APPLICATION OF DIRECT METHOD TRANSCRIPTION FOR A HUMAN-CLASS TRANSLUNAR INJECTION TRAJECTORY OPTIMIZATION

by

KEVIN ERIC WITZBERGER

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Aerospace Engineering
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2011

Abstract

This thesis presents a new trajectory optimization software package developed in the framework of a low-to-high fidelity three degree-of-freedom (3-DOF)/6-DOF vehicle simulation program named Mission Analysis Simulation Tool in Fortran (MASTIF) and its application to a translunar trajectory optimization problem. The functionality of the developed optimization package is implemented as a new "mode" in generalized settings to make it applicable for a general trajectory optimization problem. In doing so, a direct optimization method using collocation is employed for solving the problem. Trajectory optimization problems in MASTIF are transcribed to a constrained nonlinear programming (NLP) problem and solved with SNOPT, a commercially available NLP solver. A detailed description of the optimization software developed is provided as well as the transcription specifics for the translunar injection (TLI) problem.

This assessment of the final results is formulated via a metric given as the minimization of the TLI main engine burn time, which is equivalent to the maximization of the mass at main engine cutoff (MECO). Key design parameters include the initial values for three orbital angles (right ascension of ascending node, argument of perigee, and true anomaly) and three Euler angles for steering during the main engine burn. To do so, the solution starts by modeling the entire trajectory into three distinct phases. The first two phases are based on a collocation method whereas the third phase appears with a high order Runge-Kutta integration.

The next part of assessing the TLI trajectory utilizes MASTIF's vehicle simulation capabilities (the other “mode” within MASTIF). This includes the ability to design and test new and existing guidance, navigation, and control (GN&C) algorithms. As a demonstration of MASTIF’s versatility, results from the trajectory optimization (the open-loop solution) in the form of a set of initial states and specific orbital target parameters at MECO are used in a new preliminary assessment of a variant of the Space Shuttle’s flight-proven closed-loop guidance algorithm named Powered Explicit Guidance (PEG). Main engine burn times and the LVLH Euler angles from the open-loop and closed-loop solutions are compared to show approximate agreement and efficacy of MASTIF’s two distinct “modes”.

Lists of Abbreviations, Acronyms and Symbols

Abbreviations

eng	engine	31
$nNodes$	global number of nodes	36
$nNodes_{odd}$	global number of odd nodes	36
$nNodes_{ph}$	number of nodes per phase	35
$nNodes_{ph,odd}$	number of odd nodes per phase	35
$nNodes_{ph,even}$	number of even nodes per phase	35
$nNodes_{seg}$	number of nodes per segment	35
$nNodes_{veh}$	number of nodes per vehicle	36
nPh	number of phases	36
$nSeg_{ph}$	number of segments per phase	35
$nTnks$	number of fuel tanks	22
$nVeh$	number of vehicles	36
ph	phase	12
seg	segment	33
veh	vehicle	12
wrt	with respect to	28

Acronyms

BEO	Beyond Earth Orbit	1
CAIB	Columbia Accident Investigation Report	1
CoV	Calculus of Variations	26
EME2000	Earth Mean Equator and Equinox of Epoch J2000	18
ETS	Earth Transfer Stage	3
FPR	Flight Performance Reserve	10
GEO	Geosynchronous Earth Orbit	7
GN&C	Guidance, Navigation, and Control	5
HSF	Human Space Flight	1
IGM	Iterative Guidance Mode	7
IAU	International Astronomical Union	19
JSC	Johnson Space Center	7
KKT	Karush-Kuhn-Tucker	68
LEO	Low Earth Orbit	3
LGL	Legendre-Gauss-Lobatto	32
LH2	Liquid Hydrogen	11
LOX	Liquid Oxygen	11
LVLH	Local-Verical Local-Horizontal	21
MCC	Midcourse correction	6
MCME	Moon-Centered, Moon Mean Equator and IAU-Node of Epoch	19
MECO	Main Engine Cutoff	7
NASA	National Aeronautics and Space Administration	1
NLP	Nonlinear Programming	5
PEG	Powered Explicit Guidance	6
RCS	Reaction Control System	12

S-IVB	Saturn-IVB	6
SQP	Sequential Quadratic Programming	39
6-DOF	6 degrees-of-freedom	70
3-DOF	3 degrees-of-freedom	2
TLI	Translunar Injection	1
TPBVP	Two Point Boundary Value Problem	69
TT	Terrestial Time	18
VSE	Vision For Exploration	1

Software Programs

MALTO	Mission Analysis Low-Thrust Optimization	4
MASTIF	Mission Analysis Simulation Tool in Fortran	5
MAVERIC	Marshall Aerospace Vehicle Representation in C	5
MIDAS	Multidisciplinary Integrated Design Assistant for Spacecraft	4
OTIS	Optimal Trajectories by Implicit Simulation	5
POST	Program to Optimize Simulated Trajectories	4
SEPTOP	Solar Electric Propulsion Trajectory Optimization Program	4

Symbols[1]

a	Earth-centered semi-major axis	17
$a_{wrt☾}$	Moon-centered semi-major axis	28
alt_p	pericynthion altitude	28
$\mathbf{c}$	path constraints vector	25
$\boldsymbol{\chi}$	initial constraints vector	25
$\tilde{\boldsymbol{\delta}}_{☾}$	Moon's declination and rate of change of declination at J2000	75
$\delta_{☾}$	instantaneous declination of Moon	75
$\boldsymbol{\Delta}$	defect vector	33
$\boldsymbol{\Delta t}$	phase length time vector	31
Δt_k	k^{th} phase length time	35
Δt_{k-1}	k^{th}-1 phase length time	36
ΔV_{actual}	magnitude of actual velocity change	54
ΔV_{gLoss}	magnitude of velocity change due to gravity losses	54
ΔV_{ideal}	magnitude of ideal velocity change	54
e	Earth-centered eccentricity	17
$e_{wrt☾}$	Moon-centered eccentricity	28
$\boldsymbol{\eta}$	phase timing linear constraint vector	36
$\mathbf{f}$	vector containing all (scaled) linear and nonlinear constraints	26
$f(*)$	function of *	25
$\mathbf{F_{j2k}}$	external force vector in EME2000 coordinates	31
$\mathbf{g}$	vector containing all non-zero partial derivatives of $\mathbf{f}$	26
γ	flight path angle	57
$gm_{☾}$	gravitational constant of Moon	28
$\mathbf{g_{j2k}}$	gravitational acceleration in EME2000 coordinates	31
$\mathbf{g_{j2k,\oplus}}$	gravitational acceleration due to Earth in EME2000 coordinates	22

[1]NOTE: scalars are regular characters whereas vectors/arrays are **bold**

$\mathbf{g_{j2k,\leftmoon}}$	gravitational acceleration due to Moon in EME2000 coordinates	22
$\mathbf{g_{j2k,\odot}}$	gravitational acceleration due to Sun in EME2000 coordinates	22
$g_{\oplus SL}$	magnitude of Earth's gravity vector at sea level	31
$H_{wrt\leftmoon}$	Moon-centered hyperbolic anomaly	28
I_{sp}	specific impulse	31
inc	Earth-centered inclination	17
$inc_{wrt\leftmoon}$	Moon-centered inclination	28
J	cost/objective function	25
J^*	optimal objective function	53
J_2	1^{st} order harmonics	11
$\mathbf{lc}$	linear constraint vector	37
$\mathbf{lwr}$	lower bounds vector	26
m	vehicle mass	31
$[\mathbf{M}_{j2ktoM}]$	rotation matrix from EME2000 to MCME frame	76
MR	mixture ratio	31
m_0	initial mass	10
m_{ETS}	ETS dry mass	10
m_{FPR}	FPR mass	10
m_{LH2}	LH2 mass	10
m_{LOX}	LOX mass	10
m_{RCS}	RCS mass	10
$\mathbf{m_{tnk}}$	fuel tank mass vector	32
$\dot{m}_{eng}$	engine mass flow rate	31
$\dot{m}_{FPR}$	FPR fuel tank mass flow rate	24
$\dot{m}_{LH2}$	LH2 fuel tank mass flow rate	23
$\dot{m}_{LOX}$	LOX fuel tank mass flow rate	23
$\dot{m}_{RCS}$	RCS fuel tank mass flow rate	24
$\dot{\mathbf{m}}_{\mathbf{tnk}}$	fuel tank mass flow rate vector	31

$\hat{\boldsymbol{\mu}}_{j2k}$	unit thrust vector in EME2000 coordinates	22
$\mathbf{nlc}$	nonlinear constraint vector	37
ν	true anomaly	18
Ω	right ascension (RA) of ascending node	17
$\tilde{\boldsymbol{\Omega}}_{☾}$	Moon's RA and rate of change of RA at J2000	75
$\Omega_{☾}$	instantaneous right ascension of Moon	75
ω	argument of perigee	17
ϕ_{lvlh}	LVLH roll angle	27
$\boldsymbol{\Psi}$	final constraints vector	25
ψ_{lvlh}	LVLH yaw angle	27
q	quaternion	73
q_{j2ktoM}	quaternion rotation from EME2000 to MCME frame	76
q_{lvlh2b}	quaternion rotation from LVLH to body frame	77
q_x	quaternion rotation about the x-axis	73
q_y	quaternion rotation about the y-axis	73
q_z	quaternion rotation about the z-axis	73
R	magnitude of position vector	57
$\mathbf{r_{j2k}}$	position vector in EME2000 coordinates	32
$\dot{\mathbf{r}}_{\mathbf{j2k}}$	time derivative of position vector in EME2000 coordinates	30
$\dot{r}_{wrt☾}$	Moon-centered magnitude of radius vector rate of change	28
$\mathbf{s}$	state vector	32
$\mathbf{s}_i$	state vector at i^{st} global node	35
$\mathbf{s}_j$	state vector at j^{th} segment node	33
$\tilde{\mathbf{s}}_j$	interpolated state vector at j^{th} segment node	33
$\dot{\mathbf{s}}_j$	state derivate vector at j^{th} segment node	33
$\mathbf{s}_{k-1}^{-}$	state vector at last node of $k^{th}-1$ phase	37
$\mathbf{s}_k^{+}$	state vector at first node of k^{th} phase	37
$\mathbf{s}_z$	state vector at z^{th} phase node	34

$\dot{\mathbf{s}}$	derivative of state vector	31
$\dot{\tilde{\mathbf{s}}}$	interpolated state derivative vector	33
$\dot{\mathbf{s}}_z$	state derivate vector at z^{th} phase node	34
t	time	25
τ	non-dimensional phase time	34
τ_{seg}	non-dimensional segment time	33
θ_{lvlh}	LVLH pitch angle	27
Θ	generic rotation angle	73
$t_{b_{eng}}$	engine burn time	27
T_{corr}	corrected thrust	31
T_{JC}	time past J2000 in Julian centuries	75
$\mathbf{t_0}$	phase start time vector	31
t_{0_k}	k^{th} phase start time	35
$t_{0_{k-1}}$	$k^{th} - 1$ phase start time	36
T_{vac}	vacuum thrust	10
$\mathbf{u}$	control vector	31
$\mathbf{u}^*$	optimal control vector	43
$\mathbf{upr}$	upper bounds vector	26
$\mathbf{u}_i$	control vector at i^{th} global node	35
$\mathbf{u}_j$	control vector at j^{th} segment node	33
$\mathbf{u}_{k-1}^-$	control vector at last node of $k^{th} - 1$ phase	37
$\mathbf{u}_k^+$	control vector at first node of k^{th} phase	37
$\mathbf{u}_z$	control vector at z^{th} phase node	34
V	magnitude of velocity vector	57
$\mathbf{v_{j2k}}$	velocity vector in EME2000 coordinates	32
$\dot{\mathbf{v}}_{\mathbf{j2k}}$	time derivative of velocity vector in EME2000 coordinates	31
$\mathbf{v}_q$	vector part of quaternion	73
w_q	scalar part of quaternion	73

$\mathbf{x}$	vector containing all (scaled) independent variables	26
$\mathbf{x}_{mastif}$	vector of variables for internal MASTIF usage	39
$\dot{\mathbf{x}}$	state vector derivatives	25
$\boldsymbol{\zeta_s}$	linear constraints for states vector	37
$\boldsymbol{\zeta_u}$	linear constraints for controls vector	37
$^\circ$	degree	17
$\cdot$	vector dot product	73
$\times$	vector cross product	73
$\otimes$	quaternion multiplication	73
$(\dot{\ })$	time derivative	25

Subscripts and Supersubscripts

f, f	final	25
i	global node number	34
j	segment node number	33
k	phase number	32
n	integer counter	36
z	phase node number	34
0	initial	25

Acknowledgments

This thesis is the culmination of months of research and completing a thesis has been a goal of mine for nearly a decade. I would like to acknowledge Dr. Zeiler for taking on the role of my thesis advisor in non-optimal circumstances. His timely reviews, feedback, and comments were of high quality and greatly appreciated. I would like to also acknowledge Dr. Freeman and Dr. Jones for their important roles. Also, Dr. Sharma who worked with me to help this topic come into focus well over a year ago. Feedback provided pushed me to dig deeper into the literature which enriched my learning experience. There are others that I would like to acknowledge as well. Leslie Balkanyi reviewed this document and provided feedback. He has always been available for technical discussions over the last several years and those micro learning experiences helped tremendously. Waldy Sjauw – who laughed out loud when I told him that I would deal with the issue of scaling variables "later". Needless to say, we both laughed at this months later when I learned the hard way that scaling is a subproblem to the nonlinear programming optimization problem and cannot be deferred. Waldy also offered valuable technical insight. Ian Dux provided me with input data and suggested practical constraints. Additionally, he gave me a quick tutorial on using a visualization tool. Finally, I would like to acknowledge my wife and kids. I was immersed in this research for months and there were too many days in which they did not have my undivided attention.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Ever since the National Aeronautics and Space Administration's (NASA's) Apollo program ended in the early 1970's, humans have not traveled beyond Earth Orbit (BEO). Various issues have hampered BEO travel for humans including the fact that human space flight (HSF) is costly and requires continued commitment from each new administration for decades to come. HSF is also risky, as evidenced in 2003 with the Space Shuttle's Columbia disaster. In response to the Columbia Accident Investigation Report (CAIB) [1], in January 2004, President George W. Bush announced a new United States space policy–Vision For Space Exploration (VSE) [2]. The VSE established the goal of returning humans to the Moon by 2020 as preparation for a future human Mars mission. As a result of the VSE, NASA established the Constellation Program[1]. In order to return humans to the Moon, a good deal of research and trade studies were conducted within the Constellation program. In particular, and the focus of this thesis, is the transfer trajectory from Earth to the Moon, i.e. the translunar injection (TLI) trajectory. This TLI problem is a trajectory optimization problem; deliver the most mass to the Moon while meeting mission constraints. This is not a new trajectory optimization problem–it was obviously solved during the Apollo program–using an indirect optimization method.

[1]The Constellation program effectively ended during the time of this thesis writing; however, a human Lunar mission remains an option

The research described in this thesis also leads to a solution of the TLI problem; however, a direct optimization method is employed to solve the problem, transcribing it to a nonlinear programming (NLP) problem that is solved with a commercial solver, SNOPT [3]. The robustness and flexibility offered by the direct method serve as the primary motivations; it is amenable to different types of trajectory optimizations and mission constraint requirements. Furthermore, this direct method is implemented using collocation techniques [4, 5] in a generic fashion in a three degree-of-freedom (3-DOF)/6-DOF vehicle simulation program. Direct optimization methods using collocation are effective at solving trajectory optimization problems [4, 5].

Typically, vehicle simulation and trajectory optimization (hereafter referred to as just optimization) programs are separate programs. Combining optimization and 3-DOF/6-DOF vehicle simulation software into the same tool, makes MASTIF's capabilities rare. This combination addresses issues that arise from having separate programs and offers benefits as well. Frequently, the outputs of the optimization program are inputs into the vehicle simulation program. From this researcher's experience the required modeling inputs (initial conditions, propulsion, aerodynamics, etc.) are rarely exactly the same between different programs. Also, the best an analyst can do is to ensure that the model between different tools is equivalent. This takes time and is a source of potential errors. Trajectories are sometimes optimized in a piece-wise fashion because establishing a first guess for a complete trajectory can be difficult. This issue can sometimes be remedied when the the same tool has dual capabilities such as simulation and optimization (especially when closed-loop guidance is available for use in the software for vehicle simulations). For example, the entire trajectory can be simulated to achieve an initial trajectory guess (as input) for the (complete) trajectory optimization and not just a certain portion of the trajectory.

There are two other benefits of this vehicle simulation/trajectory optimization software combination. The first concerns DOF. Generally, trajectory optimization tools are limited to 3-DOF modeling of the vehicle. Extending them to 6-DOF is not readily attainable because it would require significant software development or redevelopment. However, when implemented in the same tool as a 6-DOF vehicle simulation, extending the trajectory optimization software to 6-DOF is expected to be readily attainable because the 6-DOF infrastructure (e.g. equations of motion) already exists. Lastly, when the vehicle simulation/trajectory optimization software is combined in a single software package, code maintenance is only required on one piece of software instead of two. The list below summarizes the motivations for combining vehicle simulation and trajectory optimization software.

- reduce modeling errors
- save time
- easier/better initial guesses
- readily extendable to 6-DOF optimization
- fewer software programs to maintain

As stated above, the research presented in this paper has two main parts. The first part is a description of a novel application of a direct method within the framework of a 3/6-DOF vehicle simulation software tool and its application to the TLI trajectory optimization problem. The vehicle configuration includes an Earth Transfer Stage (ETS), an orbiter, lander, and a bipropellant chemical engine. The modeled trajectory initiates from a circular Low Earth Orbit (LEO) with main engine ignition; the ETS and propellant tanks are jettisoned an hour after ignition. The trajectory ends with the spacecraft flying over the north pole of the Moon. The second part of this thesis highlights the dual capability of the software tool by utilizing the results of the TLI optimization (the open-loop solution) in the form of certain orbital states at main engine cutoff (MECO) as input data into the vehicle simulation (the other "mode" in MASTIF) that assesses a potential candidate closed-loop guidance algorithm. The closed-loop guidance algorithm assessed is a variant of the Space Shuttle's guidance named Powered Explicit Guidance (PEG); the fact that it is flight-proven makes it a good candidate. Finally, the results from the open-loop and closed-loop solution are compared to show approximate agreement and efficacy of MASTIF.

As an aid to the reader, Appendix A provides a review of key terminology used throughout this thesis including the pros and cons of the direct method.

1.1 Literature review

1.1.1 Sample Trajectory Optimization Programs Used by NASA

Trajectory optimization programs have been around since the 1960's and those developed during that time period tended to utilize indirect methods. These programs specialize in solving a specific type of trajectory problem. Here are some examples. Solar Electric Propulsion Trajectory Optimization Program (SEPTOP) [6] specializes in interplanetary solar electric propulsion problems. These trajectories could include gravity assists from planetary flybys. Multidisciplinary Integrated Design Assistant for Spacecraft (MIDAS) [7] is a patched conic interplanetary program that optimizes discrete events such as launch and flyby dates. DUKSUP [8] was a popular program for launch ascent trajectories and used through the early 1990's. In the 1970's, Program to Optimize Simulated Trajectories (POST) [9] was developed for the Space Shuttle program; it specializes in atmospheric ascent and reentry trajectory problems. These programs had/have the advantage of relatively fast execution times, but some were/are notorious for sensitivity to initial guesses and were/are sometimes hampered by the difficulty in determining good adjoint variable guesses.

Since the 1980's, when digital computers became increasingly advanced, there has been a growth in developing new trajectory optimization programs, especially those that use the direct method, driven in large part by the increasingly complex trajectory designs. Mission Analysis Low-Thrust Optimization (MALTO) [10] , Mystic [11] and Copernicous [12] all were developed in within the last 20 years. All specialize in interplanetary trajectories. MALTO is used primarily for low-thrust trajectory optimizations. NASA's Dawn mission

[13] trajectory design work was accomplished with Mystic which uses dynamic programming theory as its foundation for optimization. Copernicous uses NLP software, such as SNOPT [3], as an optimization engine. Optimal Trajectories by Implicit Simulation (OTIS) [14, 5] is a trajectory optimization program that was initially developed by Boeing for the U.S. Air Force during the 1980's. Originally, it optimized atmospheric flight trajectories. OTIS is particulary noteworthy because it was the first program to use collocation in trajectory optimization. Hargaves and Paris [4] showed the need for sparse NLP packages such as SNOPT. Nowadays, SNOPT and other sparse solvers are commonly used in many trajectory optimization programs [12, 14, 10]. These newer programs are less specialized and less sensitive to initial guesses compared to the earlier developed trajectory optimization programs. Programs using the direct method are particularly attractive because constraints can easily be added, removed, or modified with minimal source code changes because no new analytical derivations are required. The best that a trajectory optimization program using an indirect method can do is to allow for a set of constraints for which the variational equations have already been derived and coded. Any new constraints would require deriving and coding a new set of variational equations. Regardless of the technique employed to achieve an optimal solution or when it was developed, optimization programs tend to be restricted to 3-DOF; 6-DOF optimization programs are a rarity–just one has been discovered [9] by this researcher. This is worth mentioning because 3-DOF optimization programs can only determine the optimal path through space with basic vehicle modeling (e.g. point mass).

