MODIFIED CHEBYSHEV PICARD ITERATION: INTEGRATION OF PERTURBED MOTION USING MODIFIED EQUINOCTIAL ELEMENTS

A Dissertation

by

JULIE LOUISE READ

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	John L. Junkins
Co-Chair of Committee,	Ahmad Bani Younes
Committee Members,	Gregory Chamitoff
	Shankar P. Bhattacharyya
Head of Department,	Rodney Bowersox

December 2016

Major Subject: Aerospace Engineering

ABSTRACT

The topic of this dissertation is the fusion of a novel integration method, Modified Chebyshev Picard Iteration (MCPI), with Gauss' Variational Equations using a set of Modified Equinoctial Orbital Elements. This combination leads to a dramatically increased domain of Picard iteration convergence and an efficiency increase for MCPI solutions of the Initial Value Problem of Celestial Mechanics, thereby reducing the number of full gravity function calls. The set of Modified Equinoctial Orbital Elements (MEEs) are nonsingular over a large orbit variation domain, in contrast with the Classical Orbital Elements (COEs), which are singular at zero inclination and zero eccentricity, and propagation of MEEs with MCPI leads to much greater convergence time intervals for the IVP than is possible using Cartesian coordinates. This set of elements is also used to formulate the Two-Point Boundary Value Problem (TPBVP) associated with orbit transfer using a low-thrust, minimum-time control formulation and solves iteratively via a shooting method known as the Method of Particular Solutions.

DEDICATION

For my family and friends who have supported me throughout my education and encouraged me to pursue my dreams.

I am forever thankful for your kind words and patience.

I appreciate you all more than you know.

ACKNOWLEDGEMENTS

To my husband Tim for supporting me throughout my PhD- you were my biggest supporter when I said I wanted to go back to graduate school. You have spent countless hours giving me advice, encouraging me, and helping me to destress when I need to just slow down and take time to relax. You have never complained about me working nights and weekends and have been so supportive throughout my degree. I could not have gotten through my degree without you! Commuting from Houston for a PhD was only feasible with your amazing support. I am lucky to have you as my husband and my biggest fan, and I look forward to the next stage of our lives together.

To my parents Terry and Joanne for teaching me to work hard and to never give up- you can take the woman out of the farm, but you can never take the farm out of the woman! All those hours of helping out (and not to mention miles and miles of clipping rye!) taught me that luck follows those who work hard and live with integrity. You spent many years pushing me to do my best and to excel in school, and that the only true failure is to not try. As a result I now have the tenacity and resilience to tackle any challenge. Thank you for always inspiring me to be the best I can be.

To my sister- we have had many years of memories, and I look forward to many more. We share so many interests and are alike in so many ways, and I am grateful for the time we have together. Thank you for helping me get started with my PhD and giving me a place to stay in College Station my first year. I was sad to see you leave, but I am so happy you moved back to Texas! Even if I have been called Lori instead of you being called Julie, for a change.

To Dr. Junkins and Dr. Ahmad Bani Younes- thank you for helping me navigate through my studies and research and for always believing in me. I will remember your advice for years to come. I am very fortunate to have you both as my co-advisors, and the the past four years have been well-spent with your infinite wisdom guiding me.

To Dr. Bani Younes- thank you for helping me debug my code when I am stuck and for your patience in communicating from halfway across the world. Your willingness to Skype with me at all hours of the day is much appreciated! You are already on your way to becoming a well-known professor.

Dr. Junkins- thank you for all your advice regarding school, career, and life in general. I am thankful you push your students to be well-rounded individuals and that you truly care about our success. When I saw you during my Master's at A&M, I wondered if you would be willing to talk to me; I have now learned that not only would you talk to me, but that you will talk to me for as long as I will listen! You truly are a walking legend, and I hope I can soak up every bit of knowledge I can from you. I am convinced you live in a higher dimension than the rest of the world.

To Dr. Chamitoff- thank you for pushing me to pursue my PhD, even though it meant commuting from Houston. You are right, it has been a long road. As you said, "it took someone crazy to tell you do to something crazy!" to convince me to follow this path. I can see now that this was the best choice, and I value your encouragement, insight, and modesty. Thank you so much for stepping in and serving on my committee.

To Dr. Bhattacharyya- thank you for being an outstanding teacher and committee member. I learned a lot in your class, and I am grateful to have you on my committee. If I could pursue a second PhD, I would see if you were taking any new students.

Thanks to additional members of the Texas A&M MCPI research team: Robyn Woollands, Brent Macomber, Austin Probe, Abhay Masher, Nathan Budd, Chris Shelton, Xiaoli Bai, Tarek Elgohary, and Donghoon Kim. I am proud to be a part of our team, and I look forward to seeing you all succeed in your careers.

Robyn- thank you for working with me the past four years. We have made it through this program together and have overcome many hurdles. We have grown in so many ways, and I am glad we have each other to lean on.

Kevin - Thank you for all your coding advice and discussions. I appreciate you taking the time to help me become a more efficient programmer.

To Dr. Mortari- thank you for all the coffee breaks and teaching me the proper way to drink an espresso. I am very glad that I had the opportunity to take a few classes from you and attend conferences with you. You always brighten my day when I see you.

Thanks to Dr. Bob Gottlieb (Odyssey Space Research) and Dr. Terry Feagin (UHCL) for their discussions of numerical propagators, normalized gravity, and STM calculations. You can finally call me "Dr." even though you've been doing that for a few years- thanks for your confidence!

To Dr. Turner- thank you for forcing me to learn Maxima (symbolic toolbox). It has served me well during my dissertation research. Thank you also for giving me feedback on my presentations at conferences, as well as sharing knowledge from some of your industry experience with me.

Thank you to JSC (EG-5) for agreeing to a new partnership with Texas A&M and welcoming me into your group during my last semester. I have met many intelligent, dedicated employees who share my passion for space exploration, and it is an honor to join you for a few months.

Last, but not least, thank you to the sponsors for this work: AFOSR (Julie Moses and Stacie Williams), AFRL (Alok Das), and ADS (Matt Wilkins).

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

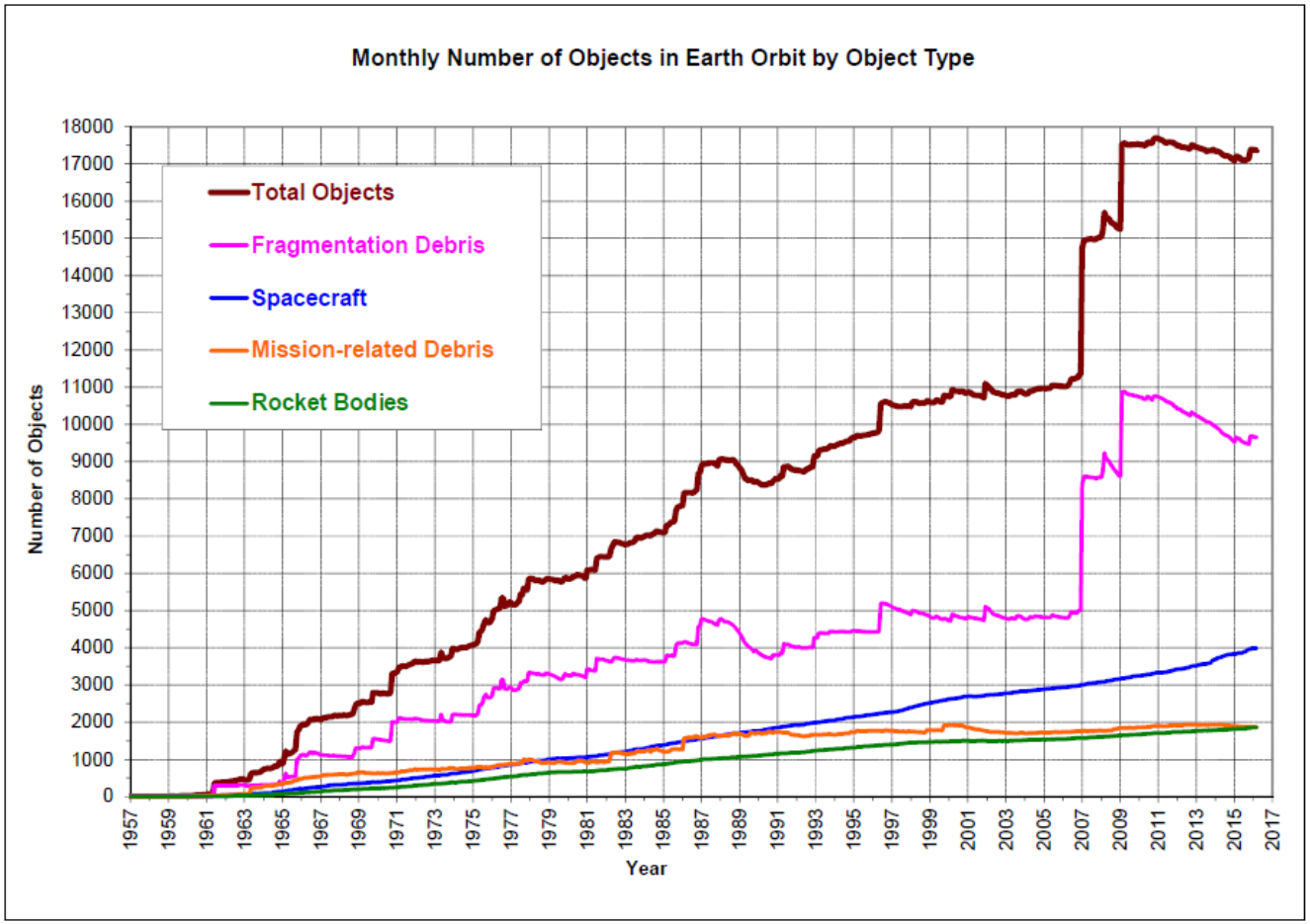
1. INTRODUCTION

1.1 Motivation

This dissertation is motivated mainly by two major contemporary challenges in astrodynamics: orbital debris mitigation and low-thrust orbital transfers. In both problems, a need exists for improved computational efficiency and robustness of the algorithms, that decreases reliance on prior empirical knowledge.

Since the beginning of the space age, orbital debris has increasingly accumulated in Earth orbit. This mostly linear trend of ~ 200 new objects per year tracked by the Department of Defense, National Aeronautics and Space Administration (NASA), and European Space Agency (ESA) [1] is shown in Figure 1.1. Most notable departures from the linear trend are two recent collision and breakup events (the collision of Iridiuim and Cosmos in 2009 and the deliberate destruction of the Fengyun satellite in 2007), where a large increase is seen in the resulting number of trackable orbital debris. Following these events, a deceptive trend shows the number of objects decreasing; however, this is due to many now-smaller and high area-to-mass-ratio objects burning up in the Earth's atmosphere. The overall trend of orbital debris will likely continue upward (especially if there are more major collisions) until measures are taken to remove objects that are most likely to collide and minimize creation of new debris.

Of the $\sim 22,000$ trackable objects presently in Earth orbit, about 2000 are operational spacecraft. An estimated 500,000 non-trackable, but lethal debris objects exist. It is evident that taking down large debris objects, such as spent boosters, before they collide is a key strategy to mitigating debris growth due to future collisions.



Monthly Number of Cataloged Objects in Earth Orbit by Object Type: This chart displays a summary of all objects in Earth orbit officially cataloged by the U.S. Space Surveillance Network. "Fragmentation debris" includes satellite breakup debris and anomalous event debris, while "mission-related debris" includes all objects dispensed, separated, or released as part of the planned mission.

Figure 1.1: Orbital Debris Categorized by Type. These data are for the publicly available space object catalog.

There is a strong need to refine forecasting of the most threatening pairs of orbital objects so they can be prioritized for mitigation prior to likely future collisions. The studies presented in this dissertation demonstrate significantly enhanced numerical methods that may be used to efficiently propagate large catalogs, find future conjunctions and thereby enable the analysis to avoid future collisions. The advent of the planned new Air Force radar fence is anticipated to increase the number of trackable objects from $\sim 22,000$ to $\sim 100,000$. Solving the inverse problem (associating multiple observation sets with the same physical object, determining accurate orbits, and updating space object catalogs) will then be many orders of magnitude more computationally taxing than is presently the case. As real-time tracking of orbital debris becomes more demanding and critical, the tradeoff beween accuracy and speed must be considered. Efforts must be re-doubled to establish the most efficient computational methods for all aspects of space situational awareness (SSA). One approach is to emphasize parallel computation through adoption of sofware/hardware architecture such as using graphics processing units (GPUs) or compute clusters [2, 38, 47]. A main advantage of the propagation method used throughout this dissertation, Modified Chebyshev Picard Iteration, is that it is inherently massively parallelizable, in constrast with many traditional step-by-step integrators. This algorithm and its derivatives promise to also find its way into enhanced orbit determination, SSA hypotheses testing, and orbit transfer computations.

Efficiently optimizing orbit transfer maneuvers to reach and/or de-orbit spacecraft to help avoid potential space collisions will assume greater importance [31]. The orbit may be represented in a number of ways, with Cartesian coordinates a popular choice. However, as shown in this dissertation, using instead a set of slowly varying orbital elements has several advantages over Cartesian coordinates for the chosen integration method. Therefore, a change in velocity (for maneuvers and orbit

transfers) may be modeled through a nonlinear transformation (coupled with the position vector) in terms of changes in osculating orbital elements.

Another motivating factor is that the Cartesian implementation of the Picard iteration integration method (MCPI) is limited to approximately three orbits (initial value problem) and one-third of an orbit (boundary value problem) before reaching a limit on the convergence domain [5]. A regularizing formulation using the KS transform increases this domain of convergence to about 90% of an orbit [62, 64]. The convergence domain is greatly enhanced ~ 1 order of magnitude by using orbital elements with a variant of MCPI [48, 49].

In addition, a controls formulation is considered where low-thrust is a feasible option. Many deep-space mssions typically require large velocity increments, and low-thrust propulsion systems allow for larger velocity increments than high-thrust propulsion systems because they utilize propellant more efficiently [30]. The most important advantage is the great reduction in the mass of propulsion hardware and propellant. Low thrust systems typically have a specific impulse of more than one order of magnitude more compared to chemical propulsion, and the low thrust rocket engines are typically two orders of magnitude lighter than the chemical propulsion engines. The disadvantage, of course, is that the "time of flight" of low-thrust engines is frequently much larger than for chemical propulsion.

The field of spacecraft trajectory optimization is inherently complicated, due to several reasons [13], including nonlinearity of the dynamic system, time-dependent forces, discontinuities in the state variables (i.e., instantaneous velocity changes), terminal conditions that are not known explicitly (for instance, the position of the departure and arrival for an interplanetary trajectory depend upon the terminal times, which are often optimization variables), and difficulty determining whether a local or global solution has been found.

Though low-thrust propulsion has been an attractive method for many years, spacecraft have only recently incorporated it [13]. This method of propulsion produces a very small thrust, of the order of magnitude $10^{-3}N$ to $10^{-5}N$, and is used continuously or nearly continuously. The continuous thrust optimal control problem must take into account continuous time histories for the thrust magnitude and direction.

Where the ratio of thrust to mass $\frac{T}{m}$ is relatively small, a minimum time problem is approximately the same as the minimum fuel problem. If a spacecraft with continuous thrust but no throttling is considered, the minimum-fuel transfer will be the same as the minimum-time transfer since the mass flow rate of the propellant will be constant [56]. Though a low-thrust system may necessitate several orbits in order to achieve the desired final state, it is one approach to achieve a real-world solution and has significant advantages over chemical propulsion with regard to the launch costs of lower propellant mass. The set of orbital elements that are used to increase the domain of convergence for the integration method are also nonsingular for most practical applications.

Striving toward practical applications during this research, wherever possible throughout this dissertation, orbit perturbations due to Earth's gravity using a spherical harmonic gravity model (EGM2008) have been used to provide a precise model [24]. More specifically, selected examples using high fidelity gravity models make it abundantly clear that the methods are not merely aimed at simplified problems of interest mainly to academics. While the full spherical harmonic gravity is included, not everything is modeled. Even higher fidelity models will require additional perturbations, such as solar radiation pressure, drag, or third-body effects.

1.2 Dissertation Outline

Chapter 1 serves as an introduction and presents the motivation behind this research.

Chapter 2 gives an overview of MCPI, including the Initial Value Problem (IVP), Boundary Value Problem (BVP), and a standardized algorithm for solving IVPs as well as a cascade solution process for second-order differential equations. Additional information is provided in Appendix A about the Chebyshev Polynomials, which serve as the set of orthogonal basis functions for the MCPI method.

Chapter 3 summarizes the Modified Equinoctial Orbital Elements (MEEs) and gives Gauss' Variational Equations, which are used for orbit propagation. Simulation results are provided that demonstrate the increased domain of convergence and reduction in MCPI iterations, as well as the advantages of orbit segmentation. Appendix C gives the derivation for the Jacobi energy integral, which is also the Hamiltonian for a particular choice of coordinates, and which may be used to verify orbit solutions for both zonal and high degree and order spherical harmonic gravity perturbed orbits.

Chapter 4 discusses a novel development of the State Transition Matrix (STM) that includes perturbations of arbitrary degree spherical harmonic gravity, including the necessary computation of Associated Legendre Functions' gradients. The corresponding Jacobian matrix derivation is given in this chapter, and additional details are provided in Appendix B. Both the zonal gravity and spherical harmonic gravity cases are considered here, and they are also both used as the basis for a local Taylor Series gravity described in Ch. 5. This local Taylor series gravity approximation is useful to model gravity in the terminal convergence of MCPI. A cascade method is also presented and shown to reduce computation time, analogous to the Cartesian

coordinate two-body problem. A new canonical form for the STM is also derived, that promises to lead to computational efficiency in some applications. Appendix D gives supplemental details about using canonical units for a formulation originally in Cartesian coordinates.

Chapter 5 provides results from a Monte Carlo study using Modified Equinoctial Orbital Elements (MEEs) and a local Taylor Series gravity expansion, where the Taylor series is a representation of the force field neighboring nodes at specific times along the nominal trajectory. The Taylor Series gravity model, given in detail in this chapter, significantly reduces the number of full gravity computations by using a Taylor Series approximation for the gravity model during intermediate Picard (MCPI) iterations. Results are given for both Matlab (serial processor) and for a compute cluster that uses parallel processing. Appendix E lists the specifications of the compute cluster used for the parallel computation aspects of this study, which is located in the Land, Air, and Space Robotics Lab (LASR) at Texas A&M University.

Chapter 6 outlines a shooting method approach to solving BVPs using MCPI known as the Method of Particular Solutions (MPS). This method is advantageous in that it avoids the computation of state transition matrices and also the inverse of matrix partials commonly used in shooting methods. An analytical solution called p-Iteration may be used to provide a starting estimate for initial velocity for this method, if Lambert's problem is considered. MPS is, however, a general BVP solver that can be used for many problems, including solving state/costate differential equations, to find a suboptimal orbit transfer control with low-thrust steering angles (Chapter 7).

Chapter 7 solves the suboptimal control problem using MEEs (which specifies an initial and a target state) by first propagating the IVP solution of Gauss' Variational Equations until the minimum semilatus rectum, p, exceeds the desired value of p.

The transfer time that is found here is then used as a conservative final time guess for the orbital transfer. Next, the suboptimal control is found using the Method of Particular Solutions (MPS) to iteratively update the steering angles. Once a solution is found, the transfer time may be decreased until convergence is no longer achieved. This solution is needed to establish an initial guess for the indirect minimum-time optimal control solution, i.e., the solution of the Pontryagin optimal control necessary condtions (Chapter 8).

Chapter 8 provides details about a minimum-time, low-thrust, optimal control problem using a state/costate formulation and Pontryagin's minimum principle, using the MEE state variables.

2. MODIFIED CHEBYSHEV PICARD ITERATION

A recently refined numerical integration technique known as Modified Chebyshev Picard Iteration (MCPI) is the focus of this dissertation. It is a fusion of two concepts: Picard iteration and Chebyshev polynomials. The acceleration is approximated along the previous orbit approximation, then the acceleration approximation is integrated term-by-term, and boundary conditions are applied to obtain the new orbit approximation. During every Picard iteration, the coefficients for a Chebyshev polynomial series for the orbit coordinates are updated to give the new state estimate that satisfies all boundary conditions. This process is repeated until convergence is achieved. At every iteration, the entire trajectory (for example, an entire orbit) is approximated at sample nodes by computing the forcing function from the current state estimate.

See Figure 2.1 to see how a "cold" start with no apriori knowledge of the trajectory iteratively converges to an orbit. This figure shows how the entire orbit is approximated at every iteration (note that not every Picard iteration step is shown in this figure). Here, the notation ($''$) follows the traditional literature (e.g., Fox[21]), and indicates that the first and last term in the summation have a scale factor of $\frac{1}{2}$. The notation ($'$) indicates that only the first term in the summation is multiplied by a scale factor of $\frac{1}{2}$. This method is inherently parallelizable since the forcing function and the coefficients may be computed independently. This approach to numerical integration differs from traditional step-by-step methods and has several advantages over these methods.

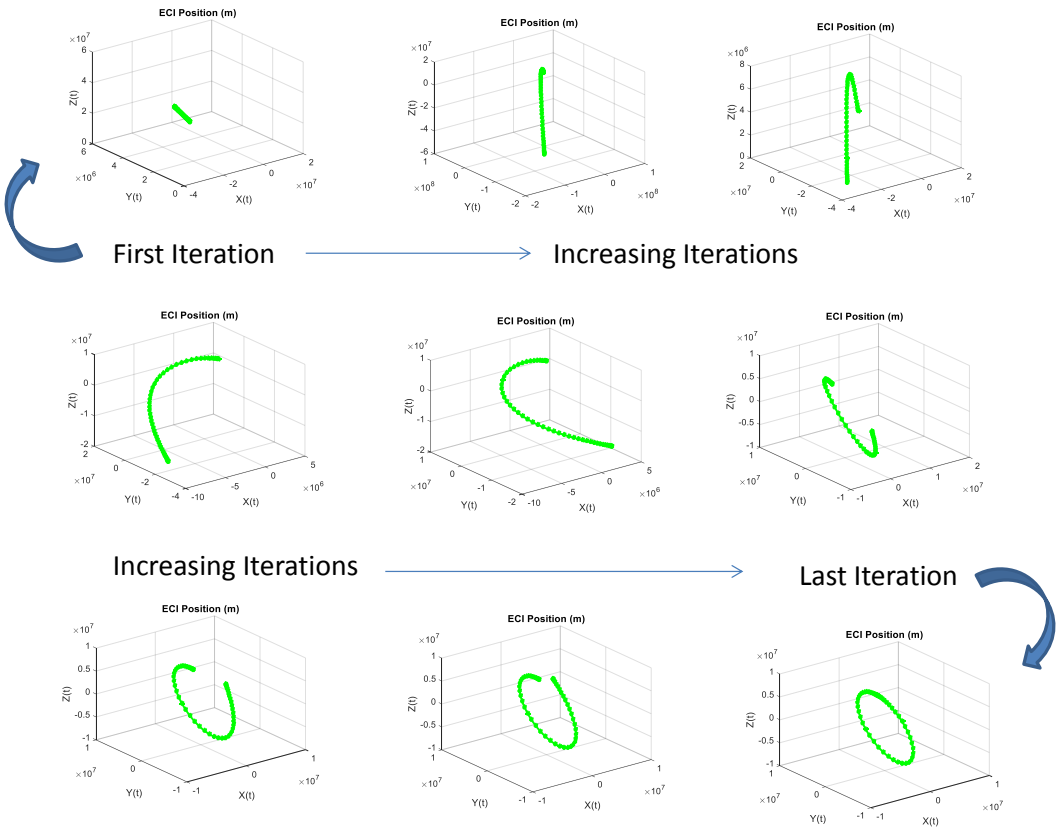


Figure 2.1: Position approximation at incremental Picard iterations with no a priori knowledge of the initial trajectory guess. It is remarkable that even a very poor starting path approximation can lead, in a few Picard iterations, to a machine precision solution for an entire orbit. Each subfigure is the current state approximation at the i^{th} Picard iteration.

2.1 MCPI Overview

MCPI is an iterative, path approximation method for solving smoothly nonlinear systems of ordinary differential equations. Clenshaw and Norton first proposed combining the orthogonal Chebyshev polynomials with Picard iteration [12]. Later authors including Shaver, Feagin and Nacozy, and Fukushima further refined the Chebyshev-Picard framework and also pointed out the parallel computing implica-

tions of the method [15, 20, 22, 55]. Additional work by Feagin and Mikkilineni applied the method to orbit determination, as well as batch and sequential estimation [17, 19, 43]. More recent developments in parallelizing MCPI give an expected increase in efficiency [4, 32, 38, 47].

MCPI generates a sequence of long-arc path approximations that can solve both linear and nonlinear, high precision, long-term orbit propagation problems. This approach iteratively refines an orthogonal function approximation for the state trajectory. At each iteration, MCPI finds a path integral solution over a large time interval, as opposed to the conventional, incremental step-by-step solution process of more familiar numerical integration strategies, such as those based on explicit numerical methods. The method introduces orthogonal function approximations of the forcing function along a previous path approximation, which can be analytically integrated term-by-term, along with exact boundary conditions enforced to produce the next path approximation. Significantly, unlike conventional integration approaches, this algorithm is ideally suited for massive parallel implementations that provides a way to further boost computational performance in comparison to most traditional numerical methods for solving differential equations. This is because the acceleration at all nodes along the current known orbit approximation can be simultaneously computed rather than sequentially as in step-by-step integration.

Enhancements to the MCPI algorithm allow for a decrease in the number of Picard iterations, which leads to a decrease in computation time. For Cartesian coordinates, typically 3 or 5 segments are used per orbit, while for the Modified Equinoctial Element case, typically one segment is used per orbit (as will be discussed in Chapter 3). For more information about MCPI segmentation, as well as a radial adaptive gravity formulation and other enhancements, see references [28, 36, 39, 46, 68].

Next, the two concepts that underlie MCPI will be discussed: Picard Iteration and Chebyshev Polynomials. After that, the initial value problem and boundary value problem formulations for MCPI will be given.

2.2 Picard Iteration

Picard showed that for a state vector $\mathbf{x}(t)$, given an initial condition $\mathbf{x_0}$ at the initial time t_0, a typical nonlinear system represented by an Ordinary Differential Equation (ODE) given as [44]

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x_0} \tag{2.1}$$

(with a smooth, single-valued, once differentiable right hand side) may be rearranged *without approximation* to obtain an integral equation:

$$\mathbf{x}(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{f}(s, \mathbf{x}(s))ds \tag{2.2}$$

For a given suitable starting approximation $\mathbf{x}^0(t)$, a unique solution to the initial value problem may be found using an iterative sequence of path approximations through Picard iteration as

$$\mathbf{x}^i(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{f}(\tau, \mathbf{x}^{i-1}(\tau))d\tau, \quad i = 1, 2, ... \tag{2.3}$$

Picard proved that this sequence converges to the unique solution of Eq. (2.1) over a finite interval $|t - t_0| < \delta$ with a starting estimate $|x^0(t) - x(t)| < \Delta$ if f is smooth and at least once differentiable [44]. The bounds δ and Δ can be conservatively estimated [27]; however, much larger bounds can typically be demonstrated numerically. The maximum time interval δ for convergence in orbital mechanics has been found to typically be hours for geocentric orbits and years for solar orbits.

Here, the integrand for each Picard iteration is approximated anew on each iteration using a Chebyshev polynomial series, and this approximation is integrated term-by-term to update the trajectory approximation. The Runge effect (large oscillatory approximation errors that are often seen near the boundaries of the approximation domain) is greatly reduced due to a cosine sampling scheme with nodes more densely clustered near the boundaries. For a comparison of linear vs. cosine sampling, see Figures 2.3a and 2.3b. For an example of how cosine sampling reduces the Runge effect, see Figures 2.4a and 2.4b.

2.3 Chebyshev Polynomials

These polynomials were established by a Russian mathematician named Pafnuty Lvovich Chebyshev in 1857 [11]. This set is useful in MCPI propagation because it is orthogonal, which results in a trivial matrix inverse in the MCPI formulation. As discussed below and in Appendix A, a discretely sampled version of the orthogonality conditions is used. There are several kinds of Chebyshev polynomials; the first kind is most useful for MCPI propagation, and the second kind may be used for taking derivatives to aid in certain MCPI boundary value problem formulations. Unless specifically noted, throughout the dissertation any reference to the Chebyshev polynomials of the first kind are simply the Chebyshev polynomials.

Chebyshev polynomials of the first kind are defined as polynomials in x of degree n, defined by the relation

$$T_n(x) = cos(n\theta) \qquad when\ x = cos\theta \tag{2.4}$$

where x exists on the interval $[-1, 1]$, and the range of θ is $[0, \pi]$. Here, $x = -1$ corresponds to $\theta = \pi$, while $x = 1$ corresponds to $\theta = 0$. The Chebyshev polynomials are generated using initial conditions $T_0(x) = 1$, $T_1(x) = x$ and the recursive

relationship

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \qquad n = 2, 3, ... \tag{2.5}$$

or through the trigonometric identity

$$T_n(x) = cos(n \arccos(x)) \tag{2.6}$$

The first six Chebyshev polynomials are plotted in Figure 2.2. It is interesting to note that $T_n(x)$ is either an even or odd function, involving either all even or all odd powers of x.

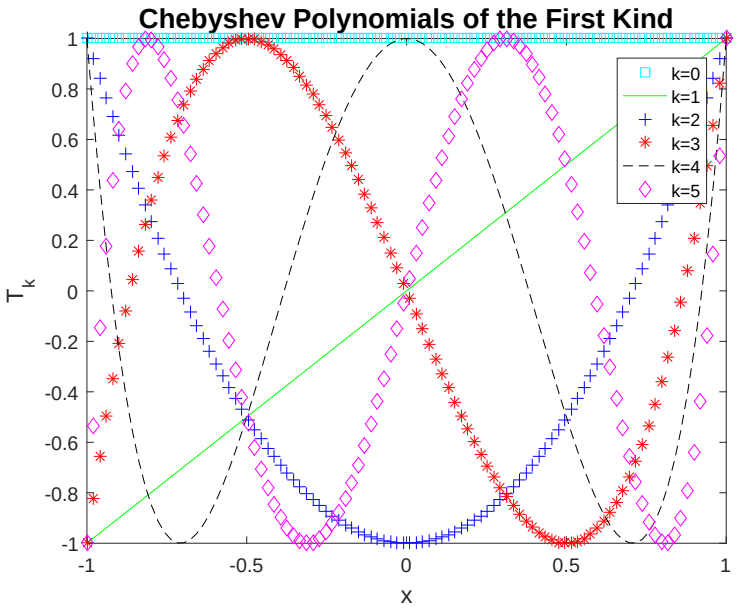


Figure 2.2: First six Chebyshev Polynomials $T_0 - T_5$

The first few Chebyshev polynomials of the first kind are written as

$$T_0(x) = 1 \tag{2.7}$$

$$T_1(x) = x \tag{2.8}$$

$$T_2(x) = 2x^2 - 1 \tag{2.9}$$

$$T_3(x) = 4x^3 - 3x \tag{2.10}$$

$$T_4(x) = 8x^4 - 8x^2 + 1 \tag{2.11}$$

$$T_5(x) = 16x^5 - 20x^3 + 5x \tag{2.12}$$

$$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1 \tag{2.13}$$

$$T_7(x) = 64x^7 - 112x^5 + 56x^3 - 7x \tag{2.14}$$

$$T_8(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1 \tag{2.15}$$

$$T_9(x) = 256x^9 - 576x^7 + 432x^5 - 120x^3 + 9x \tag{2.16}$$

$$T_{10}(x) = 512x^{10} - 1280x^8 + 1120x^6 - 400x^4 + 50x^2 - 1 \tag{2.17}$$

The Chebyshev polynomials have the following integration property, which is used to derive the MCPI formulations given throughout this dissertation [40]:

$$\int T_n(x)dx = \frac{1}{2}\left[\frac{T_{n+1}(x)}{n+1} - \frac{T_{n-1}(x)}{n-1}\right] + c_1 \quad n \neq 0, 1 \tag{2.18}$$

Note that this is an indefinite integral, and when evaluated at integration limits, the constant of integration cancels. The only exceptions to this property are

$$\int T_0(x)dx = T_1 \tag{2.19}$$

$$\int T_1(x)dx = \frac{1}{4}T_2 + c_2 \tag{2.20}$$

The constant c_1 depends upon the specific Chebyshev polynomial being integrated, while $c_2 = \frac{1}{4}$. However, when the integral is performed and the result evaluated at the limits, this constant will cancel out, so for definite integrals, $c_1 = 0$ in Eqn. (2.18).

2.4 Chebyshev-Gauss-Lobatto Nodes

The discrete nodes used to approximate the states are called the Chebyshev-Gauss-Lobatto (CGL) nodes. In constrast with uniform sampling, a cosine sampling scheme is advantagous in reducing the Runge effect, which is commonly observed in function approximation at the boundaries of curves. Cosine sampling, at either the zeros of $T_N(x)$ or the extrema of $T_N(x)$, is consistent with the discrete orthogonality condition of the Chebyshev polynomials. In addition, by judicious segmentation of approximation intervals, the more dense portion of the cosine sampled nodes can be strategically placed near perigee, where the gravity calculations require higher fidelity; fewer nodes are required at apogee where slower motion and smoother gravity are encountered. Figures 2.3a and 2.3b compare uniform sampling and cosine sampling for a Low Earth Orbit, while Figures 2.4a and 2.4b demonstrate that cosine sampling reduces the Runge effect.

Critically, approximation with high accuracy can be achieved if nodes are selected to ensure discrete orthgonality conditions are satisfied (see Appendix A). For Chebyshev polynomials, the CGL nodes are one of two choices consistent with the orthogonality conditions. The CGL (cosine) nodes locate the Chebyshev Polynomials' $(M-1)$ extrema and may be calculated simply as

$$\tau_j = -cos\left(\frac{j\pi}{M}\right),\ j = 0, 1, ..., N \tag{2.21}$$

The matrix of Chebyshev polynomials is represented as

$$[T] = \begin{pmatrix} T_0(\tau_0) & T_0(\tau_1) & T_0(\tau_2) & \cdots & T_0(\tau_{M-1}) & T_0(\tau_M) \\ T_1(\tau_0) & T_1(\tau_1) & T_1(\tau_2) & \cdots & T_1(\tau_{M-1}) & T_1(\tau_M) \\ T_2(\tau_0) & T_2(\tau_1) & T_2(\tau_2) & \cdots & T_2(\tau_{M-1}) & T_2(\tau_M) \\ T_3(\tau_0) & T_3(\tau_1) & T_3(\tau_2) & \cdots & T_3(\tau_{M-1}) & T_3(\tau_M) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ T_{N-1}(\tau_0) & T_{N-1}(\tau_1) & T_{N-1}(\tau_2) & \cdots & T_{N-1}(\tau_{M-1}) & T_{N-1}(\tau_M) \end{pmatrix} \tag{2.22}$$

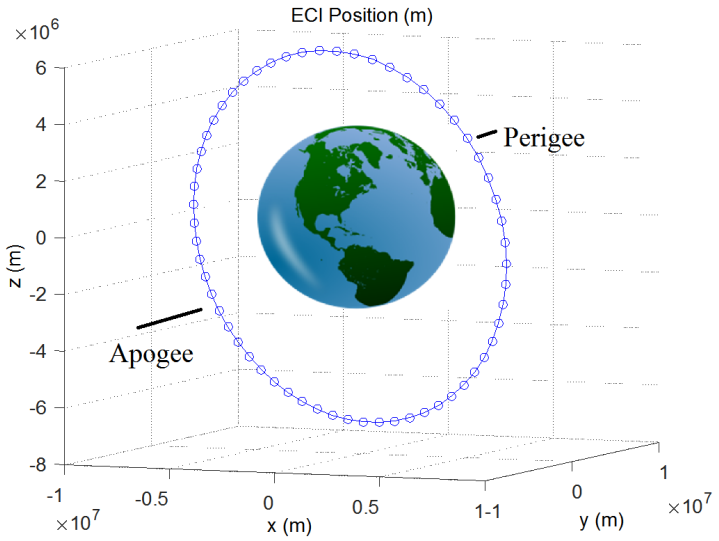
The alternate cosine sampling scheme

$$\tau_j = -\cos\frac{\pi(j+\frac{1}{2})}{M} \tag{2.23}$$

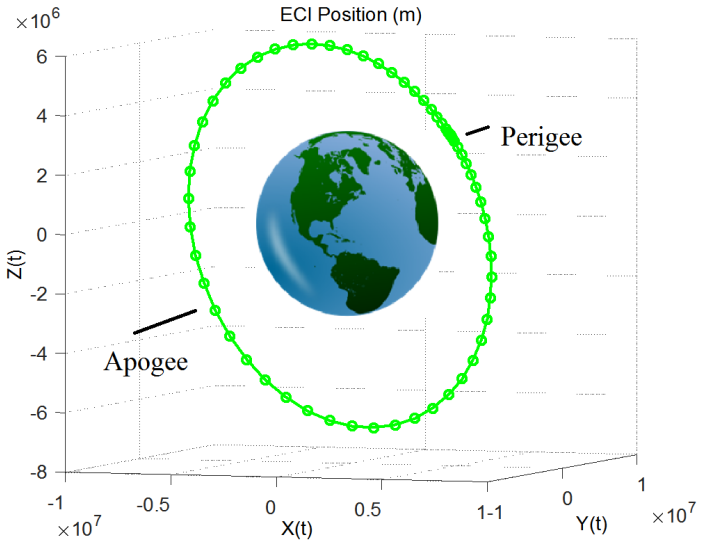
locates the zeros of the Chebyshev polynomials. It is possible to develop orthogonality conditions for either set of nodes (2.21) or (2.23). The CGL cosine nodes of Eq. (2.21) is preferred for prescribing exact terminal boundary conditions, since the zeros of Eq. (2.21) include the end points of the domain and this helps to reduce the Runge effect. The minus signs in Eqs. (2.21) and (2.23) are not universally adopted, but using this minus sign makes $\tau = -1$ correspond to τ_0, the left end of the time interval (this choice makes t and τ both increase as seen from "left to right").

2.5 Initial Value Problem

This section describes the MCPI algorithm for the initial value problem (IVP) and also gives a flow chart for an overview of the process. For more details on the

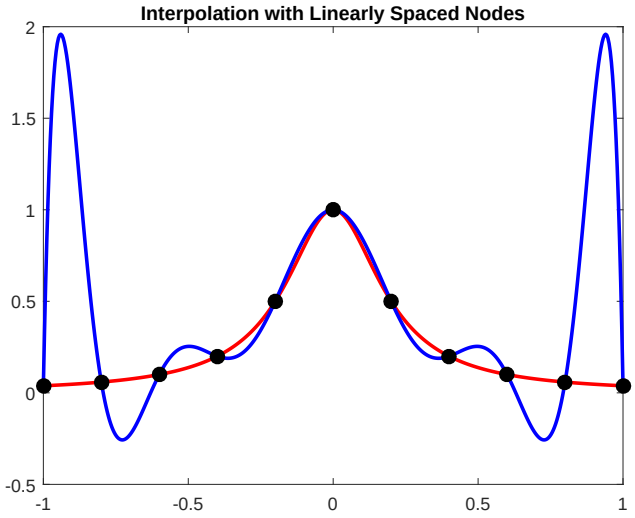


(a) Uniform Sampling

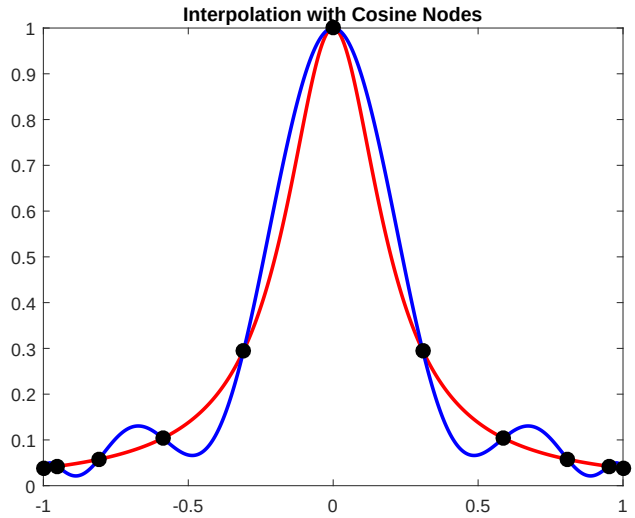


(b) Cosine Sampling

Figure 2.3: Comparison of Sampling Schemes Over One Orbit (60 Sample Points LEO). Note the Clustering of Cosine Sampling at Perigee and Sparsity of Sampling at Apogee.



(a) Runge effect for uniform sampled function approximation



(b) Reduction of Runge effect for cosine sampled function approximation

Figure 2.4: Example for mitigation of Runge effect using 11 cosine sampled nodes vs. 11 uniformly sampled nodes [57].

derivation for this method, see Bai [5], Bani Younes [68], Macomber [36, 39], and Woollands [62].*

Consider an ordinary differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}),\ t \in [t_0, t_f] \tag{2.24}$$

where the initial condition is specified to be $\mathbf{x}(t = t_0) = \mathbf{x}_0$, t_0 is the initial time, and t_f is the final time.

Since the Chebyshev polynomials are defined on the range from -1 to 1, the time variable in Eq. (2.24) must be transformed to lie on this range. A new linearly transformed time variable is introduced, which is specifed as $-1 \leq \tau \leq 1$.

The forward transformation from the time (t) domain to the τ domain is

$$t(\tau) = t_{min} + (\tau + 1)\frac{(t_{max} - t_{min})}{2} \tag{2.25}$$

It can be easily verifed that when $\tau = 1$, this transformation gives $t(1) = t_{max}$; for $\tau = -1$, $t(-1) = t_{min}$. The MCPI iterations are computed for this $t(\tau)$. Once the MCPI algorithm converges on a solution, the data can be mapped back into the original time domain using the inverse time transformation

$$\tau(t) = \frac{2(t - t_{min})}{(t_{max} - t_{min})} - 1 \tag{2.26}$$

The inverse transformation may be verified by checking from Eq. (2.26) the t which gives $\tau(t) = 1$ (which results in $t_{max} = t$) and $\tau(t) = -1$ (which results in $t_{min} = t$).

The transformation of Eq. (2.25) can be alternatively written to transform the

*Thank you also to Woollands for going into great detail in an internal MCPI tutorial document and set of codes [60].

time variable t to the new variable τ through the alternate slope/intercept form of the time transformation

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{2.27}$$

where obviously $t_0 = t_{min}$ and $t_f = t_{max}$. This equation may be substituted into Eq. (2.24) to obtain

$$\frac{d\mathbf{x}}{d\tau} = \frac{d\mathbf{x}}{dt}\frac{dt}{d\tau} \tag{2.28}$$

and then Eq. (2.24) is replaced by the transformed differential equation

$$\frac{d\mathbf{x}}{d\tau} = \mathbf{g}(\tau, \mathbf{x}) = \frac{t_f - t_0}{2}\mathbf{f}\left(\frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}, \mathbf{x}\right) \tag{2.29}$$

2.5.1 Standardized Algorithm for Cascade Solution of First and Second Order Differential Equations

This section provides a standardized approach to implementing MCPI. Bani Younes pointed out [6, 68] a more straightforward method was possible compared with the MCPI algorithm previously developed (see, for instance, PhD dissertations [5, 36, 68]); the present standardized method is a modest revision of Bani Younes' developments. In essence, these developments simply provide a more systematic approach to carry out the integrations of the cascade MCPI algorithm using vector-matrix operations.

