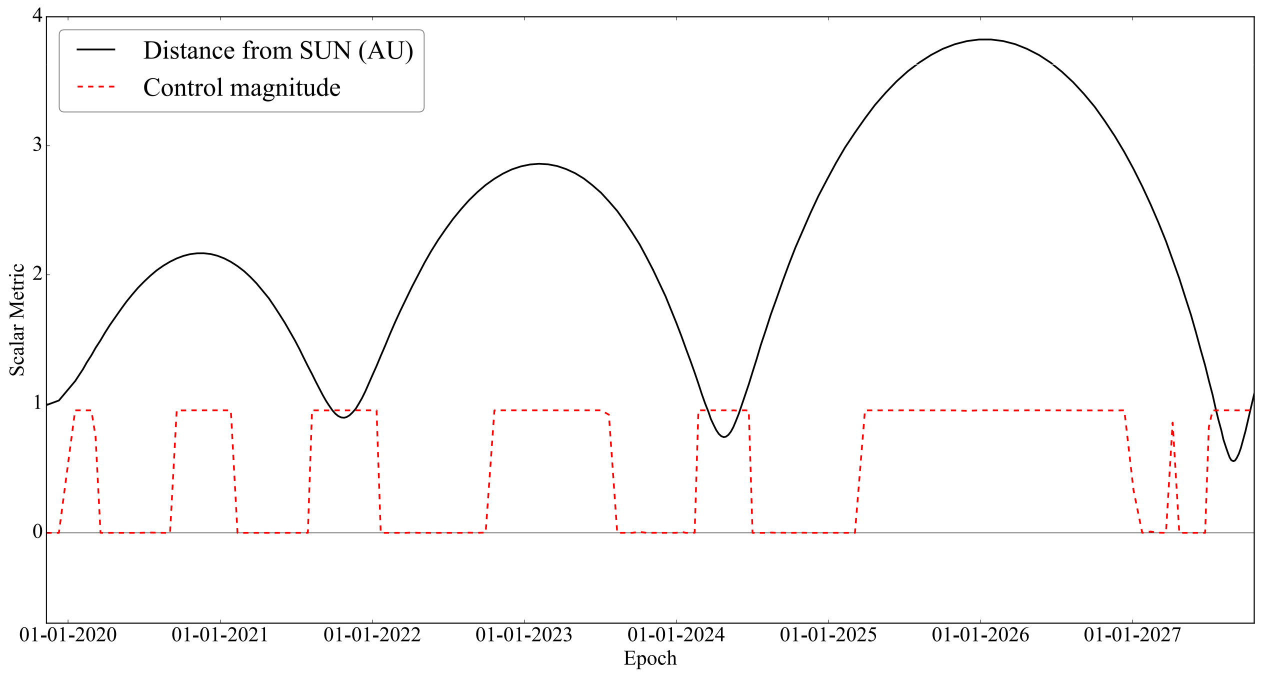


Figure 4.5: MGALTS throttle magnitude and distance from the sun as a function of time.

We will now examine how the true anomaly-based version of the MGALTS transcription fares with this problem. It is first worth noting that when the NLP solver uses total phase true anomaly as its decision variable, it is capable of recovering a solution similar to that found when using generalized anomaly as a

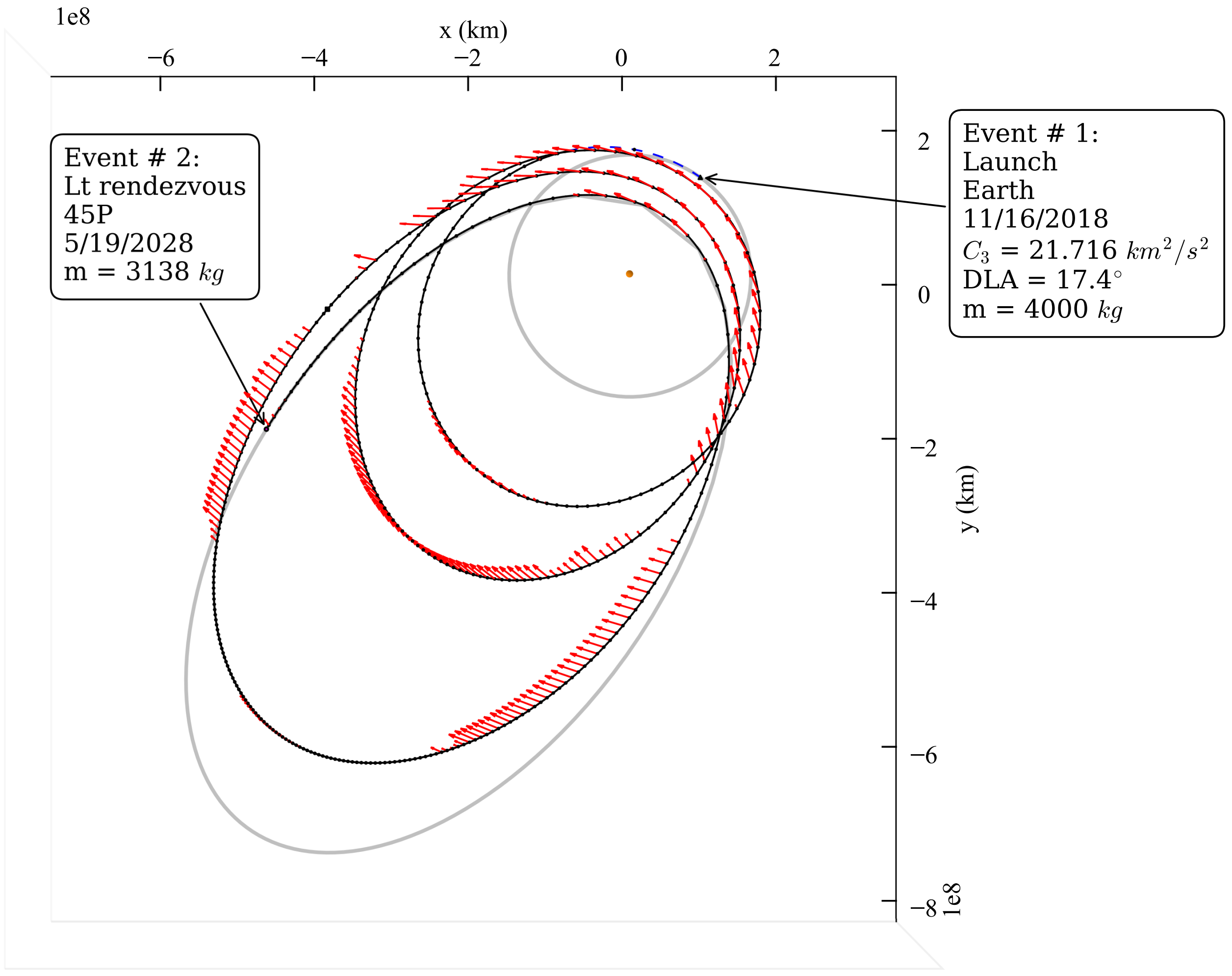


Figure 4.6: Rendezvous using MGALT with 400 segments, $\Delta t_{\text{mission}} = 3472$ days

decision variable, as shown in Figure 4.7. This added flexibility makes this version of the model especially attractive for use in generating Pareto surfaces, which are more valuable sets of data for a mission proposal team than a single point solution that extremalizes one cost function such as delivered mass [100, 101].

With the higher concentration of control nodes near periapse, however, other solution families may also be accessed. In particular, trajectories featuring much shorter flight times become available. Figure 4.8 shows a rendezvous trajectory with the comet that takes 6.43 years to complete. The redistribution of the control nodes in the true-anomaly based model allows for solutions such as this where, compared to the mission shown in Figure 4.7, 1.5 years of flight time are traded for approximately 300 kg of propellant.

The comet 45/P rendezvous problem is an illustrative example of a mission that benefits from the new regularized bounded-impulse model discussed in this chapter. The increased control authority afforded by locating more control points near periapsis allows for the transfer to be completed with less propellant

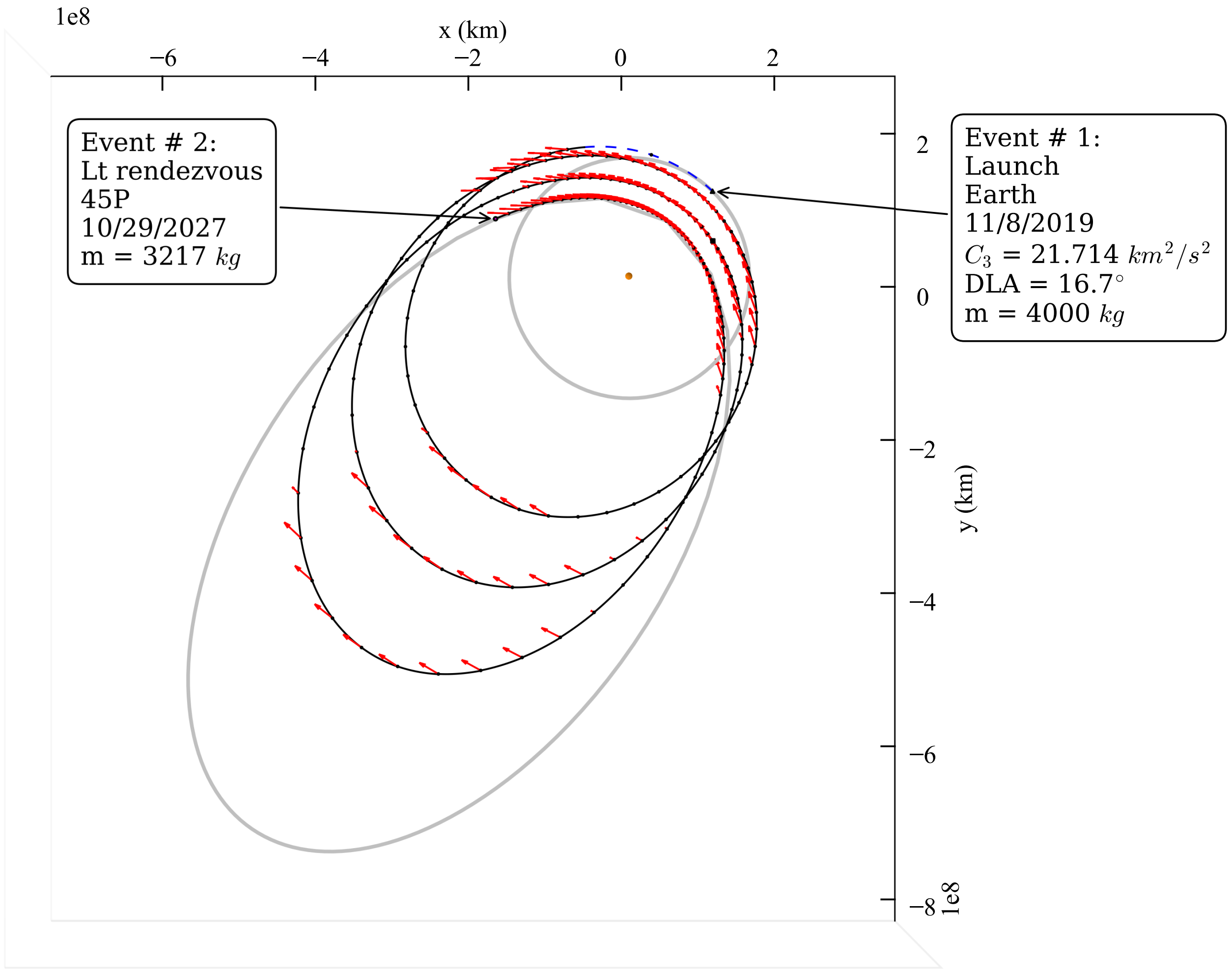


Figure 4.7: Rendezvous using the MGALTS true anomaly-based transcription with 200 segments, $\Delta t_{\text{mission}} = 2913$ days

used or with a reduced flight time. The fact that both of these solutions are easily discovered by the local optimizer suggests that the MGALTS transcription would provide greater flexibility to a global optimizer seeking to identify solutions that are Pareto optimal in mass delivered to the comet and time-of-flight.

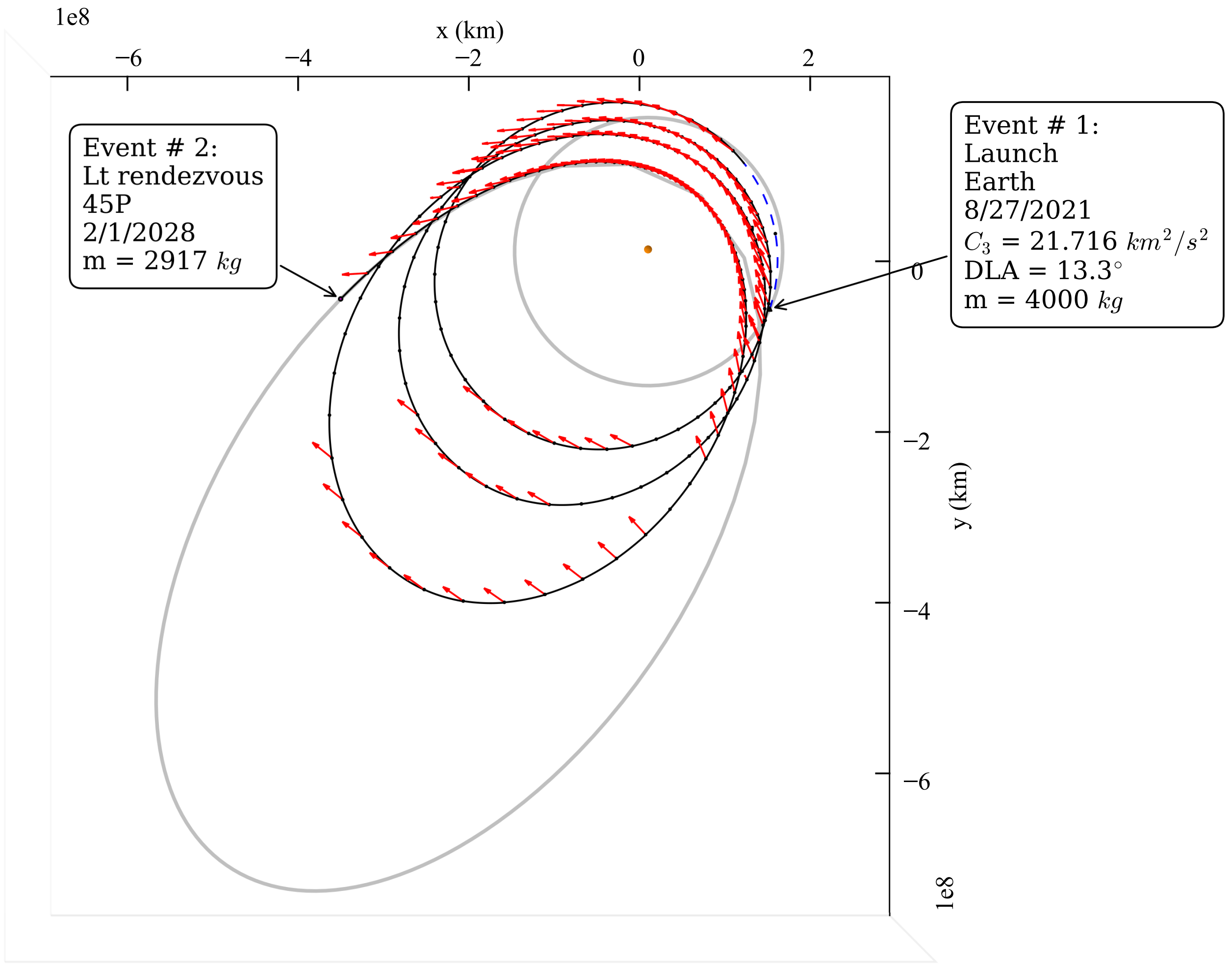


Figure 4.8: Rendezvous using the MGALTS true anomaly-based transcription with 200 segments, $\Delta t_{\text{mission}} = 2350$ days

Chapter 5

Analytic Gradient Computation for Finite-Burn Trajectory Models

Figure 5.1: 70 meter antenna at Goldstone, California.

Bounded-impulse trajectory transcriptions are capable of estimating the mass requirements for a wide variety of missions, however, they can result in an inaccurate representation of the topology of the trajectory. Higher fidelity modeling is therefore often necessary to bridge the gap between a low-fidelity solution and a high-fidelity one as would be produced by software packages such as GMAT, FreeFlyer, STK, or Mystic. Proper operation of these tools can require a great deal of human analyst time that may not be available or affordable in the context of a proposal. In addition, these tools may not even be made available to all teams wishing to submit a proposal due to their cost or restricted access. An intermediate level of design fidelity that is easy and inexpensive to use and also readily available to all potential users is therefore highly desirable.

For both high and low-thrust propulsion regimes, the first step towards achieving higher fidelity is to adopt a numerical integration scheme in favor of the Keplerian propagators at the heart of most low-fidelity tools. Another important fidelity improvement is to adopt to a finite-burn model if the spacecraft is using an electric thruster. While most maneuvers executed with a chemical rocket motor can still reasonably be approximated as impulsive events right up until the flight-fidelity solution is closed in a high-heritage tool such as MIRAGE, the same is not true for continuous-thrust motors that operate for days or weeks at a time. It is therefore important to have the ability to model finite-burns, i.e. integrate with a thrust term in the equations of motion. Doing so can yield solution topologies that can vary significantly compared with their bounded-impulse equivalents.

The "medium" fidelity trajectory model described in this Chapter integrates the equations of motion rather than approximating them, and its acceleration model includes reasonably high fidelity models of propulsion and power systems as well as significant orbit perturbations such as third-body gravity, SRP, and central body oblateness. This model is sufficient to ensure that a proposed trajectory is actually feasible and is not an artifact of the lower fidelity models available in most preliminary design tools.

As one might expect, moving to a numerically integrated model drastically changes the way that analytic gradients must be computed. Chapter 5 will describe how the gradient calculation process changes for finite-burn models, and it will describe how specifically to compute analytic derivatives for a solar electric spacecraft employing an electric thruster in the context of a high-fidelity physics engine. Due to the runtime costs associated with numerical integration, user-provided gradients become even more valuable from a time saving perspective and providing them enables the use of a finite-burn model the context of preliminary design as a result of the decreased runtimes afforded by not employing finite differencing. Methods for numerically computing gradients for a patched-conic FBLT trajectory model will be discussed in this chapter. As in chapter 3, the methods described in this chapter allow for the incorporation of an accurate spacecraft power model, which is of vital importance for mission proposal efforts. An algorithm will also be presented for computing time-of-flight gradients for a finite-burn model using an explicit integration scheme, which enables optimization of free final time problems.

The FBLT trajectory model described in this chapter is the last step required before moving to a high-heritage tool such as GMAT or STK, and represents the true trajectory that will be flown very well. However, before a trajectory model such as FBLT can be trusted, it must be verified against a high-heritage navigation tool that might be used to actually fly the mission. A demonstration of the high-heritage software package

MIRAGE's ability to track a reference trajectory designed using FBLT will be the focus of section 5.6.

5.1 Finite-Burn Dynamical Model

The FBLT transcription divides a continuous-thrust spacecraft trajectory into N_p phases, each consisting of N segments in the same manner as the MGALT model. This transcription, however, increases the level of modelling fidelity over MGALT by replacing the bounded-impulse approximated thrust arcs with continuously integrated ones. One obvious advantage that this transcription has over MGALT is its ability to model the effects of forces from sources other than the central body and the spacecraft's thruster, such as other gravitating bodies and SRP by directly modelling them in the equations of motion. This is known as Cowell's method of special perturbations. The inertial Cartesian differential equations of motion governing a continuously thrusting spacecraft in orbit around a central body subject to gravitational forces from i perturbing bodies, SRP and central body oblateness are:

$$\ddot{\mathbf{r}} = -\frac{G\left(\cancelto{0}{m} + m_\mathrm{cb}\right)}{r^2}\frac{\mathbf{r}}{r} - \sum_\mathrm{i} G\, m_\mathrm{i}\left(\frac{\mathbf{r}-\mathbf{r}_\mathrm{i}}{\|\mathbf{r}-\mathbf{r}_\mathrm{i}\|^3} + \frac{\mathbf{r}_\mathrm{i}}{r_\mathrm{i}^3}\right) + \frac{C_r A_s K \phi}{c\, m\, r_{s/\odot}^2}\frac{\mathbf{r}_{s/\odot}}{r_{s/\odot}} + \ddot{\mathbf{r}}_{J_2} + \frac{D\,T}{m}\mathbf{u} \tag{5.1}$$

$$\dot{m} = -\|\mathbf{u}\|\, D\, \dot{m}_\mathrm{max} \tag{5.2}$$

where G is the universal gravitational constant, m is the mass of the spacecraft, m_cb is the mass of the central gravitational body, m_i is the mass of the i^{th} gravitational body, $\mathbf{r}$ is the position vector of the spacecraft with respect to the central body and $\mathbf{r}_\mathrm{i}$ is the position vector of the i^{th} body w. r. t. the central body.

Regarding the variables associated with radiation pressure, C_r is the coefficient of reflectivity on the interval $[0, 2]$, A_s is the spacecraft area exposed to sunlight, K is the total percentage of the sun seen by the spacecraft ($[0, 1]$), ϕ is the solar flux and c is the speed of light in a vacuum.

The central body J_2 acceleration term ($\ddot{\mathbf{r}}_{J_2}$) must be computed in the body-centered, fixed equatorial coordinate (BCF) frame. The central body's radius is denoted by R. If the other acceleration terms are computed in a different coordinate frame, then $\ddot{\mathbf{r}}_{J_2}$ must be rotated into that frame as well after it has been computed in BCF. The acceleration due to central body oblateness, measured in the inertial frame is given by Eq. (5.3).

$$\ddot{\mathbf{r}}_{J_2} = \ddot{\mathbf{r}}_{J_{2_{ICRF}}} = M_{BCF}^{ICRF} \ddot{\mathbf{r}}_{J_{2_{BCF}}} \tag{5.3}$$

where M_{BCF}^{ICRF} is the rotation matrix from the BCF frame to the ICRF frame. Methods for computing these rotation matrices are provided in Appendix A.5. The oblateness acceleration, measured in the central body fixed frame is given by:

$$\ddot{\mathbf{r}}_{J_{2_{BCF}}} = \frac{3J_2\mu R^2}{2r_{\mathrm{BCF}}^5} \begin{bmatrix} 1 - 5\frac{x_{\mathrm{BCF}} z_{\mathrm{BCF}}^2}{r_{\mathrm{BCF}}^2} \\ 1 - 5\frac{y_{\mathrm{BCF}} z_{\mathrm{BCF}}^2}{r_{\mathrm{BCF}}^2} \\ 3 - 5\frac{z_{\mathrm{BCF}}^3}{r_{\mathrm{BCF}}^2} \end{bmatrix} \tag{5.4}$$

The thruster variable $\mathbf{u}$ is a throttle scaling vector whose three components are selected by the optimizer on the range $[-1, 1]$, D is the thruster duty cycle (modelled continuously), T is the instantaneous thrust of the electric motor and $\dot{m}_{\max}$ is the maximum instantaneous mass flow rate of the propellant through the motor. Note that in Equation (5.1) the mass of the spacecraft is neglected in the first term since $m_{\mathrm{cb}} >> m$.

The vector geometry of this problem is illustrated in Figure 5.2. The diagram on the left depicts the general case, where the central body need not be the Sun, whereas the right hand diagram shows how the problem simplifies for a heliocentric trajectory. For the situation shown on the left in Figure 5.2, the Sun is included for two reasons, despite the fact that it is not the central body. The first reason is that for most mission designs, the Sun would almost certainly be included in the problem dynamics (the summation in Equation (5.1)) due to its gravitational dominance in the solar system. The second reason is that for accurate solar electric propulsion modelling, the position of the spacecraft with respect to the Sun must be tracked at all times throughout the optimization so that the solar flux may be calculated at the spacecraft's current location. The vectors $\mathbf{r}_{\mathrm{i}}$ and $\mathbf{r}_{\odot}$ are obtained from ephemeris calls. Note that $\mathbf{r}_{\mathrm{s}/\odot}$ is the position vector of the spacecraft with respect to the Sun and $\mathbf{r}_{\mathrm{s/i}}$ is the position vector of the spacecraft with respect to the ith gravitational body, both expressed in the coordinates of the inertial frame attached to the central body, e.g.

$$\mathbf{r}_{\mathrm{s}/\odot} = \mathbf{r} - \mathbf{r}_{\odot} \tag{5.5}$$

and

$$r_{s/\odot} = \left[(x - x_\odot)^2 + (y - y_\odot)^2 + (z - z_\odot)^2\right]^{1/2} \tag{5.6}$$

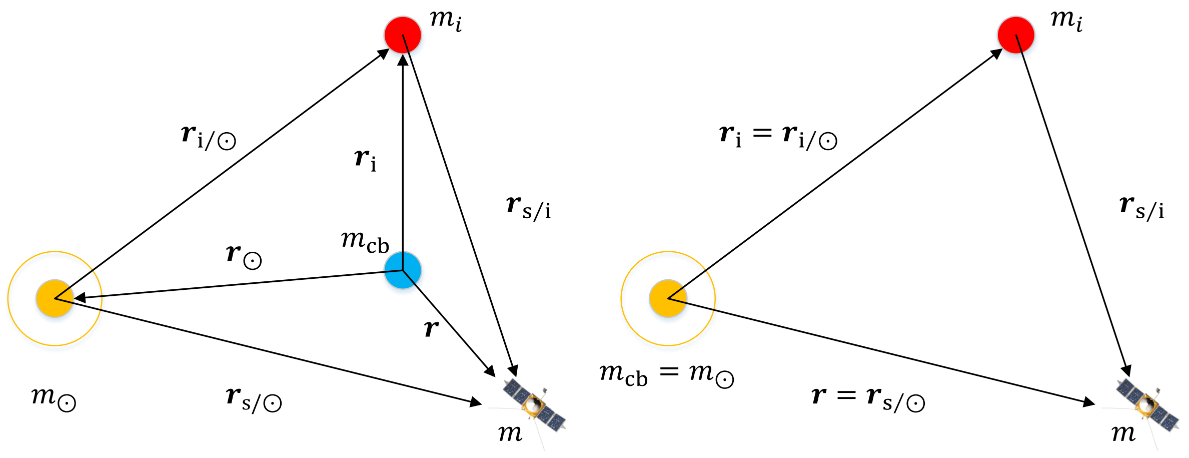


Figure 5.2: Vector diagrams for the general central body case and the heliocentric case.

5.2 Runge-Kutta Eighth Order Fixed-Step Integrator

The Runge-Kutta (RK) four stage integration method (RK4) is a classic choice for the numerical propagation of orbits [38,53]. This method has an error accumulation of $\mathcal{O}\left(h^4\right)$, where h is the size of the integration time step, and is also limited to a fixed step size. The numerical integrator employed for analysis in this chapter is a fixed-step version of the eighth order adaptive-step RK7(8)13M integrator developed by Ghosh [188]. This integrator uses $s = 13$ stages to propagate the state vector. The $RK8$ step propagation is calculated according to the following program:

$$\hat{\mathbf{X}}_{n+1} = \hat{\mathbf{X}}_n + \sum_{i=1}^{s} \hat{b}_i \mathbf{k}_i, \qquad i = 2, 3, ..., s \tag{5.7}$$

$$t_{n+1} = t_n + h \tag{5.8}$$

where

$$\mathbf{k}_1 = h \cdot \mathbf{f}\left(t_n, \quad \hat{\mathbf{X}}_n\right), \tag{5.9}$$

$$\mathbf{k}_i = h \cdot \mathbf{f}\left(t_n + c_i h, \quad \hat{\mathbf{X}}_{n(i)}\right) \tag{5.10}$$

The Runge-Kutta matrix entries a_{ij}, quadrature weights $(b_i, \hat{b}_i)$ and the nodes $(c_i = \sum_{j=1}^{i-1} a_{ij})$ are determined from the rk7(8)13M Butcher tableau [1], which is provided in the Appendix. The circonflex accent indicates that the eighth order solution tableau entries are used as opposed to the seventh order. The value of the state vector at the i^{th} intermediate stage of the n^{th} RK sub-step, $\hat{\mathbf{X}}_{n(i)}$, is computed by:

$$\hat{\mathbf{X}}_{n(i)} = \hat{\mathbf{X}}_n + \sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j \qquad i > j \quad i, j = 1, 2, 3, ..., s \tag{5.11}$$

5.3 Match Point Gradient Computation

5.3.1 General Method

Calculation of the FBLT match point derivatives is, as in the MGALT transcription, reliant on the computation of the spacecraft's state transition matrix, the derivative of the FBLT vector field with respect to initial conditions and is a linear mapping of the spacecraft's state vector $\mathbf{X}$ from an initial epoch t_0 to some later epoch t:

$$\mathbf{X}(t) = \mathbf{\Phi}(t, t_0)\ \mathbf{X}(t_0) \tag{5.12}$$

This mapping can be used to construct the partial derivatives of the match point constraints with respect to the decision variables that form the problem Jacobian. The chain rule allows multiplication of these matrices to be transitive, i.e. for three distinct time epochs t_0, t_1 and t_2:

$$\mathbf{\Phi}(t_2, t_0) = \mathbf{\Phi}(t_2, t_1) \cdot \mathbf{\Phi}(t_1, t_0) \tag{5.13}$$

It is this important property of the STM that allows for the computation of the match point constraint derivatives for both MGALT and FBLT. State transition matrix multiplication sweeps are also an integral part of some differential dynamic programming methods such as hybrid differential dynamic programming (HDDP) developed by Lantoine and Russell [44, 45]. In order to calculate the match point derivatives, the spacecraft's state transition matrix (STM) must be computed over the course of each FBLT segment as shown in Figure 5.3.

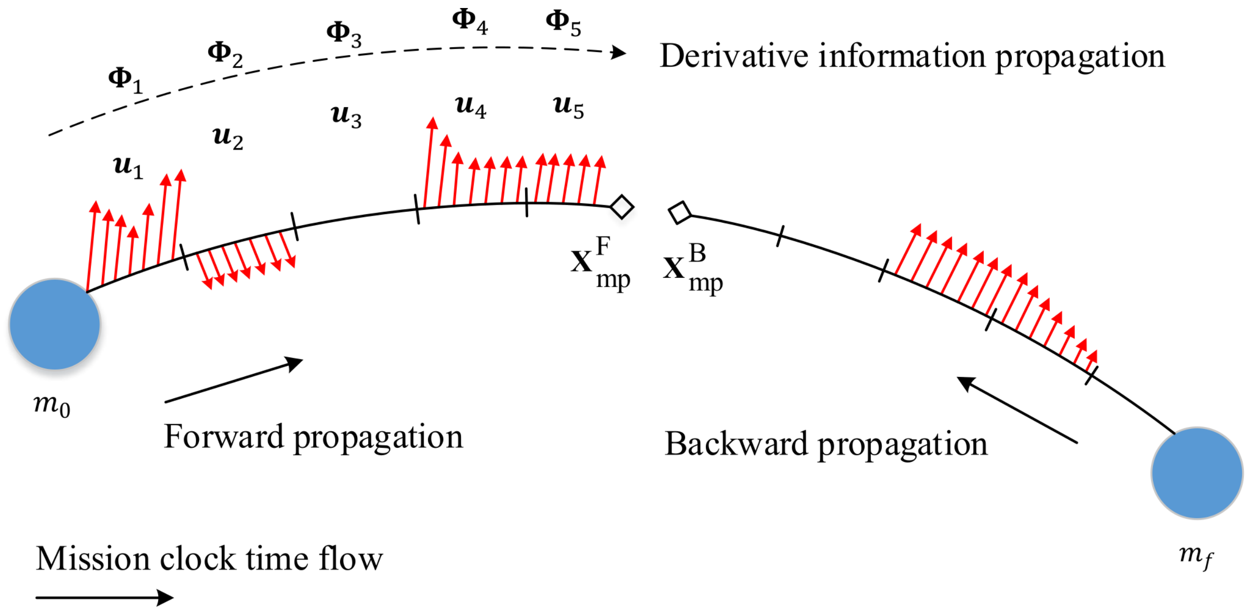


Figure 5.3: An FBLT phase with unoptimized match point constraint. Along with the spacecraft state, the entries of N STMs are also computed.

Since the three control parameters that determine the thrust level and direction in each FBLT segment u_x, u_y and u_z, and the mass at the end of each phase m_f, are all NLP decision variables, the traditional six-by-six perturbation transition matrix [160] is augmented to include entries describing the first-order sensitivities of the spacecraft's state at some epoch t due to changes in these variables at some initial epoch t_0 [189]:

$$\mathbf{\Phi}(t, t_0) = \left[\begin{array}{cc|cc} \frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{r}}{\partial \dot{\mathbf{r}}_0} & \frac{\partial \mathbf{r}}{\partial m_0} & \frac{\partial \mathbf{r}}{\partial \mathbf{u}_0} \\ \frac{\partial \dot{\mathbf{r}}}{\partial \mathbf{r}_0} & \frac{\partial \dot{\mathbf{r}}}{\partial \dot{\mathbf{r}}_0} & \frac{\partial \dot{\mathbf{r}}}{\partial m_0} & \frac{\partial \dot{\mathbf{r}}}{\partial \mathbf{u}_0} \\ \cline{1-2} \frac{\partial m}{\partial \mathbf{r}_0} & \frac{\partial m}{\partial \dot{\mathbf{r}}_0} & \frac{\partial m}{\partial m_0} & \frac{\partial m}{\partial \mathbf{u}_0} \\ \frac{\partial \mathbf{u}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{u}}{\partial \dot{\mathbf{r}}_0} & \frac{\partial \mathbf{u}}{\partial m_0} & \frac{\partial \mathbf{u}}{\partial \mathbf{u}_0} \end{array}\right] \tag{5.14}$$

This augmented STM forms the basis for all match point partial derivative computations, except for those associated with the phase times of flight or the launch epoch. Those derivatives are discussed in a later section.