Aerospace vehicle simulation programs serve a different purpose than optimization programs in that they are often the primary tool used to design and test a vehicle's guidance, navigation, and control (GN&C) system (for 6-DOF). Those used within NASA do not date back as far as some of the optimization programs. Marshall Aerospace Vehicle Representation in C (MAVERIC) [15] is a simulation program developed in the 1990's that was used recently for ARES I [16] design work. Mission Analysis Simulation Tool in Fortran (MASTIF) [17] was used to verify MAVERIC results for certain ARES I trajectories. Both programs are general enough to simulate in-space or atmospheric flight. Dispersion analysis, which can be performed with Monte-Carlo runs, is often desired. Gaining insight into the dynamics of the vehicle and knowing its response to hardware performance and environmental variations is the strength of a simulation program.

Optimization theory still has a significant role in vehicle GN&C algorithms. For example, the Space Shuttle's ascent guidance program, PEG, is based on the linear tangent steering law [18], derived using optimal control theory.

Note that a 6-DOF trajectory optimization program would have the ability to serve as a vehicle simulation program simply by switching modes (because a 6-DOF program likely has closed-loop guidance and control capabilities). In fact, POST functions this way. Typically, this is not the case for a 3-DOF optimization program because of the lack of navigation and closed-loop guidance algorithms.

1.1.2 Translunar Injection

The development of translunar injection (TLI) trajectories began in the early 1960s for the Apollo program. Two types of Earth-Moon transfers could be performed [19]. The first was a free-return that places the Saturn-IVB (S-IVB) on a high pericynthion trajectory. The free-return trajectory would flyby the Moon and return to Earth without requiring any main propulsion. This ensured a safe return of the crew in case of main engine failure, but the tradeoff was limited accessible lunar landing sites. The second option, used in later Apollo missions, was a hybrid trajectory in which the S-IVB started off on free-return and then, after the propulsion system passed checkout, performed a mid course correction (MCC) burn a few hours after TLI to place it on a non free-return Lunar approach trajectory. The hybrid trajectory offers better performance and more accessible landing sites than the free-return while still retaining a high degree of safety.

Targeting was performed using the geometrical hypersurface concept [20, 21] described by various orbital parameters (see [21]). Because the Earth and the Moon are in motion, these target parameters vary and must be constantly updated during the coast phase (prior to the TLI burn). Saturn V's on-board targeting scheme used table lookups of some of the orbital parameters as functions of a reference time. The restart preparation sequence started more than five minutes prior to the re-ignition command. Trajectory optimization results essentially controlled the moment the restart preparation began and the time of re-ignition. Later in the Apollo program, real-time targeting was performed using a computer program at what is now Johnson Space Center (JSC). This enabled updates to the on-board target parameters in case large dispersions were detected or a hybrid translunar transfer was selected [22].

At the start of the TLI burn, the hypersurface target parameters were frozen. At each guidance cycle, the on-board guidance system computed the desired MECO parameters (position vector magnitude, R, velocity vector magnitude, V, and flight path angle, γ) as functions of various orbital parameters. After determination of the MECO targets, the algorithm proceeded to calculate the desired Euler angles θ and ψ. The Iterative Guidance Mode (IGM) served as Saturn V's launch ascent and TLI guidance program [23, 24]. In the ascent portion the MECO targets were fixed (determined with trajectory optimization), whereas they were recalculated (from table lookups of numerous trajectory optimization runs) every guidance cycle for TLI.

Rocket-powered guidance systems have made advancements since Apollo. The Space Shuttle's PEG is an example [25]. Others have also contributed to launch ascent guidance advancements [26, 27]. On the other hand, any advancements made in TLI targeting/guidance has proven to be elusive to find by this researcher. This seems unsurprising considering that while rockets continue to be developed and launched, the final destination is generally some type of Earth orbit (LEO, GEO, etc.) and not the Moon. In January 2004, President George W. Bush announced the VSE for NASA that calls for returning humans to the Moon by 2020. Although returning humans to the Moon by 2020 seems in doubt because of NASA's recent budgets, it seems plausible that at some point in the future humans will be returning to the Moon – at which time TLI guidance will be critical part of the overall GN&C software just as it was during Apollo.

PEG and other more recent guidance algorithms should be able to perform TLI guidance [28]. However, these algorithms were developed for ascent to orbit and do not update the MECO target parameters. They simply fly to an optimal set of MECO targets typically determined by some other trajectory optimization program. It will be important that the MECO targets be recomputed periodically during the actual TLI burn to accommodate vehicle performance and environment variations. This will ensure that the desired Lunar arrival conditions are met while minimizing propellant expenditures from the MCC(s) along the transfer.

1.2 Thesis Contribution

This thesis work has three main contributions:

- Provide NASA and the aerospace community with a software tool that can perform trajectory optimization and serve as a low-to-high fidelity vehicle simulation program. This is accomplished by developing a new optimization mode within MASTIF. The optimization work in this thesis is restricted to 3-DOF, but this new optimization mode is developed with the ultimate goal of its extension to 6-DOF optimization in mind.
- Use this new optimization mode, described in this thesis, to determine an optimal Earth-Moon transfer trajectory for a human mission to the Moon.
- Assess the performance of a candidate closed-loop guidance system for the TLI burn using the simulation mode and determine the impact of using fixed MECO target parameters for the first analysis design "cycle". The closed-loop guidance algorithm assessed, PEG, is a candidate because it is flight-proven.

1.3 Thesis Organization

Chapter 1 introduces the TLI problem and provides the motivation for its solution. A review of the TLI problem in the context of the Apollo program is given. Sample trajectory optimization programs used within NASA are discussed as well. Chapter 2 provides an overview of the TLI trajectory modeling including relevant dynamic force models and mission constraints. Then, the software program, MASTIF, that enables this modeling numerically is described, followed by a brief generalized description of collocation. The reader is then introduced to relevant fundamental coordinate frames. The equations of motion are then described using these coordinate frames. Control variables that are to be used to steer the vehicle during the TLI burn are described. The problem formulation is then developed mathematically and includes required NLP variables such as the objective function and constraints. Finally, Chapter 2 concludes with a brief overview of the closed-loop guidance algorithm, PEG (used in the vehicle simulation, not the trajectory optimization). Chapter 3 covers trajectory optimization software development and provides the collocation method using NLP in generalized settings. A brief overview of SNOPT is given. Chapter 3 is crucial to understanding how a trajectory optimization problem (in MASTIF) is transcribed to a NLP problem and solved with SNOPT. Numerical results are provided in Chapter 4. Open-loop and closed-loop results are compared to demonstrate the efficacy of MASTIF. The last chapter, Chapter 5, summarizes the research and provides future direction.

Note that three appendices are provided for easy reference. Appendix A defines terminology that is used commonly throughout this thesis. Appendix B reviews key quaternion mathematics and important coordinate frame transformations, utilizing quaternions, are provided in Appendix C. Finally, Appendix D contains the Hermite interpolating polynomial expressions.

Chapter 2

Translunar Injection Problem

2.1 Overview

The Earth to Moon transfer problem to be solved consists of the following sequence of events and modeling assumptions. Starting in circular 29° LEO, at the epoch of January 1, 2018 (midnight), perform a main engine burn to place the spacecraft on the correct three day transfer ellipse while maximizing the mass at MECO. The start-up and shut-down transients of the engine are ignored, but can be included in the future. The atmospheric drag is also ignored for now. Approximately one hour after the start of the burn, the spacecraft jettisons the ETS. Flight performance reserve (FPR) propellant left over from the ascent to orbit and reaction control system (RCS) propellant (carried in two separate propellant tanks) are also jettisoned with the ETS. The propellant in these two tanks are not available for use for this 3-DOF optimization; it is reserved for 6-DOF simulations. Propellant remaining in the tanks from the main engine burn are carried along to the Moon, but could easily be jettisoned as modeling is refined for future needs.

The spacecraft is to arrive at the Moon flying over the north pole ($inc = 90\,^{\circ}$) at a 100 km pericynthion altitude. The relatively small ΔV required to capture the spacecraft into

Table 2.1: Propulsion and mass properties

I_{sp}	448 sec
T_{vac}	1,070.5936 kN (240,679 lb_f)
MR	5.5
m_0	187,901 kg (414,250 lb_m)
m_{ETS}	25,463 kg (56,137 lb_m)
m_{LOX}	81,887 kg (180,531 lb_m)
m_{LH2}	14,889 kg (32,824 lb_m)
m_{FPR}	1,449 kg (3,194 lb_m)
m_{RCS}	342 kg (754 lb_m)

a Lunar orbit is not currently modeled. Table 2.1, above, lists the LOX/LH2 bipropellant engine's specific impulse, I_{sp}, vacuum thrust, T_{vac}, and mixture ratio, MR. The spacecraft's initial mass, m_0, includes the mass of the ETS, m_{ETS}, and the mass of the filled fuel and oxidizer tanks, m_{LH2} and m_{LOX}, respectively. Propellant reserved for the RCS, m_{RCS}, and FPR, m_{FPR}, are also listed.

Gravitational effects and 1^{st} order harmonics (J_2) from the Earth, Sun, and Moon are modeled. The values for the physical constants for these three bodies came from [29]. Ephemeris data (Earth, Moon, and Sun time-dependent positions) came from the Jet Propulsion Laboratory's SPICE software [30].

2.2 Mission Analysis Simulation Tool in Fortran

Mission Analysis Simulation Tool in Fortran (MASTIF) is a low-to-high fidelity vehicle simulation program written primarily in Fortran 90/95/2003 with some C/C++ as well. It is an input file driven program that runs natively on Linux and Macintosh operating systems. Development started in 2006 to support NASA's ARES I project within the Constellation Program, with this researcher leading a small team of developers. The ARES I was a rocket intended to replace the Space Shuttle. MASTIF was designed to be flexible enough to support not only launch ascent trajectories, but in-space trajectories as well. Over the years, it has been used successfully to help verify and validate certain ARES I trajectory design cycles and

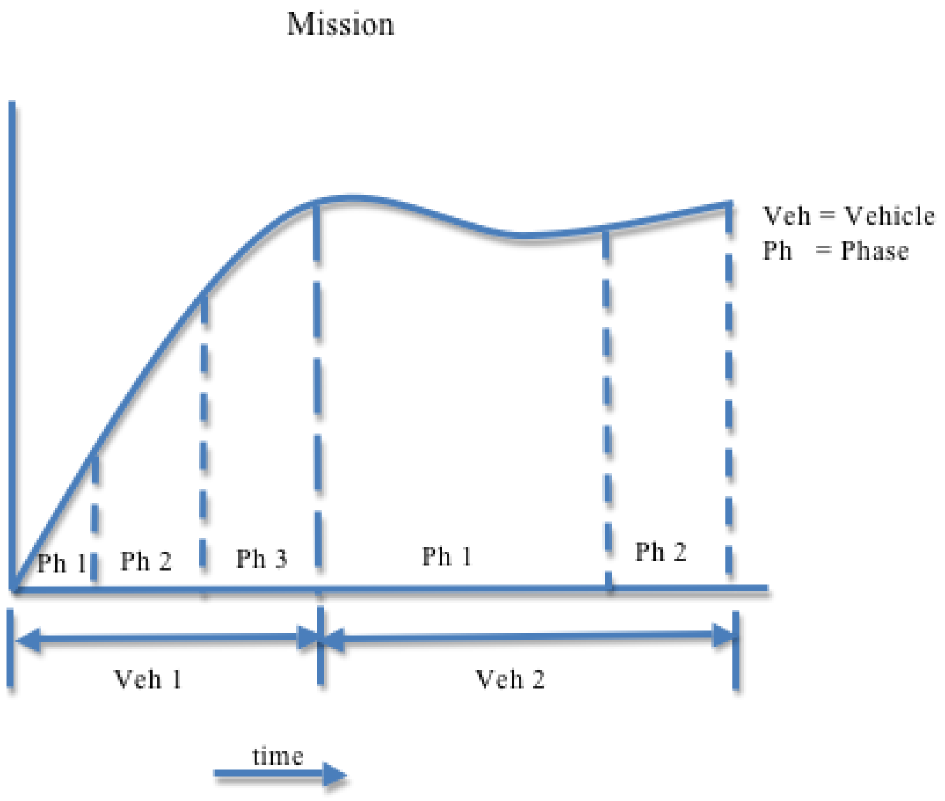


Figure 2.1: Relationship of phases, vehicles, and mission within MASTIF

for LEO analysis. It can be used for 3-DOF and 6-DOF simulations. The object-oriented language of modern Fortran enables major entities within MASTIF to be treated as derived data types (Fortran parlance), which are called structures in the C language. The three major derived data types used within MASTIF are the mission, vehicle, and phase. The mission is the most top-level derived data typ and consists of time, input/output information, and vehicles to name a few. Vehicles simply consist of other derived data types such as phases, propellant tanks, engines, RCS thrusters, actuators, etc. Mass properties, propulsion and aerodynamic models are all attributes of a vehicle. Phases are used to provide transitions, usually from one model to another. Phases are comprised of models such as environment and gravitational bodies. A top-level schematic in 2.1 shows the relationship for a notional mission.

Key features of MASTIF include the ability to easily swap out different models. These models include force models and GN&C models. Different models can be added either through source code or mathematical expressions that are character strings contained in an input file and parsed and evaluated internally. Additionally, MASTIF is very flexible regarding physical units: variables can be input and output in any unit(s), and internal working units can be controlled as well.

2.3 Collocation

Collocation is a numerical procedure that results in parameterizing the *potential* solutions to the differential equations (the states) and controls at discrete points in time (referred to as nodes or knots) [31, 32]. Local piece-wise polynomials approximate the states and controls at these nodes. Differences between these polynomial approximation solutions of the states and controls and the state and control values (obtained in some other way) at the same nodes are errors and called defects [4]. No explicit integration is required; however, only when the defects are zero does the polynomial approximation of the states become valid (meaning that the physics of the problem are not violated). The formulation of the polynomial (including the order) and defects can vary, but the polynomials still remain local. In the context of a trajectory optimization, collocation alone is not very useful because the true states and control values are still unknown. However, once combined with a NLP solver, collocation becomes very useful and effective at solving a wide-range of aerospace trajectory optimization problems [4, 5]. This is because the NLP solver is the mechanism that provides the state and control values at discrete points (the "other way" mentioned above) *while simultaneously driving the defects to zero and optimizing the trajectory.*

2.4 Translunar Injection Trajectory Discretization Modeling

Two vehicles and three (total) phases are utilized to model the problem. The first vehicle includes the spacecraft and the ETS and is modeled with two phases. The first phase is the TLI burn and the second phase is a nearly one hour coast phase. The second vehicle does not include the ETS and the FPR and RCS propellant tanks. Its primary role is to easily facilitate the jettision of the ETS and two propellant tanks within the software design of MASTIF. The second vehicle has one long coast phase. The first two phases are collocation phases (described further in Chapter 3) whereas the last phase is an explicitly integrated phase using a high order Runge-Kutta integrator [33].

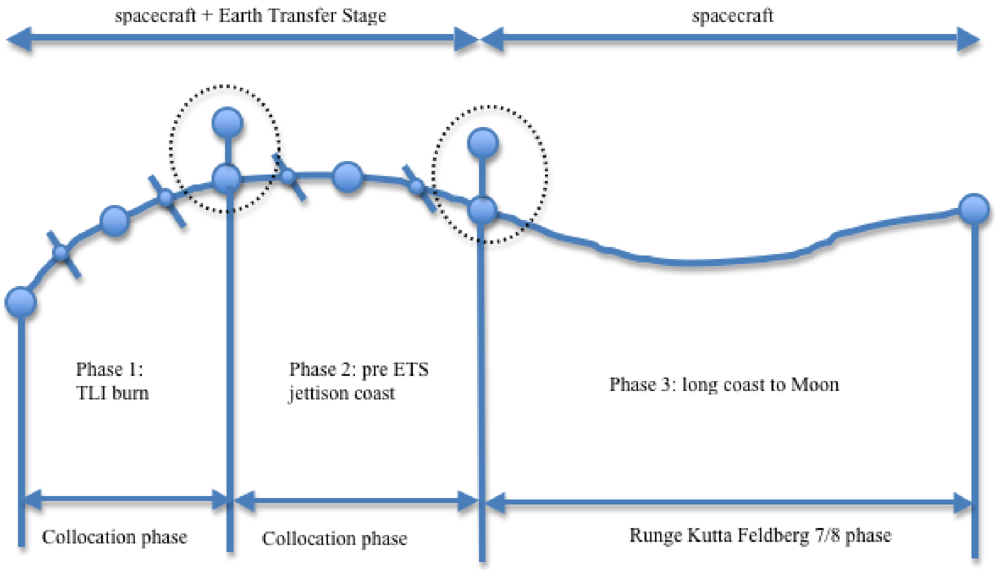

spacecraft + Earth Transfer Stage
spacecraft
Phase 1:
TLI burn
Phase 2: pre ETS
jettison coast
Phase 3: long coast to Moon
Collocation phase
Collocation phase
Runge Kutta Feldberg 7/8 phase

Figure 2.2: Vehicle and phase schematic

Figure 2.2 shows a schematic of the vehicle and phase layout (with minimal nodes for clarity) and Figure 2.3 depicts the double nodes that are utilized at all phase (k) boundaries to enable discontinuities in the vehicle states ($\mathbf{s}$) and/or controls ($\mathbf{u}$) and will also be discussed in further in Chapter 3.

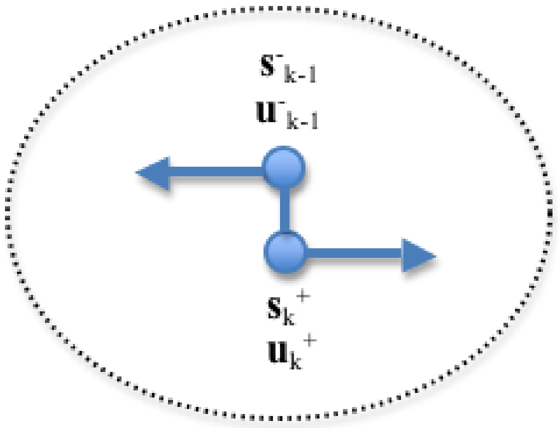


Figure 2.3: Double nodes at phase boundaries

The discretization of each phase is shown in Table 2.2. Each of the first two phases are subdivided into a number of segments, $nSeg_{ph}$, and each segment contains a certain number of nodes, $nNodes_{seg}$. The relationship between segments, nodes, and phases will be discussed in the next chapter. High order polynomials used in the 2nd phase were used to test the robustness of generating higher order polynomial approximations.

Table 2.2: TLI phase inputs

phase	method	$nSeg_{ph}$	$nNodes_{seg}$
1	collocation	14	5
2	collocation	3	21
3	Runge-Kutta 7/8	n/a	n/a

In general, there exist different discretization schemes, expecting the same results within the range of accuracy that might vary from user to user. Discretization is advantageous because it enables the user to control solution accuracy, which is especially important during the initial attempts at a solution when less accuracy is required. Permitting user control over the number of nodes per segment and the number of segments per phase is analogous to controlling the order of integration method and integration step-size, respectively, in an explicit integration. This numerical control essentially allows accuracy to vary from phase to phase according the modeled dynamics of the problem. In this thesis, the discretization process is iterative and to accurately represent the trajectory, there must be enough grid points (placement is also important) considered for the applied algorithm. Table 2.3 lists the phase (input) timing and bounds.

Table 2.3: TLI phase time inputs

phase	$t_{0_{lwr}} \leq t_0 \leq t_{0_{upr}}$	$\Delta t_{lwr} \leq \Delta t \leq \Delta t_{upr}$
1	$0 \leq 0 \leq 0$	$380sec \leq 400sec \leq 450sec$
2	$380sec \leq 400sec \leq 450sec$	$53.4min \leq 53.4min \leq 53.4min$
3	$59.2333min \leq 60.0667min \leq 60.9min$	$66.213hr \leq 70.99hr \leq 71.01hr$

The phase start times for phase 2 and 3 are free whereas it is fixed for phase 1. Likewise, the phase durations are also free for two phases and fixed for the other.