To integrate the Chebyshev polynomials efficiently, an integration operator is introduced. For instance, the integration in the following expression (previously defined as the Picard equation)

$$\mathbf{x}^i(\tau) = \mathbf{x}(-1) + \int_{-1}^{\tau} \mathbf{g}(s, \mathbf{x}^{i-1}(s))ds = \mathbf{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{k=N-1} F_k T_k(s)ds \tag{2.30}$$

may be written in terms of matrices, given the number of cosine sample points M, as

$$\mathbf{x}^i(\tau) = \mathbf{x}(-1) + \int_{-1}^{\tau} [F]^T [T_{M1}]ds = \mathbf{x}(-1) + [F]^T \int_{-1}^{\tau} [T_{M1}]ds \tag{2.31}$$

The F_k Chebyshev polynomial coefficients are computed using a discrete orthogonality condition (see Appendix A) and may be moved outside of the integral because they are constants. The matrix of Chebyshev polynomials before and after the integration is related using

$$\int T_{M1}(\tau)ds = [I_M]T_{M2}\Big|_{x_0}^{\tau} \tag{2.32}$$

where the matrix of Chebyshev polynomials T_{M2} now contains one additional row than T_{M1} since integration increases the order. Acting on the suggestion of Professor Junkins, the integration operator matrix is introduced:

$$I_M = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{6} & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{8} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{6} & 0 & \frac{1}{10} & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{8} & 0 & \frac{1}{12} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2(M-2)} & 0 & \frac{1}{2M} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2(M-1)} & 0 & \frac{1}{2(M+1)} \end{pmatrix} \tag{2.33}$$

This operator makes use of the well-known integration property of Chebyshev polynomials:

$$\int_{-1}^{\tau} T_n(x)dx = \frac{1}{2}\left[\frac{T_{n+1}(x)}{n+1} - \frac{T_{n-1}(x)}{n-1}\right] \quad n \neq 0, 1 \tag{2.34}$$

Note that integration of the first two Chebyshev polynomials does not follow this pattern, and it is easily verified that

$$\int_{-1}^{\tau} T_0(x)dx = T_1 \tag{2.35}$$

$$\int_{-1}^{\tau} T_1(x)dx = \frac{1}{4}T_2 + \frac{1}{4}T_0 \tag{2.36}$$

The use of this integration operator greatly simplifies the MCPI integration process notationally compared with the previous derivations. For each integration, simply apply the integration operator matrix to perform the integration of the forcing

function. In order to account for evaluation of the integral at the limits in Equation (2.32), the MCPI integration from Equation (2.31) becomes

$$\mathbf{x}^i(\tau) = \mathbf{x}(-1) + [\mathbf{F}]^T[I_M]\Big([T_{M2}(\tau)] - [K]\Big) \tag{2.37}$$

Here, $[T_{M2}]$ is evaluated at the upper limit of τ and denoted simply as $[T_{M2}(\tau)]$; $[K] = [T_{M2}(-1)]$ denotes the lower limit (the initial condition) at $\tau = 1$. $[T_{Mi}]$ evaluated at the lower limit becomes a matrix of constants, with values of either 1 or -1. In addition, for given nodes, the matrix $[T_{M2}]$ and therefore $[I_M]\Big([T_{M2}(\tau)]-[K]\Big)$ may be computed a priori and in this manner does not impact the MCPI integration time. So, looking at Eq. (2.37), F^T at each node can simply be multiplied by a once-computed matrix and added to the initial condition to obtain each Picard iteration update of the nodal states. Thus, the vector matrix form is directly derived in a form suitable for programming. Figure 2.6 shows a flow chart of this vector form for the standardized version of MCPI. Figures 2.5, 2.6, and 2.7 are equivalent, with the exception that the standardized version (Figures 2.5 and 2.6) does not directly provide the β_k coefficients needed for interpolation.

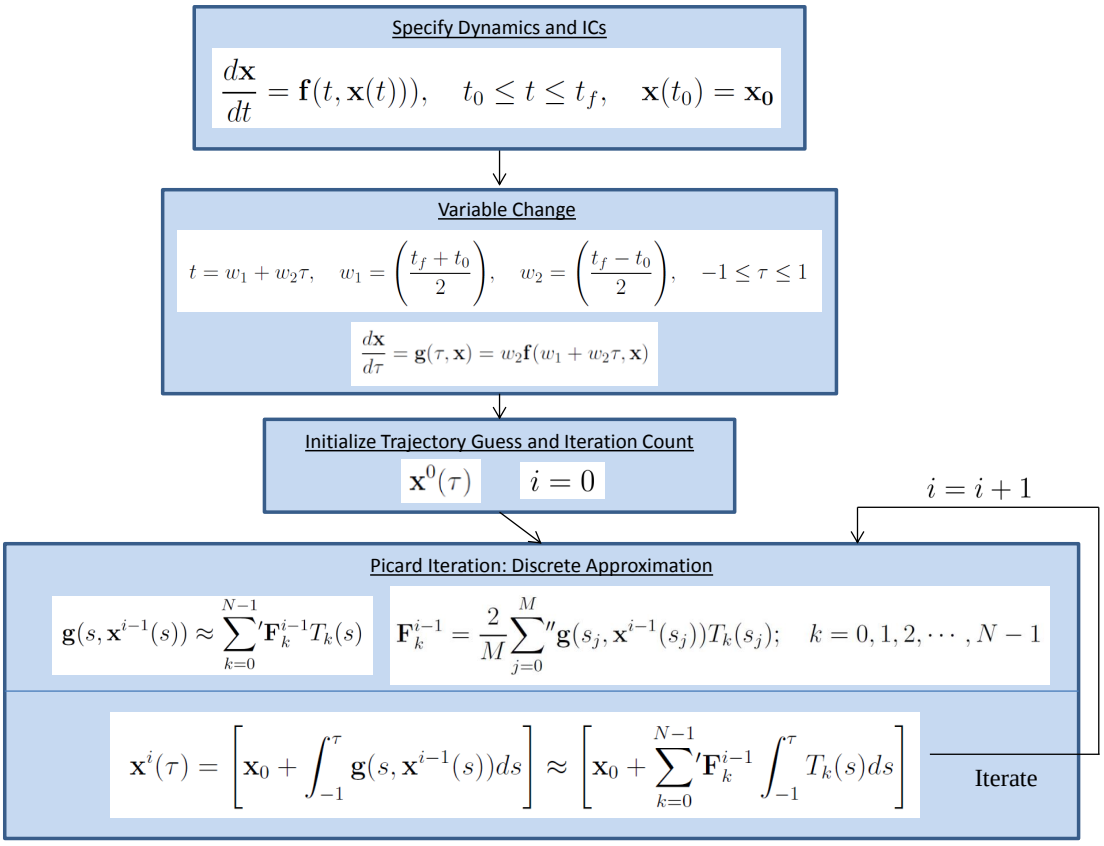


Figure 2.5: Flowchart for Standardized MCPI Algorithm (Initial Value Problem)

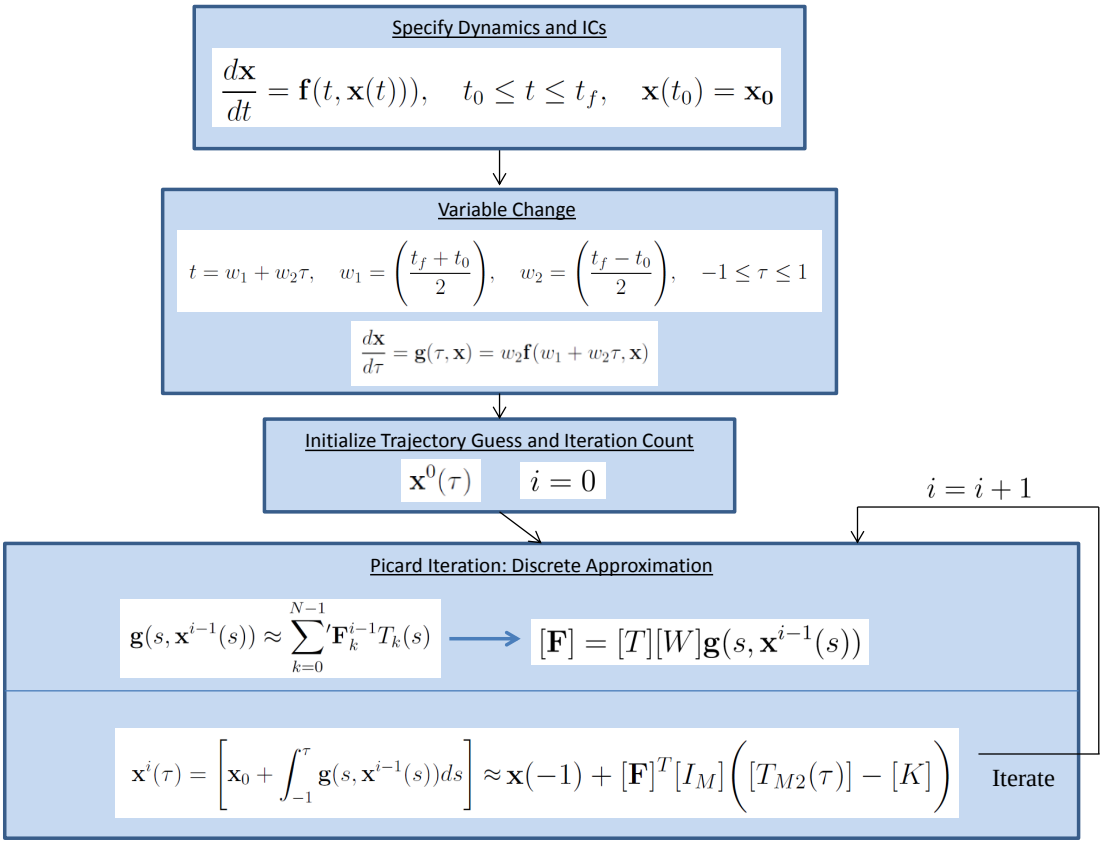


Figure 2.6: Flowchart for Vector Form of Standardized MCPI Algorithm (Initial Value Problem)

More generally, Eq. (2.37) can be used to interpolate $\mathbf{x}(\tau)$ at any time, which is especially useful because linear, constant interval ephemeris data is typically preferred in the current state of practice over the cosine sampled time nodes that MCPI utilizes to ensure orthogonality. The coefficients of the Chebyshev polynomials that represent the forcing function in Picard's equation of the form in Eq. (2.37) are updated according to the new approximated forcing function vector $\mathbf{g}$, since it is the only non-constant piece of the coefficient matrix (see Appendix A):

$$[\mathbf{F}] = [T][W]\mathbf{g} \tag{2.38}$$

Once the current values of Eq. (2.38) have been computed, the current (updated) approximation of the trajectory using Eq. (2.37) is found, and an efficient interpolation scheme provides linear sample points by first finding a β_k coefficient vector. These β_k coefficients give the coefficient of the Chebyshev polynomials needed to interpolate the state using the same method as described in Section 2.5.2.

The following sections present the MCPI formulation that has been traditionally used, for the reason that it more easily allows for connecting the results to the traditional developments in the existing literature.

2.5.2 Original Picard Iteration Expansion for IVP

For the IVP, a trajectory approximation is obtained by expanding the formula for Picard iteration in Eq. (2.3), repeated here for convenience, first on the left hand side (LHS) and then for the integrand on the right hand side (RHS) of the equation.

$$\mathbf{x}^i(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{f}(\tau, \mathbf{x}^{i-1}(\tau))d\tau, \quad i = 1, 2, ... \tag{2.39}$$

The LHS uses an N^{th} order sequence of Chebyshev polynomials to approximate the

i^{th} Picard estimate of the system states in terms of the k^{th} Chebyshev polynomial evaluated at the scaled time τ, or $T_k(\tau)$, and also the i^{th} approximation of the k^{th} state coefficient vector, or β_k^i. Here, n is the number of states in the state vector $\mathbf{x}(t)$, and there are $(N+1)$ β_k^i vectors of dimension $(n \times 1)$. The expanded state trajectory approximation on the LHS of Picard's equation is then:

$$\begin{aligned} \mathbf{x}^i(\tau) &\approx \sum_{k=0}^{N}{}'' \beta_k^i T_k(\tau) \\ &\approx \frac{1}{2}\beta_0^i T_0(\tau) + \beta_1^i T_1(\tau) + \beta_2^i T_2(\tau) + \cdots + \frac{1}{2}\beta_N^i T_N(\tau) \end{aligned} \tag{2.40}$$

Similarly, another Chebyshev polynomial sequence is used to approximate the $(i-1)^{th}$ Picard estimate of the integrand of the RHS of Eq. (2.3). In this case, the $(i-1)^{th}$ approximation of the k^{th} integrand ("force") coefficient vector is denoted as $\mathbf{F}_k^{i-1}$, which is of dimension $(n \times 1)$. It is important to note that an $(N-1)^{th}$-order Chebyshev sequence is used to approximate the RHS integral, while an N^{th}-order Chebyshev sequence is used to approximate the system states on the LHS. This is because integrating increases the order of the polynomial, thereby resulting in an N^{th}-order sequence on the RHS following the integration and allowing for kinematic consistency between the LHS and the RHS. For more details on the kinematically consistent derivation for MCPI IVP, please see [60, 62] and also Section 2.5.3.

The integrand on the RHS of Picard's equation is expanded as

$$\begin{aligned} \mathbf{g}(s, \mathbf{x}^{i-1}(s)) &\approx \sum_{k=0}^{N-1}{}' \mathbf{F}_{\mathbf{k}}^{\mathbf{i-1}} T_k(s) \\ &\approx \frac{1}{2}\mathbf{F}_0^{i-1} T_0(s) + \mathbf{F}_1^{i-1} T_1(s) + \mathbf{F}_2^{i-1} T_2(s) + \cdots + \mathbf{F}_N^{i-1} T_N(s) \end{aligned} \tag{2.41}$$

Now that Eq. (2.39) has been expanded in terms of the Chebyshev polynomials on both sides of the equation, this expression takes the form

$$\sum_{k=0}^{N}{}''\beta_k^i T_k(\tau) = \mathbf{x}(-1) + \sum_{k=0}^{N-1}\int_{-1}^{\tau}\left[\mathbf{F}_k^{i-1}T_k(s)\right]ds \tag{2.42}$$

where $\mathbf{x}^i(\tau) = \sum_{k=0}^{N}{}''\beta_k^i T_k(\tau)$ is the current state estimate. Because the integrand coefficients are constants, this equation may be rewritten as

$$\sum_{k=0}^{N}{}''\beta_k^i T_k(\tau) = \mathbf{x}(-1) + \sum_{k=0}^{N-1}\mathbf{F}_k^{i-1}\int_{-1}^{\tau}\left[T_k(s)\right]ds \tag{2.43}$$

The integrand coefficient vectors $\mathbf{F}_k^{i-1}$ may be solved using a least squares Chebyshev approximation formulation through evaluation of the integrand function of Eq. (2.41), where the forcing function approximation $\mathbf{g}$ is assumed to be known (for the current approximation) at the current Picard iteration:

$$\mathbf{F}_k^{i-1} = \frac{2}{M}\sum_{j=0}^{M}{}''\mathbf{g}(s_j, \mathbf{x}^{i-1}(s_j))T_k(s_j) \tag{2.44}$$

These coefficients essentially minimize the residual error for the least squares formulation. Note that this equation is an inner product of the acceleration (from the equations of motion) with the k^{th} orthogonal Chebyshev basis function and effecively projects the true system dynamics, to a high degree of approximation, onto a finite dimensional basis set that is valid along the $(i-1)^{th}$ Picard iteration trajectory.

2.5.2.1 MCPI Flow Chart

Now that the original MCPI algorithm has been discussed, a summary of the MCPI algorithm is given in Figure 2.7. First, the dynamics and initial conditions are specified. Next, a variable change is performed so that the integration takes place

in the τ domain, where the Chebyshev polynomials are defined. Then the iteration count is initialized and the initial trajectory guess is specified (if no knowledge of the trajectory is applied, this can be a matrix of ones or zeros; instead a "warm start" such as the unperturbed two-body solution may be applied to decrease the computation time of the perturbed solution). Next, the forcing function is computed from the current approximation of the state. This allows for a solution of the Chebyshev polynomial coefficients that represent the forcing function term on the RHS of Picard's equation. The coefficients on the LHS of Picard's equation may then be found from the RHS coefficients; note that this step is necessary in order to obtain the linearly interpolated result. If linear interpolation is not required, the flow chart in Figure 2.5 may be used instead for a more straightforward implementation. Once the LHS coefficients have been found, this gives an update for the state approximation. The iteration count is increased and Picard's equation is applied again, and this process is repeated until convergence is achieved.

Note that, for the standardized MCPI algorithm as presented in Section 2.5.1, the β_k coefficients do not need to be computed unless interpolation is required, because the updated $\mathbf{F}_k$ coefficients allow for an immediate update of the state estimate using the RHS of the Picard equation. The flow charts in Figures 2.5 and 2.7 may be compared to see how the standardized version of MCPI simplifies MCPI conceptually but does not include computation of the β_k coefficients.

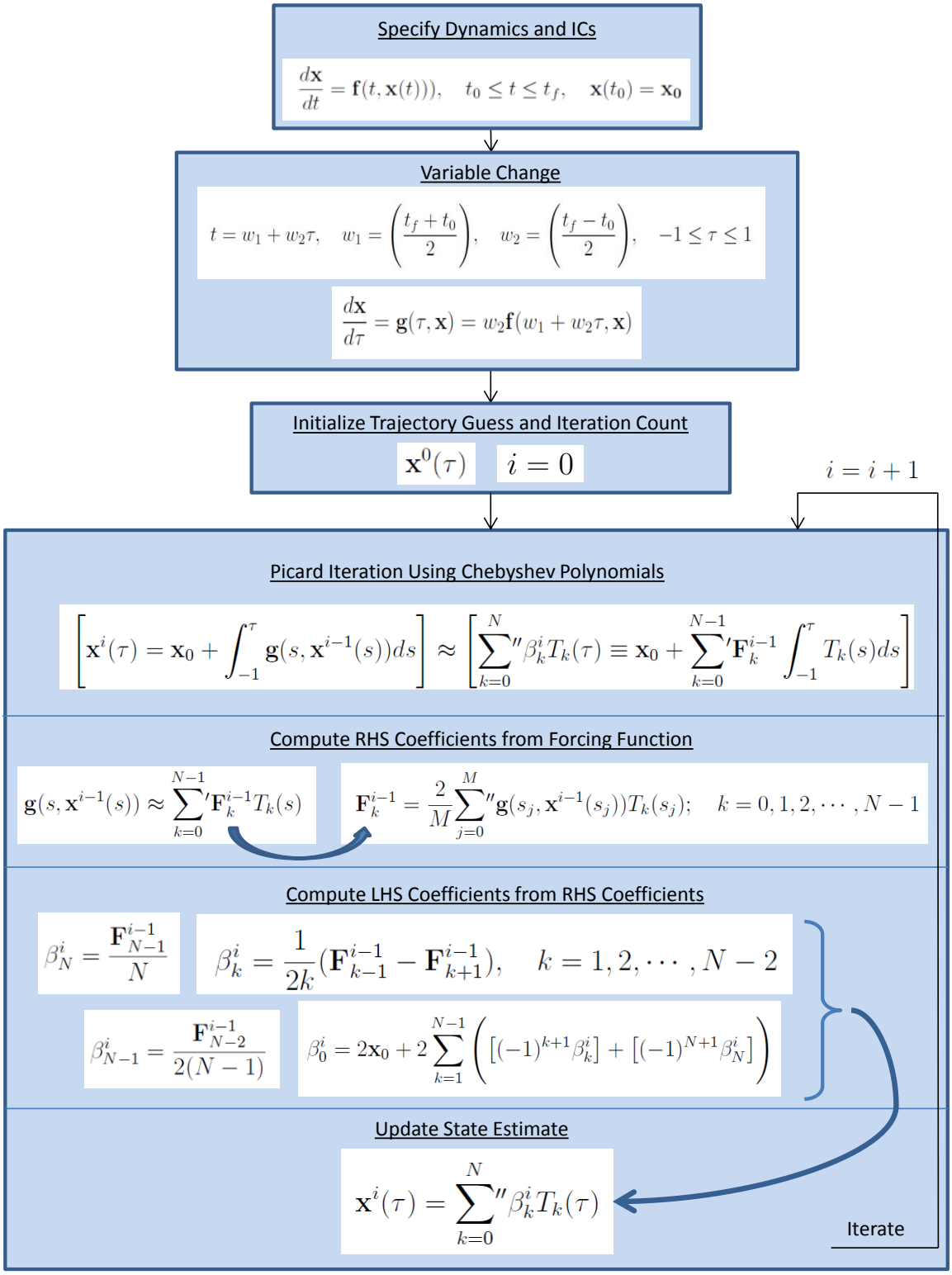


Figure 2.7: Flowchart of MCPI Algorithm for Initial Value Problem

2.5.3 Examples: Standardized Algorithm vs. Original MCPI

To show that the standardized MCPI algorithm is effectively the same as the original MCPI formulation, two examples are given. The first example shows symbolically and the second example shows numerically that the same result is obtained using both methods.

2.5.3.1 Symbolic Example of Integration Operator vs. Original MCPI

Let's consider an IVP problem that has already been formulated using MCPI and rederive it using the newly-developed method; see Section D.1 of Brent Macomber's dissertation [36] for a simple first-order MCPI example with 5^{th} order ($N = 5$) Chebyshev polynomials. Recall the Picard iteration formula in Eq. (2.3) for this scalar case:

$$x^i(\tau) = x_0 + \int_{-1}^{\tau} g(s, x^{i-1}(s))ds \quad , \quad i = 1, 2, ... \tag{2.45}$$

As described in Section 2.5.2, the LHS of this equation is approximated using Chebyshev polynomials as

$$\begin{aligned} x^i(\tau_j) &\approx {\sum_{k=0}^{N}}'' \beta_k^i T_k(j) \\ &= \frac{1}{2}\beta_0^i T_0(\tau_j) + \beta_1^i T_1(\tau_j) + \beta_2^i T_2(\tau_j) + \cdots + \frac{1}{2}\beta_N^i T_N(\tau_j) \end{aligned} \tag{2.46}$$

Similarly, the integrand of the RHS of the Picard iteration equation may be approximated by

$$g(s_j, x^{i-1}(s_j)) \approx \sum_{k=0}^{N-1}{}' F_k^{i-1} T_k(s_j)$$
$$= \frac{1}{2} F_0^{i-1} T_0(s_j) + F_1^{i-1} T_1(s_j) + F_2^{i-1} T_2(s_j) + \cdots + F_{N-1}^{i-1} T_{N-1}(s_j) \tag{2.47}$$

Note that in this equation, the summation's upper limit is $(N-1)$ because integration of the integrand $g(s_j, x^{i-1}(s_j))$ will increase the order by one to the maximum of N. This leaves the LHS and the RHS with the same polynomial degree (upper limit of N) for algebraic and kinematic consistency. The same number of CGL sample points M is used for both approximations, which is set equal to the order of the Chebyshev fit of the states ($M = N$). Therefore, the force approximation of the RHS uses a least squares fit, and upon integration it has degree N, which is appropriate for interpolation. Also, the integrand is a function of the previous MCPI iteration's state x^{i-1} and Chebyshev coefficients F_k^{i-1}, while in contrast the LHS of the Picard equation is a function of the current MCPI iteration's state x^i and Chebyshev coefficients β_k^i. During each i^{th} Picard iteration, the coefficients of the integrand approximation may be calculated directly using a least squares Chebyshev approximation following the evaluation of the integrand $g(s_j, x^{i-1}(s_j))$ at the current best estimate of the system state x^{i-1}:

$$F_k^{i-1} = \frac{2}{M} \sum_{j=0}^{M}{}'' g(s_j, x^{i-1}(s_j)) T_k(s_j) \tag{2.48}$$

Substituting Eqs. (2.46) and (2.47) into (2.45) gives a new form of the Picard iteration formula, where the initial condition is $x_0 = x(-1)$ because the time span is scaled to $[-1, 1]$, the domain in which the Chebyshev polynomials exist:

$$x^i(\tau) = \sum_{k=0}^{N}{}'' \beta_k^i T_k(\tau) = x(-1) + \int_{-1}^{\tau} \left[\sum_{k=1}^{N-1}{}' F_k^{i-1} T_k(s) \right] ds \tag{2.49}$$

After integrating the RHS of the Picard iteration formula, historically the coefficients have been equated of like index basis functions, which gives the unknown Chebyshev coefficients β_k^i in terms of the known integrand coefficients F_k^{i-1}. Both sides of the equation would be expanded, the Chebyshev polynomials would be subsituted in, and then the terms would be grouped according to the k^{th} Chebyshev basis functions (i.e., each power of τ). This method is given by Macomber [36].

Alternatively, using the integration operator method allows for the more straightforward solution that is summarized below with few mystical steps of unusual-looking inner products as seen in the traditional developments. The updated estimate of the state trajectory is obtained by first integrating the integrand, then adding the initial condition as follows.

$$\int_{-1}^{\tau} g(s, x(s)) ds = [F]^T [I_M][T_{M2}(\tau)] - [F]^T [I_M][T_{M2}(-1)] \tag{2.50}$$

where the two separate expressions on the RHS are evaluated at the limits of the integral, τ and -1, and as previously denoted in Section 2.5.1, $[T_{M2}(-1)] = [K]$. To verify the equivalence, this equation can be expanded by explicitly writing out the matrices (recall $N = M = 5$ for this example).

$$\int_{-1}^{\tau} g(s,x(s))ds = \begin{bmatrix} \frac{1}{2}F_0 & F_1 & F_2 & F_3 & F_4 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{6} & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{8} & 0 \\ 0 & 0 & 0 & -\frac{1}{6} & 0 & \frac{1}{10} \end{bmatrix} \begin{bmatrix} T_0(\tau) \\ T_1(\tau) \\ T_2(\tau) \\ T_3(\tau) \\ T_4(\tau) \\ T_5(\tau) \end{bmatrix}$$
$$- \begin{bmatrix} \frac{1}{2}F_0 & F_1 & F_2 & F_3 & F_4 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{6} & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & 0 & \frac{1}{8} & 0 \\ 0 & 0 & 0 & -\frac{1}{6} & 0 & \frac{1}{10} \end{bmatrix} \begin{bmatrix} T_0(-1) \\ T_1(-1) \\ T_2(-1) \\ T_3(-1) \\ T_4(-1) \\ T_5(-1) \end{bmatrix} \tag{2.51}$$

Multiplying out the integration operator matrix and the Chebyshev terms results in

$$\int_{-1}^{\tau} g(s,x(s))ds = \begin{bmatrix} \frac{1}{2}F_0 & F_1 & F_2 & F_3 & F_4 \end{bmatrix} \begin{bmatrix} T_1(\tau) \\ \frac{1}{4}T_0(\tau) + \frac{1}{4}T_2(\tau) \\ -\frac{1}{2}T_1(\tau) + \frac{1}{6}T_3(\tau) \\ -\frac{1}{4}T_2(\tau) + \frac{1}{8}T_4(\tau) \\ -\frac{1}{6}T_3(\tau) + \frac{1}{10}T_5(\tau) \end{bmatrix} - \begin{bmatrix} \frac{1}{2}F_0 & F_1 & F_2 & F_3 & F_4 \end{bmatrix} \begin{bmatrix} T_1(-1) \\ \frac{1}{4}T_0(-1) + \frac{1}{4}T_2(-1) \\ -\frac{1}{2}T_1(-1) + \frac{1}{6}T_3(-1) \\ -\frac{1}{4}T_2(-1) + \frac{1}{8}T_4(-1) \\ -\frac{1}{6}T_3(-1) + \frac{1}{10}T_5(-1) \end{bmatrix} \tag{2.52}$$

Now, including the initial condition $x(-1)$ and mutliplying the remaining terms gives an expression for the RHS of the Picard iteration formula.

After these steps, the expression gives the current approximation of the state at cosine sample nodes, and no further calculations are required. The results are identical to those obtained using the historial approach to MCPI. However, typically linear time spacing is desired, rather than cosine sampling. For the appropriate conversion, an expression relating the RHS coefficients with the LHS coefficients is required as shown in previous publications. To complete this step, the above expression is equated to the LHS, which is expanded simply to be

$$LHS = \frac{1}{2}\beta_0 T_0 + \beta_1 T_1 + \beta_2 T_2 + \beta_3 T_3 + \frac{1}{2}\beta_4 T_4 \tag{2.53}$$

Equating this to Eq. (2.52) and collecting terms onto $T_k(\tau)$ immediately gives the $\beta_k' s$ as a function of the $F_i' s$:

$$\boxed{\beta_N = \frac{1}{N} F_{N-1}} \tag{2.54}$$

$$\boxed{\beta_{N-1} = \frac{1}{2(N-1)} F_{N-2}} \tag{2.55}$$

$$\boxed{\beta_k = \frac{1}{2k}(F_{k-1} - F_{k+1})} \tag{2.56}$$

$$\boxed{\beta_0 = 2x(-1) + 2(\beta_1 - \beta_2 + \beta_3 - \beta_4) + \beta_5} \tag{2.57}$$

Using these expressions for the β_i coefficients gives general expressions for the i^{th} state trajectory, which is in turn used in the integrand for the next Picard iteration. Each $(i-1)^{th}$ approximation of the integrand coefficients F_k is found by utilizing an $(N-1)^{th}$ order least squares Chebyshev approximation of the integrand, which is evaluated from the current best estimate of the state. In summary, the Picard iteration method calculates the i^{th} approximation of the state coefficients using the general expressions for β_k given above. This algorithm may be written in a vector-matrix form for coding simplicity, as given in detail in [5, 36, 62, 68].

2.5.3.2 Numerical Example of Integration Operator vs. Original MCPI

The following example was developed by Texas A&M PhD student Robyn Woollands [60, 62] to verify the traditional MCPI method, using the equation (for scaling parameter $\epsilon = 0.01$)

$$\dot{x} = \epsilon x \tag{2.58}$$

This code is used to simply confirm that the standardized algorithm and the nu-

merical simulation give the same result for a first-order system, as given in Figures (2.8) and (2.9). In this case, "MCPI Old" specifies the original MCPI formulation that was given in Section 2.5.2, while "MCPI New" specifies the standardized MCPI algorithm presented in Section 2.5.1. For an additional example comparing these two methods symbolically, please see Subsection 2.5.3.

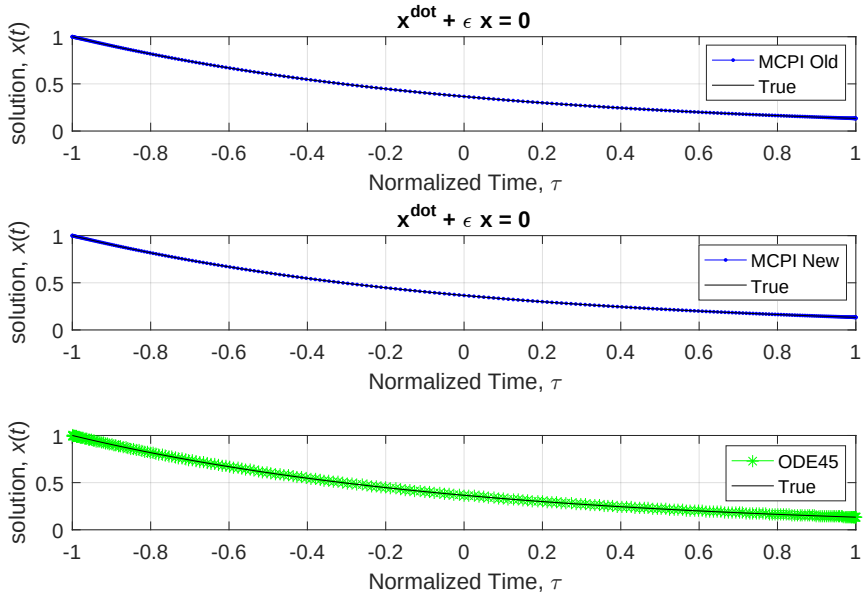


Figure 2.8: First Order Example State Comparison for Three Integrators

Figure 2.8 gives the state solution as a function of time vs. the known analytical solution for each of the two MCPI implementations. Figure 2.9 provides the norm of the error vs. time for each of the two MCPI implementations. For reference, provided in the last subplot of the first figure is the solution for, and in the second figure is the residual for, an optimized RK45 algorithm that is a Matlab built-in function called *ode45*.

As is evident, the new formulation and the resulting algorithm is graphically

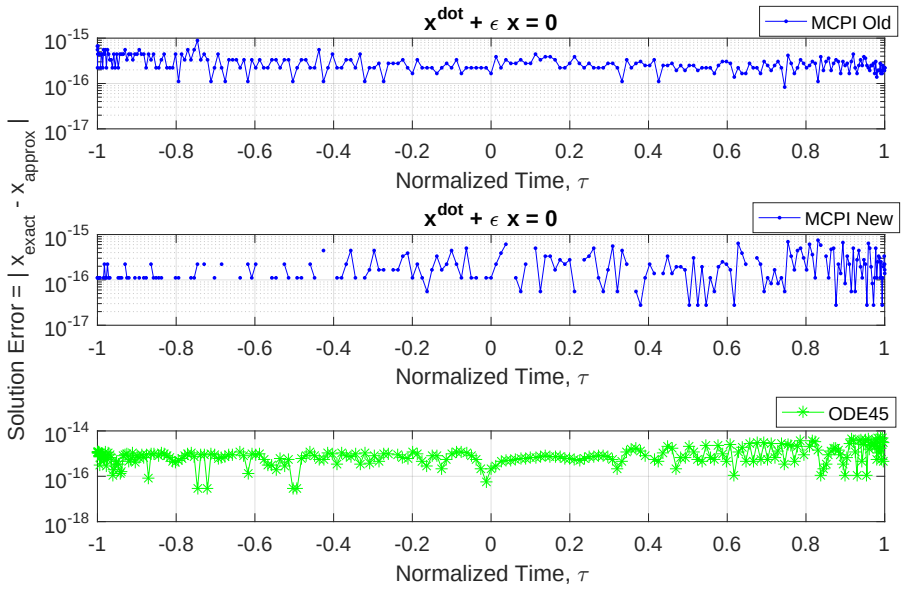


Figure 2.9: First Order Example Error Comparison for Three Integrators

validated, and the numerical precision is found to agree to better than 14 digits of accuracy. This 15-digit accuracy obviously exceeds "engineering" accuracy and also exceeds the typical accuracy achievable (with 64 bit arithmetic using a familiar explicit integrator). Also, the runtime changes negligibly between the two MCPI algorithms. This first order derivation and algorithm, however, are more efficiently extended to the cascade algorithms appropriate for second and higher order differential equations for the initial value problem. The key point is that the developments in this section, using the integration operator of Eq. (2.33), is more compact and algebraically much easier to extend to the case of second and higher order differential equations.

2.6 Boundary Value Problem

The MCPI BVP formulation is limited to about $\frac{1}{3}$ of an orbit in Cartesian coordinates, and through a KS regularization the convergence domain may be extended

to nearly a full orbit [62, 63]. This algorithm is not a shooting method in that it enforces the boundary conditions at every Picard iteration, then updates the differential equations accordingly on the next iteration. In contrast, a traditional shooting method propagates the differential equations, computes the miss distance, then updates the initial conditions on the following iteration to better achieve the target state.

Due to the nature of the nonlinear relationship between Cartesian coordinates and the Modified Equinoctial Orbital Elements (MEEs, which are the focus of this dissertation), it is shown that it is beneficial to incorporate a low-thrust controls problem for the MEEs using a shooting method rather than solving Lambert's problem using this element set. The low-thrust developments using the MEE variation of parameters formulation is shown to be an excellent setting for both direct and indirect approaches to optimal low-thrust orbit transfers. Though a shooting method is used for this work, for completeness the first and second order BVP formulation for MCPI will be discussed. For more complete details of the latest formulation of the kinematically consistent BVP for MCPI, please refer to Woollands' dissertation [62].*

2.6.1 First Order BVP MCPI

For the BVP, user-specified boundary conditions must be enforced. The initial and final states may be represented using Chebyshev polynomials as

$$x(-1) = \sum_{k=1}^{N}{}'' \beta_k T_k(-1) \tag{2.59}$$

*This development comes from Woollands' internal tutorial for the MCPI BVP [61]

$$x(1) = \sum_{k=0}^{N}{}'' \beta_k T_k(1) \tag{2.60}$$

These equations may be expanded to solve for the first two β_k coefficients, β_0 and β_1, which enforce the boundary conditions in the trajectory approximation on the LHS of Picard's equation, as a function of $\beta_2, \beta_3, \cdots \beta_N$. The $\beta_2, \cdots \beta_N$ coefficients are determined from the Picard integral.

2.6.2 Second Order BVP MCPI

Similar to the first order BVP MCPI formulation, the first and second coefficients must be found; because it is now a second order system, expressions for the α_k coefficients are found because the β_k coefficients now denote the velocity-level coefficients:

$$x(-1) = \sum_{k=1}^{N}{}'' \alpha_k T_k(-1) \tag{2.61}$$

$$x(1) = \sum_{k=0}^{N}{}'' \alpha_k T_k(1) \tag{2.62}$$

Expanding these expressions gives α_0 and α_1 in terms of the boundary conditions and other α_k coefficients. The velocity approximation may then be represented as a function of velocity-level β_k coefficients (and forcing function F_k coefficients) - see Eqs. (2.30) and (2.38). A relationship may be built between these three sets of coefficients and formed into a vector-matrix formulation that is more conducive to numerical computations.

As previously stated, the control problem presented in this dissertation does not use this MCPI BVP formulation because the MCPI BVP is limited to about $\frac{1}{3}$ of an orbit for Cartesian coordinates. Please see Chapters 6 and 7 for information about

solving the BVP using a shooting-like method known as the Method of Particular Solutions, which allows for multi-rev transfers.

2.7 Chapter Summary

This chapter describes the Modified Chebyshev Picard Iteration (MCPI) algorithm. First, a description of Picard iteration and Chebyshev polynomials is given; a combination of these two concepts forms the MCPI integration method. Combining cosine sampling with the Chebyshev-Guass-Lobatto Nodes reduces the Runge effect often seen in function approximation. Formulations for the initial value problem and the boundary value problem are presented; the boundary value problem solved in Chapter 7 uses a shooting method, but the MCPI formulation for the BVP is presented here for completeness. A standardized algorithm is presented for MCPI that allows for a cascade solution of first- and second-order differential equations that can easily be extended to higher-order systems. The MCPI integration method described in this chapter is used throughout this dissertation.

Many of the developments are agnostic with regard to the method used to solve the differential equations. However, the MCPI methods fuse naturally with these developments and lead to excellent accuracy and efficiency.

3. MODIFIED EQUINOCTIAL ORBITAL ELEMENTS*

3.1 Introduction

While previous studies show that MCPI is a powerful tool used to propagate the position and velocity of orbital motion, the results given in this chapter show that using orbital elements (Gauss' Variation of Parameters approach) to propagate the perturbed two-body state vector reduces the number of MCPI iterations and nodes required, which is especially useful for reducing the computation time when including computationally-intensive calculations such as a high degree spherical harmonic gravity model. Also, Picard iteration (MCPI) converges for 5 to 10 times as many revolutions (using a single segment) when compared with Cartesian propagation. Results for the Classical Orbital Elements (COE) and the Modified Equinoctial Orbital Elements (MEE) show that state propagation using the osculating MEEs as the basis for the Variation of Parameters equations is inherently well-suited to MCPI orbit propagation. Additional benefits are achieved using a patched time segmentation scheme [48, 49], allowing an arbitrarily long propagation.

A set similar to the Modified Equinoctial Orbital Elements (MEE) was used more than a century ago by Lagrange to approximate secular perturbation effects due to planetary gravitational perturbations while considering orbits with small eccentricities and inclinations. Lagrange used approximate analytical integration of the Variation of Parameters equations, in contrast with the high precision numerical methods for the present work. Broucke and Cefola [8] showed the original Lagrangian

*Part of the data reported in this chapter is reprinted with premission of ICCES/CMES from "Efficient Orbit Propagation of Orbital Elements Using Modified Chebyshev Picard Iteration Method," which was published both as a conference proceeding (Proceedings of ICCES 2015 Conference, Reno, NV, July 2015) and a journal article (Journal of Computer Modeling in Engineering & Sciences (CMES): Special Issue on Computational Methods in Celestial Mechanics, Vol. 111, No. 1, January 2016) [48, 49]

Equinoctial Elements set to be relatively free of singularities for zero eccentricities and both zero and ninety degree inclinations (the equations become singular as $i \rightarrow 180^\circ$ for the standard MEE formulation), and they also developed a large number of properties and equations for the set (for elliptical and hyperbolic flight). For the case of retrograde orbits (approaching 180° inclinations), a modified version of the MEE equations are given by Brouke and Cefola but are still singularity-free and well-behaved for $(180^\circ - i) \leq 90^\circ$.

Brouwer and Clemence [9] discussed the Variation of Parameters differential equations with several orbital element sets. The Equinoctial Orbital Elements as defined by Broucke and Cefola are similar to the Set III elements (which are represented by non-integrable differential relations) discussed by Brouwer and Clemence, and they utilize the h and k elements introduced below.

Shaver integrated orbital motion using Cartesian coordinates as well as Equinoctial Orbital Elements [55]. He noted that the smooth nature of the element rates makes this set of elements easy to approximate with low-order Chebyshev polynomial series, and that using the Variation of Parameters formulation leads to convergence in significantly fewer iterations. A drag model and low order gravity model were included in these results. Another previous study using only the J2 gravity term concluded that using MEEs gives a more accurate solution than Classical Orbital Elements [25]. This chapter expands Shaver's results on modern processors and incorporates a high order gravity model to gain insight into the convergence domain and accuracy of using orbital elements.

The MEEs use a variation of the original Equinoctial Orbital Elements that allows them to be nonsingular for perturbed elliptical orbits and approach the parabolic case except near 180° and are defined in terms of the Classical Orbital Elements in Equations (3.1) - (3.6) [58, 59].

$$p = a(1 - e^2) \tag{3.1}$$

$$f = ecos(\omega + \Omega) \tag{3.2}$$

$$g = esin(\omega + \Omega) \tag{3.3}$$

$$h = tan\left(\frac{i}{2}\right) cos(\Omega) \tag{3.4}$$

$$k = tan\left(\frac{i}{2}\right) sin(\Omega) \tag{3.5}$$

$$L = \Omega + \omega + \nu \tag{3.6}$$

where p is the semilatus rectum, a is the semimajor axis, e is the eccentricity, ω is the argument of perigee, Ω is the right ascension of the ascending node, i is the inclination, ν is the true anomaly, and L is the true longitude. Note that h and k are singular for $i \to 180°$. For the most common orbits, $0 < i < 120°$, these coordinates are well behaved. The Equinoctial reference frame is shown in Figure (3.1). This reference frame is an inertial orbit frame for zero perturbations and is slowly varying for small perturbations. Elements f and g are the $\hat{\mathbf{f}}$ and $\hat{\mathbf{g}}$ unit vector components, respectively, of the eccentricity vector in the equinoctial reference frame. Similarly, elements h and k are the $\hat{\mathbf{f}}$ and $\hat{\mathbf{g}}$ components, respectively, of the ascending node unit vector in the equinoctial reference frame [14]. The inverse relationship is

$$\Omega = tan^{-1}\left(\frac{k}{h}\right), \quad at\ k = 0, h = 0 : \Omega = 0 \tag{3.7}$$

$$\bar{\omega} \equiv \omega + \Omega = tan^{-1}\left(\frac{g}{f}\right), \quad at\ g = 0, f = 0 : \bar{\omega} = 0 \tag{3.8}$$

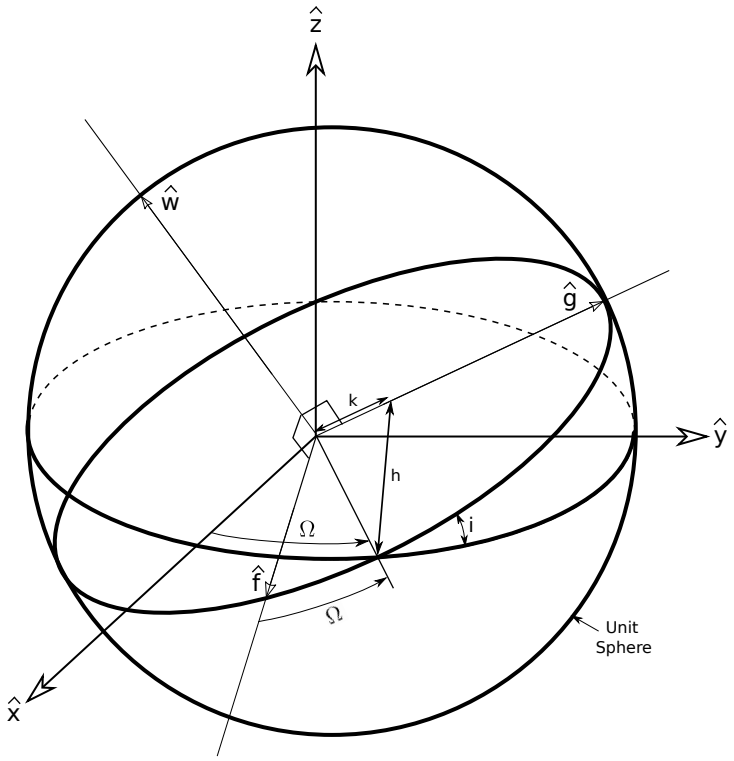


Figure 3.1: Equinoctial Reference Frame

$$\omega = \bar{\omega} - \Omega \tag{3.9}$$

$$i = 2tan^{-1}(\sqrt{h^2 + k^2}) \tag{3.10}$$

$$e = \sqrt{f^2 + g^2} \tag{3.11}$$

$$a = \frac{p}{(1 - e^2)} \tag{3.12}$$

$$\nu = L - \bar{\omega} \tag{3.13}$$

Some singularities may be encountered for ω and Ω when $i \to 0, 180^\circ$ and $e \to 0$. Specifically, when $i \to 0$, $\Omega = \frac{0}{0}$ and when $e \to 0$, $\omega = \frac{0}{0}$. Also, when $e \to 1$, $p \to 0$, then a becomes singular. The $i \to 0$ singularities in Eq. (3.7), (3.8) are easily eliminated as shown.