As an example of how these augmented STMs are used to compute match point derivatives, consider the case for decision vector entries $p = u_x$, u_y, or u_z. These partial derivatives account for the vast majority of the dense Jacobian entries and their computation alone affords substantial computational speed-up during SNOPT's local optimization procedure. Suppose that the partial derivative of the match point constraint with respect to the control variables in the i^{th} segment are desired. First, the STM immediately adjacent to the match point has its right three columns zeroed/stripped:

$$\mathbf{\Phi}_s(t,t_0) = \left[\begin{array}{cc|cc} \frac{\partial \mathbf{r}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{r}}{\partial \dot{\mathbf{r}}_0} & \frac{\partial \mathbf{r}}{\partial m_0} & \mathbf{0}_{3\times 3} \\ \frac{\partial \dot{\mathbf{r}}}{\partial \mathbf{r}_0} & \frac{\partial \dot{\mathbf{r}}}{\partial \dot{\mathbf{r}}_0} & \frac{\partial \dot{\mathbf{r}}}{\partial m_0} & \mathbf{0}_{3\times 3} \\ \hline \frac{\partial m}{\partial \mathbf{r}_0} & \frac{\partial m}{\partial \dot{\mathbf{r}}_0} & \frac{\partial m}{\partial m_0} & \mathbf{0}_{1\times 3} \\ \frac{\partial \mathbf{u}}{\partial \mathbf{r}_0} & \frac{\partial \mathbf{u}}{\partial \dot{\mathbf{r}}_0} & \frac{\partial \mathbf{u}}{\partial m_0} & \mathbf{0}_{3\times 3} \end{array}\right] \tag{5.15}$$

This STM is then right-multiplied by the next segment's "stripped" STM. This procedure proceeds sequentially until the segment of interest is reached at which point the final unaltered STM from Eq. (5.14) associated with the k^{th} segment is right-multiplied to the resultant "stripped" STM.

$$\mathbf{\Phi}(t_{\text{mp}}, t_k) = \mathbf{\Phi}_s(t_{\text{mp}}, t_{\text{mp}-1}) \cdot \ldots \cdot \mathbf{\Phi}_s(t_{k+2}, t_{k+1}) \cdot \mathbf{\Phi}(t_{k+1}, t_k) \tag{5.16}$$

In Eq. (5.16) t_k refers to the epoch at the beginning of the k^{th} segment and t_{mp} refers to the epoch at the match point. The resultant matrix $\mathbf{\Phi}(t_{\text{mp}}, t_k)$ contains explicit Jacobian entries for the match point constraint with respect to the control vector from the k^{th} segment $\mathbf{u}_k$.

5.3.2 Numerical Computation of the FBLT State Transition Matrix

Since the FBLT transcription is comprised of piecewise numerically integrated arcs, the unknown STM entries must be computed via integration as well [190, 191]. Entries of the STM $\mathbf{\Phi}$ are computed with the following matrix differential equation:

$$\dot{\mathbf{\Phi}} = \mathbf{A}\mathbf{\Phi} \tag{5.17}$$

where $\mathbf{A} = \frac{\partial \dot{\mathbf{X}}}{\partial \mathbf{X}}$ is the state propagation matrix. The differential equations in Eq. (5.17) are collectively known as the variational equations. A similar approach for the numerical computation of finite-burn state transition matrices was employed by Ocampo and Munoz [192] as well as Russell [73] in indirect trajectory optimization frameworks and Ocampo and Mathur [193] for a direct method. Equation (5.17) is also known as the fundamental set or the fundamental solution and has long been used for low-thrust spacecraft guidance applications [194].

$$\mathbf{A} = \left[\begin{array}{ccc|ccc|c|ccc}
\frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} & \frac{\partial v_x}{\partial v_x} & \frac{\partial v_x}{\partial v_y} & \frac{\partial v_x}{\partial v_z} & \frac{\partial v_x}{\partial m} & \frac{\partial v_x}{\partial u_x} & \frac{\partial v_x}{\partial u_y} & \frac{\partial v_x}{\partial u_z} \\
\frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} & \frac{\partial v_y}{\partial v_x} & \frac{\partial v_y}{\partial v_y} & \frac{\partial v_y}{\partial v_z} & \frac{\partial v_y}{\partial m} & \frac{\partial v_y}{\partial u_x} & \frac{\partial v_y}{\partial u_y} & \frac{\partial v_y}{\partial u_z} \\
\frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} & \frac{\partial v_z}{\partial v_x} & \frac{\partial v_z}{\partial v_y} & \frac{\partial v_z}{\partial v_z} & \frac{\partial v_z}{\partial m} & \frac{\partial v_z}{\partial u_x} & \frac{\partial v_z}{\partial u_y} & \frac{\partial v_z}{\partial u_z} \\
\hline
\frac{\partial a_x}{\partial x} & \frac{\partial a_x}{\partial y} & \frac{\partial a_x}{\partial z} & \frac{\partial a_x}{\partial v_x} & \frac{\partial a_x}{\partial v_y} & \frac{\partial a_x}{\partial v_z} & \frac{\partial a_x}{\partial m} & \frac{\partial a_x}{\partial u_x} & \frac{\partial a_x}{\partial u_y} & \frac{\partial a_x}{\partial u_z} \\
\frac{\partial a_y}{\partial x} & \frac{\partial a_y}{\partial y} & \frac{\partial a_y}{\partial z} & \frac{\partial a_y}{\partial v_x} & \frac{\partial a_y}{\partial v_y} & \frac{\partial a_y}{\partial v_z} & \frac{\partial a_y}{\partial m} & \frac{\partial a_y}{\partial u_x} & \frac{\partial a_y}{\partial u_y} & \frac{\partial a_y}{\partial u_z} \\
\frac{\partial a_z}{\partial x} & \frac{\partial a_z}{\partial y} & \frac{\partial a_z}{\partial z} & \frac{\partial a_z}{\partial v_x} & \frac{\partial a_z}{\partial v_y} & \frac{\partial a_z}{\partial v_z} & \frac{\partial a_z}{\partial m} & \frac{\partial a_z}{\partial u_x} & \frac{\partial a_z}{\partial u_y} & \frac{\partial a_z}{\partial z} \\
\hline
\frac{\partial \dot{m}}{\partial x} & \frac{\partial \dot{m}}{\partial y} & \frac{\partial \dot{m}}{\partial z} & \frac{\partial \dot{m}}{\partial v_x} & \frac{\partial \dot{m}}{\partial v_y} & \frac{\partial \dot{m}}{\partial v_z} & \frac{\partial \dot{m}}{\partial m} & \frac{\partial \dot{m}}{\partial u_x} & \frac{\partial \dot{m}}{\partial u_y} & \frac{\partial \dot{m}}{\partial u_z} \\
\hline
\frac{\partial \dot{u}_x}{\partial x} & \frac{\partial \dot{u}_x}{\partial y} & \frac{\partial \dot{u}_x}{\partial z} & \frac{\partial \dot{u}_x}{\partial v_x} & \frac{\partial \dot{u}_x}{\partial v_y} & \frac{\partial \dot{u}_x}{\partial v_z} & \frac{\partial \dot{u}_x}{\partial m} & \frac{\partial \dot{u}_x}{\partial u_x} & \frac{\partial \dot{u}_x}{\partial u_y} & \frac{\partial \dot{u}_x}{\partial u_z} \\
\frac{\partial \dot{u}_y}{\partial x} & \frac{\partial \dot{u}_y}{\partial y} & \frac{\partial \dot{u}_y}{\partial z} & \frac{\partial \dot{u}_y}{\partial v_x} & \frac{\partial \dot{u}_y}{\partial v_y} & \frac{\partial \dot{u}_y}{\partial v_z} & \frac{\partial \dot{u}_y}{\partial m} & \frac{\partial \dot{u}_y}{\partial u_x} & \frac{\partial \dot{u}_y}{\partial u_y} & \frac{\partial \dot{u}_y}{\partial u_z} \\
\frac{\partial \dot{u}_z}{\partial x} & \frac{\partial \dot{u}_z}{\partial y} & \frac{\partial \dot{u}_z}{\partial z} & \frac{\partial \dot{u}_z}{\partial v_x} & \frac{\partial \dot{u}_z}{\partial v_y} & \frac{\partial \dot{u}_z}{\partial v_z} & \frac{\partial \dot{u}_z}{\partial m} & \frac{\partial \dot{u}_z}{\partial u_x} & \frac{\partial \dot{u}_z}{\partial u_y} & \frac{\partial \dot{u}_z}{\partial z}
\end{array}\right] = \left[\begin{array}{c|c|c|c}
\mathbf{0}_{3\times3} & \mathbb{I}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\
\hline
\mathbf{A}_{21} & \mathbf{0}_{3\times3} & \mathbf{A}_{23} & \mathbf{A}_{24} \\
\hline
\mathbf{A}_{31} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times1} & \mathbf{A}_{34} \\
\hline
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3}
\end{array}\right] \tag{5.18}$$

Expressions for the partitions $\mathbf{A}_{21}$, $\mathbf{A}_{23}$, $\mathbf{A}_{24}$ and $\mathbf{A}_{41}$ are found by taking partial derivatives of Equations (5.1) and (5.2).

Letting $\mu_{\text{cb}} = Gm_{\text{cb}}$ and $\mu_i = Gm_i$

$$\begin{aligned}
\mathbf{A}_{21} &= \frac{\partial \ddot{\mathbf{r}}}{\partial \mathbf{r}} \\
&= \frac{3\mu_{cb}\mathbf{r}\mathbf{r}^T}{r^5} - \frac{\mu_{cb}}{r^3}\mathbb{I}_{3x3} + \frac{3\mu_i \ \mathbf{r}_{s/i}\mathbf{r}_{s/i}^T}{r_{s/i}^5} - \frac{\mu_i}{r_{s/i}^3}\mathbb{I}_{3x3} \\
&- \frac{3C_r A_s K \phi}{c \ m} \frac{\mathbf{r}_{s/\odot}\mathbf{r}_{s/\odot}^T}{r_{s/\odot}^5} + \frac{1}{r_{s/\odot}^3}\mathbb{I}_{3x3} \\
&+ \frac{\partial \ddot{\mathbf{r}}_{J_2}}{\partial \mathbf{r}} \\
&+ \frac{\mathbf{u} \ D}{m_s} \cdot \frac{\partial T}{\partial P} \cdot \frac{\partial P}{\partial r_{s/\odot}} \cdot \frac{\partial r_{s/\odot}}{\partial \mathbf{r}}
\end{aligned} \tag{5.19}$$

where

$$\frac{\partial \ddot{\mathbf{r}}_{J_2}}{\partial \mathbf{r}} = \frac{\partial \ddot{\mathbf{r}}_{J_{2_{ICRF}}}}{\partial \mathbf{r}_{ICRF}} = \frac{\partial \ddot{\mathbf{r}}_{J_{2_{ICRF}}}}{\partial \ddot{\mathbf{r}}_{J_{2_{BCF}}}} \frac{\partial \ddot{\mathbf{r}}_{J_{2_{BCF}}}}{\partial \mathbf{r}_{BCF}} \frac{\partial \mathbf{r}_{BCF}}{\partial \mathbf{r}_{ICRF}} \tag{5.20}$$

and

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \tag{5.21}$$

and

$$\frac{\partial r_{s/\odot}}{\partial \mathbf{r}} = \begin{bmatrix} \frac{\partial r_{s/\odot}}{\partial x} & \frac{\partial r_{s/\odot}}{\partial y} & \frac{\partial r_{s/\odot}}{\partial z} \end{bmatrix} \tag{5.22}$$

Note that $\frac{\partial \ddot{\mathbf{r}}_{J_{2_{ICRF}}}}{\partial \ddot{\mathbf{r}}_{J_{2_{BCF}}}}$ is the rotation matrix from BCF→ICRF and $\frac{\partial \mathbf{r}_{BCF}}{\partial \mathbf{r}_{ICRF}}$ is the rotation matrix from ICRF→BCF.

The J_2 partials in the BCF frame $\left(\frac{\partial \ddot{\mathbf{r}}_{J_{2_{BCF}}}}{\partial \mathbf{r}_{BCF}}\right)$ are computed as follows (where the BCF frame subscript has been omitted):

$$\begin{aligned}
\frac{\partial \ddot{x}_{J_2}}{\partial x} &= \frac{3J_2\mu R^2}{2r^9}(4x^4 + 3x^2y^2 - 27x^2z^2 - y^4 + 3y^2z^2 + 4z^44) \\
\frac{\partial \ddot{x}_{J_2}}{\partial y} &= \frac{3J_2\mu R^2}{2r^9}xy(x^2 + y^2 - 6z^2) \\
\frac{\partial \ddot{x}_{J_2}}{\partial z} &= \frac{3J_2\mu R^2}{2r^9}xz(3x^2 + 3y^2 - 4z^2) \\
\frac{\partial \ddot{y}_{J_2}}{\partial x} &= \frac{3J_2\mu R^2}{2r^9}xy(x^2 + y^2 - 6z^2) \\
\frac{\partial \ddot{y}_{J_2}}{\partial y} &= \frac{3J_2\mu R^2}{2r^9}(-x^4 + 3x^2y^2 + 3x^2z^2 + 4y^4 - 27y^2z^2 + 4z^4) \\
\frac{\partial \ddot{y}_{J_2}}{\partial z} &= \frac{3J_2\mu R^2}{2r^9}yz(3x^2 + 3y^2 - 4z^2) \\
\frac{\partial \ddot{z}_{J_2}}{\partial x} &= \frac{3J_2\mu R^2}{2r^9}xz(3x^2 + 3y^2 - 4z^2) \\
\frac{\partial \ddot{z}_{J_2}}{\partial y} &= \frac{3J_2\mu R^2}{2r^9}yz(3x^2 + 3y^2 - 4z^2) \\
\frac{\partial \ddot{z}_{J_2}}{\partial z} &= \frac{3J_2\mu R^2}{2r^9}(3x^4 + 6x^2y^2 - 24x^2z^2 + 3y^4 - 24y^2z^2 + 8z^4)
\end{aligned}$$

The entries in Equation 5.22 are computed by taking partial derivatives of Equation 5.6.

$$\mathbf{A}_{23} = \frac{\partial \ddot{\mathbf{r}}}{\partial m} = \frac{-G\ \mathbf{r}}{r^3} - \frac{\mathbf{u}\ D\ T}{m^2} + \frac{C_r A_s K \phi}{c\ m^2\ r^2_{s/\odot}} \frac{\mathbf{r}_{s/\odot}}{r_{s/\odot}} \tag{5.23}$$

Since we are neglecting the mass of the spacecraft in the first term of Equation (5.1), the partial derivative reduces to:

$$\mathbf{A}_{23} = \frac{\partial \ddot{\mathbf{r}}}{\partial m} = -\frac{\mathbf{u}\ D\ T}{m^2} + \frac{C_r A_s K \phi}{c\ m^2\ r^2_{s/\odot}} \frac{\mathbf{r}_{s/\odot}}{r_{s/\odot}} \tag{5.24}$$

$$\frac{\partial \ddot{\mathbf{r}}}{\partial \mathbf{u}} = \frac{D\ T}{m} \mathbb{I}_{3x3} \tag{5.25}$$

$$\mathbf{A}_{31} = \left(\frac{\partial \dot{m}}{\partial \mathbf{r}}\right)^T = \left[-\|\mathbf{u}\|\ D\ \frac{\partial \dot{m}_{max}}{\partial P} \cdot \frac{\partial P}{\partial r_{s/\odot}} \cdot \frac{\partial r_{s/\odot}}{\partial \mathbf{r}}\right]^T \tag{5.26}$$

$$\mathbf{A}_{34} = \left(\frac{\partial \dot{m}}{\partial \mathbf{u}}\right)^T = \left[-\frac{\partial \|\mathbf{u}\|}{\partial \mathbf{u}}\ D\ \dot{m}_{max}\right]^T \tag{5.27}$$

where,

$$\frac{\partial \|\mathbf{u}\|}{\partial \mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\| + \epsilon} \tag{5.28}$$

The quantity $\epsilon = 1.0 \times 10^{-25}$ prevents a singularity in the event that the thruster is turned off by the optimizer. The $m \times m$ equations of Equation (5.17) are appended to the state vector and all $k = m + m \times m = 110$ differential equations are integrated alongside the other state vector entries:

$$\dot{\mathbf{X}} = \mathbf{f} = \begin{bmatrix} \dot{\mathbf{r}} \\ \ddot{\mathbf{r}} \\ \dot{m} \\ \dot{u}_x \\ \dot{u}_y \\ \dot{u}_z \\ \dot{s}_{11} \\ \dot{s}_{12} \\ \vdots \\ \dot{s}_{1010} \end{bmatrix} \tag{5.29}$$

The s_{ij} are defined using:

$$\mathbf{\Phi} = \begin{bmatrix}
\frac{\partial x}{\partial x_0} & \frac{\partial x}{\partial y_0} & \frac{\partial x}{\partial z_0} & \frac{\partial x}{\partial v_{x_0}} & \frac{\partial x}{\partial v_{y_0}} & \frac{\partial x}{\partial v_{z_0}} & \frac{\partial x}{\partial m_0} & \frac{\partial x}{\partial u_{x_0}} & \frac{\partial x}{\partial u_{y_0}} & \frac{\partial x}{\partial u_{z_0}} \\
\frac{\partial y}{\partial x_0} & \frac{\partial y}{\partial y_0} & \frac{\partial y}{\partial z_0} & \frac{\partial y}{\partial v_{x_0}} & \frac{\partial y}{\partial v_{y_0}} & \frac{\partial y}{\partial v_{z_0}} & \frac{\partial y}{\partial m_0} & \frac{\partial y}{\partial u_{x_0}} & \frac{\partial y}{\partial u_{y_0}} & \frac{\partial y}{\partial u_{z_0}} \\
\frac{\partial z}{\partial x_0} & \frac{\partial z}{\partial y_0} & \frac{\partial z}{\partial z_0} & \frac{\partial z}{\partial v_{x_0}} & \frac{\partial z}{\partial v_{y_0}} & \frac{\partial z}{\partial v_{z_0}} & \frac{\partial z}{\partial m_0} & \frac{\partial z}{\partial u_{x_0}} & \frac{\partial z}{\partial u_{y_0}} & \frac{\partial z}{\partial u_{z_0}} \\
\frac{\partial v_x}{\partial x_0} & \frac{\partial v_x}{\partial y_0} & \frac{\partial v_x}{\partial z_0} & \frac{\partial v_x}{\partial v_{x_0}} & \frac{\partial v_x}{\partial v_{y_0}} & \frac{\partial v_x}{\partial v_{z_0}} & \frac{\partial v_x}{\partial m_0} & \frac{\partial v_x}{\partial u_{x_0}} & \frac{\partial v_x}{\partial u_{y_0}} & \frac{\partial v_x}{\partial u_{z_0}} \\
\frac{\partial v_y}{\partial x_0} & \frac{\partial v_y}{\partial y_0} & \frac{\partial v_y}{\partial z_0} & \frac{\partial v_y}{\partial v_{x_0}} & \frac{\partial v_y}{\partial v_{y_0}} & \frac{\partial v_y}{\partial v_{z_0}} & \frac{\partial v_y}{\partial m_0} & \frac{\partial v_y}{\partial u_{x_0}} & \frac{\partial v_y}{\partial u_{y_0}} & \frac{\partial v_y}{\partial u_{z_0}} \\
\frac{\partial v_z}{\partial x_0} & \frac{\partial v_z}{\partial y_0} & \frac{\partial v_z}{\partial z_0} & \frac{\partial v_z}{\partial v_{x_0}} & \frac{\partial v_z}{\partial v_{y_0}} & \frac{\partial v_z}{\partial v_{z_0}} & \frac{\partial v_z}{\partial m_0} & \frac{\partial v_z}{\partial u_{x_0}} & \frac{\partial v_z}{\partial u_{y_0}} & \frac{\partial v_z}{\partial u_{z_0}} \\
\frac{\partial m}{\partial x_0} & \frac{\partial m}{\partial y_0} & \frac{\partial m}{\partial z_0} & \frac{\partial m}{\partial v_{x_0}} & \frac{\partial m}{\partial v_{y_0}} & \frac{\partial m}{\partial v_{z_0}} & \frac{\partial m}{\partial m_0} & \frac{\partial m}{\partial u_{x_0}} & \frac{\partial m}{\partial u_{y_0}} & \frac{\partial m}{\partial u_{z_0}} \\
\frac{\partial u_x}{\partial x_0} & \frac{\partial u_x}{\partial y_0} & \frac{\partial u_x}{\partial z_0} & \frac{\partial u_x}{\partial v_{x_0}} & \frac{\partial u_x}{\partial v_{y_0}} & \frac{\partial u_x}{\partial v_{z_0}} & \frac{\partial u_x}{\partial m_0} & \frac{\partial u_x}{\partial u_{x_0}} & \frac{\partial u_x}{\partial u_{y_0}} & \frac{\partial u_x}{\partial u_{z_0}} \\
\frac{\partial u_y}{\partial x_0} & \frac{\partial u_y}{\partial y_0} & \frac{\partial u_y}{\partial z_0} & \frac{\partial u_y}{\partial v_{x_0}} & \frac{\partial u_y}{\partial v_{y_0}} & \frac{\partial u_y}{\partial v_{z_0}} & \frac{\partial u_y}{\partial m_0} & \frac{\partial u_y}{\partial u_{x_0}} & \frac{\partial u_y}{\partial u_{y_0}} & \frac{\partial u_y}{\partial u_{z_0}} \\
\frac{\partial u_z}{\partial x_0} & \frac{\partial u_z}{\partial y_0} & \frac{\partial u_z}{\partial z_0} & \frac{\partial u_z}{\partial v_{x_0}} & \frac{\partial u_z}{\partial v_{y_0}} & \frac{\partial u_z}{\partial v_{z_0}} & \frac{\partial u_z}{\partial m_0} & \frac{\partial u_z}{\partial u_{x_0}} & \frac{\partial u_z}{\partial u_{y_0}} & \frac{\partial u_z}{\partial u_{z_0}}
\end{bmatrix} \tag{5.30}$$

$$= \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} & s_{15} & s_{16} & s_{17} & s_{18} & s_{19} & s_{110} \\ s_{21} & s_{22} & s_{23} & s_{24} & s_{25} & s_{26} & s_{27} & s_{28} & s_{29} & s_{210} \\ s_{31} & s_{32} & s_{33} & s_{34} & s_{35} & s_{36} & s_{37} & s_{38} & s_{39} & s_{310} \\ s_{41} & s_{42} & s_{43} & s_{44} & s_{45} & s_{46} & s_{47} & s_{48} & s_{49} & s_{410} \\ s_{51} & s_{52} & s_{53} & s_{54} & s_{55} & s_{56} & s_{57} & s_{58} & s_{59} & s_{510} \\ s_{61} & s_{62} & s_{63} & s_{64} & s_{65} & s_{66} & s_{67} & s_{68} & s_{69} & s_{610} \\ s_{71} & s_{72} & s_{73} & s_{74} & s_{75} & s_{76} & s_{77} & s_{78} & s_{79} & s_{710} \\ s_{81} & s_{82} & s_{83} & s_{84} & s_{85} & s_{86} & s_{87} & s_{88} & s_{89} & s_{810} \\ s_{91} & s_{92} & s_{93} & s_{94} & s_{95} & s_{96} & s_{97} & s_{98} & s_{99} & s_{910} \\ s_{101} & s_{102} & s_{103} & s_{104} & s_{105} & s_{106} & s_{107} & s_{108} & s_{109} & s_{1010} \end{bmatrix} \tag{5.31}$$

In FBLT, the thrust vector is held constant over one segment (both magnitude and direction), i.e.

$$\begin{aligned} \dot{u}_x &= 0; \quad u_x(0) = u_x \\ \dot{u}_y &= 0; \quad u_y(0) = u_y \\ \dot{u}_z &= 0; \quad u_z(0) = u_z \end{aligned} \tag{5.32}$$

The STM matrix integration proceeds from the initial conditions:

$$\dot{s}_{ij}(0) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{5.33}$$

5.4 FBLT Match Point Phase Time of Flight Gradients

Like the MGALT transcription, FBLT encodes the phase flight time as a decision variable. The match point constraints are completely dense with respect to the flight time decision variable, therefore the NLP solver requires the gradient of those constraints with respect to the flight time. Conceptually, these gradients are quite simple. The perturbation of the spacecraft's state vector at the match point due to a perturbation in the time of flight (Δt_p) is given by the time differential of the state vector. For example, in order to determine how the spacecraft's position vector at the match point $\mathbf{r}_m$ would be altered by a slight change in the time of flight (to first order), one would just multiply the spacecraft's velocity at the match point $\mathbf{v}_{\text{mp}}$ by the time-of-flight perturbation.

$$\delta\mathbf{r}_{\text{mp}} = \mathbf{v}_{\text{mp}} \cdot \delta\Delta t_p \tag{5.34}$$

However, FBLT does not propagate the spacecraft's state analytically, but rather numerically, therefore, the match point defect gradients must be computed by taking the partial derivative of Eq. (5.7) with respect to the time of flight.

$$\frac{\partial\hat{\mathbf{X}}_{n+1}}{\partial\Delta t_p} = \frac{\partial\hat{\mathbf{X}}_n}{\partial\Delta t_p} + \sum_{i=1}^{s} \hat{b}_i \frac{\partial\mathbf{k}_i}{\partial\Delta t_p} \tag{5.35}$$

where

$$\frac{\partial\mathbf{k}_i}{\partial\Delta t_p} = \frac{\partial h}{\partial\Delta t_p}\mathbf{f} + h\frac{\partial\mathbf{f}}{\partial\Delta t_p} \tag{5.36}$$

For a fixed-step integrator $\frac{\partial h}{\partial\Delta t_p} = 0$, except for the partial segment leading into the match point. The partial derivative of $\mathbf{k}_i$ (specifically $\mathbf{f}$) with respect to Δt_p must be computed at each intermediate stage of the RK step and stored in memory as Eq. (5.35) includes their summation across all s stages. These partial derivatives are most efficiently computed alongside the vector $\mathbf{f}$. We are only interested in calculating the partial derivatives with respect to Δt_p of the first seven entries in the $\mathbf{f}$ vector (i.e. $\dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}$ and $\dot{m}$, the time derivatives of the match point states). To facilitate the computation of $\frac{\partial\mathbf{f}}{\partial\Delta t_p}$, the Δt_p gradient of the state vector approximation at each stage must be computed (i.e. the partial derivative of Eq. (5.11) is required):

$$\frac{\partial\hat{\mathbf{X}}_{n(i)}}{\partial\Delta t_p} = \frac{\partial\hat{\mathbf{X}}_n}{\partial\Delta t_p} + \sum_{j=1}^{i-1} a_{ij} \frac{\mathbf{k}_j}{\partial\Delta t_p} \tag{5.37}$$

The first three entries in the vector $\frac{\partial\mathbf{f}}{\partial\Delta t_p}$ are given by the fourth, fifth and sixth entries in Eq. (5.37). To simplify notation, it is assumed that the following partial derivatives are all computed in the i^{th} stage of the n^{th} RK sub-step and these subscripts have been omitted from the following partial derivative calculations.

$$\frac{\partial\dot{\mathbf{r}}}{\partial\Delta t_p} = \frac{\partial\hat{\mathbf{X}}_{4,5,6}}{\partial\Delta t_p} \tag{5.38}$$

In order to compute the sensitivity of the spacecraft's acceleration with respect to the time of flight, the chain rule is applied to Eq. (5.1):

$$\frac{\partial \ddot{\mathbf{r}}}{\partial \Delta t_p} = \frac{\partial \ddot{\mathbf{r}}}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \Delta t_p} + \sum_i \frac{\partial \ddot{\mathbf{r}}}{\partial \mathbf{r}_i} \cdot \frac{\partial \mathbf{r}_i}{\partial \Delta t_p} + \frac{\partial \ddot{\mathbf{r}}}{\partial m} \cdot \frac{\partial m}{\partial \Delta t_p} \tag{5.39}$$

The first term of Eq. (5.39) is computed using Equations (5.19) and (5.37) for the two partial derivatives respectively. The first parital derivative of the second term of Eq. (5.39) is computed as follows:

$$\frac{\partial \ddot{\mathbf{r}}}{\partial \mathbf{r}_i} = -\mu_i \left(\frac{3\mathbf{r}_{s/i}\mathbf{r}_{s/i}^T}{r_{s/i}^5} - \frac{\mathbb{I}_{3x3}}{r_{s/i}^3} - \frac{3\mathbf{r}_i\mathbf{r}_i^T}{r_i^5} + \frac{\mathbb{I}_{3x3}}{r_i^3} \right) \tag{5.40}$$

It is important to note that, for the special where body i is the sun, Equation (5.40) has an additional term:

$$\begin{aligned} \frac{\partial \ddot{\mathbf{r}}}{\partial \mathbf{r}_\odot} = &- \mu_\odot \left(\frac{3\mathbf{r}_{s/\odot}\mathbf{r}_{s/\odot}^T}{r_{s/\odot}^5} - \frac{\mathbb{I}_{3x3}}{r_{s/\odot}^3} - \frac{3\mathbf{r}_\odot\mathbf{r}_\odot^T}{r_\odot^5} + \frac{\mathbb{I}_{3x3}}{r_\odot^3} \right) \\ &+ \frac{3C_r A_s K \phi}{c\ m} \frac{\mathbf{r}_{s/\odot}\mathbf{r}_{s/\odot}^T}{r_{s/\odot}^5} - \frac{1}{r_{s/\odot}^3}\mathbb{I}_{3x3} \\ &+ \frac{\mathbf{u}\ D}{m_s} \cdot \frac{\partial T}{\partial P} \cdot \frac{\partial P}{\partial r_{s/\odot}} \cdot \frac{\partial r_{s/\odot}}{\partial \mathbf{r}_\odot} \end{aligned} \tag{5.41}$$

This additional term is present since a change in the position of the spacecraft with respect to the Sun affects not only their gravitational interaction, but also the power generated by the solar arrays of the spacecraft, and hence the maximum acceleration of the spacecraft due to the thrusters and solar pressure. At first glance it seems as though Eq. (5.41) and Eq. (5.19) are double-counting some terms. To resolve this, one need just examine the partial derivative directions of the spaceraft's thrust with respect to time-of-flight:

$$\frac{\partial T}{\partial \Delta t_p} = \frac{\partial T}{\partial P} \cdot \frac{\partial P}{\partial \Delta t_p} \tag{5.42}$$

where

$$\begin{aligned}
\frac{\partial P}{\partial \Delta t_p} &= \frac{\partial P}{\partial r_{s/\odot}} \cdot \frac{\partial r_{s/\odot}}{\partial \Delta t_p} + \frac{\partial P}{\partial t} \cdot \frac{\partial t}{\partial \Delta t_p} \\
&= \frac{\partial P}{\partial r_{s/\odot}} \cdot \left(\frac{\partial r_{s/\odot}}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \Delta t_p} - \frac{\partial r_{s/\odot}}{\partial \mathbf{r}_\odot} \cdot \frac{\partial \mathbf{r}_\odot}{\partial \Delta t_p} \right) + \frac{\partial P}{\partial t} \cdot \frac{\partial t}{\partial \Delta t_p}
\end{aligned} \tag{5.43}$$

The first term in Eq. (5.43) is accounted for in Eq. (5.19), the second term is accounted for in Eq. (5.41) and the final term is the explicit time derivative of power. This last term is non-zero if hardware decay is accounted for. The third term in Eq. (5.39) uses Eq. (5.24) and (5.37). Finally, the phase flight time sensitivity of the mass flow rate differential equation must be computed:

$$\frac{\partial \dot{m}}{\partial \Delta t_p} = -\|\mathbf{u}\| \; D \; \frac{\partial \dot{m}_{max}}{\partial P} \cdot \frac{\partial P}{\partial \Delta t_p} \tag{5.44}$$

where the maximum mass flow rate time-of-flight gradient is calculated similarly to Eq. (5.42).

5.5 Example: Finite-Burn Comet Sample Return

The typical use case for the FBLT model is to locally re-converge a low-fidelity bounded-impulse initial guess. The FBLT optimization does not usually require MBH to re-converge an MGALT initial guess. This is especially true for interplanetary trajectories where perturbative accelerations are relatively small in magnitude. Switching the propagation method from two-body to numerical integration does however result in a sharp increase in execution time. For these reasons, avoiding finite differences is primarily important from a runtime perspective. Since the MGALT initial guess is fairly close to the finite-burn, n-body solution, the optimizer is typically robust enough to converge when using a finite-differenced Jacobian. The runtime cost of evaluating the finite-burn objective function can render the FBLT transcription impractical for a low-thrust transfer with many control segments and a complex dynamics model.

In order to quantify the advantages of computing the FBLT Jacobian with the variational equations, the comet sample return example from section 3.11 is re-converged with an n-body gravity model and a cannonball SRP model. The mission parameters specific to the FBLT optimization are shown in Table 5.1. Two one hour runs were executed: one using the variational equations to compute the STMs/Jacobian and one using finite differencing to do so. Information associated with the two runs is provided in Table 5.2. The first timing metric is how long it took SNOPT to complete a single major iteration. For this first major iteration, 99 minor iterations were required for both Jacobian calculation strategies. The variational equations improve the execution speed for this problem by over two orders of magnitude allowing for the optimizer to improve the problem feasibility metric. The finite differencing run was not allowed to iterate beyond one hour, so it is not known if it would achieve the same feasibility. The locally optimal solution found by SNOPT (using the variational equations) is shown in Fig. 5.4.

Table 5.1: Physics model parameters for the 67P comet sample return mission.

Parameter	Value
Point mass gravity sources	Mercury, Venus, Mars, Jupiter, Saturn, Uranus, Neptune
Propagator	fixed-step RK8
Integration step size	1 day
Spacecraft surface area	100 m^2
Coefficient of reflectivity	2
Solar flux	1360 W/m^2

Table 5.2: Physics model parameters for the 67P comet sample return mission.