Table 2.4: TLI constraints

phase	type	number of constraints
1	$\boldsymbol{\chi}$	3
2	none	0
3	$\boldsymbol{\Psi}$	3

Each phase's number of constraints and type are listed in Table 2.4. A total of six constraints are utilized; three initial, $\boldsymbol{\chi}$, and three final, $\boldsymbol{\Psi}$. The second phase is the only phase without active constraints.

2.5 Coordinate Frames

2.5.1 Keplerian

References to various Keplerian elements are used throughout this thesis. Figure 2.4 depicts five of the six classical elements for an elliptical orbit around Earth. Their definitions are provided here for convenience and can be found in just about any textbook covering orbital mechanics such as [34].

- a: semi-major axis (not shown) describes the size of the orbit
- e: eccentricity vector–points from center of Earth to perigee, the magnitude describes the shape of the orbit
- i: inclination–the angle between the angular momentum vector, $\bar{h}$, and the unit vector in the $\bar{Z}$ direction (hereafter referenced as inc)
- Ω: right ascension of ascending node–angle from the Vernal Equinox to the ascending node
- ω: argument of perigee–the angle from the ascending node to the eccentricity vector
- ν: true anomaly–the angle from the eccentricity vector to the satellite's position vector

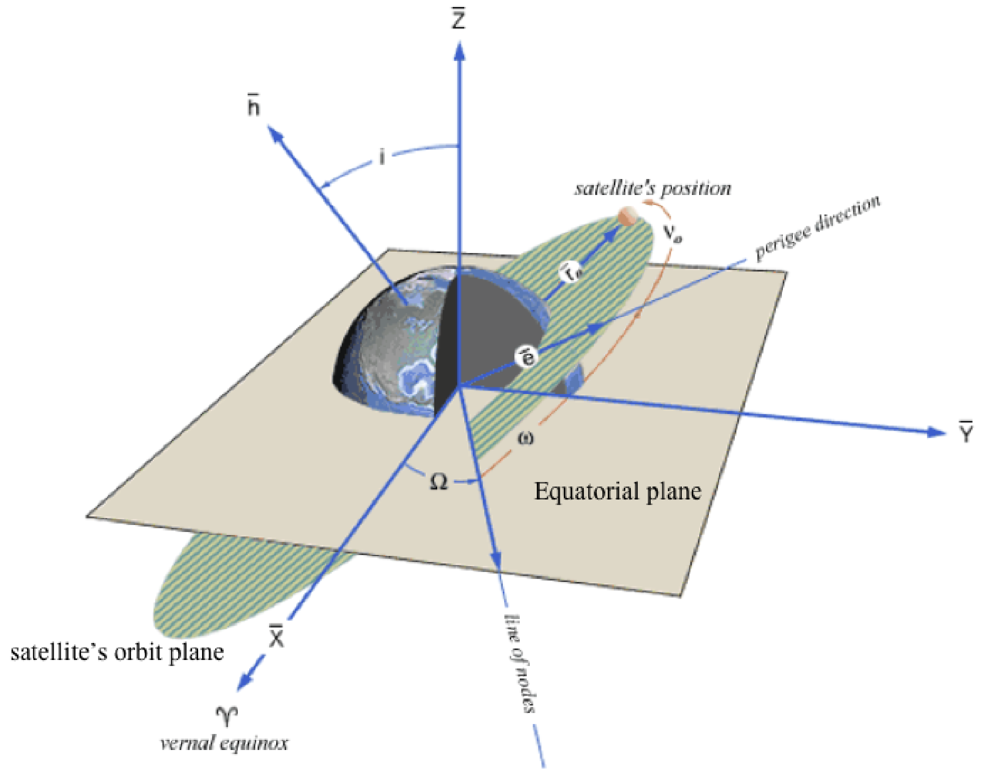


Figure 2.4: Keplerian classical orbital elements [35]

Some of these angles (ω and ν) become undefined for circular orbits and new definitions arise. Numerical singularities arising from circular orbits ($e = 0$) and/or equatorial orbits ($inc = 0\,^{\circ}$ or $180\,^{\circ}$) must be avoided.

2.5.2 Cartesian

Four primary Cartesian coordinate frames are referenced often. Two are inertial (non-accelerating) and two are non-inertial (relative). The first inertial frame is the Earth Mean Equator and Equinox of Epoch J2000 (EME2000) . The standard reference epoch is January 1, 2000, noon Terrestrial Time (TT). For a detailed description of the time scales (including TT) the reader is referred to [36]. The x-axis points in the direction of the Vernal Equinox

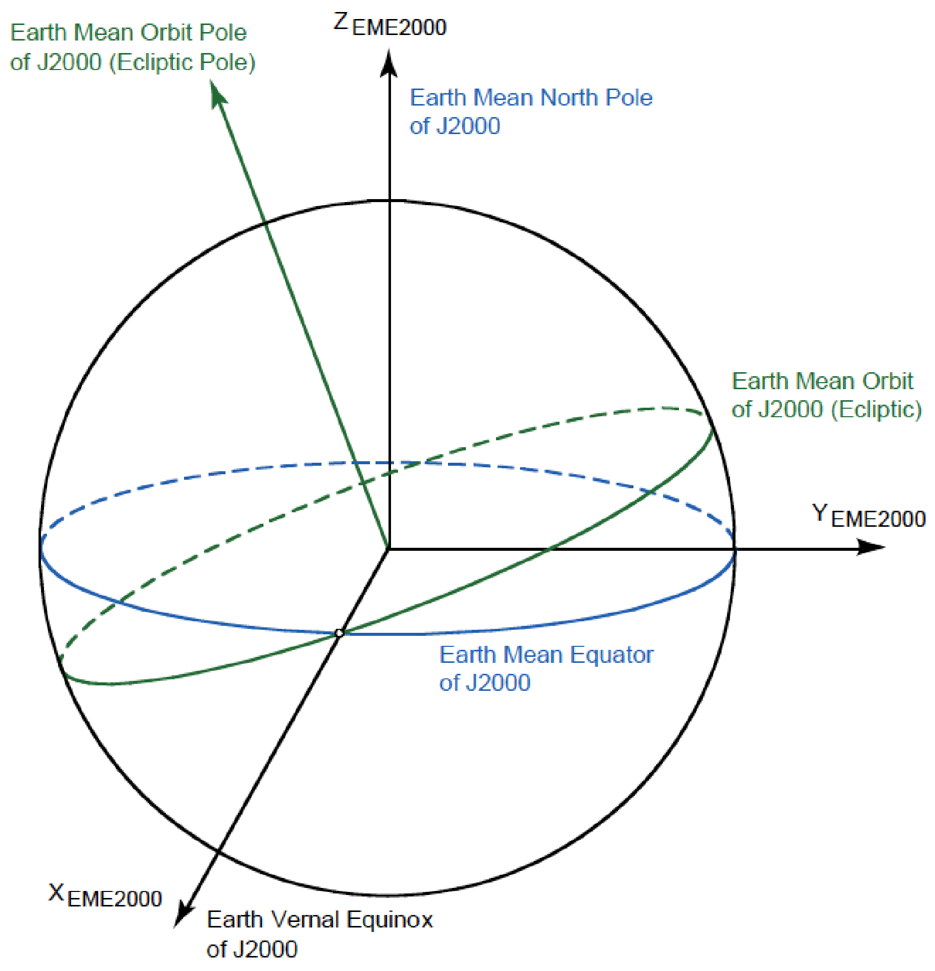


Figure 2.5: Earth Mean Equator and Equinox of Epoch J2000 [29]

of the Earth mean orbit at J2000. The z-axis is normal to the Earth mean equator at J2000 and the y-axis completes the right-handed system. The second inertial frame is the Moon-Centered, Moon Mean Equator of Epoch and IAU-Node of Epoch (MCME). Figures 2.5 and 2.6 illustrate both of these frames. The coordinate transformation from EME2000 to the Moon frame is given in Appendix C.

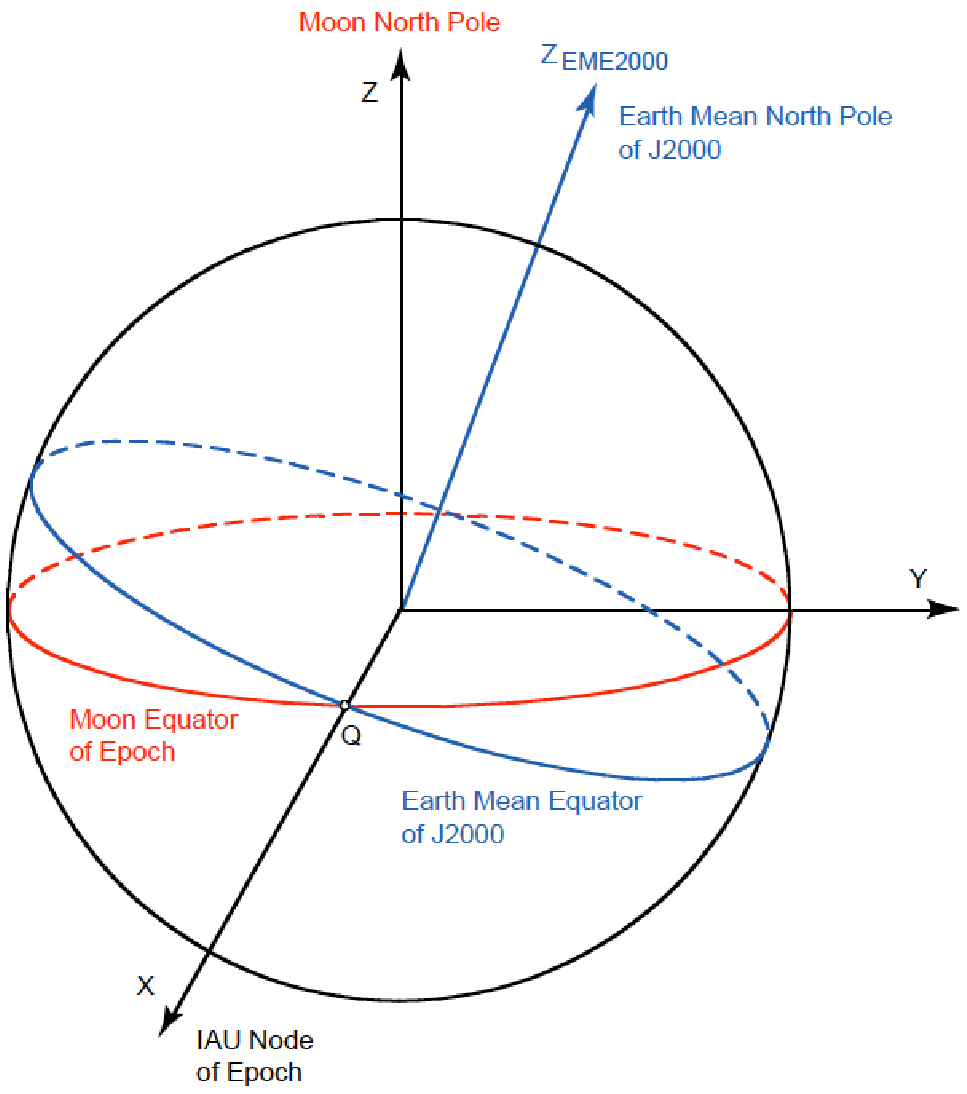


Figure 2.6: Moon-Centered, Moon Mean Equator of Epoch and IAU-Node of Epoch [29]

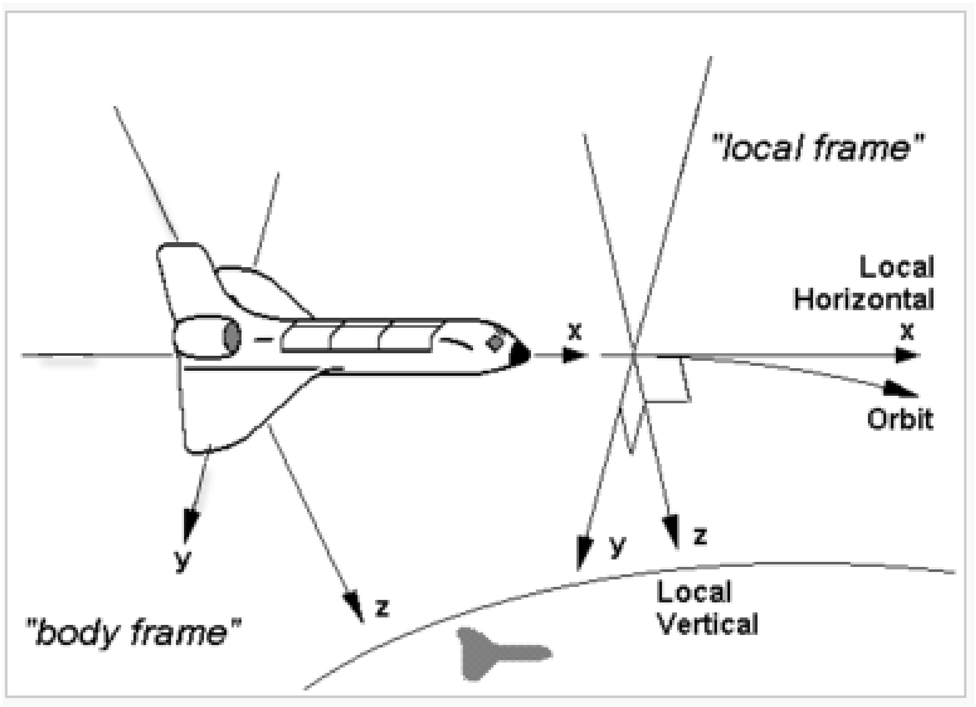


Figure 2.7: Body and LVLH frames [37]

The two relative frames are useful for describing the attitude of the vehicle. The origin of both is at the vehicle's center of mass. The first is the vehicle's body frame. The x-axis points out the nose of the vehicle, the y-axis points out the right-hand side (assuming a "heads-up" orientation) and the z-axis completes the right-handed system. The other relative frame is the Local-Vertical Local-Horizontal (LVLH) frame. The z-axis (local vertical) points from the vehicle's center of mass along the (negative) radius vector. The positive y-axis points along the negative orbit angular momentum vector and the x-axis (local horizontal) completes the right-handed system. Three Euler angles, ϕ, θ, and ψ, describe the relationship between these two frames. These Euler angles are loosely referred to as roll, pitch, and yaw, respectively. Figure 2.7 shows these two frames. Appendix B provides the coordinate transformations between them. Note that, in general, the derivatives of roll, pitch, and yaw, with respect to time, are not the same as the body rates (commonly referred to as p, q, and r).

2.6 Equations of Motion[1]

The equations of motion are modeled in EME2000 Cartesian coordinates. The velocity vector is given by (2.1) (EME2000 is also referred to as J2000, hence the subscript name). Because thrust is the only (modeled) non gravitational force exerted on the vehicle, the acceleration term, (2.2), becomes

$$\dot{\mathbf{r}}_{\mathbf{j2k}} = \mathbf{v}_{\mathbf{j2k}} \tag{2.1}$$

$$\dot{\mathbf{v}}_{\mathbf{j2k}} = \frac{T_{vac}}{m}\hat{\boldsymbol{\mu}}_{\boldsymbol{j2k}} + \mathbf{g}_{\mathbf{j2k}} \tag{2.2}$$

where $\hat{\boldsymbol{\mu}}_{\boldsymbol{j2k}}$ is the direction of thrust (unit magnitude) expressed in EME2000 coordinates and T_{vac} is the vacuum thrust (atmospheric effects are neglected) listed in Table 2.1. The acceleration due to gravity is the sum of gravitational accelerations of the Earth, Sun, and Moon:

$$\mathbf{g}_{\mathbf{j2k}} = \mathbf{g}_{\mathbf{j2k},\oplus} + \mathbf{g}_{\mathbf{j2k},\odot} + \mathbf{g}_{\mathbf{j2k},☾} \tag{2.3}$$

and the evaluation is based on that given in [38].

The mass of the vehicle depends on the phase and is determined from a table lookup as a function of total remaining propellant:

$$m = \begin{cases} f(table[m_{LOX} + m_{LH2} + m_{FPR} + m_{RCS}]) + m_{ETS} & \text{if } phase \leq 2, \\ f(table[m_{LOX} + m_{LH2}]) & \text{if } phase = 3. \end{cases} \tag{2.4}$$

To complete the dynamical model, the propellant tank mass flow rates are

[1]NOTE: scalars are regular characters whereas vectors/arrays are **bold**

$$\dot{\mathbf{m}}_{\mathbf{tnk}} = \begin{cases} \begin{bmatrix} \dot{m}_{LOX} \\ \dot{m}_{LH2} \\ \dot{m}_{FPR} \\ \dot{m}_{RCS} \end{bmatrix} & \text{if } phase \leq 2, \\ \\ \begin{bmatrix} \dot{m}_{LOX} \\ \dot{m}_{LH2} \end{bmatrix} & \text{if } phase = 3 \end{cases} \tag{2.5}$$

where the LOX and LH2 tank mass flow rates are calculated as:

$$\begin{aligned} \dot{m}_{LOX} &\equiv \frac{-\dot{m}_{eng} MR}{MR + 1} \\ &= -206.2 kg/s \; (-454.6 lb_m/s) \end{aligned} \tag{2.6}$$

and

$$\begin{aligned} \dot{m}_{LH2} &\equiv \frac{-\dot{m}_{eng}}{MR + 1} \\ &= -37.5 kg/s \; (-82.7 lb_m/s) \end{aligned} \tag{2.7}$$

where

$$\begin{aligned} \dot{m}_{eng} &\equiv \frac{T_{corr}}{I_{sp} g_{\oplus SL}} \\ &= 243.7 kg/s \; (537.3 lb_m/s) \end{aligned} \tag{2.8}$$

and $g_{\oplus SL}$ is the magnitude of the Earth's gravity vector at sea level. Note that

$$\dot{m}_{FPR} = \dot{m}_{RCS} = 0 \tag{2.9}$$

because no propellant is drawn from these two tanks.

2.7 Problem Formulation[2]

Let the dynamical system states, $\mathbf{x}(t)$, be the solutions to the equations of motion, (2.1), (2.2), and (2.5), defined in the previous section and the control variables (or simply control), $\mathbf{u}(t)$, be some or all of the Euler angles that relate the vehicle body frame to the LVLH frame (defined more precisely in the next section), then an optimal control problem minimizes a cost function

$$J = J(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.10}$$

subject to the system's equations of motion,

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.11}$$

and any initial constraints

$$\boldsymbol{\chi}(\mathbf{x}(t_0), t_0) = \mathbf{0} \tag{2.12}$$

terminal constraints

$$\boldsymbol{\Psi}(\mathbf{x}(t_f), t_f) = \mathbf{0} \tag{2.13}$$

and path contraints

$$\mathbf{c}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{0} \tag{2.14}$$

where

$t_0 \leq t \leq t_f,\ \mathbf{x} \in \mathbb{R}^{n\times 1},\ \mathbf{u} \in \mathbb{R}^{m\times 1},\ \boldsymbol{\chi} \in \mathbb{R}^{p\times 1},\ \boldsymbol{\Psi} \in \mathbb{R}^{q\times 1},\ and\ \mathbf{c} \in \mathbb{R}^{r\times 1}$ and t_0 and t_f are the initial and final times, respectively. This is a more generalized formulation than is needed, because no path constraints were used for the TLI trajectory optimization.

[2]NOTE: scalars are regular characters whereas vectors/arrays are **bold**

Also, recognize that this formulation does not depend on the number or type of states and controls.

The exact form (i.e. expression) of (2.10) is determined from the method selected to solve the optimal control problem. If a Calculus of Variations (CoV) approach is pursued then the form of (2.10) would be either the Bolza, Lagrange, or Mayer form [31] depending on which is preferred for the specific problem being solved. It is highly desirable for the optimization mode in MASTIF to be robust and flexible and so a direct approach is pursued instead of an indirect approach. The optimal control problem is transcribed to a nonlinear constrained optimization problem of the form:

$$\textit{minimize } J(\mathbf{x}) \quad \textit{subject to} \qquad \mathbf{L} \leq \begin{pmatrix} \mathbf{x} \\ \mathbf{f}(\mathbf{x}) \\ \mathbf{g}(\mathbf{x}) \end{pmatrix} \leq \mathbf{U} \tag{2.15}$$

and solved by SNOPT. The first element in (2.15), $\mathbf{x}$, is an array containing all the independent variables *including the controls*, $\mathbf{u}$ ($\mathbf{x}$ and $\mathbf{u}$ given in (2.10) are now combined into a single array). At the start of the problem, it contains the initial guesses for all the independent variables. During the optimization process, it is an output array from SNOPT. The second element in (2.15), $\mathbf{f}(\mathbf{x})$, is an input array into SNOPT containing linear and nonlinear constraints and the objective function, $J(\mathbf{x})$. The last element, $\mathbf{g}(\mathbf{x})$, is an (optional) input array into SNOPT containing the 1^{st} order partial derivatives of $\mathbf{f}(\mathbf{x})$. It is assumed that the derivatives are smooth and continuous up to 2^{nd} order. $\mathbf{L}$ and $\mathbf{U}$ are the given lower and upper bounds.