Inclination is very weakly perturbed by the dominant natural perturbations (gravity and drag), so it is easy to avoid the inclination singularity, and Eqs. (3.1) - (3.6) and their inverse (3.7) - (3.13) are for most practical purposes singularity-free. However, if man-made (i.e., thrust) perturbations move the osculating orbit through $i = 180^\circ$ (equatorial, counter-clockwise retrograde orbit), the Variation of Parameters formulation based upon these coordinates will be poorly behaved and alternate coordinates may be needed. The simplest fix is to rotate about 180° to a new inertial frame with the z-axis toward the south pole. With this variation, the actual 180° inclination effectively has zero inclination within the altered inertial frame. Reference orbits near 180° inclination are exceedingly rare, so this case is not considered in this dissertation.

In contrast with the original Equinoctial Orbital Elements set, the MEE set utilizes p, the semilatus rectum instead of a, the semimajor axis and also L, the true longitude instead of λ, the mean longitude. One benefit of this element set is that p is defined when approaching parabolic orbits, whereas $a \to \infty$ for the near parabolic case. This nearly singularity-free equinoctial formulation utilizes the longitudes λ, F, L instead of the classical anomalies Mean Anomaly, Eccentric Anomaly, and True Anomaly, M, E, ν respectively [14]:

$$\lambda = M + \omega + \Omega \tag{3.14}$$

$$F = E + \omega + \Omega \tag{3.15}$$

$$L = \nu + \omega + \Omega \tag{3.16}$$

In this formulation, it is advantageous to write Kepler's equation in terms of the

eccentric longitude F, rather than the eccentric anomaly E, to compute the position vector. This equation and the corresponding radius vector may then be written as

$$\lambda = F + gcos(F) - fsin(F) \tag{3.17}$$

$$r = a[1 - gsin(F) - fcos(F)] \tag{3.18}$$

These quantities remain well-defined for the cases of circular or equatorial orbits, eliminating such singular cases that exist for the Classical Orbital elements. The radius may alternatively be written as

$$r = \frac{p}{1 + fcos(L) + gsin(L)} \tag{3.19}$$

The transformation from Classical Orbital Elements and Modified Equinoctial Elements is given in Equations (3.1) - (3.6), and the inverse transformation is easily derived [25]. The integration of perturbed orbits requires the transformation between orbital elements and ECI Cartesian coordinates in order to compute the perturbing acceleration, and this transformation between equinoctial frame and ECI frame (and vice versa) is given in detail by Cefola and Broucke in [10]. Analogously to the Classical Orbital Elements case, the three cartesian coordinates (x, y, z) may be obtained by premultiplying the coordinates relative to the equinoctial frame by the direction cosine matrix [8]

$$[NE] = \frac{1}{1 + k^2 + h^2} \begin{bmatrix} 1 - k^2 + h^2 & 2kh & 2k \\ 2kh & 1 + k^2 - h^2 & -2h \\ -2k & 2h & 1 - k^2 - h^2 \end{bmatrix} \tag{3.20}$$

Note that this matrix is a function of two variables, rather than three variables as in the most general rotation direction cosine matrix. Note that Eq. (3.20) is a special case of the Euler-Rodriguez parameterization of an orthogonal matrix [54]. This observation does not appear in the literature and suggests that related coordinates such as the Modified Rodriquez parameters (which are nonsingular over $\pm 360^\circ$ range, not just $\pm 180^\circ$ as are the Classical Rodriguez Parameters) be explored as alternatives [54]. The even larger non-singular range and other advantages may result in yet another and more attractive set of Equinoctial Elements.

3.2 Gauss' Equations

For this study, Gauss' equations for the variation of the Modified Equinoctial Elements are preferred since they are more general than Lagrange's original Variation of Parameters, which hold for the classical elements subject to conservative perturbations [53]. The fusion of the MEEs with the MCPI propagation method leads to an enlarged domain of convergence and greater efficiency. As shown in previous publications by the author [48, 49], these elements increase the domain of MCPI convergence over using either Cartesian coordinates or the Classical Orbital Elements, and also reduces the number of sample nodes, MCPI iterations, and gravity function calls compared with the Cartesian case. The chosen Variation of Parameters equations are [58, 59]

$$\frac{dp}{dt} = \frac{2pa_\theta}{w}\sqrt{\frac{p}{\mu}} \tag{3.21}$$

$$\frac{df}{dt} = \sqrt{\frac{p}{\mu}}\left\{a_r S sin(L) + \frac{\big[(w+1)cos(L)+f\big]a_\theta}{w} - \frac{g\big[hsin(L)-kcos(L)\big]a_h}{w}\right\} \tag{3.22}$$

$$\frac{dg}{dt} = \sqrt{\frac{p}{\mu}} \left\{ -a_r cos(L) + \frac{\big[(w+1)sin(L) + g\big]a_\theta}{w} + \frac{f\big[hsin(L) - kcos(L)\big]a_h}{w} \right\} \tag{3.23}$$

$$\frac{dh}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 a_h}{2w} cos(L) \tag{3.24}$$

$$\frac{dk}{dt} = \sqrt{\frac{p}{\mu}} \frac{s^2 a_h}{2w} sin(L) \tag{3.25}$$

$$\frac{dL}{dt} = \sqrt{\mu p} \left(\frac{w}{p}\right)^2 + \sqrt{\frac{p}{\mu}} \frac{\big[hsin(L) - kcos(L)\big]a_h}{w} \tag{3.26}$$

where $s^2 = 1 + h^2 + k^2$, $w = \frac{p}{r} = 1 + fcos(L) + gsin(L)$ and a_r, a_θ, a_h are the components of the perturbing acceleration in the directions along the radius vector outward, perpendicular to the radius vector in the direction of motion, and normal to the orbital plane in the direction of the angular momentum vector, respectively. The acceleration is computed using Cartesian coordinates, so during every Picard iteration, the MEEs must be transformed to position and velocity. This also allows for direct computation of the STM for Cartesian coordinates, as given in Chapter 4, since no additional transformations are needed.

3.3 Simulation: Increased Domain of Convergence

Simulation results are obtained on a Windows 8 machine using 64-bit arithmetic, Matlab R2013a, where all MCPI results are tuned such that the near-optimum performance is achieved while still maintaining a conserved energy (constant Hamiltonian) approaching machine precision. The initial conditions used for Low-Earth Orbit (LEO) and Medium-Earth Orbit (MEO) are given in Tables 3.1 and 3.2; both orbits start at perigee in this case.

Table 3.1: LEO (e = 0.1) Trajectory Initial Conditions

Position (km)	[2,865.408457;	5,191.131097;	2,848.416876]
Velocity (km/s)	[-5.386247766;	-0.3867151905;	6.123151881]

Table 3.2: MEO (e = 0.3) Trajectory Initial Conditions

Position (km)	[2,865.408457;	5,191.131097;	2,848.416876]
Velocity (km/s)	[-5.855468656;	-0.4204037347;	6.656567888]

3.3.1 Comparison: MEE vs. Cartesian MCPI Orbit Propagations

LEO results for both the Classical Orbital Elements and the Modified Equinoctial Elements are verified by comparing with the integration of Cartesian coordinates using MCPI, as well as spot checked with Gauss Jackson (8th Order). Figures 3.2, 3.3, and 3.4 show that the solution obtained (and then converted to Cartesian coordinates) is the same to machine precision as the solution obtained using Cartesian coordinates to directly integrate the ECI solution. Figure 3.2 compares the zonal $J2-J6$ COE solution with the ECI solution, 3.3 compares the $J2-J6$ MEE solution with the ECI solution, and 3.4 compares the spherical harmonic MEE solution with the ECI solution. In addition, the following metric is used to compute the error based on both the position and velocity and is found to be machine precision for all these cases:

$$\epsilon = \frac{|\mathbf{r}_{MEE} - \mathbf{r}_{Cartesian}|}{|\mathbf{r}_{Cartesian}|} + \frac{|\dot{\mathbf{r}}_{MEE} - \dot{\mathbf{r}}_{Cartesian}|}{|\dot{\mathbf{r}}_{Cartesian}|} \tag{3.27}$$

Integration of Cartesian coordinates using MCPI has been extensively compared and validated against several currently existing methods, including Gauss Jackson (8th), RK1210, RK78, and RK45 [16, 18, 37]. For the comparision with Cartesian

coordinates, position and velocity are integrated using a spherical harmonic gravity model with no a priori knowledge of the orbit. Next, the orbital elements are integrated (also with no a priori knowledge of the orbit), and then the solutions are converted to position and velocity for this comparison. Integration of the Cartesian coordinates requires a different number of sample points per orbit (i.e., for $J2 - J6$, $N = 100$) than the Orbital Elements cases (i.e., for $J2 - J6$ gravity terms, $N = 130$ for Classical and $N = 65$ for Equinoctial), so the results are interpolated for this analysis. A smaller number of sample points is needed for convergence using the Orbital Elements cases because the MCPI MEE differential equations are heuristically less nonlinear due to their regularized behavior and are well-suited to these variables. These variables vary very slowly with time and, whenever required, the MEEs can be transformed to the Cartesian coordinates. Once the MCPI coefficients have been computed, the state may be found at any point on the trajectory by using the Chebyshev Polynomials as the basis functions.

The major advantage of using either set of orbital elements is that the solution is convergent for a large number of orbits. For the LEO orbit using the first 6 zonal harmonic $J2 - J6$ perturbations only, the MCPI solutions computed using Gauss' equations for the Classical Orbital Elements converges precisely for 12 orbits, as can be seen in Figure 3.9; the Hamiltonian is conserved until the 13th orbit. The more regular set of Modified Equinoctial Orbital elements, remarkably, converges for over 50 orbits for $J2 - J6$ before the Hamiltonian check starts to fail, as can be seen in Figure 3.10. For many applications, only these zonal accelerations are needed to give the desired accuracy; for higher-fidelity applications, a full gravity model can be utilized.

An example of the osculating MEEs for a LEO orbit (e = 0.1) is shown in Figure 3.5. These osculating elements are converted to the COEs and plotted in Figure 3.6.

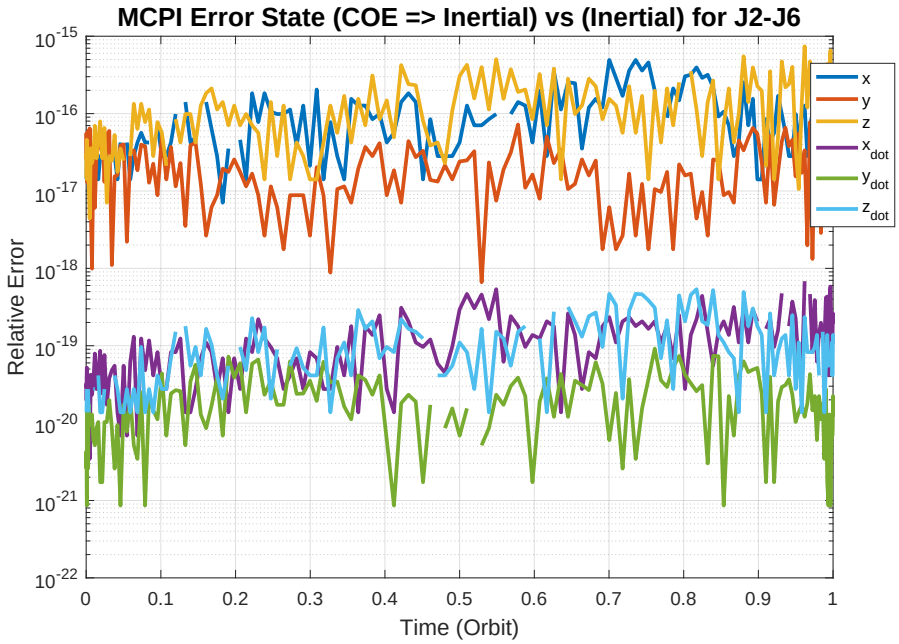


Figure 3.2: Verification of Classical Orbital Elements Solution vs. Cartesian for One Orbit Using Zonal Harmonics Gravity [48]

Figures 3.7 and 3.8 show the same orbit solution as given in Figures 3.5 and 3.6, but in Cartesian coordinates for comparison. By observation, the MEE variation as a function of time is an order of magnitude smaller than that of the Cartesian case (variations are in the 3^{rd} significant figure for the MEEs and in the 1^{st} significant figure for Cartesian). This quality gives a relatively smaller magnitude for the time derivatives of the MEEs as computed in Gauss' Variational Equations, compared with Cartesian, and allows for convergent MCPI solutions. For this reason, the MEEs are an attractive set for the MCPI propagation method. Note that the true longitude L in Figure 3.5 and true anomaly f in Figure 3.6 have been modulated to

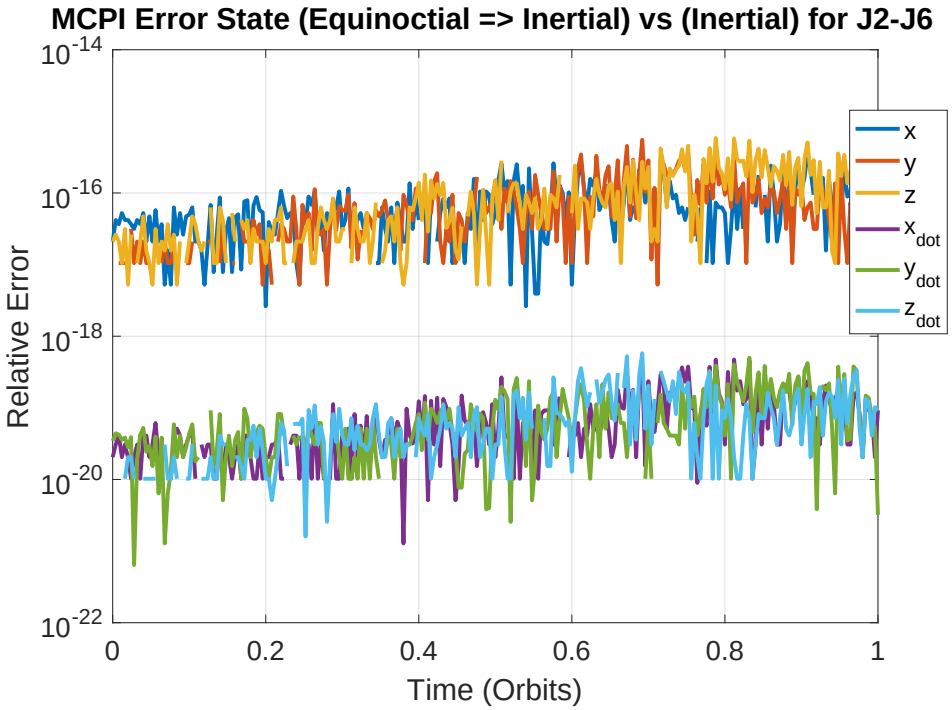


Figure 3.3: Verification of Modified Equinoctial Orbital Elements Solution vs. Cartesian for One Orbit Using Zonal Harmonics Gravity [48]

lie in the 0 to 2π range; otherwise, their plots would be nearly linear and continuous.

Since the MEEs are the author's set of choice, a spherical harmonic gravity model is included in the simulation results to provide a more precise solution. The Hamiltonian is conserved for 17 orbits using LEO initial conditions, as is seen in Fig. (3.11). This number is larger than the maximum number of orbits (up to three) possible with Cartesian coordinates using a single segment to propagate position and velocity. The solution is verified against Gauss Jackson (8th) since the energy check over a large number of orbits may not reveal an error in the direction of the velocity. Figures (3.12) - (3.14) show the maximum number of orbits for which MCPI will converge, as

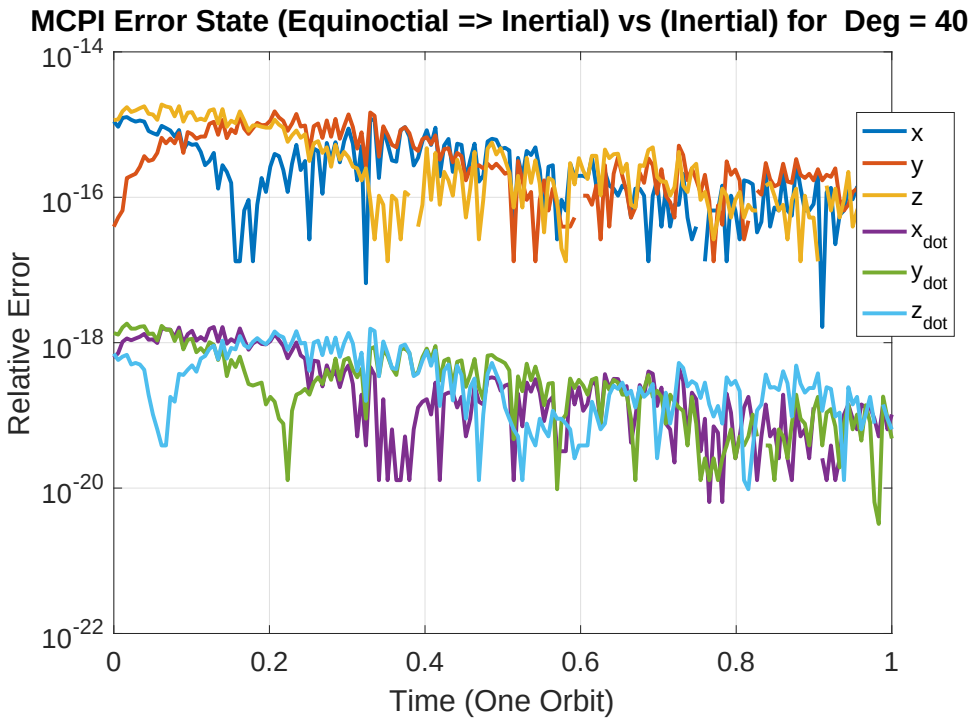


Figure 3.4: Verification of Modified Equinoctial Orbital Elements Solution vs. Cartesian for One Orbit Using Spherical Harmonic Gravity Degree and Order 40 [48]

a function of degree and order gravity, as well as the number of MCPI iterations and number of cosine nodes (sample points) per orbit for both LEO and MEO cases. The present method increases the domain of convergence by > 5.5x compared with the MCPI Cartesian solution. These results have been hand-tuned to provide the best solution (i.e., satisfies Hamiltonian conservation) with the fewest number of nodes and largest tolerance possible. However, optimizing the tuning process may provide better results [35, 36].

Note in Figure 3.15 that MCPI using the MEE coordinates requires about $\frac{1}{3}$ the number of Picard iterations compared to MCPI using Cartesian coordinates. Notice

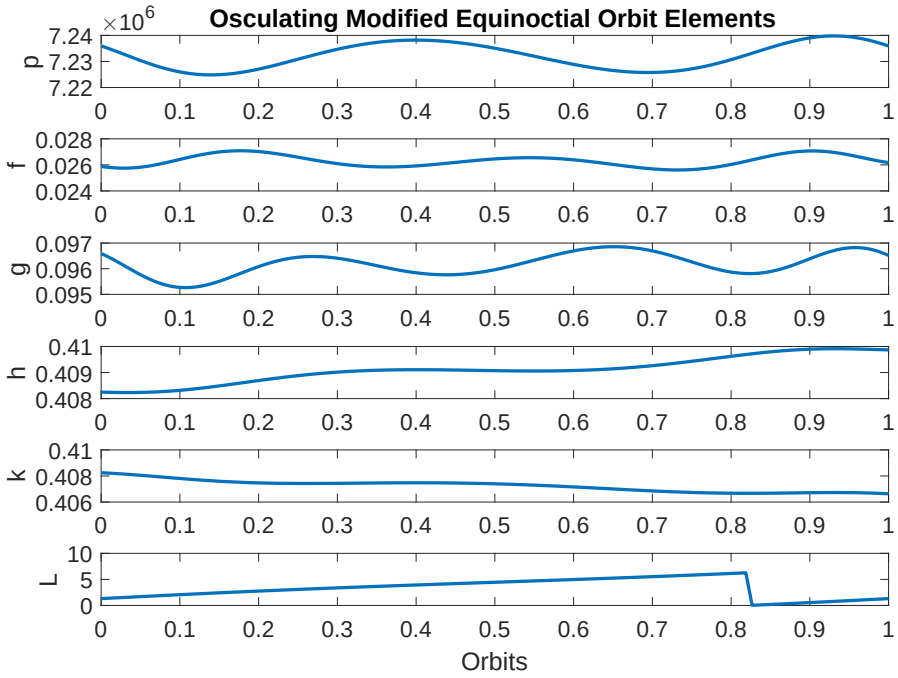


Figure 3.5: Osculating Modified Equinoctial Orbital Elements Degree and Order 40, LEO (e = 0.1)

that this advantage remains constant with increasing force model fidelity. Figure 3.16 shows that there is little advantage until the gravity degree is about 35. Note that high precision gravity models for LEO are about degree and order 200, so substantial speedup is anticipated. For a specified degree of gravity, a "transportation cost" of transformation from MEE coordinates to Cartesian coordinates (and back), in order to compute gravity partially degrades the advantages anticipated otherwise. Figure 3.17 shows that MEE coordinates require fewer gravity calls than Cartesian coordinates, and the advantage increases as the gravity field degree increases.

3.3.2 Segmentation

Previous work [28, 29, 36] has shown that segmenting the trajectory increases efficiency; this method is implemented for the present work as well. Optimal seg-

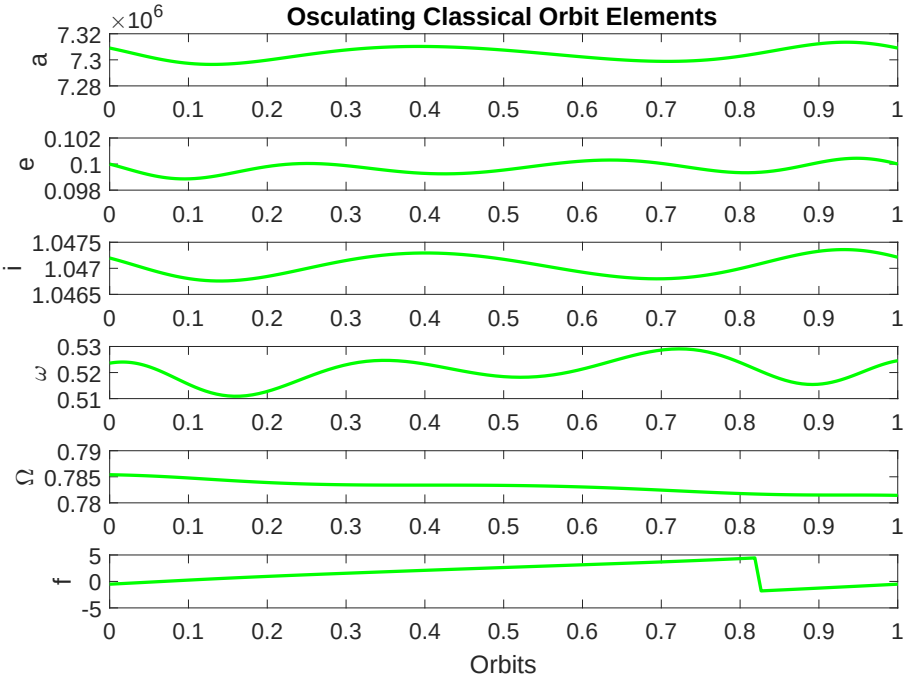


Figure 3.6: Osculating Modified Equinoctial Orbital Elements, Converted to Classical Orbital Elements, Degree and Order 40, LEO (e = 0.1)

mentation for Cartesian coordinates utilizes a fraction of an orbit (typically 1/3 or 1/5 of an orbit per segment). However, since the orbital elements solution converges over a larger number of orbits, a larger segment is used. For this analysis, one orbit per segment is used; in this manner, the final state of the previous segment is used as the initial conditions for the next segment. Analogously to Cartesian integration of Earth orbits, segmenting the MEE propagation allows for increased efficiency and decreased number of nodes, even though more MCPI iterations are required. This leads to a reduction in the number of full gravity computations required; since gravity is computationally expensive, initial studies using Matlab show a decreased computation time. Figures (3.18) - (3.20) show results using a one-orbit-per-segment scheme versus using a single segment over the entire trajectory. For this study, the

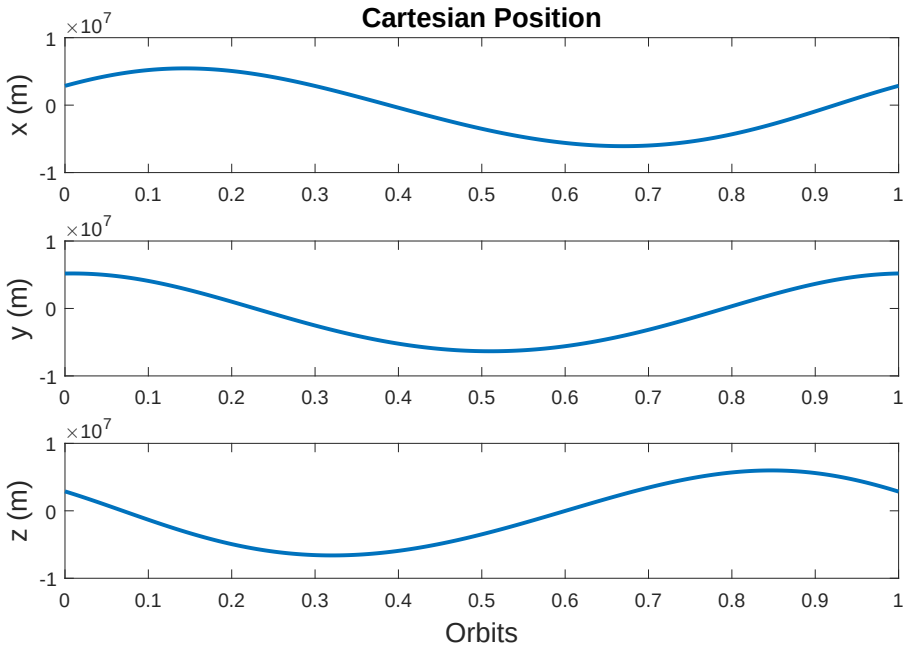


Figure 3.7: Cartesian Position Components, Degree and Order 40, LEO (e = 0.1)

maximum number of orbits as a function of degree and order gravity is used to determine the results; for instance, by looking at Figure (3.12) a 25^{th} degree and order gravity model will allow the method to converge for 25 orbits for the LEO case and for 22 orbits for the MEO case. By setting this number of orbits as the final time for each orbit respectively, the number of MCPI iterations per orbit are computed for Figure (3.13), the number of sample points per orbit for Figure (3.14), and so on.

Notice in Figure 3.18 the difference between using one segment over many orbits vs. using one segment per orbit; the number of MCPI iterations is approximately a linear function of the time span. However, the number of function evaluations per orbit increases when many orbits are covered by one segment (Figures 3.19 and 3.20). Additional studies show that using more than on orbit per segment (i.e., two orbits per segment) may be more computationally efficient than using one orbit

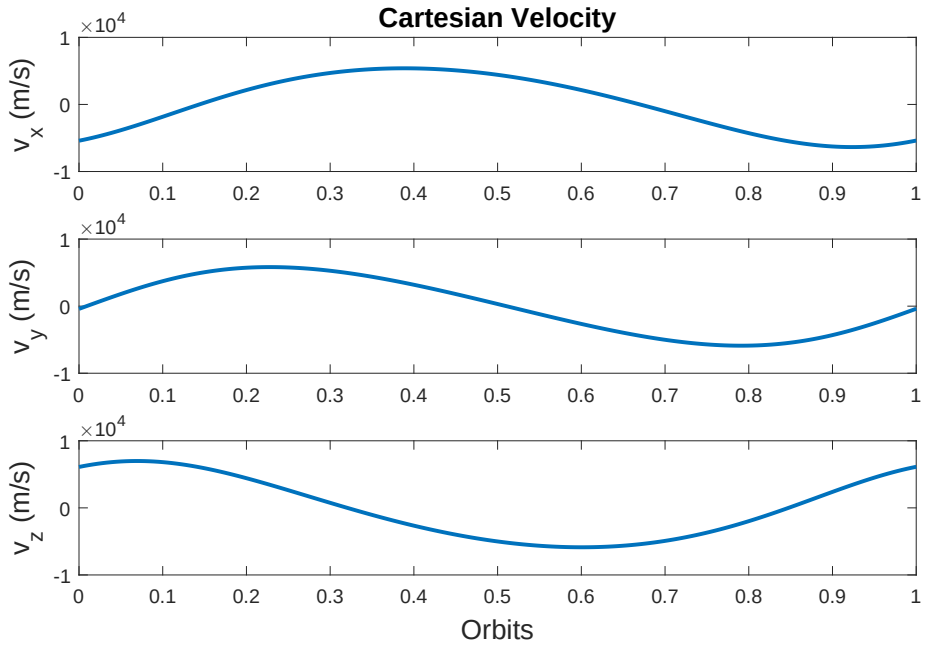


Figure 3.8: Cartesian Velocity Components, Degree and Order 40, LEO (e = 0.1)

per segment, but for simplicity one orbit per segment is chosen for the work in this dissertation. One additional advantage of using one orbit per segment for the MEE case is that the CGL nodes may be clustered always at perigee and sparse at apogee; in contrast, using 3 or 5 segments per orbit with Cartesian coordinates results in dense nodes not only at perigee also unnecessarily throughout factions of the orbit. If a non-integer number of orbits is required for the integration, most of the trajectory may be computed using the one-orbit-per-segment scheme to allow the dense cosine nodes to be clustered at perigee. If the initial time does not start at perigee, a single fraction-of-an-orbit segment may be used to reach perigee. Similarly, an additional segment may be computed from the last perigee approach to the final time. This ensures higher-fidelity gravity computations near perigee.

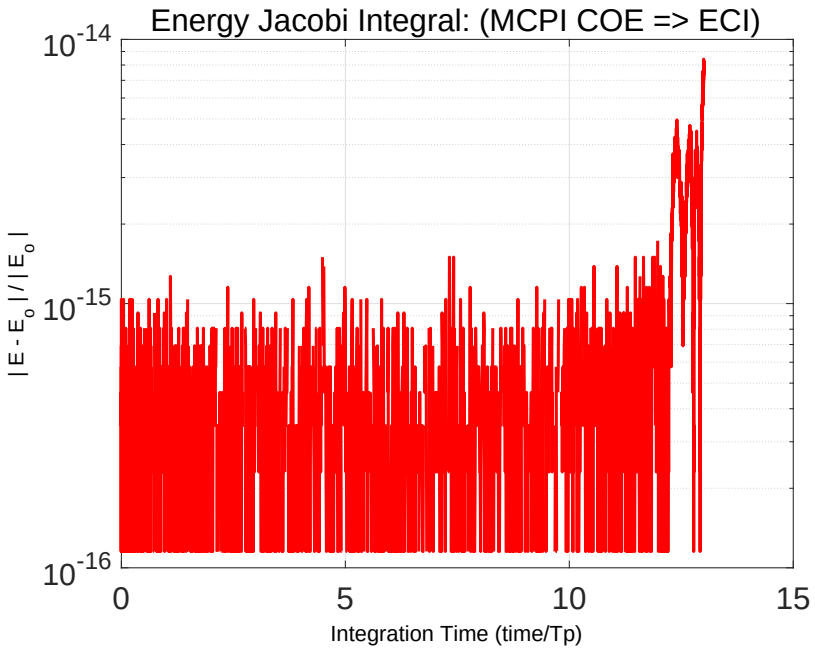


Figure 3.9: Energy Check for Classical Orbital Elements Solution for 13 Orbits (LEO) Using a Single MCPI Solution Segment [48]

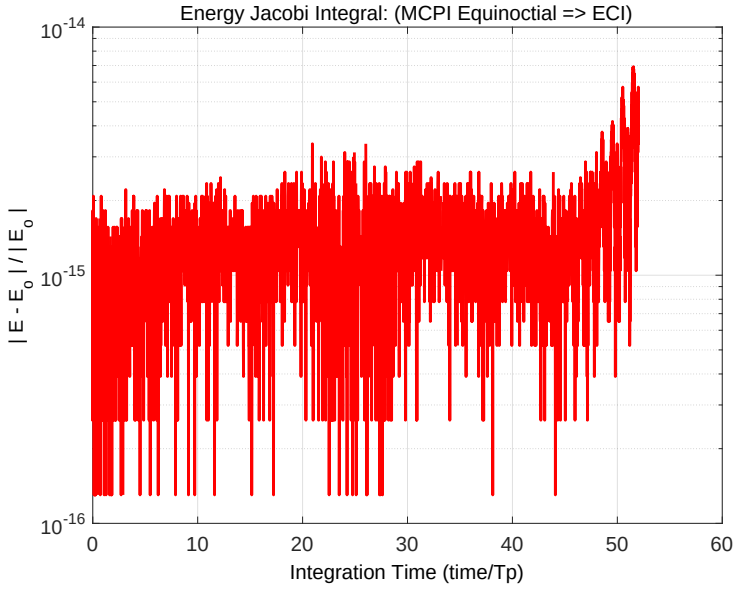


Figure 3.10: Energy Check for Modified Equinoctial Orbital Elements Solution for 53 Orbits (LEO) Using a Single MCPI Solution Segment [48, 49]

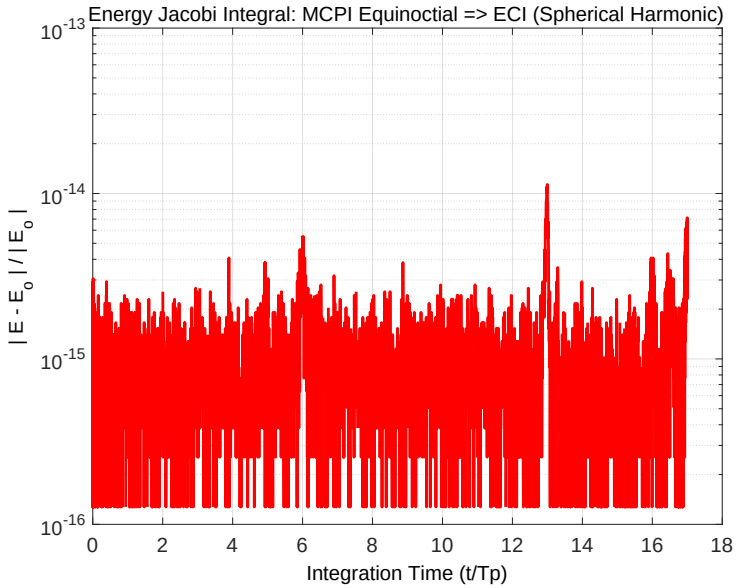


Figure 3.11: 40^{th} Degree and Order Spherical Harmonic Energy Check for Modified Equinoctial Orbital Elements Solution (LEO) for 17 Orbits Using a Single MCPI Solution Segment [48, 49]

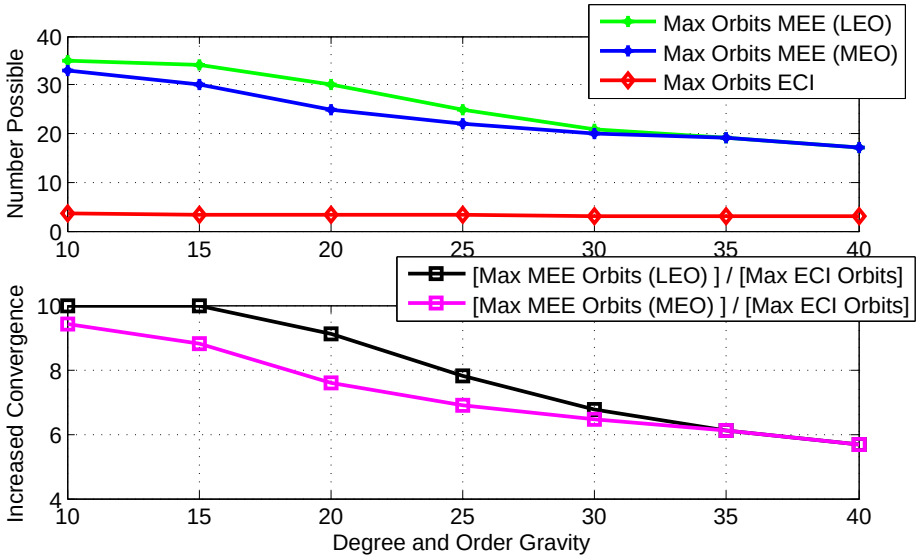


Figure 3.12: Maximum Number of Orbits Over Which MCPI Will Converge as a Function of Varying Degree and Order Spherical Harmonic Gravity [48]

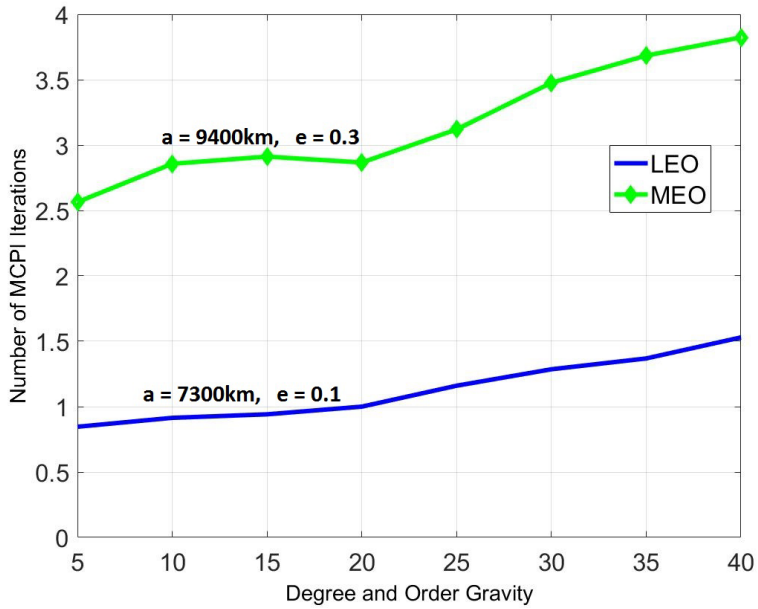


Figure 3.13: Number of MCPI Iterations Per Orbit as a Function of Varying Degree and Order Spherical Harmonic Gravity, for Moderate ($e = 0.1$) and Intermediate ($e = 0.3$) Eccentricity [48]

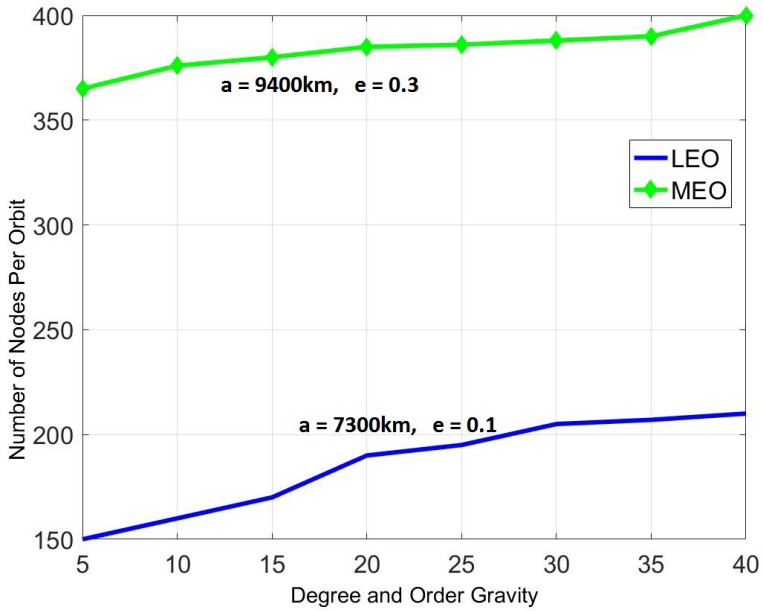


Figure 3.14: Number of Acceleration Sample Points (Nodes) Per Orbit as a Function of Varying Degree and Order Spherical Harmonic Gravity, for Moderate ($e = 0.1$) and Intermediate ($e = 0.3$) Eccentricity [48]

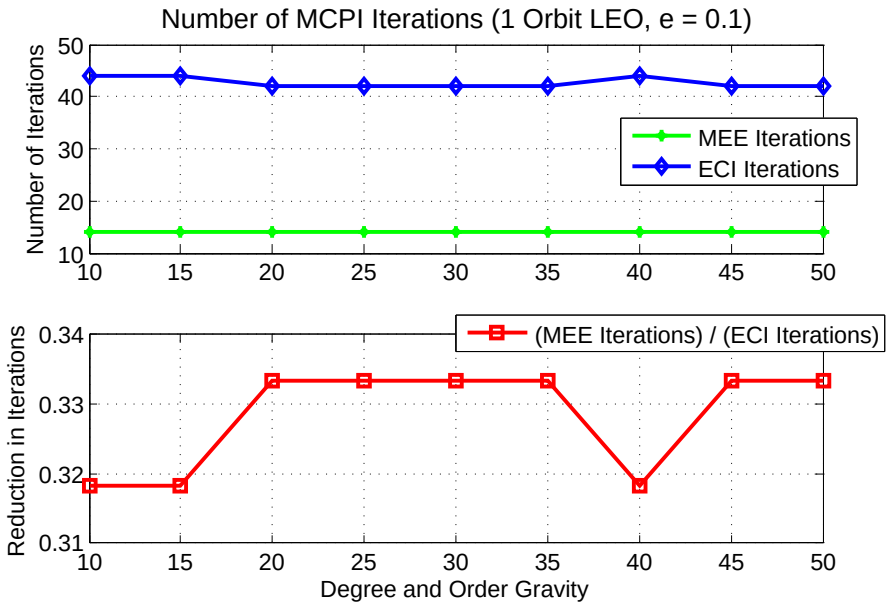


Figure 3.15: Comparison of MCPI Iterations Per Orbit for MEE versus Cartesian as a Function of Varying Degree and Order Spherical Harmonic Gravity [48]

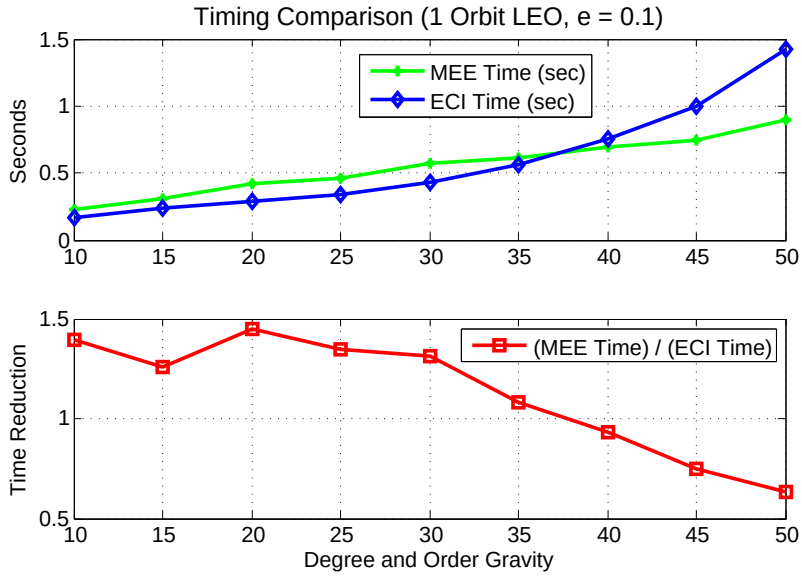


Figure 3.16: Comparison of MCPI Time Per Orbit for MEE versus Cartesian as a Function of Varying Degree and Order Spherical Harmonic Gravity [48]

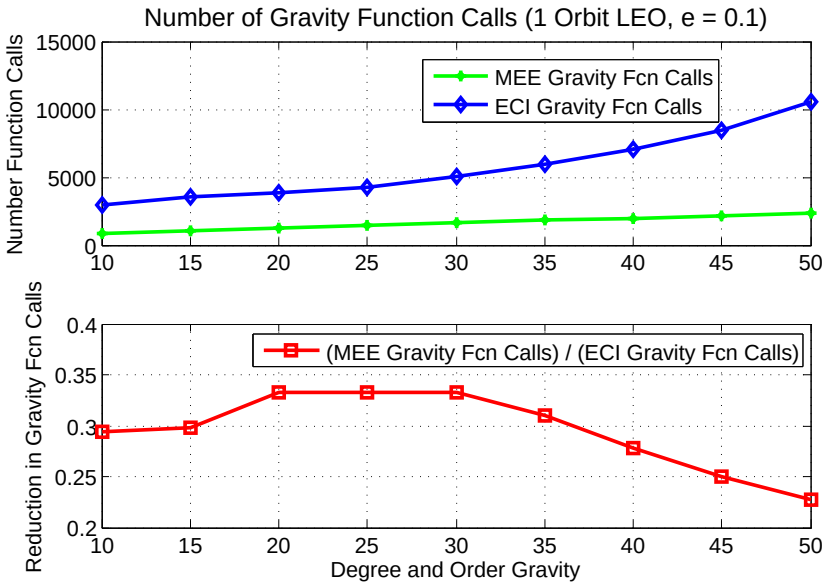


Figure 3.17: Comparison of MCPI Gravity Function Calls Per Orbit for MEE versus Cartesian as a Function of Varying Degree and Order Spherical Harmonic Gravity [48]

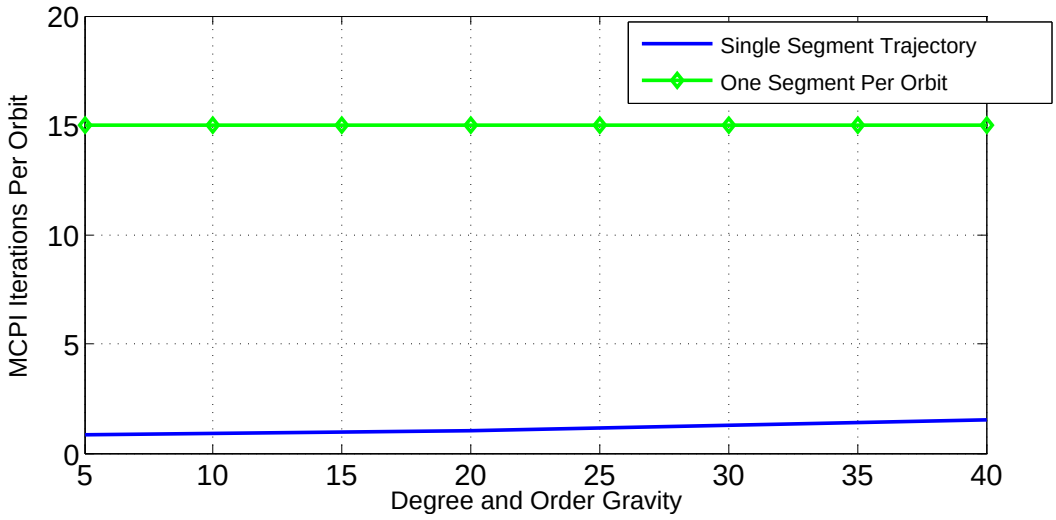


Figure 3.18: Segmented Increase in Number of MCPI Iterations Per Orbit as a Function of Varying Degree and Order Spherical Harmonic Gravity [48]

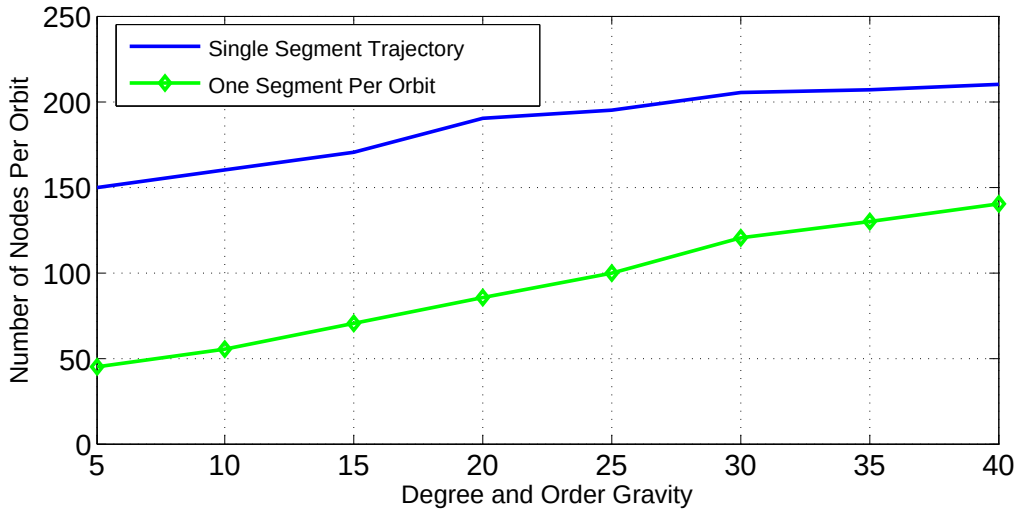


Figure 3.19: Segmented Decrease in Number of MCPI Nodes Per Orbit as a Function of Varying Degree and Order Spherical Harmonic Gravity [48]

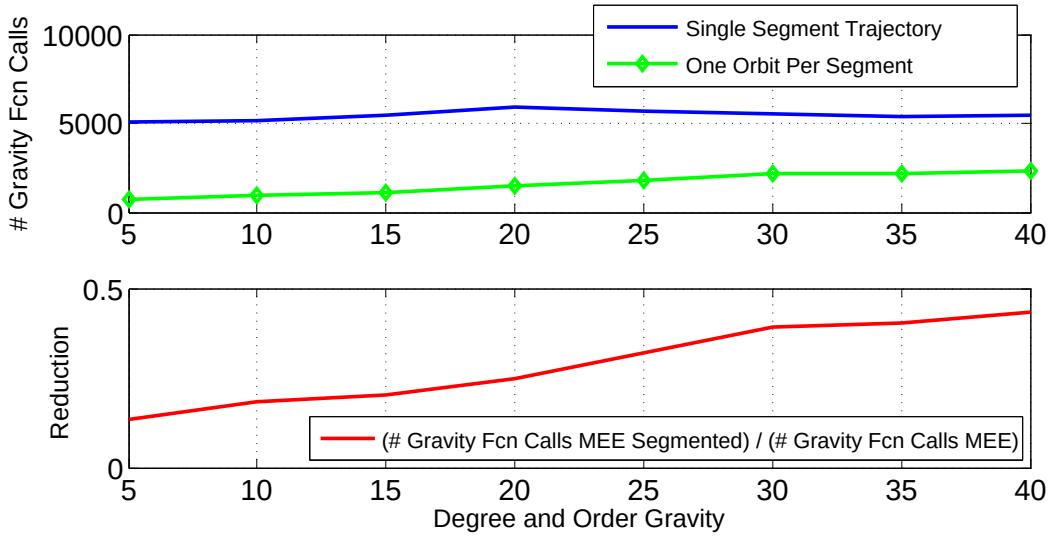


Figure 3.20: Segmented Decrease in Number of Gravity Function Calls Per Orbit as a Function of Varying Degree and Order Spherical Harmonic Gravity [48]

3.4 Chapter Summary

Propagation of either the Classical or the Modified Equinoctial Elements is an attractive method to solving the perturbed two-body problem using Modified Chebyshev Picard Iteration. Both differential equations result in MCPI convergence for a large number of orbits, while Cartesian coordinates used in conjunction with Modified Chebyshev Picard Iteration only converges for a few orbits in Cartesian coordinates, using a single segment. The Modified Equinoctial Elements avoid singularities that are problematic for the Classical Orbital Elements and give a slightly more accurate solution, so they are the preferred choice of variables. Higher order gravity models (such as the spherical harmonic gravity implemented here) lead to analogously long intervals for convergence, albeit with an increase in the number of basis functions.