Metric	Variational	FD
Time to execute 1 SNOPT major iteration	1.83 s	235 s
SNOPT major iterations completed in one hour	2111	7
NLP feasibility achieved	1e-15	1e-6

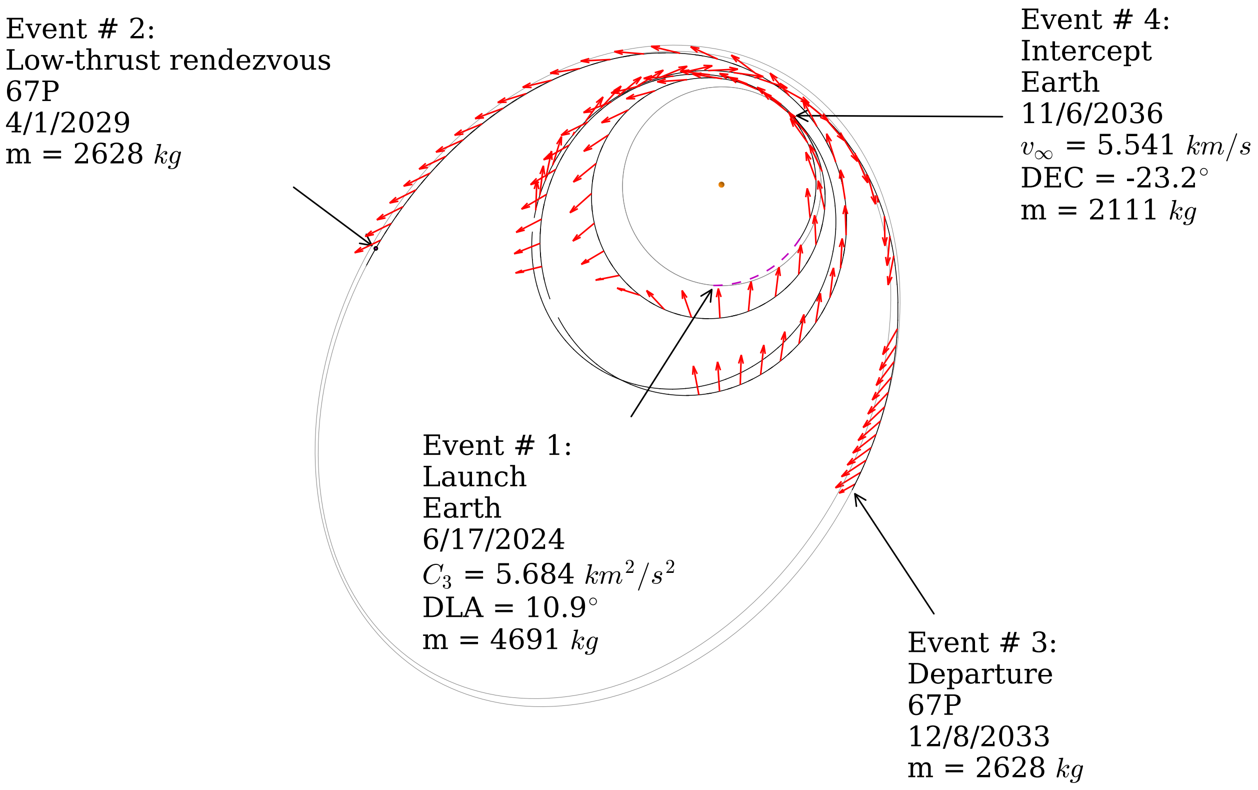


Figure 5.4: Optimal round-trip finite-burn trajectory to 67P.

An immediate observation is that closing the trajectory in the higher fidelity physics model has improved the cost function value over the MGALT solution shown in Fig. 3.21, with ten additional kilograms returned to Earth. One clear shortcoming of the modeling fidelity is that the Earth is not included as a perturbing point mass. The Earth is not included in the physics model due to the use of the patched conic approximation. Augmenting the FBLT transcription to model 3D flybys requires a straightforward modification to the current model and is a topic of immediate research.

5.6 FBLT Dynamical Model Verification and Validation Using MIRAGE

In this section, the extensions made to the FBLT [16] acceleration model that were presented in this chapter (third-body perturbations and solar radiation pressure) are verified for accuracy using the operational navigation tool MIRAGE [195]. MIRAGE is a licensed version of DPTRAJ/ODP originally developed at JPL to support high-fidelity orbit determination, trajectory propagation and maneuver planning [196]. MIRAGE has flight heritage from the 1970s to the present day and has been used to navigate many deep-space missions, including NEAR, Stardust, Genesis, and many other missions at JPL, and, more recently at KinetX, MESSENGER, New Horizons, and OSIRIS-REx. Most relevant to the analysis in this section, MIRAGE was used to navigate the Deep Space 1 mission, which employed solar electric propulsion. MIRAGE is therefore an ideal environment in which to check the accuracy of EMTG's FBLT mode.

5.6.1 Mission to 9712 Nauplius and 1143 Odysseus

A notional solar-electric propulsion (SEP) mission to the Trojan asteroids Nauplius and Odysseus is used as the benchmark case to compare EMTG and the software package MIRAGE. This benchmark was chosen for this analysis because the "Trojan Tour and Rendezvous" was listed as one of the acceptable targets in the New Frontiers 4 Announcement of Opportunity [197]. New Frontiers 4 step one has since been completed, however, a Trojan Tour and Rendezvous remains an ideal case with which to validate a new design tool that may be used to propose similar missions in the future. The Announcement of Opportunity states that the Trojan Tour and Rendezvous "is intended to examine two or more small bodies sharing the orbit of Jupiter, including one or more flybys followed by an extended rendezvous with a Trojan object." Accordingly, our benchmark mission performs a flyby of the magnitude 10.7 Trojan 9712 Nauplius before rendezvousing with 1143 Odysseus, which at magnitude 7.93 is one of the largest Trojans. The operations at Odysseus are not considered in this paper. This benchmark is appropriate for validating EMTG against MIRAGE and also provides a test case for an operational validation process that could be used for any mission or mission proposal.

The assumptions for the benchmark mission to Nauplius and Odysseus are listed in Table 5.3. Table 5.4 lists the major events in the mission, and Figure 5.5 is a plot of the optimal trajectory. The acceleration model settings used in the comparison are given in Table 5.5 provided in the next section.

Table 5.3: Assumptions for the benchmark mission to Nauplius and Odysseus

Option	Value
Launch window	1/1/2024 - 12/31/2024
Flight time upper bound	12 years
Arrival conditions	Nauplius: flyby Odysseus: rendezvous
Launch vehicle	Atlas V 551
Launch asymptote declination bounds	$[-28.5, 28.5]$ (Kennedy Space Center)
Post-launch/Pre-flyby/Post-flyby coast durations	60/30/30 days
Solar array P_0	40 kW
Solar array coefficients γ_i	$[1, 0, 0, 0, 0]$
Spacecraft power coefficients $a_{s/c} - c_{s/c}$	$[0.8, 0, 0]$
Propulsion system	2 NEXT in "high-Thrust" mode [198]
Throttle logic	minimum number of thrusters
Duty cycle	90%
Power margin	15%
Number of segments per phase	40

The NEXT thruster coefficients for use in Eq. (2.15) and (2.16) are provided in Table A.1. The multi-thruster logic used for this problem is provided in Algorithm2.

Table 5.4: Significant events in the Nauplius-Odysseus mission

Event #	Date	Event	Location	v_∞ (km/s)	Mass (kg)
1	12/31/2024	launch	Earth	5.587	3256.2
2	7/13/2033	intercept	9712 Nauplius	2.229	1852.4
3	12/31/2036	Low-thrust rendezvous	1143 Odysseus	–	1653.1

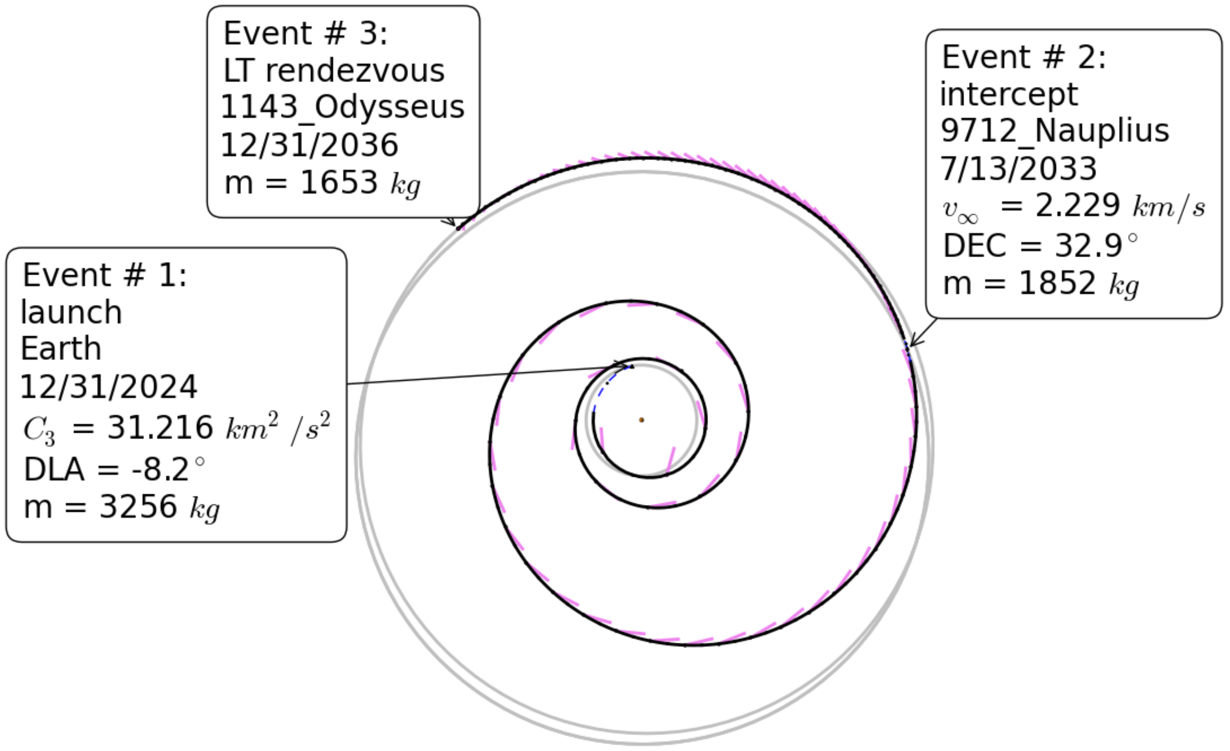


Figure 5.5: Optimal trajectory to Nauplius and Odysseus.

5.6.2 Trajectory Validation

The low-thrust trajectory generated by EMTG can be converted into a near-equivalent trajectory created by the MIRAGE navigation software. Demonstration of the MIRAGE targeting is done for two reasons: to perform independent verification and validation of the EMTG propagation and to create a database suitable for conducting orbit determination (OD), covariance analysis, and trajectory prediction and management during flight operations. MIRAGE models each thrust segment produced by EMTG as a long finite-burn based on a specified thrust profile and direction. The SEPV module of MIRAGE is employed to search for a Δv that achieves the target position state at the end of each thrust arc. To avoid overlap of finite-burn segments, the actual thrust arc, as modeled by MIRAGE, is shortened by up to one hour to allow margin for the Δv targeting algorithm. This slight foreshortening amounts to a tiny fraction of the overall thrust arc. Due to small mismatches in force models between the two sets of software, small adjustments or tweaks to the thrust level are required to achieve convergence of spacecraft state, including velocity and mass, as well as position. The MIRAGE analysis presented in this section was performed by Ken Williams from KinetX aeropsace and was the subject of a conference paper, on which the author of this work was a co-author [199].

For the scenario used as a test case for this study, force models and planetary ephemerides used by EMTG and MIRAGE were simplified as much as possible, as shown in Table 5.5, to facilitate comparison of results. Note that the Trojans are modeled as massless particles in this analysis.

Table 5.5: Simplified force model assumptions used for the EMTG-MIRAGE comparison.

Category	Model	Notable Details
Planetary ephemerides	DE433	• Available bodies (SPICE ID): SUN (10), MERCURY BARYCENTER (1), MERCURY(199), VENUS BARYCENTER (2), VENUS (299), EARTH BARYCENTER (3), EARTH (399), MOON (301), MARS BARYCENTER (4), MARS (499), JUPITER BARYCENTER (5), SATURN BARYCENTER (6), URANUS BARYCENTER (7), NEPTUNE BARYCENTER (8), PLUTO BARYCENTER (9) • Time span (ET or TDB): 1899 DEC 04 00:00:00 - 2050 OCT 17 00:00:00
Gravity	Sun, Earth and Jupiter only μ $[km^3/s^2]$	• $\mu_\odot$ = 1.32712440041939E+11 • $\mu_\oplus$ = 3.98694790080000E+05 • $\mu_{♃}$ = 1.26686510964000E+08
Solar radiation pressure	Spacecraft bus	• $A_s = 114.1\ m^2$ • $C_r = 1.0$ • ϕ = 1.015242216E+08 $\frac{kg\ km^3}{m^2 s^2}$ equivalent to solar luminosity $L_\odot = 1360 \frac{W}{m^2}$

Accelerations associated with specific forces were checked and model inputs tuned to ensure good agreement between the two software sets. It is understood that for an actual flight mission, complexity and fidelity of models would be increased systematically to preserve the consistency of models between the two software sets.

Figures 5.6-5.8 provide deviations of the MIRAGE derived states relative to EMTG for the Earth-Nauplius phase of the representative low-thrust mission. The red dashed lines indicate a point within a specific thrust arc where power limitations necessitated powering down one of the active thrusters. These changes were modeled in terms of thrust and mass flow changes, which can be characterized in MIRAGE as quartic and cubic polynomials of elapsed time, respectively. Since these inflection points were so approximated, this engendered certain state discrepancies, as shown in Figures 5.6-5.8. Nevertheless, the resultant maximum difference in targeted position over the entire trajectory phase was only 87.3 m. The maximum velocity difference of 1.17 m/s coincided with one of the thrust mode changes, but this discrepancy dissipated exponentially over the remainder of the trajectory phase, resulting in a final mass difference of 742 g, a final velocity difference of about 15 cm/s and an overall discrepancy in Δv usage of only 2.24 m/s out of a total of 17.34 km/s of total Δv expended, well within the operational tolerances that could be anticipated for such a mission.

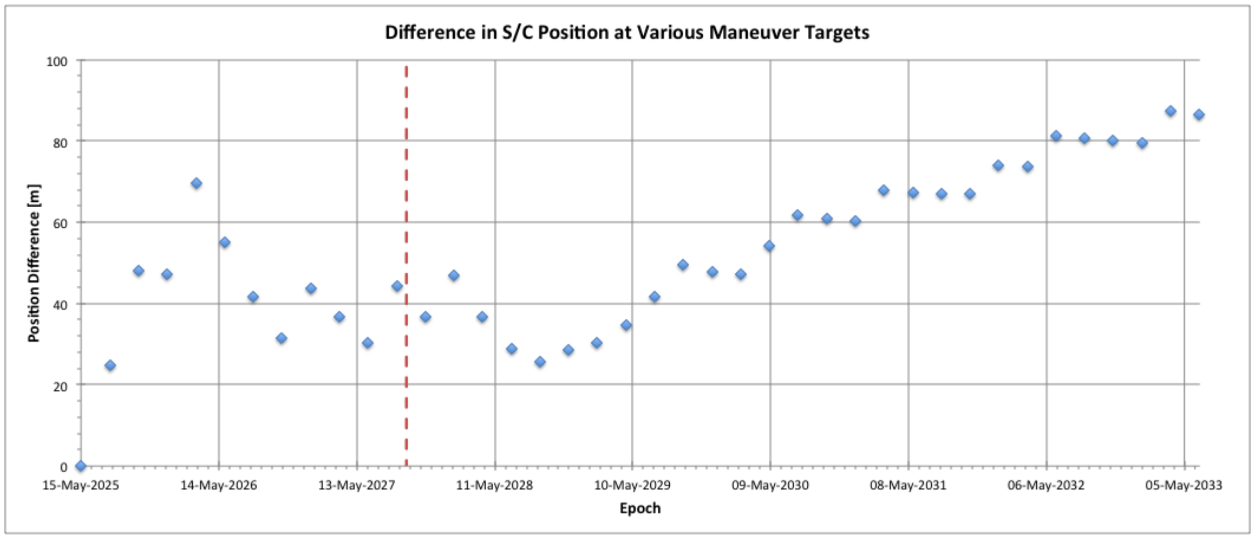


Figure 5.6: Differences in propagated spacecraft position of MIRAGE relative to EMTG for the Earth-Nauplius phase.

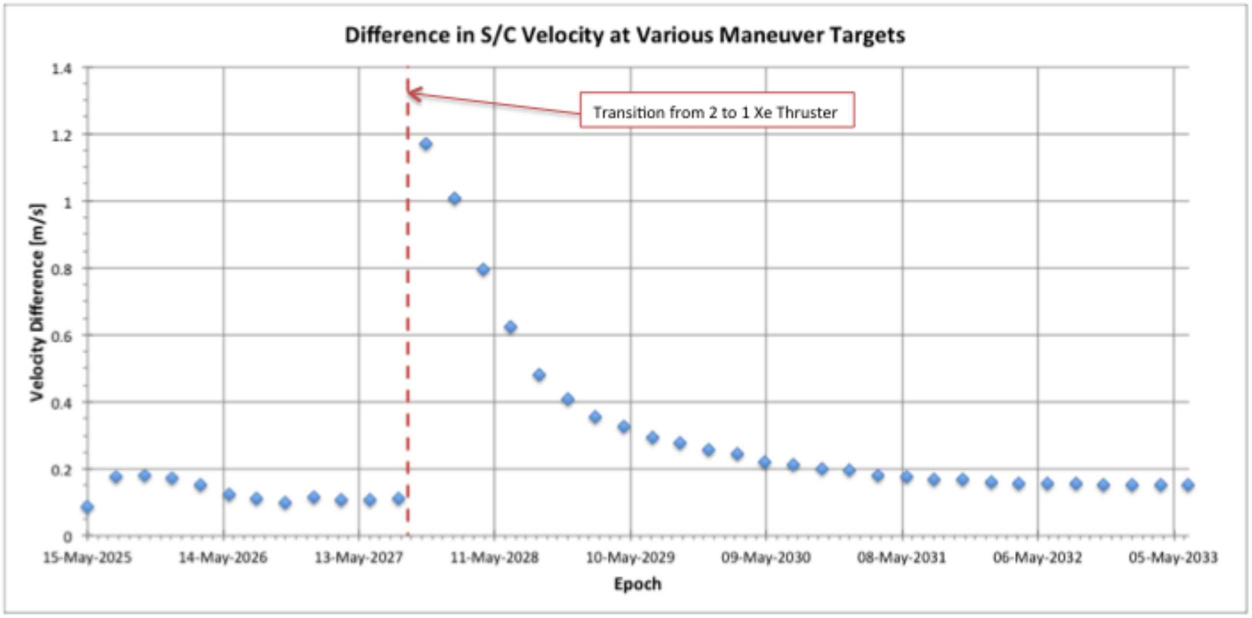


Figure 5.7: Differences in propagated spacecraft velocity of MIRAGE relative to EMTG for the Earth-Nauplius phase.

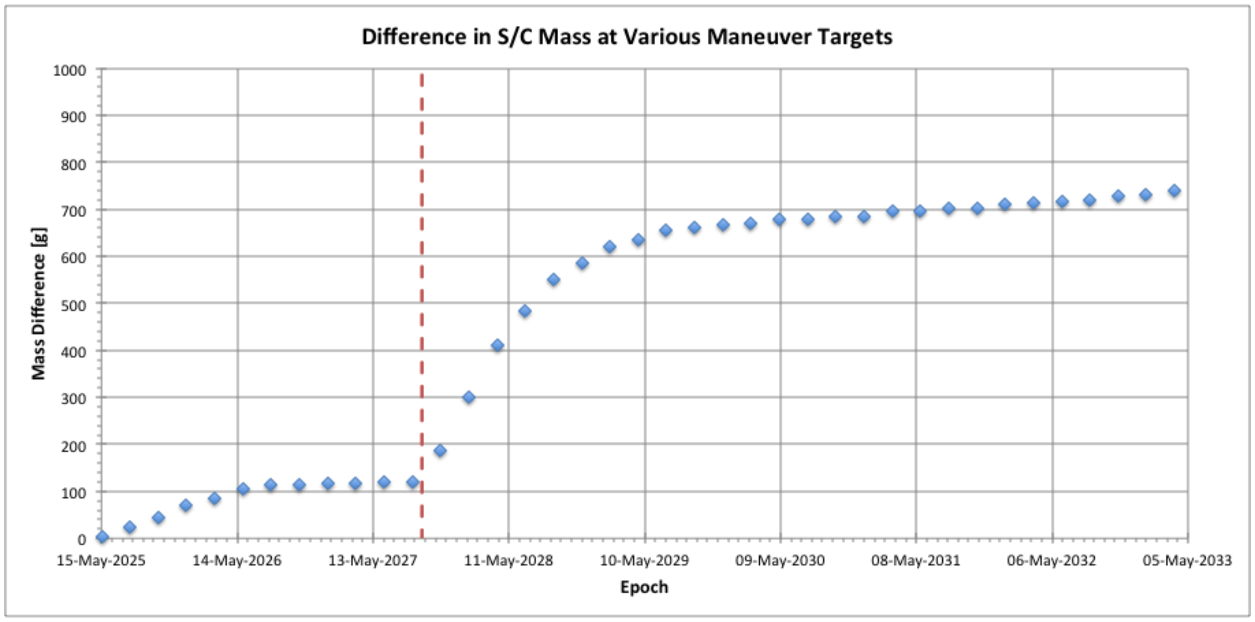


Figure 5.8: Differences in propagated spacecraft mass of MIRAGE relative to EMTG for the Earth-Nauplius phase.

Also, if the problematic thrust arc is partitioned at the point where one of the thrusters is deactivated, the agreement between EMTG and MIRAGE is somewhat improved. Although the maximum position discrepancy is now 162.3 m, the maximum velocity difference is now reduced to 69.6 cm/s, the final mass difference falls to 449 g, and the overall discrepancy in Δv usage is now only 940 cm/s. Hence, as expected, it is always possible to improve the agreement between the two software sets by improving the modeling fidelity of all thrust arcs.

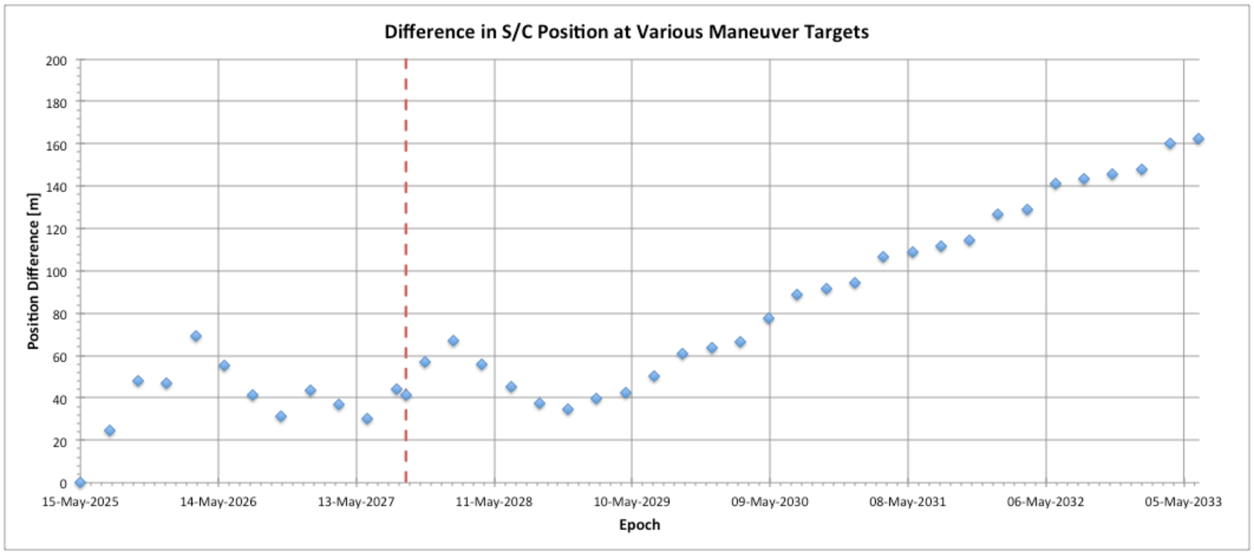


Figure 5.9: Differences in propagated spacecraft position of MIRAGE relative to EMTG for the Earth-Nauplius phase after thrust segment partition.

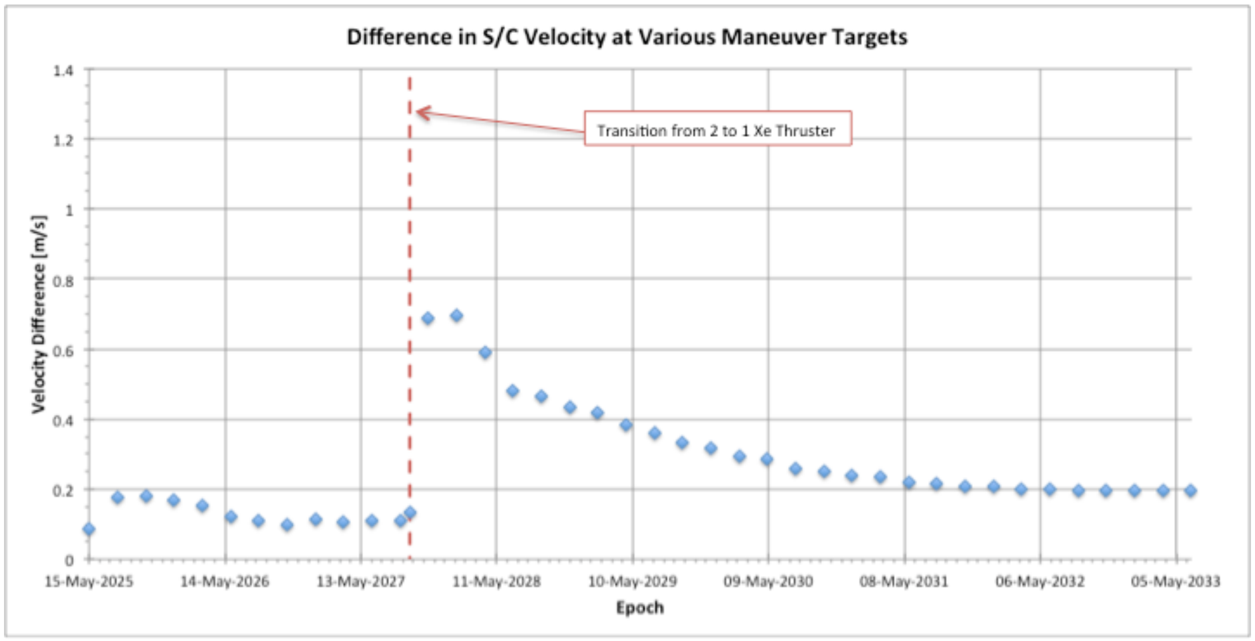


Figure 5.10: Differences in propagated spacecraft velocity of MIRAGE relative to EMTG for the Earth-Nauplius phase after thrust segment partition.

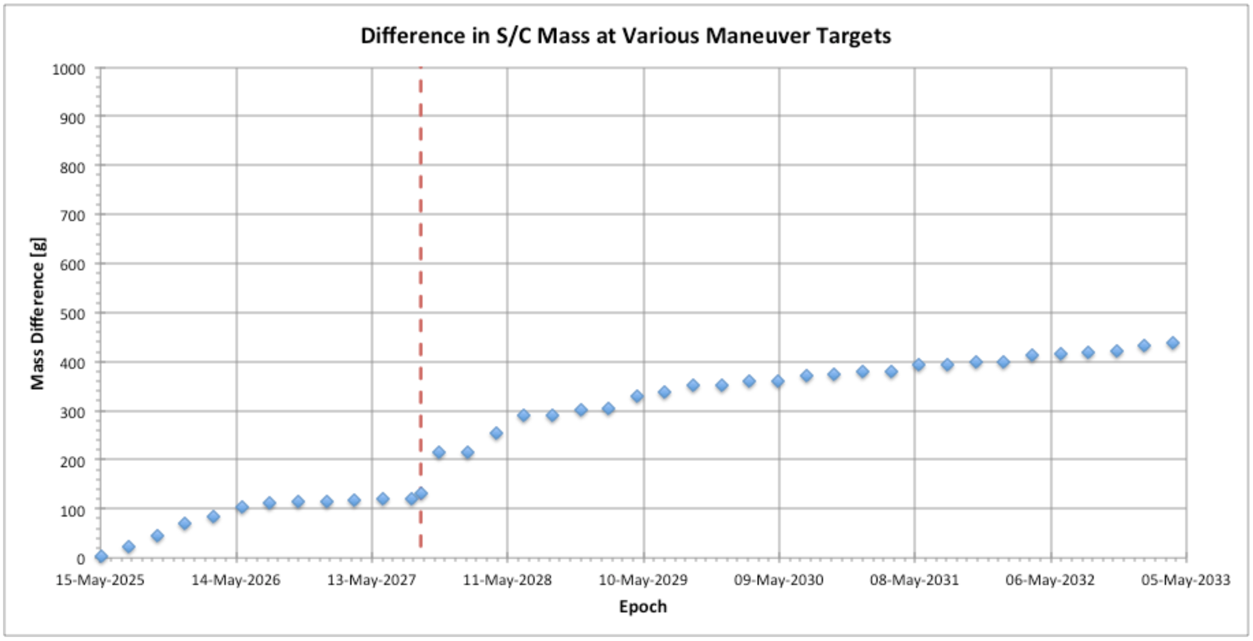


Figure 5.11: Differences in propagated spacecraft mass of MIRAGE relative to EMTG for the Earth-Nauplius phase after thrust segment partition.

Figures 5.12-5.14 show similar data for the Nauplius-Odysseus trajectory phase. There were no thruster mode changes within any thrust arcs over this phase, so consequently the agreement between EMTG and MIRAGE was somewhat improved, with a final mass discrepancy of 542 g, and a difference of only 14 mm/s out of 1.63 km/s of total Δv expended, again well within operational tolerances.

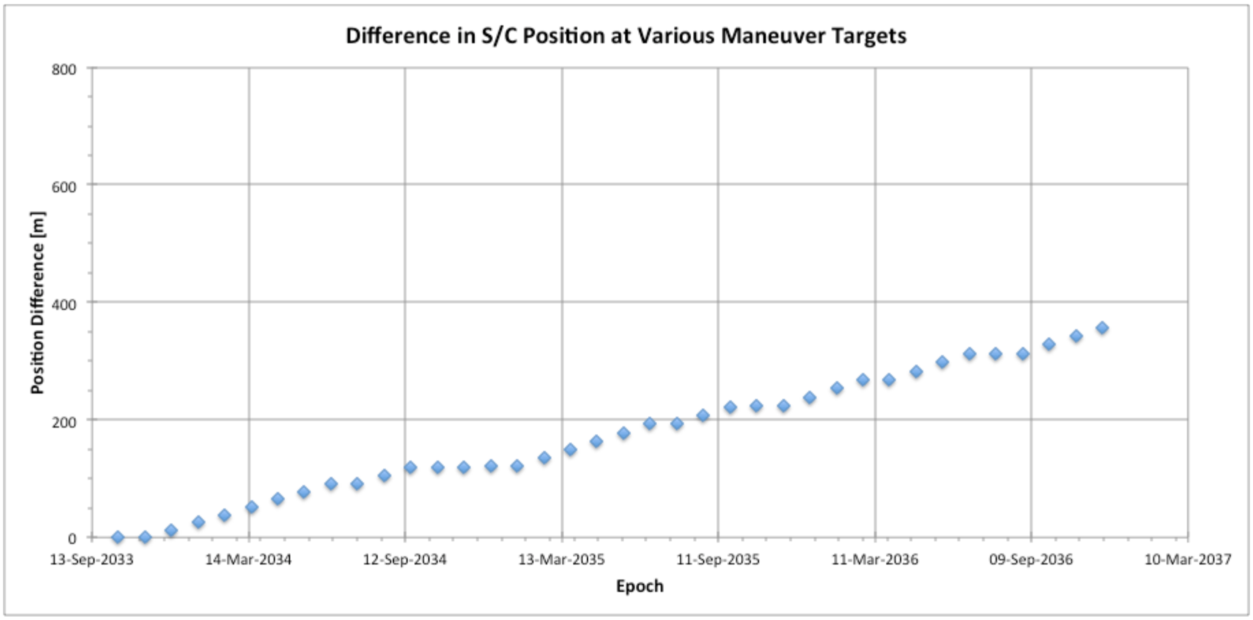


Figure 5.12: Differences in propagated spacecraft position of MIRAGE relative to EMTG for the Nauplius-Odysseus phase.

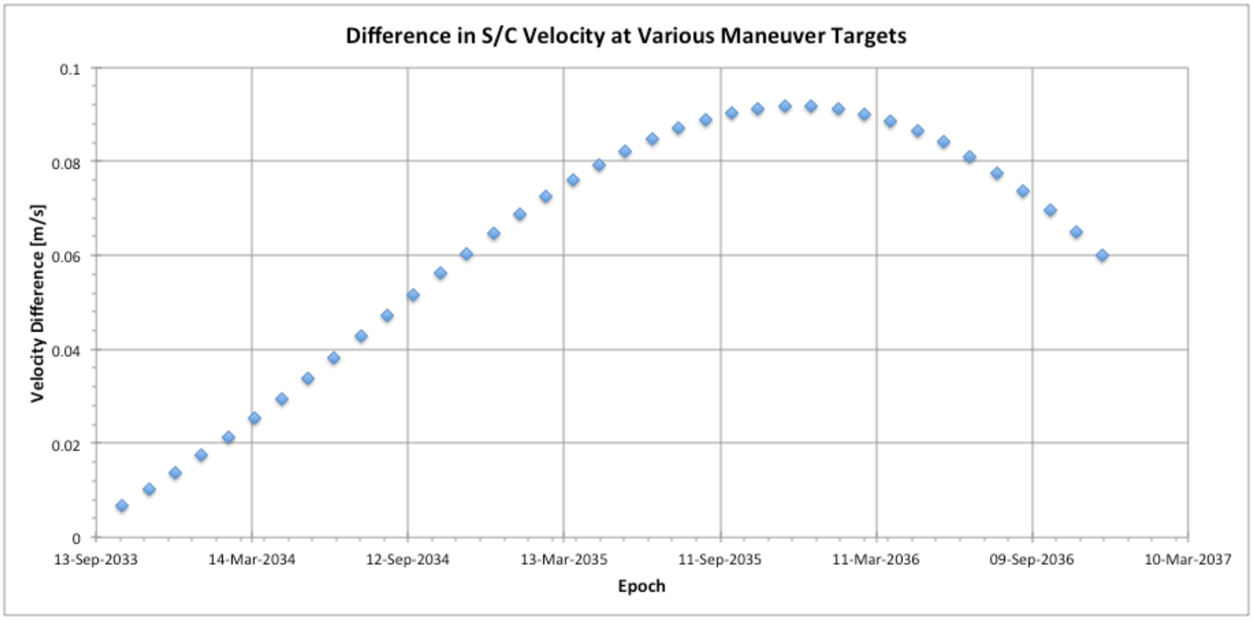


Figure 5.13: Differences in propagated spacecraft velocity of MIRAGE relative to EMTG for the Nauplius-Odysseus phase.