2.7.1 Controls, Objective Function, and Constraints[3]

To enable the TLI trajectory optimization problem to be transcribed to a constrained NLP problem, the controls, $\mathbf{u}$, the objection function, J, and the constraints are defined in this section. Phase 1 is the only phase with active controls and an LVLH Euler angle set is selected (see Figure 2.7). The control input is thus defined as:

$$\mathbf{u} = \begin{bmatrix} \phi_{lvlh} \\ \theta_{lvlh} \\ \psi_{lvlh}. \end{bmatrix}. \tag{2.16}$$

To meet the mission objective of maximizing mass, the objective function is the minimum engine burn time, $t_{b_{eng}}$, which is equivalent to maximizing cutoff mass. It is also the same as minimizing the duration of phase 1 (Δt_1):

$$\begin{aligned} J &= t_{f_1} - t_{0_1} \\ &= t_{b_{eng}} \end{aligned} \tag{2.17}$$

where t_{f_1} and t_{0_1} are final time and initial time, respectively, in phase 1.

The initial LEO is constrained in a way that ensures the orbit is nearly circular, has the correct size and is correctly tilted. The orientation of the orbit plane is prescribed by the ascending node angle which is unconstrained. Optimal timing is achieved by allowing the argument of perigee and true anomaly to be unconstrained. The three initial constraints are:

(correct size)

$$\chi_1 = a(t_{0_1}) - a \tag{2.18}$$

[3]NOTE: scalars are regular characters whereas vectors/arrays are **bold**

(correct tilt)

$$\chi_2 = inc(t_{0_1}) - inc \tag{2.19}$$

(correct shape)

$$\chi_3 = e(t_{0_1}) - e \tag{2.20}$$

where a, inc, and e are listed in Table 2.5.

Table 2.5: Initial semi-major axis, inclination, and eccentricity

a km (nmi)	inc degree	e
6,563 (3,544)	29	$1.0e^{-8}$

To ensure that the spacecraft arrives on the desired trajectory, three final constraints, relative to the Moon, are used in the last phase (phase 3)

$$\mathbf{\Psi} = \begin{bmatrix} alt_p(t_{f_3}) - alt_p \\ inc_{wrt☾}(t_{f_3}) - inc_{wrt☾} \\ \dot{r}_{wrt☾}(t_{f_3}) - \dot{r}_{wrt☾} \end{bmatrix} \tag{2.21}$$

where the radial velocity component is given by

$$\dot{r}_{wrt☾} = \frac{e_{wrt☾}\sqrt{-a_{wrt☾}\, gm_{☾}}\,\sinh H_{wrt☾}}{a_{wrt☾}(1 - e_{wrt☾}\cosh H_{wrt☾})}. \tag{2.22}$$

The radial velocity component constraint ensures that the pericynthion altitude is attained and is the altitude of closest approach. It is equivalent to arriving at a hyperbolic anomaly of zero. The three desired lunar arrival values are listed in Table 2.6. The semi-major axis, $a_{wrt☾}$, is negative because the trajectory is hyperbolic relative to the Moon. $H_{wrt☾}$ and $gm_{☾}$ are the hyperbolic anomaly and the Moon's gravitational constant, respectively.

At this point, the equations of motion given by (2.1), (2.2) and (2.5), and the controls

Table 2.6: Desired Lunar arrival conditions

alt_p	$inc_{wrt☾}$	$\dot{r}_{wrt☾}$
km (nmi)	degree	km/s (ft/s)
100 (54)	90	$1.0e^{-8}$ ($3.28e^{-5}$)

given by (2.16) have been defined for the TLI trajectory problem. The NLP problem formulation has also been provided by (2.15). Chapter 3 describes the procedure to transcribe (2.1), (2.2), (2.5), and (2.16) into a NLP problem of the form given by (2.15) for a general trajectory optimization problem. Before proceding on to Chapter 3, a brief description of the closed-loop guidance algorithm that is used in the TLI vehicle simulation is provided in the next section. As stated previously, key (optimal) outputs from the TLI trajectory optimization are inputs for the vehicle simulation.

2.8 Closed-loop Guidance

2.8.1 Powered Explicit Guidance

PEG is the closed loop guidance algorithm used on the Space Shuttle to handle all phases of its exoatmospheric powered flight. It uses closed-form equations for the propulsive acceleration term (called thrust integrals) to accommodate constant thrust or constant acceleration phases of flight. The fuel-optimal time history of the unit thrust vector and its rate of change is derived using CoV. PEG can support up to 7 different end conditions. PEG's simplicity and heritage make it attractive. Heritage is an important factor in the design selection process because proven designs can significantly reduce risk and costs associated with testing. Slightly different versions of PEG appear in the literature [25, 24, 39] and have minor differences in the thrust integral terms. The version in MASTIF is based on [24].

Chapter 3

Trajectory Optimization Software Development

3.1 Transcription Procedure[1]

The procedure to transcribe a general trajectory optimization problem to a constrained NLP problem includes discretizing the equations of motion through a collocation scheme [31, 4, 14]. In collocation, the states and controls are parameterized and (local) polynomials are used to approximate the solution to the equations of motion at strategic (collocation) points (referred to as nodes). These parameterized states and controls are free variables; after the transcription procedure they are part of the NLP problem. Hermite interpolating polynomials, provided in Appendix D, approximate the solution to the differential equations (the states). An efficient numerical implementation is given in [40].

A partial set of Cartesian equations of motion is given by:

$$\dot{\mathbf{r}}_{\mathbf{j2k}} = \mathbf{v}_{\mathbf{j2k}} \tag{3.1}$$

[1]NOTE: scalars are regular characters whereas vectors/arrays are **bold**

and

$$\dot{\mathbf{v}}_{\mathbf{j2k}} = \frac{\mathbf{F}_{\mathbf{j2k}}}{m} + \mathbf{g}_{\mathbf{j2k}} \tag{3.2}$$

where $\dot{\mathbf{r}}_{\mathbf{j2k}}$ is the derivative of the position vector of the vehicle expressed in EME2000 coordinates. The acceleration term, $\dot{\mathbf{v}}_{\mathbf{j2k}}$, is a function of all the forces acting on the vehicle, $\mathbf{F}_{\mathbf{j2k}}$, the vehicle's mass, m, and the acceleration due to gravity, $\mathbf{g}_{\mathbf{j2k}}$. To complete this set of equations of motion, a mass flow rate term must be included. The mass flow rate of an engine is defined in Chapter 2 and is repeated here for convenience:

$$\dot{m}_{eng} = \frac{T_{corr}}{I_{sp} g_{\oplus SL}} \tag{3.3}$$

where T_{corr} represents the vacuum thrust that has been corrected due to atmospheric pressure effects (if any) and I_{sp} is the specific impulse of the engine. The mass flow rates in the equations of motion are the mass flow rates of the propellant tanks:

$$\dot{m}_{tnk} = \begin{cases} -\dot{m}_{eng} & \text{if } \textit{monoprop engine}, \\ -\dot{m}_{eng}(f(MR)) & \text{if } \textit{biprop engine} \end{cases} \tag{3.4}$$

where $f(MR)$ indicates that for a bipropellant engine the mass flow rate, $\dot{m}_{tnk}$, is also a function of the mixture ratio (MR), and it is understood that the subscript *tnk* is a generic name. The total number of equations of motion variables is then 6 + number of propellant tanks. The state variables (the solutions to (3.1), (3.2), and (3.4)) are a subset of variables contained in $\mathbf{x}$ from (2.15). Control parameters, $\mathbf{u}$, phase start times, $\mathbf{t_0}$, and phase durations, $\mathbf{\Delta t}$, when combined with the state variables comprise the full set of independent variables contained in $\mathbf{x}$. For organizational convenience, combine (3.1), (3.2), and (3.4) into a single array as

$$\dot{\mathbf{s}} = \begin{bmatrix} \dot{\mathbf{r}}_{j2k} \\ \dot{\mathbf{v}}_{j2k} \\ \dot{\mathbf{m}}_{tnk} \end{bmatrix} \tag{3.5}$$

where the states are

$$\mathbf{s} = \begin{bmatrix} \mathbf{r}_{j2k} \\ \mathbf{v}_{j2k} \\ \mathbf{m}_{tnk} \end{bmatrix}. \tag{3.6}$$

The 1^{st} array element, $\mathbf{x}$, from (2.15) becomes

$$\mathbf{x} = \begin{bmatrix} \mathbf{s} \\ \mathbf{u} \\ \mathbf{t}_0 \\ \mathbf{\Delta t} \end{bmatrix} \tag{3.7}$$

where $\mathbf{s} \in \mathbb{R}^{(6+nTnks)\times 1},\ \mathbf{u} \in \mathbb{R}^{3\times 1},\ \mathbf{t}_0 \in \mathbb{R}^{nPh\times 1}, and\ \mathbf{\Delta t} \in \mathbb{R}^{nPh\times 1}$.

The discretization scheme implemented is inspired by the scheme described in [4] and follows nearly exactly that given in [14] including the exact expressions for the polynomials. Each phase, k, is discretized into segments that contain nodes. Dimensional time is transformed by a Legendre-Gauss-Lobatto (LGL) distribution method [41] that results in placing the nodes on a non-dimensional time grid, τ_{seg}, spanning from -1 on the left segment boundary to $+1$ on the right segment boundary. Each segment contains at least three nodes to enable interpolated state and control values to be obtained at every other node. There is no maximum limit on the number of nodes; however, it must be an odd number due the mathematical formulation adopted. The number of nodes (per segment) corresponds directly to the order of the interpolating polynomial. Figure 3.1 shows a schematic of a segment using

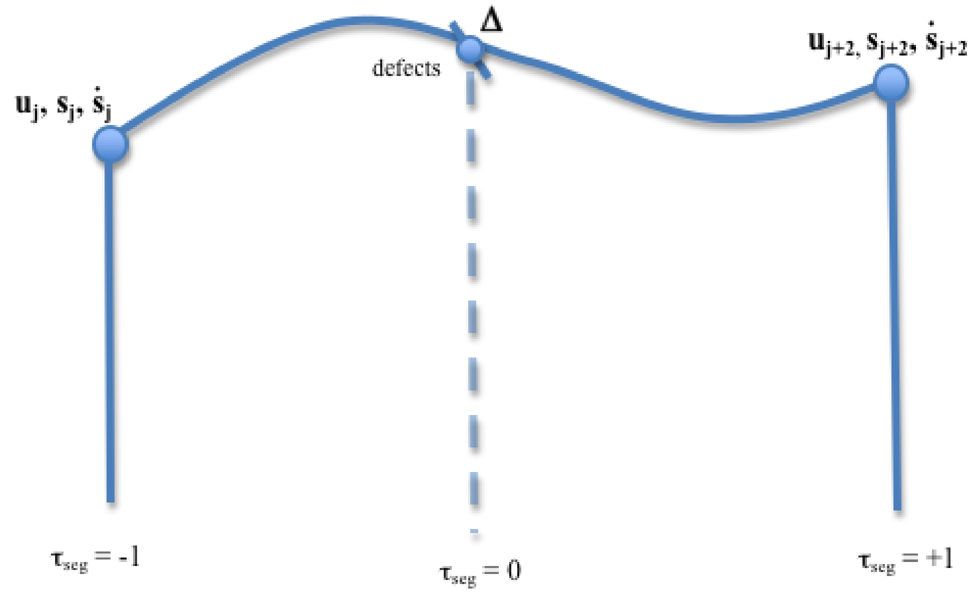

$\mathbf{u}_j, \mathbf{s}_j, \dot{\mathbf{s}}_j$
Δ
defects
$\mathbf{u}_{j+2}, \mathbf{s}_{j+2}, \dot{\mathbf{s}}_{j+2}$
$\tau_{seg} = -1$
$\tau_{seg} = 0$
$\tau_{seg} = +1$

Figure 3.1: Segment schematic

three nodes (for clarity).

Potential solutions to $\mathbf{s}_j$ and $\mathbf{u}_j$ are provided by SNOPT at every odd-numbered node. Starting with the the segment's 1st node ($\tau_{seg} = -1$ and $j = 1$), the derivatives, $\dot{\mathbf{s}}_j$, are evaluated at all the odd-numbered nodes with (3.1), (3.2), and (3.4). A Hermite polynomial interpolation [40] is performed using $\mathbf{s}_j$ and $\dot{\mathbf{s}}_j$ at all the segment's odd-numbered nodes to estimate the states at the even-numbered nodes, $\tilde{\mathbf{s}}_{j+1}$, as well as the derivatives, $\dot{\tilde{\mathbf{s}}}_{j+1}$. The interpolated states, $\tilde{\mathbf{s}}_{j+1}$, are used to obtain the true derivatives, $\dot{\mathbf{s}}_{j+1}$, at these even-numbered nodes. The differences between the derivatives estimated from a (Hermite) polynomial fit, $\dot{\tilde{\mathbf{s}}}$, and the derivatives obtained from the evaluation of the equations of motion, $\dot{\mathbf{s}}$, are called "defects", $\mathbf{\Delta}$. The defects are calculated as

$$\mathbf{\Delta}_{j+1} = \dot{\tilde{\mathbf{s}}}_{j+1} - \dot{\mathbf{s}}_{j+1},\ j^2 = 1, 3, 5, ... \tag{3.8}$$

and driven to zero (within some tolerance) by SNOPT. These defects are nonlinear

[2]segment node

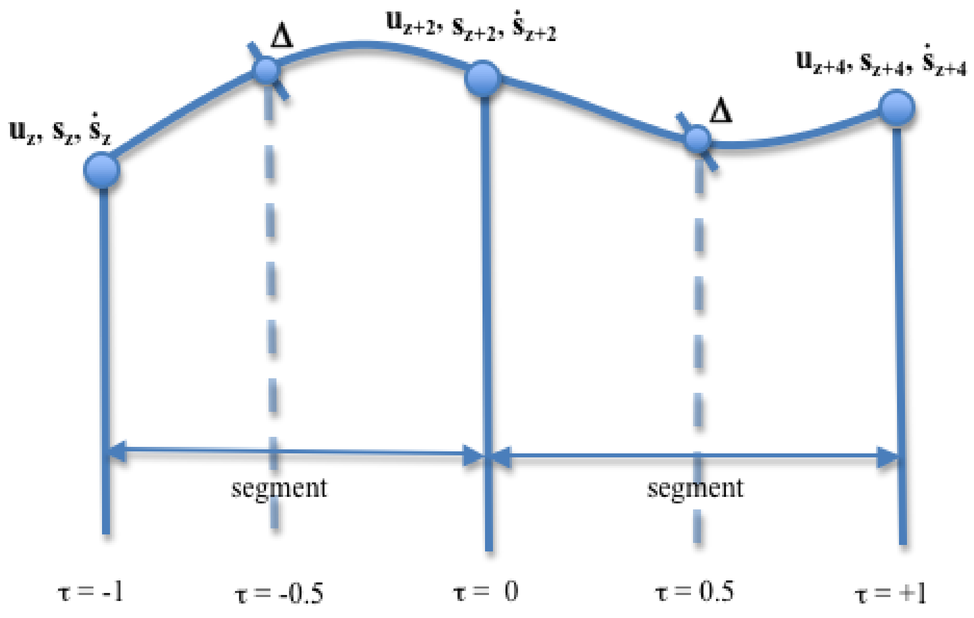

$\mathbf{u}_z, \mathbf{s}_z, \dot{\mathbf{s}}_z$
Δ
$\mathbf{u}_{z+2}, \mathbf{s}_{z+2}, \dot{\mathbf{s}}_{z+2}$
Δ
$\mathbf{u}_{z+4}, \mathbf{s}_{z+4}, \dot{\mathbf{s}}_{z+4}$
segment
segment
τ = -1
τ = -0.5
τ = 0
τ = 0.5
τ = +1

Figure 3.2: Collocation phase discretization

constraints.

The controls at the even-numbered nodes, $\mathbf{u}_{j+1}$, are prescribed by a simple linear interpolation of the controls at the adjacent odd-numbered nodes, $\mathbf{u}_j$ and $\mathbf{u}_{j+2}$. Control rates are not part of the equations of motion; therefore they are not included in the defects.

Each phase consists of at least one segment; multiple segments are equally spaced within a phase and their non-dimensional time grids, τ_{seg}, are mapped to the phase non-dimensional time grid, τ. Figure 3.2 shows a simple schematic for a collocation phase containing two segments with three nodes each.

A different subscript, z, for the phase nodes indicates different numbering than a segment. The node index for a segment is reset to 1 at the start of a new segment whereas the node index for a phase is reset to 1 at the beginning of new phase ($z = 1$ and $\tau = -1$). This node numbering scheme cascades up from the most local entity (a segment), through each phase, and continues for each vehicle until a sort of global numbering scheme is achieved for the entire mission. This global numbering scheme is represented hereafter with the subscript i (note that $i = z$ for a mission with one phase).

This discretization process results in (3.7) becoming discretized as

$$\mathbf{x} = \begin{bmatrix} \mathbf{s}_i \\ \mathbf{u}_i \\ \mathbf{t}_{0_k} \\ \mathbf{\Delta t}_k \end{bmatrix}, i^3 = 1, 3, 5, ..., \; k^4 = 1, 2, 3... \, . \tag{3.9}$$

To allow discontinuities of the states due to the need to model such events as mass jettisons, two nodes are placed at the phase boundaries. These two nodes share the same time point on the grid, but are allowed to have different state and/or control values. This feature is controlled by the user though an integer input variable named *stepState* that corresponds to the state (and/or control) variable(s) that is/are allowed to be discontinuous. For example, $stepState = 4, 5, 6$ would allow a discontinuity in all components of the velocity vector. The default is for these states and controls to be continuous across all phases.

As an aid in determining the size of $\mathbf{x}$, the number of nodes within each phase, $nNodes_{ph}$, is calculated from the number the of segments per phase, $nSeg_{ph}$, and the number of nodes per segment, $nNodes_{seg}$,

$$nNodes_{ph} = nSeg_{ph}(nNodes_{seg}) - (nSeg_{ph} - 1) \tag{3.10}$$

and used to calculate the number of odd nodes (per phase)

$$nNodes_{ph,odd} = \frac{nNodes_{ph} + 1}{2} \tag{3.11}$$

as well as the number of even nodes

[3]global node
[4]phase number

$$nNodes_{ph,even} = nNodes_{ph} - nNodes_{ph,odd}. \tag{3.12}$$

Nodes across all phases for a vehicle are summed up

$$nNodes_{veh} = \sum_{n=1}^{nPh} nNodes_{ph_n} - (nPh - 1) \tag{3.13}$$

and finally the (global) number of nodes is calculated

$$nNodes = \sum_{n=1}^{nVeh} nNodes_{veh_n} - (nVeh - 1). \tag{3.14}$$

The (global) number of odd nodes is then given by:

$$nNodes_{odd} = \frac{nNodes + 1}{2} + nRepeat \tag{3.15}$$

where *nRepeat* accounts for the extra node at phase boundaries,

$$nRepeat = nPh - 1. \tag{3.16}$$

Defects are then evaluated at all the even-numbered nodes as a function of an interpolating polynomial that uses iterated state and control values provided by SNOPT at all the odd-numbered nodes and their subsequently calculated derivatives. After exiting SNOPT, the (final) state and control values are recorded at the odd-numbered nodes. Time histories are generated by fitting a cubic-spline curve fit through these data points.