The combination of the Modified Equinoctial Orbital Elements with MCPI leads to decreased number of nodes, MCPI iterations, and gravity function calls when compared with Cartesian coordinates, which is typically the standard method used in orbit propagation. Optimizing the algorithm by using a segmentation scheme decreases the number of nodes and gravity function calls, at the cost of adding a few more MCPI iterations, to reduce the overall computation time. The computational results in this chapter make a commanding case that the fusion of MCPI and MEE coordinates provide very significant computational advantages that will affect many dimensions of astrodynamics.

4. STATE TRANSITION MATRIX FOR SPHERICAL HARMONIC GRAVITY*

In this chapter the MCPI method is applied to solve the differential equations governing the State Transition Matrix (STM), for the case of perturbed motion. This perturbed STM has applications in many areas including celestial mechanics and control systems. Propagation of the STM is useful in determining the sensitivity of the IVP solution to the initial conditions; for instance, the STM predicts how deviations from the initial conditions will cause the trajectory of a spacecraft to deviate from a nominal path. The STM may be used to approximate the time evolution of variations off nominal for the state vector, even for highly nonlinear systems, such as the two-body problem perturbed by an arbitrary degree spherical harmonic gravity. In cases where the initial deviation is small at time t_0, a linear approximation may be used to determine the locally linear state deviation at time t. This linear approximation is generated using the STM, a matrix of partial derivatives for the instantaneous position and velocity with respect to the initial position and velocity; this means that the STM is initially equal to the identity matrix [7, 54]. These deviations can be approximated, to a problem-dependent accuracy, by using the unperturbed analytical solution for the Keplerian STM; however, it is frequently desirable to be able to include a prescribed number the spherical harmonic perturbations, and in general,

*Part of the data reported in this chapter is reprinted with permission of Springer from "State Transition Matrix Propagation for Perturbed Orbital Motion Using Modified Chebyshev Picard Iteration", The Journal of the Astronautical Sciences, June 2015, Volume 62, Issue 2, pp. 148-167. In addition, part of the data reported in this chapter is reprinted with permission of AAS from "State Transition Matrix Propagation for Perturbed Orbital Motion Using Modified Chebyshev Picard Iteration"; this paper was originally presented at the 38th Annual AAS Rocky Mountain Section Guidance and Control Conference held January 30 February 4, 2015, Breckenridge, U.S.A., and was originally published in the American Astronautical Society (AAS) publication Guidance, Navigation and Control 2015, edited by Ian J. Gravseth, American Astronautical Society (AAS) Advances in the Astronautical Sciences, Volume 154, 2016, pp. 1015-1026 (Copyright 2015 by American Astronautical Society Publications Office, P.O. Box 28130, San Diego, CA 92198, U.S.A.; Web Site: http://www.univelt.com) [50, 52]

other perturbations.

Qualitatively, since the STM is associated with (approximate) perturbations from an underlying nonlinear motion, it is evident that some level of approximation is admissible when computing the STM. If 10 digits accuracy are desired for the high fidelity state and force models, it is unlikely that more than 5 digits would ever be required for the STM. However, the present STM algorithms account for the high fidelity perturbations and adjusting the force model fidelity is accounted for in the STM. The present chapter provides an efficient algorithm to include these higher order gravitational perturbations, and allows the accuracy to be adjusted as desired/required in the STM.

The computation of the STM for the spherical harmonic gravity model requires second partials of the gravity potential with respect to spherical geocentric coordinates: radius, latitude, and longitude. The first partials of the potential, of course, give the three gravitational acceleration components, and the second partials give the symmetric nine element tensors known as the gravity gradient. These required partial derivatives include second partials of the normalized, Associated Legendre Functions (ALFs), and these functions (used often in a wide variety of science and engineering applications) are related to the Legendre polynomials as will be discussed in this chapter.

The normalized version of the ALFs is preferred for the associated recursion used to stably compute these functions. Without the normalization, the ALFs are known to tend toward weak numerical instability as a higher degree and order gravity model is used. Once the normalized ALFs are computed by stable recursions, the gravity and associated derivatives are computed in part by introducing the appropriate scale factor to generate the un-normalized version of the ALFs. Since a spherical harmonic model is used and this gravity potential is a rigorous solution of the Laplace equation,

the ALFs' second partial expression for an arbitrary order spherical harmonic series may be verified by checking the accuracy to which the series satisfies the Laplace equation.

Both the trajectory and the STM are computed in a rotating, Earth-centered, Earth-fixed (ECEF) frame, which is transformed into an Earth-centered inertial frame (ECI) for subsequent integration of the differential equations at each Picard iteration. For the case of MCPI, this is shown to be an efficient method of integration. The rotating ECEF frame is used to avoid the explicit time dependence of the potential and also to derive the associated Jacobi integral (which is a constant Hamiltonian for the perturbed motion).

While very high precision of the STM is seldom required in practice, high fidelity validation is performed for the STM differential equations and the MCPI solution of these equations. Though enforcement of the governing differential equation is inherent in the present study, the STM is confirmed to accurately satisfy (with maximum relative errors of $< 10^{-14}$ in a Matlab implementation and up to one orbit, for the most precise tuning) the theoretical STM group properties as well as the STM symplectic property (discussed in Section 4.6: MCPI STM Results) associated with natural conservative dynamical systems [7, 54]. However, in many practical applications, the STM precision can frequently be relaxed (to, say four or five digit precision) even when double precision is required for propagating the orbit. This means that the spherical harmonic expansion order for the STM solution can frequently be truncated to, say, degree and order gravity of five even when a much higher degree gravity model is required to compute the orbit. The results for this algorithm are given in two publications [50, 52].

4.1 Dynamic Model

The gravity-perturbed acceleration $\ddot{\mathbf{r}} = \mathbf{g}(t, \mathbf{r})$ may be represented in state space notation for the MCPI algorithm as

$$\mathbf{x} \equiv \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \equiv \begin{bmatrix} \mathbf{r} \\ \dot{\mathbf{r}} \end{bmatrix}, \quad \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{bmatrix} = \mathbf{f}(t, \mathbf{x}) = \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{g}(t, \mathbf{x}_1) \end{bmatrix} \tag{4.1}$$

The differential equation used to integrate the STM is [7]

$$\dot{\Phi}(t, t_0) = \mathbf{A}\Phi(t, t_0) \tag{4.2}$$

where

$$\mathbf{A}(t) = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ G_{n\times n} & 0_{n\times n} \end{bmatrix} \triangleq \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right] = \begin{bmatrix} \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_2} \\ \frac{\partial \mathbf{g}}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{g}}{\partial \mathbf{x}_2} \end{bmatrix} = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ \frac{\partial \mathbf{g}}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{g}}{\partial \mathbf{x}_2} \end{bmatrix} \tag{4.3}$$

and

$$G_{n\times n} = \left[\frac{\partial \mathbf{g}(t, \mathbf{r})}{\partial \mathbf{r}}\right] \tag{4.4}$$

Obviously, when drag is included in the force model, $\frac{\partial \mathbf{g}}{\partial \mathbf{x}_2} \neq 0$. For the current developments, only the spherical harmonic gravity potential is included in the force model. The STM is computed as partials of the linear approximation of the instantaneous departure $\mathbf{X}(t)$ from the state vector $\mathbf{x}(t)$ to the neighboring state $\tilde{\mathbf{x}}(t)$:

$$\Phi(t, t_0) = \frac{\partial \mathbf{X}(t)}{\partial \mathbf{X}(t_0)} \tag{4.5}$$

where $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) + \mathbf{X}(t)$. Then the G matrix can be written in terms of the partials

of the generally perturbed gravitational acceleration case; for example, the partial of the first component of acceleration with respect to the first component in a body-fixed frame is written with respect to geocentric radius, latitude, and longitude as

$$\frac{\partial \mathbf{a}_x}{\partial x} = \frac{\partial \mathbf{a}_x}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial \mathbf{a}_x}{\partial \phi}\frac{\partial \phi}{\partial x} + \frac{\partial \mathbf{a}_x}{\partial \lambda}\frac{\partial \lambda}{\partial x} \tag{4.6}$$

4.2 State Transition Matrix Using Spherical Harmonic Gravity

A spherical harmonic gravity model is considered in this section. The Jacobian of the acceleration, needed for Eq. (4.3), is most efficiently computed in the Earth-fixed body frame but transformed into the inertial frame prior to each integration step; this method is presented in the following section. The full gravitational potential function is defined as [24]

$$U(r,\phi,\lambda) = \frac{\mu}{r}\left[1 + \sum_{n=2}^{\infty}\sum_{m=0}^{n}\left(\frac{R_e}{r}\right)^n P_{nm}(\sin\phi)\left[C_{nm}\cos(m\lambda) + S_{nm}\sin(m\lambda)\right]\right] \tag{4.7}$$

where r is the radial distance to the object, ϕ is the geocentric latitude of the object, λ is the longitude of the object, R_e is the Earth's equatorial radius, n is the degree of the series, m is the order of the series, and P_{nm} are the Associated Legendre Functions. Defining values for n and m gives the degree and order gravity specified by the user once the corresponding gravity acceleration terms are computed. The Cartesian components of the Earth-fixed spherical harmonic series representation of gravitational acceleration may be represented as the gradient of Eq. (4.7).

$$\mathbf{a}_g = \frac{\partial U}{\partial r}\left(\frac{\partial r}{\partial \mathbf{r}}\right)^T + \frac{\partial U}{\partial \phi}\left(\frac{\partial \phi}{\partial \mathbf{r}}\right)^T + \frac{\partial U}{\partial \lambda}\left(\frac{\partial \lambda}{\partial \mathbf{r}}\right)^T \tag{4.8}$$

Expressions for the partials of the gravity potential in Equation (4.7) with respect

to spherical coordinates (r, ϕ, λ) are written explicitly as

$$\frac{\partial U}{\partial r} = U_r = -\frac{\mu}{r^2}\left[1+\sum_{n=2}^{\infty}\sum_{m=0}^{n}\left(n+1\right)\left(\frac{R_e}{r}\right)^n P_{nm}(\sin\phi)\Big[C_{nm}\cos(m\lambda)+S_{nm}\sin(m\lambda)\Big]\right] \tag{4.9}$$

$$\frac{\partial U}{\partial \phi} = U_\phi = \frac{\mu}{r}\sum_{n=2}^{\infty}\sum_{m=0}^{n}\left(\frac{R_e}{r}\right)^n \frac{\partial P_{nm}(\sin\phi)}{\partial\phi}\Big[C_{nm}\cos(m\lambda)+S_{nm}\sin(m\lambda)\Big]\Bigg] \tag{4.10}$$

$$\frac{\partial U}{\partial \lambda} = U_\lambda = \frac{\mu}{r}\sum_{n=2}^{\infty}\sum_{m=0}^{n}\left(\frac{R_e}{r}\right)^n P_{nm}(\sin\phi)m\Big[S_{nm}\cos(m\lambda)-C_{nm}\sin(m\lambda)\Big]\Bigg] \tag{4.11}$$

Next, expressions are found to compute the Jacobian of the potential in Eq. (4.4), (4.8). The Cartesian components of the gravity perturbed acceleration $\mathbf{a}_g$, which are components of Eq. (4.5), are

$$\mathbf{a}_X = \frac{\partial U}{\partial r}\left(\frac{\partial r}{\partial x}\right) + \frac{\partial U}{\partial \phi}\left(\frac{\partial \phi}{\partial x}\right) + \frac{\partial U}{\partial \lambda}\left(\frac{\partial \lambda}{\partial x}\right) \tag{4.12}$$

$$\mathbf{a}_Y = \frac{\partial U}{\partial r}\left(\frac{\partial r}{\partial y}\right) + \frac{\partial U}{\partial \phi}\left(\frac{\partial \phi}{\partial y}\right) + \frac{\partial U}{\partial \lambda}\left(\frac{\partial \lambda}{\partial y}\right) \tag{4.13}$$

$$\mathbf{a}_Z = \frac{\partial U}{\partial r}\left(\frac{\partial r}{\partial z}\right) + \frac{\partial U}{\partial \phi}\left(\frac{\partial \phi}{\partial z}\right) + \frac{\partial U}{\partial \lambda}\left(\frac{\partial \lambda}{\partial z}\right) \tag{4.14}$$

These acceleration components are taken along Earth-fixed axes; these can be projected into some arbitrary inertial frame by multiplying by an appropriate direction cosine matrix (this process will be described in more detail later in this chapter).

For the derivatives with respect to β which represents any one of the Cartesian coordinates (x, y, z), the general chain rule is used:

$$\frac{\partial}{\partial\beta}(\cdot) = \frac{\partial}{\partial r}(\cdot)\left(\frac{\partial r}{\partial\beta}\right) + \frac{\partial}{\partial\phi}(\cdot)\left(\frac{\partial\phi}{\partial\beta}\right) + \frac{\partial}{\partial\lambda}(\cdot)\left(\frac{\partial\lambda}{\partial\beta}\right); \quad \beta \to x, y, z \tag{4.15}$$

The individual components of the Jacobian G from Eq. (4.4) then follow a pattern (see Appendix B.1 for all G component expressions). The spherical coordinate partials with respect to Cartesian coordinates are given by

$$\frac{\partial r}{\partial\alpha} = \frac{\alpha}{r}; \quad \alpha \to x, y, z \tag{4.16}$$

and

$$\frac{\partial\phi}{\partial x} = \frac{-xz}{r^2\sqrt{x^2+y^2}}; \quad \frac{\partial\phi}{\partial y} = \frac{-yz}{r^2\sqrt{x^2+y^2}}; \quad \frac{\partial\phi}{\partial z} = \frac{\left(1-\frac{z^2}{r^2}\right)}{\sqrt{x^2+y^2}} \tag{4.17}$$

$$\frac{\partial\lambda}{\partial x} = \frac{-y}{x^2+y^2}; \quad \frac{\partial\lambda}{\partial y} = \frac{x}{x^2+y^2}; \quad \frac{\partial\lambda}{\partial z} = 0 \tag{4.18}$$

Then, the second partials of the gravity potential are found by direct partial differentiation, where the partials of the ALFs are described in the following section. The matrix of second partials is symmetric, and the six expressions for the distinct elements are as follows:

$$\frac{\partial^2 U}{\partial r^2} = \frac{\mu}{r^3}\left[2+\sum_{n=2}^{\infty}\sum_{m=0}^{n}\big(n+1\big)\big(n+2\big)\left(\frac{R_e}{r}\right)^n P_{nm}(\sin\phi)\Big[C_{nm}\cos(m\lambda)+S_{nm}\sin(m\lambda)\Big]\right] \tag{4.19}$$

$$\frac{\partial^2 U}{\partial r \partial \phi} = -\frac{\mu}{r^2} \sum_{n=2}^{\infty} \sum_{m=0}^{n} (n+1) \left(\frac{R_e}{r} \right)^n \frac{\partial P_{nm}(\sin\phi)}{\partial \phi} \Big[C_{nm} \cos(m\lambda) + S_{nm} \sin(m\lambda) \Big] \Bigg] \tag{4.20}$$

$$\frac{\partial^2 U}{\partial r \partial \lambda} = -\frac{\mu}{r^2} \sum_{n=2}^{\infty} \sum_{m=0}^{n} (n+1) \left(\frac{R_e}{r} \right)^n P_{nm}(\sin\phi) \Big[S_{nm} \cos(m\lambda) - C_{nm} \sin(m\lambda) \Big] \Bigg] \tag{4.21}$$

$$\frac{\partial^2 U}{\partial \phi^2} = \frac{\mu}{r} \sum_{n=2}^{\infty} \sum_{m=0}^{n} \left(\frac{R_e}{r} \right)^n \frac{\partial^2 P_{nm}(\sin\phi)}{\partial \phi^2} \Big[C_{nm} \cos(m\lambda) + S_{nm} \sin(m\lambda) \Big] \Bigg] \tag{4.22}$$

$$\frac{\partial^2 U}{\partial \phi \partial \lambda} = \frac{\mu}{r} \sum_{n=2}^{\infty} \sum_{m=0}^{n} \left(\frac{R_e}{r} \right)^n \frac{\partial P_{nm}(\sin\phi)}{\partial \phi} m \Big[S_{nm} \cos(m\lambda) - C_{nm} \sin(m\lambda) \Big] \Bigg] \tag{4.23}$$

$$\frac{\partial^2 U}{\partial \lambda^2} = -\frac{\mu}{r} \sum_{n=2}^{\infty} \sum_{m=0}^{n} \left(\frac{R_e}{r} \right)^n P_{nm}(\sin\phi) m^2 \Big[S_{nm} \cos(m\lambda) + C_{nm} \sin(m\lambda) \Big] \Bigg] \tag{4.24}$$

4.3 Computation of Associated Legendre Functions

A key component of the spherical harmonic gravity calculations is the set of Associated Legendre Functions, P_{nm}. These functions are related to the Lengendre polynomials, $P_n(u)$ [34] through

$$P_{n0}(u) = P_n(u) = \frac{1}{2^n n!} \frac{d^n}{du^n} (u^2 - 1)^n \tag{4.25}$$

$$P_{nm}(u) = (1-u^2)^{\frac{m}{2}} \frac{d^m}{du^m} P_n(u) \tag{4.26}$$

After substituting the expression for $P_n(u)$ in Eq. (4.25) into Eq. (4.26) results in

$$P_{nm}(u) = (1-u^2)^{\frac{m}{2}} \frac{1}{2^n n!} \frac{d^{m+n}}{du^{m+n}} (u^2-1)^n \tag{4.27}$$

This expression is commonly seen in the literature to represent the ALFs. In many practical applications that require numerical stability for higher degree and order gravity, however, the potential function is represented using the so-called derived Associated Legendre Functions. The derived ALFs are defined as follows:

$$A_{nm}(u) = \frac{1}{2^n n!} \frac{d^{m+n}}{du^{m+n}} (u^2-1)^n = \frac{d^m}{du^m} P_n(u) \tag{4.28}$$

The derived ALFs and spherical harmonic coefficients are normalized to improve accuracy for high degree and order gravity models [24, 50, 52]. The normalized and un-normalized, derived ALFs are related by a normalization scaling factor N_{nm}, written generally for $n \neq m \neq 0$ as [34]

$$\bar{A}_{nm} = \left[\frac{(n-m)!(2n+1)(2-\delta_{0m})}{(n+m)!} \right]^{\frac{1}{2}} A_{nm} = N_{nm} A_{nm} \tag{4.29}$$

Here, the Kronecker delta function is defined to be

$$\delta_{0m} = \begin{cases} 1 & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases} \tag{4.30}$$

The recursion formula chosen for this work is the normalized, derived ALFs from Table 2, option I of [34] for $u = \sin\phi$:

$$\bar{A}_{nm} = u\left[\frac{(2n+1)(2n-1)}{(n-m)(n+m)}\right]^{\frac{1}{2}}\bar{A}_{n-1,m} - \left[\frac{(2n+1)(n-m-1)(n+m-1)}{(2n-3)(n+m)(n-m)}\right]^{\frac{1}{2}}\bar{A}_{n-2,m} \tag{4.31}$$

Since a normalized version of the ALFs is used, the final un-normalized result may be obtained by applying the appropriate scale factor:

$$m = 0: \quad S_f = \sqrt{\frac{n(n+1)}{2}} \tag{4.32}$$

$$m = n: \quad S_f = 0 \tag{4.33}$$

$$m \neq n \neq 0: \quad S_f = \sqrt{(n-m)(m+n+1)} \tag{4.34}$$

Computation of the STM for spherical harmonic gravity requires the partial derivative of P_{nm} with respect to ϕ, where C_{nm} and S_{nm} are the normalized Stokes coefficients determined from satellite motion observations, and N_{nm} is the scale factor given in Eq. (4.29). The derivations for the first and second partial derivatives of P_{nm} with respect to ϕ is given fully in Appendix B and has been published [52]. These partials of the ALFs are incorporated in the calculations for the partials of the gravity potential, U, through the computation of the corresponding ALFs and the appropriate scale factors.

4.4 Earth-Centered-Interial Jacobian

Numerical integration performed in the inertial frame requires the Jacobian also to be computed in the Earth-Centered Inertial (ECI) frame using coordinates X, Y, Z. These coordinates are related to the Earth-Centered-Earth-Fixed (ECEF) frame co-

ordinates x, y, z through a rotation matrix C with the following forward and reverse transformations:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [C(t)] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.35}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \left[C^T(t)\right] \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.36}$$

Also,

$$\begin{bmatrix} a_X \\ a_Y \\ a_Z \end{bmatrix} = \left[C^T(t)\right] \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{4.37}$$

where

$$[C(t)] = \begin{bmatrix} \cos\theta(t) & \sin\theta(t) & 0 \\ -\sin\theta(t) & \cos\theta(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.38}$$

and

$$\theta(t) = \theta_0 + \omega_e(t - t_0) \tag{4.39}$$

where this last expression is a function of the Earth's rotation and time. Each component of C may be written as

$$C_{11} = \frac{\partial x}{\partial X} \quad , \quad C_{12} = \frac{\partial x}{\partial Y} \quad , \quad C_{13} = \frac{\partial x}{\partial Z} \quad , \quad etc. \tag{4.40}$$

which means that

$$\big[C(t)\big] = \left[\frac{\partial(x,y,z)}{\partial(X,Y,Z)}\right] \quad \Longleftrightarrow \quad \Big[C^T(t)\Big] = \left[\frac{\partial(X,Y,Z)}{\partial(x,y,z)}\right] \tag{4.41}$$

For a general function, F, a chain rule expansion may be used to find partials with respect to body frame coordinates r, ϕ, λ, i.e.,

$$\frac{\partial F}{\partial x} = \frac{\partial F}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial F}{\partial \phi}\frac{\partial \phi}{\partial x} + \frac{\partial F}{\partial \lambda}\frac{\partial \lambda}{\partial x} \tag{4.42}$$

$$\frac{\partial F}{\partial y} = \frac{\partial F}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial F}{\partial \phi}\frac{\partial \phi}{\partial y} + \frac{\partial F}{\partial \lambda}\frac{\partial \lambda}{\partial y} \tag{4.43}$$

$$\frac{\partial F}{\partial z} = \frac{\partial F}{\partial r}\frac{\partial r}{\partial z} + \frac{\partial F}{\partial \phi}\frac{\partial \phi}{\partial z} + \frac{\partial F}{\partial \lambda}\frac{\partial \lambda}{\partial z} \tag{4.44}$$

Written in matrix form, this gives

$$\begin{bmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial \phi}{\partial x} & \frac{\partial \lambda}{\partial x} \\ \frac{\partial r}{\partial y} & \frac{\partial \phi}{\partial y} & \frac{\partial \lambda}{\partial y} \\ \frac{\partial r}{\partial z} & \frac{\partial \phi}{\partial z} & \frac{\partial \lambda}{\partial z} \end{bmatrix} \begin{bmatrix} \frac{\partial F}{\partial r} \\ \frac{\partial F}{\partial \phi} \\ \frac{\partial F}{\partial \lambda} \end{bmatrix} \equiv \Big[D(x,y,z)\Big] \begin{bmatrix} \frac{\partial F}{\partial r} \\ \frac{\partial F}{\partial \phi} \\ \frac{\partial F}{\partial \lambda} \end{bmatrix} \tag{4.45}$$

For the same general function F, a chain rule expansion may also be used to find partials with respect to inertial frame coordinates:

$$\frac{\partial F}{\partial X} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial X} + \frac{\partial F}{\partial y}\frac{\partial y}{\partial X} + \frac{\partial F}{\partial z}\frac{\partial z}{\partial X} \tag{4.46}$$

$$\frac{\partial F}{\partial Y} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial Y} + \frac{\partial F}{\partial y}\frac{\partial y}{\partial Y} + \frac{\partial F}{\partial z}\frac{\partial z}{\partial Y} \tag{4.47}$$

$$\frac{\partial F}{\partial Z} = \frac{\partial F}{\partial x}\frac{\partial x}{\partial Z} + \frac{\partial F}{\partial y}\frac{\partial y}{\partial Z} + \frac{\partial F}{\partial z}\frac{\partial z}{\partial Z} \tag{4.48}$$

Written in matrix form, this gives

$$\begin{bmatrix} \frac{\partial F}{\partial X} \\ \frac{\partial F}{\partial Y} \\ \frac{\partial F}{\partial Z} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial X} & \frac{\partial y}{\partial X} & \frac{\partial z}{\partial X} \\ \frac{\partial x}{\partial Y} & \frac{\partial y}{\partial Y} & \frac{\partial z}{\partial Y} \\ \frac{\partial x}{\partial Z} & \frac{\partial y}{\partial Z} & \frac{\partial z}{\partial Z} \end{bmatrix} \begin{bmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{bmatrix} \equiv \Big[C^T(t)\Big] \begin{bmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{bmatrix} \tag{4.49}$$

Combining Eqs. (4.45) and (4.49) gives

$$\begin{bmatrix} \frac{\partial F}{\partial X} \\ \frac{\partial F}{\partial Y} \\ \frac{\partial F}{\partial Z} \end{bmatrix} = \Big[C^T(t)\Big]\Big[D\Big] \begin{bmatrix} \frac{\partial F}{\partial r} \\ \frac{\partial F}{\partial \phi} \\ \frac{\partial F}{\partial \lambda} \end{bmatrix} \tag{4.50}$$

Specializing Eq. (4.50) to the case of spherical harmonic gravity, where U represents the gravity potential, the perturbed acceleration is obtained in inertial coordinates:

$$\begin{bmatrix} a_X \\ a_Y \\ a_Z \end{bmatrix} = \begin{bmatrix} \frac{\partial U}{\partial X} \\ \frac{\partial U}{\partial Y} \\ \frac{\partial U}{\partial Z} \end{bmatrix} = \Big[C^T(t)\Big]\Big[D\Big] \begin{bmatrix} \frac{\partial U}{\partial r} \\ \frac{\partial U}{\partial \phi} \\ \frac{\partial U}{\partial \lambda} \end{bmatrix} \tag{4.51}$$

As shown in the next section, it is more efficient to compute the Jacobian in the ECEF frame, then transform it into the inertial frame.

4.5 Earth-Centered-Earth-Fixed Jacobian: Transformation to Inertial

The most computationally efficient method to integrate the Jacobian matrix is to first compute it in Earth-Centered-Earth-Fixed (ECEF) coordinates, then tranform

it to Earth-Centered-Inertial (ECI) coordinates. This method of computing the Jacobian in ECEF coordinates and transforming into ECI is more computationally efficient, as is evident by observation, because fewer mathematical operations are required compared with computing the Jacobian directly in the ECI frame. These two methods are otherwise identical and provide solutions that are comparatively accurate to machine precision.

For a general rotation matrix $C(t)$, body frame (x, y, z), and inertial frame (X, Y, Z), the following transformations relate these two frames:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [C(t)] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4.52}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \left[C^T(t)\right] \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4.53}$$

The matrix form of the velocity and acceleration kinematics follow by direct differentiation:

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= \left[\dot{C}\right] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \left[C\right] \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \\ &= -\left[\widetilde{\omega}\right]\left[C\right] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \left[C\right] \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \end{aligned} \tag{4.54}$$

where $\left[\dot{C}\right] = -\left[\widetilde{\omega}\right]\left[C\right]$ and $\widetilde{\omega}$ is the skew symmetric matrix

$$\widetilde{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{4.55}$$

Differentiating again results in

$$\begin{aligned} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = &-\left[\dot{\widetilde{\omega}}\right]\left[C\right]\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \left[\widetilde{\omega}\right]\left[\dot{C}\right]\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \left[\widetilde{\omega}\right]\left[C\right]\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \\ &+ \left[\dot{C}\right]\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} + \left[C\right]\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} \end{aligned} \tag{4.56}$$

Because $\dot{\widetilde{\omega}} = 0$ and again using $\left[\dot{C}\right] = -\left[\widetilde{\omega}\right]\left[C\right]$, this equation is rewritten as

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \left[\widetilde{\omega}\right]^2\left[C\right]\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - 2\left[\widetilde{\omega}\right]\left[C\right]\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} + \left[C\right]\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} \tag{4.57}$$

For the orbit problem, $\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x})$, or

$$\frac{d}{dt}\begin{bmatrix} X \\ Y \\ Z \\ \frac{dX}{dt} \\ \frac{dY}{dt} \\ \frac{dZ}{dt} \end{bmatrix} = \begin{bmatrix} \frac{dX}{dt} \\ \frac{dY}{dt} \\ \frac{dZ}{dt} \\ \frac{d^2X}{dt^2} \\ \frac{d^2Y}{dt^2} \\ \frac{d^2Z}{dt^2} \end{bmatrix} \tag{4.58}$$

where

$$\begin{bmatrix} \frac{d^2X}{dt^2} \\ \frac{d^2Y}{dt^2} \\ \frac{d^2Z}{dt^2} \end{bmatrix} = -\begin{bmatrix} \frac{\partial U(x,y,z)}{\partial X} \\ \frac{\partial U(x,y,z)}{\partial Y} \\ \frac{\partial U(x,y,z)}{\partial Z} \end{bmatrix} = -\Big[C^T\Big]\begin{bmatrix} \frac{\partial U(x,y,z)}{\partial x} \\ \frac{\partial U(x,y,z)}{\partial y} \\ \frac{\partial U(x,y,z)}{\partial z} \end{bmatrix} \tag{4.59}$$

Recall that the state transition matrix $\Phi(t, t_0)$ satisfies the differential equation

$$\dot{\Phi}(t, t_0) = \mathbf{A}(t, \mathbf{x})\Phi(t, t_0), \qquad \Phi(t_0, t_0) = \mathbf{I} \tag{4.60}$$

where

$$\mathbf{A}(t, \mathbf{x}) = \begin{bmatrix} 0 & I \\ G & 0 \end{bmatrix} \tag{4.61}$$

This Jacobian may therefore be computed in the ECEF frame first, then transformed into the ECI frame using the transformation

$$G = -\begin{bmatrix} \frac{\partial^2 U}{\partial X^2} & \frac{\partial^2 U}{\partial X \partial Y} & \frac{\partial^2 U}{\partial X \partial Z} \\ \frac{\partial^2 U}{\partial Y \partial X} & \frac{\partial^2 U}{\partial Y^2} & \frac{\partial^2 U}{\partial Y \partial Z} \\ \frac{\partial^2 U}{\partial Z \partial X} & \frac{\partial^2 U}{\partial Z \partial Y} & \frac{\partial^2 U}{\partial Z^2} \end{bmatrix} = -\Big[C^T\Big]\begin{bmatrix} \frac{\partial^2 U}{\partial x^2} & \frac{\partial^2 U}{\partial x \partial y} & \frac{\partial^2 U}{\partial x \partial z} \\ \frac{\partial^2 U}{\partial y \partial x} & \frac{\partial^2 U}{\partial y^2} & \frac{\partial^2 U}{\partial y \partial z} \\ \frac{\partial^2 U}{\partial z \partial x} & \frac{\partial^2 U}{\partial z \partial y} & \frac{\partial^2 U}{\partial z^2} \end{bmatrix}\Big[C\Big] \tag{4.62}$$

which may be written simply as

$$\left[G(X,Y,Z)\right] = -\left[C^T\right]\left[G(x,y,z)\right]\left[C\right] \tag{4.63}$$

Note that the gradient of the gravity potential U:

$$\nabla U = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \\ \frac{\partial U}{\partial z} \end{bmatrix} \tag{4.64}$$

and the Hessian (Jacobian) of U:

$$\nabla^2 U = \begin{bmatrix} \frac{\partial^2 U}{\partial x^2} & \frac{\partial^2 U}{\partial x \partial y} & \frac{\partial^2 U}{\partial x \partial z} \\ \frac{\partial^2 U}{\partial y \partial x} & \frac{\partial^2 U}{\partial y^2} & \frac{\partial^2 U}{\partial y \partial z} \\ \frac{\partial^2 U}{\partial z \partial x} & \frac{\partial^2 U}{\partial z \partial y} & \frac{\partial^2 U}{\partial z^2} \end{bmatrix} \tag{4.65}$$

are first- and second-order Cartesian tensors, respectively. Also, the projections of these operators through an orthogonal transformation, evident in the rightmost of Eqs. (4.59) and (4.62), hold for all first- and second-order Cartesian tensors.

4.6 MCPI STM Results

The STM propagation using MCPI is verified using a variety of checks. The first and second partials of the Associated Legendre Functions (ALFs) are verified using a finite difference method, as shown in Figure 4.1. The second partial of the gravity potential with respect to latitude (which involves the second partial of the ALFs) is shown in Figure (4.2); the relative error is of the order 10^{-14}.

The STM should conform to a simple analytic matrix inverse formula, which means that it is symplectic [54]. The symplectic check is computed using

$$[\Phi]^T[J][\Phi] = [J] \tag{4.66}$$

where

$$J = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ -I_{n\times n} & 0_{n\times n} \end{bmatrix} \tag{4.67}$$

This means that if this equation is premultiplied by $[J]$ and postmultiplied by $[\Phi]^{-1}$, then the matrix inverse of $[\Phi]$ is given by

$$[\Phi]^{-1} = -[J][\Phi]^T[J] \tag{4.68}$$

If the STM matrix is partitioned into $(n \times n)$ submatrices as

$$[\Phi] = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \tag{4.69}$$

then the matrix inverse of Φ may be expressed analytically through the expression

$$[\Phi]^{-1} = \begin{bmatrix} \Phi_{22}^T & -\Phi_{12}^T \\ -\Phi_{21}^T & \Phi_{11}^T \end{bmatrix} \tag{4.70}$$

It can be shown [54] that the only condition necessary for the STM considered here to be symplectic is that $[G] = [G]^T$ must be symmetric. Figure 4.3 gives the error components of the symplectic check for the numerically propagated STM.

A finite difference check is used to verify that each column of the STM is correct. Figure 4.4 shows that the first column is accurate compared with the finite difference check to at least 8 digits; each column gives comparable accuracy but is not shown here to avoid redundancy. Obviously, a finite difference check is not expected to ap-

proach machine precision; 8 digit agreement, together with the additional symplectic check and other validations, is judged to be sufficient in the present discussion. 8 digit precision likely exceeds the "practical" accuracy required for STM applications.

The Jacobian for the STM is computed in two ways: 1) directly in ECI and 2) in ECEF, which is then transformed into ECI. Figure 4.5 shows that these two methods give the same result. See Appendix B for the specific equations used to compute each Jacobian. A timing comparison between these two Jacobian calculation methods is given in Figure 4.6, showing that method 2) is the preferred method.

This formulation permits an arbitrary degree and order gravity model to compute the STM, but most likely degree and order ten for the STM is the maximum that would be needed in practice. Note that Earth rotation is included in all spherical harmonic simulation results.

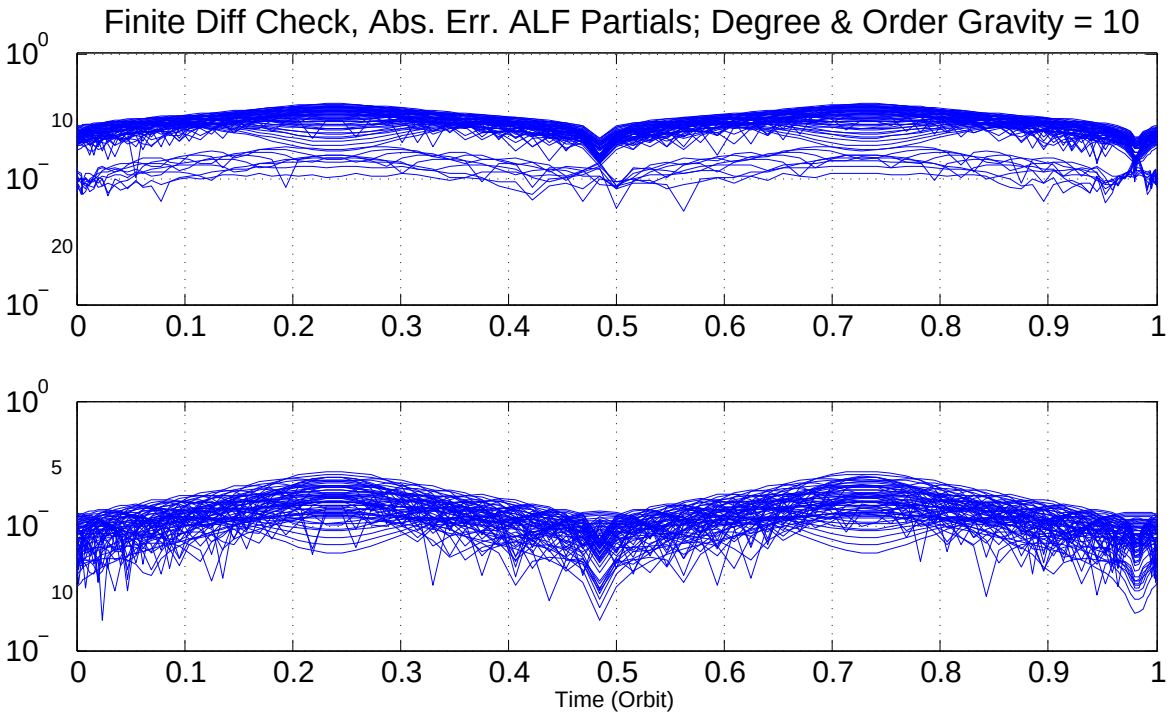


Figure 4.1: Finite Difference Check for First and Second Partials of ALFs (Degree and Order Gravity = 10) [50, 52]

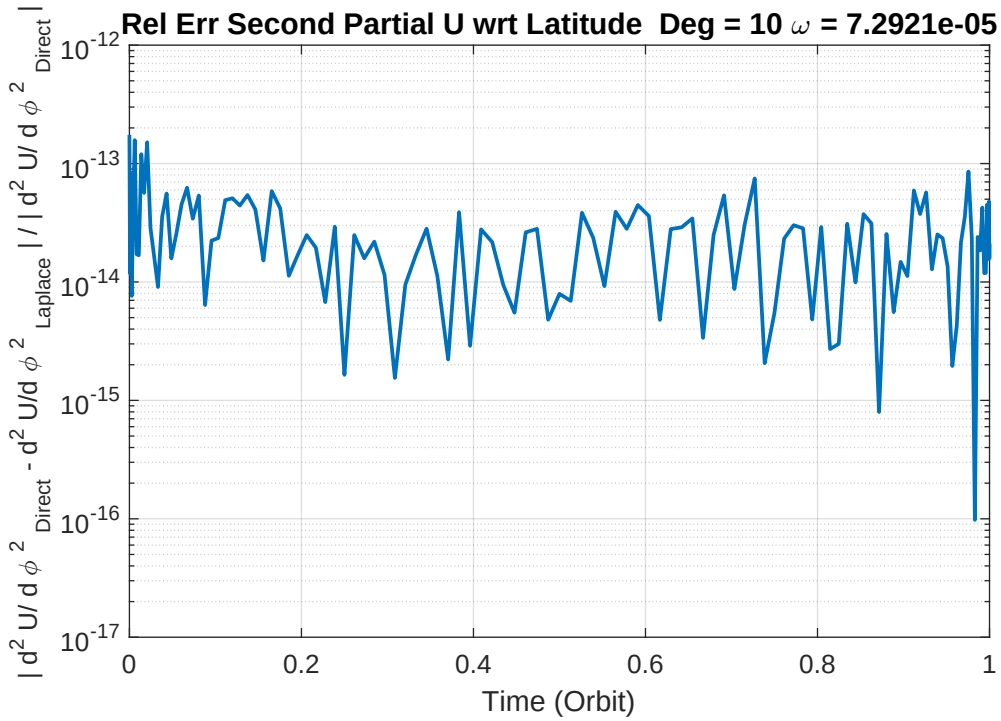


Figure 4.2: Relative Error Check for Second Partial of Gravity Potential with Respect to Latitude (Degree and Order Gravity = 10, Earth Rotation Included); Note This Figure Gives Absolute Value, Compared to Conference and Journal Figures in [50, 52]

4.7 Optimized State Transition Matrix Calculations

The baseline STM algorithm given so far in this chapter may be optimized. Because the STM requires only as input the position vector at every sample node, the trajectory may first be integrated. Next, the STM may be propagated using the converged trajectory solution. This method is more efficient, as is shown by the comparison in Figure 4.7. Additional optimization techniques will be presented in the following two subsections.