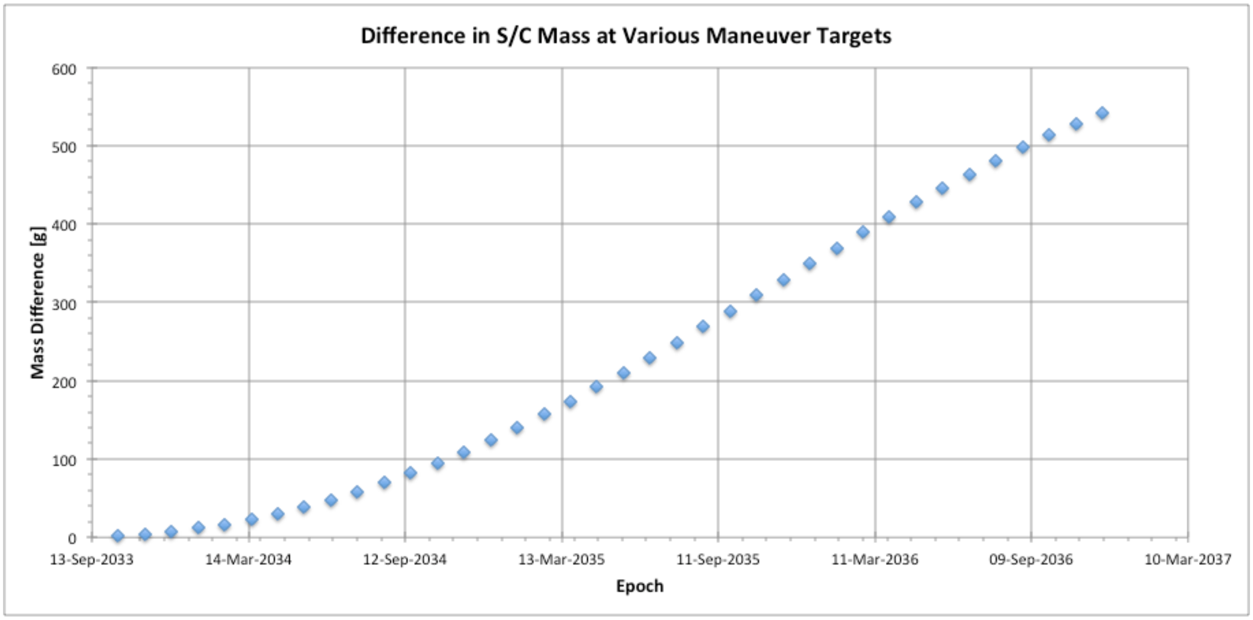


Figure 5.14: Differences in propagated spacecraft mass of MIRAGE relative to EMTG for the Nauplius-Odysseus phase.

Chapter 6

Towards a Gas Giant Satellite Tour Automaton

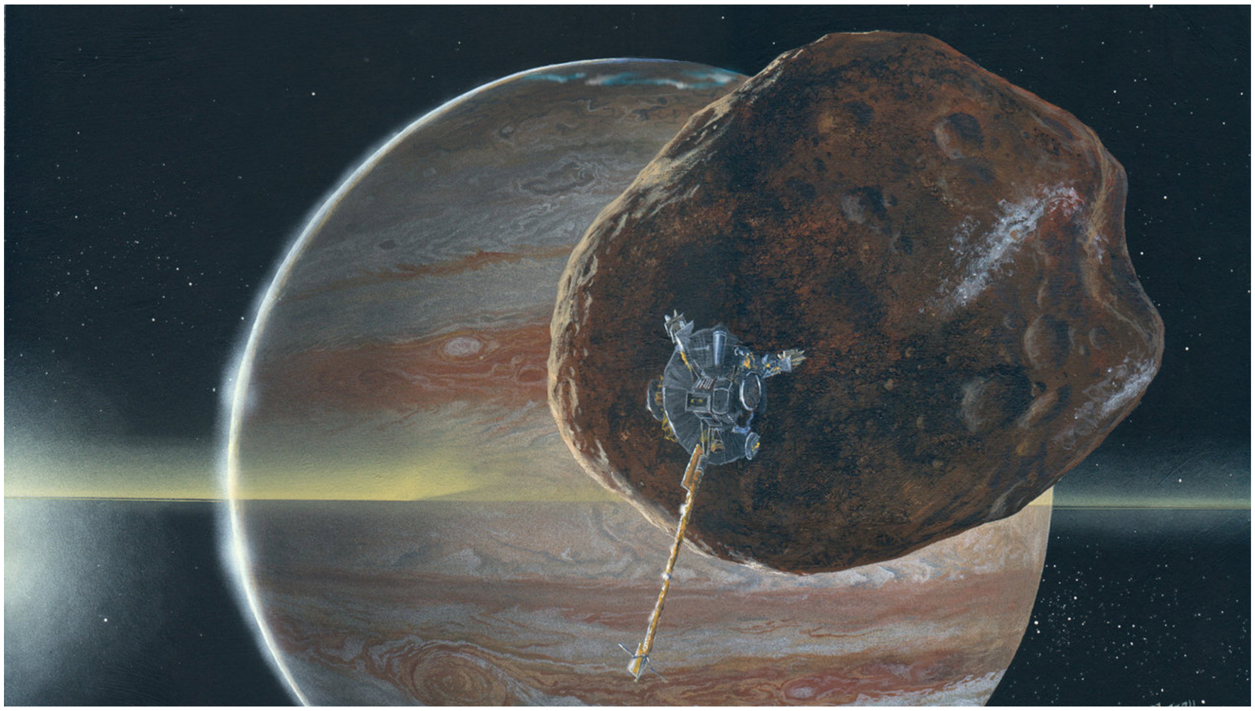


Figure 6.1: Artist's concept of Galileo at Jupiter.

The success of missions such as Galileo and Cassini are a testament to the maturity of satellite tour mission design techniques. The processes used for their design, however, can be manually intensive. Recent research in this domain of mission design is typically characterized by broad exhaustive searches of the planet-centered tour design space, accompanied by a decoupled optimization of the interplanetary portion of the overall trajectory. In particular, no preliminary design method exists that is capable of autonomously optimizing a trajectory from launch through the planetary capture and early stages of a satellite tour.

Studies such as the ones performed by Lynam et al. [200], Lynam [125,126] and Didion and Lynam [127] perform a thorough analysis of the capture design using exhaustive grid searches resulting in large solution databases or in high-fidelity point solutions. Solution databases suffer from the primary drawback of being inflexible to changing system or mission level requirements. Similarly, single point solutions, while useful as a final flight deliverable, are similarly laborious to generate in high-heritage tools such as STK and must be regenerated every time requirements change. The systematic tour design process developed by Scott et al. alleviates some of these problems and incorporates optimization into the design of planetary satellite-aided capture subject to interplanetary trajectory constraints, but still only considers a piecewise approach where the interplanetary transfer to a hyperbolic arrival asymptote at a gas giant is constructed separately from the capture design [128].

This chapter seeks to further close the gap between autonomous interplanetary trajectory design and the recent work that has been done towards automating the satellite tour pathfinding problem. In particular, a path forward will be described that combines the interplanetary HOC methods of Englander and Conway [47] with the Lambert grid-based methods of Lantukh and Russell [9], the latter of which has been demonstrated to be successful at performing satellite tour design. The combination of these methods is enabled by the analytical gradient algorithms developed in the previous chapters of this work in addition to several innovations that extend the work of Lantukh and Russell, including acceleration of the tree search algorithm using parallel computing architectures.

6.1 Lambert's Problem

"Determination of an orbit, having a specified transfer time and connecting two position vectors, is called Lambert's Problem" [160]. Where Kepler's problem is concerned with solving the two-body initial value problem, Lambert's problem solves the two-body boundary value problem for the following differential equation:

$$\ddot{\mathbf{r}} + \mu \frac{\mathbf{r}}{r^3} = \mathbf{0} \tag{6.1}$$

In precise mathematical terms, Lambert's problem can be stated as: "Given two times t_1 and t_2 and two position vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, find a solution to the differential equation in Eq. (6.1) $\mathbf{r}_{\text{trans}}$, such that $\mathbf{r}_{\text{trans}}(t_1) = \mathbf{r}_1$ and $\mathbf{r}_{\text{trans}}(t_2) = \mathbf{r}_2$". The two position vectors, are separated by the angle θ as shown in Figure 6.2.

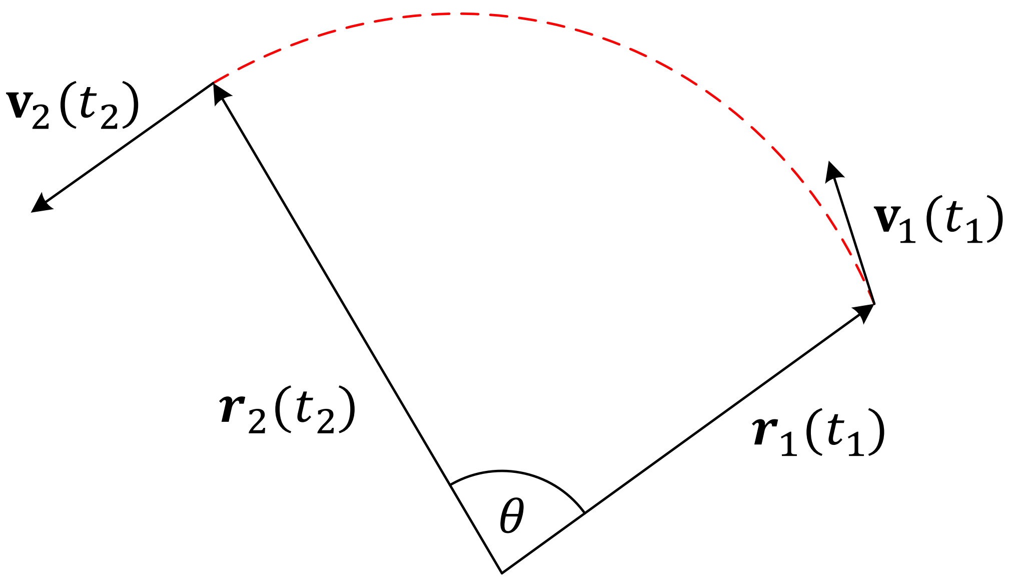


Figure 6.2: Lambert transfer geometry.

The objective of solving the two-body boundary value problem is to determine the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. Lambert's problem does not have a unique solution; any Keplerian arc connecting the position vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ with the same TOF, is a possible Lambert solution. Since Keplerian motion is confined to a plane, the final position and velocity vectors can be expressed as functions of the initial position and velocity vectors according to Eq. (3.1).

For any fixed $(\mathbf{r}_1, \mathbf{r}_2, \theta)$ geometry there exist $2N_{\text{rev}}+1$ prograde and $N_{\text{rev}}+1$ retrograde Lambert transfers, where N_{rev} is the number of complete revolutions of the central body required for the transfer (i.e. $N_{\text{rev}} = 0$ solutions have a transfer angle $\theta < 2\pi$). For $N_{\text{rev}} = 0$ the two solutions are simply the prograde and retrograde transfers. For every $N_{\text{rev}} \geq 1$, the four solutions are classified as combinations of prograde/retrograde + short-period/long-period, with each combination corresponding to a different transfer semimajor axis (or equivalently TOF).

Historically, solution methods for Lambert's problem served as a means for computing the orbits of celestial bodies. Today, Lambert arc computations are essential components of many preliminary mission design and spacecraft navigation methods due to their fast computation times. The exhaustive grid search tool STOUR, for example, computes body-to-body transfers by solving Lambert's Problem. The iterative Lambert algorithm used in this work is due to Arora and Russell [201] and implemented in C++ primarily by Matt Vavrina [152].

6.2 Lambert Grid Using True Anomaly Discretization

The complexity of a moon tour, where flybys can number in the tens to hundreds, can be considerably greater than a typical interplanetary MGA trajectory. Some of the most effective methods for solving the interplanetary MGA problem are relatively ineffective at addressing the enormous design space of a complex moon tour. Systematic grid searches, however, provide a means for approaching the problem, even though total enumeration is not possible.

The basis for the planetary satellite tour pathfinding framework proposed in this Chapter is a true anomaly-based grid, where body-to-body Lambert transfers are computed between grid nodes distributed around the orbits of the two bodies. The nodes are separated by a user-specified true anomaly spacing.

Pathfinding using a true anomaly grid proceeds as follows:

1. The user selects the number of discretization points for each flyby target in the pathfinding problem.
2. The grid search is initiated by selecting the epoch of departure from the first body (i.e. the launch body).
3. For a given value of N_{rev} the minimum TOF from the launch body to the second body in the flyby sequence is determined (that satisfies user-placed bounds on departure C3). For $N_{\text{rev}} = 0$ a solution will always exist, although the departure C3 will be large. For multi-revolution Lambert transfers a valid transfer will not exist for all possible TOF, therefore a bracketing search is used to determine the smallest TOF for which a departure C3 is defined.
4. The true anomaly of the target body is advanced by a user-specified grid spacing.
5. The Lambert algorithm is executed to determine if a valid transfer exists for the new transfer geometry and current N_{rev} value.
6. If a transfer exists it is stored
7. Repeat steps 4 - 6 until the maximum allowable TOF is reached.
8. Repeat steps 4 - 7 for each desired value of N_{rev} for the first leg in the flyby sequence.
9. Advance to the next flyby body in the sequence
10. For each subsequent body in the flyby sequence, repeat all steps using a C3 matching algorithm to determine valid flyby maneuvers.

6.3 C3 Matching Algorithm

In order to incorporate Lambert solutions into an MGA pathsolving algorithm, a method is required for determining if a feasible flyby maneuver is possible at a target body connecting two Lambert arcs. In order to ensure that a feasible gravity-assist can be achieved, the two constraints in Equations (2.6) and (2.7) must be satisfied.

The algorithm for connecting any two Lambert transfers with a flyby event proceeds as follows:

1. Solve Lambert's problem for the current geometry and TOF: $\mathbf{r}_1(t_1), \mathbf{r}_2(t_2)$ to get the outbound $C3_0$ for the flyby at $\mathbf{r}_1(t_1)$.
2. If Lambert solution is found, check if Eq. (2.6) is satisfied, i.e. $C3_0$ - $C3_{\text{target}} = 0$, where $C3_{\text{target}}$ is the incoming C3 at $\mathbf{r}_1(t_1)$.
3. If the C3 magnitude matches, then proceed to step 7.
4. Advance the true anomaly grid. If the new TOF is greater than the maximum allowed flight time, then terminate C3 matching.
5. Solve the new Lambert problem to get the new outbound $C3_1$
6. If the C3 magnitude matches, then proceed to step 7 otherwise, if $\text{sgn}(C3_0 - C3_{\text{target}}) \neq \text{sgn}(C3_1 - C3_{\text{target}})$, then a solution was missed and a bisection search is initiated to locate it. Otherwise, a match was not found, return to step 4.
7. Check if Eq. (2.7) is satisfied. If the turn angle is valid, then a flyby can be used to connect the two Lambert legs.
8. Repeat steps 4-7.

An example C3 vs. TOF plot is shown in Fig. 6.3. The above procedure will search along the C3 curve and identify the two intersection points labeled "Match #1" and "Match #2", which represent two valid C3 matches for a Venus to Earth Lambert transfer. After identifying these matches, Eq. (2.7) would need to be enforced in order to determine if the C3 match also corresponded to a physically realizable gravity-assist maneuver.

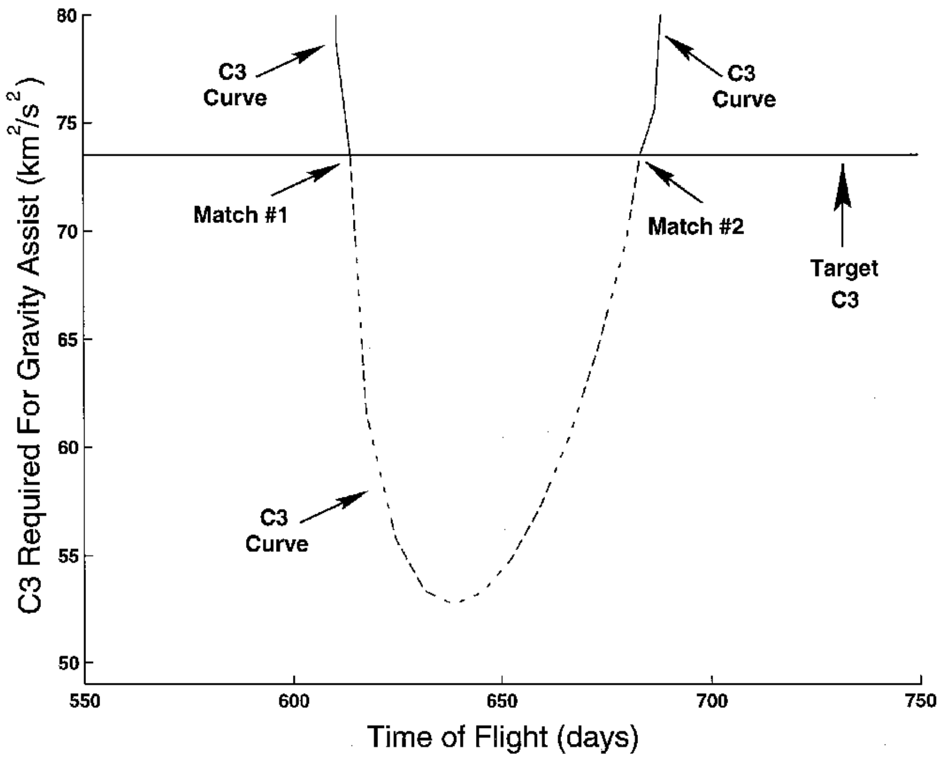


Figure 6.3: Example of a C3 matching plot for a transfer from Venus to Earth (from [4]).

6.4 Linearly Interpolated Ephemeris

Section 2.5.1 discussed the reasons why ephemeris calls can be an expensive computational bottleneck. This is especially true if these calls must be made to the hard drive, such as with the SPICE ephemeris system. Random access memory based readers, such as FIRE and SplineEphem significantly reduce ephemeris access costs, however, in an effort to accelerate the Lambert grid search as much as possible, a simplified linearly interpolated ephemeris was implemented for this work. The ephemeris works similar to SplineEphem in that JPL HORIZONS data is obtained over a mission's period of interest at discrete time intervals Δt. Then a linear interpolation is performed between ephemeris points to obtain planetary positions and velocities.

In Algorithm 3 r_{ephem}, v_{ephem} and t_{ephem} are equal-length arrays containing correlated position, velocity and epoch information respectively for an ephemeris object. The Julian Date for which the ephemeris data is being computed is denoted as JD and Δt is the interval in days between two successive ephemeris data points (e.g. $\Delta t = t_{\text{ephem}}[1] - t_{\text{ephem}}[0]$).

A linearly interpolated ephemeris does not provide sufficient accuracy for most mission design applications, however, when the low-fidelity solution produced by the tree search is input as an initial guess into a higher fidelity design tool, the local optimizer can compensate for the discrepancies between the linearized ephemeris and a higher accuracy one such as SplineEphem or SPICE. This sacrifice in accuracy due to employing the linear ephemeris model is offset by about a factor of four increase in execution speed compared with SplineEphem.

Algorithm 3 Linear Ephemeris Call

```
1: inputs are $r_{\text{ephem}}$, $v_{\text{ephem}}$, $t_{\text{ephem}}$, $JD$, $\Delta t$
2: procedure LinearEphem
3:     indexLow ← $\text{floor}\left(\frac{JD - t_{\text{ephem}}[0]}{\Delta t}\right)$
4:     indexHigh ← $\text{ceil}\left(\frac{JD - t_{\text{ephem}}[0]}{\Delta t}\right)$
5:     $\mathbf{r} \leftarrow r_{\text{ephem}}[\text{indexLow}] + \frac{r_{\text{ephem}}[\text{indexHigh}] - r_{\text{ephem}}[\text{indexLow}]}{\Delta t}$
6:     $\mathbf{v} \leftarrow v_{\text{ephem}}[\text{indexLow}] + \frac{v_{\text{ephem}}[\text{indexHigh}] - v_{\text{ephem}}[\text{indexLow}]}{\Delta t}$
7: end procedure
```

6.5 Pathfinding Software Framework

6.5.1 N-ary Tree

The pathfinding problem, as it pertains to MGA trajectories, is best represented as a tree. A tree is a construct from graph theory. Specifically, a tree is a connected graph that does not feature any cycles (acyclic). A graph is connected if each of its vertices is reachable from all other vertices, and is acyclic if none of its vertices can be reached from itself (i.e. there is no path leaving from any vertex that can be traced back to itself). The tree topology used to represent the MGA pathfinding problem in this work is the n-ary tree. Every node in an n-ary tree has at most n children. An example of an n-ary tree is shown in Figure 6.4. Every node in this tree, except the leaf nodes, has two child nodes, and is also an example of a binary tree. This figure also gives one an appreciation for the combinatoric complexity of the gravity-assist problem. Here, we only consider two child nodes per node, however, for the flyby pathfinding problem, there typically exist many more than two subsequent decision branches that can be explored.

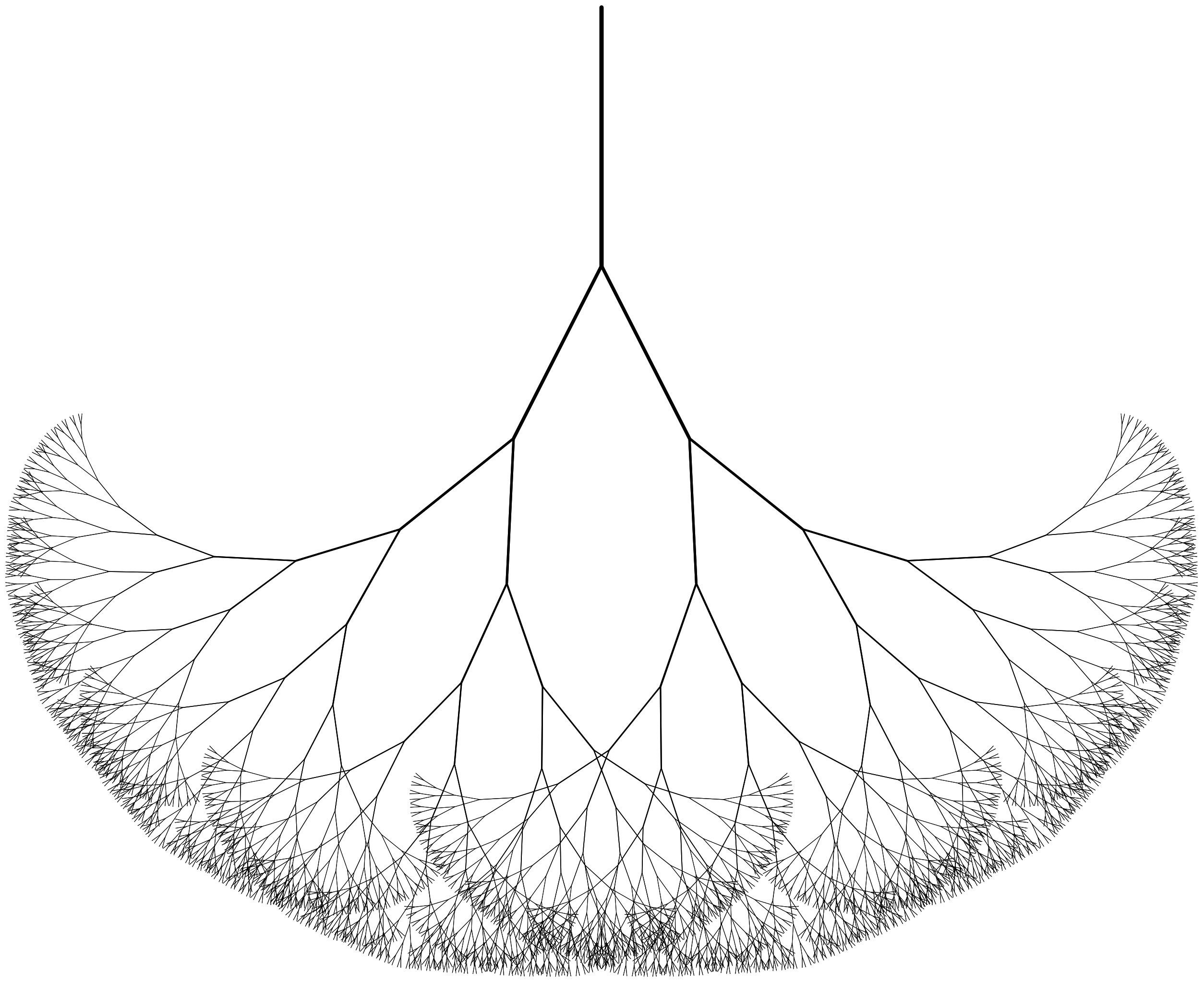


Figure 6.4: A complete binary tree propagated to a depth of 12.

6.5.2 The Trajectory Forest Data Structure

A gravity-assist pathfinding tree is created by encoding a particular launch/departure date as the root node of the tree. The root node serves no other purpose other than to encode this particular date and does not represent an actual body-to-body trajectory. The child nodes of the root node (nodes with a depth of one) represent feasible trajectories that connect the first body in the flyby sequence to the second body. These departure legs all share a common launch/departure date, but can represent transfers with different flight times, maneuver geometries, resonances, number of revolutions around the central body etc.

A collection of multiple trajectory trees forms a forest. The trajectory forest represents the total enumeration of the gravity-assist pathfinding problem, subject to the level of discretization imposed by the practitioner. An example trajectory forest is shown in Fig. 6.5.

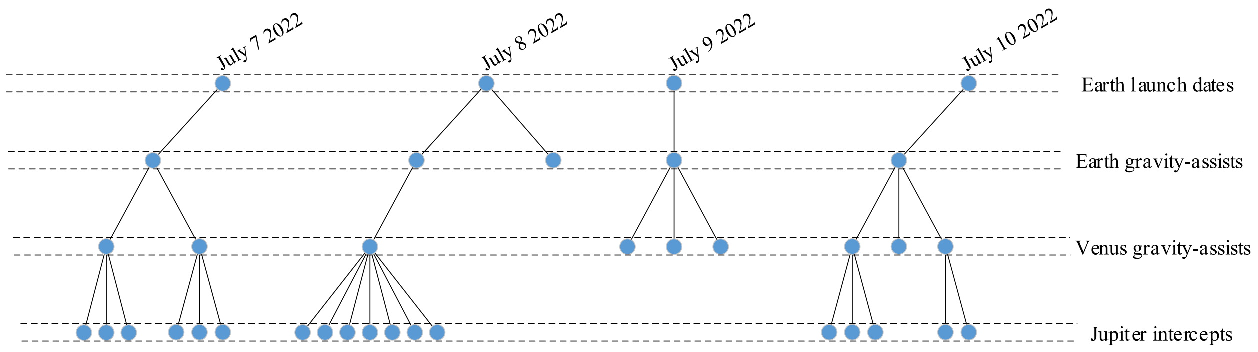


Figure 6.5: Example trajectory forest for an E-EV-J gravity-assist sequence.

6.5.3 Tree Search Algorithms

The use of a tree data structure to encode the MGA problem is convenient from a data organizational aspect, but it also flexibly supports a large variety of search strategies. Several studies focusing on multiple flyby trajectories have employed different tree search algorithms. Izzo et al. [74] employed a novel search strategy called the lazy race tree search (LRTS) that selected the fifty shortest flight time transfers from leaves at *any* level in the tree to branch. The shortest flight time strategy was effective for the GTOC 6 problem [202] as the problem's cost function rewarded solutions that generally featured the largest number of flybys in a fixed time. The LRTS attempted to counter the greediness of a beam search by allowing leaf nodes to remain in contention for branching at any point during the search process. By contrast, beam search only selects a certain subset of the current leaf nodes for branching and all other leaf nodes are permanently removed (pruned) from the tree. The Monte Carlo tree search (MCTS) is another branching strategy that has been applied to the multiple flyby pathfinding problem [203]. This search strategy uses random sampling of the decision space to focus on the most promising decisions (and therefore the most promising regions) in the tree.

Beam search is perhaps the most straightforward tree search strategy, and is the strategy employed for the application examples presented in this chapter. Beam search is initiated by branching all of the root nodes in the forest. Then the level one leaf nodes are sorted according to some partial cost function, and a certain number or percentage are selected for branching to create the level two leaf nodes. The process is repeated until some termination criterion is met. Figure 6.6 depicts a beam search algorithm searching through three levels of a trajectory forest. The beam search algorithm is inherently greedy and is useful for RAM-limited problems where only a fraction of the design space can be explored.

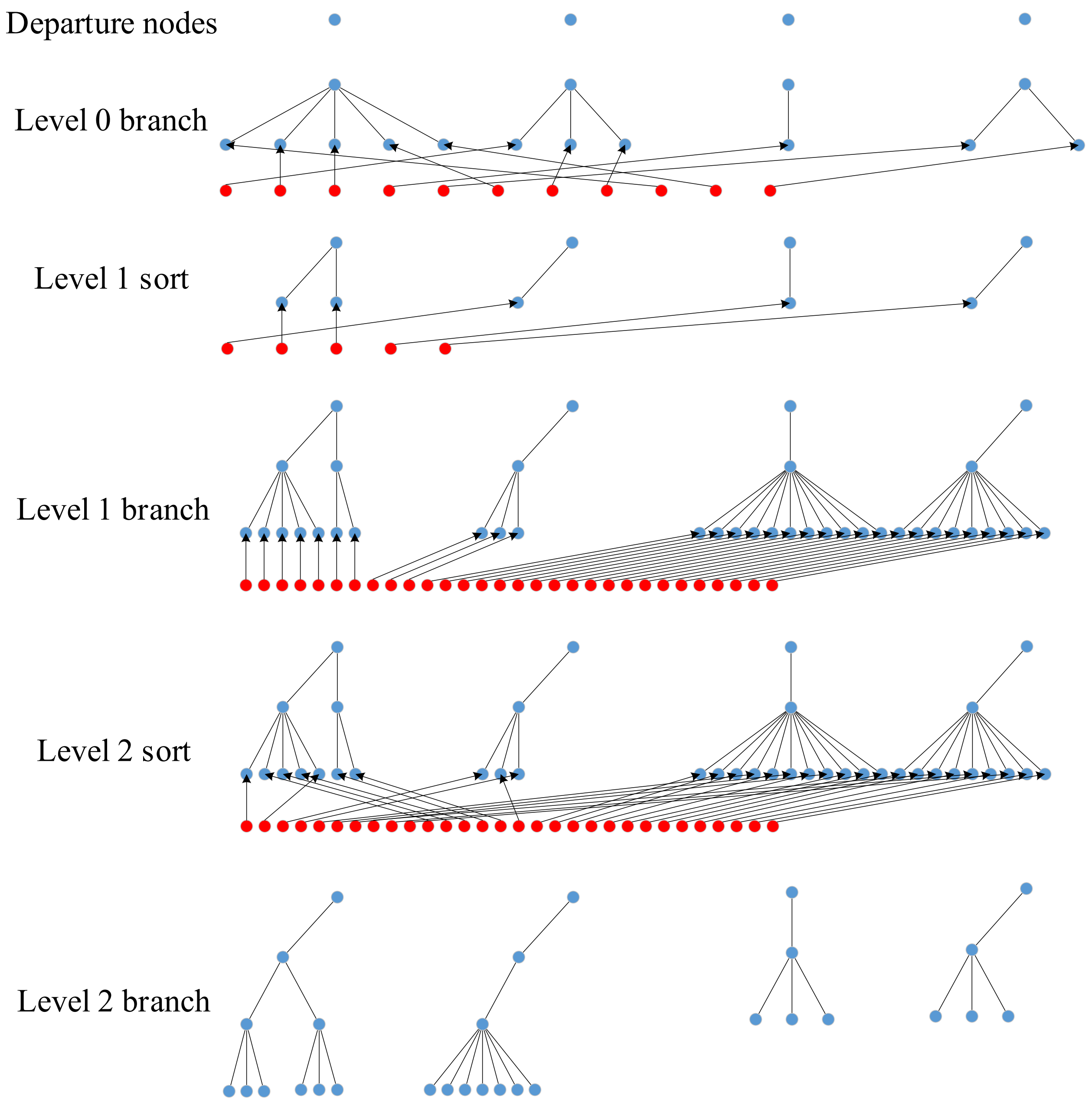


Figure 6.6: Dynamic trajectory forest exploration using beam search.

Specific tree search strategies are not a primary focus of this work, and the identification of more suitable search algorithms represents an worthwhile avenue for future research, especially in the area of satellite tour design.

6.5.4 Software Implementation

The software implementation of an n-ary tree is ordinarily based on the construction of an individual node class (assuming an object oriented framework) whose interconnected instances form the tree structure. A node object will possess a pointer to its parent node in the tree, which is set to null if the current node is the root of the tree. Each node also has either a pointer to each child that branches from it, or a pointer to its first child and a pointer to its adjacent sibling node. In this way, any node in the tree can be reached from any other node. The tree can also be easily pruned using recursion. Due to the large memory footprint associated with a trajectory forest representing an MGA Lambert grid, the implementation used to produce the results in this work does not connect the nodes of each tree using pointer variables. Furthermore, only the current leaf nodes that are eligible for branching are retained in RAM. All ancestor nodes are archived on the hard drive using serialized binary files. Each node in the tree is given knowledge of its parent using two integer identification numbers: one number indicates which binary file the serialized parent object is stored in, the other encodes a unique number identifying the parent node object.