The last required array to describe in (2.15) is $\mathbf{f}(\mathbf{x})$. This array contains constraints and the objective function. Linear constraints ensure that start time of one phase is also the end time of the previous phase. These constraints are given as

$$\boldsymbol{\eta} = \mathbf{t}_{0_{k-1}} + \boldsymbol{\Delta}\mathbf{t}_{k-1} - \mathbf{t}_{0_k}, \qquad k = 2, 3, 4, \ldots \, . \tag{3.17}$$

Additional linear constraints given below are required to ensure continuity with the states, $\mathbf{s}$, and controls, $\mathbf{u}$, across the phases.

$$\boldsymbol{\zeta}_s = \mathbf{s}_k^+ - \mathbf{s}_{k-1}^-, \qquad k = 2, 3, 4, \ldots \tag{3.18}$$

$$\boldsymbol{\zeta}_u = \mathbf{u}_k^+ - \mathbf{u}_{k-1}^-, \qquad k = 2, 3, 4, \ldots \tag{3.19}$$

where $\mathbf{s}_k^+$ indicates the state values at the 1st node of phase k and $\mathbf{s}_{k-1}^-$ indicates the state values at the last node of phase $k-1$ (likewise for $\mathbf{u}_k^+$ and $\mathbf{u}_{k-1}^-$). The linear constraint array then becomes

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\zeta}_s \\ \boldsymbol{\zeta}_u \end{bmatrix}. \tag{3.20}$$

The nonlinear constraints consist of initial phase constraints, $\boldsymbol{\chi(t_{0_k})}$, final phase constraints, $\boldsymbol{\Psi(t_{f_k})}$, the defects, $\boldsymbol{\Delta_{i+1}}$, and path constraints, $\boldsymbol{c}$. Path constraints differ from the other nonlinear constraints in that they are enforced (evaluated) at the even-numbered nodes as well ($\tilde{\mathbf{c}} = \textit{all nodes}$). An array containing all of the nonlinear constraints becomes

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\chi(t_{0_k})} \\ \boldsymbol{\Psi(t_{f_k})} \\ \tilde{\boldsymbol{c}} \\ \boldsymbol{\Delta_{i+1}} \end{bmatrix}, i = 1, 3, 5, \ldots, \; k = 1, 2, 3 \ldots \, . \tag{3.21}$$

The 2nd array element in (2.15) in now completely defined as

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} J \\ \boldsymbol{\delta} \\ \boldsymbol{\xi} \end{bmatrix}. \tag{3.22}$$

The last array element in (2.15), the 1st order partial derivatives, can also be calculated and supplied to SNOPT. A finite-differencing technique is utilized in SNOPT for any or all missing gradients. To save software development time, $\mathbf{g}(\mathbf{x})$, is not calculated by MASTIF; it is intended to be a future implementation task.

To take full advantage of the direct method, all the nonlinear constraints and the objective function are character string inputs that are internally evaluated by an expression parser in MASTIF. This way, different trajectory optimization problems can be performed without having to modify source code. Tables 3.1 and 3.2 list some key vehicle and phase user inputs.

Table 3.1: Vehicle user inputs

initial states	initial attitude
fuel tanks	mass properties
engine	hardware elements (e.g. ETS)

Table 3.2: Phase user inputs

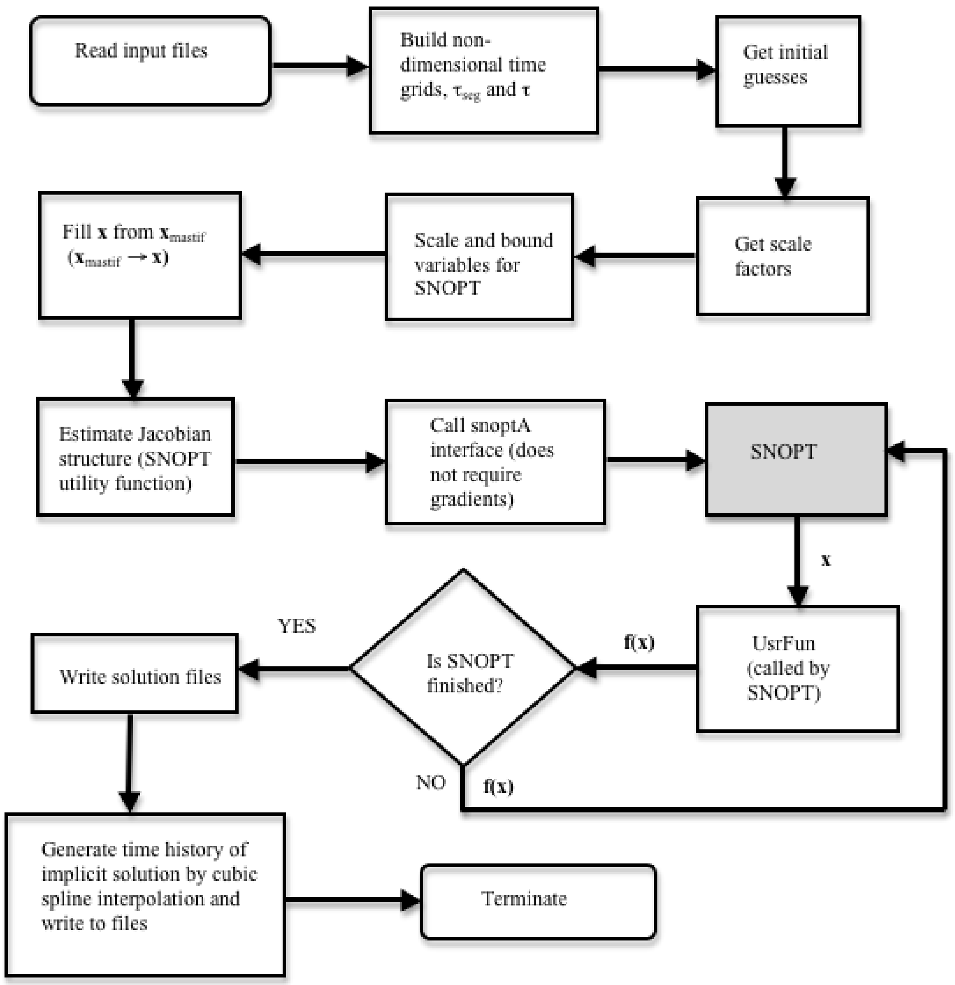
$nSeg_{ph}$	phase time duration (Δt)
$nNodes_{seg}$	$stepStates$
controlSet, $\mathbf{u}$, (e.g. LVLH Euler angles)	constraints ($\boldsymbol{\chi}$, $\boldsymbol{\Psi}$, $\boldsymbol{c}$)
method (collocation or Runge-Kutta)	integrator (e.g. Runge-Kutta 7/8)
phase start time (t_0)	integration step size

If the user selects a Runge-Kutta method instead of collocation, $\dot{\mathbf{s}}_{\mathbf{i}}$ is explicitly integrated from the phase k start time, t_{0_k}, to the end of the phase, t_{f_k}, to determine the state values. All the constraint equations previously developed remain unchanged.

Scaling is critically important for NLP problems. Scaling ensures that all the variables are roughly the same order of magnitude. In MASTIF, scaling of the constraints and the objective function is easily handled through user inputs. Initial values for position and velocity are scaled by the absolute value of the largest component in each (3-D) vector. The other variables in the independent variable array, $\mathbf{x}$, utilize a similar strategy. These scale factors are also used to scale the defects, $\boldsymbol{\Delta}$. As the optimization process proceeds and the initial guesses are updated, these scale factors are also updated to ensure compatibility with the initial guesses because significant changes in the values of the initial guesses could have occurred. Updating the scale factors also results in less iterations to converge. A high-level algorithm describing the optimization process is shown in Figure 3.3 and Figure 3.4 shows a flow chart of the UsrFun subroutine. This subroutine is where data is exchanged between SNOPT and MASTIF.

3.2 SNOPT

SNOPT is a sequential quadratic programming (SQP) method used to solve constrained nonlinear optimization problems. It is written in FORTRAN 77. It is a commercial solver; licensing and pricing information can be found at [42]. For detailed information the reader is referred to [3]. SNOPT accepts as input some or all of the 1^{st} order partial derivatives of $\mathbf{f}(\mathbf{x})$ which are assumed to be smooth and continuous. SNOPT uses finite-differencing to estimate the derivatives that are not provided. There are various input parameters that the user has control over such as optimality and feasibility tolerances and detailed output summaries. SNOPT satisfies the 1^{st} necessary conditions for the optimality and feasibility tolerances. MASTIF uses SNOPT version 7.2.4 (version 7's user's guide is dated February 12, 2006). Because it is a sparse solver, the structure of the Jacobian must be determined once prior to solving the problem. It is preferable that this structure is determined by the program using SNOPT, but SNOPT will determine this structure if requested.



Read input files
Build non-dimensional time grids, τ_{seg} and τ
Get initial guesses
Get scale factors
Scale and bound variables for SNOPT
Fill **x** from $\mathbf{x}_{mastif}$ ($\mathbf{x}_{mastif} \rightarrow \mathbf{x}$)
Estimate Jacobian structure (SNOPT utility function)
Call snoptA interface (does not require gradients)
SNOPT
x
UsrFun (called by SNOPT)
f(x)
Is SNOPT finished?
YES
NO
f(x)
Write solution files
Generate time history of implicit solution by cubic spline interpolation and write to files
Terminate

Figure 3.3: MASTIF optimization algorithm

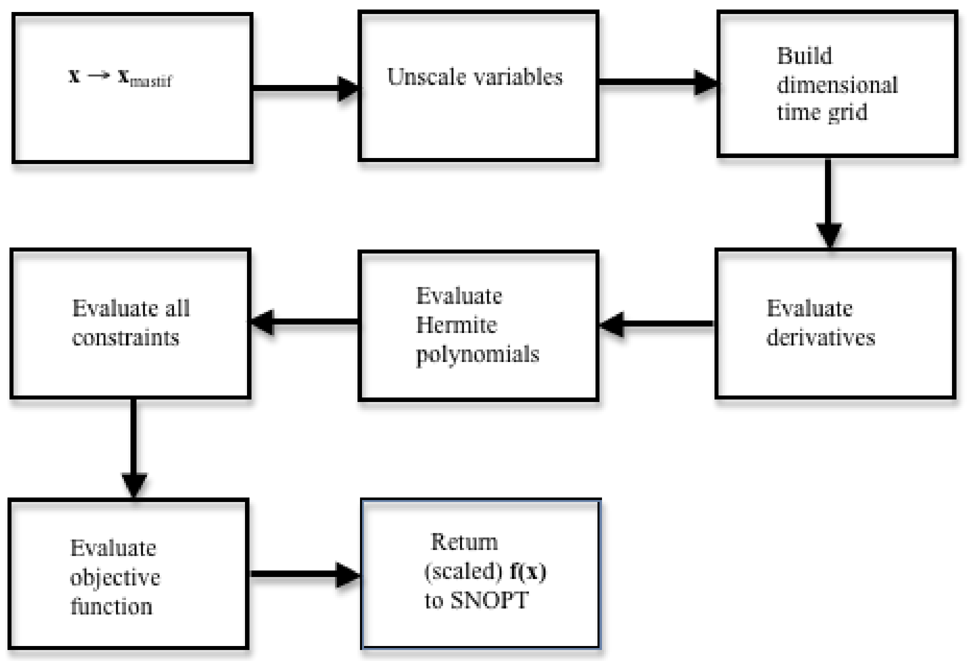

$\mathbf{x} \rightarrow \mathbf{x}_{mastif}$
Unscale variables
Build dimensional time grid
Evaluate derivatives
Evaluate Hermite polynomials
Evaluate all constraints
Evaluate objective function
Return (scaled) $\mathbf{f}(\mathbf{x})$ to SNOPT

Figure 3.4: UsrFun subroutine

3.3 Solution Procedure

A generic, iterative, step-by-step procedure to solve a constrained NLP problem with MASTIF is shown in Figure 3.5. The primary reason that iterations are required is for solution accuracy. A procedure known as grid or mesh refinement [32] ensures that the quantity and/or placement of nodes represents the solution accurately.

A current shortcoming in MASTIF is the absence of a grid refinement algorithm; however, the solution accuracy can still be assessed and even improved. This improvement is provided by the ability to add more nodes. Automated control over nodal placement (i.e. a grid refinement algorithm) is preferred.

Solution accuracy is assessed by taking the optimal control time histories, $\mathbf{u}^*(t)$, (captured at all the odd nodes), and the optimal initial vehicle states, $\mathbf{s}^*(t_0)$, (captured at the first node of the first phase), from the optimization mode and then explicitely integrating

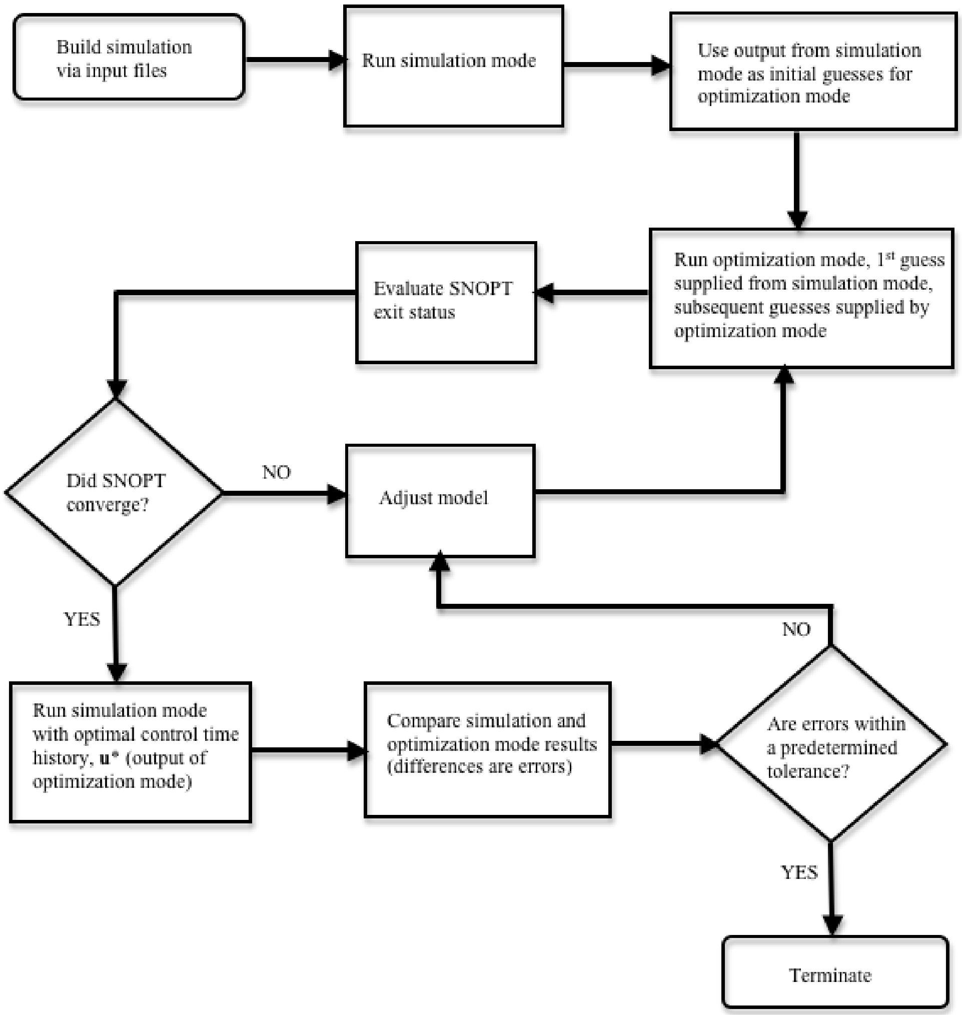

Build simulation via input files
Run simulation mode
Use output from simulation mode as initial guesses for optimization mode
Run optimization mode, 1st guess supplied from simulation mode, subsequent guesses supplied by optimization mode
Evaluate SNOPT exit status
Did SNOPT converge?
NO
Adjust model
YES
NO
Run simulation mode with optimal control time history, u* (output of optimization mode)
Compare simulation and optimization mode results (differences are errors)
Are errors within a predetermined tolerance?
YES
Terminate

Figure 3.5: MASTIF solution procedure

the equations of motion using $\mathbf{u}^*(t)$ and $\mathbf{s}^*(t_0)$ from the initial start time, t_0, to the final time, t_f. The resulting trajectory is hereafter referred to as the explicit solution whereas the solution obtained from the optimization is hereafter referred to as the implicit solution because of the two collocation phases that do not require integration of the equations of motion. Any differences between the two solutions are errors with the explicitly integrated trajectory regarded as the truth model. In MASTIF, these errors can be significantly reduced by adding more nodes. With grid refinement the errors could be eliminated entirely within some specified tolerance. Because the last phase of the TLI problem is a Runge-Kutta phase, the errors are due to the implicit solutions in the first two phases. This process of accessing the solution accuracy revealed that (not surprisingly) the Lunar arrival constraints had the largest errors.

Table 3.3: Implicit vs. explicit absolute Lunar arrival errors

$nNodes_{odd}$	alt_p km (ft)	$inc_{wrt☾}$ degree	$\dot{r}_{wrt☾}$ km/s (ft/s)
21	571.179 (1,873,947)	7.085	1.157 (3,796)
35	20.707 (67,936)	1.65	0.183 (600)
61	1.521 (4,990)	0.00016	0.00055 (2)

Table 3.3 shows how these errors are reduced with additional nodes ($nNodes_{odd}$). There will be a tradeoff between accuracy and execution time. For the purpose of this research, the errors from the solution using 61 odd nodes were deemed acceptable and will be used throughout the rest of this thesis.

Chapter 4

Numerical Implementation and Results

4.1 Open-loop Solution

The solution to the constrained NLP problem, (2.15), results in the optimal control, $\mathbf{u}^*$, for the powered portion of flight. These controls ensure that mass is maximized, while meeting all initial and terminal constraints. The LVLH Euler ϕ, θ, and ψ time histories are shown as cubic spline curve fits through the data points at the odd nodes for the powered portion of flight in Figures 4.1, 4.2 and 4.3, respectively.

Because nearly all the motion is within the Earth-to-Moon transfer plane, out-of-plane motion is minimal; therefore, ϕ and ψ angles are small ($< 1^\circ$) and are shown for completeness.

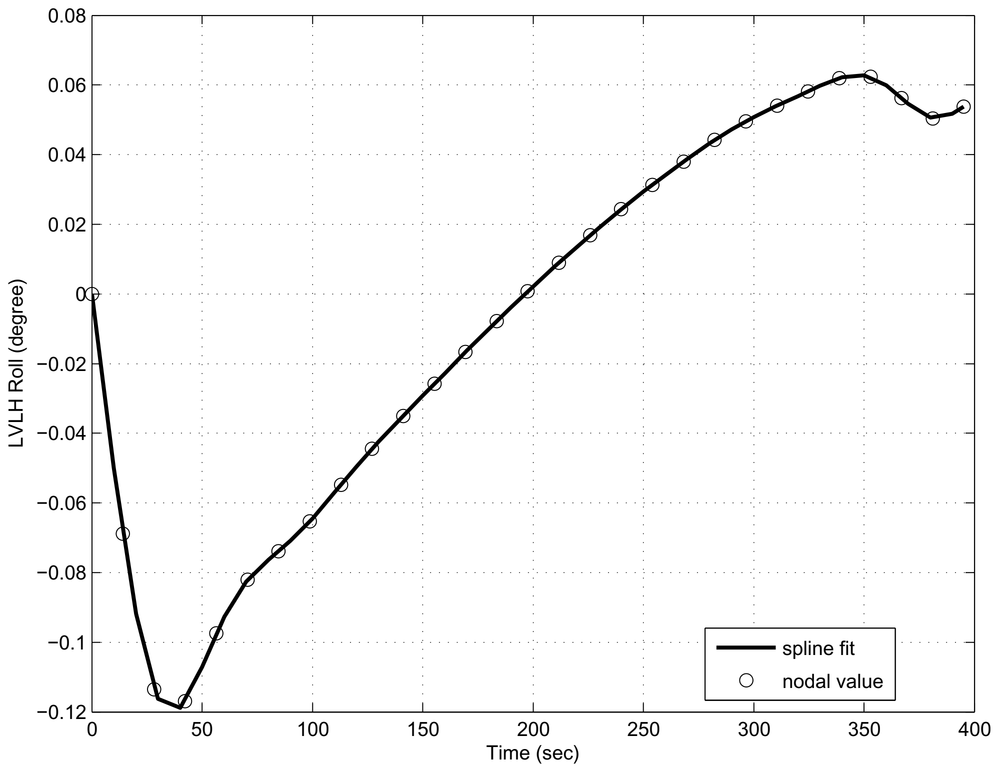


Figure 4.1: Optimal ϕ time history

The mass time history is shown in Figure 4.4. Mass decreases linearly during the main engine burn until MECO. It remains constant during the second phase of flight. One hour after the start of the main engine burn, two propellant tanks and the ETS are jettisoned which result in the mass discontinuity in the figure. This discontinuity is enabled by the double nodes at phase boundaries as discussed in Chapter 3 and shown schematically in Chapter 2. The rest of the transfer is unpowered; therefore, mass remains constant.