4.7.1 Cascade Method

The MCPI method may solve either first- or second-order differential equations. However, some numerical integrators, such as RKN12(10), require a second-order

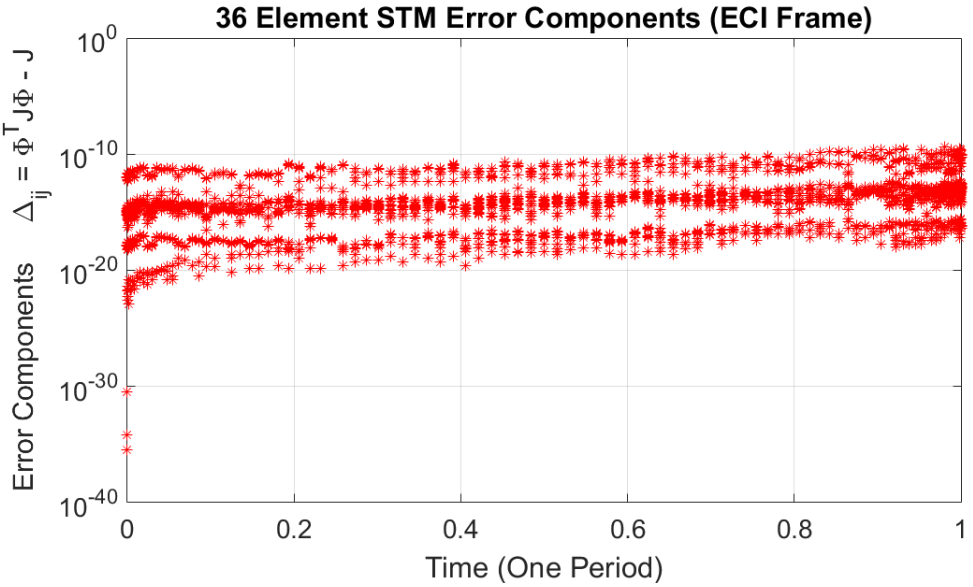


Figure 4.3: Symplectic Check of State Transition Matrix (Degree and Order Gravity = 10) [50, 52]

formulation. To increase efficiency of STM calculations, a second-order differential equation may be used in place of

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(t, \mathbf{x}(t)), \qquad t \in [a, b] \tag{4.71}$$

This method, developed below, is called the MCPI cascade method [5, 52]. The STM differential equation for the conservative case is rearranged to solve a pair of second-order equations as follows. Since $\dot{\Phi} = \mathbf{A}\Phi$, or

$$\begin{bmatrix} \dot{\Phi}_{11} & \dot{\Phi}_{12} \\ \dot{\Phi}_{21} & \dot{\Phi}_{22} \end{bmatrix} = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ G_{3\times3} & 0_{3\times3} \end{bmatrix} \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \tag{4.72}$$

the individual components using the G matrix are written from Eq. (4.4) as

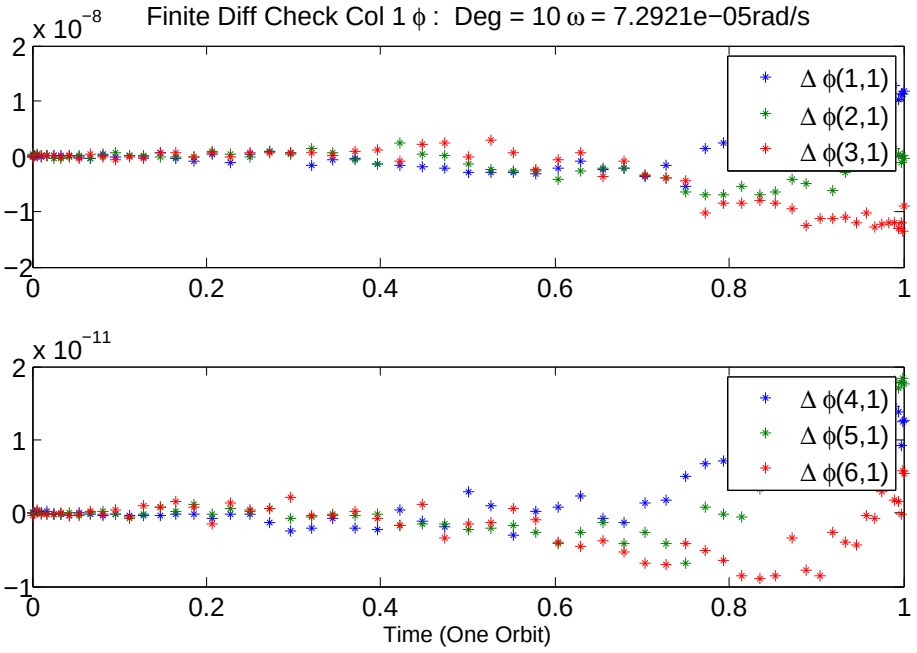


Figure 4.4: Finite Difference Check for First Column of STM Over One Orbit (Degree and Order Gravity = 10) [50, 52]

$$\dot{\Phi}_{11} = \Phi_{21} \tag{4.73}$$

$$\dot{\Phi}_{12} = \Phi_{22} \tag{4.74}$$

$$\dot{\Phi}_{21} = G\Phi_{11} \tag{4.75}$$

$$\dot{\Phi}_{22} = G\Phi_{12} \tag{4.76}$$

Taking the time derivative of Eqs. (4.73) and (4.74) gives

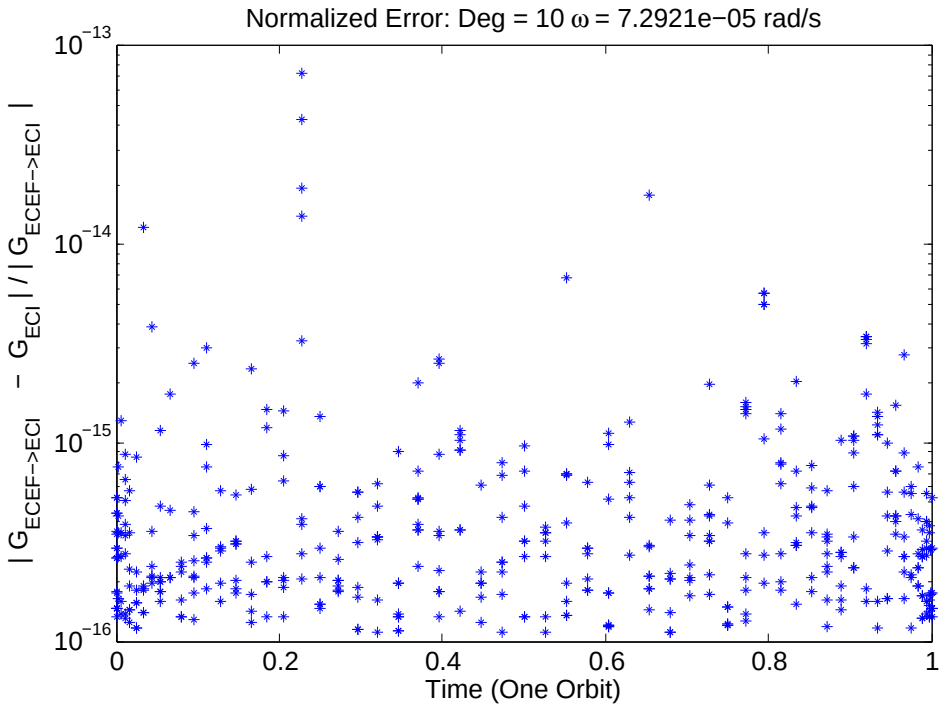


Figure 4.5: Rel Err for G: (ECEF $\rightarrow$ ECI) vs. ECI (Degree and Order Gravity = 10)

$$\ddot{\Phi}_{11} = \dot{\Phi}_{21} \tag{4.77}$$

$$\ddot{\Phi}_{12} = \dot{\Phi}_{22} \tag{4.78}$$

Substituting Eq. (4.75) into Eq. (4.77) and Eq. (4.76) into Eq. (4.78) gives the two second-order differential equations required for integration:

$$\ddot{\Phi}_{11} = G\Phi_{11}, \quad \Phi_{11}(t_0, t_0) = I \tag{4.79}$$

$$\ddot{\Phi}_{12} = G\Phi_{12}, \quad \Phi_{12}(t_0, t_0) = 0 \tag{4.80}$$

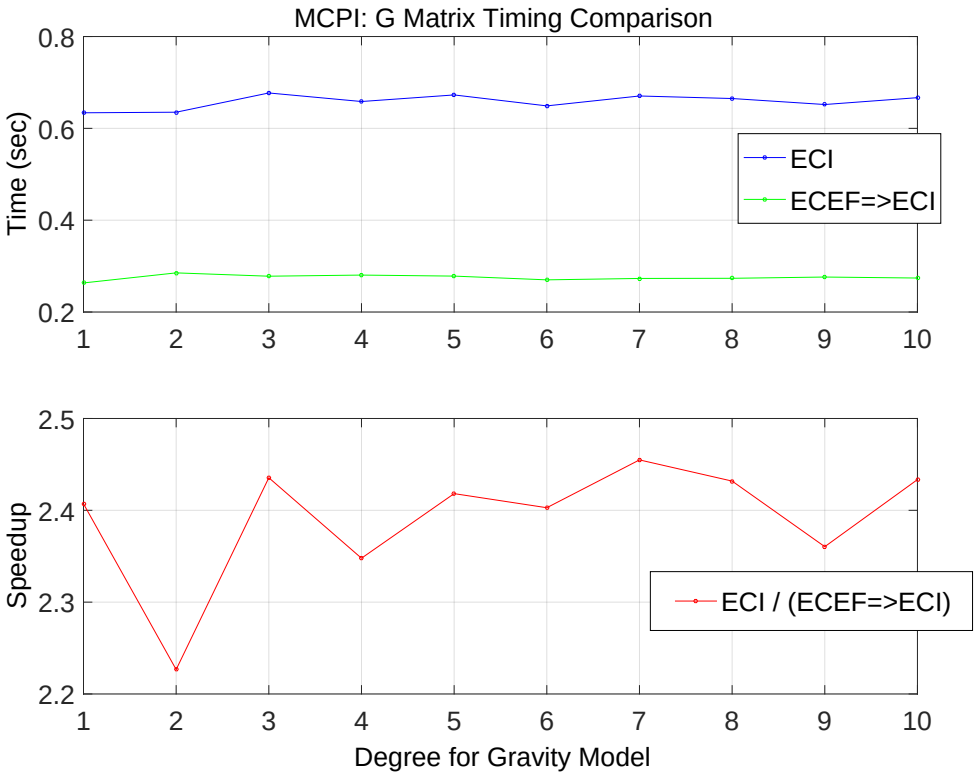


Figure 4.6: Timing Comparison for G: (ECEF $\rightarrow$ ECI) vs. ECI (Degree and Order Gravity = 10)

The other two sub-matrices of the STM, Φ_{21} and Φ_{22}, are obtained from the converged solution as

$$\Phi_{21} = \dot{\Phi}_{11} \tag{4.81}$$

$$\Phi_{22} = \dot{\Phi}_{12} \tag{4.82}$$

Note the cascade solution of Eqs. (4.79), (4.80) for Φ_{11} and Φ_{12} additionally produces the first time derivatives of $\dot{\Phi}_{11}$ and $\dot{\Phi}_{12}$. Φ_{21} and Φ_{22} are obtained from Eqs. (4.81), (4.82) without further computation. The initial condition for the STM is known to be

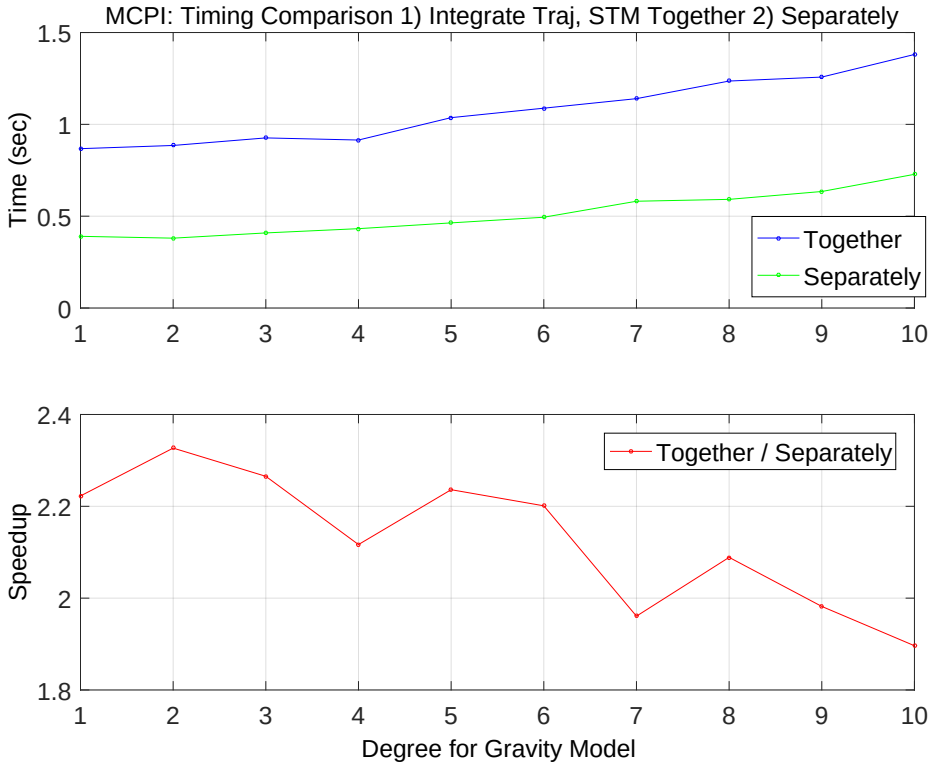


Figure 4.7: Timing Comparison of Trajectory and STM: Propagated Both Simultaneously and Separately Over One Orbit [52]

$$\begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} = \begin{bmatrix} I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{bmatrix} \tag{4.83}$$

so for the second-order formulation, the initial conditions are

$$[\Phi_{11} \quad \Phi_{12}] = [I_{3\times3} \quad 0_{3\times3}] \tag{4.84}$$

$$\begin{bmatrix} \dot{\Phi}_{11} & \dot{\Phi}_{12} \end{bmatrix} = \begin{bmatrix} \Phi_{21} & \Phi_{22} \end{bmatrix} = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \end{bmatrix} \tag{4.85}$$

The initial conditions automatically satisfied are $\Phi_{21} = 0_{3\times3}$ and $\Phi_{22} = I_{3\times3}$. The above results for the cascade method can be readily generalized to accommodate velocity dependence in the force model (e.g., drag). In the case of velocity dependence,

the equation $\dot{\Phi} = \mathbf{A}\Phi$ generalizes as

$$\begin{bmatrix} \dot{\Phi}_{11} & \dot{\Phi}_{12} \\ \dot{\Phi}_{21} & \dot{\Phi}_{22} \end{bmatrix} = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ G_{3\times3} & D_{3\times3} \end{bmatrix} \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \tag{4.86}$$

where

$$D_{3\times3} = \begin{bmatrix} \frac{\partial \mathbf{g}(t,\mathbf{r})}{\partial \dot{\mathbf{r}}} \end{bmatrix} \tag{4.87}$$

The resulting formulation is the same except for two additional terms of the form

$$\ddot{\Phi}_{11} = G\Phi_{11} + D\frac{\dot{\Phi}_{11}}{dt} \tag{4.88}$$

$$\ddot{\Phi}_{12} = G\Phi_{12} + D\frac{\dot{\Phi}_{12}}{dt} \tag{4.89}$$

Similar to the standard MCPI approach, computation time may be reduced by first obtaining a solution for the trajectory and then propagating the STM rather than propagating both at the same time, as shown in Figure (4.8); the advantage grows with increasing gravity model degree, approaching 40% if degree and order 10 is used.

It is important to note the pragmatic truth that may require the state history accurate to better than 11 digits, but it is very rare that the STM is required to be accurate to more than 4 digits. This means that the conclusion in Figure (4.7) would change dramatically if a $(200, 200)$ gravity were required to obtain 11 digit state history, but only 5 digit accurate STM. Experiments indicate that a 4 digit STM can be obtained with degree and order gravity less than 10. Therefore, the"separate" solution is expected to be even more dramatically advantageous for this case.

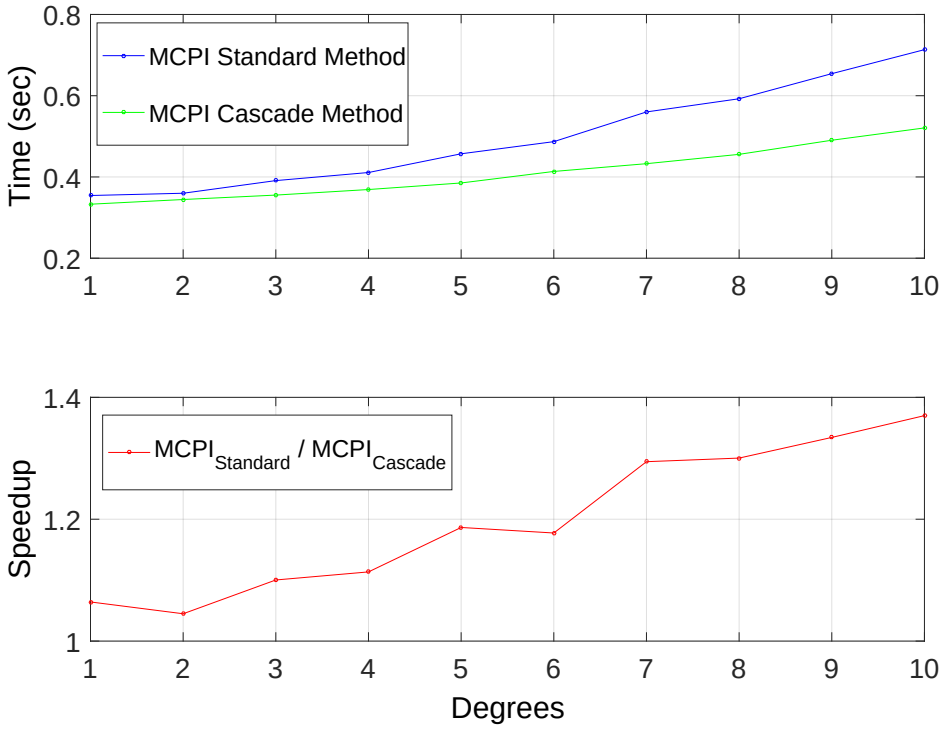


Figure 4.8: Timing Comparison of Baseline vs. Cascade Method for Computing Trajectory and Subsequently STM From Converged Position [52]

4.7.2 State Transition Matrix in Canonical Units

Using canonical units often allows many computations in astrodynamics to be simplified [54]. For instance, if Cartesian coordinate are used, the position values are typically several orders of magnitude larger than the velocity values. Converting to canonical units effectively normalizes the values so that they have relatively the same number of significant figures. In addition, the range of exponents for canonical values are much smaller than those given in metric units, which results in computational advantages to produce a precise numerical integration in the event that fixed point arithmetic, for some onboard computers, is used. More information about the Cartesian conversion to and from canonical units is given in Appendix D.

The STM may also be converted to and from canonical units. Consider the

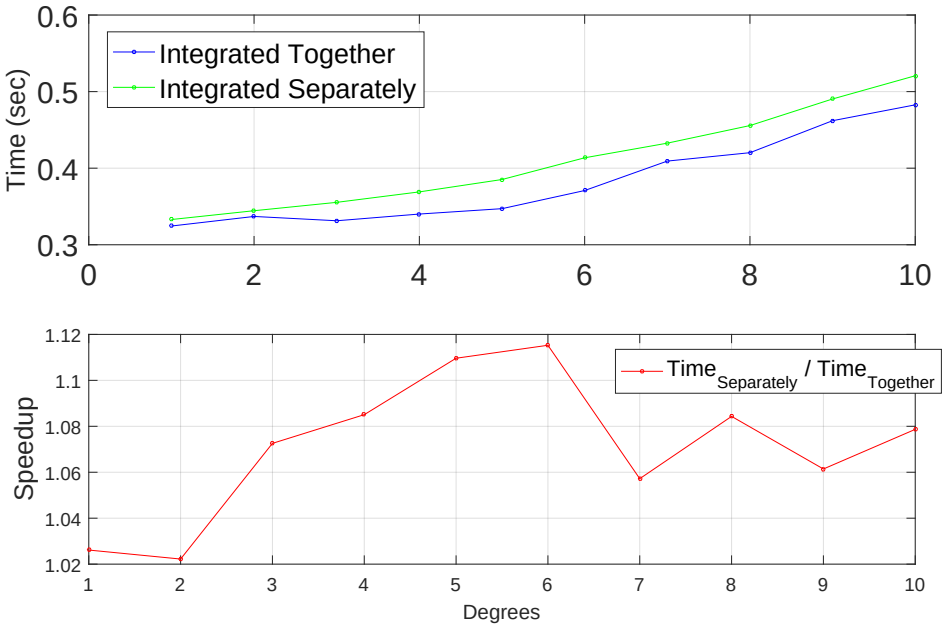


Figure 4.9: Timing Comparison of Cascade Method for Trajectory and Subsequently STM From Converged Position vs. Computing Trajectory and STM Together [52]

perturbed two-body problem,

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{a}_d \tag{4.90}$$

and manipulate this equation as follows using $r_\oplus = 1$ Earth radius:

$$r_\oplus \frac{d^2\left(\frac{\mathbf{r}}{r_\oplus}\right)}{dt^2} = -\frac{\frac{\mu_\oplus}{r_\oplus^3}}{\left(\frac{r}{r_\oplus}\right)^3} r_\oplus \left(\frac{\mathbf{r}}{r_\oplus}\right) + \mathbf{a}_d \tag{4.91}$$

Next, introduce a non-dimensional set of coordinates by defining

$$\vec{\mathcal{R}} \equiv \frac{\mathbf{r}}{r_\oplus} \tag{4.92}$$

and

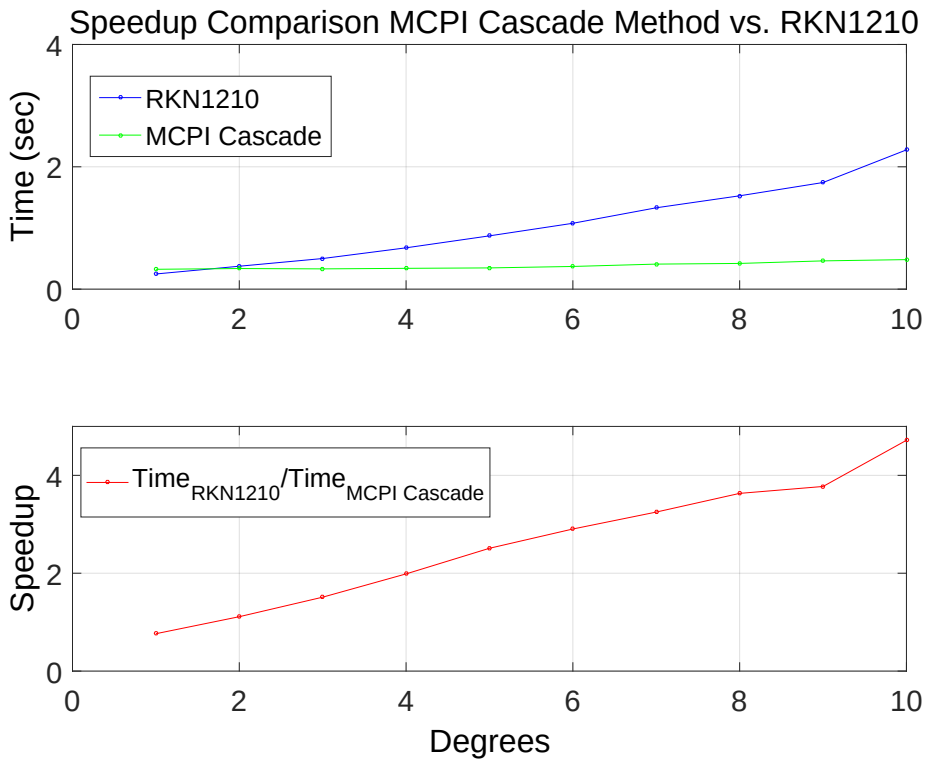


Figure 4.10: Timing Comparison for Computing Trajectory and STM of RK12(10) with MCPI Cascade Method (Traj and STM Integrated Together) [52]

$$|\vec{\mathcal{R}}| = \mathcal{R} \equiv \frac{r}{r_\oplus} \tag{4.93}$$

This means that the Cartesian coordinates may be expressed in terms of canonical terms using

$$r = r_\oplus \mathcal{R} \tag{4.94}$$

$$x = r_\oplus \mathcal{X} \quad x \to y, z; \quad \mathcal{X} \to \mathcal{Y}, \mathcal{Z} \tag{4.95}$$

Now the equations of motion become

$$r_\oplus \frac{d^2\vec{\mathcal{R}}}{dt^2} = -\frac{\frac{\mu_\oplus}{r_\oplus^2}}{\mathcal{R}^3}\vec{\mathcal{R}} + \mathbf{a}_d \tag{4.96}$$

Multiply both sides by $\frac{r_\oplus^2}{\mu_\oplus}$ to get

$$\frac{d^2\vec{\mathcal{R}}}{\left(\frac{\mu_\oplus}{r_\oplus^3}\right)dt^2} = -\frac{1}{\mathcal{R}^3}\vec{\mathcal{R}} + \frac{r_\oplus^2}{\mu_\oplus}\mathbf{a}_d \tag{4.97}$$

Next, define

$$\mathbf{A}_d \equiv \frac{r_\oplus^2}{\mu_\oplus}\mathbf{a}_d \tag{4.98}$$

and define the canonical time unit as

$$\tau = \sqrt{\frac{\mu_\oplus}{r_\oplus^3}}t \tag{4.99}$$

Taking the derivative of time with respect to τ and solving for $d\tau^2$ gives

$$\frac{dt}{d\tau} = \sqrt{\frac{r_\oplus^3}{\mu_\oplus}} \tag{4.100}$$

$$d\tau^2 = \left(\frac{\mu_\oplus}{r_\oplus^3}\right)dt^2 \tag{4.101}$$

This lets Equation (4.97) be written as

$$\frac{d^2\vec{\mathcal{R}}}{d\tau^2} = -\frac{1}{\mathcal{R}^3}\vec{\mathcal{R}} + \mathbf{A}_d \tag{4.102}$$

where the canonical acceleration may computed from its Cartesian value as

$$\mathbf{A}_d = \left(\frac{r_\oplus^2}{\mu}\right)\mathbf{a}_d \tag{4.103}$$

and also,

$$\frac{d\vec{\mathcal{R}}}{d\tau} = \sqrt{\frac{r_\oplus^3}{\mu_\oplus}}\frac{d\mathbf{r}}{dt} \tag{4.104}$$

To find the first derivative terms for X, Y, Z, expand by using the chain rule $(X \to Y, Z)$:

$$\begin{aligned}\frac{dX}{dt} &= \frac{d}{dt}(r_\oplus x) = r_\oplus\frac{dx}{dt} = r_\oplus\frac{dx}{d\tau}\frac{d\tau}{dt} = r_\oplus\frac{dx}{d\tau}\sqrt{\frac{\mu_\oplus}{r_\oplus^3}} \\ &= \sqrt{\frac{\mu_\oplus}{r_\oplus}}\frac{dx}{d\tau}\end{aligned} \tag{4.105}$$

Now, the state vector is rewritten as

$$\mathbf{r} = \begin{bmatrix} X \\ Y \\ Z \\ \frac{dX}{dt} \\ \frac{dY}{dt} \\ \frac{dZ}{dt} \end{bmatrix} = \begin{bmatrix} r_\oplus I & 0 \\ 0 & \sqrt{\frac{\mu_\oplus}{r_\oplus}} I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \frac{dx}{d\tau} \\ \frac{dy}{d\tau} \\ \frac{dz}{d\tau} \end{bmatrix} \equiv [\mathbf{M}] \begin{bmatrix} x \\ y \\ z \\ \frac{dx}{d\tau} \\ \frac{dy}{d\tau} \\ \frac{dz}{d\tau} \end{bmatrix} \tag{4.106}$$

If

$$\mathbf{X} = \begin{bmatrix} \mathbf{r} \\ \dot{\mathbf{r}} \end{bmatrix} \quad , \quad \chi = \begin{bmatrix} \vec{\mathcal{R}} \\ \dot{\vec{\mathcal{R}}} \end{bmatrix} \tag{4.107}$$

then Eq. (4.106) becomes

$$\mathbf{X} = [\mathbf{M}]\vec{\chi} \tag{4.108}$$

and for the time-varying system,

$$\mathbf{X}(t) = [\mathbf{M}]\vec{\chi}(\tau) \tag{4.109}$$

For the STM in canonical units (Ψ),

$$\Phi(t, t_0) = \frac{\partial \mathbf{X}(t)}{\partial \mathbf{X}(t_0)} \quad \Longleftrightarrow \quad \frac{\partial \vec{\chi}(\tau)}{\partial \vec{\chi}(\tau_0)} \equiv \Psi(\tau, \tau_0) \tag{4.110}$$

These equations may be related, by using Eq. (4.109), as

$$\begin{aligned} \left[\frac{\partial \mathbf{X}(t)}{\partial \mathbf{X}(t_0)}\right] &= [\mathbf{M}]\left[\frac{\partial \vec{\chi}(\tau)}{\partial \mathbf{X}(t_0)}\right] \\ &= [\mathbf{M}]\left[\frac{\partial \vec{\chi}(\tau)}{\partial \vec{\chi}(\tau_0)}\right]\left[\frac{\partial \vec{\chi}(\tau_0)}{\partial \mathbf{X}(t_0)}\right] \end{aligned} \tag{4.111}$$

Note that because M is a constant matrix,

$$\frac{\partial [\mathbf{M}]}{\partial \mathbf{X}(t_0)} = 0 \tag{4.112}$$

From Eq. (4.109),

$$\mathbf{X}(t_0) = [\mathbf{M}]\vec{\chi}(\tau_0) \tag{4.113}$$

which means that

$$\frac{\partial \vec{\chi}(\tau_0)}{\partial \mathbf{X}(t_0)} = [\mathbf{M}]^{-1} \tag{4.114}$$

Then Eq. (4.111) can be written as

$$\left[\frac{\partial \mathbf{X}(t)}{\partial \mathbf{X}(t_0)}\right] = [\mathbf{M}]\left[\frac{\partial \vec{\chi}(\tau)}{\partial \vec{\chi}(\tau_0)}\right][\mathbf{M}]^{-1} \tag{4.115}$$

This gives the final form for the cartesian STM Φ in terms of the canonical STM Ψ:

$$\Phi(t, t_0) = [\mathbf{M}]\Psi(\tau, \tau_0)[\mathbf{M}]^{-1} \tag{4.116}$$

This equation is then expanded into submatrices:

$$\Phi(t, t_0) = \begin{bmatrix} r_\oplus I & 0 \\ 0 & \sqrt{\frac{\mu_\oplus}{r_\oplus}} I \end{bmatrix} \Psi(\tau, \tau_0) \begin{bmatrix} \frac{1}{r_\oplus} I & 0 \\ 0 & \sqrt{\frac{r_\oplus}{\mu_\oplus}} I \end{bmatrix} \tag{4.117}$$

$$\begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} = \begin{bmatrix} \Psi_{11} & \frac{r_\oplus^{\frac{3}{2}}}{\sqrt{\mu_\oplus}}\Psi_{12} \\ \frac{\sqrt{\mu_\oplus}}{r_\oplus^{\frac{3}{2}}}\Psi_{21} & \Psi_{22} \end{bmatrix} \tag{4.118}$$

To convert the cartesian STM to canonical units, from Eq. (4.116) which means that

$$\Psi(t, t_0) = [\mathbf{M}]^{-1}\Phi(\tau, \tau_0)[\mathbf{M}] \tag{4.119}$$

The STM is very efficient to numerically integrate, so using the canonical form to propagate the STM by itself doesn't provide noticeable improved efficiency; however, more complicated algorithms that require the use of the STM (such as a gradient method) that are computed using canonical units may see some improvement in efficiency.

4.8 Chapter Summary

A gap exists in the literature, and especially in available software for computing the STM, when considering a general spherical harmonic expansion. Namely, the gap is the absence of an STM algorithm that allows user-specified spherical harmonic gravity perturbations to be included. This chapter and the validated software developed address this gap.

The derivations and expressions needed to compute the State Transition Matrix (STM) for the general spherical harmonic gravity model were given, and results were shown to verify that the propagation using MCPI is correct. The Jacobian may be computed two ways: 1) directly in ECI, or 2) in ECEF and transformed into ECI. Method 2) is shown to give the same result as method 1) to machine precision, but with reduced computation time. The mathematical expressions for computing the Jacobian are given in detail in Appendix B for the two-body case, the zonal harmonic gravity case, and the full spherical harmonic gravity case; also included in this Appendix is the derivation of the first and second partials of the Associated Legendre Functions required for the STM computation.

The STM propagation formulation is verified using a finite difference check and also a spot check of the STM group properties. Additional optimization methods are provided in the later parts of this Chapter to further reduce computation time.

Zonal and spherical harmonic versions of the STM calculations are used in Chapter 5 for a Monte Carlo analysis to compute a local Taylor Series gravity approximation, reducing the number of full gravity computations. During MEE propagation, a conversion to Cartesian coordinates very Picard iteration is required to calculate the gravity acceleration; therefore, the STM as expressed using Cartesian coordinates may be directly computed without additional nonlinear transformations.

5. MONTE CARLO ANALYSIS USING MODIFIED EQUINOCTIAL ORBITAL ELEMENTS*

This chapter gives a Monte Carlo analysis that utilizes a local Taylor Series model to reduce the computational time for the general spherical harmonic gravity model and provides a high-accuracy solution through propagating the Modified Equinoctial Elements (MEEs) [51]. This set of elements has proven to be well-suited to MCPI propagation and increases the domain of convergence compared with using Cartesian coordinates [48, 49].

5.1 Taylor Series Gravity Expansion

A local Taylor Series method is implemented for Monte Carlo simulation; this method has previously shown to decrease the computation time of Cartesian MCPI iterations for Monte Carlo simulations [38]. The idea is motivated by this consideration: using the fixed nodes of the nominal solution as expansion points for a low degree, local Taylor series of the spherical harmonic gravity model, a high accuracy neighboring gravity can be computed less expensively than using the full spherical harmonic gravity on the neighboring orbits. A spherical harmonic gravity field of degree and order (40,40) is used for the first MCPI iterations, to converge to the nominal trajectory. Once MCPI converges on this solution, local states, Jacobians, and accelerations are stored in memory at every MCPI node for computation of the

*Part of the data reported in this chapter is reprinted with permission of AAS from "Monte Carlo Propagation of Orbital Elements Using Modified Chebyshev Picard Iteration"; This paper was originally presented at the 26th AAS/AIAA Space Flight Mechanics Meeting held February 1418, 2016, Napa, California, U.S.A., and was originally published in the American Astronautical Society (AAS) publication Spaceflight Mechanics 2016, edited by Renato Zanetti, Ryan P. Russell, Martin T. Ozimek and Angela L. Bowes, American Astronautical Society (AAS) Advances in the Astronautical Sciences, Volume 158, Part III, 2016, pp. 2589-2604 (Copyright 2016 by American Astronautical Society Publications Office, P.O. Box 28130, San Diego, CA 92198, U.S.A.; Web Site: http://www.univelt.com)' [51]

local Taylor series approximation near each node; the coefficients (derivations) of a one or two term Taylor series is also stored at each node.

The local Taylor series approximation is justified because all of the terminal iterations are in the close neighborhood of the final converged solution, as are the neighboring Monte Carlo trajectories; prior simulations have established the maximum distance away from the expansion point to invalidate the local Taylor series approximation to a given accuracy. Thus, the efficiently computable Taylor series gravity approximation compares with known precision to the high-fidelity (spherical harmonic) full gravity solution. The low-fidelity (zonal) gravity is updated using the Taylor series in the following manner with Cartesian coordinates, where $\mathbf{g_F}$ is the full spherical harmonic gravity, $\mathbf{g_z}$ is the zonal gravity, $\mathbf{n}$ is the position under consideration (at a sample node) for the neighboring approximating trajectory, $\mathbf{n_0}$ is the position (a typical Taylor expansion point) used to compute the nominal trajectory, and ∇ is the gradient function:

$$[\mathbf{g_F}(\mathbf{n}) - \mathbf{g_z}(\mathbf{n})] \cong [\mathbf{g_F}(\mathbf{n_0}) - \mathbf{g_z}(\mathbf{n_0})] + \nabla[\mathbf{g_F}(\mathbf{n_0}) - \mathbf{g_z}(\mathbf{n_0})][\mathbf{n} - \mathbf{n_0}] + \mathbf{H.O.T.} \tag{5.1}$$

Rearranging this equation and neglecting the higher-order terms provides an approximation of the full gravity that allows for more efficient computation of many Monte Carlo simulations:

$$\mathbf{g}_F(\mathbf{n}) = \mathbf{g}_z(\mathbf{n}) + \mathbf{c_0} + [A_0][\Delta\mathbf{n}] \tag{5.2}$$

where a constant term, plus the gradient times the position difference, are used:

$$\mathbf{c_0} = [\mathbf{g_F}(\mathbf{n_0}) - \mathbf{g_z}(\mathbf{n_0})] \tag{5.3}$$

$$A_0 = \nabla[\mathbf{g}_F(\mathbf{n}) - \mathbf{g}_z(\mathbf{n})]\Big|_{\mathbf{n_0}} \equiv \frac{\partial(\mathbf{g}_F(\mathbf{n}) - \mathbf{g}_z(\mathbf{n}))}{\partial \mathbf{n}}\Big|_{\mathbf{n_0}} \tag{5.4}$$

$$\Delta\mathbf{n} = [\mathbf{n} - \mathbf{n_0}] \tag{5.5}$$

To compute the gradient difference term A_0, two Jacobians are computed as outlined in Ch. 4 and Appendix B. The low fidelity Jacobian is computed using the fully spherical harmonic gravity, but only with degree and order six. The high fidelity Jacobian is computed using the spherical harmonic gravity model for the desired degree and order. These Jacobians are computed in the ECEF frame, then transformed to the ECI frame for integration because this was found to be slightly more efficient than computing the Jacobian directly in the ECI frame.

An overview of the Monte Carlo computations using this Taylor Series expansion scheme is given in Figure (5.1).

5.2 Monte Carlo Simulation Results

5.2.1 Matlab R2013a

Simulation results are obtained on a Windows 8 machine using Matlab R2013a, where all MCPI results are tuned such that the best performance is achieved while still maintaining an energy check (constant Hamiltonian). For this study, two cases are considered: LEO (e = 0.1) and MEO (e = 0.3). Figures (5.6) - (5.9) show the Monte Carlo point clouds for both the MEE solution and the ECI solution (integration of the Cartesian acceleration form of the equations of motion using ECI coordinates) for the LEO case. The Cartesian and MEE Solutions match to 10

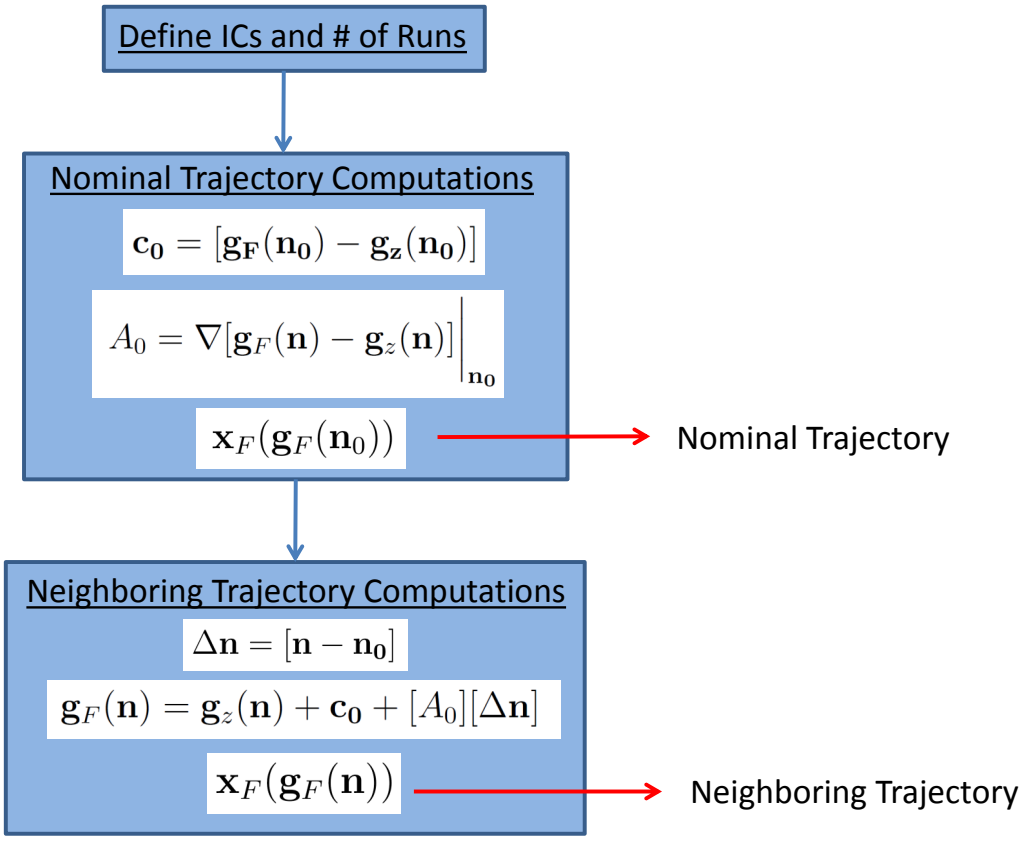


Figure 5.1: Flowchart for Taylor Series Gravity Monte Carlo

digits of accuracy in the final state point clouds; the overlapping points showing different colors is simply an artifact of the graphics (a 3-D rotation in Matlab shows that they overlap very well). Implementation on a compute cluster further increases the efficiency of this method, as described in the next section. All Monte Carlo trajectories may be implemented in parallel on the compute cluster, decreasing the computation time.