While many problems are sufficiently small that additional layers of the forest, or even perhaps the entire forest, may be retained in memory, this is not true in general. A long term launch date scan, or a fine time-of-flight grid can result in a single layer of the forest containing hundreds of thousands or millions of nodes. Retaining only branching leaf nodes in RAM allows the program to accommodate the largest problem size possible.

Reconstruction of a particular trajectory is straightforward. The parent ID variable that each node possesses allows for the reconstruction of a multiple flyby trajectory via recursive hard disk accesses. The parent IDs represent the connections in the underlying mathematical tree, and in effect create a network of reverse linked lists that makes recursive tracing of the tree possible. The necessary and sufficient information required to reconstruct any trajectory identified in the search process is just the list of leaf node identification numbers. In practice the final product of a tree search is a vector of tuples, where each tuple contains the ID of a leaf node, the ID of the file that its contained in, as well as any relevant scoring information for that node (e.g. cumulative flight time, delta-v, launch C3 etc.). The final vector of information allows for Pareto sorting of the final trajectories, without having to reconstruct any one trajectory object. An analyst is then free to reconstruct any individual trajectory (for plotting, or to obtain other more detailed information) by recursively tracing one of the linked lists starting with one of the leaf nodes.

The reverse linked list concept just described is similar to the one implemented by Lantukh [9], however, our implementation extends this with additional software engineering strategies that will be described in the next section.

6.5.5 RAM Management Using Object Pooling

The large branching factor of most gravity-assist pathfinding problems makes any software implementation of the tree search inherently RAM limited. Storing only the leaf nodes and their immediate children in RAM helps to mitigate the data storage limitation, however, very large problems can consume all of the available RAM on a computer. As the algorithm dynamically explores the flyby forest, object creation and destruction can also lead to significant slow-downs. In order to prevent solutions from being missed due to the unavailability of RAM, and to prevent the high frequency creation/destruction of node objects as the program proceeds, an object pool is introduced to the search program. An object pool is a software design pattern whose operation is based on a "pool" of pre-constructed software objects. These objects are drawn upon by the program whenever an object instance is required, and then when it is not needed any longer, it is returned to the pool instead of being destructed. An example where object pooling is beneficial is a program that utilizes many network connections and calls. Network connection objects, such as sockets or database connections can be time consuming to create. A pool of these objects would eliminate the need to constantly create a new object whenever the program requests one, by instead simply providing one from the pool.

Maintaining an object pool provides two main benefits. First, the computational overhead associated with the constant construction and destruction of objects is eliminated. Second, by allocating a pre-determined number of objects to the pool, the RAM footprint of the program using the pool can be bounded before execution begins. The trajectory forest search procedure is made more efficient by using a node pool, and is also size-restricted eliminating the possibility of locking up a compute node due to RAM exhaustion.

6.6 Non-Ballistic Pathsolving

6.6.1 Deep-Space Maneuvers

Incorporating DSMs into the search increases the size of the design space, and results in a more complicated Boundary Value Problem (BVP) compared to the ballistic body-to-body planar Lambert transfer. By inserting a DSM into the trajectory, the initial and final central body relative velocities of the spacecraft can no longer be determined using Lambert's method alone. A DSM trajectory that remains in the Lambert plane will experience a change in orbital energy due to the added impulse, or should the transfer contain a broken-plane maneuver, Lambert's method by definition can no longer be used as the trajectory will be split between two transfer planes.

In order to accommodate the inclusion of impulsive maneuvers, the ballistic Lambert transfer is augmented. First, the spacecraft's post departure position ($\mathbf{r}^{+}_{\text{departure}}$) and velocity ($\mathbf{v}^{+}_{\text{s/c-departure}}$) are propagated for a period of time $\Delta t_{\text{DSM}_1} = t_{\text{DSM}} - t_{\text{departure}}$ by solving Kepler's equation, which provides the pre-DSM inertial state of the spacecraft $\mathbf{r}_{\text{DSM}}$ and $\mathbf{v}^{-}_{\text{s/c-DSM}}$. Lambert's problem is then solved using the boundary conditions $\mathbf{r}^{+}_{DSM}$, $\mathbf{r}^{-}_{\text{arrival}}$, and TOF $= \Delta t_{\text{DSM}_2}$. Solving this problem will yield the unknown boundary velocities $\mathbf{v}^{+}_{\text{s/c-DSM}}$ and $\mathbf{v}^{-}_{\text{s/c-arrival}}$. The magnitude of the DSM required to complete the transfer is then calculated:

$$\Delta \mathbf{v}_{\text{DSM}} = \mathbf{v}^{+}_{\text{s/c-DSM}} - \mathbf{v}^{-}_{\text{s/c-DSM}} \tag{6.2}$$

There now remains one final unknown in the problem posed, that being the initial departure velocity of the spacecraft relative to the central body ($\mathbf{v}^{+}_{\text{s/c-departure}}$). For a ballistic transfer, this would be determined as one of the boundary velocity outputs when solving Lambert's problem, however, the first half of the trajectory has been replaced with a Keplerian propagation arc, therefore, this velocity must be provided as an input to the problem. The enumeration of the outgoing flyby velocity orientation is achieved by discretizing the flyby "pump" and "crank" angles, which are denoted as θ and ϕ respectively in Figure 6.7

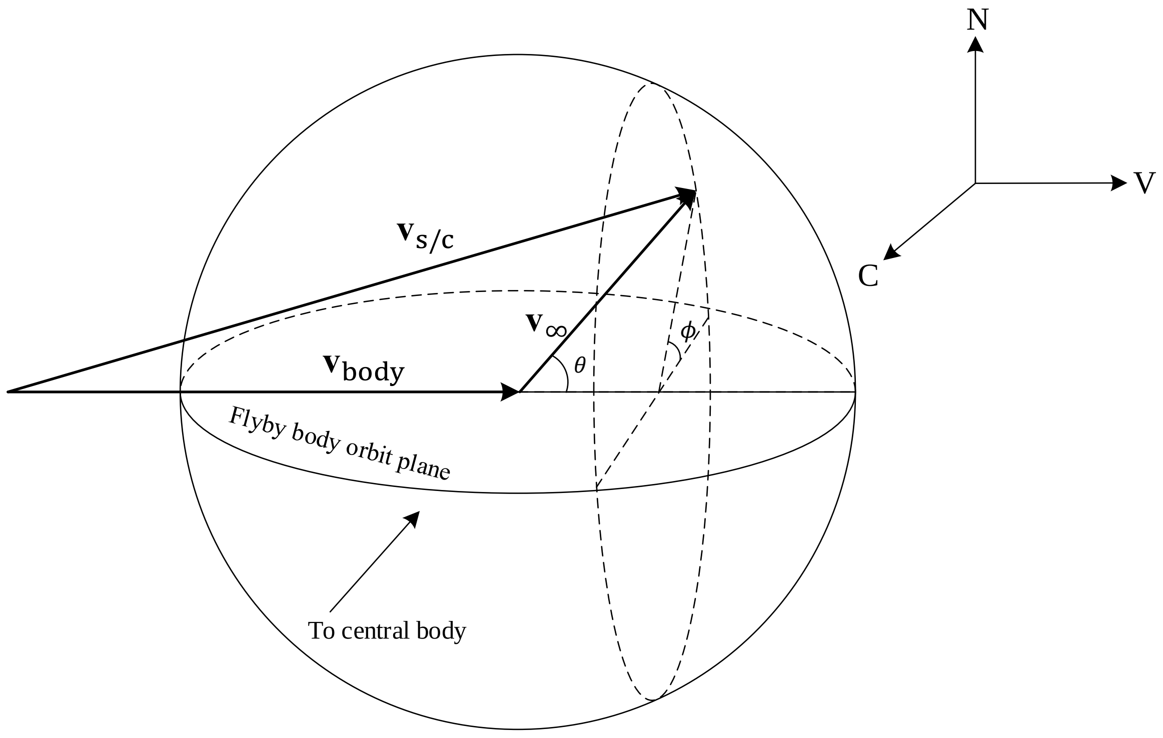


Figure 6.7: Flyby pump θ and crank ϕ angles. Reproduced from [3]

The magnitude of the outgoing spacecraft hyperbolic excess velocity vector in the velocity-normal-conormal frame $\left[\hat{V}\hat{N}\hat{C}\right]$ is determined using Eq. (6.5):

$$\mathbf{v}_\infty^+ = v_\infty^+ \begin{bmatrix} \cos(\theta) \\ \sin(\theta)\sin(\phi) \\ -\sin(\theta)\cos(\phi) \end{bmatrix} \tag{6.3}$$

This velocity is expressed in the inertial frame with respect to the central body using the following coordinate transformation:

$$\mathbf{v}_{\mathrm{s/c}}^+ = \mathbf{v}_{\mathrm{body}} + T_{\mathrm{RTN}}^{\mathrm{ICRF}} \mathbf{v}_\infty^+ \tag{6.4}$$

where

$$T_{\mathrm{VNC}}^{\mathrm{ICRF}} = \begin{bmatrix} \hat{V} & \hat{N} & \hat{C} \end{bmatrix} \tag{6.5}$$

The unit vectors defining the $\left[\hat{V}\hat{N}\hat{C}\right]$ frame are computed using the position and velocity of the flyby body at the flyby epoch:

$$\hat{V} = \frac{\mathbf{v}_{\text{body}}}{\|\mathbf{v}_{\text{body}}\|} \tag{6.6}$$

$$\hat{N} = \frac{\mathbf{r}_{\text{body}} \times \mathbf{v}_{\text{body}}}{\|\mathbf{r}_{\text{body}} \times v_{\text{body}}\|} \tag{6.7}$$

$$\hat{C} = \hat{T} \times \hat{N} \tag{6.8}$$

Flybys that result in a change only to the pump angle ($\Delta\theta$) will alter the inertial velocity of the spacecraft with respect to the central body. This means that the semimajor axis, and therefore the orbital period of the post-flyby trajectory, will also be adjusted. Gravity-assists that increase the pump angle are referred to as "pump downs" as the inertial velocity of the spacecraft is decreased, and the semimajor axis reduced. Conversely, "pump up" maneuvers increase the semimajor axis as a result of the larger $\mathbf{v}_\infty$ decreasing the pump angle.

A gravity-assist that changes only the crank angle will alter the inclination and eccentricity of the spacecraft's orbit, but preserve the size of the semimajor axis. If the inclination of the spacecraft's pre-flyby orbit plane is small (i.e. near the central body's equator), then the inclination change will be large, while the eccentricity change will not be very significant. The opposite is true if the pre-flyby inclination is large. A cranking flyby at high inclinations will not result in a large inclination change, however, the eccentricity change will be more pronounced. The procedure for performing a DSM trajectory search is summarized in Algorithm 4.

6.6.2 Resonant Flybys

Two bodies are in orbital resonance if there exists an integer relationship between the periods of their orbits about the central body. In other words, $p : q$ resonance exists between a spacecraft and a potential flyby target if the spacecraft completes p revolutions about the central body in the time it takes the flyby target to complete q revolutions. Incorporating resonant transfers into multiple gravity-assist trajectories is beneficial for several reasons.

Algorithm 4 Departure to a DSM

```
1: inputs are $\phi$, $\theta$, $TOF$, $v_\infty^+$, $t_{\text{DSM}}$
2: procedure DSM
3:     compute $\mathbf{v}_\infty^+$
4:     compute $T_{\text{RTN}}^{\text{ICRF}}$
5:     compute $\mathbf{v}_{\text{s/c}}^+$
6:     if departing from flyby then
7:         compute required turn angle
8:         if required turn angle > maximum possible turn angle then
9:             return
10:        end if
11:    end if
12:    propagate spacecraft's post-flyby state ($\mathbf{r}_{\text{s/c}}^+$, $\mathbf{v}_{\text{s/c}}^+$) for $\Delta t_{\text{DSM}_1}$ $\implies$ $\mathbf{r}_{\text{DSM}}^-$ $\mathbf{v}_{\text{DSM}}^-$
13:    solve Lambert transfer from $\mathbf{r}_{\text{DSM}}$ to $\mathbf{r}_{\text{arrival}}$ with transfer fime $\Delta t_{\text{DSM}_2}$
14:    if Lambert solution exists then
15:        compute the magnitude of the DSM using Eq. (6.2)
16:        save DSM transfer
17:    end if
18: end procedure
```

Resonant orbits set up repeat encounters with a gravitating body, which is useful for queuing future flyby maneuvers, but also for scientific reasons. When investigating a body, such as a planetary satellite, repeat visits to that body are beneficial as they offer additional opportunities to make observations and collect scientific data. For example, repeat encounters with a gravitational body allow a spacecraft to refine its measurements of the gravity field and multiple ground tracks are a necessity for generating surface maps.

Including resonant transfers in a Lambert-based pathfinding algorithm is challenging as Lambert's problem is singular for transfers of 0 and 180 degrees. Furthermore, in the context of a real ephemeris model, exact orbital resonances do not exist. With this in mind, this study implements near-resonant transfers by inserting a small impulsive maneuver near the midpoint of the transfer in order to target a precise flyby of the target body. The target arrival date is computed using the following equation:

$$t_{\text{arr}} = t_{\text{dep}} + qT_{\text{body}}, \tag{6.9}$$

where t_{dep} is the flyby departure epoch, and T_{body} is the orbital period of the target flyby body, calculated at the departure epoch. Incorporating near-resonances into the flyby search in this manner is convenient as these trajectories are a subset of the DSM transfers described in the last section. In particular, a near-resonance transfer is one where the flyby pump angle α remains unchanged and the change in crank angle ϕ can vary on the interval $[0, 2\pi]$.

The incorporation of resonant transfers into the Lambert tree search algorithm is demonstrated with a repeat-encounter trajectory of Saturn's moon Enceladus. The tree search problem setup is summarized in Table 6.1.

Table 6.1: Lambert tree search settings for repeat Enceladus encounter sequence.

Parameter	Value
Start date (JD)	2463598.5
min. departure C3	1.96 $\mathrm{km}^2/\mathrm{s}^2$
max. departure C3	4.0 $\mathrm{km}^2/\mathrm{s}^2$
ΔC3	0.1 $\mathrm{km}^2/\mathrm{s}^2$
$\Delta\phi$	1 degree
max. revs.	1
min. flyby altitude	20 km
max. flyby altitude	100 km

In this example, the spacecraft begins at Enceladus with a relatively low velocity relative to the satellite, and then executes ten consecutive π transfers, however, since linearly interpolated SPICE ephemeris data is used, these transfers are not exact π transfers. The minimum Δv ten-encounter trajectory identified by the tree search is shown in Fig. 6.8.

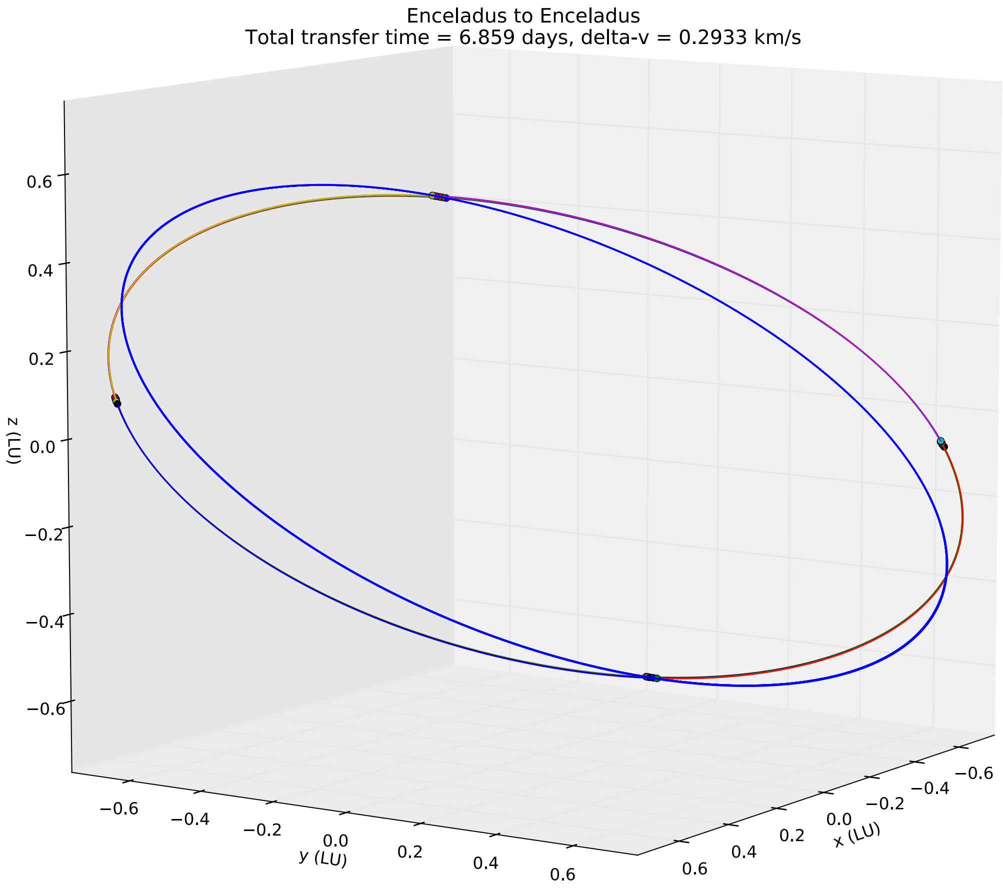


Figure 6.8: Repeat Enceladus backflip encounters using the Lambert tree search and linearly interpolated SPICE data. Dots indicate flyby encounters and maneuver locations. Plot is shown in the J2000 ecliptic frame.

The repeat “backflip” encounters with the moon accumulate a significant amount of Δv during the Lambert search. In order to increase the ephemeris fidelity, and reduce the propellant cost, the tree search results are input to a local-optimizer using the MGAnDSMs transcription. The optimizer is free to adjust the epoch, magnitude and direction of the targeting maneuvers as well as the epochs of the encounters slightly to reduce the cost function (total Δv). Local optimization using MGAnDSMs reduced the required Δv for the complete ten encounter sequence from 293 m/s to 195 m/s. An example Enceladus-to-Enceladus transfer from the optimal trajectory is shown in Fig. 6.9.

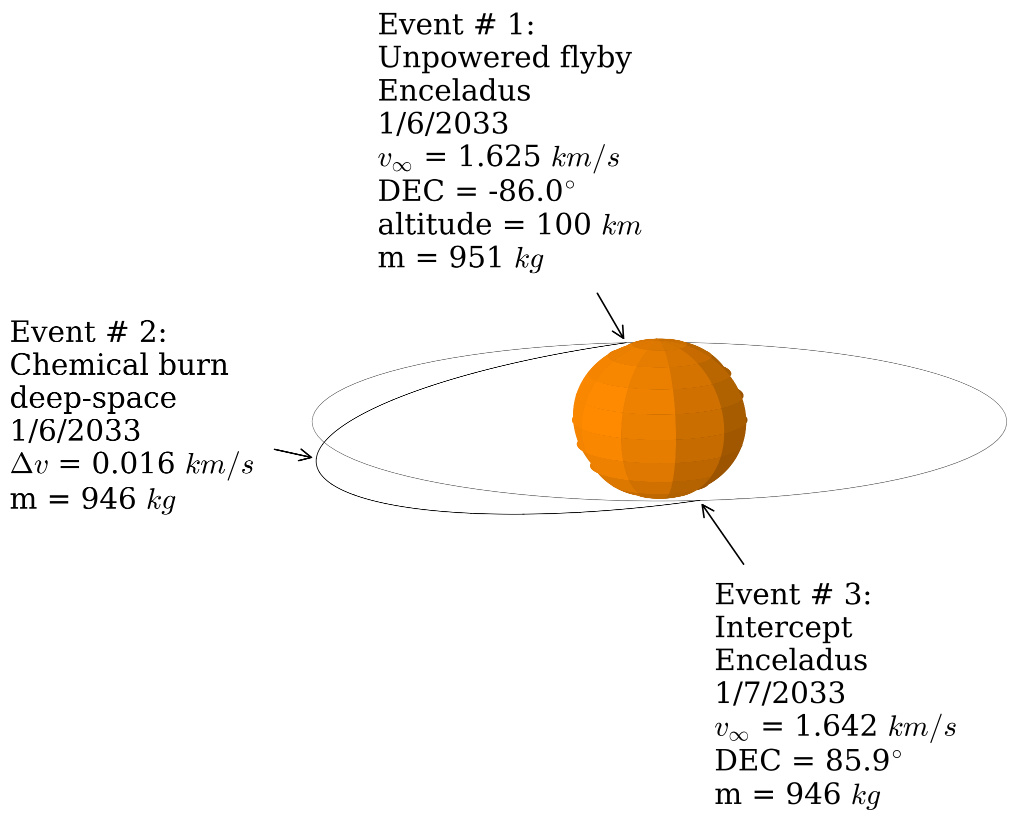


Figure 6.9: Example Enceladus to Enceladus π transfer locally optimized using the MGAnDSMs transcription. Ephemeris data is spline-fit SPICE (SplineEphem). The original Lambert transfer used 36.71 m/s of Δv. Plot shows the trajectory in the J2000 Saturn equatorial frame.

6.7 Parallel N-ary Tree Exploration

Trajectory forest traversal is an embarrassingly parallelizable problem. The branching of any two leaf nodes in the forest are independent operations that may be carried out in parallel, in a straightforward manner. To begin with, the launch dates to be considered are distributed evenly amongst all available processing cores. Each core is responsible for branching each launch date node that is assigned to it. Once the first generation of leaf nodes has been produced (representing departure legs), the leaf nodes that are selected for further branching are again evenly distributed amongst the available processors and the procedure repeats. In this manner, the entire problem is executed in parallel except for a few select operations such as sorting and file i/o. Even these operations can be performed in parallel as there exist algorithms for performing parallel file read/write as well as sorting.

This parallelization of the tree search problem is realizable in both distributed and shared memory configurations. It is worth noting that a distributed implementation of this algorithm would not require substantial inter-processor communication overhead, and is therefore especially conducive to processor grid scaling. For this work, a shared memory parallelization scheme is considered, primarily due to the compute hardware available, but the analyses and observations are equally valid for a distributed version of the tree search algorithm.

6.7.1 Multithreading Using OpenMP

OpenMP is a specific implementation of processor multithreading that follows the fork-join operational concept and is available for the C, C++ and Fortran programming languages. OpenMP is attractive as it is relatively straighforward to implement, being controlled almost exclusively by way of compiler directives in the source code. Essentially, a programmer need only designate areas of the code that are to be executed in parallel (e.g. a for loop whose iterations are independent of one another) and the OpenMP API carries out the necessary low-level actions required to perform the parallel execution.

In the OpenMP paradigm, the runtime environment designates a master thread that subsequently designates (forks) its own slave threads amongst which the system divides the computational work to be performed. Individual threads in this thread pool carry out their alloted instructions in the designated parallel region of code and then idle at the end of the parallel region until all threads in the pool have finished their execution (i.e. they join). Being a shared memory scheme, all threads executing in an OpenMP parallel region exist on the same compute node and therefore draw from the same source of RAM.

6.7.2 Intel Xeon Cluster vs. Intel i7 Quad-Core

The parallel tree search framework described in this chapter is demonstrated in this work by deployment on a Linux computing cluster furnished with four Intel Xeon E-74890 processors. Each Xeon chip houses 15 individual processors, which in turn have two logical threads each, for a total of 160 compute threads across the system. The cluster also features 512 GB of total RAM, which is an attractive feature for the RAM-limited breadth-first tree traversal algorithm.

In order to quantify the execution speed increases gained from parallelization of the flyby tree traversal algorithm, an example MGA interplanetary mission to Saturn is considered. A fixed flyby sequence of Earth-Earth-Venus-Venus-Earth-Saturn is assumed. Only ballistic trajectories are considered, and the search is performed from Saturn backwards through the flyby sequence to Earth. We have found this backwards search to be beneficial when considering trajectories to the outer planets as the tree search is only performed on trajectory legs that successfully connect from the inner solar system to the outer solar system. The backward search also filters out any Lambert solutions with excessively long times-of-flight between the inner and outer planets, thus eliminating early in the search many trajectories that will violate a total flight time constraint imposed by the mission designer.

For this problem, a 1000 day Saturn arrival window scan in one day increments is considered. The orbits of Venus and Earth are discretized into 225 and 365 grid points of true anomaly respectively, achieving a 1 day flight time resolution for the Lambert flight time grid. The breadth-first tree search algorithm retained the top 30% of all partial solutions at each level (flyby) in the search. Tree nodes were ranked according to the following partial cost function:

$$C = \frac{1}{\sqrt{2}}\sqrt{TOF^2 + v_{\infty_f}^2} \tag{6.10}$$

where TOF is the time-of-flight of the current partial solution and v_{∞_f} is the arrival hyperbolic velocity at Saturn. This is a partial cost function in that it is applied against each partial flyby sequence upon completion of each level in the tree search. More sophisticated partial cost functions and Pareto sorting algorithms have been described by both Johnson [108] and Lantukh [9] that seek to mitigate the greediness associated with pruning the flyby tree as the search is executed, however, those strategies are not a focus of this work.

The tree search was executed on three separate hardware configurations in order to capture the benefits of the parallel search:

1. Serial search on an Intel i7-7820 @ 2.90 GHz
2. Parallel OpenMP search on an Intel i7-7820 @ 2.90 GHz
3. Parallel OpenMP search on the Linux cluster with Intel Xeon E-74890 @ 2.80 GHz

The Intel i7 runs were executed on a laptop with four processors (for a total of eight logical threads). The solution with the lowest Saturn encounter velocity is shown in Fig. 6.10.

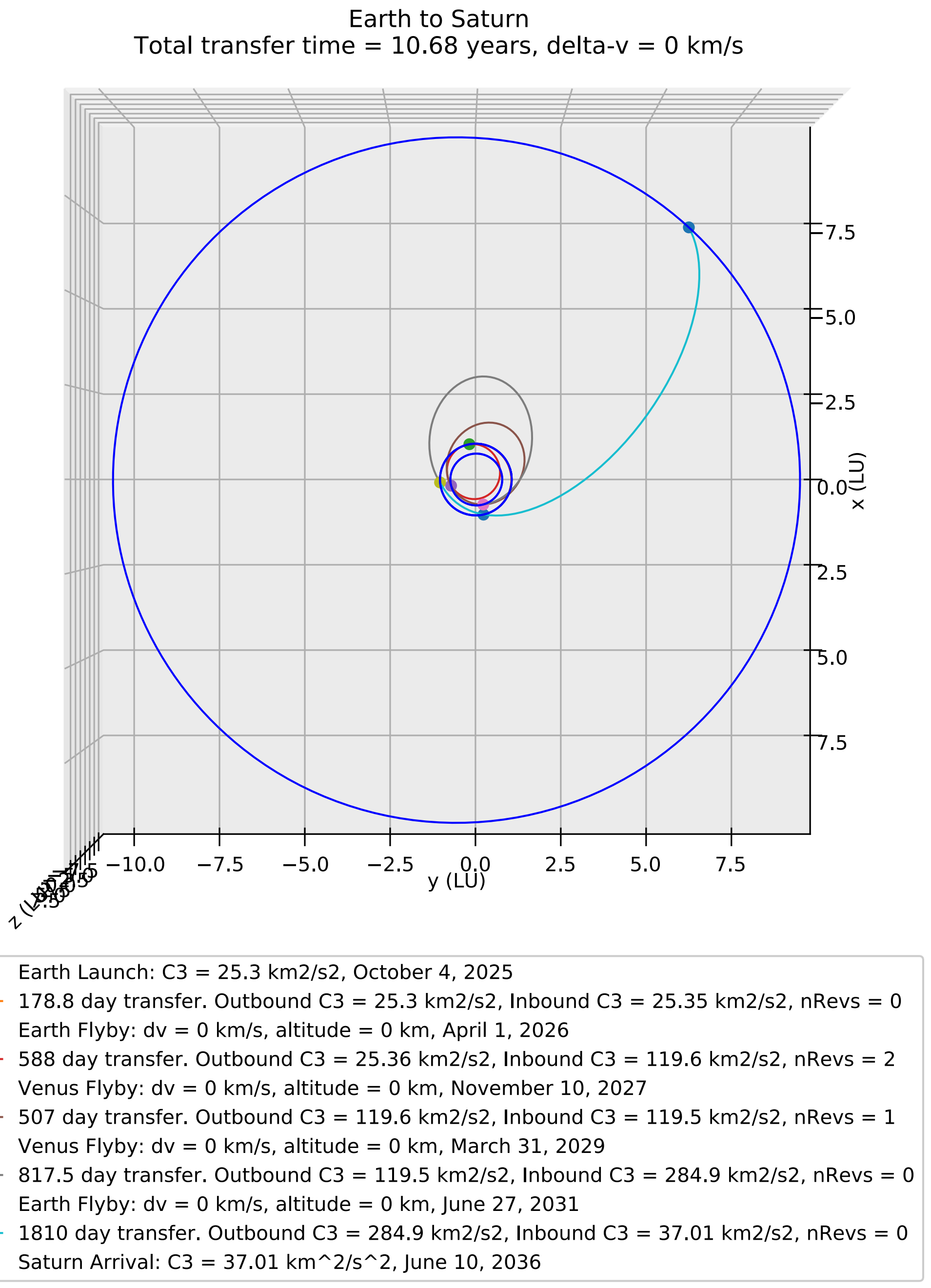


Figure 6.10: Ballistic Lambert MGA trajectory from Earth to Saturn, using an E-EVVE-S flyby sequence.

The execution times associated with each hardware setup are summarized in Table 6.2.

Table 6.2: Lambert tree search execution times for the E-EVVE-S problem.

Architecture	Runtime
Serial i7	3.06 hrs.
OpenMP i7	0.54 hrs.
OpenMP Linux cluster	4.97 min.

The embarrassing parallelism of the breadth-first flyby tree search affords substantial speed-up when OpenMP multithreading is exploited. All three cases computed 1 153 641 feasible Lambert trajectory legs and 579 full E-EVVE-S sequences were identified (with the lowest 70% partial sequences being pruned at each level in the tree search).

6.8 Satellite Tour Automaton

The combinatorics of designing a planetary satellite tour make an exhaustive survey of the problem space impossible even for the world's most powerful super computers. The tour design problem is an ideal candidate for deployment of the tree search framework outlined thus far in this chapter. The level of fidelity provided by the Lambert solutions that form the basis for the tree search is insufficient for anything beyond an academic exercise, therefore, a useful design workflow will feature a means for taking the output of the tree search and increasing its accuracy. This section will illustrate preliminary work that contributes to an automated moon tour design process and will showcase the progress made with an example 35-flyby Jovian moon tour design that is optimized from launch through tour completion. The example problem is solved using the MGAnDSMs transcription described in Chapter 3 and therefore relies heavily on the analytic gradient formulae discussed there.

The traditional means for converting a Keplerian patched-conic initial guess to high-fidelity is to manually forward-shoot the spacecraft state in a high-heritage tool such as STK. Forward differential correction is usually employed to satisfy equality constraints, targeting states provided by the initial guess. If optimization is applied to the trajectory design, it is usually applied in a piecewise manner, as was the case with the solutions to the GTOC6 problem submitted by the ESA-ACT [74] and University of Texas at Austin [9] teams. The ESA-ACT team computed optimal encounter-to-encounter transfers using differential evolution. The UT-Austin team computed their entire tour sequence using Lambert arcs including VILMs and $n\pi$ transfers, and then converted the maneuvers to finite-burn arcs using a local optimizer based on Hybrid

Differential Dynamic Programming (HDDP). The local low-thrust optimization was performed one transfer at a time, where the v-infinity body encounter states provided by the Lambert search were targeted by the optimizer.

The focus of this section is the design and optimization of a complex interplanetary transfer to Jupiter, followed by a Jovian satellite tour. A high-thrust chemical propulsion system is assumed, with an I_{sp} of 320 seconds. A fixed interplanetary flyby sequence of E-VVE-JSOI is used, where JSOI denotes arrival at Jupiter's sphere of influence. The flyby sequence is the same as the one used by Cassini en route to Saturn. Problem assumptions for the interplanetary trajectory to Jupiter are shown in Table 6.3 and the optimal trajectory found is shown in Fig. 6.11. Upon arrival at Jupiter's sphere of influence, the spacecraft then targets a flyby of the moon Ganymede and then Callisto. The terminal intercept v_∞ of the spacecraft at Callisto is constrained to be less than 6 km/s, which is more than sufficient to ensure capture around Jupiter. The entire sequence of launch through the first Callisto intercept was optimized, without the need for an initial seed state, using the MBH heuristic and was modeled with the MGAnDSMs transcription that was described in Chapter 3. The optimization objective was to maximize mass delivered to Callisto. The optimizer was allowed to insert up to one deep-space maneuver anywhere in each phase of the trajectory, but their presence was not assumed. In particular, it is worth emphasizing that the Jupiter orbital insertion (JOI) and apojove raise maneuvers (ARM) were placed by the optimizer and were completely driven by the terminal constraint that the Callisto intercept occur with a relative velocity of no more than 6 km/s. Comparing with terminology present in the literature, the optimizer essentially performed a single satellite-aided capture flyby, followed by the JOI and ARM maneuvers before the first Callisto flyby.

Table 6.3: Optimization parameters for interplanetary and Jupiter capture sequence.