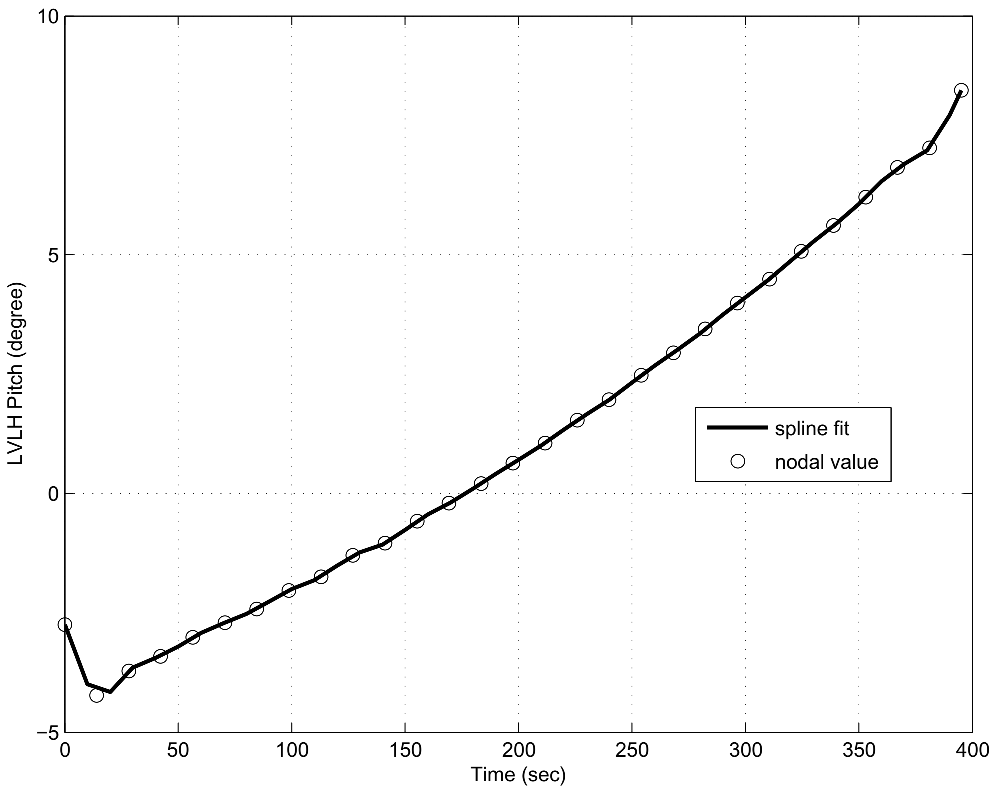


Figure 4.2: Optimal θ time history

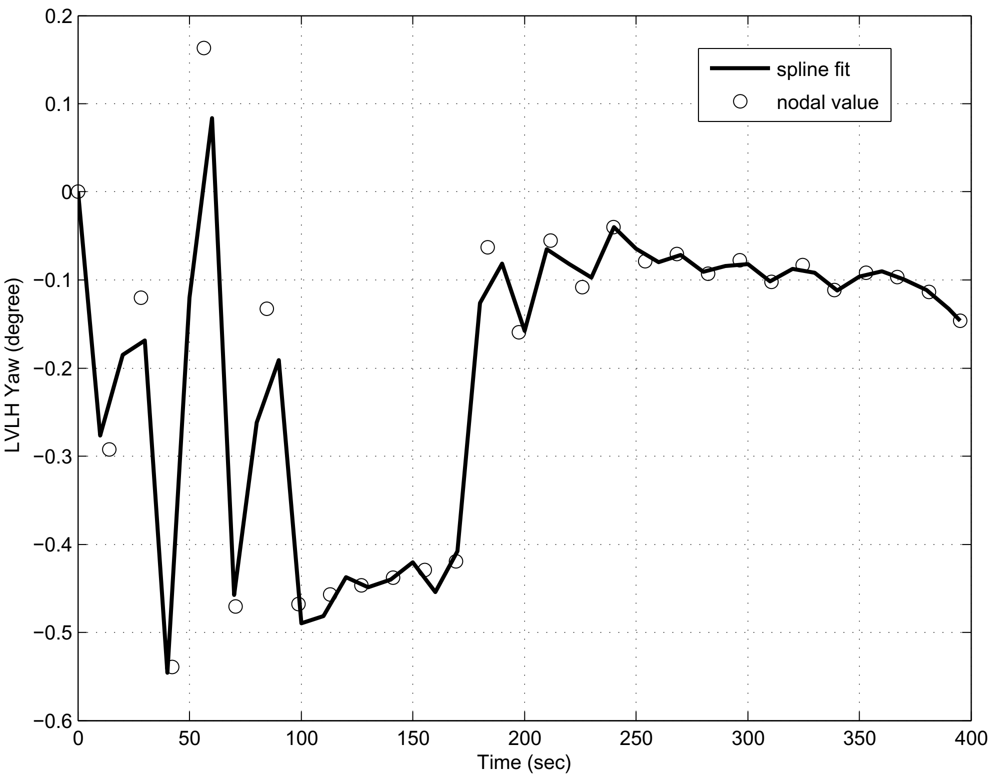


Figure 4.3: Optimal ψ time history

The pericynthion altitude, Moon-centered inclination, and Moon-centered radial velocity terminal constraint time histories are shown in Figures 4.5, 4.6, and 4.7, respectively. As shown above, altitude monotonically decreases (after 3 hours from main engine ignition) to the targeted altitude and remains nearly constant for the last ten hours.

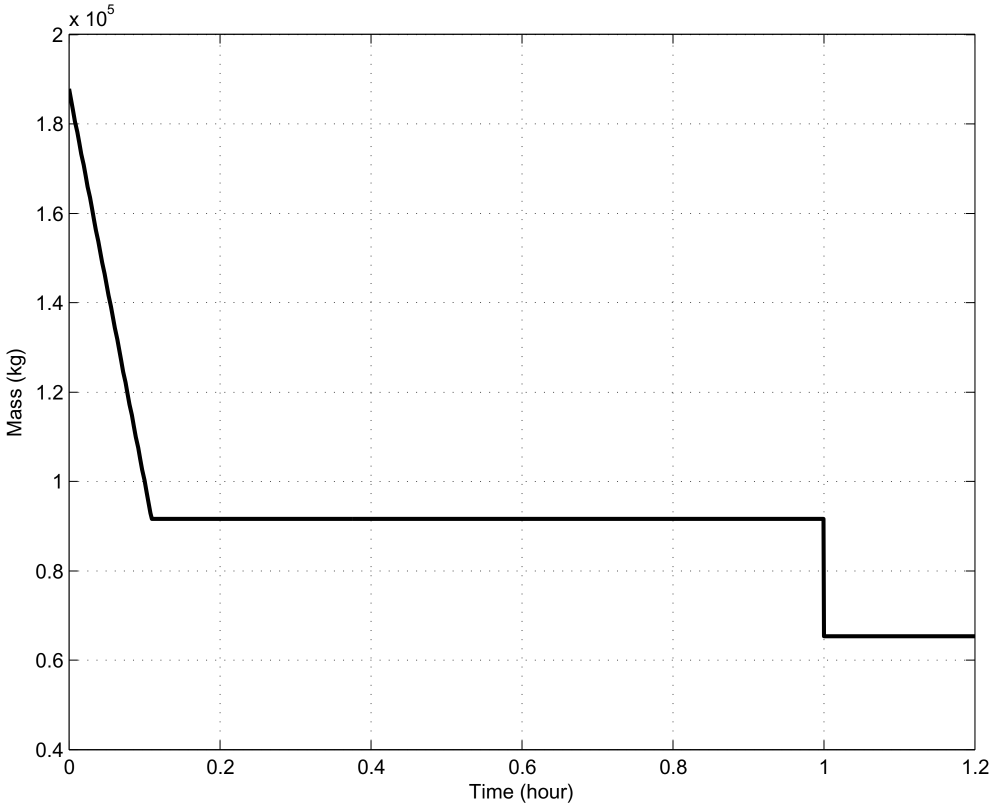


Figure 4.4: Mass time history (mass is constant after 1 hour)

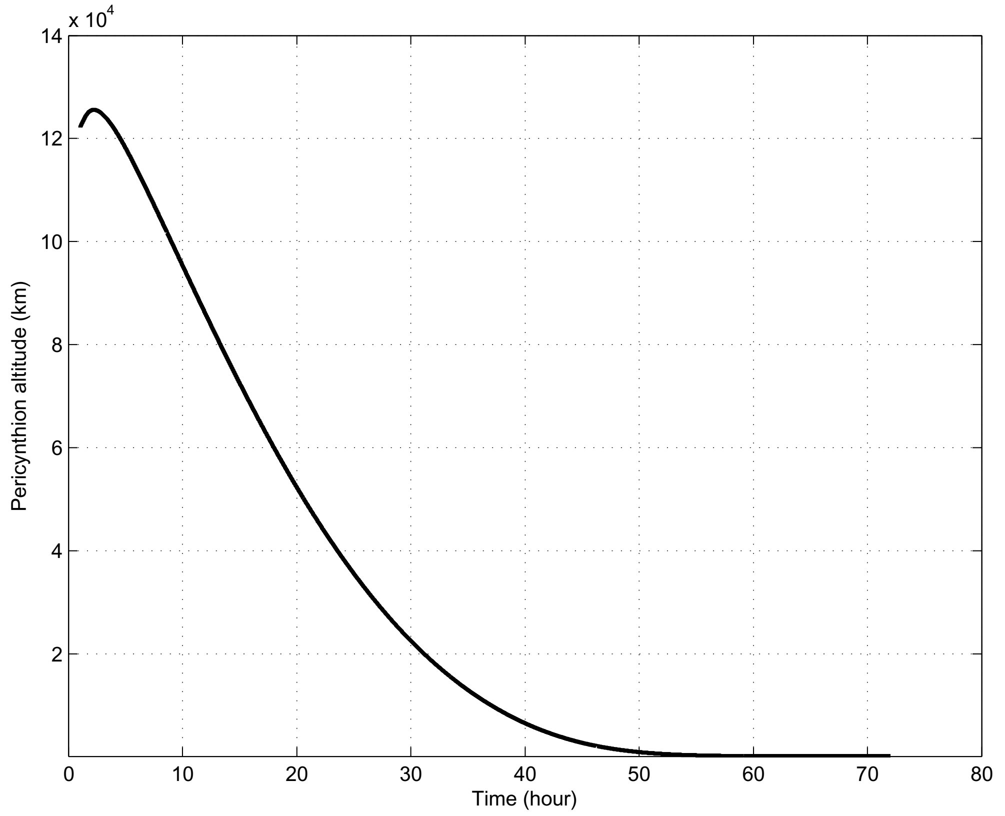


Figure 4.5: Instantaneous pericynthion altitude

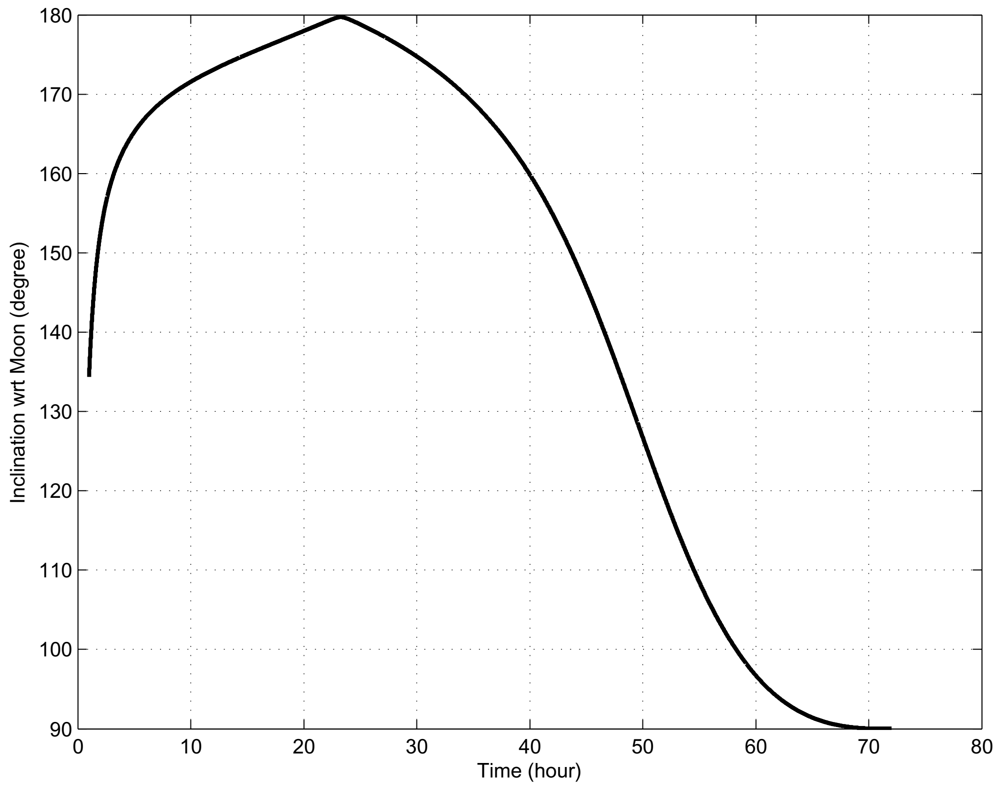


Figure 4.6: Moon-centered inclination

The inclination time history, above, shows inclination increasing by 45° to 180° at nearly 23 hours after main engine ignition. After 23 hours, inclination decreases and eventually reduces to the desired 90°.

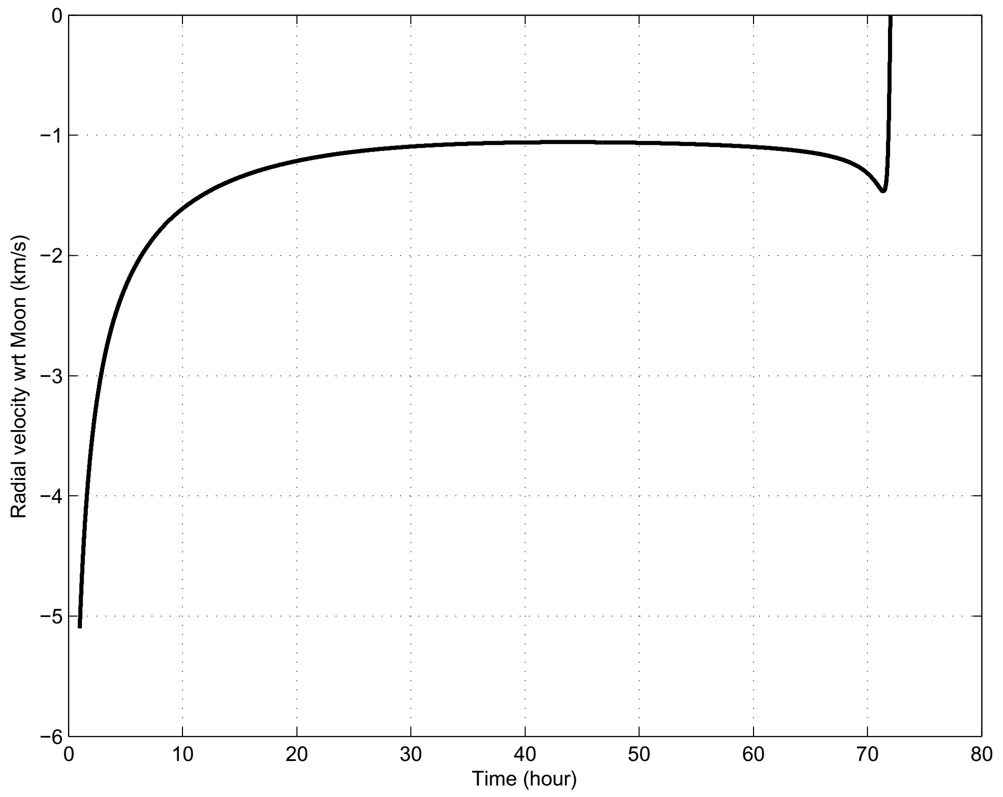


Figure 4.7: Moon-centered radial velocity

The effect of the Moon's gravitational pull on the spacecraft accounts for the fast rate of change in the radial velocity during the last hour of flight as shown above.

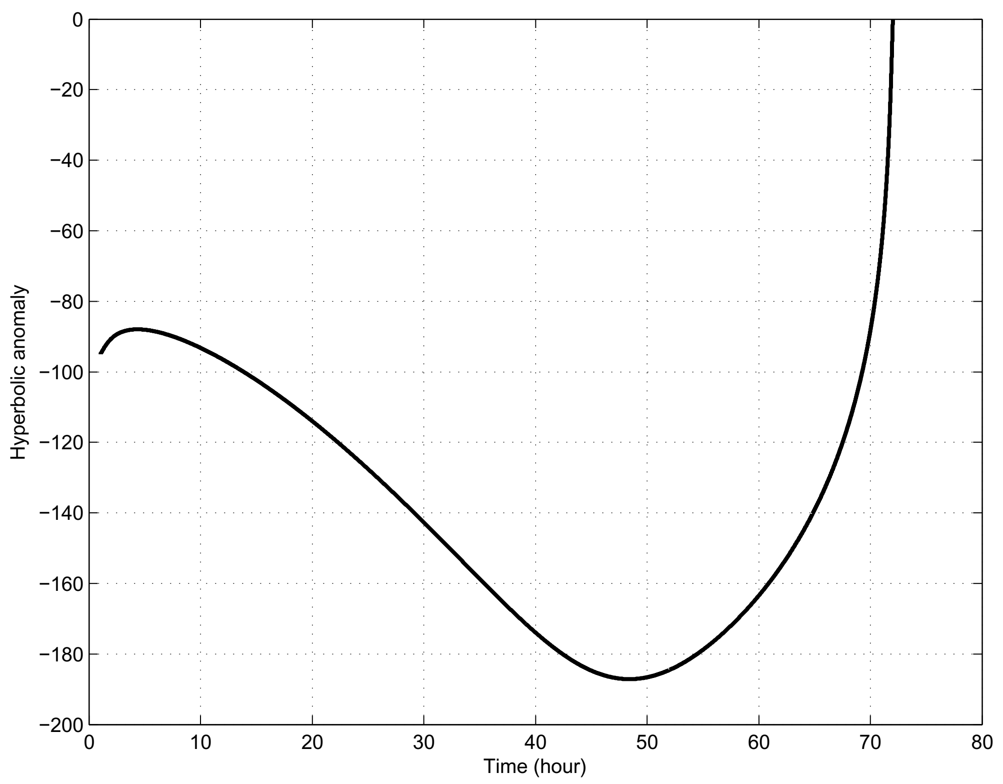


Figure 4.8: Moon-centered hyperbolic anomaly

Figure 4.8, above, shows the hyperbolic anomaly time history. It shows that the spacecraft arrives at an anomaly of zero verifying that its closest approach is at the desired pericynthion altitude.

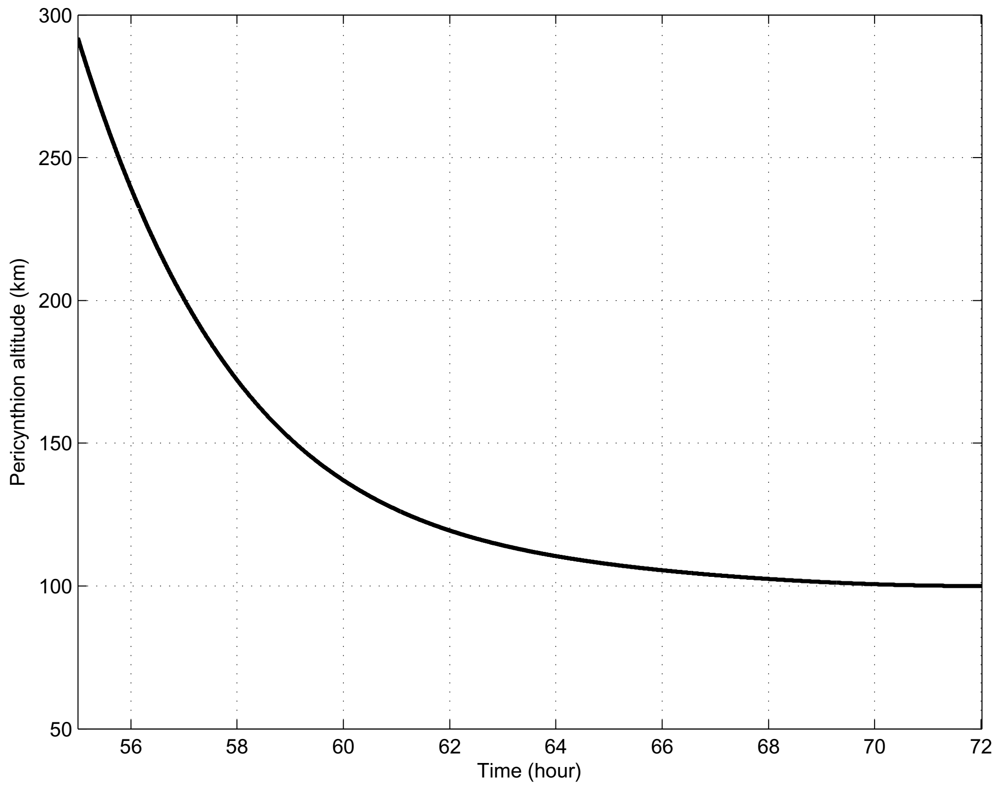


Figure 4.9: Pericynthion altitude during the final 17 hours

Figure 4.9 shows the pericynthion altitude changes by less than 200 km during the final 17 hours.