Figure (5.10) shows the speedup achieved when using the local Taylor Series gravity model vs. using spherical harmonic gravity for a one-orbit LEO case (integrating ECI coordinates with a "Cowell-like" integration of Cartesian coordinates using MCPI). A study on a compute cluster using only the constant offset term of

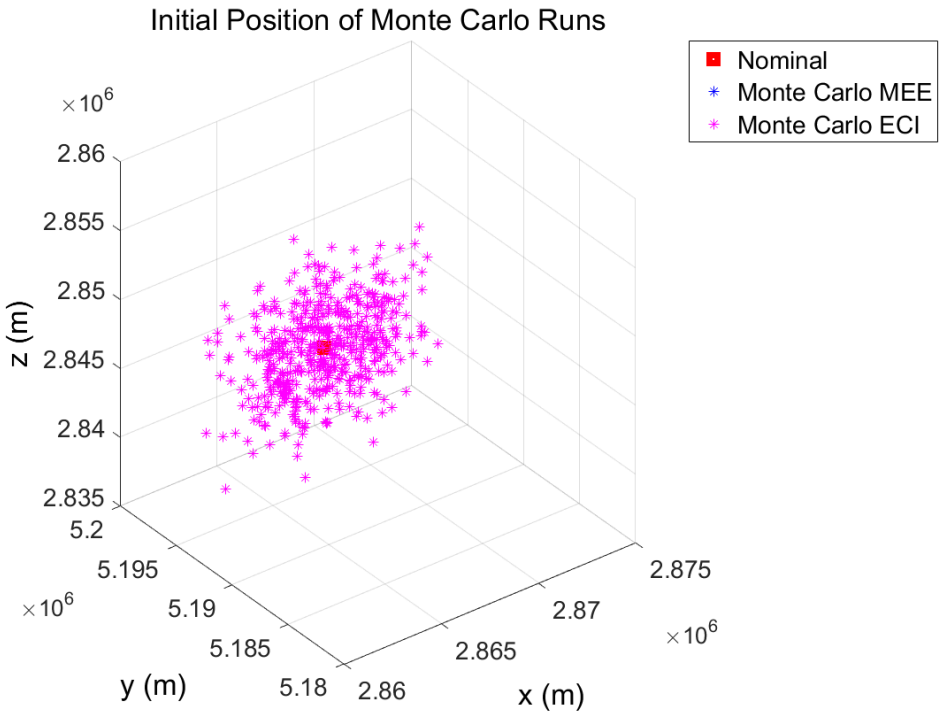


Figure 5.2: Initial Position Point Cloud for Monte Carlo Simulation (LEO) Comparing MEE Solutions with Cartesian Solutions, for 1000 Data Points [51]

the Taylor Series expansion, for intermediate MCPI iterations on each neighboring trajectory, shows some speedup as well.

Note that the MEE propagation requires a transformation to position and velocity at every Picard iteration in order to compute the acceleration, so computing the Jacobian using Cartesian coordinates is a straightforward extension of the IVP code.

5.2.2 Compute Cluster

The simulation results presented in this section were executed on the Texas A&M University's Land, Air, and Space Robotics (LASR) Laboratory Space Situational Awareness (SSA) Cluster. The LASR SSA Cluster is a 16 node compute cluster dedicated to astrodynamics research. Each of the compute nodes has a pair of Intel Xeon 2.6GHz CPUs and 64GB of RAM. In total the LASR SSA Cluster has 192 cores

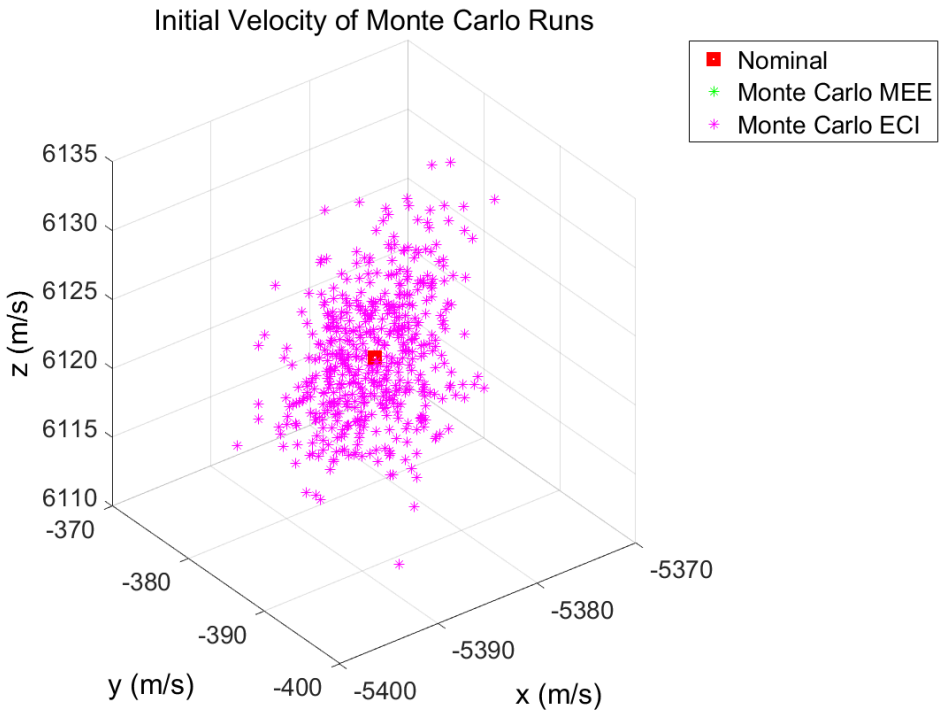


Figure 5.3: Initial Velocity Point Cloud for Monte Carlo Simulation (LEO) Comparing MEE Solutions with Cartesian Solutions, for 1000 Data Points [51]

and a theoretical maximum compute capacity of approximately 1.99 TeraFlops. A Message Passing Inteface (MPI) is used for this configuration. The full specs are given in Appendix E

This simulation takes advantage of existing MCPI optimization schemes such as segmenting one orbit per segment, using radial adaptive gravity, and a version of Taylor Series gravity that incorporates only the constant offset term [35, 38, 45]. However, this study currently computes the full gravity at least once for each trajectory, rather than just for the nominal trajectory, while incorporating the Taylor Series gravity for intermediate iterations. The speedup was only 50% due to various parallel computation overhead issues and also the high level of optimization of the serial code. Future simulations could expand such that the same method is

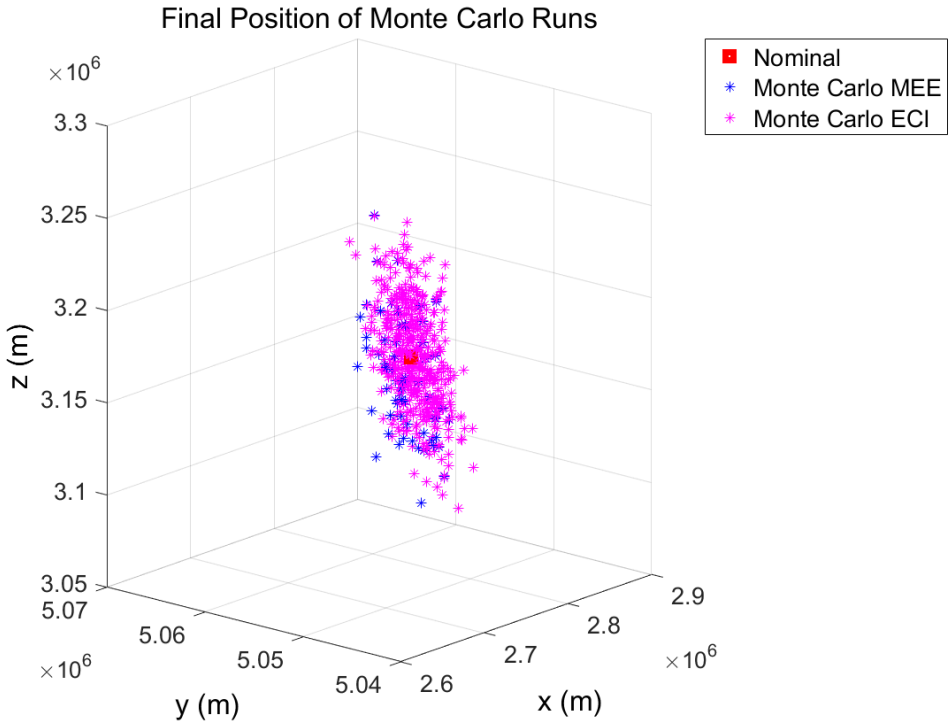


Figure 5.4: Final Position Point Cloud for Monte Carlo Simulation (LEO) Comparing MEE Solutions with Cartesian Solutions, for 1000 Data Points [51]

used as in the Matlab studies, to include the second term of the Taylor Series expansion (which includes the Jacobian term) and to use this method for computing trajectories neighboring the nominal. One million MEO trajectories are used for this simulation,* where for the particular case displayed, the reference orbit initial conditions are specified to be

$$\mathbf{r_0}^T = [-6365.554 \quad 2087.458 \quad 878.918]\, km \tag{5.6}$$

$$\mathbf{v_0}^T = [-1.635 \quad -6.597762 \quad 3.5058499]\, km/s \tag{5.7}$$

*Credit is due to fellow PhD students Austin Probe and Abhay Masher, who helped with the setup of this simulation

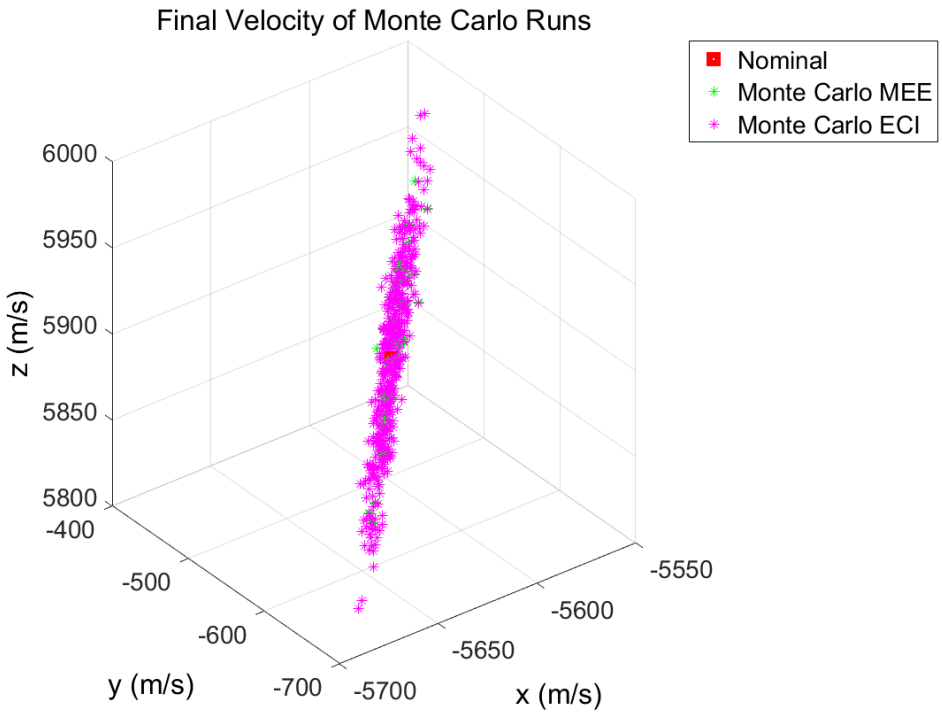


Figure 5.5: Final Velocity Point Cloud for Monte Carlo Simulation (LEO) Comparing MEE Solutions with Cartesian Solutions, for 1000 Data Points [51]

In principle, the nominal Taylor series can be used rather than recalculating the trajectories on each Monte Carlo iteration. The current process was done to ensure Taylor series errors were negligible and for programming convenience. Given that the Monte Carlo precision does not provide more than 7 digit accuracy to obtain valid statistics, it is evident that the nominal trajectory Taylor series will suffice in most cases with a speedup by a factor of two or more.

The initial and final position of the set of Monte Carlo trajectories are shown in Figure (5.11). The total computation time for the Cartesian case is 58,478 seconds, while the total computation time for the MEE Case is 52,969 seconds. When taking into account that the cluster utilizes 192 cores for each simulation, this gives a computation time of about 5.1 minutes for the Cartesian case and about 4.6 minutes for the

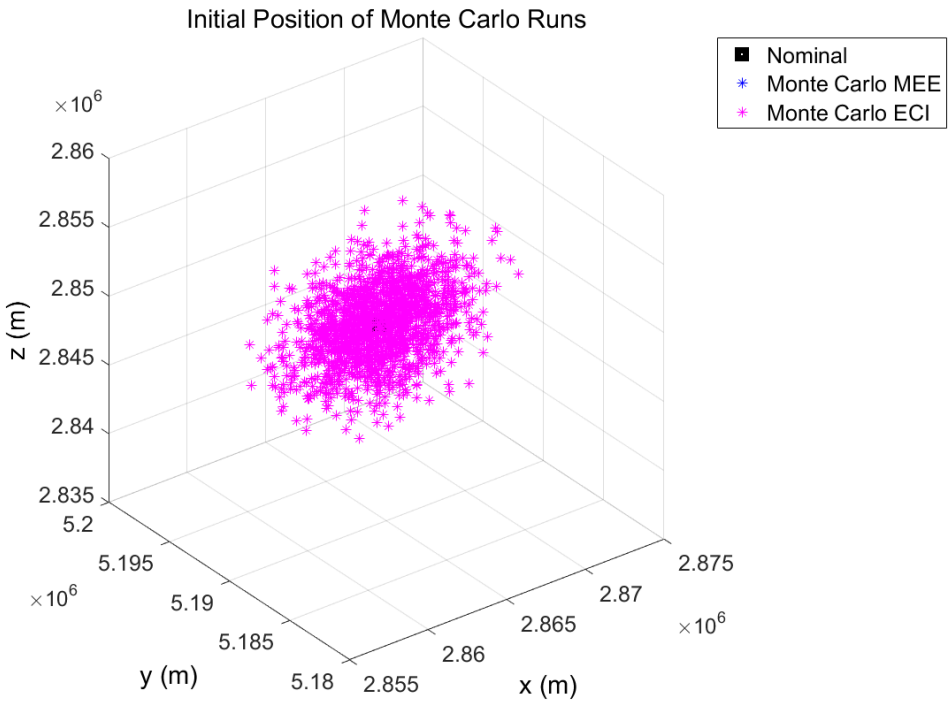


Figure 5.6: Initial Position Point Cloud for Monte Carlo Simulation (MEO) Comparing MEE Solutions with Cartesian Solutions, for 1000 Data Points [51]

MEE case (only a 10% reduction in computation time). Thus, the speedup was not dramatic in this implementation. However, the MEE formulation has several attractive properties that make them alternative conditions for various applications, such as multi-revolution optimal low-thrust orbit transfers. Of significance, the Cartesian MCPI code has been the subject of several years' optimization; it is anticipated that the MEE propagation can be further optimized and will likely improve the speeedup in the process.

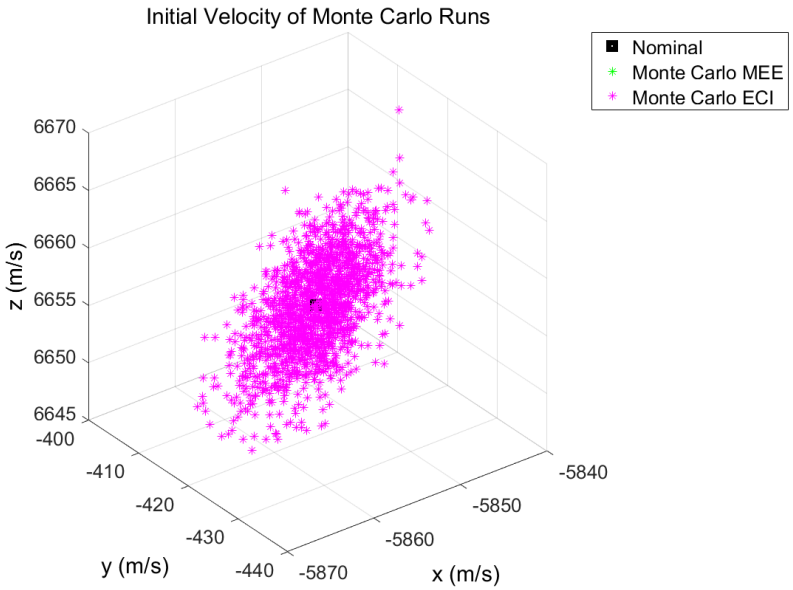


Figure 5.7: Initial Velocity Point Cloud for Monte Carlo Simulation (MEO) Comparing MEE Solutions with Cartesian Solutions, for 1000 Data Points [51]

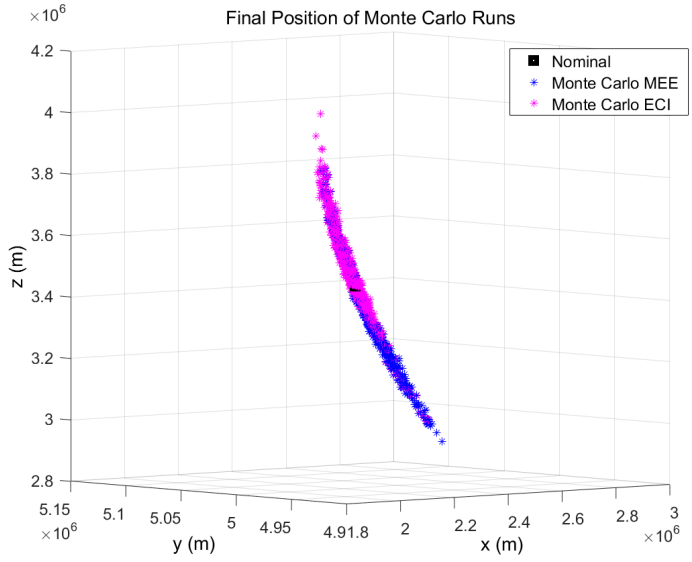


Figure 5.8: Final Position Point Cloud for Monte Carlo Simulation (MEO) Comparing MEE Solutions with Cartesian Solutions, for 1000 Data Points [51]

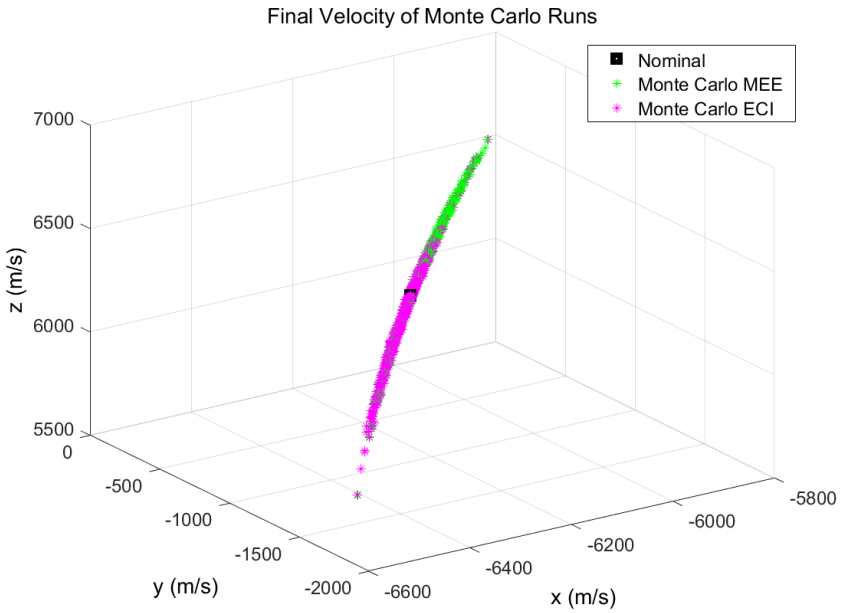


Figure 5.9: Final Velocity Point Cloud for Monte Carlo Simulation (MEO) Comparing MEE Solutions with Cartesian Solutions, for 1000 Data Points [51]

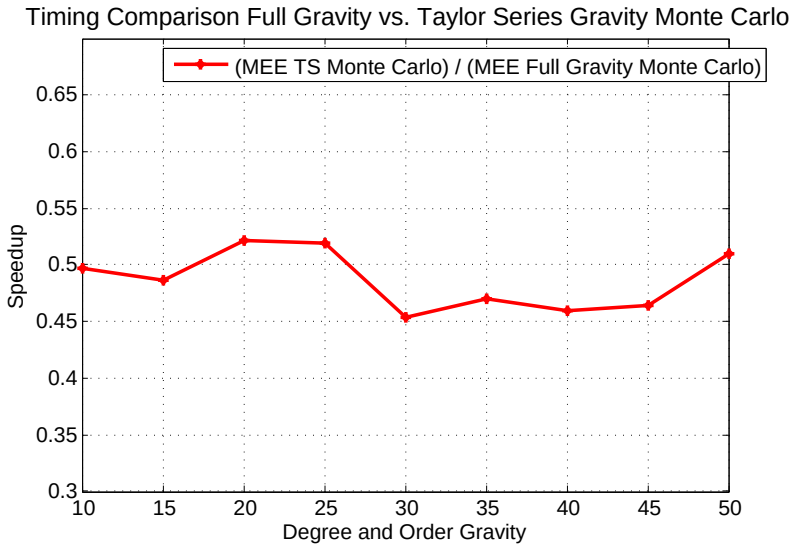


Figure 5.10: Timing Comparison Over 1 LEO Orbit Using Taylor Series Gravity vs. Spherical Harmonic Gravity [51]

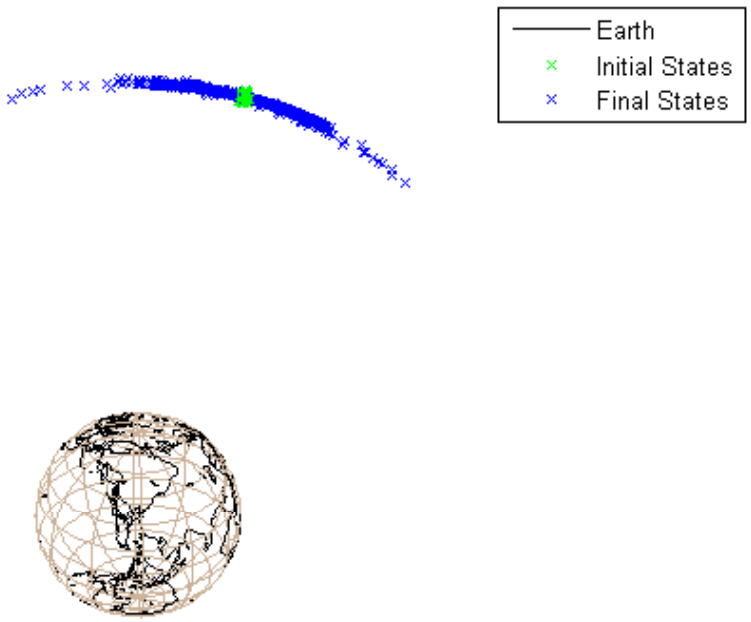


Figure 5.11: Monte Carlo Using One Million Trajectories in a MEO Orbit [51]

5.3 Chapter Summary

This chapter gives a Monte Carlo analysis using the set of Modified Equinoctial Orbital Elements propagated by MCPI. A local Taylor Series gravity approximation is used to reduce the number of full gravity computations required and to thereby reduce the computation time. This method is justified because the terminal iterations are in the close neighborhood of the final converged solution. A low-fidelity gravity and associated Jacobian is used to compute the Taylor series gravity. The Jacobian formulation for the spherical harmonic, zonal, and two-body cases are given in Chapter 4 with additional details in Appendix B. Because the acceleration is computed in Cartesian coordinates for MEE IVP propagation, a transformation from MEEs to Cartesian is required; the corresponding Jacobian may then be easily computed for use in the Taylor Series expansion without any additional nonlinear transformations.

Results are given for LEO and MEO orbits in serial and also for a MEO orbit on a compute cluster. The Taylor series model is shown to be more efficient for the MEE propagation than for the ECI propagation in a serial environment, and the cluster simulation shows a modest reduction in computation time for the MEE propagation vs. the ECI propagation. Further optimization to the MEE propagation in a parallel environment, as well as the inclusion of the second Taylor Series term requiring the Jacobian, will likely show an additional decrease in computation time.

6. METHOD OF PARTICULAR SOLUTIONS

This chapter describes a shooting method that avoids computation or approximation of the state transition matrix, which requires the tedious task of finding explicit partial derivatives. This Method of Particular Solutions (MPS) may be used to solve problems such as Lambert's Two-Point Boundary Value Problem or for steering angles using a direct optimal control method, as described in Chapter 7. MPS is a general solver, and perturbations such as spherical harmonic gravity may be considered. This method was developed by Miele and Iyer [41, 42].

This method will be described using Lambert's problem in Cartesian coordinates (which are easier to visualize than the set of MEEs). Lambert's problem is the two-point boundary value problem of the two-body problem in celestial mechanics. The initial and final positions are specified, while the initial velocity is iteratively solved until the target error (or *miss distance*) is sufficiently small. Lambert's problem is sensitive to the guess for the intial velocity, but a good initial guess may be generated using the p-Iteration method.

For more information about the Method of Particular Solutions for the perturbed Lambert problem using MCPI, see [65, 66, 67]. The Cartesian results presented in these publications are not included here since the focus of this dissertation is the set of Modified Equinoctial Orbital Elements, but an overview will be given with Cartesian coordinates for visualization purposes.

6.1 p-Iteration

For the two-impulse Lambert problem, a good initial guess for the $\Delta\mathbf{v}$ may be generated using p-Iteration for a two-body gravity field [54]. This method does not allow perturbations to be included, but is sufficiently close to the perturbed solution

to allow Lambert's problem to be solved using a shooting method or a shooting method such as MPS as described later in this chapter. It is important to note that p-Iteration encounters a singularity for $\Delta f = 180^\circ$ scenarios.

The initial and final position vectors are specified as $\mathbf{r}_1$ and $\mathbf{r}_2$, as well as the desired flight time Δt. From these position vectors, the true anomaly difference $\Delta\nu$ is computed, as well as the chord length $\mathbf{c} = \mathbf{r}_2 - \mathbf{r}_1$. The initial velocity $\dot{\mathbf{r}}_1$ that allows the spacecraft to arrive at the target state $\mathbf{r}_2$ after a flight time of Δt must be computed. For a regular shooting method, an initial velocity guess would be provided, the orbit would be propagated for the desired flight time Δt, and the target error (miss distance) would be determined. Then, the initial velocity guess would be updated, and the numerical iteration would be repeated until convergence. p-Iteration, however, always determines a trajectory that will reach the target state; the flight time must also be evaluated to provide an iterative correction to the p guess until the desired flight time is achieved. According to Lambert's Theorem, for specified $\mathbf{r}_1$, $\mathbf{r}_2$, and $\left(t_2 - t_1 < \frac{2\pi a^{\frac{3}{2}}}{\sqrt{\mu}}\right)$, the orbit is unique unless $|\mathbf{r}_1 \times \mathbf{r}_2| \to 0$, in which case the plane is undefined.

The initial departure velocity is expressed in terms of $r_1, r_2, \mathbf{r}_1, \mathbf{r}_2, \Delta f$, and p using the following expression [54]:

$$\dot{\mathbf{r}}_1 = \frac{\sqrt{\mu p}}{r_1 r_2 \sin\Delta f}\left[\mathbf{c} + \frac{r_2}{p}(1 - \cos\Delta f)\mathbf{r}_1\right], \quad \mathbf{c} = \mathbf{r}_2 - \mathbf{r}_1 \tag{6.1}$$

The only unknown in this equation is p, the semilatus rectum; thus, the problem becomes that of a one-dimensional search across a single parameter that must be numerically determined to yield the correct time of flight.

First, a value is guessed for the initial velocity. Not all values of p will yield real elliptic trajectories that connect the initial and final position vectors. For instance,

any guess that gives less than a critical speed will not allow the spacecraft to reach its target. To provide a realistic guess for the initial value of p, first consider the minimum-energy orbit that connects the two points $\mathbf{r}_1$ and $\mathbf{r}_2$. These orbits are possible only if

$$a > a_m = \frac{r_1 + r_2 + c}{4}, \quad c = |\mathbf{r}_2 - \mathbf{r}_1| \tag{6.2}$$

Once a value for the minimum a is determined, this value is mapped into the corresponding value of p that leads to a feasible orbit using Eq. (6.25), which will soon be discussed. Now that an initial guess for p is specified, the corresponding flight time from this guess may be determined. Given the position vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, the radii r_i are

$$r_i = \sqrt{\mathbf{r}_i \cdot \mathbf{r}_i} \tag{6.3}$$

Then, solve for the true anomaly change between the departure and arrival states:

$$\Delta f = \cos^{-1} \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} \tag{6.4}$$

Note that the above equation is for a short transfer ($\Delta f < 180^\circ$), and if the transfer is longer than this the following equation is instead applied:

$$\Delta f = 2\pi - \cos^{-1} \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2} \tag{6.5}$$

Each $(p, \Delta f)$ provides a feasible solution but with different Δv orbit corrections. Given this $(p, \Delta f)$, the analytical F and G solution to the two-body problem may be used to determine the required F, G, and $\dot{F}$ values to solve for the semimajor axis. The analytical equations used are

$$F = 1 - \frac{r_2}{p}(1 - \cos \Delta f) \tag{6.6}$$

$$\dot{F} = \sqrt{\frac{\mu}{p}} \tan \frac{\Delta f}{2} \left(\frac{1 - \cos \Delta f}{p} - \frac{1}{r_1} - \frac{1}{r_2} \right) \tag{6.7}$$

$$G = \frac{r_1 r_2}{h} \sin \Delta f = \frac{r_1 r_2}{\sqrt{\mu p}} \sin \Delta f \tag{6.8}$$

$$\dot{G} = 1 - \frac{r_1}{p}(1 - \cos \Delta f) \tag{6.9}$$

These expressions may be written in terms of the corresponding change in eccentric anomaly ΔE:

$$F = 1 - \frac{a}{r_1}(1 - \cos \Delta E) \tag{6.10}$$

$$\dot{F} = -\frac{\sqrt{\mu a}}{r_1 r_2} \sin \Delta E \tag{6.11}$$

$$G = (t - t_0) + \sqrt{\frac{a^3}{\mu}}(\sin \Delta E - \Delta E) \tag{6.12}$$

$$\dot{G} = 1 - \frac{a}{r_2}(1 - \cos \Delta E) \tag{6.13}$$

At this point, seven variables are being considered: $r_1, r_2, \Delta f$, and t are known in the three equations 6.11 - 6.13; p, a, and ΔE are unknown. Using the orbit energy equation, a closed-form relationship is found between a and the two orbit position vectors (where $\mathbf{v}_1 = \dot{\mathbf{r}}_1$):

$$\frac{v_1^2}{2} - \frac{\mu}{r_1} = -\frac{\mu}{2a} \tag{6.14}$$

Now, set the expressions for $\dot{F}$ equal to each other, i.e., Eq. (6.7) = Eq. (6.11). First, use the identity

$$\tan\frac{\Delta f}{2} = \frac{1-\cos\Delta f}{\sin\Delta f} \tag{6.15}$$

Then equate the two $\dot{F}$ expressions to obtain

$$\frac{1-\cos\Delta f}{\sqrt{p}\sin\Delta f}\left(\frac{1-\cos\Delta f}{p} - \frac{1}{r_1} - \frac{1}{r_2}\right) = -\frac{\sqrt{a}\sin\Delta E}{r_1 r_2} \tag{6.16}$$

Next, from setting the expressions for F equal to each other, i.e., Eq. (6.10) = Eq. (6.6),

$$1 - \frac{r_2}{p}(1-\cos\Delta f) = 1 - \frac{a}{r_1}(1-\cos\Delta E) \tag{6.17}$$

a can be solved for:

$$a = \frac{r_1 r_2}{p}\frac{1-\cos\Delta f}{1-\cos\Delta E} \tag{6.18}$$

Substituting this expression for a into Eq. (6.16) yields

$$\frac{1-\cos\Delta f}{\sqrt{p}\sin\Delta f}\left(\frac{1-\cos\Delta f}{p} - \frac{1}{r_1} - \frac{1}{r_2}\right) = -\sqrt{\frac{r_1 r_2(1-\cos\Delta f)}{p(1-\cos\Delta E)}}\frac{\sin\Delta E}{r_1 r_2} \tag{6.19}$$

After cancelling common factors and solving for the semilatus rectum, an elegant expression is derived for p:

$$p = \frac{r_1 r_2 (1 - \cos \Delta f)}{r_1 + r_2 - 2\sqrt{r_1 r_2 \cos \frac{\Delta f}{2} \cos \frac{\Delta E}{2}}} \tag{6.20}$$

Next, define these three constants in terms of the known quantities $(r_1, r_2, \Delta f)$:

$$k = r_1 r_2 (1 - \cos \Delta f) \tag{6.21}$$

$$l = r_1 + r_2 \tag{6.22}$$

$$m = r_1 r_2 (1 + \cos \Delta f) \tag{6.23}$$

Using these definitions, a may be written compactly as (see [54] for derivation)

$$a(p) = \frac{mkp}{(2m - l^2)p^2 + 2klp - k^2} \tag{6.24}$$

Notice that (k, l, m) of Eqs. (6.21) - (6.23) are all functions of known constants, and therefore Eq. (6.24) gives $a(p)$. Mathematically, the singularity at $\Delta f = 180^\circ$ occurs when $m = 1$ and therefore $a(p) = 0$. The inverse mapping of Eq. (6.24) is needed to initiate the p-Iteration algorithm; it is only used to determine the initial guess of p [54]:

$$p(a) = \frac{2akl - km + k\sqrt{m(8a^2 - 4al + m)}}{2(al^2 - 2am)} \tag{6.25}$$

Now that the values for a and p are consistent, the eccentric anomaly change may be computed that in turn gives an expression for the time of flight. Let

$$B = \tan^{-1} \frac{r_0 \tan \frac{\Delta f}{2}}{\sqrt{ap} - \sigma_0 \sqrt{a} \tan \frac{\Delta f}{2}} \tag{6.26}$$

Then, if:

$$\begin{cases} 0 \leq \Delta f < \pi & \text{then } \Delta E = 2B \\ & \text{else } \Delta E = 2(\pi - B) \end{cases} \tag{6.27}$$

Note that the quadrants of $\frac{\Delta f}{2}$ and $\frac{\Delta E}{2}$ are always the same. The time of flight is next found that corresponds to the current guess for p. Solving the following equation (G from the analytical two-body solution) for the flight time

$$G = (t - t_0) + \sqrt{\frac{a^3}{\mu}} (\sin \Delta E - \Delta E) \tag{6.28}$$

results in the expression

$$\Delta t = \Delta t(p) = G + \sqrt{\frac{a^3}{\mu}} (\Delta E - \sin \Delta E) \tag{6.29}$$

Typically, this computed time of flight, based on the current p, is not equal to the desired time, so the value of p must be iteratively updated to find the desired time of flight. To establish a Newton root solving algorithm for p, define a function

$$g(p) = \Delta t(p) - \Delta t_{desired} \tag{6.30}$$

When a feasible p is found, then $g(p) = 0$. Since no closed-form solution exists to this problem, the updated p value is solved numerically. Newton's method is chosen for this iteration, which requires the derivative $\frac{dg}{dp}$ that can be approximated using the secant method:

$$\frac{dg}{dp} = \frac{g - g_{old}}{p - p_{old}}; \tag{6.31}$$

so that the p values is udpated using

$$p_{new} = p - \frac{g}{\frac{dg}{dp}}; \tag{6.32}$$

The flight time is solved for iteratively from Eq. 6.29 to see if $g(p)$ is sufficiently small, and the process is repeated until convergence. For short (less than one revolution) transfers, this algorithm has been found to typically converge in 3 to 5 iterations. The converged initial velocity is then used as an input guess to the MPS method, which can accommodate perturbations and is described in the next section.

For long, multi-revolution orbit transfers, the story is complicated because multiple roots for p exist. Starting estimates for p-roots can be found by plotting Eq. 6.30 with dense p values to find all zero crossings. The approximate p value near each zero crossing is the starting estimate.

6.2 Method of Particular Solutions

The Method of Particular Solutions (MPS) is a shooting method that is used to solve Two-Point Boundary Value Problems. Though it is a general solver, if the perturbed Lambert's problem is considered, a good initial guess may be found using the p-Iteration method as described in the previous section.

The method of particular solutions makes use of a reference trajectory, and all neighboring solutions can be re-formulated exactly in terms of a departure motion. Refer to Figure (6.1) for the following development. This shooting method is somewhat analogous to Newton's method, but without requiring partial derivatives (i.e., from computing a state transition matrix). For a general gravity, drag, and force

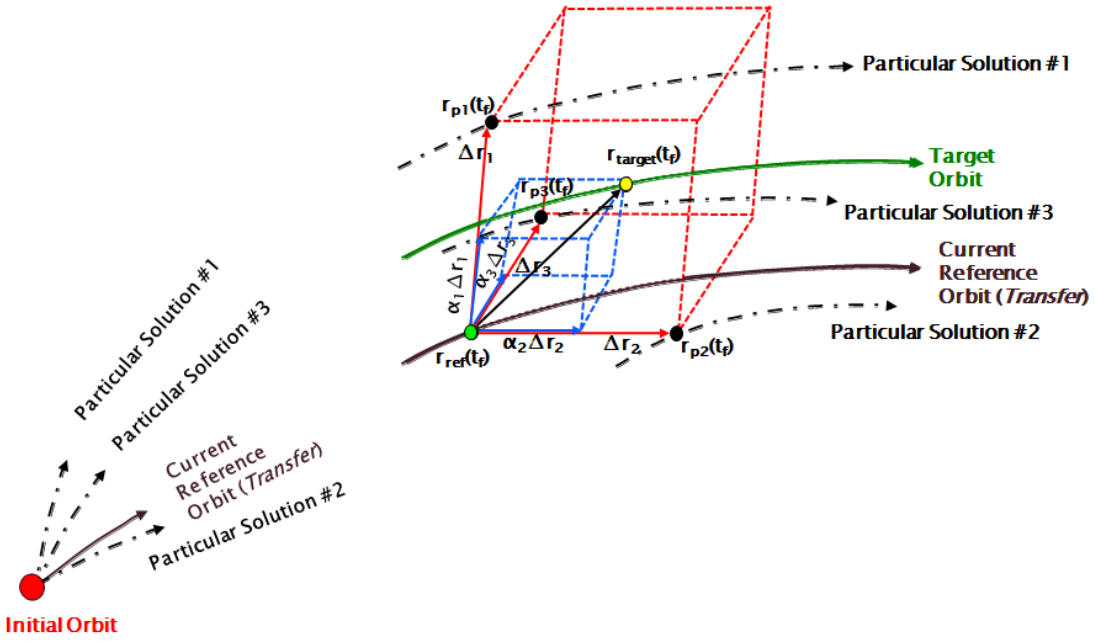


Figure 6.1: Overview of the Method of Particular Solutions Method

model, and for a non-affine control u,

$$\ddot{\mathbf{r}} = \mathbf{g}(t, \mathbf{r}, \dot{\mathbf{r}}, \mathbf{u}) \tag{6.33}$$

The method of particular solutions makes use of a reference trajectory $\mathbf{r}_{ref}(t)$, $\dot{\mathbf{r}}_{ref}(t)$, $\ddot{\mathbf{r}}_{ref}(t)$, and all neighboring solutions can be reformulated exactly in terms of a departure motion $\Delta\mathbf{r}(t)$ as

$$\mathbf{r}(t) = \mathbf{r}_{ref}(t) + \Delta\mathbf{r}, \quad \dot{\mathbf{r}}(t) = \dot{\mathbf{r}}_{ref}(t) + \Delta\dot{\mathbf{r}}, \quad \ddot{\mathbf{r}}(t) = \ddot{\mathbf{r}}_{ref}(t) + \Delta\ddot{\mathbf{r}} \tag{6.34}$$

Using these relationships, Equation (6.33) may now be written in terms of the exact departure motion:

$$\ddot{\mathbf{r}} = \mathbf{g}\big(t, \mathbf{r}_{ref}(t) + \Delta\mathbf{r}(t), \dot{\mathbf{r}}_{ref}(t) + \Delta\dot{\mathbf{r}}(t), \ddot{\mathbf{r}}_{ref}(t) + \Delta\ddot{\mathbf{r}}(t), \mathbf{u}(t)\big) \tag{6.35}$$

Consider three trajectories neighboring the reference trajectory, where the initial velocity is varied by small linearly independent perturbations. The neighboring initial velocities are then

$$\dot{\mathbf{r}}_j(t_0) = \dot{\mathbf{r}}_{ref}(t_0) + \Delta\dot{\mathbf{r}}_j(t_0); \quad j = 1, 2, 3 \tag{6.36}$$

The exact departure motions (particular solutions) are then given by

$$\Delta\mathbf{r}_j(t) = \mathbf{r}_j(t) - \mathbf{r}_{ref}(t) \tag{6.37}$$

Since independent velocity initial conditions were used, these trajectories are assumed to span the space of interest and all neighboring trajectories of interest that also approximately satisfy the linear departure motion dynamics.

The linear combination of particular solutions of a linear differential equation satisfies the differential equation as well, and with appropriate choice of $(\alpha_1, \alpha_2, \alpha_3)$, any general solutions (for example, with initial position fixed, all three velocity components are available for variation) can be written as a linear combination of the three (in general n) departure motions in the form:

$$\Delta\mathbf{r}(t) \approx \sum_{j=1}^{3} \alpha_j \Delta\mathbf{r}_j(t) \tag{6.38}$$

and the time-dependent position may be approximated as

$$\mathbf{r}(t) \approx \mathbf{r}_{ref}(t) + \sum_{j=1}^{3} \alpha_j \Delta\mathbf{r}_j(t) \tag{6.39}$$

Evaluating Equation (6.39) at the final time and imposing the desired result that $\mathbf{r}(t_f) = \mathbf{r}_f$, leads to the solution for the coefficients of linear combination

$$\begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{Bmatrix} \approx [\Delta\mathbf{r}_1(t_f) \quad \Delta\mathbf{r}_2(t_f) \quad \Delta\mathbf{r}_3(t_f)]^{-1}[\mathbf{r}_f - \mathbf{r}_{ref}(t_f)] \tag{6.40}$$

Taking the time derivative of Equation (6.39) and imposing the desired result that $\mathbf{r}(t_f) = \mathbf{r}_f$, leads to a new estimate for the initial velocity to be calculated:

$$\dot{\mathbf{r}}_{new}(t_0) = \dot{\mathbf{r}}_{ref}(t_0) + \sum_{j=1}^{3} \alpha_j \Delta\dot{\mathbf{r}}_j(t_0) \tag{6.41}$$

The values of the α_j's are improved using Eq. (6.40) iteratively, where each of the particular solutions may be computed simultaneously using parallel processing if desired.

6.3 Chapter Summary

This chapter describes a shooting method that may be used to solve boundary value problems. For Lambert's problem, the p-iteration method may be used to get an initial guess for the Δv required to reach a target state. Though p-iteration doesn't allow perturbations to be considered, the initial guess it provides is almost always close enough to the desired value for the Method of Particular Solutions to converge. Also, in almost all cases, the Keplerian Lambert solution for the multi-revolution cases are found to approximate the multiple perturbed orbit transfer solutions sufficiently well to start the MPS iteration and converge in 3 or 4 iterations. MPS uses three particular solutions to update the transfer trajectory guess for this generally perturbed boundary value problem. The MPS method described in this chapter is expanded for the suboptimal and optimal control described in the next two sections.

7. SUBOPTIMAL CONTROL USING STEERING ANGLES

A minimum-time, low-thrust orbit transfer is sought that takes multiple revolutions to reach a target state. Since the optimal control formulation discussed in the next chapter is very sensitive to estimates of the initial costate variables, a nonlinear programming problem (NLP) is first solved to provide a suboptimal solution prior to implementing the full state/co-state equations. The NLP formulation is based on a control parameterization and corresponding trajectory approximation that assumes a flight control parameterization structure in advance. A minimum-correction-norm, gradient-based search method is chosen to solve this NLP (gradient-based adjustment of the control parameters to satisfy the constraints and find a feasible solution with near optimal performance).

Although using a direct approach is not guaranteeed to give the optimal solution, it is often accurate enough for practical applications. For instance, Bahls and Paris [3] used a direct approach with trajectory segmentation to develop the Gravity Assisted Low Thrust Optimization Program (GALTOP). In this software tool, individual thrust arcs are computed using Chebyshev collocation and these curves are patched together to give the suboptimal trajectory approximation. In their work, this direct method produced relatively accurate results with a speedup of 30% when compared with other then-state-of-the-art direct method optimizers such as the Heliocentric Interplanetary Low Thrust Optimization Program (HILTOP). For this research, a NLP approach is adopted both for the possibility to be used in missions and for those cases for which rigorous satisfaction of the Pontryagin optimal control necessary conditions is ultimately sought (these suboptimal trajectories can be used as a starting iterative).