Parameter	Value
Earliest allowed launch date	February 1st, 2015
Launch window	365.25 days
Maximum flight time	8 years
Maximum allowed initial mass	6000 kg
Launch vehicle	Atlas V (551) NLS-2
Launch declination bounds	$[-28.5°, 28.5°]$
Thruster I_{sp}	320 s
Flyby sequence	E-VVE-JSOI-GC
Ephemeris model	SplineEphem
Maximum number of maneuvers per phase	1
Arrival type	Callisto intercept
Maximum v_∞ at Callisto intercept	6 km/s
Minimum Jupiter approach distance	200 000 km
SNOPT feasibility tolerance	1.0e-5
Max MBH runtime	8 hrs
Max SNOPT runtime	30 s
Objective function	maximize final mass

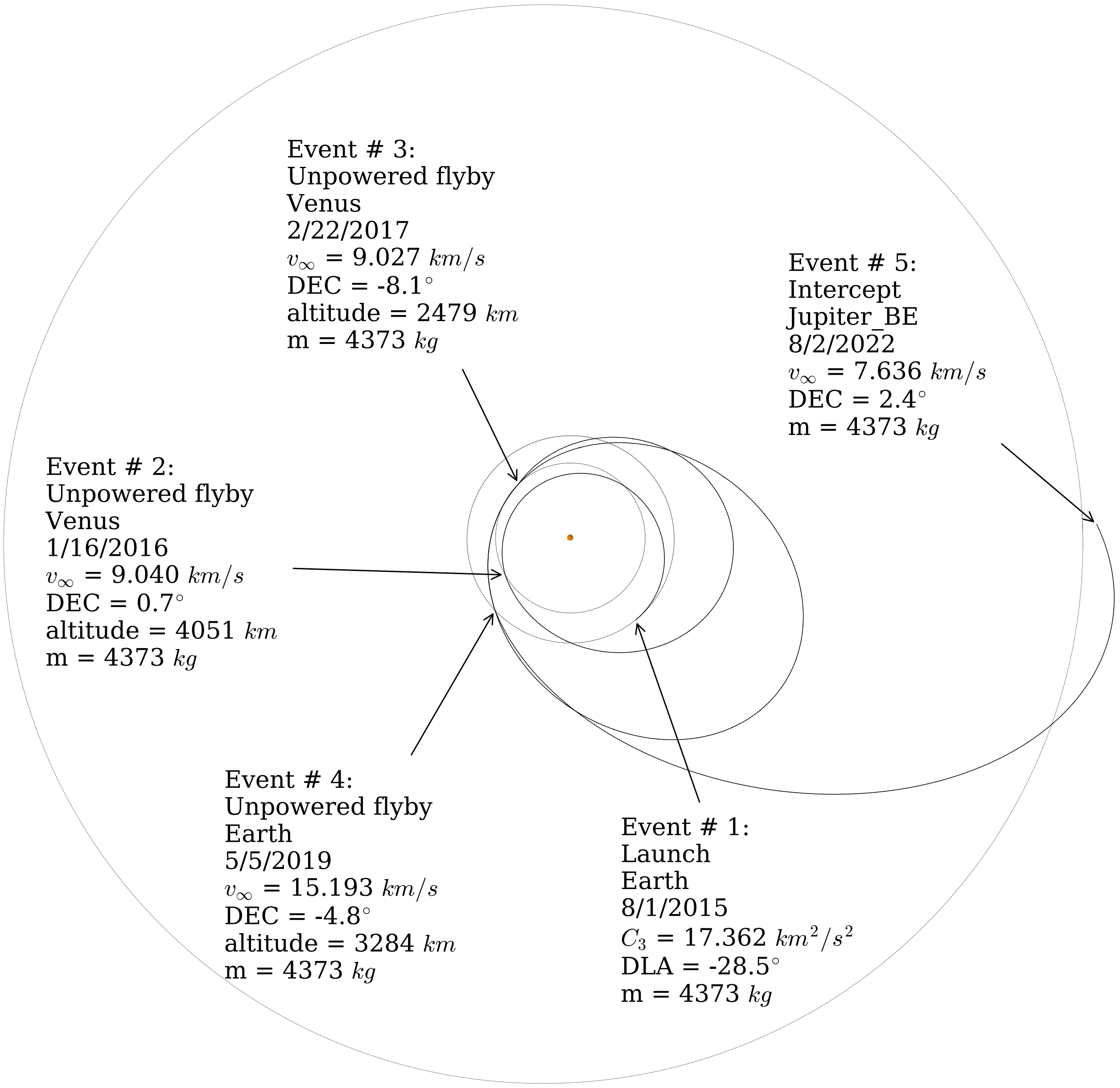


Figure 6.11: Interplanetary transfer to Jovian SOI.

6.8.1 Ephemeris Referenced Boundary Interface

The capture sequence at Jupiter was modeled using a series of multiple shooting phases across the boundary of and inside the planet's sphere of influence (SOI), using the technique described by Sarli et al. [204]. To accomplish this, a phase boundary is placed, by the optimizer, on the surface of the SOI and at a periapse point with respect to the planet. The SOI phase boundary connects the last flyby of the interplanetary cruise with the periapse phase boundary. The periapse phase boundary then connects to the first satellite flyby encounter. The positions of these two phase boundaries are referenced with respect to an ephemeris point of the target planet, not the sun. Assuming the spacecraft has a chemical propulsion system (i.e. the MGAnDSMs transcription is employed) the optimizer is free to place a maneuver (or maneuvers) anywhere along either phase. In addition, the optimizer can place a maneuver exactly at the periapse point, should it be optimal to perform the orbital insertion maneuver there.

Shooting arcs are propagated from both sides of each phase boundary, just as they are in a typical FBS transcription used for interplanetary cruise, and a defect constraint at the center of each phase must be satisfied. The various half-phase propagations and boundary locations are illustrated in Fig. 6.12. By locating a phase boundary on the SOI, any required coordinate system change occurs there as opposed to in the middle of a phase propagation, where it could complicate derivative calculations and cost function evaluation in general. The periapse phase boundary is defined via a constraint on the spacecraft's state at that point:

$$\mathbf{c}_{\mathrm{p}} = \mathbf{r} \cdot \mathbf{v} = \mathbf{0} \tag{6.11}$$

It is important to note that not all incoming hyperbolic trajectories that preface an orbital insertion maneuver will necessarily pass through a periapse point. For instance if a significant inclination change needs to be performed upon arrival at the target planetary system, the insertion maneuver will preferentially be located at a node crossing. Locating the orbital insertion maneuver at a node means that the subsequent periapse raise maneuver, which occurs near apoapse will also be located at or very close to the opposing node. The periapse raise maneuver and inclination change can be combined into a single maneuver in a location that is efficient for both orbital geometry adjustments (i.e. apoapse for a periapse raise maneuver and nodal crossing for an inclination change). While this work considers a periapse boundary, the framework discussed in this section does not preclude the use of other boundary types, including an arbitrary unconstrained free point in space.

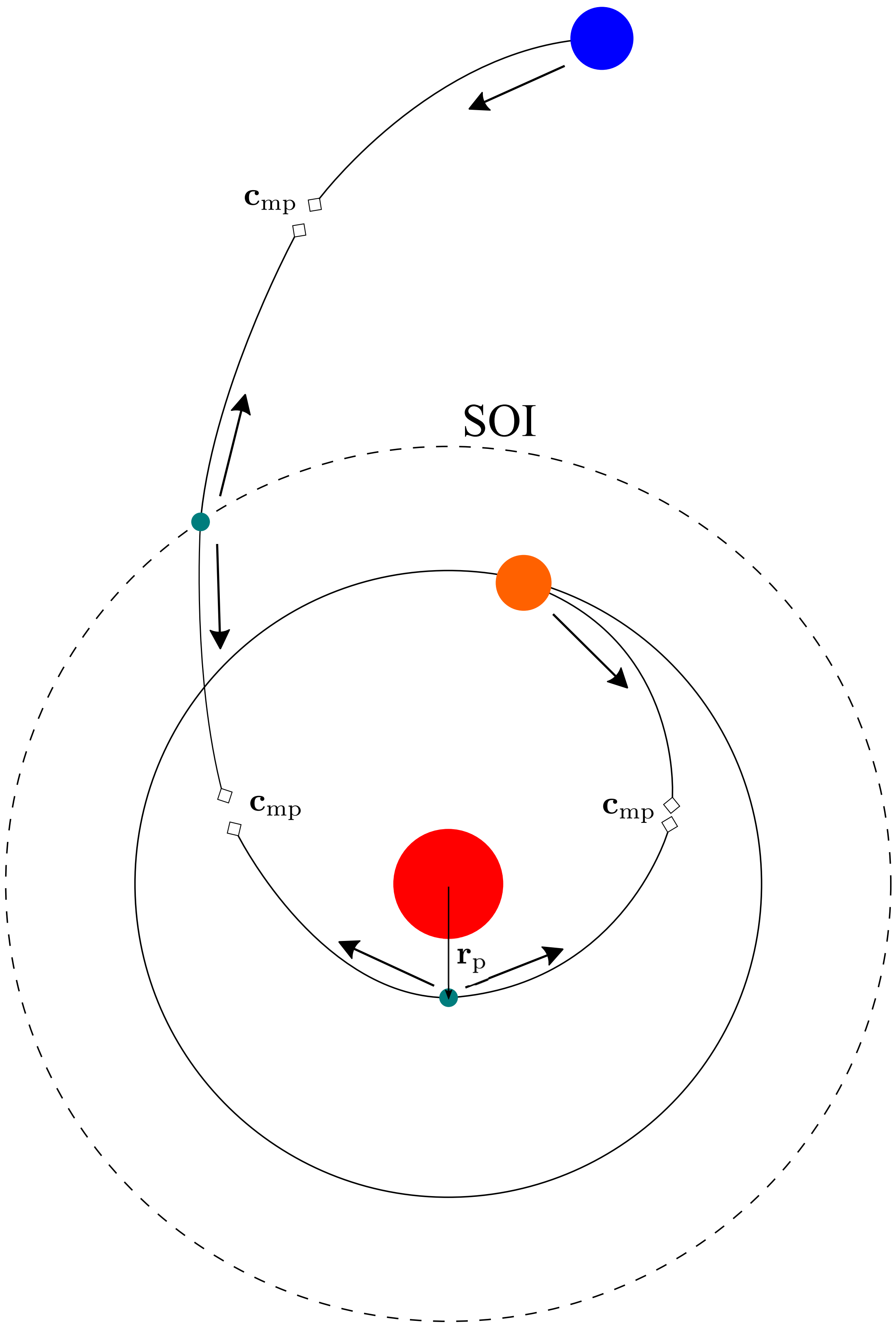

SOI
$\mathbf{c}_{\mathrm{mp}}$
$\mathbf{c}_{\mathrm{mp}}$
$\mathbf{c}_{\mathrm{mp}}$
$\mathbf{r}_{\mathrm{p}}$

Figure 6.12: Ephemeris referenced boundary multiple shooting procedure.

6.8.2 Satellite Tour Lambert Tree Search

After an optimized interplanetary trajectory and Jupiter capture sequence was found, the arrival $\mathbf{v}_\infty$ vector at the first Callisto flyby was used as an initial condition for a flyby tree search. The purpose of this search is to construct a patched-Lambert initial guess for a satellite tour that can be optimized in a gradient based solver.

The tree search was executed on a laptop with an Intel i7 processor with a 2.90 GHz nominal clock speed. OpenMP multithreading was used to accelerate the search. Deep-space maneuvers were not considered, except for use during resonance targets. A beam search was used where 20 000 trajectories were branched at every level (flyby) in the tree. The search required 5.37 hours and resulted in a trajectory tree consisting of 45 317 952 individual Lambert transfers. Branching selection considered partial trajectories that required the smallest Δv. Since no deep-space maneuvers were considered, this essentially favored feasible non-resonant transfers, which did not require a targeting maneuver. As a secondary selection criteria, the terminal maximum flyby bend angle was used to discern between sibling leaf nodes that had identical propellant requirements. Maximum bend angle was used as it is a good indicator of how effective a particular transfer will be at accessing a larger number of possible Lambert transfers during the next phase of branching. Connecting a large number of flybys together in the Jovian system is fairly straightforward due to the large masses of the Galilean moons, and 132 095 feasible 35 encounter solutions were identified by the tree search, which is a very small subset of all possible feasible solutions considering only 20 000 leaf nodes were branched at every level in the tree. In fact, some tree levels featured well over three million valid partial trajectories.

After the tree search was complete, the solution comprised of patched Lambert arcs was combined with the interplanetary cruise and Ganymede-Callisto capture sequence to create an initial guess for the gradient-based solver, which was then used to feasiblize and locally optimize the complete end-to-end trajectory from launch through tour completion. The tour parameters and flyby sequence is shown in Table 6.3.

Table 6.4: Optimization parameters for full interplanetary cruise + 35 flyby tour optimization (beyond those specified in Table 6.3).

Parameter	Value
Maximum flight time	11 years
Flyby altitude bounds for Galilean satellite flybys	[200 km, 10 000 km]
Flyby sequence	E-VVE-JSOI-GC-CCCCCCCCCC -GGGGGGGGGGGG -EEEEEEEEEEE
Arrival type	Europa intercept
Max SNOPT runtime	30 min

The MBH heuristic was not employed to perform this final optimization as the initial guess was nearly feasible and the use of a fully analytically specified Jacobian, using the techniques described in Chapter 3, ensured robust iteration of the local optimizer. The NLP solver was executed for 30 minutes to allow it to thoroughly refine the cost function. The final locally optimal tour beginning at Jupiter SOI is shown in Fig. 6.13. The orbital insertion and perijove raise maneuver details are shown in Fig. 6.14.

It should be stressed that this problem is a complete end-to-end optimization of a moon tour with a similar number of flybys as the mission executed by the Galileo spacecraft, and represents a significant improvement to the state-of-the-art of moon tour optimization. The 35 flyby example is by no means a stress case for this optimizer, as it was deployed in the latest NASA New Frontiers 4 proposal round where it was used to optimize tours of 60+ flybys with ease. The analytic techniques that were described in chapter 3 are an enabling feature of the solver that allows it to locally optimize a problem of this size and sensitivity. Furthermore, the entire mission pathsolving process was fully automated, from launch through final Europa flyby, and required no manual intervention by the analyst. Pathfinding (i.e. sequence selection) was not treated in this example, with a fixed flyby sequence being provided to the solver a priori.

The next step in the development of the automated moon tour generation framework is to optimize the trajectory in an n-body gravity model. During the tour, and especially while the spacecraft is flying on the initial capture petal orbit, the Sun is a significant perturber. Closer to Jupiter, the gravity of the Galilean satellites must also be accounted for. Some planets, Saturn in particular, have significant oblateness, which must be modeled in a practical tour design tool. The time domain fixed-step propagator described in chapter 5 is not suitable for high-fidelity optimization of planetary tours. In order to properly model the trajectory at both periapse and apoapse of the capture petal orbit a very small integration step size needs to be used, driven by the spacecraft's speed at periapse. Unfortunately, this results in prohibitively slow objective function execution speeds. One solution is to switch to a time-regularized version of the FBLT transcription, which is left as future work.

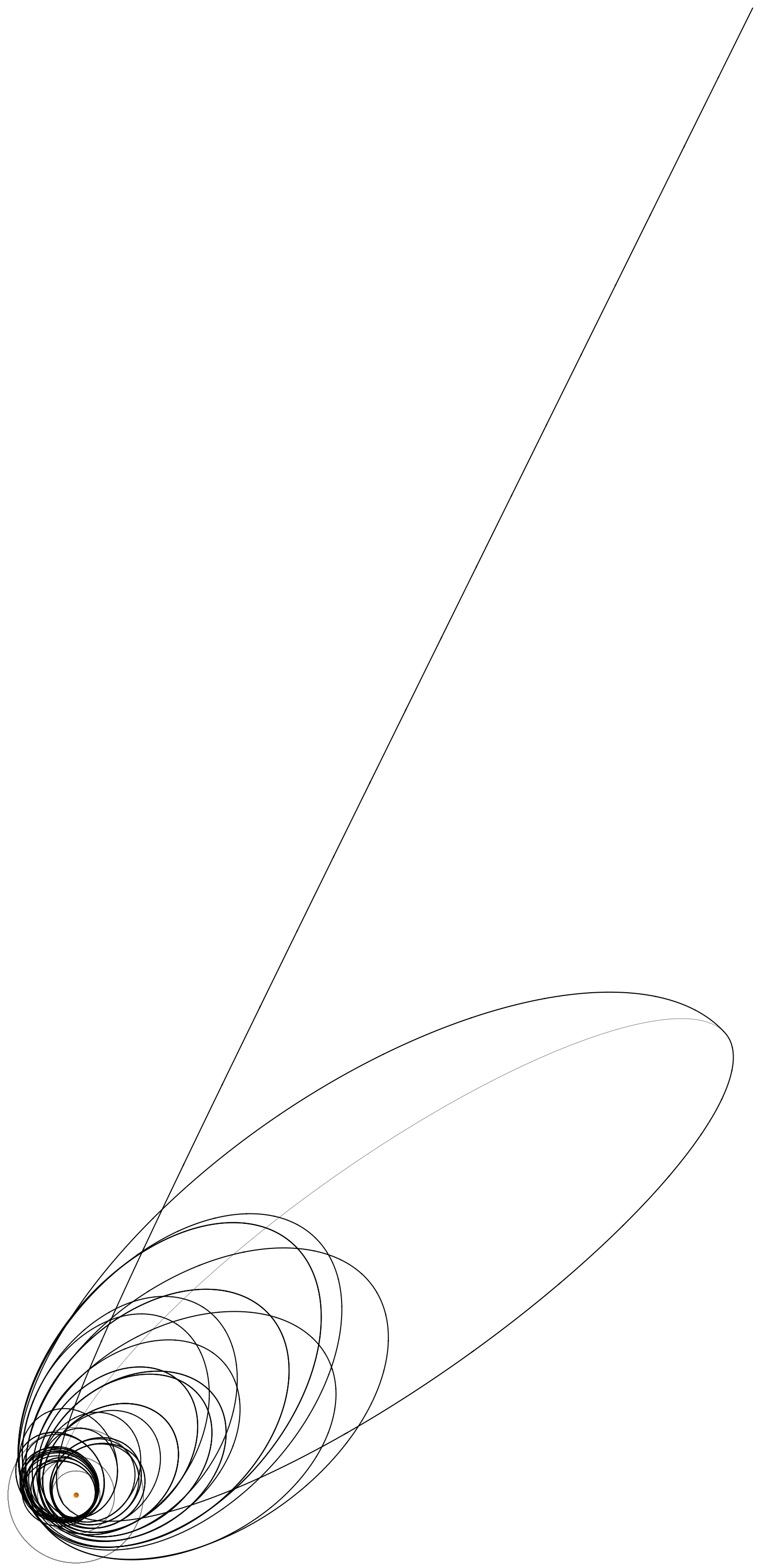


Figure 6.13: Locally optimized Galilean moon tour C:11, G:13, E:11, 3.82 year flight time from SOI interface to final Europa intercept. The large grey petal ellipse represents the osculating orbit of the spacecraft prior to the periapse raise maneuver that occurs at apojove of the first capture ellipse. 1.425 km/s deterministic Δv (JOI maneuver: 1.079 km/s, perijove raise maneuver: 0.346 km/s).

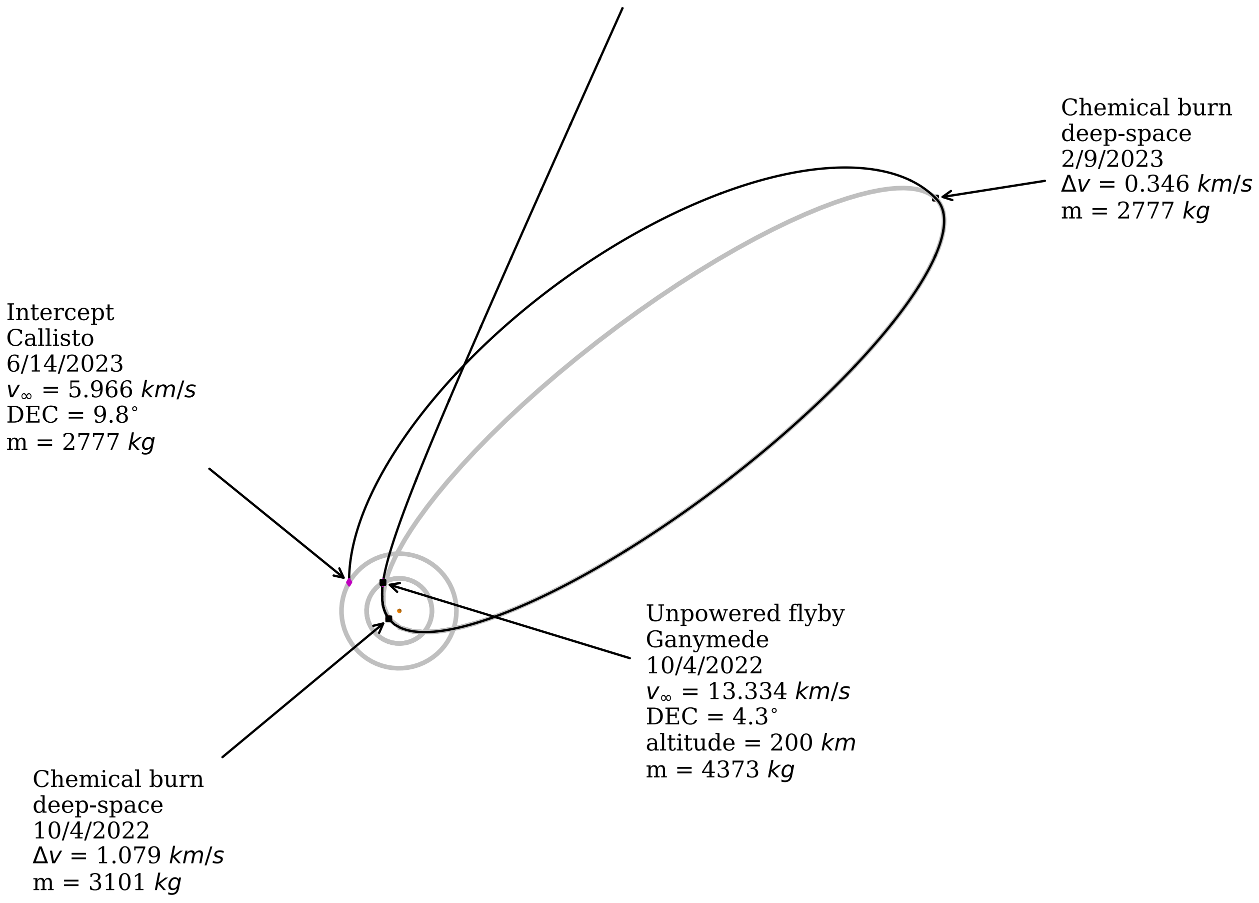


Figure 6.14: Jupiter orbit insertion and perijove raise maneuvers.

6.8.3 Future Search Improvements and Practical Navigation Considerations

The flyby pathfinding search performed for the satellite tour presented in the previous section identified low Δv feasible solutions and made no concessions for important operational factors that must be considered in the context of a real mission design effort. Probably the most important consideration that was ignored during the tree search phase of this tour design is the relative positioning of the Sun, Earth, central body, and the spacecraft during the satellite encounters and other critical mission events. In particular, it is vital to maintain a communication line-of-sight, free of radio interference, between the Earth and the spacecraft to confirm maneuver success. For example, during its Jupiter orbital insertion maneuver, Galileo had a direct line-of-sight to Earth so that entry probe data could be relayed to Earth and so Doppler ranging could confirm that the insertion burn had successfully occurred (Fig. 6.15).

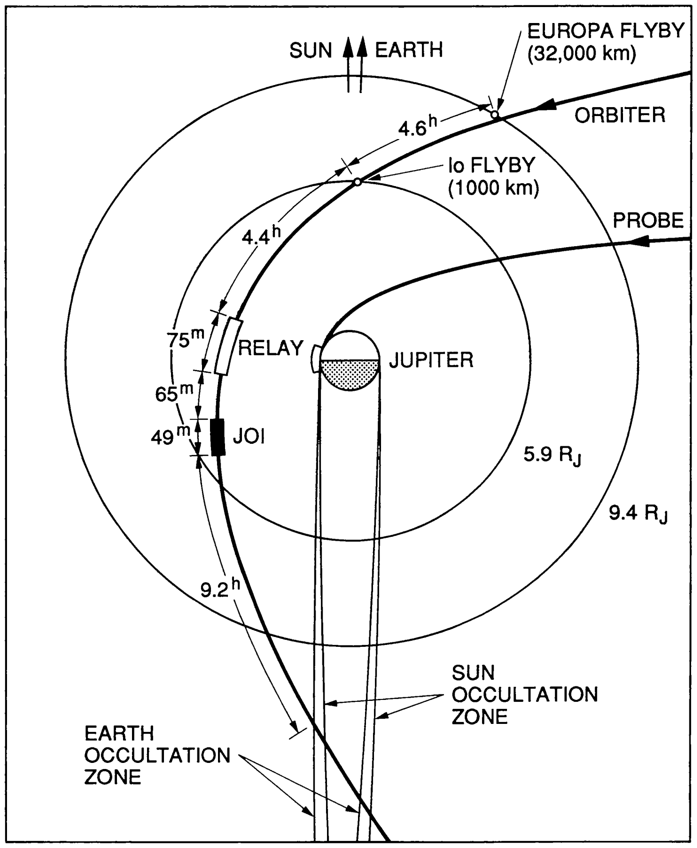


Figure 6.15: Jupiter arrival (December 7, 1995): Io flyby, Probe relay, and Jupiter orbit insertion (from [5]).

Three things could disrupt a communication link between the DSN and the spacecraft: 1) obstruction by the central body 2) obstruction by a satellite and 3) a solar conjunction. None of these problems arise for the tour presented in this chapter, however, these constraints were not explicitly enforced by the tree search algorithm or the local optimizer.

One particular concern that does exist with the mission presented is the encounter frequency for the Europa flybys in particular. Towards the end of the mission, some of the Europa encounters occur only six days apart, which is too high a frequency to ensure accurate navigation via Doppler ranging and Δ-DOR measurements from the deep space network. After an encounter, a spacecraft ground system must measure the flyby bend angle, perform updated orbit determination and then command the spacecraft should a trajectory correction be required. Current navigation methods, and spacecraft technology levels, do not allow for such a rapid operational cadence. These rapid encounters could easily be corrected by enforcing a minimum encounter-to-encounter flight time during the tree search. Comparing to the Galileo flight plan, a constraint was placed on the trajectory that no two satellite encounters could occur sooner than 35 days apart [5]. The Galilean prime tour is shown in Fig. 6.16 and the difference between that figure and Fig. 6.13 is noticeable. The semimajor axes of the Galileo prime orbits are noticeably larger than the majority of the orbits in the tour that was designed in this chapter (Fig. 6.13).

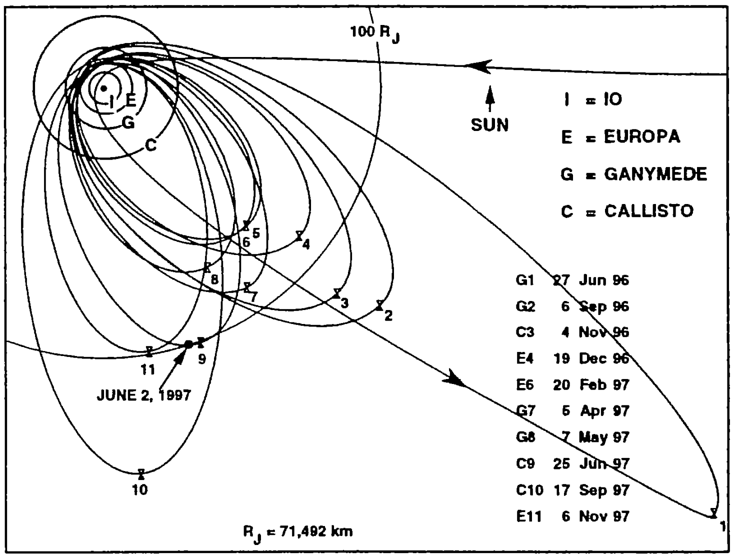


Figure 6.16: Satellite tour petal plot in the Jupiter-Sun rotating frame (from [6]).

Environmental hazards also pose a challenge to spacecraft performing satellite tours. The radiation environment at Jupiter, as well as the ring systems at Saturn and Uranus, make for dangerous operational environments. Previous studies have incorporated maximum radiation dosage constraints [129] as well as inclination reduction and ring avoidance [130, 205] into the mission design process. Radiation dosage measurements as well as physical hazard avoidance must be accounted for and should be incorporated into the tree search as well as the local gradient-based optimization routine. The tree traversal filter must account for these factors as the local optimizer will not always be able to sufficiently correct a trajectory that violates these hazard constraints.

Specific mission science objectives, risk tolerance levels, and spacecraft configurations will drive which specific events (e.g. flybys during a solar conjunction) are allowed or disallowed during flight, however, to be useful as a practical design tool, the tree search must be able to enforce practical constraints such as these during its filtering process should the practitioner require them to be. In addition, its filter should be easily configurable such that it can take any reasonable engineering constraint as an input. Development of a more flexible filter interface is left as future work.

Chapter 7

Conclusions

7.1 Bounded-Impulse Gradient Conclusions and Future Work

The material contained in chapters 2 and 3 is in essence a mathematical specification for an operational preliminary design framework. The primary contributions in these chapters are the analytic algorithms for computing the Jacobian matrix for two bounded-impulse trajectory transcriptions, one that models continuous-thrust spacecraft and the other that may be used to model high-thrust chemical trajectories.

An important characteristic of the particular calculation methods that are presented is that they allow for the inclusion of accurate spacecraft power models. The power models described have been made robust to switching events (e.g. thruster power cut-off thresholds) by modeling these discontinuous events using connected exponential fall-off functions that maintain gradient continuity. These power models are of sufficient accuracy for the preliminary design work that is typically performed for NASA Discovery/New Frontiers pre-Phase A proposal efforts.

Ephemeris data retrieval systems are also problematic for gradient-based optimizers. Frameworks such as SPICE, which is essentially a table lookup state machine, typically favor accuracy over suitability for use inside a gradient-based search algorithm. In order to preserve the smoothness of the time related variables in the transcriptions discussed, an algorithm is presented that fits SPICE ephemeris data to cubic B-splines creating a smooth ephemeris retrieval system that also maintains sufficient accuracy such that it may still be employed for practical preliminary design activities.

Finally, the benefits of implementing the analytic techniques described are illustrated through various practical example problems. The Uranus intercept and Venus spiral-down examples showcase the optimizer's improved ability to converge on the locally optimal solution when the solver is initialized in the vicinity of the solution. The improved convergence characteristics observed quantify the robustness improvements gained through the adoption of analytic Jacobian calculations in favor of using the method of finite-differences. The Odysseus rendezvous and Cassini benchmark examples explore the improvements exhibited by a gradient-based optimizer operating in cooperation with a stochastic search heuristic. The Odysseus problem demonstrates that the a non-smooth ephemeris model is particularly problematic and noticeably degrades the robustness of the solver even when all non-time related Jacobian entries are specified analytically.

Despite the flexibility of the techniques presented at incorporating various practical constraints, propulsion system variants and mission types, there remains a fidelity gap between the models presented in this work and hardware realizations that they represent. While polynomial mass-flow rate and generated power curves are substantially more accurate than assuming constant array power or even a "one over r squared" model, true electric propulsion systems are significantly more complex. The PPU of a modern ion thruster such as NEXT operates in a two dimensional setting grid with mass flow rate and grid voltage being the two input controls, not simply a single throttle parameter as was used in this work. Recently Knittel et al. [55] began an effort to capture the complexity of these physical systems more accurately in the trajectory models with methods that remain compatible with gradient-based optimizers. While Knittel's work represents the state-of-the-art in this area of research, there remains significant potential to improve upon his methods. In particular, the 2D throttle grid models that they produced are not as robust when employed by an NLP solver compared to the 1D polynomial curve fits. Future work should address this issue.

7.2 Time Regularized Bounded-Impulse Model Conclusions and Future Work

The development of the MGALTS transcription represents one of the first significant innovations on the Sims-Flanagan transcription since its introduction nearly twenty years ago. The primary shortcoming of the original Sims-Flanagan model is its equal temporal spacing of the control nodes. For many trajectories, spacing control nodes in time is sufficient for a medium-fidelity study, however, certain mission classes are enabled by a more efficient discretization. Transfers to highly eccentric taeets, for example, benefit greatly from clustering control nodes at periapsis where orbital energy changes are more efficiently executed.

Transformation of the orbital differential equations of motion using a Sundman transformation, which changes the independent variable of integration, is a popular method for regularizing a propagator that uses numerical integration. Regularization of the medium-fidelity Sims-Flanagan model is less straightforward. An independent variable change from time to an angular anomaly is still performed, however, unlike an integration scheme where the only required change is a slight alteration to the differential equations, changing the independent variable for the two-body initial value problem effectively changes the nature of the problem being solved. Given a flight time, solving Kepler's equation for change in angular position always requires an iterative root-finding algorithm. The procedure introduced in chapter 4 inverts the orbital initial value problem in a sense, computing a time-of-flight from a given change in angular anomaly. Then, by introducing new time decision variables and constraints, the angular independent variable selected by the nonlinear optimizer can be rectified with the time constraints placed on the trajectory by the mission designer.

The new regularized low-thrust bounded-impulse transcription is flexible in that it retains the same basic structure and components as the Sims-Flanagan model (e.g. Kepler solver), so an existing implementation of the latter can be extended to the new regularized version. It is also straightforward to adopt different independent angular variables via algebraic manipulations of the universal Kepler solver. Finally, as shown in chapter 4 analytic derivatives can also be computed for this transcription.

A valuable future research effort would be to perform a thorough analysis of the robustness and general relative performance of this new transcription compared to the original Sims-Flanagan algorithm. Since MGALTS removes the need to iteratively solve Kepler's equation, it is expected that this new model will execute more quickly, however, since it also requires the addition of new NLP decision variables and constraints that affect the SQP iteration, the execution speeds of the two algorithms could be similar. It is suspected that high-eccentricity transfers are not the only mission types that will benefit from a time-regularized transcription. A comparative study that illustrates the merits of adopting MGALTS for other trajectory families would be a valuable future contribution.

7.3 Finite-Burn Gradient Conclusions and Future Work

The medium-fidelity finite-burn trajectory transcription described in this work was originally introduced by Englander et al. [159] in order to bridge the gap between the low-fidelity bounded impulse models such as those described in chapter 3 and high fidelity tools such as GMAT. The FBLT transcription shares the same decision vector as the MGALT model, and therefore facilitates an automated process for converting a low-fidelity bounded-impulse solution to a patched-integrated version of the same trajectory. The introduction of numerical integration allows for the straightforward inclusion of perturbing accelerations by way of Cowell's method. The transition from bounded-impulse to a patched-integrated trajectory that models n-body gravity and solar radiation pressure was illustrated by reconverging the comet sample return example presented in chapter 3.