The numerical values for initial and final constraints are listed below in Table 4.1.

Table 4.1: Constraint value precision (6 decimals)

$\chi_1(t_{0_1})$ km	$\chi_2(t_{0_1})$ degree	$\chi_3(t_{0_1})$	$\Psi_1(t_{f_3})$ km	$\Psi_2(t_{f_3})$ degree	$\Psi_3(t_{f_3})$ km/s
6,563.340000	29.000000	2.0e-6	100.000984	90.000003	0.000000

All constraints are satisfied within acceptable tolerances.

Table 4.2 shows the TLI burn lasts for nearly 400 seconds and lists the amount of fuel and oxidizer remaining in the tanks at MECO. The TLI burn consumes over 99% of the available propellent. The optimized values of the three free orbital angles (Ω, ω, and

Table 4.2: Burn time and propellant remaining (nearest kg/lbm)

$t_{b_{eng}}$ sec	m_{LOX} kg (lb_m)	m_{LH2} kg (lb_m)
395.260595	387 (853)	70 (155)

ν) shown in Table 4.3 show that the burn starts at nearly 18° ($\sim$ 1 minute) before perigee passage ($\nu = 360°$)

Table 4.3: Optimal LEO departure angles

Ω_0 degree	ω_0 degree	ν_0 degree
100.1550	226.9605	341.8067

The actual ΔV required to perform the TLI is compared to the ideal ΔV in Table 4.4. The difference is due to the gravity loss associated with having a non-zero flight path angle for the finite burn and a zero flight path angle for the instantaneously modeled burn (the ideal velocity change).

Table 4.4: ΔV

ΔV_{ideal} [43] m/s (ft/s)	ΔV_{actual} m/s (ft/s)	ΔV_{gLoss} m/s (ft/s)
3,142.843 (10,311)	3,157.415 (10,359)	14.572 (48)

Because all three bodies (Earth, Moon, and spacecraft) are in motion, ensuring the correct phasing is important for this problem. This is the role of the initial guess. In order for this type of problem to have any hope of converging, the Earth-Moon transfer opportunities must be identified (this could be automated eventually by solving Lambert's Equation [44], but is not currently). Figure 4.10 shows the alignment of the Earth and Moon at epoch and the trajectory of the spacecraft. There is a difference between the Moon's osculating (instantaneous) orbit and mean orbit; the Moon's location at epoch is on the osculating orbit. The departure trajectory as viewed near Earth is shown in Figure 4.11 superimposed over the Moon's orbital path. The spacecraft's North to South Lunar flyby is shown in Figure 4.12 as well as the orbital path of the Moon. The trajectory as viewed from the vicinity of the Moon near arrival is shown in Figure 4.13.

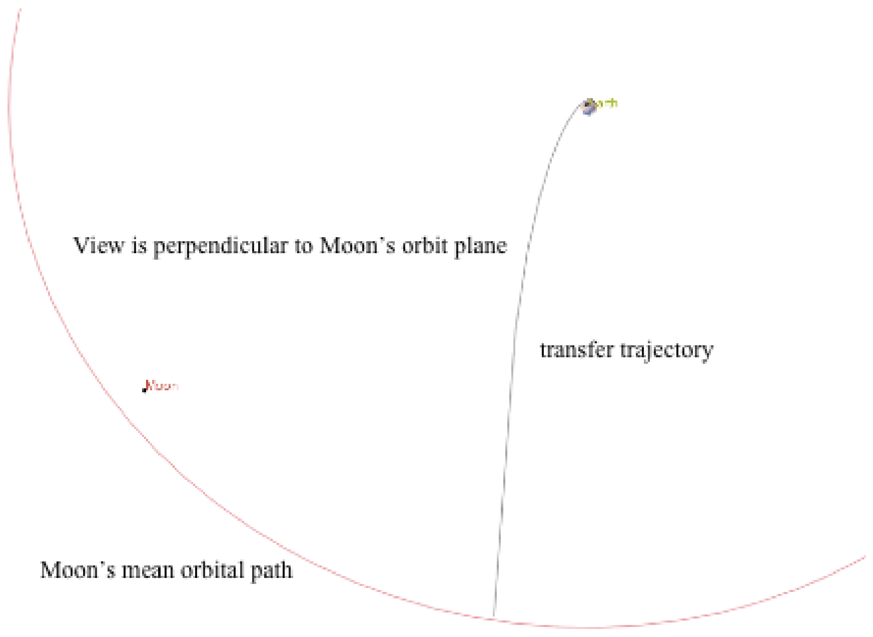


Figure 4.10: Earth and Moon alignment at epoch (counterclockwise motion)

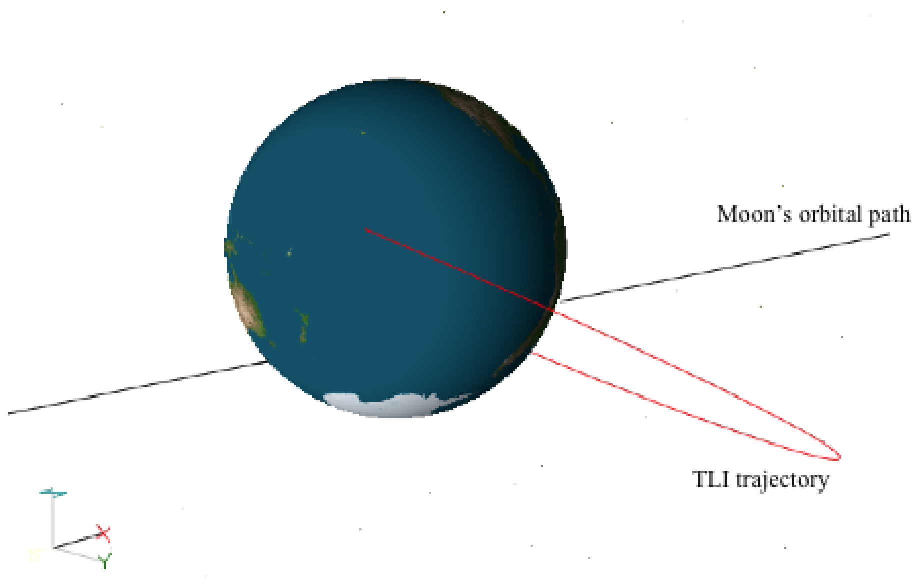


Figure 4.11: Earth departure trajectory (J2000 coordinate axes shown on left)

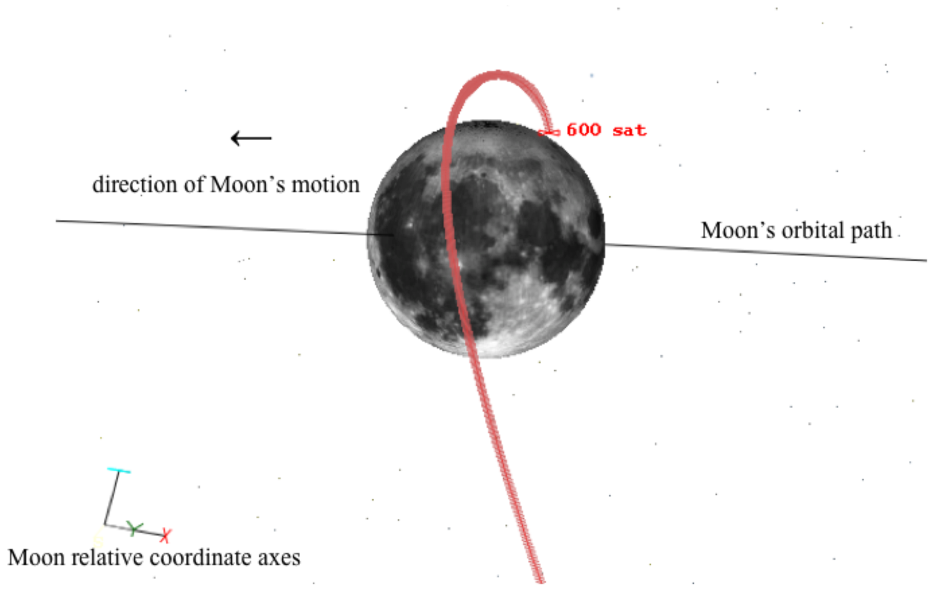


Figure 4.12: Lunar approach trajectory (600 sat is name of spacecraft in visualization tool)

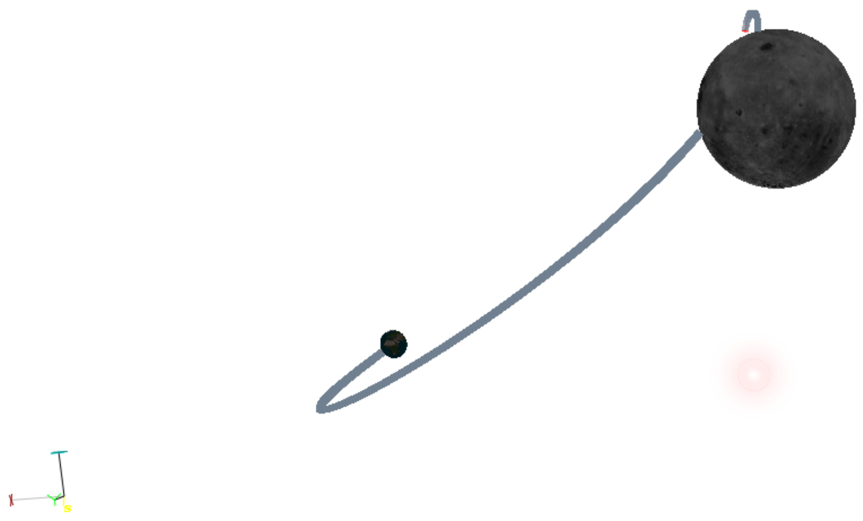


Figure 4.13: Transfer trajectory viewed from the vicinity of the Moon at arrival (MCME coordinate axes shown on left)

4.2 Closed-loop Solution

4.2.1 Powered Explicit Guidance

The targeted end conditions for PEG are the MECO values of flight path angle, radius and velocity magnitudes, inclination, and right ascension, which are obtained from MASTIF's optimization mode, and are listed in Table 4.5. Perfect navigation state knowledge is assumed. Guidance is called at a frequency of $1H_z$. Numerical integration of the equations of motion is performed with a second order Runga-Kutta with a fixed step-size of 0.01 seconds.

Table 4.5: PEG target MECO parameters (SI units and angles)

R	km (ft)	6,730.546482 (22,081,845)
V	km/s (ft/s)	10.796449 (35,421)
γ	degree	8.018735
inc	degree	28.938730
Ω	degree	100.042378

4.3 Open-loop and Closed-loop Comparisions

The open-loop and closed-loop LVLH Euler angle θ time histories during the TLI burn are shown in Figure 4.14. After five iterations, PEG converges to the same initial θ angle as the optimized open-loop solution. At each guidance cycle, the converged solution from the previous cycle is used to start the iteration. The simulation proceeds in this manner until the guidance commanded engine shutdown, which signals that the vehicle has arrived at the MECO targets. Differences of 1.5° are seen at the end of the burn.

An open-loop and closed-loop performance comparison, as characterized by the burn time required (and the resulting remaining propellant), is shown in Table 4.6. The open-loop burn time is rounded to two decimal places because the finest resolution achievable for the

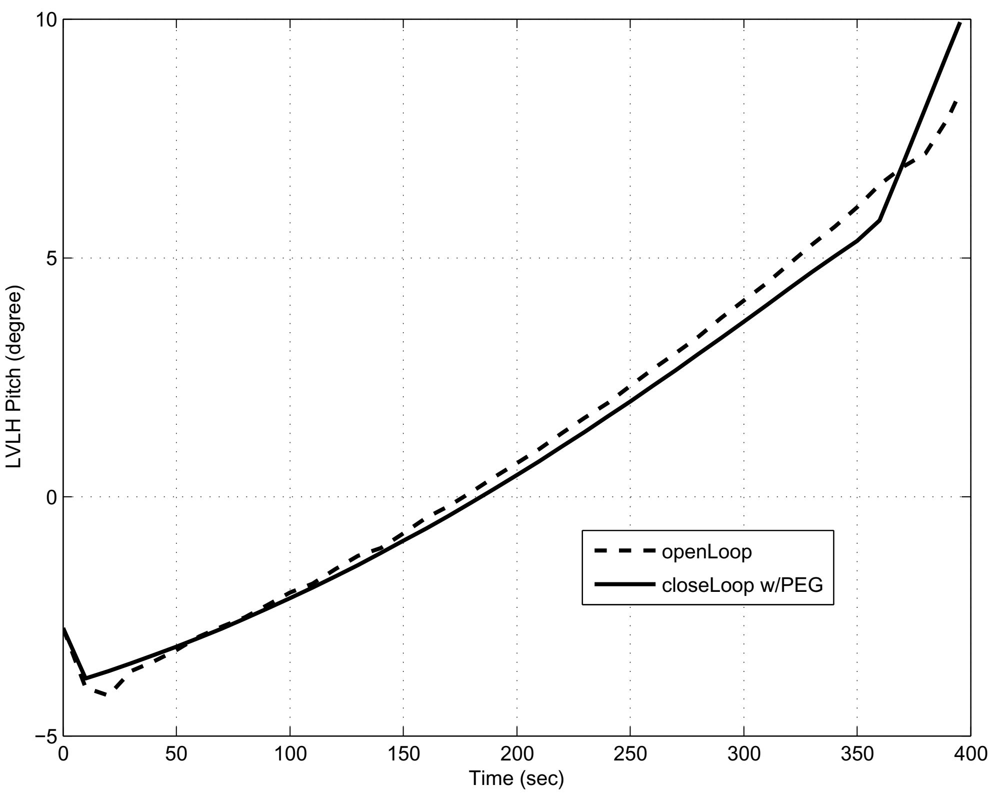


Figure 4.14: θ angle comparison

Table 4.6: Open-loop and closed-loop MECO time and LOX/LH2 remaining

method	$t_{b_{eng}}$	m_{LOX}	m_{LH2}
	sec	kg (lb_m)	kg (lb_m)
open-loop	395.26	387 (853)	70 (155)
closed-loop	395.29	381 (840)	69 (153)

closed-loop burn time is $100H_z$ due to the selected integration frequency. The performance of PEG compares favorably with the open-loop solution.

In order for the closed-loop solution to have the exact Lunar arrival conditions, the errors at MECO must be zero which they are not, as shown in Table 4.7. These errors are on the order seen during the ARES I trajectory design cycles. Errors are present because closed-loop systems use approximate formulations to derive closed-form solutions.

Different PEG input parameters can be adjusted to minimize the errors at MECO, but result in slightly different θ profiles. No adjustments could be made to null all the

Table 4.7: Closed-loop MECO errors

R	km (ft) [%]	1.318 (4,326) [0.02]
V	km/s (ft/s) [%]	0.00132 (4) [0.01]
γ	degree [%]	0.044 [0.5]
inc	degree [%]	0.00015 [0.0005]
Ω	degree [%]	0.00044 [0.0004]

MECO errors. Table 4.8 shows that MECO errors translate to Lunar arrival errors (for the closed-loop system).

Table 4.8: Lunar arrival conditions

parameter	open-loop	closed-loop
alt_p (km)	100	325
$inc_{wrt☾}$ (degree)	90	96
$\dot{r}_{wrt☾}$ (km/s)	0	-0.4

The error in Pericynthion altitude is 225 km and the inclination error is 6°. To ensure that the spacecraft arrives at the desired Lunar location, MCC(s) would be needed during the transfer and/or MECO target parameters could be updated (instead of fixed) during the burn. Both a targeting scheme and MCC modeling are potential future tasks.

4.4 Optimality Verification Procedure for Open-loop Solution

In order to verify that the objective function is at a local minimum, the 1^{st} order necessary and 2^{nd} order sufficient conditions must be satisfied. Verification of 2^{nd} order optimality conditions for NLP is an on-going area of research [45]. SNOPT satisfies the 1^{st} order necessary condition of the NLP problem within the feasibility and optimality tolerances, but 2^{nd} order sufficient conditions cannot be verified by SNOPT because it does not make use of 2^{nd} order derivatives. Although it is rare for 1^{st} order conditions to be satisfied, but not 2^{nd} order [3], the solution to the TLI problem cannot not be claimed as optimal until the 2^{nd} order conditions are verified. This verification process may be able to be performed numerically as a post-processing functionality. One potential method for performing this verification have been identified in the literature. The method consists of evaluating the Hessian of the Lagrangian function from the NLP problem and verifying that its eigenvalues are positive [46]. There are likely to be issues with this method because 2^{nd} order derivatives can become unstable. At the time of this thesis writing, no post-processing optimality verification has been implemented in MASTIF. The author is unaware of any trajectory optimization program (using NLP) mentioned in Chapter 1 that performs this test including the award-winning OTIS [47].

Chapter 5

Conclusions and Future Work

5.1 Thesis Summary

Trajectory optimization software utilizing a direct optimization method using collocation was developed and implemented as a new mode in a low-to-high fidelity 3-DOF/6-DOF vehicle simulation tool named MASTIF. This newly developed optimization mode is used to solve a human-class TLI trajectory problem by transcribing the trajectory optimization problem to a constrained NLP problem. To the best of this researcher's knowledge, this thesis is the first published work using a direct method to solve this human-class translunar injection problem.

This capabilities provided by the software are novel because it can be utilized as a trajectory optimization program and a vehicle simulation tool to design and test GN&C algorithms. The 6-DOF infrastructure can readily be leveraged to extend the optimization mode to 6-DOF. Currently, this researcher is aware of only one program (POST) used within NASA with that functionality. To demonstrate MASTIF's versatility, a closed-loop guidance scheme with heritage, PEG, is assessed using the results, in the form of fixed MECO target parameters and optimal initial states, from the optimization (open-loop) solution. Results show the efficacy of the discretization scheme and, additionally, that using fixed MECO target parameters would be acceptable for the first design cycle analysis, but that one or more MCCs would be required to ensure the spacecraft's desired Lunar arrival conditions are satisfied.

5.2 Future Direction

There are three important algorithms that remain to be implemented. First, an estimate of the 1^{st} order partial derivatives, using finite-differencing techniques, needs to be calculated by MASTIF and supplied to SNOPT. This is recommended in the SNOPT user's guide. This modification will result in faster execution times and more control of over the finite-differencing methods. Second, a grid refinement algorithm needs to be implemented to allow better placement of the nodes. The level of effort required to implement algorithms for 1^{st} order derivative estimations and grid refinement are considered minor relative to the effort required to develop the entire optimization mode itself. Lastly, an algorithm to numerically verify the 2^{nd} order sufficient conditions for the TLI problem needs further research to develop.

A larger effort would be to extend the optimization to 6-DOF. This is a relatively new area of research. A dual time-scale discretization scheme for collocation [48] has recently been designed and used successfully for a proof-of-concept 6-DOF Earth reentry trajectory optimization problem [49]. The motivation for this design is to reduce the size of the NLP problem which becomes much larger with 6-DOF.

Finally, MASTIF is a software program that is intended to be freely distributed. Numerous manuals such as a user's manual, installation instructions, etc. are being written. However, before MASTIF is released it must be in compliance with NASA's software development policies [50] which is an on-going effort.

Bibliography

[1] Columbia Accident Investigation Board, "The Columbia Accident Investigation Board Report," Website; accessed August 2010, 2003, `http://caib.nasa.gov/`.

[2] NASA, "The Vision for Space Exploration," Website; accessed August 2010, Feb 2004, `http://www.nasa.gov/pdf/55583main_vision_space_exploration2.pdf`.