7.1 Problem Statement

Given the equations of motion (see Chapter 8 for further details).

$$\frac{d\mathbf{e}}{dt} = \mathbf{M}\frac{T}{m}\mathbf{u} + \mathbf{D} \tag{7.1}$$

Here, $\mathbf{M} = \mathbf{M}(\mathbf{e})$ is a 6x3 matrix that represents the thrust influence coefficient matrix given by Gao and Kluever [23], $T = (g\dot{m})I_{sp}$ is the thrust magnitude, and $m = m(t) = m_0 - m(t - t_0)$ is the spacecraft mass. The time rate of change of $\mathbf{e}$ when perturbing forces are not present is $\mathbf{D}$= $[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \sqrt{\mu p}\left(\frac{w}{p}\right)^2]^T$, where $w = \frac{p}{r} = 1 + fcos(L) + gsin(L)$. If $\dot{m}$ is small, the final solution can be started by using a constant $\frac{T}{m}$ solution. Consider a suboptimal control using steering angles pitch (α) and yaw (δ):

$$\mathbf{u} = -\begin{bmatrix} \cos\delta(t)\cos\alpha(t) & \cos\delta(t)\sin\alpha(t) & \sin\delta(t) \end{bmatrix}^T \tag{7.2}$$

Note that the pitch angle is frequently a slowly varying function, except near switches. For many cases, it is useful computationally to assume that variations in $\alpha(t)$ are locally approximated such that $\Delta\alpha \approx$ linear function of variations in $\alpha(t), \delta(t)$. However, variations in $\alpha(t)$, $\delta(t)$ are not generally small.

Instead, Chebyshev polynomials are chosen as the basis functions for the parameterization; see Chapter 2, Section 2.3 and Appendix A for more details:

$$\alpha(t) = \alpha_0 + \alpha_1 T_1(t) + \alpha_2 T_2(t) + \cdots + \alpha_n T_n(t) \tag{7.3}$$

$$\delta(t) = \delta_0 + \delta_1 T_1(t) + \delta_2 T_2(t) + \cdots + \delta_n T_n(t) \tag{7.4}$$

Therefore, the steering angles can be represented compactly as series of polynomials:

$$\alpha(t) = \sum_{k=0}^{l_1} \alpha_k T_k(\tau) \tag{7.5}$$

$$\delta(t) = \sum_{k=0}^{l_2} \delta_k T_k(\tau) \tag{7.6}$$

The α_k, δ_k coefficients can be taken as unknowns for an open loop trajectory via NLP. Assuming $l_1 = l_2 = n$, group the unknown parameters in a vector (this method does not require $l_1 = l_2$):

$$\mathbf{P} = \begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \dots \alpha_n & | & \delta_0 & \delta_1 & \delta_2 & \dots \delta_n \end{bmatrix}^T \tag{7.7}$$

This algorithm may be solved using an extension of the Method of Particular Solution to establish a NLP algorithm, which is a shooting method discussed in Chapter 6, to iteratively update the parameter vector. Note that, if necessary, n can be restricted to a small number in order to get the macroscopic shape of the control correct and to increase dimensionality. This suboptimal control solution may be approximated as a minimum time problem by slowly decreasing the flight time until the shortest time over which the NLP converges, to heuristically give the best possible minimum time solution by this NLP algorithm.

7.2 Suboptimal Control Simulation Results

In order to generate a feasible trajectory reachable with a multi-rev, low-thrust MEE formulation, several steps are taken.

First, the IVP is used to generate a reachable trajectory over several orbits using specified initial conditions and control vector. Next, in order to find a guess for the flight time, the IVP is used again to propagate the ICs *without* using any knowledge

of the previously generated trajectory, to simulate a real-world application. When the value of the semilatus rectum, p, exceeds the targeted value of p (or alternatively, a may be used where $a_{target} = \frac{p}{1-e^2}$ where e is found from the prescribed MEEs), the corresponding flight time is increased to establish a conservative flight time starting guess (i.e., multiply this value by 1.5). This heuristic control simply reaches a target semimajor axis without satisfying the remaining constraints. "Gravity turn" control may be used as the first estimate of the thrust along the velocity vector. Specifically, if $a_{desired} > a_{initial}$, the thrust should be $\mathbf{a}_d = +\frac{T}{m}\frac{\dot{\mathbf{r}}}{|\dot{\mathbf{r}}|}$ and if $a_{desired} < a_{initial}$ then $\mathbf{a}_d = -\frac{T}{m}\frac{\dot{\mathbf{r}}}{|\dot{\mathbf{r}}|}$. This is because a thrust along the velocity vector will increase the semimajor axis, while a thrust opposite to the velocity vector will deacrease the semimajor axis as seen in the $\frac{da}{dt}$ Gauss' Variational Equation [54]

$$\frac{da}{dt} = \frac{2a^2}{h}\Big(e\sin\nu a_r + \frac{p}{r}a_\theta\Big) \tag{7.8}$$

where a is the semimajor axis, h is the magnitude of the angular momentum vector, e is the eccentricity, ν is the true anomaly, p is the semilatus rectum, r is the radius, a_r is the acceleration component in the radial direction, and a_θ is the acceleration component in the orthogonal direction.

The steering angles are represented using an arbitrary number of coefficients in the polynomial series as given earlier in this chapter. The Method of Particular Solutions (MPS) as described in Chapter 6 is used to compute the miss distance of the final MEE state relative to the target state, and to accordingly update the steering angle coefficients until convergence is achieved.

7.3 Simulation Results

7.3.1 Example 1

The initial conditions for a Low-Earth Orbit (LEO) case are represented in Classical Orbital Elements (COEs) and converted to MEEs, since the COEs are easier to visualize. The BCs for the following plots are given in Tables 7.1 and 7.2 using Cartesian coordinates, which are converted to canonical units for integration (see Appendix D for more details).

Table 7.1: Example 1 LEO Trajectory Initial Conditions

a_0	e_0	i_0	Ω_0	ω_0	M_0
$8000km$	0.1	5°	0	0	0

Table 7.2: Example 1 LEO Trajectory Target State

a_f	e_f	i_f	Ω_f	ω_f	M_f
$8200km$	0.1	5.3°	0	0	0

where ν is used in this discussion to denote the true anomaly because f is used for one of the MEE elements. Figure 7.1 shows that about 2.5 orbits are needed to reach the desired final state. Also, a spherical harmonic gravity of degree and order 10 is included in addition to the thrust.

Figure 7.2 shows how efficiently MPS converges. Over 10 orders of magnitude reduction in final constraint residual norm is achieved in 5 iterations. Figure 7.3 shows the three thrust vector components (in the instantaneous radial, normal, and

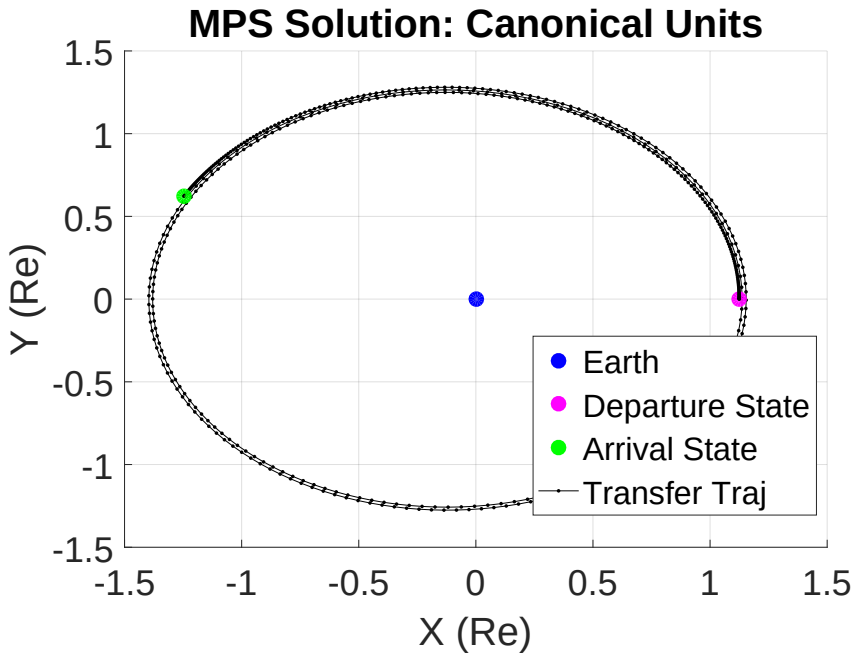


Figure 7.1: Trajectory Solution Using Suboptimal Control Formulation with MEEs and MCPI for Example 1

transverse directions). Figure 7.4 shows the corresponding steering angles of Eqs. (7.3) and (7.4) with $n = 10$ so that 20 total parameters are solved for. Note an $\alpha(t)$ of 90° would correspond to thrusting along the velocity vector, while $\delta(t)$ of zero would correspond to the thrust vector lying in the osculating orbit plane. Quantitatively, the orbit transfer increases the orbit semimajor axis from $8000km$ to $8200km$, while increasing the inclination by 0.3° and the eccentricity from 0.1 to 0.11. Takeoff occurs at perigee. The orbit transfer is depicted in 2D Figure 7.5.

7.3.2 Example 2

The COEs are used again to prescribe a second LEO example. The departure and target elements for this case are given in Tables 7.3 and 7.4. Again, takeoff occurs at perigee.

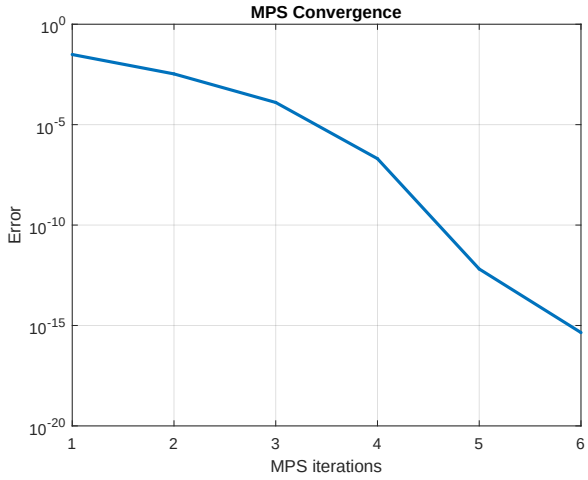


Figure 7.2: MPS Convergence as a Function of MCPI Iterations Using Suboptimal Control Formulation with MEEs and MCPI for Example 1

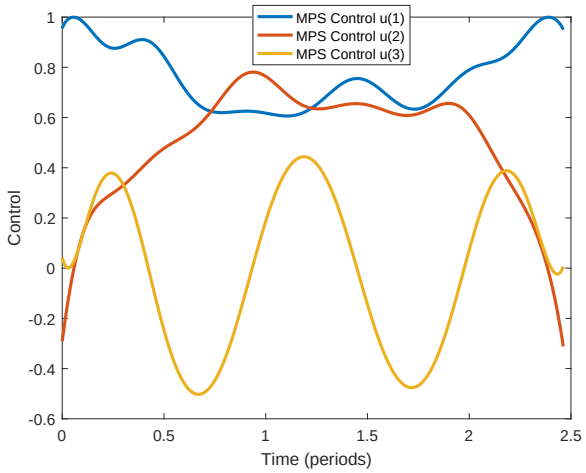


Figure 7.3: Control Vector Using Suboptimal Control Formulation with MEEs and MCPI for Example 1

The result of this simulation is shown in Figures 7.5 - 7.9. The orbit transfer is depicted in 2D and 3D in Figures 7.5 and 7.6. Once again, MPS converges rapidly (Figure 7.7) and this time, the suboptimal controls oscillate more than for Example

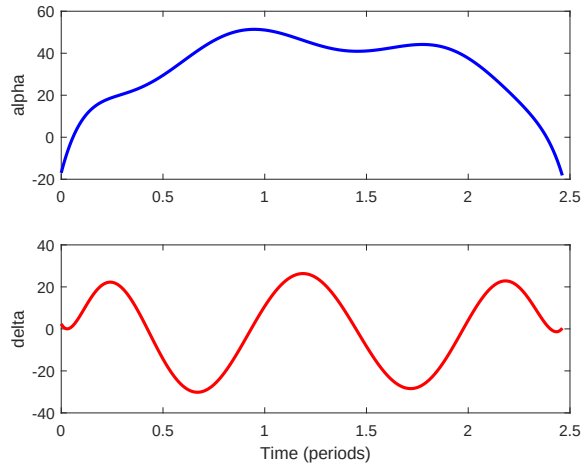


Figure 7.4: Steering Angles Using Suboptimal Control Formulation with MEEs and MCPI for Example 1

Table 7.3: Example 2 LEO Trajectory Initial Conditions

a_0	e_0	i_0	Ω_0	ω_0	M_0
$8000km$	0.1	0	0	0	0

Table 7.4: Example 2 LEO Trajectory Target State

a_f	e_f	i_f	Ω_f	ω_f	M_f
$8500km$	0.11	0.2°	0	0	0

1, as is evident in Figures 7.8 and 7.9, but this is somewhat deceptive. Note in Figure 7.1 that this maneuver is over 2.5 orbits, whereas Example 2 in Figure 7.5 is over almost 5 orbits. However, the larger change in inclination of Example 2 requires a more complicated transer orbit with more out-of-plane thrusting, as is clear in Figure 7.6.

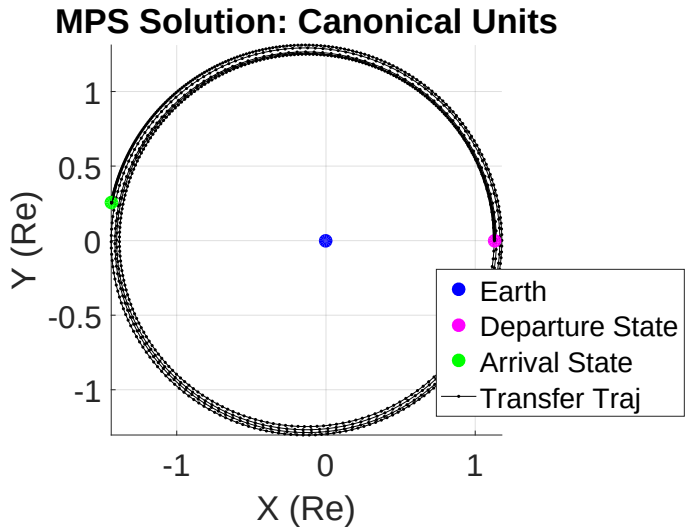


Figure 7.5: 2-D Projection of Trajectory Solution Using Suboptimal Control Formulation with MEEs and MCPI for Example 2

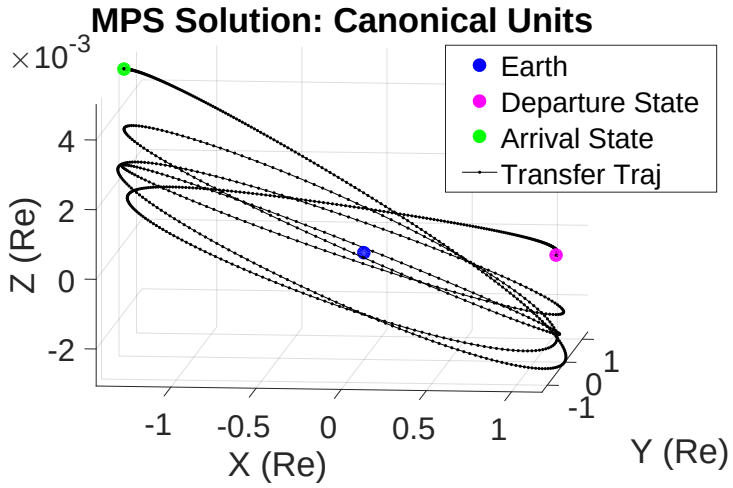


Figure 7.6: Trajectory Solution Using Suboptimal Control Formulation with MEEs and MCPI for Example 2

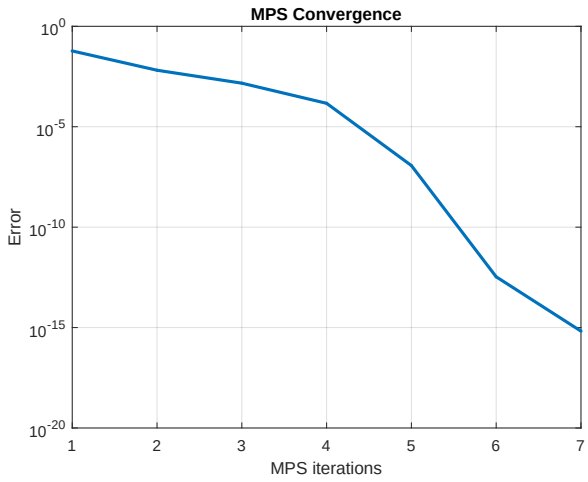


Figure 7.7: MPS Convergence as a Function of MCPI Iterations Using Suboptimal Control Formulation with MEEs and MCPI for Example 2

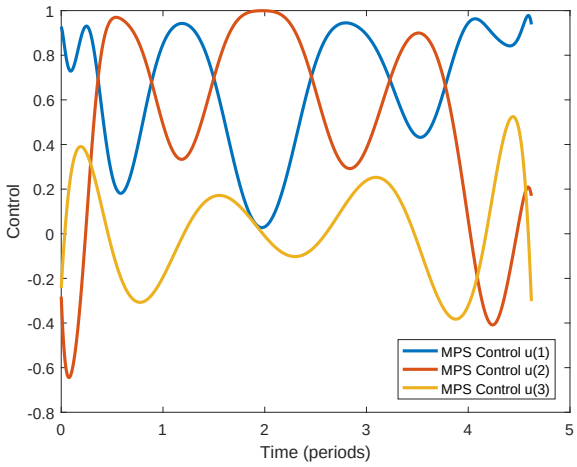


Figure 7.8: Control Vector Using Suboptimal Control Formulation with MEEs and MCPI for Example 2

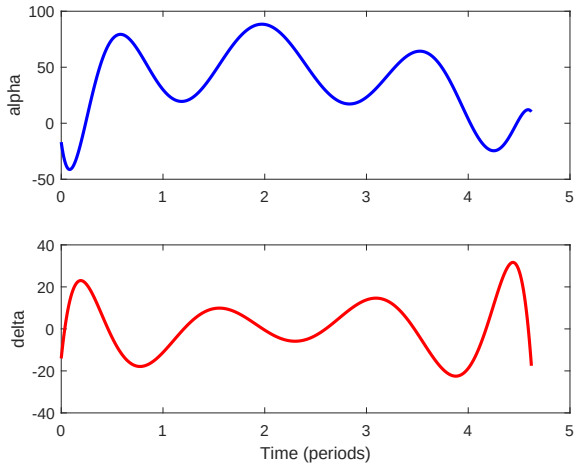


Figure 7.9: Steering Angles Using Suboptimal Control Formulation with MEEs and MCPI for Example 2

7.4 Chapter Summary

This chapter discusses a suboptimal control formulation that uses MCPI to propagate the MEEs and reach a desired target using a low-thrust control and MPS. By finding the desired flight time and steering angles necessary, and reducing the flight time gradually until the solution does not converge, an approximation of the minimum-time optimal control is found that may be used as the initial guess for the optimal control formulation given in Chapter 8. The challenge of the optimal control problem is finding appropriate guesses for the initial costates.

The results given above show that the minimum norm adjustment leads quickly to feasible solutions. In both cases, the assigned final time can be reduced until MPS can no longer isolate the final boundary conditions, indicating at least local infeasibility. This final convergence is accepted as the suboptimal (near minimum time) continuous transfer to indicate an indirect optimal approach with a good starting estimate.

8. LOW-THRUST OPTIMAL CONTROL

Low-thrust optimal control problems are often more challenging to solve than high-thrust systems, partly because low-thrust propulsion operates over larger times for a large portion of the mission duration rather than isolated events. Obviously, small local control variations over short time intervals have small effects on the final state, so there is an inherent low sensitivity that can manifest itself as a poorly conditioned state/costate system of differential equations. Therefore, the control variables, if modeled as continuous functions and continuous optimization leads to a function-space optimal control problem (an indirect method, solved using variational calculus), which is frequently difficult to solve [30]. Of course, heuristic parameterizations may be introduced to discretize the unknown control to a finite dimensional approximation (direct method), such as in Chapter 7. In fact, this is frequently the first step in the current state of practice and is the approach pursued here.

The first step in pursuing the indirect (calculus of variations) approach is to establish the optimal control necessary conditions for the problem at hand. The present work presents an indirect method to solve an optimal continuous thrust problem using a set of Modified Equinoctial Orbital Elements (MEE) in place of Cartesian coordinates, which has shown to have convergent Picard iterations over a much larger number of orbits using MCPI. As shown in Chapter 3, the IVP MCPI convergence domain is increased from around 3 orbits to 17 orbits, for the full spherical harmonic (40x40) gravity model case.

Since the optimal control formulation is known to be very sensitive to the a priori unknown initial costate conditions, first a heuristic control approximation is introduced to solve a nonlinear programming problem (NLP) that provides a sub-

optimal solution prior to implementing the full state/co-state equations. As noted by Gao and Kluever [23], using a NLP allows a solution that assumes a flight control parameterization in advance. However, a judicious parameterization can be used that allows adaptive refinement of the control parameter model. This formulation is given in Chapter 7.

8.1 Problem Statement

The flight time of the maneuver is to be minimized, and the final time is specified to be free. In this way, t_f may be regarded as a control parameter to be chosen in addition to the control functions to minimize the performance index and satisfy the constraints. The cost function then is simply

$$J = \int_{t_0}^{t_f} 1dt = t_f - t_0 \tag{8.1}$$

The equations of motion under disturbing forces may be written generally as

$$\frac{d\mathbf{e}}{dt} = \mathbf{B}(\mathbf{P} + \mathbf{G}) + \mathbf{D} \tag{8.2}$$

where $\mathbf{e} = [p\ f\ g\ h\ k\ L]^T$ is the MEE set introduced in Chapter 3, $\mathbf{B} = \mathbf{B}(e)$ is the coefficient matrix acting on non-two-body accelerations for the Varation of Parameters (for the MEE case) given above, $\mathbf{G} = \mathbf{G}(\mathbf{e})$ is the gravitational acceleration, and $\mathbf{P}$ is the thrust acceleration. These accelerations are expressed with components in the radial, transverse, and orbit normal in the rotating frame. The time rate of change of $\mathbf{e}$ when perturbing forces are not present is $\mathbf{D}$= $[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \sqrt{\mu p}\left(\frac{w}{p}\right)^2]^T$, where $w = \frac{p}{r} = 1 + fcos(L) + gsin(L)$. When considering the low-thrust only case (with relative gravity perturbations), this equation may be written specifically as

$$\frac{d\mathbf{e}}{dt} = \mathbf{M}\frac{T}{m}\mathbf{u} + \mathbf{D} \tag{8.3}$$

Here, $\mathbf{M} = \mathbf{M}(\mathbf{e})$ is a 6x3 matrix that represents the thrust influence coefficient matrix (given by Gao and Kluever [23]), T is the thrust magnitude, and $m = m(t) = m_0 - m(t - t_0)$ is the spacecraft mass. For the present discussion, the mass is assumed to be constant so its corresponding costate equation is not included. The direction unit vector associated with the thrust may be expressed in terms of the local pitch and yaw steering angles. The yaw angle δ is measured about the radial unit vector from the orbit plane to the plane containing the thrust vector, as shown in Figure (8.1).

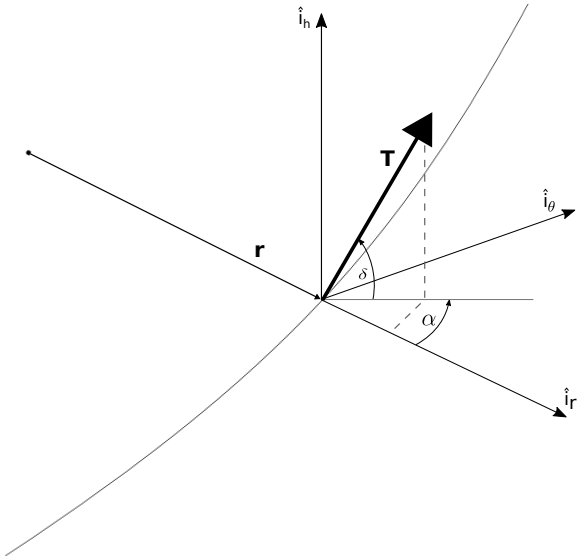


Figure 8.1: Thrust Vector for Low-Thrust Optimal Control

$$\mathbf{T} = T[cos\delta\ cos\alpha\ \hat{i}_r \quad cos\delta\ sin\alpha\ \hat{i}_\theta \quad sin\delta\ \hat{i}_h]^T \tag{8.4}$$

The thrust magnitude is modeled as

$$T = 2\eta P/c \tag{8.5}$$

where P is the input power, η is the thruster efficiency, and $c = gI_{sp}$ is the engine exhaust velocity. For the present study, constant values are used for P, I_{sp}, η.

The Hamiltonian is then formed as

$$H = 1 + \lambda^T \left[\mathbf{M} \frac{T}{m} \mathbf{u} + \mathbf{D} \right] \tag{8.6}$$

Here, $\lambda = [\lambda_p \; \lambda_f \; \lambda_g \; \lambda_h \; \lambda_k \; \lambda_L]$ is the costate vector associated with the MEEs.

To obtain the optimal thrust direction, thrust direction unit vector $\mathbf{u}(t)$ must be obtained to minimize H according to Pontryagin's Principle and enforce the constraint that the admissible set of u_i's must be a smooth vector function that satisfies $\mathbf{u}^T\mathbf{u} = 1$. An admissible $\mathbf{u}(t)$ is sought to make the term $[\lambda^T\mathbf{M}]\mathbf{u}(t)$ as small as possible. This term of the Hamiltonian may be rewritten as

$$\frac{T}{m}\left[\lambda^T\mathbf{M}\right] \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \frac{T}{m}\left[m_1 \; m_2 \; m_3\right] \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \frac{T}{m}\mathbf{m}^T\mathbf{u} \tag{8.7}$$

where $\mathbf{m} \equiv \mathbf{M}^T\lambda$, $\hat{\mathbf{m}} = \frac{\mathbf{m}}{|\mathbf{m}|}$.

Based on Pontryagin's Principle, the optimal control vector $\mathbf{u}(t)$ will then be in the opposite direction of the $\mathbf{m}$ vector to minimize H with respect to $\mathbf{u}$. Although each u_i component may change signs, the unit vector itself will move in three-dimensional space with a continuous motion (MCPI is well-suited to smooth functions). The optimal constrained thrust vector is then written as

$$\mathbf{u}* = -\frac{[\lambda^T \mathbf{M}]^T}{||\hat{\lambda}^T \mathbf{M}||} = -\frac{[\mathbf{M}^T \lambda]}{\sqrt{\lambda^T \mathbf{M}\mathbf{M}^T \lambda}} = -\hat{\mathbf{m}} \tag{8.8}$$

As shown in [26], the λ's do not have a unit magnitude, and the initial λ's can be scaled such that $\left[\lambda \mathbf{M}\mathbf{M}^T \lambda\right]_{t_0} = 1$, to make the initial magnitude unique. So, $\lambda^{\mathbf{T}}\mathbf{M}$ is expected to be of order 1; thereafter, $\lambda^T \mathbf{M}$ will vary, of course. This ad hoc scaling has consequences, however, because as noted below, the free final time boundary condition leads to a generally inconsistent scaling. The costate equations are obtained by taking the partial derivative of the Hamiltonian with respect to the states:

$$\dot{\lambda} = -\frac{\partial H}{\partial \mathbf{e}} = -\left(\lambda^T \frac{\partial \mathbf{M}}{\partial \mathbf{e}} \frac{T}{m} \mathbf{u} + \lambda^T \frac{\partial \mathbf{D}}{\partial \mathbf{e}} \right) \tag{8.9}$$

The partial derivatives of matrices $\mathbf{M}$ and $\mathbf{D}$ with respect to the states are given in the Appendix of Gao and Kluever [23].

For a continuous, nonlinear optimal control problem, the transversality condition is given as [33]

$$(\phi_x + \psi_x^T - \lambda)^T \Big|_T d\mathbf{e}(T) + (\phi_t + \psi_t^T \nu + H) \Big|_T dT = 0 \tag{8.10}$$

If the final state is considered to be fixed with respect to the final (free) time, then $d\mathbf{e}(T) = 0$ and only the second part of the transversality condition equation above must be addressed:

$$(\phi_t + \psi_t^T \nu + H) \Big|_T dT = 0 \tag{8.11}$$

Here $\phi = 0$, and since the final state is fixed, the final state $\mathbf{e}(T)$ is required to lie on a target set, $\mathbf{p}(T)$, such that the following six conditions are met:

$$\psi(T) = \mathbf{e}(T) - \mathbf{p}(T) = 0 \tag{8.12}$$

Since $\phi_t = 0$ and $\psi_T = 0$, from Eq. (8.11) the Hamiltonian should vanish at the optimal final time:

$$H(T) = 0 \tag{8.13}$$

Notice from Eq. (8.6), that the condition (**??**) implicitly imposed a scale factor on $\lambda(t_f)$. If an initial scaling is adopted for $\lambda(t_0)$, the 1 in Eq. (8.6) may need to be replaced by some other constant consistent with the $\lambda(t_0)$ scaling [26].

This optimal control problem cannot immediately be solved by implementing Modified Chebyshev Picard Iteration (MCPI) because the $\left[\lambda^T \mathbf{M}\mathbf{M}^T \lambda\right]$ term causes the boundary conditions to be a nonlinear function of the Chebyshev coefficients. However, in the end game, following a warm start the term $\left[\lambda^T \mathbf{M}\mathbf{M}^T \lambda\right]^{\frac{1}{2}}$ can be considered an approximately known function of time from the immediately preceeding function. Thus, only the numerator of Eq. (8.8) is expanded as a Chebyshev series on each iteration, so that costate boundary conditions can be imposed on the costate Chebyshev coefficients. The denominator function $\left[\lambda^T \mathbf{M}\mathbf{M}^T \lambda\right]^{\frac{1}{2}}$ can be updated after the fact and again approximately considered known on each MCPI iteration.

The time t may be written in terms of a new time variable τ as

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{8.14}$$

Then, taking the partial derivative with respect to τ gives

$$\frac{dt}{d\tau} = \frac{t_f - t_0}{2}, \quad -1 \leq \tau \leq 1 \tag{8.15}$$

Therefore,

$$\frac{d(\cdot)}{d\tau} = \frac{d(\cdot)}{dt}\frac{dt}{d\tau} = \left(\frac{t_f - t_0}{2}\right)\frac{d(\cdot)}{dt} \tag{8.16}$$

Define ϕ as an unknown constant of final time and let

$$J_f = x_{n+1} = t_f = \phi \tag{8.17}$$

then introduce the additional constraint

$$\lambda_{n+1}(t_f) = \frac{\partial \phi}{\partial x_{n+1}} = \frac{\partial x_{n+1}}{\partial x_{n+1}} = 1 \tag{8.18}$$

All other $\lambda_i(t_f)$ are unknown. This formulation has the effect of mapping the $(t_f - t_0)$ interval, with t_f unknown, onto a fixed τ interval $(-1 \leq \tau \leq 1)$ while introducing the free final time as the optimization variable; also, consider t_f an n^{th} state variable with an associated costate final boundary condition. For the final-time free case using MCPI, Equation (8.18) is the replacement constraint that indicates a solution has been found corresponding to the final time.

This method requires the use of the standardized MCPI method, as described in Section 2.5.1, to solve the equations in a cascade fashion. Trying to implement this algorithm using the traditional MCPI method, for instance as given in several PhD dissertations [5, 36, 68] leads to a long, tedious derivation that inevitably requires the use of a symbolic toolbox such as *Mathemtica* or *Maxima*. In contrast, the standardized method allows for a more straightforward method of implementing MCPI with one caveat that the nonlinear λ term is "lagged" one Picard iteration, since its

inclusion makes the problem have a non-standard form. A formal convergence proof is not yet available; however, "encouraging numerical results" are expected.

9. CONCLUSION

The work presented in this dissertation may be used for SSA such as efficient orbital debris propagation and planning corresponding mitigation strategies through identification of potential conjunctions. Real-time tracking efforts continually become computationally burdensome as the number of trackable objects increases, leading to an increased need for efficient computational methods such as parallel processing. The integration method used for this dissertation, MCPI, is inherently, massively parallelizable, therefore addressing this need. One particular area of interest within SSA is optimized orbit transfer maneuvers to reach or de-orbit orbital debris, where long transfer time is a feasible option to reduce the propellant cost and increase the specific impulse. To address this need, a low-thrust orbit transfer is considered to allow for multi-revolution solutions. Both low-thrust propulsion and parallel processing are becoming more practical as technology continues to develop in these areas.

MCPI is a novel integration technique that has proven to be an efficient and robust algorithm, when compared with other state-of-the-practice numerical integrators. It is an iterative, path approximation method for solving smoothly nonlinear systems of ordinary differential equations comprised of two concepts: Picard iteration and Chebyshev polynomials. Picard iteration requires integration of the acceleration along a previous orbit approximation, while Chebyshev polynomials are used to approximate the current trajectory estimate. During every Picard iteration, the coefficients for a Chebyshev polynomial series for the orbit coordinates are updated to give the new state estimate that satisfies all boundary conditions. The entire trajectory along a segment is approximated at sample nodes by computing the forcing

function from the current state estimate, in contrast with traditional step-by-step methods. MCPI uses the CGL nodes to ensure discrete orthogonality conditions for the Chebyshev polynomials' extrema are satisfied, high accuracy is achieved, and the Runge effect that is often seen in function approximation is reduced.

Perturbed orbit propagation of the set of slowly varying MEEs with MCPI leads to a greatly increased domain of convergence (over the more commonly used Cartesian coordinate solution), a reduction in the number of MCPI iterations, and a reduction in the number of full spherical harmonic gravity function calls. Optimizing the MEE propagation through a segmentation scheme decreases the number of nodes and gravity function calls, at the cost of adding a few more MCPI iterations, to reduce the overall computation time. The computational results show that MCPI propagation using MEE coordinates provides significant computational advantages that will affect many dimensions of astrodynamics.

A Monte Carlo analysis provides statistical information, which utilizes the MCPI MEE propagation. The STM is also used in this algorithm to compute a local Taylor Series gravity approximation, thereby allowing for a full spherical harmonic gravity computation only for the nominal trajectory; all neighboring trajectories are computed using this gravity approximation. Serial and parallel results show a reduction in computation time when using the MEEs, compared with Cartesian propagation for the same Monte Carlo algorithm. The STM derivation addresses a gap in the literature to allow for user-specified spherical harmonic gravity perturbations to be included. Because the acceleration is computed in Cartesian coordinates for MEE IVP propagation, a transformation from MEEs to Cartesian is required; the corresponding Jacobian may then be easily computed for use in the Taylor Series expansion without any additional nonlinear transformations.

A low-thrust formulation using steering angles gives a suboptimal control that

allows a spacecraft to reach a target state after a multi-revolution transfer. Although using a direct approach is not guaranteeed to give the optimal solution, it is often accurate enough to be used in practice. This nonlinear programming approach is adopted both for the possibility to be used in missions and for those cases where rigorous satisfaction of the Pontryagin optimal control necessary conditions is ultimately sought in future work. The suboptimal control algorithm represents the steering angles using Chebyshev polynomials and incorporates the MPS shooting method to achieve its solution. The result obtained from this solution is intended to be used in future work to solve the full state/costate optimal control problem, where the initial guess for the costates is necessary but not easily determined.

The algorithms presented in this dissertation could potentially be used in the future for on-board trajectory computation, and the computational efficiency could be further increased through the use of parallel processors.

REFERENCES

[1] National Aeronautics and Space Administration. *Orbital Debris Quarterly News.* 20:14, April 2016, `https://orbitaldebris.jsc.nasa.gov/quarterly-news/newsletter.html`.

[2] Nitin Arora. High performance algorithms to improve the runtime computation of spacecraft trajectories. *PhD Dissertation, Georgia Institute of Technology*, 2013.

[3] D. L. Bahls and S. W. Paris. A mission analysis tool for complex low thrust interplanetary missions. *AIAA Astodynamics Conference, Danvers, MA*, AIAA-80-1674, DOI: 10.2514/6.1980-1674, August 11 -13, 1980.

[4] X. Bai and J.L. Junkins. Solving initial value problems by the Picard-Chebyshev method with nvidia GPUs. *Proceedings of the 20th Spaceflight Mechanics Meeting, San Diego, CA*, AAS 10-197, 2010.

[5] Xiaoli Bai. Modified chebyshev-picard iteration methods for solution of initial value and boundary value problems. *PhD Dissertation, Texas A&M University*, 2010.

[6] Ahmad Bani Younes and John L. Junkins. An adaptive approach for modified Chebyshev Picard iteration. *Proceedings of* 26^{th} *AAS/AIAA Space Flight Mechanics Meeting*, AAS 16-427, February 2016.

[7] Richard H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition.* AIAA Education Series, 1999.

[8] R. Broucke and P. Cefola. On the equinoctial orbit elements. *Celestial Mechanics*, 5(3):303-310, 1972.

[9] D. Brouwer and G.M. Clemence. *Methods of Celestial Mechanics*. Academic Press, 1961.

[10] P. Cefola and R. Broucke. On the formulation of the gravitational potential in terms of equinoctial variables. *AIAA 13th Aerospace Sciences Meeting*, DOI 10.2514/6.1975-9, 1975.

[11] Pafnuty Lvovich Chebyshev. Théorie des mécanismes connus sous le nom de parallélogrammes. *Mémoires des Savants éstrangers preèsentès à l'Académie de Saint Pétersbourg*, 7:539–586, 1857.

[12] C.W. Clenshaw and H.J. Norton. The solution of nonlinear ordinary differential equations in chebyshev series. *The Computer Journal*, 6:88–92, 1963.

[13] Bruce Conway. *Spacecraft Trajectory Optimization*. Cambridge University Press, 2010.

[14] D. A. Danielson, C.P. Sagovac, B. Neta, and L.W. Early. *Semianalytic Satellite Theory, NPS-MA-95-002*. Mathematics Department, Naval Postgraduate School Report. Monterey, CA, Feb 1995.

[15] T. Feagin. The numerical solution of two point boundary value problems using Chebyshev series. *PhD Dissertation, The University of Texas at Austin*, 1972.

[16] T. Feagin. High-order explicit runge-kutta methods using m-symmetry. *Neural, Parallel & Scientific Computations*, 20(4):437–458, 2012.

[17] T. Feagin. The iterative solution of the problem of orbit determination using Chebyshev series. *NASA CR-141317*, 31 January, 1975.

[18] T. Feagin. High-order m-symmetric runge-kutta methods. *Proceedings of the 23^{rd} Biennial Conference on Numerical Analysis, Strathclyde University, Glasgow, Scotland*, June 23 - 26, 2009.

[19] T. Feagin and R.P. Mikkilineni. The use of series solutions for batch and sequential estimation. *AAS Paper, AAS75-056*, 1975.

[20] T. Feagin and P. Nacozy. Matrix formulation of the Picard method for parallel computation. *Celestial Mechanics*, 29:107–115, 1983.

[21] Leslie L. Fox and I. B. Parker. *Chebyshev Polynomials in Numerical Analaysis*. Oxford University Press, 1968.

[22] T. Fukushima. Vector integration of dynamical motions by the Picard-Chebyshev method. *The Astronomical Journal*, 113:2325–2328, 1997.

[23] Y. Gao and C.A. Kluever. Low-thrust interplanetary orbit transfers using hybrid trajectory optimization method with multiple shooting. *Proceedings of AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Providence, Rhode Island*, AIAA 2004-5088, 16-19 August 2004.

[24] R.G. Gottlieb. Fast gravity, gravity partials, normalized gravity, gravity gradient torque and magnetic field: Derivation, code, and data. *Lyndon B. Johnson Space Center Internal Report*, NASA-CR-188243, NAS 1.26:188243, February 1993.

[25] J. Hyun Jo, I. Kwan Park, N. Choe, and M. Choi. The comparison of the classical keplerian orbit elements, non-singular orbital elements (equinoctial elements), and the Cartesian state variables in Lagrange planetary equations with j2 perturbation: Part 1. *Journal of Astronautical Space Sciences*, 28(1):37–54, 2011.

[26] J.L. Junkins and J.D. Turner. *Optimal Spacecraft Rotational Maneuvers*. Elsevier Science Ltd: Studies in Astronautics, 1986.

[27] John L. Junkins, Ahmad Bani Younes, Robyn M. Woollands, and Xiaoli Bai. Picard iteration, Chebyshev Polynomials, and Chebyshev-Picard methods: Application in astrodynamics. *Journal of Astronautical Sciences*, 60:623–653, 2013.

[28] D. Kim and J.L. Junkins. Multi-segment adaptive modified Chebyshev Picard iteration method. *Proceedings of* 24^{th} *AAS/AIAA Space Flight Mechanics Conference, Santa Fe, NM*, DOI: 10.13140/RG.2.1.1495.9441, 2014.

[29] D. Kim, J.L. Junkins, and J. Turner. Multisegment scheme applications to modified Chebyshev Picard iteration method for highly elliptical orbits. *Mathematical Problems in Engineering*, 1, DOI: 10.1155/2015/290781, January 2015.

[30] Mischa Kim. Continuous low-thrust trajectory optimization: Techniques and applications. *PhD Dissertation, Virginia Polytechnic Institute and State University*, 2001.

[31] Heiner Klinkrad. *Space Debris: Models and Risk Analysis*. Springer/Praxis Publishing Ltd, Chichester, UK; Printed in Germany, 2006.

[32] D. Koblick, M. Poole, and P. Shankar. Parallel high-precision orbit propagation using the modified picard-chebyshev method. *ASME International Mechanical Engineering Congress and Exposition, Houston, TX*, 93:587–605, IMECE2012-87878, November 9-15, 2012.

[33] Frank L. Lewis and Vassilis L. Syrmos. *Optimal Control,* 3^{rd} *Edition*. John Wiley & Sons, Inc. New York, NY, 1995.

[34] J.B. Lundberg and B. Schutz. Recursion formulas of Legendre functions for use with nonsingular geopotential models. *Journal of Guidance and Control 11(1):31-38*, DOI: 10.2514/3.20266, January 1988.

[35] B. Macomber, A. Probe, R. Woollands, and J.L. Junkins. Automated tuning parameter selection for orbit propagation with modified Chebyshev Picard iteration. *Proceedings of* 25^{th} *AAS/AIAA Space Flight Mechanics Meeting, Williamsburg, VA*, AAS 15-417, 2015.

[36] Brent Macomber. Enhancements to Chebyshev-Picard iteration efficiency for generally perturbed orbits and constrained dynamical systems. *PhD Dissertation, Texas A&M University*, 2015.

[37] Brent Macomber, Donghoon Kim, Robyn Woollands, and J.L. Junkins. Terminal convergence approximation modified Chebyshev Picard iteration for efficient numerical integration of orbital trajectories. *Proceedings of Advanced Maui Optical and Space Surveilliance Technologies Conference, Maui, HI*, page E20, September 9-12, 2014.

[38] Brent Macomber, Austin Probe, Robyn Woollands, and John L. Junkins. Parallel modified-Chebyshev Picard iteration for orbit catalog propagation and monte carlo analysis. *Proceedings of the 38th Annual AAS/AIAA Guidance and Control Conference, Breckenridge, CO*, AAS 15-009, 2015.

[39] Brent Macomber, Austin Probe, Robyn Woollands, Julie Read, and J.L. Junkins. Enhancements to modified Chebyshev-Picard iteration efficiency for perturbed orbit propagation. *Journal of Computer Modeling & Sciences (CMES) Special Issue on Computational Methods in Celestial Mechanics*, 111(1), 2016.

[40] John C. Mason and David Handscomb. *Chebyshev Polynomials*. Chapman & Hall/CRC Press LLC, Boca Raton, Florida, 2000.

[41] A. Miele and R.R. Lyer. General technique for solving nonlinear, two-point boundary-value problems via the method of particular solutions. *Journal of Optimization Theory and Applications*, 5:382–399, 1970.

[42] Angelo Miele. Method of particular solutions for linear, two-point boundary-value problems. *Journal of Optimization Theory and Applications*, 2:260–273, 1968.

[43] R.P. Mikkilineni and T. Feagin. The determination of orbits using Picard iteration. *NASA X-582-75-273, Goddard Space Flight Center*, (1-6), 1975.

[44] Émile Picard. Sur l'application des méthodes d'approximations successives à l'étude de certaines équations différentielles ordinaires. *Journal de Mathématiques Pures et Appliquées*, 9:217–272, 1893.

[45] Austin Probe, Brent Macomber, Julie Read, Robyn Woollands, and John L. Junkins. Radially adaptive evaluation of the spherical harmonic gravity series for numerical orbital propagation. *Proceedings of* 25^{th} *AAS/AIAA Space Flight Mechanics Meeting, Williamsburg, VA*, AAS 15-440, 2015.

[46] Austin Probe, Brent Macomber, Julie Read, Robyn Woollands, and John L. Junkins. Radially adaptive evaluation of the spherical harmonic gravity series for numerical orbit propagation. *Proceedings of* 25^{th} *AAS/AIAA Space Flight Mechanics Conference, Williamsburg, VA*, AAS 15-440, 2015.

[47] Austin Probe, Julie Read, Brent Macomber, and John L. Junkins. Massively parallel implementation of modified Chebyshev Picard iteration for perturbed

orbit propagation. *Proceedings of AAS/AIAA Astrodynamics Specialist Conference, Vail, CO*, pages 587–605, AAS 15-793, 2015.