Chapter 5 describes numerical methods for computing the Jacobian matrix of the FBLT model that allow for the incorporation of the power and ephemeris models presented in chapter 2. An algorithm for computing time-of-flight derivatives was also described that allows one to solve final time free optimization problems. These methods are based on the variational equations for integrating a numerical state transition matrix, and therefore are only accurate to machine precision when used with a fixed-step integration scheme. Applying the Sundman transformation to the FBLT model would be a natural extension to the current transcription and would alleviate some of the shortcomings of the current time discretization.

The utility of a particular trajectory model is directly related to how physically realistic it is. With this in mind, analysis was performed with help from engineers at KinetX Aerospace that determined how accurately the operational navigation software MIRAGE can track a reference trajectory optimized using the FBLT model. The validation exercise showed that MIRAGE could track an FBLT trajectory to within operational tolerances. This exercise is important as it reinforces the notion that an FBLT trajectory is "flyable" even though it is not actually flight fidelity.

The current two-sided multiple shooting evaluation of an FBLT trajectory can exhibit sensitivity for complex trajectories. Development of a parallel shooting variant of the FBLT model would be a straightforward extension of the work presented in Chapter 5. The gradient computations for such a transcription would be simplified as the single phase match point would be replaced with multiple match points along the phase, which would eliminate the need for long multiplication chains of state transition matrices. Explicit Runge-Kutta integration could be retained [38], however, an investigation into convergence properties using other solution techniques (i.e. implicit integration) would be valuable. Adoption of a parallel shooting variant of the FBLT transcription would also allow one to model a discrete duty cycle concept of operations (CONOPS). The parallel shooting subdivisions of a low-thrust phase make it straightforward to impose forced coast periods required for navigation activities such as communications with Earth and orbit determination. For this reason alone, development of a high-fidelity parallel shooting variant of FBLT would be an important contribution.

7.4 Satellite Tour Automaton Conclusions and Future Work

The first contribution of chapter 6 is a tree traversal algorithm that leverages modern software engineering concepts such as object pooling and multithread parallelism using OpenMP. The tree search routine can be deployed on a variety of platforms from single-node laptops to multi-core high-memory bandwidth platforms. The algorithm seeks to maximize execution speed and uses an ephemeris system that linearly interpolates body state data points from SPICE. The tree search framework also supports the inclusion of deep space maneuvers and near-resonant transfers. The tree search algorithm and associated data structures off a significantly more flexible pathfinding solution than a total enumeration grid search of the MGA problem. The reverse linked list data structure can accommodate a wide variety of tree exploration programs and filtering criteria.

The second contribution described in Chapter 6 is the formulation of a trajectory optimization automaton that is capable of performing a continuous end-to-end optimization of a moon tour mission from launch at Earth through tour completion. The framework models a finite SOI at the target planet where the tour occurs and the optimizer is capable of autonomously placing orbital insertion maneuvers to ensure capture around the planet. A piecewise Lambert moon tour solution is constructed using the parallel tree search and is provided as a seed to a local gradient-based optimizer that employs multiple shooting as well as the analytic Jacobian calculation methods described in Chapter 3. This workflow is shown to be capable of optimizing tours with tens of gravity-assists using a two-body patched-conic gravity model, and is predicted to have no trouble converging tours consisting of well over 100 flybys in a single optimization run.

Potential future work for the satellite tour automaton could focus on the computer science aspects of the Lambert search algorithm, tree exploration and search filtering strategies or trajectory modeling fidelity. A hybrid parallel deployment of the tree search on computational infrastructure that combines multithread parallelism as well as multinode distributed parallelism using a standard such as message passing interface (MPI) is one possible research vector. Scaling of the search to larger distributed clusters would allow for the large design spaces associated with satellite tours to be more efficiently and thoroughly explored. This work did not place particular emphasis on filtering strategies for the breadth-first tree search, and future investigations into how to effectively prune partial solutions from the tree, yet still adequately explore the design space would have a great deal of practical value.

Incorporation of maneuvers into the Lambert search drastically increases the complexity of the trajectory combinatorics and a method that relies less on exhaustive exploration would be valuable, with recent work by Landau based on primer vector theory being especially promising [206]. Finally, an important next step in the moon tour optimization research effort is to increase the fidelity by replacing the Keplerian propagator with an integrated one. Doing so will require applying a regularization transformation (e.g. Sundman) to the FBLT model in order to accommodate the large disparity in integration step sizes needed for the various stages of a satellite tour.

Appendix A

A.1 Monotonic Basin Hopping

Monotonic basin hopping is a method that searches for the global extremum of problems that have many local optima, which is a characteristic of most trajectory optimization problems. The MBH algorithm was first applied in the field of computational chemistry for solving molecular conformation problems [107]. The incorporation of this stochastic search technique into a preliminary trajectory design workflow is a relatively recent advancement that was pioneered by Yam et al. [110, 111]. In that work, MBH was combined with a bounded-impulse low-thrust trajectory transcription. Since then, several research efforts have incorporated MBH into the trajectory optimization process [47, 48, 78, 90, 109, 207, 208]. The primary advantage of using MBH to guide a local search is that the algorithm does not require an initial guess in order to begin its iteration, it can be initialized from a random decision vector. The MBH algorithm is described in Figure A.1. The algorithm is initiated by running the local optimizer on a randomly initialized decision vector $\mathbf{x}$. If a feasible point is obtained, then it is stored as the current point $\mathbf{x}_\mathrm{c}$ if its cost function is better than the incumbent current point. The decision vector is then randomly perturbed before the local optimizer is executed again. The MBH algorithm is often described as a guided multi-start.

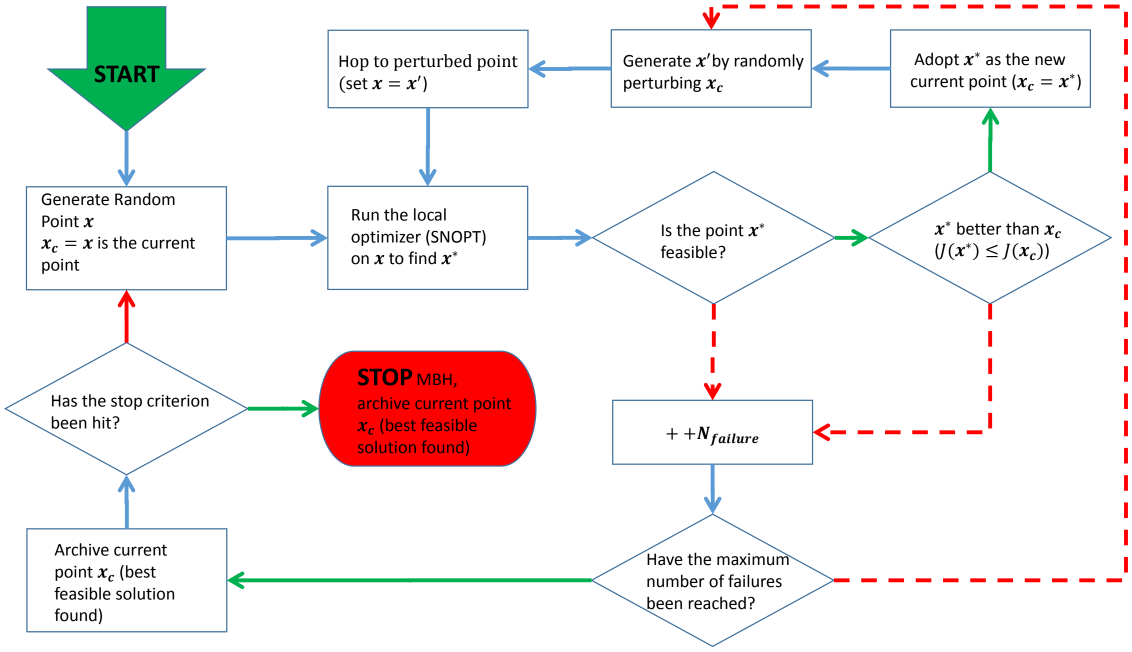

START
Generate Random Point $\boldsymbol{x}$ $\boldsymbol{x_c} = \boldsymbol{x}$ is the current point
Run the local optimizer (SNOPT) on $\boldsymbol{x}$ to find $\boldsymbol{x}^*$
Hop to perturbed point (set $\boldsymbol{x} = \boldsymbol{x}'$)
Generate $\boldsymbol{x}'$ by randomly perturbing $\boldsymbol{x_c}$
Adopt $\boldsymbol{x}^*$ as the new current point ($\boldsymbol{x_c} = \boldsymbol{x}^*$)
Is the point $\boldsymbol{x}^*$ feasible?
$\boldsymbol{x}^*$ better than $\boldsymbol{x_c}$ ($J(\boldsymbol{x}^*) \leq J(\boldsymbol{x_c})$)
$++N_{failure}$
Have the maximum number of failures been reached?
Archive current point $\boldsymbol{x_c}$ (best feasible solution found)
Has the stop criterion been hit?
STOP MBH, archive current point $\boldsymbol{x_c}$ (best feasible solution found)

Figure A.1: Monotonic basin hopping algorithm flowchart.

During each MBH iteration, the new decision vector $\mathbf{x}'$ is generated by perturbing each element of $\mathbf{x}$ using a random number drawn from a probability distribution. The original MBH algorithm performed this perturbation using a uniform distribution, however, more recent research by Englander [48] and Englander and Englander [112] has shown that heavy-tailed distributions such as Cauchy and Pareto distributions are significantly better-suited to trajectory optimization problems.

A.2 Thruster Performance Curves

Thrust polynomial coefficients provide a relationship between input power (kW) and the maximum possible applied thrust (mN). Mass flow rate coefficients provide a curve fit in mg/s.

Table A.1: Thruster performance parameters.

Parameter	NEXT TT10 High-I_{sp}	NEXT TT11 [131] High-Thrust	NSTAR [209]
a_T	-1.92224e-6	11.9388817	26.337459
b_T	54.05382	16.0989424	-51.694393
c_T	-14.41789	11.4181412	90.486509
d_T	2.96519	-2.04053417	-36.720293
e_T	-0.19082	0.101855017	5.145602
a_m	2.13781	2.75956482	2.5060
b_m	0.03211	-1.71102132	-5.3568
c_m	-0.09956	1.21670237	6.2539
d_m	0.05717	-0.207253445	-2.5372
e_m	-0.004776	0.011021367	0.36985
$P_{\max}$ (kW)	7.266	7.36	2.6
$P_{\min}$ (kW)	0.638	0.64	0.525

A.3 Fundamental Perturbation STM

Vector expressions for the 3x3 perturbation STM partitions [160]:

$$\begin{aligned}\tilde{\mathbf{R}} =& \frac{r_{k+1}}{\mu}(\mathbf{v}_{k+1}-\mathbf{v}_k)(\mathbf{v}_{k+1}-\mathbf{v}_k)^T \\ &+ \frac{1}{r_k^3}\left[r_k(1-F)\mathbf{r}_{k+1}\mathbf{r}_k^T + C\mathbf{v}_{k+1}\mathbf{r}_k^T\right] + F\mathbb{I}_{3\times3} \qquad \text{(A.1)}\\ \mathbf{R} =& \frac{r_k}{\mu}(1-F)\left[(\mathbf{r}_{k+1}-\mathbf{r}_k)\mathbf{v}_k^T - (\mathbf{v}_{k+1}-\mathbf{v}_k)\mathbf{r}_k^T\right] \\ &+ \frac{C}{\mu}\mathbf{v}_{k+1}\mathbf{v}_k^T + G\mathbb{I}_{3\times3} \qquad \text{(A.2)}\\ \tilde{\mathbf{V}} =& -\frac{1}{r_k^2}(\mathbf{v}_{k+1}-\mathbf{v}_k)\mathbf{r}_k^T - \frac{1}{r_{k+1}^2}\mathbf{r}_{k+1}(\mathbf{v}_{k+1}-\mathbf{v}_k)^T \\ &+ F_t\Bigg[\mathbb{I}_{3\times3} - \frac{1}{r_{k+1}^2}\mathbf{r}_{k+1}\mathbf{r}_{k+1}^T \\ &+ \frac{1}{\mu r_{k+1}}(\mathbf{r}_{k+1}\mathbf{v}_{k+1}^T - \mathbf{v}_{k+1}\mathbf{r}_{k+1}^T)\mathbf{r}_{k+1}(\mathbf{v}_{k+1}-\mathbf{v}_k)^T\Bigg] \\ &- \frac{\mu C}{r_{k+1}^3 r_k^3}\mathbf{r}_{k+1}\mathbf{r}_k^T \qquad \text{(A.3)}\\ \mathbf{V} =& \frac{r_k}{\mu}(\mathbf{v}_{k+1}-\mathbf{v}_k)(\mathbf{v}_{k+1}-\mathbf{v}_k)^T \\ &+ \frac{1}{r_{k+1}^3}\left[r_k(1-F)\mathbf{r}_{k+1}\mathbf{r}_k^T - C\mathbf{r}_{k+1}\mathbf{v}_k^T\right] + G_t\mathbb{I}_{3\times3} \qquad \text{(A.4)}\end{aligned}$$

In Equations (A.1) - (A.4), $\mathbb{I}$ is the 3x3 identity matrix. The quantity C is calculated using the universal functions:

$$\sqrt{\mu}C = 3U_5 - \chi U_4 - \sqrt{\mu}(t-t_0)U_2 \qquad \text{(A.5)}$$

The functions U_n have traditionally been calculated with continued fraction expansions [160,163], however, the following recursion relation can be used to compute higher-order universal functions:

$$U_n(\chi,\alpha) + \alpha U_{n+2}(\chi,\alpha) = \frac{\chi^n}{n!} \qquad n = 0,1,2,3,... \qquad \text{(A.6)}$$

A.4 Runge Kutta Eighth Order Butcher Tableau

c_i	a_{ij}												$\hat{b}_i$	b_i
0													$\frac{14005451}{335480064}$	$\frac{13451932}{455176623}$
$\frac{1}{18}$	$\frac{1}{18}$												0	0
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$											0	0
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$										0	0
$\frac{5}{16}$	$\frac{5}{16}$	0	$\frac{-75}{64}$	$\frac{75}{64}$									0	0
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$								$\frac{-59238493}{1068277825}$	$\frac{-808719846}{976000145}$
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736538}{692538347}$	$\frac{-28693883}{1125000000}$	$\frac{23124283}{1800000000}$							$\frac{181606767}{758867731}$	$\frac{1757004468}{5645159321}$
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$\frac{-180193667}{1043307555}$						$\frac{561292985}{797845732}$	$\frac{656045339}{265891186}$
$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$\frac{-433636366}{683701615}$	$\frac{-421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$					$\frac{-1041891430}{1371343529}$	$\frac{-3867574721}{1518517206}$
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$\frac{-37695042795}{15268766246}$	$\frac{-309121744}{1061227803}$	$\frac{-12992083}{490766935}$	$\frac{6005943493}{2108947869}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$				$\frac{760417239}{1151165299}$	$\frac{465885868}{322736535}$
$\frac{1201146811}{1299019798}$	$\frac{-1028468189}{846180014}$	0	0	$\frac{8478235783}{508512852}$	$\frac{1311729495}{1432422823}$	$\frac{-10304129995}{1701304382}$	$\frac{-48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$\frac{-45442868181}{3398467696}$	$\frac{3065993473}{597172653}$			$\frac{118820643}{751138087}$	$\frac{53011238}{667516719}$
1	$\frac{185892177}{718116043}$	0	0	$\frac{-3185094517}{667107341}$	$\frac{-477755414}{1098053517}$	$\frac{-703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$\frac{-4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$		$\frac{-528747749}{2220607170}$	$\frac{2}{45}$
1	$\frac{403863854}{491063109}$	0	0	$\frac{-5068492393}{434740067}$	$\frac{-411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$\frac{-13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$\frac{-160528059}{685178525}$	$\frac{248638103}{1413531060}$	0	$\frac{1}{4}$	0

Table A.2: Butcher tableau for the Dormand and Prince $RK7(8)13M$ method [1]

A.5 IAU Cartographic and Rotational Calculations

An orientation/rotation model of the planets is required for the implementation of a high-order gravity field model, or computing trajectories that may span multiple reference frames in general. The rotational model used for this work, and also implemented in the EMTG software, is defined by the International Astronomical Union (IAU) Working Group on Cartographic Coordinates and Rotational Elements [7].

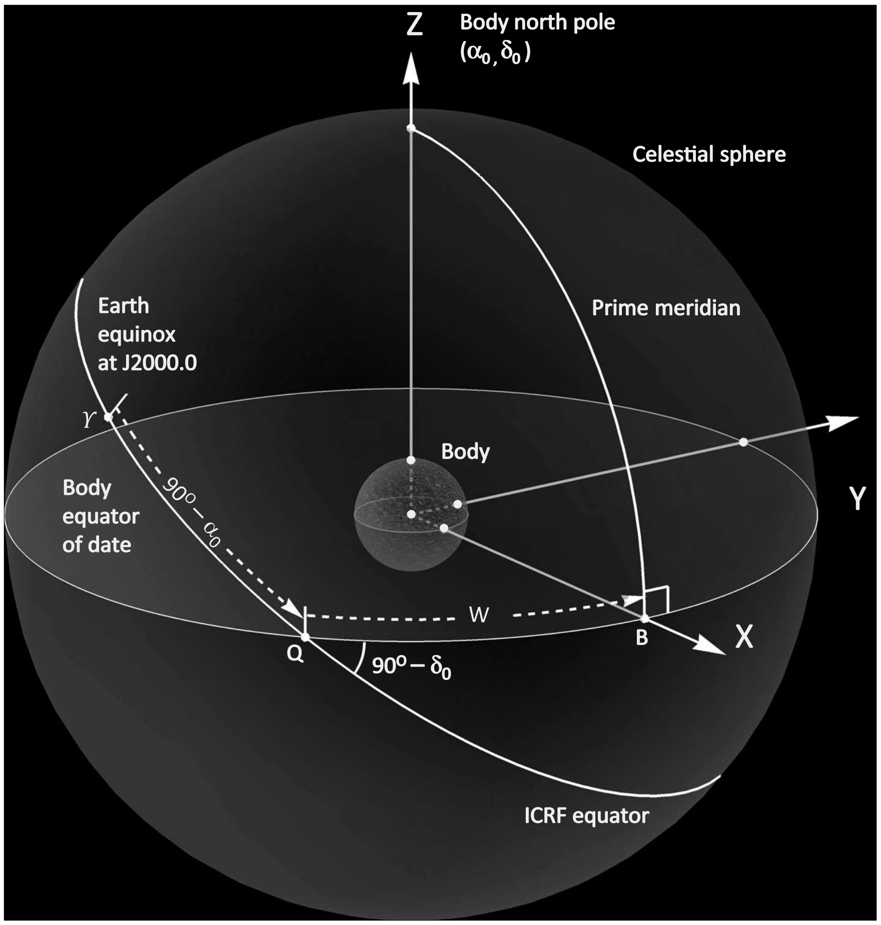


Figure A.2: Reproduced from [7]. Reference system used to define orientation of the planets and their satellites. For $\dot{W}(t) > 0$, body rotation is prograde (e.g.,Mercury, Jupiter). For $\dot{W}(t) < 0$, body rotation is retrograde (e.g., Venus, Uranus)

A body's mean axis of rotation and body-dependent longitudinal point define the planetary coordinate systems. The IAU has developed approximate expressions for the rotational elements that relate the planetary coordinates to the ICRF reference frame. The ICRF frame is an epochless frame, which differs from the J2000.0 mean dynamical frame by a rotation of less than 0.1 arcs.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\pi/2 - \delta_0) & -\sin(\pi/2 - \delta_0) \\ 0 & \sin(\pi/2 - \delta_0) & \cos(\pi/2 - \delta_0) \end{bmatrix} \tag{A.7}$$

$$R_z = \begin{bmatrix} \cos(\pi/2 + \alpha_0) & -\sin(\pi/2 + \alpha_0) & 0 \\ \sin(\pi/2 + \alpha_0) & \cos(\pi/2 + \alpha_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.8}$$

The rotation matrix that maps body-centered inertial coordinates to the ICRF frame calculated as follows:

$$R_{\text{BCI}}^{\text{ICRF}} = R_z R_x \tag{A.9}$$

Similarly, the rotation matrix that maps body-centered fixed coordinates to body-centered inertial coordinates is:

$$R_{\text{BCF}}^{\text{BCI}} = \begin{bmatrix} \cos(W) & -\sin(W) & 0 \\ \sin(W) & \cos(W) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.10}$$

The variable quantities in Eq. (A.7)-(A.10) are expressed in units of days (86400 seconds) or Julian centuries (36 525 days). For example, the angular quantities for Jupiter are:

$$
\begin{aligned}
\alpha_0 = {} & 268.056595^\circ - 0.006499^\circ T \\
& + 0.000117^\circ \ \sin(99.360714^\circ + 4850.4046^\circ T) \\
& + 0.000938^\circ \ \sin(175.895369^\circ + 1191.9605^\circ T) \\
& + 0.001432^\circ \ \sin(300.323162^\circ + 262.5475^\circ T) \\
& + 0.000030^\circ \ \sin(114.012305^\circ + 6070.2476^\circ T) \\
& + 0.002150^\circ \ \sin(49.511251^\circ + 64.3000^\circ T)
\end{aligned} \tag{A.11}
$$

$$
\begin{aligned}
\delta_0 = {} & 64.495303^\circ - 0.002413^\circ T \\
& + 0.000050^\circ \ \cos(99.360714^\circ + 4850.4046^\circ T) \\
& + 0.000404^\circ \ \cos(175.895369^\circ + 1191.9605^\circ T) \\
& + 0.000617^\circ \ \cos(300.323162^\circ + 262.5475^\circ T) \\
& + 0.000013^\circ \ \cos(114.012305^\circ + 6070.2476^\circ T) \\
& + 0.000926^\circ \ \cos(49.511251^\circ + 64.3000^\circ T)
\end{aligned} \tag{A.12}
$$

$$
W = 284.95^\circ + 870.5360000d \tag{A.13}
$$

where T is the number of Julian centuries since the standard epoch and d is the interval in days from the standard epoch.

References

[1] Prince, P. J. and Dormand, J. R., "High order embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, Vol. 7, No. 1, 1981, pp. 67–75.

[2] Hughes, K. M., *Gravity-Assist Trajectories to Venus, Mars, and The Ice Giants: Mission Design with Human and Robotic Applications*, Ph.D. thesis, Purdue University, 2016.

[3] Strange, N., Russell, R., and Buffington, B., "Mapping the V-infinity globe," in "Astrodynamics 2007 - Advances in the Astronautical Sciences, Proceedings of the AAS/AIAA Astrodynamics Specialist Conference," Vol. 129 PART 1, 2007, pp. 423–446.

[4] Bonfiglio, E. P., Longuski, J. M., and Vinh, N. X., "Automated Design of Aerogravity-Assist Trajectories," *Journal of Spacecraft and Rockets*, Vol. 37, No. 6, 2000, pp. 768–775, doi:10.2514/2.3649.

[5] D'Amario, L. A., Bright, L. E., and Wolf, A. A., "Galileo Trajectory Design," *Space Science Reviews*, Vol. 60, 1992, pp. 23–78.

[6] Wilson, M. G., Potts, C. L., Mase, R. A., Halsell, C. A., and Byrnes, D. V., "Maneuver Design for Galileo Jupiter Approach and Orbital Operations," in "12th International Symposium on Space Flight Dynamics," Darmstadt, Germany, 1997.

[7] Archinal, B., Acton, C. H., A'Hearn, M. F., Conrad, A., Consolmagno, G. J., Duxbury, T., Hestroffer, D., Hilton, J. L., Kirk, R. L., Klioner, S. A., McCarthy, D., Meech, K., Oberst, J., Ping, J., Seidelmann, P. K., Tholen, D. J., Thomas, P. C., and Williams, I. P., "Report of the IAU Working Group on Cartographic Coordinates and Rotational Elements: 2015," *Celestial Mechanics and Dynamical Astronomy*, Vol. 130, No. 22, 2018, pp. 1–10, doi:10.1007/s10569-017-9805-5.

[8] Rinderle, E. A., "Galileo User's Guide, Mission Design System, Satellite Tour Analysis and Design Subsystem," Tech. Rep. JPL D-263, Jet Propulsion Laboratory, California Institute of Technology, 1986.

[9] Lantukh, D. V., *Preliminary Design of Spacecraft Trajectories for Missions to Outer Planets and Small Bodies*, Ph.D. thesis, The University of Texas at Austin, 2015.

[10] Petropoulos, A. E. and Longuski, J. M., "Shape-Based Algorithm for Automated Design of Low-Thrust, Gravity-Assist Trajectories," *Journal of Spacecraft and Rockets*, Vol. 41, No. 5, 2004, pp. 787–796, doi:10.2514/1.13095.

[11] Wall, B. J. and Conway, B. A., "Shape-Based Approach to Low-Thrust Rendezvous Trajectory Design," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 95–101, doi:10.2514/1.36848.

[12] De Pascale, P. and Vasile, M., "Preliminary Design of Low-Thrust Multiple Gravity-Assist Trajectories," *Journal of Spacecraft and Rockets*, Vol. 43, No. 5, 2006, pp. 1065–1076, doi:10.2514/1.19646.

[13] Vasile, M., De Pascale, P., and Casotto, S., "On the optimality of a shape-based approach based on pseudo-equinoctial elements," *Acta Astronautica*, Vol. 61, No. 1-6, 2007, pp. 286–297, doi:10.1016/j.actaastro.2007.01.017.

[14] Taheri, E. and Abdelkhalik, O., "Shape-Based Approximation of Constrained Low-Thrust Space Trajectories Using Fourier Series," *Journal of Spacecraft and Rockets*, Vol. 49, No. 3, 2012, pp. 535–545, doi:10.2514/1.A32099.

[15] Sims, J. A. and Flanagan, S. N., "Preliminary Design of Low-Thrust Interplanetary Missions," in "AAS/AIAA Astrodynamics Specialist Conference," Paper AAS 99-338, Girdwood, Alaska, 1999.

[16] Englander, J. A., Ellison, D. H., and Conway, B. A., "Global Optimization of Low-Thrust, Multiple-Flyby Trajectories at Medium and Medium-High Fidelity," in "AAS/AIAA Space-Flight Mechanics Meeting, Santa Fe, NM," AAS, 2014.

[17] Hong Yam, C., Izzo, D., and Biscani, F., "Towards a High Fidelity Direct Transcription Method for Optimisation of Low-Thrust Trajectories," in "International Conference on Astrodynamics Tools and Techniques," Madrid, 2010.

[18] Byrnes, D. V. and Hooper, H. L., "AIAA 70-1062 Multi-Conic: A Fast and Accurate Method of Computing Space Flight Trajectories," in "AAS/AIAA Astrodynamics Conference," Santa Barbara, CA, 1970,
doi:10.2514/6.1970-1062.

[19] D'Amario, L. A., *Minimum Impulse Three-Body Trajectories*, Ph.D. thesis, Massachusetts Institute of Technology, 1973.

[20] Wilson, R. S. and Howell, K. C., "Trajectory Design in the Sun ? Earth ? Moon System Using Lunar Gravity Assists," *Journal of Spacecraft and Rockets*, Vol. 35, No. 2, 1998, pp. 191–198, doi:10.2514/2.3309.

[21] Danby, J., *Fundamentals of celestial mechanics*, Macmillan, 1962.

[22] Diehl, R. E., Kaplan, D. I., and Penzo, P. A., "AIAA-83-0101 Satellite Tour Design for the Galileo Mission," in "AIAA 21st Aerospace Sciences Meeting," Reno, NV, 1983,
doi:10.2514/6.1983-101.

[23] "General Mission Analysis Tool," , 2017. `https://gmat.gsfc.nasa.gov/`.

[24] "FreeFlyer," , 2017. `https://ai-solutions.com/freeflyer/`.

[25] "Systems Tool Kit," , 2017. `https://www.agi.com/products/engineering-tools`.

[26] Ocampo, C., *Elements of a Software System for Spacecraft Trajectory Optimization*, Cambridge University Press, pp. 79–111, 2010.

[27] Whiffen, G. J., "Mystic: Implementation of the Static Dynamic Optimal Control Algorithm for High-Fidelity, Low-Thrust Trajectory Design," in "AIAA/AAS Astrodynamics Specialist Conference and Exhibit, AIAA Paper 2006-6741," Keystone, Colorado, 2006,
doi:10.2514/6.2006-6741.

[28] Breakwell, J. V., "The Optimization of Trajectories," *Journal of the Society of Industrial and Applied Mathematics (SIAM)*, Vol. 7, No. 2, 1959, pp. 215–247,
doi:10.1137/0107018.

[29] Lawden, D. F., *Optimal Trajectories for Space Navigation*, Butterworths Mathematical Texts, Butterworths, London, 1963.

[30] Lion, P. M. and Handelsman, M., "Primer Vector on Fixed-Time Impulsive Trajectories," *AIAA Journal*, Vol. 5, No. 1, 1968, pp. 127–132,
doi:10.2514/3.4452.

[31] Merec, J.-P., *Optimal Space Trajectories*, Studies in Astronautics, Elsevier Scientific Publishing Company, 1979.

[32] Melbourne, W. G. and Sauer, J., C. G., "Optimum Interplanetary Rendezvous Trajectories with Power Limited Vehicles," *AIAA Journal*, Vol. 1, 1963, pp. 54–60,
doi:10.2514/3.1468.

[33] Sauer, J., C. G., "Optimization of Multiple Target Electric Propulsion Trajectories," in "AIAA Paper 73-205," , 1975,
doi:10.2514/6.1973-205.

[34] Jezewski, D. J., "Primer Vector Theory and Applications," Tech. Rep. TR-R454, NASA, 1975.

[35] Bryson, A. and Ho, Y., *Applied Optimal Control*, Taylor and Francis, 1975.

[36] Hull, D. G., *Optimal Control Theory for Applications*, Springer, New York, 2003.

[37] Enright, P. J. and Conway, B. A., "Optimal Finite-Thrust Spacecraft Trajectories Using Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 5, 1991, pp. 981 – 985,
doi:10.2514/3.20739.

[38] Enright, P. and Conway, B., "Discrete Approximations to Optimal Trajectories Using Direct Transcription and Nonlinear Programming," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 4, 1992, pp. 994–1002,
doi:10.2514/3.20934.

[39] Herman, A. and Conway, B., "Optimal Spacecraft Attitude Control Using Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 5, 1992, pp. 1287 – 1289,
doi:10.2514/3.20983.

[40] Herman, A. L. and Conway, B. A., "Direct Optimization Using Collocation Based on High-Order Gauss-Lobatto Quadrature Rules," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 3, 1996, pp. 592 – 599,
doi:10.2514/3.21662.

[41] Herman, A. L. and Conway, B. A., "Optimal, Low-Thrust, Earth-Moon Orbit Transfer," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 1, 1998, pp. 141 – 147,
doi:10.2514/2.4210.

[42] Tang, S. and Conway, B. A., "Optimization of Low-Thrust Interplanetary Trajectories Using Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 3, 1995, pp. 599 – 604,
doi:10.2514/3.21429.

[43] Whiffen, G. J., "Static/Dynamic Control for Optimizing a Useful Objective," United States Patent No. 6 496 741, 2002. Filed March 25, 1999.

[44] Lantoine, G. and Russell, R. P., "A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 1: Theory," *Journal of Optimization Theory and Applications*, Vol. 154, No. 2, 2012, pp. 382–417,
doi:10.1007/s10957-012-0039-0.

[45] Lantoine, G. and Russell, R. P., "A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 2: Application," *Journal of Optimization Theory and Applications*, Vol. 154, No. 2, 2012, pp. 418–442,
doi:10.1007/s10957-012-0038-1.

[46] Englander, J., Conway, B., and Williams, T., "Automated Mission Planning via Evolutionary Algorithms," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 6, 2012, pp. 1878–1887,
doi:10.2514/1.54101.

[47] Englander, J. A. and Conway, B. A., "An Automated Solution of the Low-Thrust Interplanetary Trajectory Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 1, 2017, pp. 15–27, doi:10.2514/1.G002124.

[48] Englander, J. A., *Automated Trajectory Planning for Multiple-Flyby Interplanetary Missions*, Ph.D. thesis, 2013.

[49] Sims, J., Finlayson, P., Rinderle, E., Vavrina, M., and Kowalkowski, T., "Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design," in "AIAA/AAS Astrodynamics Specialist Conference and Exhibit," AIAA Paper 2006-6746, 2006,
doi:10.2514/6.2006-6746.

[50] McConaghy, T. T., *GALLOP Version 4.5 User's Guide*, School of Aeronautics and Astronautics, Purdue University, 2005.

[51] Vavrina, M. and Howell, K., "Global Low Thrust Trajectory Optimization through Hybridization of a Genetic Algorithm and a Direct Method," in "AIAA/AAS Astrodynamics Specialist Conference and Exhibit, AIAA Paper 2008-6614," Honolulu, Hawaii, 2008,
doi:10.2514/6.2008-6614.

[52] Team, E. A. C., "PaGMO (Parallel Global Multiobjective Optimizer)," `https://github.com/esa/pagmo`, accessed 6/26/2016.

[53] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.

[54] *Spacecraft Trajectory Optimization*, Cambridge Aerospace Series, Cambridge University Press, 2010, doi:10.1017/CBO9780511778025.

[55] Knittel, J. M., Englander, J. A., Ozimek, M. T., Atchison, J. A., and Gould, J. J., "Improved Propulsion Modeling For Low-Thrust Trajectory Optimization," in "27th AAS/AIAA Space Flight Mechanics Meeting," San Antonio, Texas, 2017.

[56] Minovitch, M. A., "Tech. Rep. 32-464 The Determination and Characteristics of Ballistic Interplanetary Trajectories Under the Influence of Multiple Planetary Attractions," Tech. rep., Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 1963.

[57] Cummings, R., *Brigands of the Moon*, A. C. McClurg & Co., 1931.

[58] Clarke, A. C., *The Sands of Mars*, Sidgwick & Jackson, 1951.