[3] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm For Large Scale Constrained Optimization," *Society for Industrial and Applied Mathematics*, Vol. 47, No. 1, 2005, pp. 99–131.

[4] Hargraves, C. R. and Paris, S. W., "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 4, 1987, pp. 338–342.

[5] Riehl, J. P., Paris, S. W., and Sjauw, W. K., "Comparison of Implicit Integration Methods for Solving Aerospace Trajectory Optimization Problems," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Keystone, Colorado, Aug. 21–24, 2006.

[6] Sauer, C. G., "Solar Electric Performance for Medlite and Delta Class Planetary Missions," *AAS/AIAA Astrodynamics Specialist Conference and Exhibit*, Sun Valley, Idaho, Aug. 4–7, 1997.

[7] Sauer, C. G., "MIDAS: Mission Design and Analysis Software for the Optimization of Ballistic Interplanetary Trajectories," *Journal of the Astronautical Sciences*, Vol. 37, No. 3, 1989, pp. 251–259.

[8] Balkanyi, L. R. and Spurlock, O. F., "DUKSUP–A High Thrust Trajectory Optimization Code," *AIAA/AHS/ASEE Aerospace Design Conference*, Washington, DC, 1993.

[9] Striepe, S. A. and et al, "Program To Optimize Simulated Trajectories (POST II)," Tech. rep., NASA Langely Research Center and Lockheed Martin Corporation.

[10] Sims, J. A., Finlayson, P. A., Rinderle, E. A., Vavrina, M. A., and Kowalkowski, T. D., "Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Keystone, Colorado, Aug. 21–24, 2006.

[11] Whiffen, G. J., “Mystic: Implementation of the Static Dynamic Optimal Control Algorithm for High-Fidelity Low-Thrust Trajectory Design,” *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Keystone, Colorado, Aug. 21–24, 2006.

[12] Williams, J., Senent, J. S., Ocampo, C., Mathur, R., and Davis, E. C., “Overview and Software Architecture of the Copernicus Trajectory Design and Optimization System,” *4th International Conference on Astrodynamics Tools and Techniques*, Madrid, Spain, May 3–6, 2010.

[13] Rayman, M. D., Fraschetti, T. C., Raymond, C. A., and Russell, C. T., “Dawn: A Mission in Development for Exploration of Main Belt Astreroids Vesta and Ceres,” *Acta Astronautica*, Vol. 58, 2006, pp. 605–616.

[14] Paris, S. W., Riehl, J. P., and Sjauw, W. K., “Enhanced Procedures for Direct Trajectory Optimization Using Nonlinear Programming and Implicit Integration,” *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Keystone, Colorado, Aug. 21–24, 2006.

[15] McCarter, J., “MAVERIC Version 3.6 User’s Guide,” Tech. rep., NASA Marshall Space Flight Center, June 2009.

[16] Cook, S. A. and Vanhooser, T., “The Next Giant Leap: NASA’s Ares Launch Vehicles Overview,” *Aerospace Conference, 2008 IEEE*, Big Sky, Montana, March 1–8, 2008.

[17] Witzberger, K. E., Smith, D. A., Martini, M. C., and Wright, T. W., “MASTIF User’s Manual,” to be released, NASA Glenn Research Center.

[18] Lawden, D. F., *Optimal Trajectories for Space Navigation*, London: Butterworths, 1963.

[19] Frank, M. P., “Detailed Mission Planning Considerations and Constraints,” TMX–70045, NASA Johnson Space Center, Houson, Texas, Nov. 1966.

[20] Martin, D. T. and Hooper, H. L., “A Simplified Cutoff Hypersurface for Iterative Guidance During the Lunar Injection Burn,” TMX-53025, undated, NASA.

[21] Martin, D. T. and et al, “Saturn V Guidance, Navigation, and Targeting,” *Journal of Spacecraft and Rockets*, Vol. 4, No. 7, 1967, pp. 891–898.

[22] Martin, D. T. and Sievers, R. F., “An Emperical Simulation of the Translunar Injection Burn for Apollo,” *Journal of Spacecraft and Rockets*, Vol. 6, No. 4, 1969, pp. 483–486.

[23] Smith, I. E., “General Formulation of the Iiterative Guidance Mode,” TMX-53414, NASA, MSFC, Huntsville, Alabama, March 1966.

[24] Jaggers, R. F., “An Explicit Solution to the Exoatmospheric Powered Flight Guidance and Trajectory Optimization Problem for Rocket Propelled Vehicles,” *Guidance and Control Conference*, Hollywood, Florida, Aug. 8–10, 1977.

[25] McHenry, R. L. and et al, “Space Shuttle Ascent Guidance Navigation, and Control,” *Journal of the Astronautical Sciences*, Vol. XXVII, No. 1, 1979, pp. 1–38.

[26] Dukeman, G., “Atmospheric Ascent Guidance for Rocket-Powered Launch Vehicles,” *AIAA Guidance, Navigation, and Control Conference*, Monterey, California, Aug. 5–8, 2002.

[27] Calise, A. J., Melamed, N., and Lee, S., "Design and Evaluation of a Three-Dimensional Optimal Ascent Guidance Algorithm," *Journal of Computational and Applied Mathematics*, Vol. 21, No. 6, 1998, pp. 867–875.

[28] Bentley, E., "Revisiting Apollo: Trans-Lunar Injection Guided Burn to the Moon," PowerPoint Prensentation, United Space Alliance, AIAA Annual Technical Symposium 2006, May 2006.

[29] Roncoli, R. B., "Lunar Constants and Models Document," JPL D-32296, JPL, Jet Propulsion Laboratory, Pasadena, CA, Sept. 2005.

[30] Navigation and Facility, A. I., "SPICE," Website; accessed August 2010, `http://naif.jpl.nasa.gov/naif/`.

[31] Ben-Asher, J. Z., *Optimal Control Theory with Aerospace Applications*, American Institute of Aeronautics and Astronautics., 2010.

[32] Betts, J. T., *Practical Methods for Optimal Control Using Nonlinear Programming*, Society for Industrial and Applied Mathematics, 2001.

[33] Fehlberg, E., "Classical fifth-, sixth-, seventh- and eight-order runge-kutta formulas with step-size control," NASA TR r-287, NASA, 1968.

[34] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, Dover Publications, Inc., 1971.

[35] unknown author, "Orbital Elements," Website; accessed August 2010, `http://spaceflight.nasa.gov/realdata/elements/graphs.html`.

[36] Kaplan, G. H., "The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models," Circular No. 179, United States Naval Observatory, Oct. 2005.

[37] Wikipedia, "Yaw, pitch, and roll," Website; accessed August 2010, `http://en.wikipedia.org/wiki/Yaw,_pitch,_and_roll`.

[38] Schaub, H. and Junkins, J. L., *Analytical Mechanics of Space Systems*, American Institute of Aeronautics and Astronautics, Inc., 2003, p. 484.

[39] Jaggers, R. F., "Asymmetrical Booster Ascent Guidance and Control System Design Study, Volume V, Space Shuttle Powered Explicit Guidance," NASA CR 140191, NASA, 1976.

[40] Schubert, G. R., "Algorithm 211, Hermite Interpolation," *Communications of the ACM*, Vol. 6, No. 10, 1963, pp. 617.

[41] Canuto, C., Hussaini, A. Q., and Tang, T. A., *Spectral Methods in Fluid Dynamics*, Springer-Verlang, 1987.

[42] Stanford Business Software Inc, "SNOPT," Website; accessed August 2010, `http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.html`.

[43] "Exlx_Flt3.0_Option_1_North_to_South_RKF78_MODIFIED.txt," Plain text file, 2010, From NASA Johnson Space Center.

[44] Gedeon, G. S., “A Practical Note on the Use of Lambert’s Equation,” *AIAA Journal*, Vol. 3, No. 1, Jan. 1965, pp. 149–150.

[45] Andreani, R., Martinez, J. M., and Schuverdt, M. L., “On Second Order Optimality for Nonlinear Programming,” *Optimization*, Vol. 56, No. 5 & 6, Oct. 2007, pp. 529–542.

[46] Büskens, C. and Maurer, H., “SQP-methods for solving control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control,” *Journal of Computational and Applied Mathematics*, Vol. 120, 2000, pp. 85–108.

[47] NASA Tech Briefs, Tech. rep., NASA, Sept. 2009.

[48] Desai, P. N. and Conway, B. A., “Two-Timescale Discretization Scheme for Collocation,” *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, 2008, pp. 1316–1322.

[49] Desai, P. N. and Conway, B. A., “Six-Degree-of-Freedom Trajectory Optimization Utilizing a Two-Timescale Collocation Architecture,” *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, 2008, pp. 1308–1315.

[50] Office of the Chief Engineer, “NASA Procedural Requirements, NASA Software Engineering Requirements,” NASA NPR 7150.2a, NASA, 2009.

[51] Bryson, A. E. and Ho, Y., *Applied Optimal Control*, Blaisdell Publishing Company, 1969.

[52] Gill, P., Murray, M., and Wright, M. H., *Practical Optimization*, Academic Press, 1981.

[53] Kirk, D. E., *Optimal Control Theory*, Prentice-Hall, 1970.

[54] Hamiltion, W. R., “On a New Species of Imaginary Quantities Connected with a Theory of Quaternions,” *Dublin Proceedings*, Vol. 2, No. 13, 1843, pp. 424–434.

[55] Kuipers, J. B., *Quaternions and Rotation Sequences*, Princeton University Press, 1999.

[56] Collins, G.W., “Fundamental Numerical Methods and Data Analysis,” Website; accessed August 2010, `http://ads.harvard.edu/books/1990fnmd.book/`.

Appendices

Appendix A

Terminology

A.1 Optimality

In optimal control theory, 1^{st} order necessary and 2^{nd} order sufficient conditions for optimality must be satisfied for a candidate solution to be a local minimal [51]. The conditions for optimality depend on the problem formulation and whether the problem is constrained or unconstrained. For example, in a Calculus of Variations (CoV) approach, requiring the 1^{st} variation of the performance function to vanish satisfies the 1^{st} order necessary conditions. Such a solution is called an extremal or stationary point and is a candidate for a local extrema. Additionally, if the matrix containing the 2^{nd} variation of the performance function (hessian) is positive semidefinite, the solution has satisfied the 2^{nd} order optimality condition and is a local minimum. In nonlinear programming theory, optimization with inequality constraints makes use of the Karush-Kuhn-Tucker (KKT) conditions [51] which govern the optimality conditions [52]. Solutions that satisfy only 1^{st} order necessary conditions, but have not had 2^{nd} order conditions verified cannot be considered local minimizers because the solution could be a conjugate point [31].

A.2 Indirect vs. Direct Methods

Optimization solution techniques for trajectory optimization problems are typically categorized as indirect or direct. Indirect methods often use the CoV and/or Pontryagin's minimum principle. In dynamic programming, solving the Hamilton-Jacobi-Bellman equation [53] accomplishes the minimization of the performance measure. Direct methods discretize the problem and transform it (known as transcription) to a constrained parameter optimization problem. Each method has its strengths and weaknesses. Indirect methods with the variational approach require solving a two-point-boundary value problem (TPBVP), which can be difficult to solve. Generally speaking, the strength of indirect methods includes accuracy, speed of execution (usually), and the ability to readily verify optimality. Their weakness when trying to solve the TPBVP is robustness (the ability to converge from a poor initial guess) and flexibility (the ability to optimize different types of trajectories). Direct methods have the strength of robustness and flexibility, but can be less accurate and optimality can be difficult to verify because of discretization.

A.3 Open-loop vs. Closed-loop Solutions

If the intermediate outputs from a system are not used to modify the future inputs, the final output is called an open-loop solution. Solutions from trajectory optimization programs, whether direct or indirect, are open-loop solutions. A closed-loop solution is one in which the intermediate outputs effect the future inputs (typically through feedback laws). Rocket and missile guidance systems are closed-loop examples. Aerospace vehicle such as rockets and missles tend to utilize indirect methods for closed-loop guidance schemes because of the fast execution speed required on flight hardware and data storage limitations.

A.4 Guidance, Navigation and Control

Navigation interprets the acceleration and angular rates from sensor readings and provides smooth vehicle rotational and translational states at a set frequency for use by guidance and control algorithms.

The role of guidance is to safely transfer a vehicle from some initial state (i.e. position and velocity) to some desired end state while optimizing (often minimizing) some performance measure (e.g. mass). If required, guidance must also handle any additional constraints such as maneuver rates or acceleration limits.

The role of a control system is basically to convert the guidance and the navigation signals to actual effector commands while stabilizing the vehicle's response. This is a simplification, but is true of most control systems.

A.5 Software Simulation vs. Optimization

Software vehicle simulation refers to software that models the dynamics of a rigid body vehicle and its environment. These vehicle dynamics can be represented with three translational degrees-of-freedom (3-DOF) and with an additional three rotational degrees-of-freedom (6-DOF). The vehicle's mass distribution is accounted for with 6-DOF, but is treated essentially as a point mass with 3-DOF modeling/simulation. Additionally, in the context of this thesis, software simulation also is understood to include the capability to utilize existing closed-loop systems as well as design, implement, and test new ones. Feedback laws enable these closed-loop systems for guidance (3-DOF) and control (6-DOF).

Optimization refers to software that has the primary purpose of minimizing some performance measure of a trajectory simulation. Typically, mass distribution is either unknown or unaccounted for and so optimization is limited to the vehicle's three translational states.

A.6 Explicit Integration vs. Implicit Integration

Explicit integration is an integration scheme that integrates a set of (continuous) differential equations in a desired direction (i.e. forward or backwards) by stepping through time to generate each value one by one. Any Runge-Kutta scheme is an example of an explicit integration. The states are continuously integrated and are known for any time point.

Implicit integration, in the context of this thesis, refers to the method of solving a (discretized) set of differential equations simultaneously at all discrete points (commonly referred to as nodes or knots). Collocation is an example of an implicit integration method.

A.7 Perigee vs. Pericynthion

Perigee refers to the nearest point of a satellite's orbit around the Earth whereas pericynthion is the closest point around the Moon.

Appendix B

Quaternion Review

Quaternions are vectors in four-dimensional space and originated with Sir W. R. Hamilton [54]. A quaternion can be represented with a three-dimensional vector and a scalar

$$q = \begin{pmatrix} \sin(\frac{\Theta}{2})n_1 \\ \sin(\frac{\Theta}{2})n_2 \\ \sin(\frac{\Theta}{2})n_3 \\ \cos(\frac{\Theta}{2}) \end{pmatrix} \tag{B.1}$$

where the scalar component, $\cos(\frac{\Theta}{2})$, contains the half-angle of rotation and

$$\begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} \tag{B.2}$$

is a vector.

Special quaternions, ones with unit magnitude, are useful for representing coordinate frame rotations. These unit quaternions, called rotation quaternions, are analogous to direction cosine matrices. The advantages of using quaternions, instead of rotation matrices, are computational efficiency (a rotation represented with a quaternion does so with fewer variables) and avoidance of singularities in certain kinematic equations. For more details on quaternions the reader is referred to [55].

B.1 Multiplication

The following quaternions are represented with the scalar part w_q and vector part $\mathbf{v}_q$.

$$q_1 = \mathbf{v}_{q1} + w_{q1} \tag{B.3}$$

$$q_2 = \mathbf{v}_{q2} + w_{q2} \tag{B.4}$$

$$q_1 \otimes q_2 = w_{q1}(w_{q2}) - \mathbf{v}_{q1} \cdot \mathbf{v}_{q2} + w_{q1}(\mathbf{v}_{q2}) + w_{q2}(\mathbf{v}_{q1}) + \mathbf{v}_{q1} \times \mathbf{v}_{q2} \tag{B.5}$$

B.2 Single Axis Rotations

The scalar part is the last element. The generic rotation angle is represented by Θ.

$$q_x(\Theta) = \begin{pmatrix} \sin\frac{\Theta}{2} \\ 0 \\ 0 \\ \cos\frac{\Theta}{2} \end{pmatrix} \tag{B.6}$$

$$q_y(\Theta) = \begin{pmatrix} 0 \\ \sin\frac{\Theta}{2} \\ 0 \\ \cos\frac{\Theta}{2} \end{pmatrix} \tag{B.7}$$

$$q_z(\Theta) = \begin{pmatrix} 0 \\ 0 \\ \sin\frac{\Theta}{2} \\ \cos\frac{\Theta}{2} \end{pmatrix} \tag{B.8}$$

Appendix C

Coordinate Transformations

C.1 EME2000 to Moon

$$\tilde{\boldsymbol{\Omega}}_{☾} = \begin{bmatrix} 269.9949\ (degrees) \\ 0.0031\ (\frac{degrees}{Julian\ century}) \end{bmatrix} \tag{C.1}$$

$$\tilde{\boldsymbol{\delta}}_{☾} = \begin{bmatrix} 66.5392\ (degrees) \\ 0.0130\ (\frac{degrees}{Julian\ century}) \end{bmatrix} \tag{C.2}$$

T_{JC} = time past J2000 in Julian centuries

$$\Omega_{☾} = \tilde{\Omega}_{☾}(1) + \tilde{\Omega}_{☾}(2)(T_{JC}) \tag{C.3}$$

$$\delta_{☾} = \tilde{\delta}_{☾}(1) + \tilde{\delta}_{☾}(2)(T_{JC}) \tag{C.4}$$

Convert $\Omega_{☾}$ and $\delta_{☾}$ to radians.

$$\boldsymbol{moonPole} = \begin{bmatrix} \cos\Omega_{☾}\,(\cos\delta_{☾}\,) \\ \sin\Omega_{☾}\,(\cos\delta_{☾}\,) \\ \sin\delta_{☾} \end{bmatrix} \tag{C.5}$$

$$\boldsymbol{j2kPole} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{C.6}$$

Perform the following vector cross products.

$$\boldsymbol{iauVector} = \boldsymbol{j2kPole} \times \boldsymbol{moonPole} \tag{C.7}$$

$$\boldsymbol{yVector} = \boldsymbol{moonPole} \times \boldsymbol{iauVector} \tag{C.8}$$

$$M_{j2ktoMoon} = \begin{bmatrix} \boldsymbol{iauVector} \\ \boldsymbol{yVector} \\ \boldsymbol{moonPole} \end{bmatrix} \tag{C.9}$$

$$q_{j2ktoMoon} = DCM2Quat(M_{j2ktoMoon}) \tag{C.10}$$

where $DCM2Quat$ is a function in MASTIF that converts a direction cosine matrix into a quaternion.

C.2 LVLH to Body

Three successive rotations are required to rotate from the LVLH frame to the body frame. The 2-3-1 sequence starts with a rotation about the LVLH y-axis through the angle θ, followed by a rotation about the intermediate z-axis through the angle ψ and ends with a rotation about an intermediate x-axis through the angle ϕ. A rotation quaternion, described in Appendix A, represents this rotation as

$$q_{lvlh2body} = q_y(\theta) \otimes q_z(\psi) \otimes q_x(\phi) \tag{C.11}$$

where the LVLH Euler angles are loosely referred to as

$$\theta = pitch, \quad \psi = yaw, \quad \phi = roll$$

and utilizes quaternion multiplication described in Appendix B.

Appendix D

Hermite Interpolating Polynomial

The derivation of these polynomials can be found in [56], whereas the expressions below are nearly exactly as those given in [14].

The interpolated state value is given by

$$\tilde{s}(t) = \sum_{i=1}^{n} \alpha_i(t) s(t_i) + \sum_{i=1}^{n} \beta_i(t) \dot{s}(t_i) \tag{D.1}$$

where

$$n = \frac{nNodes_{seg} + 1}{2} \tag{D.2}$$

and the generic functions, $\alpha_j(t)$ and $\beta_j(t)$ are

$$\beta_j(t) = \left[\prod_{\substack{i=1 \\ i \neq j}}^{n} \left(\frac{t - t_i}{t_j - t_i} \right)^2 \right] (t - t_j) \tag{D.3}$$

$$\alpha_j(t) = \left[\prod_{\substack{i=1 \\ i \neq j}}^{n} \left(\frac{t - t_i}{t_j - t_i} \right)^2 \right] (1 + 2(t_j - t)) \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{1}{t_j - t_i} \tag{D.4}$$

and it is understood that the subscript j is a dummy index.

Likewise, the interpolated state derivative value is

$$\dot{\tilde{s}}(t) = \sum_{i=1}^{n} \dot{\alpha}_i(t) s(t_i) + \sum_{i=1}^{n} \dot{\beta}_i(t) \dot{s}(t_i) \tag{D.5}$$

where $\dot{\alpha}_i(t)$ and $\dot{\beta}_i(t)$ can be obtained by the product rule of differentiation.