[59] Lawden, D. F., "Perturbation Manoeuvers," *Journal of British Interplanetary Society*, Vol. 13, No. 5.

[60] Strange, N. J. and Longuski, J. M., "Graphical Method for Gravity-Assist Trajectory Design," *Journal of Spacecraft and Rockets*, Vol. 39, No. 1, 2002, pp. 9–16,
doi:10.2514/2.3800.

[61] Campagnola, S. and Russell, R. P., "The Endgame problem part 1: V-Infinity Leveraging Technique and the Leveraging Graph," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 463–475,
doi:10.2514/1.44258.

[62] Campagnola, S. and Russell, R. P., "Endgame Problem Part 2: Multibody Technique and the Tisserand-Poincaré Graph," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 2, 2010, pp. 476–486,
doi:10.2514/1.44290.

[63] Labunsky, A., Papkov, O., and Sukhanov, K., *Multiple Gravity Assist Interplanetary Trajectories*, Earth Space Institute Book Series, Taylor & Francis, 1998.

[64] Russell, R. P. and Ocampo, C. A., "Geometric Analysis of Free-Return Trajectories Following a Gravity-Assisted Flyby," *Journal of Spacecraft and Rockets*, Vol. 42, No. 1, 2005, pp. 138–152, doi:10.2514/1.5571.

[65] Woolley, R. C. and Scheeres, D. J., "Applications of V-Infinity Leveraging Maneuvers to Endgame Strategies for Planetary Moon Orbiters," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 5, 2011, pp. 1298–1310,
doi:10.2514/1.50428.

[66] Lantukh, D. V. and Russell, R. P., "Automated Inclusion of n-pi Transfers in Gravity-Assist Flyby Tour Design," in "AIAA/AAS Astrodynamics Specialist Conference," Minneapolis, MN, 2012, doi:10.2514/6.2012-4749.

[67] Boutonnet, A., Martens, W., and Schoenmaekers, J., "AAS 13-300 SOURCE : A MATLAB-Oriented Tool For Interplanetary Trajectory Global Optimization. Fundamentals," in "AAS/AIAA Space-Flight Mechanics Meeting," Kauai, HI, 2013.

[68] Williams, S. N., *Automated Design of Multiple Encounter Gravity-Assist Trajectories*, Master's thesis, Purdue University, 1990.

[69] Henning, G. A., Edelman, P. J., and Longuski, J. M., "Design and Optimization of Interplanetary Aerogravity-Assist Tours," *Journal of Spacecraft and Rockets*, Vol. 51, No. 6, 2014, pp. 1849–1856, doi:10.2514/1.A32881.

[70] Patel, M., *Automated Design of Delta-V Gravity-Assist Trajectories for Solar System Exploration*, Master's thesis, Purdue University, West Lafayette, Indiana, 1993.

[71] Petropoulos, A. E., Longuski, J. M., and Bonfiglio, E. P., "Trajectories to Jupiter via Gravity Assists from Venus, Earth, and Mars," *Journal of Spacecraft and Rockets*, Vol. 37, No. 6, 2000, pp. 776–783, doi:10.2514/2.3215.

[72] McConaghy, T. T., Debban, T. J., Petropoulos, A. E., and Longuski, J. M., "Design and Optimization of Low-Trust Trajectories with Gravity Assists," *Journal of Spacecraft and Rockets*, Vol. 40, No. 3, 2003, pp. 380–387,
doi:10.2514/2.3973.

[73] Russell, R. P., "Primer Vector Theory Applied to Global Low-Thrust Trade Studies," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 460–472,
doi:10.2514/1.22984.

[74] Izzo, D., Simões, L. F., Märtens, M., de Croon, G. C. H. E., Heritier, A., and Yam, C. H., "Search for a Grand Tour of the Jupiter Galilean Moons," in "Genetic and Evolutionary Computation Conference (GECCO 2013)," , 2013, pp. 1301–1308,
doi:10.1145/2463372.2463524.

[75] Izzo, D., Becerra, V. M., Myatt, D. R., Nasuto, S. J., and Bishop, J. M., "Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories," *Journal of Global Optimization*, Vol. 38, No. 2, 2007, pp. 283–296,
doi:10.1007/s10898-006-9106-0.

[76] Bernelli-Zazzera, F., Berz, M., Lavagna, M., Armellin, R., Di Lizia, P., and Topputo, F., "Global Trajectory Optimisation: Can We Prune the Solution Space When Considering Deep Space Maneuvers," Tech. Rep. 20271, 2007.

[77] Vinkó, T. and Izzo, D., "Global Optimisation Heuristics and Test Problems for Preliminary Spacecraft Trajectory Design," Tech. Rep. September, ESA ACT, 2008.

[78] Vasile, M., Minisci, E., and Locatelli, M., "Analysis of Some Global Optimization Algorithms for Space Trajectory Design," *Journal of Spacecraft and Rockets*, Vol. 47, No. 2, 2010, pp. 334–344, doi:10.2514/1.45742.

[79] Kennedy, J. and Eberhart, R., "Particle swarm optimization," in "IEEE International Conference on Neural Networks - Conference Proceedings," Vol. 4, 1995, pp. 1942–1948.

[80] Dorigo, M. and Stützle, T., *Ant Colony Optimization*, The MIT Press, Cambridge, MA, 2004.

[81] Ceriotti, M. and Vasile, M., "Automated Multigravity Assist Trajectory Planning with a Modified Ant Colony Algorithm," *Journal of Aerospace Computing, Information, and Communication*, Vol. 7, No. September, 2010, pp. 261–293,
doi:10.2514/1.48448.

[82] Ceriotti, M. and Vasile, M., "MGA trajectory planning with an ACO-inspired algorithm," *Acta Astronautica*, Vol. 67, No. 9-10, 2010, pp. 1202–1217,
doi:10.1016/j.actaastro.2010.07.001.

[83] Gage, P. J., Braun, R. D., and Kroo, I. M., "Interplanetary Trajectory Optimization Using a Genetic Algorithm," in "Astrodynamics Conference," Scottsdale, AZ, 1994,
doi:10.2514/6.1994-3773.

[84] Vasile, M., "AAS 03-141 A Global Approach to Optimal Space Trajectory Design," in "AAS/AIAA Space Flight Mechanics Meeting," Ponce, Puerto Rico, February, 2003, pp. 1–20.

[85] De Pascale, P. and Vasile, M., "Preliminary Design of Low-Thrust Multiple Gravity-Assist Trajectories," *Journal of Spacecraft and Rockets*, Vol. 43, No. 5, 2006, pp. 1065–1076,
doi:10.2514/1.19646.

[86] Storn, R. and Price, K., "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, Vol. 11, 1997, pp. 341–359,
doi:10.1023/A:100820282.

[87] Olds, A. D., Kluever, C. A., and Cupples, M. L., "Interplanetary Mission Design Using Differential Evolution," *Journal of Spacecraft and Rockets*, Vol. 44, No. 5, 2007, pp. 1060–1070,
doi:10.2514/1.27242.

[88] Izzo, D., Ruciński, M., and Ampatzis, C., "Parallel global optimisation meta-heuristics using an asynchronous island-model," *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pp. 2301–2308,
doi:10.1109/CEC.2009.4983227.

[89] Izzo, D., Ruciński, M., and Biscani, F., "The generalized Island model," *Studies in Computational Intelligence*, Vol. 415, 2012, pp. 151–169,
doi:10.1007/978-3-642-28789-3-7.

[90] Englander, J. A., Conway, B. A., and Williams, T., "Automated Mission Planning via Evolutionary Algorithms," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 6, 2012, pp. 1878–1887,
doi:10.2514/1.54101.

[91] Holland, J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[92] Rauwolf, G. A. and Coverstone-Carroll, V. L., "Near-Optimal Low-Thrust Orbit Transfers Generated by a Genetic Algorithm," *Journal of Spacecraft and Rockets*, Vol. 33, No. 6, 1996, pp. 859–862,
doi:10.2514/3.26850.

[93] Conway, B. A., Chilan, C. M., and Wall, B. J., "Evolutionary principles applied to mission planning problems," *Celestial Mechanics and Dynamical Astronomy*, Vol. 97, No. 2, 2007, pp. 73–86,
doi:10.1007/s10569-006-9052-7.

[94] Wall, B. J. and Conway, B. A., "Genetic algorithms applied to the solution of hybrid optimal control problems in astrodynamics," *Journal of Global Optimization*, Vol. 44, No. 4, 2009, pp. 493–508,
doi:10.1007/s10898-008-9352-4.

[95] Woo, B., Coverstone, V. L., and Cupples, M., "Low-Thrust Trajectory Optimization Procedure for Gravity-Assist, Outer-Planet Missions," *Journal of Spacecraft and Rockets*, Vol. 43, No. 1, 2006, pp. 121–129,
doi:10.2514/1.14665.

[96] Gad, A. and Abdelkhalik, O., "Hidden Genes Genetic Algorithm for Multi-Gravity-Assist Trajectories Optimization," *Journal of Spacecraft and Rockets*, Vol. 48, No. 4, 2011, pp. 629–641,
doi:10.2514/1.52642.

[97] Abdelkhalik, O. and Gad, A., "Dynamic-Size Multiple Populations Genetic Algorithm for Multigravity-Assist Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 2, 2012, pp. 520–529,
doi:10.2514/1.54330.

[98] Coverstone-Carroll, V., Hartmann, J. W., and Mason, W. J., "Optimal multi-objective low-thrust spacecraft trajectories," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, No. 2-4, 2000, pp. 387–402,
doi:10.1016/s0045-7825(99)00393-x.

[99] Srinivas, N. and Deb, K., "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, Vol. 2, No. 3, 1994, pp. 221–248,
doi:10.1162/evco.1994.2.3.221.

[100] Vavrina, M., Englander, J. A., and Ghosh, A. R. M., "Coupled Low-Thrust Trajectory and Systems Optimization via Multi-Objective Hybrid Optimal Control," in "AAS/AIAA Space Flight Mechanics Meeting, Williamsburg, VA," , 2015.

[101] Englander, J. A., Vavrina, M., and Ghosh, A. R. M., "Multi-Objective Hybrid Optimal Control for Multiple-Flyby Low-Thrust Mission Design," in "AAS/AIAA Space Flight Mechanics Meeting, Williamsburg, VA," , 2015.

[102] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182–197,
doi:10.1109/4235.996017.

[103] Corana, A., Marchesi, M., Martini, C., and Ridella, S., "Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm," *ACM Transactions on Mathematical Software*, Vol. 13, No. 3, 1987, pp. 262–280,
doi:10.1145/66888.356281.

[104] Glover, F., "Tabu Search - Part I," *ORSA Journal on Computing*, Vol. 1, No. 3, 1989, pp. 190–206,
doi:10.1287/ijoc.1.3.190.

[105] Glover, F., "Tabu Search - Part II," *ORSA Journal on Computing*, Vol. 2, No. 1, 1990, pp. 4–32,
doi:10.1287/ijoc.2.1.4.

[106] Wales, D. J. and Doye, J. P. K., "Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms," *Journal of Physical Chemistry A*, Vol. 101, No. 28, 1997, pp. 5111–5116,
doi:10.1021/jp970984n.

[107] Leary, R. H., "Global Optimization on Funneling Landscapes," *Journal of Global Optimization*, Vol. 18, No. 4, 2000, pp. 367–383,
doi:10.1023/A:1026500301312.

[108] Johnson, G. P., *A Tabu Search Methodology for Spacecraft Tour Trajectory Optimization*, Ph.D. thesis, University of Texas at Austin, 2014.

[109] Addis, B., Cassioli, A., Locatelli, M., and Schoen, F., "A global optimization method for the design of space trajectories," *Computational Optimization and Applications*, Vol. 48, No. 3, 2011, pp. 635–652, doi:10.1007/s10589-009-9261-6.

[110] Yam, C. H., Di Lorenzo, D., and Izzo, D., "Constrained global optimization of low-thrust interplanetary trajectories," in "2010 IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010," , 2010, doi:10.1109/CEC.2010.5586019.

[111] Yam, C. H., Lorenzo, D. D., and Izzo, D., "Low-thrust trajectory design as a constrained global optimization problem," in "Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering," Vol. 225, 2011, pp. 1243–1251, doi:10.1177/0954410011401686.

[112] Englander, J. A. and Englander, A. C., "Tuning Monotonic Basin Hopping: Improving the Efficiency of Stochastic Search as Applied to Low-Thrust Trajectory Optimization," in "International Symposium on Space Flight Dynamics 2014," Laurel, MD, 2014.

[113] Ellison, D. H., Conway, B. A., Englander, J. A., and Ozimek, M. T., "Application and Analysis of Bounded-Impulse Trajectory Models with Analytic Gradients," *Journal of Guidance, Control, and Dynamics*, doi:10.2514/1.G003078.

[114] Sussmann, H. J., "A maximum principle for hybrid optimal control problems," in "Conference on Decision & Control," IEEE, Phoenix, AZ, December, 1999, pp. 425–430, doi:10.1109/CDC.1999.832814.

[115] von Stryk, O. and Glocker, M., "Decomposition of Mixed-Integer Optimal Control Problems Using Branch and Bound and Sparse Direct Collocation," in "International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems," Dortmund, Vol. September, 2000, pp. 99–104, doi:10.1.1.37.9525.

[116] von Stryk, O. and Glocker, M., "Numerical Mixed-Integer Optimal Control and Motorized Traveling Salesmen Problems," *APII-JESA (Journal europen des systmes automatiss - European Journal of Control)*, Vol. 35, No. 4, 2001, pp. 519–533.

[117] Ross, I. M. and D'Souza, C., "Hybrid Optimal Control Framework for Mission Planning," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 4, 2005, pp. 686–697.

[118] Peressini, A. L., Sullivan, F. E., and Uhl Jr., J. J., *The Mathematics of Nonlinear Programming*, Undergraduate Texts in Mathematics, Springer, 1988.

[119] Chilan, C. M. and Conway, B. A., "Automated Design of Multiphase Space Missions Using Hybrid Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 5, 2013, pp. 1410–1424, doi:10.2514/1.58766.

[120] Stanbridge, D., Williams, K., Williams, B., Jackman, C., Weaver, H., Berry, K., Sutter, B., and Englander, J., "AAS 17-632 LUCY : Navigating a Jupiter Trojan Tour," in "AAS/AIAA Spaceflight Mechanics Meeting," Stevenson, WA, 2017.

[121] Longuski, J. M. and Williams, S. N., "Automated design of gravity-assist trajectories to Mars and the outer planets," *Celestial Mechanics and Dynamical Astronomy*, Vol. 52, No. 3, 1991, pp. 207–220, doi:10.1007/BF00048484.

[122] Byrnes, D. V. and Bright, L. E., "AAS 95-307 Design of High-Accuracy Multiple Flyby Trajectories Using Constrained Optimization," in "AAS/AIAA Astrodynamics Specialist Conference," Halifax, Nova Scotia, 1995.

[123] Hatfield, J. N., "CATO (Computer Algorithm for Trajectory Optimization) An Implementation of Fortran 95 Object-Based Programming," in "ACM Fortran Forum," Vol. 22, 2003, pp. 2–7.

[124] Lantukh, D. V., Russell, R. P., and Campagnola, S., "V-Infinity Leveraging Boundary-Value Problem and Application in Spacecraft Trajectory Design," *Journal of Spacecraft and Rockets*, Vol. 52, No. 3, 2015, pp. 697–710,
doi:10.2514/1.A32918.

[125] Lynam, A. E., "Broad-search algorithms for finding triple-and quadruple-satellite-aided captures at Jupiter from 2020 to 2080," *Celestial Mechanics and Dynamical Astronomy*, Vol. 121, No. 4, 2015, pp. 347–363,
doi:10.1007/s10569-015-9602-y.

[126] Lynam, A. E., "Broad search for trajectories from Earth to Callisto?Ganymede?JOI double-satellite-aided capture at Jupiter from 2020 to 2060," *Celestial Mechanics and Dynamical Astronomy*, Vol. 124, No. 1, 2016, pp. 33–50,
doi:10.1007/s10569-015-9649-9.

[127] Didion, A. M. and Lynam, A. E., "Impulsive Trajectories from Earth to Callisto?Io?Ganymede Triple Flyby Jovian Capture," *Journal of Spacecraft and Rockets*, Vol. 52, No. 3, 2015, pp. 746–753,
doi:10.2514/1.A33159.

[128] Scott, C. J., Ozimek, M. T., Haapala, A. F., Siddique, F. E., and Buffington, B. B., "Dual Satellite-Aided Planetary Capture with Interplanetary Trajectory Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 3, 2017, pp. 548–562,
doi:10.2514/1.G002102.

[129] Longo, A., *Analysis and optimization of improved trajectory for the JUICE mission*, Ph. d., Sapienza Università di Roma, 2014.

[130] Strange, N. J., Landau, D. F., and Longuski, J. M., "AAS 13-801 Design of Initial Inclination Reduction Sequence for Uranian Gravity-Assist Tours," in "AAS/AIAA Astrodynamics Specialist Conference," Hilton Head, SC, 2013.

[131] "NASAs Evolutionary Xenon Thruster (NEXT) Ion Propulsion GFE Component Information Summary for Discovery Missions July 2014," , 2014. `http://discovery.larc.nasa.gov/discovery/pdf_files/20-NEXT-C_AO_Guidebook_11July14.pdf`.

[132] Wächter, A. and Biegler, L. T., "On the Implementation of a Primal-Dual Interior Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, Vol. 106, No. 1, 2006, pp. 25–57,
doi:10.1007/s10107-004-0559-y.

[133] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Rev.*, Vol. 47, No. 1, 2005, pp. 99–131,
doi:10.1137/S0036144504446096.

[134] Karush, W., *Minima of Functions of Several Variables with Inequalities as Side Constraints*, Master's thesis, University of Chicago, 1939.

[135] Kuhn, H. W. and Tucker, A. W., "Nonlinear Programming," in "Proceedings of the 2nd Berkeley Symposium," University of California Press, Berkeley, California, 1951, pp. 481–492.

[136] Pellegrini, E. and Russell, R. P., "On the Computation and Accuracy of Trajectory State Transition Matrices," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 11, 2016, pp. 2485–2499,
doi:10.2514/1.G001920.

[137] Lyness, J. N. and Moler, C. B., "Numerical Differentiation of Analytic Functions," *SIAM Journal on Numerical Analysis*, Vol. 4, No. 2, 1967, pp. 202–210,
doi:10.1137/0704019.

[138] Lyness, J. N., "Differentiation Formulas for Analytic Functions," *Mathematics of Computation*, Vol. 22, No. 102, 1968, pp. 352–362,
doi:10.2307/2004665.

[139] Lantoine, G., Russell, R. P., and Dargent, T., "Using Multicomplex Variables for Automatic Computation of High-Order Derivatives," *ACM Transactions on Mathematical Software*, Vol. 38, No. 3, 2012, p. 21 pages,
doi:10.1145/2168773.2168774.

[140] Wengert, R. E., "A Simple Automatic Derivative Evaluation Program," *Commun. ACM*, Vol. 7, No. 8, 1964, pp. 463–464,
doi:10.1145/355586.364791.

[141] Linnainmaa, S., *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*, Master's thesis, University of Helsinki, 1970.

[142] Linnainmaa, S., "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, Vol. 16, No. 2, 1976, pp. 146–160,
doi:10.1007/BF01931367.

[143] Ellison, D. H., Conway, B. A., Englander, J. A., and Ozimek, M. T., "Analytic Gradient Computation for Bounded-Impulse Trajectory Models Using Two-Sided Shooting," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 7, 2018, pp. 1449–1462,
doi:10.2514/1.G003077.

[144] McConaghy, T. T., *Design and optimization of interplanetary spacecraft trajectories*, PhD dissertation, School of Aeronautics and Astronautics, Purdue University, 2004.

[145] Bracewell, R., ed., *The Fourier Transform & Its Applications, 3rd ed.*, McGraw-Hill Science/Engineering/Math, 2000.

[146] "SPICE Ephemeris," , 2013. `http://naif.jpl.nasa.gov/naif/`.

[147] Arora, N. and Russell, R. P., "A fast, accurate, and smooth planetary ephemeris retrieval system," *Celestial Mechanics and Dynamical Astronomy*, Vol. 108, No. 2, 2010, pp. 107–124,
doi:10.1007/s10569-010-9296-0.

[148] Galassi, M., *GNU Scientific Library Reference Manual (3rd edition).*

[149] Prussing, J. E. and Chiu, J. H., "Optimal Multiple-Impulse Time-Fixed Rendezvous Between Circular Orbits," *Journal of Guidance, Control, and Dynamics*, Vol. 9, No. 1, 1986, pp. 17–22,
doi:10.2514/3.20060.

[150] Zimmer, S. and Ocampo, C., "Use of Analytical Gradients to Calculate Optimal Gravity-Assist Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 2, 2005, pp. 324–332,
doi:10.2514/1.4825.

[151] Zimmer, S. and Ocampo, C., "Analytical Gradients for Gravity Assist Trajectories Using Constant Specific Impulse Engines," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 4, 2005, pp. 753–760,
doi:10.2514/1.9917.

[152] Vavrina, M., Englander, J. A., and Ellison, D. H., "Global Optimization of N-Maneuver, High-Thrust Trajectories Using Direct Multiple Shooting," in "AIAA/AAS Space Flight Mechanics Meeting, AAS Paper 16-272, Napa Valley, CA," , 2016.

[153] Leary, R., "Global Optimization on Funneling Landscapes," *Journal of Global Optimization*, Vol. 18, No. 4, 2000, pp. 367–383,
doi:10.1023/A:1026500301312.

[154] Vasile, M., Minisci, E., and Locatelli, M., “Analysis of Some Global Optimization Algorithms for Space Trajectory Design,” *Journal of Spacecraft and Rockets*, Vol. 47, No. 2, 2010, pp. 334–344, doi:10.2514/1.45742.

[155] Englander, J. A. and Englander, A. C., “Tuning Monotonic Basin Hopping: Improving the Efficiency of Stochastic Search as Applied to Low-Thrust Trajectory Optimization,” in “24th International Symposium on Space Flight Dynamics, Laurel, MD,” , 2014.

[156] Ghosh, A. and Coverstone, V., “Application of parallel algorithmic differentiation to optimal CubeSat trajectory design,” *Acta Astronautica*, Vol. 130, No. April 2016, 2017, pp. 1–14, doi:10.1016/j.actaastro.2016.08.018.

[157] Prussing, J. and Conway, B., *Orbital Mechanics, Second Edition*, Oxford University Press, New York, 2012.

[158] Yam, C. H., Biscani, F., and Izzo, D., “Global Optimization of Low-Thrust Trajectories via Impulsive Delta-V Transcription,” in “27th International Symposium on Space Technology and Science,” Tsukuba, Japan, 2009.

[159] Englander, J. A., Ellison, D. H., and Conway, B. A., “Global Optimization of Low-Thrust, Multiple-Flyby Trajectories at Medium and Medium-High Fidelity,” in “AAS/AIAA Space Flight Mechanics Meeting, Santa Fe, NM,” , 2014.

[160] Battin, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition*, American Institute of Aeronautics and Astronautics Inc., Reston, Virginia, 1999.

[161] Der, G. J., “An Elegant State Transition Matrix,” in “AAS/AIAA Astrodynamics Conference, San Diego, Ca,” , 1996.

[162] Conway, B. A., “An Improved Method due to Laguerre for the Solution of Kepler’s Equation,” *Celestial Mechanics*, Vol. 39, No. 2, 1986, pp. 199–211.

[163] Shepperd, S. W., “Universal Keplerian State Transition Matrix,” *Celestial Mechanics*, Vol. 35, No. 2, 1985, pp. 129–144,
doi:10.1007/BF01227666.

[164] Goodyear, W. H., “A General Method for the Computation of Cartesian Coordinates and Partial Derivatives of the Two-Body Problem,” Tech. Rep. cr-522, NASA, 1966.

[165] Sperling, H., “Computation of Keplerian Conic Sections,” *American Rocketry Society*, Vol. 31, 1961, pp. 660–661.

[166] Herrick, S. H., “Universal Variables,” *Astronomical Journal*, Vol. 70, 1965, pp. 309–315, doi:10.1086/109728.

[167] Lemmon, W. W. and Brooks, J. E., “A Universal Formulation for Conic Trajectories-Basic Variables and Relationships,” Tech. Rep. 3400-601 9-tu000, TRW/Systems, Redondo Beach, CA, 1965.

[168] Pitkin, E. T., “Second transition partial derivatives via universal variables,” *Journal of Astronautical Sciences*, Vol. 13, 1966, p. 204.

[169] Lantoine, G. and Russell, R. P., “A Fast Second-Order Algorithm for Preliminary Design of Low-Thrust Trajectories,” in “59^{th} International Astronautical Congress, Paper IAC-08-C1.2.5, Glasgow, Scotland,” , 2008.

[170] Ghosh, A. R. M. and Coverstone, V., “Optimal cooperative CubeSat maneuvers obtained through parallel computing,” *Acta Astronautica*, Vol. 107, 2015, pp. 130–149, doi:10.1016/j.actaastro.2014.10.042.

[171] Hubbard, W. B., “Ice Giants Decadal Study,” Tech. rep., NASA Planetary Sciences Decadal Survey, 2010. `http://sites.nationalacademies.org/SSB/SSB_059331`.

[172] Council, N. R., *Vision and Voyages for Planetary Science in the Decade 2013-2022*, The National Academies Press, Washington, DC, 2011,
doi:10.17226/13117.

[173] Sauer Jr., C. G., "AAS 97-726 Solar Electric Performance for Medlite and Delta Class Planetary Missions," in "AAS/AIAA Astrodynamics Specialist Conference," Sun Valley, Id, 1997.

[174] Debban, T. J., McConaghy, T. T., and Longuski, J. M., "AIAA 2002-4729 Design and Optimization of Low-Thrust Gravity-Assist Trajectories to Selected Planets," in "AIAA/AAS Astrodynamics Specialist Conference and Exhibit," Monterey, CA, August, 2002, pp. 1–10,
doi:10.2514/6.2002-4729.

[175] Kluever, C. A., "Comet Rendezvous Mission Design Using Solar Electric Propulsion Spacecraft," *Journal of Spacecraft and Rockets*, Vol. 37, No. 5, 2000, pp. 698–700,
doi:10.2514/2.3621.

[176] Woo, B., Coverstone, V. L., and Cupples, M., "Application of Solar Electric Propulsion to a Comet Surface Sample Return Mission," *Journal of Spacecraft and Rockets*, Vol. 43, No. 6, 2006, pp. 1225–1230,
doi:10.2514/1.23371.

[177] Veverka, J., Johnson, L., and Edward, R., "Comet Surface Sample Return (CSSR) Mission," Tech. rep., NASA Planetary Sciences Decadal Survey, 2008. `http://sites.nationalacademies.org/cs/groups/ssbsite/documents/webpage/ssb_059325.pdf`.

[178] Choukroun, M. and Wadhwa, M., "COmet Nucleus Dust and Organics Return (CONDOR): a New Frontiers 4 Mission Proposal," in "European Planetary Science Congress," Riga, Latvia, Vol. 11, 2017.

[179] "NASA Launch Services Program Launch Vehicle Performance Website," `http://elvperf.ksc.nasa.gov/elvMap/`, 2011.

[180] McKinnon, D. V., Weigl, H., and Maack, L., "AIAA 2016-5724 Lockheed Martin's A2100 Spacecraft Bus Modernization," in "34th AIAA International Communications Satelllite Systems Conference," Cleveland, OH, October, 2016,
doi:10.2514/6.2016-5724.

[181] Hofer, R., "High-Specific Impulse Operation of the BPT-4000 Hall Thruster for NASA Science Missions," in "46th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Nashville, TN," , 2010.

[182] Roa, J., *Regularization in Orbital Mechanics: Theory and Practice*, De Gruyter Studies in Mathematical Physics, De Gruyter, 2017.

[183] Levi-Civita, T., "Sur La Résolution Qualitative Du Problème Restreint Des Trois Corps," *Acta Mathematica*, Vol. 30, 1906, pp. 305–327.

[184] Williams, T. R., *Crommelin, Andrew Claude de la Cherois*, Springer New York, New York, NY, pp. 262–262, 2007,
doi:10.1007/978-0-387-30400-7_316.

[185] Pellegrini, E., Russell, R. P., and Vittaldev, V., "F and G Taylor Series Solutions to the Stark and Kepler Problems with Sundman Transformations," *Celestial Mechanics and Dynamical Astronomy*, Vol. 118, No. 4, 2014, pp. 355–378,
doi:10.1007/s10569-014-9538-7.

[186] Nacozy, P., "The Intermediate Anomaly," *Celestial Mechanics*, Vol. 16, No. 3, 1977, pp. 309–313,
doi:10.1007/BF01232657.

[187] Berry, M. and Healy, L., "The generalized Sundman transformation for propagation of high-eccentricity elliptical orbits," in "Proceedings of the AAS/AIAA Space Flight Mechanics Meeting, San Antonio, Texas," AAS-02-109, 2002.

[188] Ghosh, A. M., *Multi-Cubesat Mission Planning Enabled Through Parallel Computing*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 2013.

[189] Ellison, D. H., Conway, B. A., and Englander, J. A., "Numerical Computation of a Continuous-Thrust State Transition Matrix Incorporating Accurate Hardware and Ephemeris Models," in "AAS/AIAA Space-Flight Mechanics Meeting, Williamsburg, VA," , 2015.

[190] Hughes, S. P., Cooley, D. S., and Guzman, J. J., "A Direct Method for Fuel Optimal Maneuvers of Distributed Spacecraft in Multiple Flight Regimes," in "Proceedings of the AAS/AIAA Space Flight Mechanics Meeting, Copper Mountain, Colorado," AAS-05-158, 2005.

[191] Hughes, S. P. and Dove, E., "Optimal Control and Near Optimal Guidance for the Magnetospheric MultiScale Mission (MMS)," in "Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Pittsburgh, Pennsylvania," AAS-09-330, 2009.

[192] Ocampo, C. and Munoz, J.-P., "Variational Equations for a Generalized Spacecraft Trajectory Model," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 5, 2010, pp. 1615–1622.

[193] Ocampo, C. and Mathur, R., "Variational Model for Optimization of Finite-Burn Escape Trajectories Using a Direct Method," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 2, 2012, pp. 598–608.

[194] Mitchell, E. D., *Guidance of Low-Thrust Interplanetary Vehicles*, Ph.D. thesis, Massachusetts Institute of Technology, 1964.

[195] Englander, J. A., Knittel, J. M., Williams, K., Stanbridge, D., and Ellison, D. H., "AAS 17-204 Validation of a Low-Thrust Mission Design Tool Using Operational Navigation Software," in "AAS/AIAA Space Flight Mechanics Meeting," San Antonio, TX, Vol. 160, 2017.

[196] Breckheimer, P. J., "DPTRAJ/ODP - DOUBLE PRECISION TRAJECTORY ANALYSIS AND ORBIT DETERMINATION PROGRAM," Tech. rep., NASA, 1994.

[197] "Announcement of Opportunity: New Frontiers 4, NNH16ZDA011O," `newfrontiers.larc.nasa.gov`, 2016.

[198] "NASA's Evolutionary Xenon Thruster (NEXT): Ion Propulsion GFE Component Information Summary for Discovery Missions, July 2014," `http://discovery.larc.nasa.gov/discovery/pdf_files/20-NEXT-C_AO_Guidebook_11July14.pdf`, 2014.

[199] Englander, J. A., Knittel, J. M., Williams, K., Stanbridge, D., and Ellison, D. H., "Validation of a Low-Thrust Mission Design Tool Using Operational Navigation Software," in "27^{th} AAS/AIAA Space Flight Mechanics Meeting," San Antonio, Texas, 2017.

[200] Lynam, A. E., Kloster, K. W., and Longuski, J. M., "Multiple-satellite-aided capture trajectories at Jupiter using the Laplace resonance," *Celestial Mechanics and Dynamical Astronomy*, Vol. 109, No. 1, 2011, pp. 59–84,
doi:10.1007/s10569-010-9307-1.

[201] Arora, N. and Russell, R. P., "AAS 13-728 A Fast and Robust Multiple Revolution Lambert Algorithm Using a Cosine Transformation," in "AAS/AIAA Astrodynamics Specialist Conference," Hilton Head, SC, 2013.

[202] Petropoulos, A., "GTOC 6 Problem Statement," , 2012. `https://sophia.estec.esa.int/gtoc_portal/wp-content/uploads/2012/11/gtoc6_problem_stmt-2.pdf`.

[203] Hennes, D. and Izzo, D., "Interplanetary Trajectory Planning with Monte Carlo Tree Search," *IJCAI International Joint Conference on Artificial Intelligence*, Vol. 2015-Janua, No. Ijcai, 2015, pp. 769–775.

[204] Sarli, B. V., Knittel, J. M., Englander, J. A., and Barbee, B. W., "Mission Design and Optimal Asteroid Deflection for Planetary Defense," in "68th International Astronautical Congress," Adelaide, Australia, 2018.

[205] McAdams, J., Scott, C., Guo, Y., Dankanich, J., and Russell, R., "AAS 11-188 Conceptual Mission Design of a Polar Uranus Orbiter and Satellite Tour," *Advances in the Astronautical Sciences*, Vol. 140, 2011, pp. 1257–1270.

[206] Landau, D., "Efficient Maneuver Placement for Automated Trajectory Design," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 7, 2018, pp. 1531–1541, doi:10.2514/1.G003172.

[207] Ellison, D. H., Englander, J. A., and Conway, B. A., "Robust Global Optimzation of Low-Thrust, Multiple-Flyby Trajectories," in "AAS/AIAA Astrodynamics Specialist Conference, Hilton Head, SC," , 2013.

[208] Cassioli, A., Izzo, D., Di Lorenzo, D., Locatelli, M., and Schoen, F., "Global Optimization Approaches for Optimal Trajectory Planning," in "Springer Optimization and Its Applications," Springer Science + Business Media New York, Vol. 73, chap. Chapter 5, pp. 111–140, 2012, doi:10.1007/978-1-4614-4469-5_5.

[209] Oh, D. and Goebel, D., "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, AIAA Paper 2006-4466, Sacramento, CA," , 2006, doi:10.2514/6.2006-4466.