[48] J. L. Read, A. Bani Younes, and J.L. Junkins. Efficient orbit propagation of orbital elements using modified Chebyshev Picard iteration method. *Journal of Computer Modeling & Sciences (CMES) Special Issue on Computational Methods in Celestial Mechanics*, 111(1), 2016.

[49] J.L. Read, A. Bani Younes, and J.L. Junkins. Efficient orbit propagation of orbital elements using modified Chebyshev Picard iteration method. *Proceedings of the ICCES Conference, Reno, NV*, 2015.

[50] Julie Read, Ahmad Bani Younes, J. Turner, and J.L. Junkins. State transition matrix propagation for perturbed orbital motion using modified Chebyshev Picard iteration. *Proceedings of AAS 38th Annual Guidance and Control Conference, Breckenridge, CO*, AAS 15-008, 2015.

[51] Julie Read, Tarek Elgohary, Austin Probe, and J.L. Junkins. Monte carlo propagation of orbital elements using modified Chebyshev Picard iteration. *Proceedings of* 26^{th} *AAS/AIAA Space Flight Mechanics Conference, Napa Valley, CA*, AAS 16-520, 2016.

[52] Julie Read, Brent Macomber, Ahmad Bani Younes, J. Turner, and J. L. Junkins. State transition matrix propagation for perturbed orbital motion using modified Chebyshev Picard iteration. *Journal of Astronautical Sciences, DOI 10.1007/s40295-015-0051-3, ISSN 0021-9142*, 62:148–167, June 2015.

[53] E.A. Roth. The gaussian form of the variation-of-parameter equations formulated in equinoctial elements - applications: Airdrag and radiation pressure. *Acta Astronautica*, 12:719–730, 1985.

[54] H. Schaub and J.L. Junkins. *Analytical Mechanics of Space Systems, 3^{rd} Edition.* AIAA Education Series, 2014.

[55] J.S. Shaver. Formulation and evaluation of parallel algorithms for the orbit determination problem. *PhD Thesis, MIT*, 1980.

[56] James Dana Thorne. Optimal Continuous-Thrust Orbit Transfers. *PhD Dissertation, Air Force Institute of Technology*, 1996.

[57] Boise State Matlab Tutorials. The runge phenomenon. Viewed Online 2016, `http://math.boisestate.edu/~calhoun/teaching/matlab-tutorials/lab_11/html/lab_11.html`.

[58] M.J.H. Walker, B. Ireland, and J. Owens. Errata: A set of modified equinoctial orbit elements. *Celestial Mechanics*, 36(4):409–419, 1985.

[59] M.J.H. Walker, B. Ireland, and J. Owens. A set of modified equinoctial orbit elements. *Celestial Mechanics*, 36(4):409–419, 1985.

[60] Robyn Woollands. Modified Chebyshev Picard iteration: Derivation and tutorial for the initial value problem, *Internal Research Team Report.* 2013.

[61] Robyn Woollands. Modified Chebyshev Picard iteration: Derivation and tutorial for the boundary value problem, *Internal Research Team Report.* 2015.

[62] Robyn Woollands. Regularization and computational methods for precise solution of perturbed orbit transfer problems. *PhD Dissertation, Texas A&M University*, 2016.

[63] Robyn Woollands, Ahmad Bani Younes, and John L. Junkins. New solutions for Lambert's problem utilizing regularization and Picard iteration. *AIAA Journal*

of Guidance, Control, and Dynamics: Special Issue in Honor of Richard Battin, 39(9):1548–1562, 2015.

[64] Robyn Woollands, J.L. Read, Brent Macomber, Austin Probe, Ahmad Bani Younes, and John L. Junkins. Method of particular solutions and kustaanheimo-stiefel regularized Picard iteration for solving two-point boundary value problems. *Proceedings of* 25^{th} *AAS/AIAA Space Flight Mechanics Conference, Williamsburg, VA*, AAS 15-373, 2015.

[65] Robyn M. Woollands, Julie L. Read, Brent Macomber, Austin B. Probe, Ahmad Bani Younes, and John L. Junkins. Parallel generation of extremal field maps for optimal multi-revolution continuous thrust orbit transfers. *Proceedings of AAS/AIAA Astrodynamics Specialist Conference, Vail, CO*, AAS 15-791, 2015.

[66] Robyn M. Woollands, Julie L. Read, Austin B. Probe, and John L. Junkins. Method of particular solutions and modified Chebyshev-Picard iteration for solving multi-revolution perturbed Lambert problems. *DOI: 10.1007/s40295-016-0107-z, Journal of Astronautical Sciences*, (Under Review), 2016.

[67] Robyn M. Woollands, Julie L. Read, Austin B. Probe, and John L. Junkins. Adaptive two-point boundary value problem tool for accurate and efficient computation of perturbed orbit transfers. *Proceedings of* 26^{th} *AAS/AIAA Space Flight Mechanics Conference, Napa Valley, CA*, AAS 16-497, 2016.

[68] Ahmad Bani Younes. Orthogonal polynomial approximation in higher dimensions: Applications in astrodynamics. *PhD Dissertation, Texas A&M University*, 2013.

APPENDIX A

CHEBYSHEV POLYNOMIALS

Chebyshev polynomials form the orthogonal basis set used throughout this dissertation to approximate both the forcing function in the integral of the right hand side of Picard's equation, and in the original MCPI formulation, also the current estimate of the trajectory on the left hand side of Picard's equation introduced in Chapter 2. Picard's equation is repeated here for convenience:

$$\mathbf{x}^i(t) = \mathbf{x}(t_0) + \int_{t_0}^{t} \mathbf{f}(\tau, \mathbf{x}^{i-1}(\tau))d\tau, \quad i = 1, 2, ... \tag{A.1}$$

After expanding in terms of the Chebyshev polynomials T_k, this equation becomes

$$\sum_{k=0}^{N}{}'' \beta_k^i T_k(\tau) = \mathbf{x}(-1) + \int_{-1}^{\tau} \left[\mathbf{F}_k^{i-1} T_k(s) \right] ds \tag{A.2}$$

A.1 Orthogonality

Two functions $f(x)$ and $g(x)$ are defined as orthogonal on the interval $[a, b]$ with respect to a given weight function $w(x)$ (which must be continuous and non-negative) if the inner product is zero:

$$\langle f,\ g \rangle \equiv \int_a^b w(x) f(x) g(x) dx = 0 \tag{A.3}$$

For the weight function $w(x) = (1 - x^2)^{-\frac{1}{2}}$, the Chebyshev polynomials are orthogonal [40]:

$$\langle T_i,\ T_j \rangle = \int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}} dx = 0 \quad (i \neq j) \tag{A.4}$$

For the case where $n = m = 0$, Equation (A.4) is equal to π, and when $n = m \neq 0$, Equation (A.4) is equal to $\frac{\pi}{2}$. An anaologous discrete orthogonality property exists when the Chebyshev polynomials are sampled either at their extrema or at their zeroes. The Chebyshev-Gauss-Lobatto (CGL) nodes are used for the MCPI algorithm, where the extrema of T_N are located. These nodes are the extrema of T_M and are computed through

$$\tau_j = -cos\left(\frac{\pi}{M} j\right) \quad j = 0, 1, 2, ..., M \tag{A.5}$$

When the number of CGL nodes is equal to the order of the Chebyshev approximation ($M = N$), the discrete orthogonality condition becomes [21]

$$\sum_{j=0}^{M} w_j T_n(\tau_j) T_m(\tau_j) = \begin{cases} 0 & \text{if } n \neq m \\ M & \text{if } n = m = 0 \\ \frac{M}{2} & \text{if } n = m \neq 0 \end{cases}$$

In this case, orthogonality requires the discrete weight function to depend upon the node, where the known boundary nodes are $w_0 = w_N = \frac{1}{2}$ and all other interior $w_j = 1$. To follow the notation conventions used in previous related publications (e.g., [21]), throughout this dissertation a ($''$) symbol within the summation means that the first and last terms have a weight of $\frac{1}{2}$, while the rest of the terms have a weight of 1. Similarly, a ($'$) symbol means that the first term is multiplied by a $\frac{1}{2}$, while the rest of the terms are multiplied by 1.

A.2 Method of Approximation

For the MCPI algorithm described throughout this dissertation the order of the Chebyshev approximation, N, and the number of CGL sample points, ($M \geq N$) are user-specified. The Chebyshev polynomial basis set is used to approximate a discrete function using either least squares ($M > N$) or a direct interpolation ($M = N$). Since the user chooses the values of M and N, this implies that the user also chooses to use either the direct interpolation or least squares methods (which differes by an extra $\frac{1}{2}$ term at the end of the series for the interpolation case). For the case of $M > N$ samples at the CGL nodes, the N^{th} order discrete least squares approximation of a forcing function $g(x(\tau))$, for instance, two-body acceleration, is given by

$$g(x(\tau)) = \sum_{k=0}^{M}{}' F_k T_k(x) \tag{A.6}$$

where the coefficients of the Chebyshev polynomial series are

$$F_k = \frac{2}{M} \sum_{j=0}^{M}{}'' g(x(\tau_j)) T_k(\tau_j) \tag{A.7}$$

When $M = N$ CGL nodes, the interpolation formula is an exact fit at the chosen CGL nodes. This approach yields the approximation

$$g(x(\tau)) = \sum_{k=0}^{M}{}'' F_k T_k(x) \tag{A.8}$$

$$F_k = \frac{2}{M} \sum_{j=0}^{M}{}'' g(x(\tau_j)) T_k(\tau_j) \tag{A.9}$$

For more information, see Chapter 2, Section 2.5.

APPENDIX B

SUPPLEMENT TO STATE TRANSITION MATRIX

This appendix provides additional derivation details for the State Transition Matrix (STM) as described in Ch. (4), first for spherical harmonic (in both Earth-Centered Earth-Fixed and Earth Centered Inertial frames), then for zonal gravity, and finally for two-body (unperturbed) gravity. The zonal gravity accelerations and associated STM may be used either for a low-fidelity standalone model, or when using a local Taylor Series gravity expansion as described in Chapter 5, Section 5.1.

B.1 Jacobian Matrix G for Spherical Harmonic Gravity in ECEF Frame

All of the expressions in this section are used to compute the STM for the spherical harmonic gravity model, as described in Chapter 4. The ECEF Jacobian presented in this section may be used instead of the ECI Jacobian presented in Chapter 4, although computing the ECEF Jacobian and transforming it to the ECI frame for propagation is more efficient.

The components of the ECEF Jacobian Matrix required to compute the Spherical Harmonic gravity model are given individually here in terms of spherical coordinates (r, ϕ, λ) as well as the partials of the gravity potential, i.e., U_r is the partial of gravity potential U with respect to r. Following the individual terms for G_{ij} given here, the expressions for partial derivative terms in Earth-Centered, Earth-Fixed (ECEF) components (x, y, z) are also given. Once the Jacobian G is computed in the ECEF frame, it may be transformed into the ECI frame using a direction cosine matrix for propagation (see Chapter 4).

$$[G] = \begin{bmatrix} \frac{\partial}{\partial x}(a_X) & \frac{\partial}{\partial y}(a_X) & \frac{\partial}{\partial z}(a_X) \\ \frac{\partial}{\partial x}(a_Y) & \frac{\partial}{\partial y}(a_Y) & \frac{\partial}{\partial z}(a_Y) \\ \frac{\partial}{\partial x}(a_Z) & \frac{\partial}{\partial y}(a_Z) & \frac{\partial}{\partial z}(a_Z) \end{bmatrix} \tag{B.1}$$

$$\begin{aligned} G_{11} &= \frac{\partial}{\partial x}(a_X) \\ &= \left(\frac{\partial r}{\partial x}\right)^2 \left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial \phi}{\partial x}\right)^2 \left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial x}\right)^2 \left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\ &+ 2\left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + 2\left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\ &+ 2\left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) + \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial x^2}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial x^2}\right) \\ &+ \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial x^2}\right) \end{aligned} \tag{B.2}$$

$$\begin{aligned} G_{12} &= \frac{\partial}{\partial y}(a_X) \\ &= \left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + \left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\ &+ \left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial U_\phi}{\partial r}\right) + \left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) \\ &+ \left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial U_\lambda}{\partial r}\right) + \left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial U_\lambda}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\ &+ \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial x \partial y}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial x \partial y}\right) + \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial x \partial y}\right) \end{aligned} \tag{B.3}$$

$$
\begin{aligned}
G_{13} &= \frac{\partial}{\partial z}(a_X) \\
&= \left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + \left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial U_\phi}{\partial r}\right) + \left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial U_\lambda}{\partial r}\right) + \left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial U_\lambda}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\
&+ \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial x \partial z}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial x \partial z}\right) + \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial x \partial z}\right)
\end{aligned}
\tag{B.4}
$$

$$
\begin{aligned}
G_{21} &= \frac{\partial}{\partial x}(a_Y) \\
&= \left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + \left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial U_\phi}{\partial r}\right) + \left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial U_\lambda}{\partial r}\right) + \left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial U_\lambda}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\
&+ \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial y \partial x}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial y \partial x}\right) + \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial y \partial x}\right)
\end{aligned}
\tag{B.5}
$$

$$
\begin{aligned}
G_{22} &= \frac{\partial}{\partial y}(a_Y) \\
&= \left(\frac{\partial r}{\partial y}\right)^2\left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial \phi}{\partial y}\right)^2\left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial y}\right)^2\left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\
&+ 2\left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + 2\left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\
&+ 2\left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) + \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial y^2}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial y^2}\right) \\
&+ \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial y^2}\right)
\end{aligned}
\tag{B.6}
$$

$$
\begin{aligned}
G_{23} &= \frac{\partial}{\partial z}(a_Y) \\
&= \left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + \left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial U_\phi}{\partial r}\right) + \left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial U_\lambda}{\partial r}\right) + \left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial U_\lambda}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\
&+ \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial y \partial z}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial y \partial z}\right) + \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial y \partial z}\right)
\end{aligned}
\tag{B.7}
$$

$$
\begin{aligned}
G_{31} &= \frac{\partial}{\partial x}(a_Z) \\
&= \left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + \left(\frac{\partial r}{\partial x}\right)\left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial U_\phi}{\partial r}\right) + \left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \phi}{\partial x}\right)\left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial U_\lambda}{\partial r}\right) + \left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial U_\lambda}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial x}\right)\left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\
&+ \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial z \partial x}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial z \partial x}\right) + \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial z \partial x}\right)
\end{aligned}
\tag{B.8}
$$

$$
\begin{aligned}
G_{32} &= \frac{\partial}{\partial y}(a_Z) \\
&= \left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + \left(\frac{\partial r}{\partial y}\right)\left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial U_\phi}{\partial r}\right) + \left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \phi}{\partial y}\right)\left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) \\
&+ \left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial U_\lambda}{\partial r}\right) + \left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial U_\lambda}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial y}\right)\left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\
&+ \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial z \partial y}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial z \partial y}\right) + \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial z \partial y}\right)
\end{aligned}
\tag{B.9}
$$

$$
\begin{aligned}
G_{33} &= \frac{\partial}{\partial z}(a_Z) \\
&= \left(\frac{\partial r}{\partial z}\right)^2 \left(\frac{\partial U_r}{\partial r}\right) + \left(\frac{\partial \phi}{\partial z}\right)^2 \left(\frac{\partial U_\phi}{\partial \phi}\right) + \left(\frac{\partial \lambda}{\partial z}\right)^2 \left(\frac{\partial U_\lambda}{\partial \lambda}\right) \\
&+ 2\left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial U_r}{\partial \phi}\right) + 2\left(\frac{\partial r}{\partial z}\right)\left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial U_r}{\partial \lambda}\right) \\
&+ 2\left(\frac{\partial \phi}{\partial z}\right)\left(\frac{\partial \lambda}{\partial z}\right)\left(\frac{\partial U_\phi}{\partial \lambda}\right) + \left(\frac{\partial U}{\partial r}\right)\left(\frac{\partial^2 r}{\partial z^2}\right) + \left(\frac{\partial U}{\partial \phi}\right)\left(\frac{\partial^2 \phi}{\partial z^2}\right) \\
&+ \left(\frac{\partial U}{\partial \lambda}\right)\left(\frac{\partial^2 \lambda}{\partial z^2}\right)
\end{aligned}
\tag{B.10}
$$

See the following section for the partial deriatives of spherical coordinates with respect to (x, y, z) that are used to compute this Jacobian matrix.

B.1.1 Partial Derivatives for Jacobian Matrix G in ECEF Frame Using Spherical Harmonic Gravity

The partial derivatives of spherical coordinates, given in this subsection, are used to compute the Jacobian matrix G as required in the expressions in the previous section. The partials are taken with respect to the components (x, y, z), which are in the ECEF frame. These partials of the gravity potential U with respect to spherical coordinates are already given in Ch. 4; all of the partial derivatives are given here (the first partials are given in Ch. 4 and repeated here for convenience).

The first partial derivatives of spherical coordinates with respect to Cartesian coordinates are given by

$$
\frac{\partial r}{\partial \alpha} = \frac{\alpha}{r}; \quad \alpha \to x, y, z \tag{B.11}
$$

$$\frac{\partial \phi}{\partial x} = \frac{-xz}{r^2\sqrt{x^2+y^2}}; \quad \frac{\partial \phi}{\partial y} = \frac{-yz}{r^2\sqrt{x^2+y^2}}; \quad \frac{\partial \phi}{\partial z} = \frac{\left(1-\frac{z^2}{r^2}\right)}{\sqrt{x^2+y^2}} \tag{B.12}$$

$$\frac{\partial \lambda}{\partial x} = \frac{-y}{x^2+y^2}; \quad \frac{\partial \lambda}{\partial y} = \frac{x}{x^2+y^2}; \quad \frac{\partial \lambda}{\partial z} = 0 \tag{B.13}$$

The second partial derivatives of spherical coordinates with respect to Cartesian coordinates are given by

$$\frac{\partial^2 r}{\partial x^2} = \frac{1}{r} - \frac{x^2}{r^3} \tag{B.14}$$

$$\frac{\partial^2 \phi}{\partial x^2} = -\frac{z(y^2z^2 + y^4 - x^2y^2 - 2x^4)}{(y^2+x^2)^{\frac{3}{2}}(z^2+y^2+x^2)^2} \tag{B.15}$$

$$\frac{\partial^2 \lambda}{\partial x^2} = \frac{2xy}{(y^2+x^2)^2} \tag{B.16}$$

$$\frac{\partial^2 r}{\partial x \partial y} = -\frac{xy}{(z^2+y^2+x^2)^{\frac{3}{2}}} \tag{B.17}$$

$$\frac{\partial^2 \phi}{\partial x \partial y} = \frac{xyz(z^2+3y^2+3x^2)}{(y^2+x^2)^{\frac{3}{2}}(z^2+y^2+x^2)^2} \tag{B.18}$$

$$\frac{\partial^2 \lambda}{\partial x \partial y} = \frac{2y^2}{(y^2+x^2)^2} - \frac{1}{y^2+x^2} \tag{B.19}$$

$$\frac{\partial^2 r}{\partial x \partial z} = -\frac{xz}{(z^2+y^2+x^2)^{\frac{3}{2}}} \tag{B.20}$$

$$\frac{\partial^2 \phi}{\partial x \partial z} = \frac{x(z^2 - y^2 - x^2)}{\sqrt{y^2 + x^2}(z^2 + y^2 + x^2)^2} \tag{B.21}$$

$$\frac{\partial^2 \lambda}{\partial x \partial z} = 0 \tag{B.22}$$

$$\frac{\partial^2 r}{\partial y^2} = \frac{1}{\sqrt{z^2 + y^2 + x^2}} - \frac{y^2}{(z^2 + y^2 + x^2)^{\frac{3}{2}}} \tag{B.23}$$

$$\begin{aligned}\frac{\partial^2 \phi}{\partial y^2} = &-\frac{z}{\sqrt{y^2 + x^2}(z^2 + y^2 + x^2)} + \frac{y^2 z}{(y^2 + x^2)^{\frac{3}{2}}(z^2 + y^2 + x^2)} \\ &+ \frac{2y^2 z}{\sqrt{y^2 + x^2}(z^2 + y^2 + x^2)^2}\end{aligned} \tag{B.24}$$

$$\frac{\partial^2 \lambda}{\partial y^2} = -\frac{2xy}{(y^2 + x^2)^2} \tag{B.25}$$

$$\frac{\partial^2 r}{\partial y \partial z} = -\frac{yz}{(z^2 + y^2 + x^2)^{\frac{3}{2}}} \tag{B.26}$$

$$\frac{\partial^2 \phi}{\partial y \partial z} = \frac{2yz^2}{\sqrt{y^2 + x^2}(z^2 + y^2 + x^2)^2} - \frac{y\left(1 - \frac{z^2}{z^2 + y^2 + x^2}\right)}{(y^2 + x^2)^{\frac{3}{2}}} \tag{B.27}$$

$$\frac{\partial^2 \lambda}{\partial y \partial z} = 0 \tag{B.28}$$

$$\frac{\partial^2 r}{\partial z^2} = \frac{1}{\sqrt{z^2 + y^2 + x^2}} - \frac{z^2}{(z^2 + y^2 + x^2)^{\frac{3}{2}}} \tag{B.29}$$

$$\frac{\partial^2 \phi}{\partial z^2} = \frac{1}{\sqrt{y^2+x^2}}\left(\frac{2z^3}{(z^2+y^2+x^2)^2} - \frac{2z}{(z^2+y^2+x^2)}\right) \tag{B.30}$$

$$\frac{\partial^2 \lambda}{\partial z^2} = 0 \tag{B.31}$$

B.2 Jacobian Matrix G for Spherical Harmonic Gravity in ECI Frame

The partial derivatives of spherical coordinates, given in this subsection, are used to compute the Jacobian matrix G in the ECI frame as given in Eq. (B.50). The partial derivatives are taken with respect to the components (X, Y, Z), which are in the ECI frame, as opposed to partial derivatives with respect to the ECEF frame (as given in the previous section). The ECI Jacobian presented in this section may be used instead of the ECEF Jacobian presented in the previous section, although computing the ECEF Jacobian and transforming it to the ECI frame for propagation is more efficient, as described in Chapter 4.

The partials of the gravity potential U with respect to spherical coordinates are already given in Ch. 4. All of the partial derivatives are given here (the first partials are given in Ch. 4 and repeated here for convenience).

The first partial derivatives of spherical coordinates with respect to Cartesian coordinates in ECEF are given by

$$\frac{\partial r}{\partial \alpha} = \frac{\alpha}{r}; \quad \alpha \to x, y, z \tag{B.32}$$

$$\frac{\partial \phi}{\partial x} = \frac{-xz}{r^2\sqrt{x^2+y^2}}; \quad \frac{\partial \phi}{\partial y} = \frac{-yz}{r^2\sqrt{x^2+y^2}}; \quad \frac{\partial \phi}{\partial z} = \frac{\left(1-\frac{z^2}{r^2}\right)}{\sqrt{x^2+y^2}} \tag{B.33}$$

$$\frac{\partial \lambda}{\partial x} = \frac{-y}{x^2 + y^2}; \quad \frac{\partial \lambda}{\partial y} = \frac{x}{x^2 + y^2}; \quad \frac{\partial \lambda}{\partial z} = 0 \tag{B.34}$$

The following first partial derivatives of spherical coordinates with respect to Cartesian coordinates in ECI are given using the rotation matrix

$$C(t) = \left[\frac{\partial(x, y, z)}{\partial(X, Y, Z)} \right] \tag{B.35}$$

Note that this rotation matrix implies that

$$C^T(t) = \left[\frac{\partial(X, Y, Z)}{\partial(x, y, z)} \right] \tag{B.36}$$

These partial derivatives are given in terms of individual rotation matrix components C_{ij} as

$$\frac{\partial r}{\partial X} = \cos\phi C_{21} \sin\lambda + \cos\phi C_{11} \cos\lambda + \sin\phi C_{31} \tag{B.37}$$

$$\frac{\partial \phi}{\partial X} = \frac{1}{r}\big(- \sin\phi C_{21} \sin\lambda - \sin\phi C_{11} \cos\lambda + \cos\phi C_{31}\big) \tag{B.38}$$

$$\frac{\partial \lambda}{\partial X} = \frac{(C_{21} \cos\lambda)}{(r \cos\phi)} - \frac{(C_{11} \sin\lambda)}{(r \cos\phi)} \tag{B.39}$$

$$\frac{\partial r}{\partial Y} = \cos\phi C_{22} \sin\lambda + \cos\phi C_{12} \cos\lambda + \sin\phi C_{32} \tag{B.40}$$

$$\frac{\partial \phi}{\partial Y} = \frac{1}{r}\big(- \sin\phi C_{22} \sin\lambda - \sin\phi C_{12} \cos\lambda + \cos\phi C_{32}\big) \tag{B.41}$$

$$\frac{\partial \lambda}{\partial Y} = \frac{(C_{22}\cos\lambda)}{(r\cos\phi)} - \frac{(C_{12}\sin\lambda)}{(r\cos\phi)} \tag{B.42}$$

$$\frac{\partial r}{\partial Z} = \cos\phi C_{23}\sin\lambda + \cos\phi C_{13}\cos\lambda + \sin\phi C_{33} \tag{B.43}$$

$$\frac{\partial \phi}{\partial Z} = \frac{1}{r}\big(-\sin\phi C_{23}\sin\lambda - \sin\phi C_{13}\cos\lambda + \cos\phi C_{33}\big) \tag{B.44}$$

$$\frac{\partial \lambda}{\partial Z} = \frac{(C_{23}\cos\lambda)}{(r\cos\phi)} - \frac{(C_{13}\sin\lambda)}{(r\cos\phi)} \tag{B.45}$$

For a general function F, a chain rule expansion may be used to find partials with respect to body frame coordinates, i.e.,

$$\frac{\partial F}{\partial \alpha} = \frac{\partial F}{\partial r}\frac{\partial r}{\partial \alpha} + \frac{\partial F}{\partial \phi}\frac{\partial \phi}{\partial \alpha} + \frac{\partial F}{\partial \lambda}\frac{\partial \lambda}{\partial \alpha} \quad \alpha \to x, y, z \tag{B.46}$$

Next, define a matrix of these partial derivatives with respect to ECI coordinates as

$$D = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial \phi}{\partial x} & \frac{\partial \lambda}{\partial x} \\ \frac{\partial r}{\partial y} & \frac{\partial \phi}{\partial y} & \frac{\partial \lambda}{\partial y} \\ \frac{\partial r}{\partial z} & \frac{\partial \phi}{\partial z} & \frac{\partial \lambda}{\partial z} \end{bmatrix} \tag{B.47}$$

Written in matrix form and using this matrix, Eqns. (B.46) give

$$\begin{bmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{bmatrix} = [D(x,y,z)] \begin{bmatrix} \frac{\partial F}{\partial r} \\ \frac{\partial F}{\partial \phi} \\ \frac{\partial F}{\partial \lambda} \end{bmatrix} \tag{B.48}$$

Concatenate the spherical harmonic acceleration terms:

$$\mathbf{a} = \begin{bmatrix} \frac{\partial U}{\partial r} \\ \frac{\partial U}{\partial \phi} \\ \frac{\partial U}{\partial \lambda} \end{bmatrix} \tag{B.49}$$

Then the Jacobian matrix G in the ECI frame may be computed as

$$G = \begin{bmatrix} \frac{\partial}{\partial X}(\mathbf{a}) & \frac{\partial}{\partial Y}(\mathbf{a}) & \frac{\partial}{\partial Z}(\mathbf{a}) \end{bmatrix} \tag{B.50}$$

where the inertial second partials of the potential (i.e., the first partials of spherical harmonic gravity) are

$$\frac{\partial}{\partial X}(\mathbf{a}) = [C]^T \left[\frac{\partial D}{\partial X}\right] \mathbf{a} + [C]^T [D] \frac{\partial X}{\partial \mathbf{a}} \tag{B.51}$$

$$\frac{\partial}{\partial Y}(\mathbf{a}) = [C]^T \left[\frac{\partial D}{\partial Y}\right] \mathbf{a} + [C]^T [D] \frac{\partial Y}{\partial \mathbf{a}} \tag{B.52}$$

$$\frac{\partial}{\partial Z}(\mathbf{a}) = [C]^T \left[\frac{\partial D}{\partial Z}\right] \mathbf{a} + [C]^T [D] \frac{\partial Z}{\partial \mathbf{a}} \tag{B.53}$$

The partial derivatives in these expressions are inertial partial derivatives of the ECEF components (x, y, z) and the spherical components (r, ϕ, λ):

$$\left[\frac{\partial D}{\partial X}\right] = \begin{bmatrix} \frac{\partial}{\partial X}\left(\frac{\partial r}{\partial x}\right) & \frac{\partial}{\partial X}\left(\frac{\partial \phi}{\partial x}\right) & \frac{\partial}{\partial X}\left(\frac{\partial \lambda}{\partial x}\right) \\ \frac{\partial}{\partial X}\left(\frac{\partial r}{\partial y}\right) & \frac{\partial}{\partial X}\left(\frac{\partial \phi}{\partial y}\right) & \frac{\partial}{\partial X}\left(\frac{\partial \lambda}{\partial y}\right) \\ \frac{\partial}{\partial X}\left(\frac{\partial r}{\partial z}\right) & \frac{\partial}{\partial X}\left(\frac{\partial \phi}{\partial z}\right) & \frac{\partial}{\partial X}\left(\frac{\partial \lambda}{\partial z}\right) \end{bmatrix} \tag{B.54}$$

$$\left[\frac{\partial D}{\partial Y}\right] = \begin{bmatrix} \frac{\partial}{\partial Y}\left(\frac{\partial r}{\partial x}\right) & \frac{\partial}{\partial Y}\left(\frac{\partial \phi}{\partial x}\right) & \frac{\partial}{\partial Y}\left(\frac{\partial \lambda}{\partial x}\right) \\ \frac{\partial}{\partial Y}\left(\frac{\partial r}{\partial y}\right) & \frac{\partial}{\partial Y}\left(\frac{\partial \phi}{\partial y}\right) & \frac{\partial}{\partial Y}\left(\frac{\partial \lambda}{\partial y}\right) \\ \frac{\partial}{\partial Y}\left(\frac{\partial r}{\partial z}\right) & \frac{\partial}{\partial Y}\left(\frac{\partial \phi}{\partial z}\right) & \frac{\partial}{\partial Y}\left(\frac{\partial \lambda}{\partial z}\right) \end{bmatrix} \tag{B.55}$$

$$\left[\frac{\partial D}{\partial Z}\right] = \begin{bmatrix} \frac{\partial}{\partial Z}\left(\frac{\partial r}{\partial x}\right) & \frac{\partial}{\partial Z}\left(\frac{\partial \phi}{\partial x}\right) & \frac{\partial}{\partial Z}\left(\frac{\partial \lambda}{\partial x}\right) \\ \frac{\partial}{\partial Z}\left(\frac{\partial r}{\partial y}\right) & \frac{\partial}{\partial Z}\left(\frac{\partial \phi}{\partial y}\right) & \frac{\partial}{\partial Z}\left(\frac{\partial \lambda}{\partial y}\right) \\ \frac{\partial}{\partial Z}\left(\frac{\partial r}{\partial z}\right) & \frac{\partial}{\partial Z}\left(\frac{\partial \phi}{\partial z}\right) & \frac{\partial}{\partial Z}\left(\frac{\partial \lambda}{\partial z}\right) \end{bmatrix} \tag{B.56}$$

$$\frac{\partial X}{\partial \mathbf{a}} = \begin{bmatrix} \frac{\partial}{\partial X}\left(\frac{\partial U}{\partial r}\right) \\ \frac{\partial}{\partial X}\left(\frac{\partial U}{\partial \phi}\right) \\ \frac{\partial}{\partial X}\left(\frac{\partial U}{\partial \lambda}\right) \end{bmatrix} \tag{B.57}$$

$$\frac{\partial Y}{\partial \mathbf{a}} = \begin{bmatrix} \frac{\partial}{\partial Y}\left(\frac{\partial U}{\partial r}\right) \\ \frac{\partial}{\partial Y}\left(\frac{\partial U}{\partial \phi}\right) \\ \frac{\partial}{\partial Y}\left(\frac{\partial U}{\partial \lambda}\right) \end{bmatrix} \tag{B.58}$$

$$\frac{\partial Z}{\partial \mathbf{a}} = \begin{bmatrix} \frac{\partial}{\partial Z}\left(\frac{\partial U}{\partial r}\right) \\ \frac{\partial}{\partial Z}\left(\frac{\partial U}{\partial \phi}\right) \\ \frac{\partial}{\partial Z}\left(\frac{\partial U}{\partial \lambda}\right) \end{bmatrix} \tag{B.59}$$

The terms within these expressions are given as follows, where $\phi = \sin^{-1}\frac{z}{r}$ and $\lambda = \tan^{-1}\frac{y}{x}$.

$$\frac{\partial}{\partial X}\left(\frac{\partial U}{\partial r}\right) = \frac{\partial}{\partial X}\left(U_r\right) = \frac{\partial U_r}{\partial r}\frac{\partial r}{\partial X} + \frac{\partial U_r}{\partial \phi}\frac{\partial \phi}{\partial X} + \frac{\partial U_r}{\partial \lambda}\frac{\partial \lambda}{\partial X} \tag{B.60}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial U}{\partial \phi}\right) = \frac{\partial}{\partial X}\left(U_\phi\right) = \frac{\partial U_\phi}{\partial r}\frac{\partial r}{\partial X} + \frac{\partial U_\phi}{\partial \phi}\frac{\partial \phi}{\partial X} + \frac{\partial U_\phi}{\partial \lambda}\frac{\partial \lambda}{\partial X} \tag{B.61}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial U}{\partial \lambda}\right) = \frac{\partial}{\partial X}\left(U_\lambda\right) = \frac{\partial U_\lambda}{\partial r}\frac{\partial r}{\partial X} + \frac{\partial U_\lambda}{\partial \phi}\frac{\partial \phi}{\partial X} + \frac{\partial U_\lambda}{\partial \lambda}\frac{\partial \lambda}{\partial X} \tag{B.62}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial U}{\partial r}\right) = \frac{\partial}{\partial Y}\left(U_r\right) = \frac{\partial U_r}{\partial r}\frac{\partial r}{\partial Y} + \frac{\partial U_r}{\partial \phi}\frac{\partial \phi}{\partial Y} + \frac{\partial U_r}{\partial \lambda}\frac{\partial \lambda}{\partial Y} \tag{B.63}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial U}{\partial \phi}\right) = \frac{\partial}{\partial Y}\left(U_\phi\right) = \frac{\partial U_\phi}{\partial r}\frac{\partial r}{\partial Y} + \frac{\partial U_\phi}{\partial \phi}\frac{\partial \phi}{\partial Y} + \frac{\partial U_\phi}{\partial \lambda}\frac{\partial \lambda}{\partial Y} \tag{B.64}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial U}{\partial \lambda}\right) = \frac{\partial}{\partial Y}\left(U_\lambda\right) = \frac{\partial U_\lambda}{\partial r}\frac{\partial r}{\partial Y} + \frac{\partial U_\lambda}{\partial \phi}\frac{\partial \phi}{\partial Y} + \frac{\partial U_\lambda}{\partial \lambda}\frac{\partial \lambda}{\partial Y} \tag{B.65}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial U}{\partial r}\right) = \frac{\partial}{\partial Z}\left(U_r\right) = \frac{\partial U_r}{\partial r}\frac{\partial r}{\partial Z} + \frac{\partial U_r}{\partial \phi}\frac{\partial \phi}{\partial Z} + \frac{\partial U_r}{\partial \lambda}\frac{\partial \lambda}{\partial Z} \tag{B.66}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial U}{\partial \phi}\right) = \frac{\partial}{\partial Z}\left(U_\phi\right) = \frac{\partial U_\phi}{\partial r}\frac{\partial r}{\partial Z} + \frac{\partial U_\phi}{\partial \phi}\frac{\partial \phi}{\partial Z} + \frac{\partial U_\phi}{\partial \lambda}\frac{\partial \lambda}{\partial Z} \tag{B.67}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial U}{\partial \lambda}\right) = \frac{\partial}{\partial Z}\left(U_\lambda\right) = \frac{\partial U_\lambda}{\partial r}\frac{\partial r}{\partial Z} + \frac{\partial U_\lambda}{\partial \phi}\frac{\partial \phi}{\partial Z} + \frac{\partial U_\lambda}{\partial \lambda}\frac{\partial \lambda}{\partial Z} \tag{B.68}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial r}{\partial x}\right) = -\cos\phi\frac{\partial \lambda}{\partial X}\sin\lambda - \sin\phi\frac{\partial \phi}{\partial X}\cos\lambda \tag{B.69}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial r}{\partial y}\right) = \cos\phi\frac{\partial \lambda}{\partial X}\cos\lambda - \sin\phi\frac{\partial \phi}{\partial X}\sin\lambda \tag{B.70}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial r}{\partial z}\right) = \cos\phi\frac{\partial \phi}{\partial X} \tag{B.71}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial \phi}{\partial x}\right) = \frac{(\sin\phi\frac{\partial \lambda}{\partial X}\sin\lambda)}{r} + \frac{(\sin\phi\frac{\partial r}{\partial X}\cos\lambda)}{r^2} - \frac{(\cos\phi\frac{\partial \phi}{\partial X}\cos\lambda)}{r} \tag{B.72}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial \phi}{\partial y}\right) = \frac{(\sin\phi\frac{\partial r}{\partial X}\sin\lambda)}{r^2} - \frac{(\cos\phi\frac{\partial \phi}{\partial X}\sin\lambda)}{r} - \frac{(\sin\phi\frac{\partial \lambda}{\partial X}\cos\lambda)}{r} \tag{B.73}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial \phi}{\partial z}\right) = -\frac{(\cos\phi\frac{\partial r}{\partial X})}{r^2} - \frac{(\sin\phi\frac{\partial \phi}{\partial X})}{r} \tag{B.74}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial \lambda}{\partial x}\right) = \frac{(\frac{\partial r}{\partial X}\sin\lambda)}{(r^2\cos\phi)} - \frac{(\sin\phi\frac{\partial \phi}{\partial X}\sin\lambda)}{(r\cos^2\phi)} - \frac{(\frac{\partial \lambda}{\partial X}\cos\lambda)}{(r\cos\phi)} \tag{B.75}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial \lambda}{\partial y}\right) = -\frac{(\frac{\partial \lambda}{\partial X}\sin\lambda)}{(r\cos\phi)} - \frac{(\frac{\partial r}{\partial X}\cos\lambda)}{(r^2\cos\phi)} + \frac{(\sin\phi\frac{\partial \phi}{\partial X}\cos\lambda)}{(r\cos^2\phi)} \tag{B.76}$$

$$\frac{\partial}{\partial X}\left(\frac{\partial \lambda}{\partial z}\right) = 0 \tag{B.77}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial r}{\partial x}\right) = -\cos\phi\frac{\partial \lambda}{\partial Y}\sin\lambda - \sin\phi\frac{\partial \phi}{\partial Y}\cos\lambda \tag{B.78}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial r}{\partial y}\right) = \cos\phi\frac{\partial \lambda}{\partial Y}\cos\lambda - \sin\phi\frac{\partial \phi}{\partial Y}\sin\lambda \tag{B.79}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial r}{\partial z}\right) = \cos\phi\frac{\partial \phi}{\partial Y} \tag{B.80}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial \phi}{\partial x}\right) = \frac{(\sin\phi\frac{\partial \lambda}{\partial Y}\sin\lambda)}{r} + \frac{(\sin\phi\frac{\partial r}{\partial Y}\cos\lambda)}{r^2} - \frac{(\cos\phi\frac{\partial \phi}{\partial Y}\cos\lambda)}{r} \tag{B.81}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial \phi}{\partial y}\right) = \frac{(\sin\phi\frac{\partial r}{\partial Y}\sin\lambda)}{r^2} - \frac{(\cos\phi\frac{\partial \phi}{\partial Y}\sin\lambda)}{r} - \frac{(\sin\phi\frac{\partial \lambda}{\partial Y}\cos\lambda)}{r} \tag{B.82}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial \phi}{\partial z}\right) = -\frac{(\cos\phi\frac{\partial r}{\partial Y})}{r^2} - \frac{(\sin\phi\frac{\partial \phi}{\partial Y})}{r} \tag{B.83}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial \lambda}{\partial x}\right) = \frac{(\frac{\partial r}{\partial Y}\sin\lambda)}{(r^2\cos\phi)} - \frac{(\sin\phi\frac{\partial \phi}{\partial Y}\sin\lambda)}{(r\cos^2\phi)} - \frac{(\frac{\partial \lambda}{\partial Y}\cos\lambda)}{(r\cos\phi)} \tag{B.84}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial \lambda}{\partial y}\right) = -\frac{(\frac{\partial \lambda}{\partial Y}\sin\lambda)}{(r\cos\phi)} - \frac{(\frac{\partial r}{\partial Y}\cos\lambda)}{(r^2\cos\phi)} + \frac{(\sin\phi\frac{\partial \phi}{\partial Y}\cos\lambda)}{(r\cos^2\phi)} \tag{B.85}$$

$$\frac{\partial}{\partial Y}\left(\frac{\partial \lambda}{\partial z}\right) = 0 \tag{B.86}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial r}{\partial x}\right) = -\cos\phi\frac{\partial \lambda}{\partial Z}\sin\lambda - \sin\phi\frac{\partial \phi}{\partial Z}\cos\lambda \tag{B.87}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial r}{\partial y}\right) = \cos\phi\frac{\partial \lambda}{\partial Z}\cos\lambda - \sin\phi\frac{\partial \phi}{\partial Z}\sin\lambda \tag{B.88}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial r}{\partial z}\right) = \cos\phi\frac{\partial \phi}{\partial Z} \tag{B.89}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial \phi}{\partial x}\right) = \frac{(\sin\phi\frac{\partial \lambda}{\partial Z}\sin\lambda)}{r} + \frac{(\sin\phi\frac{\partial r}{\partial Z}\cos\lambda)}{r^2} - \frac{(\cos\phi\frac{\partial \phi}{\partial Z}\cos\lambda)}{r} \tag{B.90}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial \phi}{\partial y}\right) = \frac{(\sin\phi\frac{\partial r}{\partial Z}\sin\lambda)}{r^2} - \frac{(\cos\phi\frac{\partial \phi}{\partial Z}\sin\lambda)}{r} - \frac{(\sin\phi\frac{\partial \lambda}{\partial Z}\cos\lambda)}{r} \tag{B.91}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial \phi}{\partial z}\right) = -\frac{(\cos\phi\frac{\partial r}{\partial Z})}{r^2} - \frac{(\sin\phi\frac{\partial \phi}{\partial Z})}{r} \tag{B.92}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial \lambda}{\partial x}\right) = \frac{(\frac{\partial r}{\partial Z}\sin\lambda)}{(r^2\cos\phi)} - \frac{(\sin\phi\frac{\partial \phi}{\partial Z}\sin\lambda)}{(r\cos^2\phi)} - \frac{(\frac{\partial \lambda}{\partial Z}\cos\lambda)}{(r\cos\phi)} \tag{B.93}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial \lambda}{\partial y}\right) = -\frac{(\frac{\partial \lambda}{\partial Z}\sin\lambda)}{(r\cos\phi)} - \frac{(\frac{\partial r}{\partial Z}\cos\lambda)}{(r^2\cos\phi)} + \frac{(\sin\phi\frac{\partial \phi}{\partial Z}\cos\lambda)}{(r\cos^2\phi)} \tag{B.94}$$

$$\frac{\partial}{\partial Z}\left(\frac{\partial \lambda}{\partial z}\right) = 0 \tag{B.95}$$

B.3 Components of Jacobian Matrix G for Zonal Gravity

Often a full spherical harmonic gravity STM is not required; the zonal terms may give enough accuracy for many applications (see Chapter 5, Section 5.1 for an example of how to reduce the number of full gravity computations using intermediate low-fidelity gravity calculations for a local Taylor Series gravity expansion model). While the zonal results are contained as a special case of the spherical harmonic

series, the zonal Jacobian is of sufficient importance that it is developed in explicit detail here to allow well-optimized codes to more efficiently capture this special case.

In this case, the G matrix in the ECEF frame is computed analogously to that given in Section B.1 for the spherical harmonic case with the only difference being that the partials are computed for the zonal acceleration $\mathbf{a}_J$ only:

$$G = \begin{bmatrix} \frac{\partial}{\partial x}(a_{J_X}) & \frac{\partial}{\partial y}(a_{J_X}) & \frac{\partial}{\partial z}(a_{J_X}) \\ \frac{\partial}{\partial x}(a_{J_Y}) & \frac{\partial}{\partial y}(a_{J_Y}) & \frac{\partial}{\partial z}(a_{J_Y}) \\ \frac{\partial}{\partial x}(a_{J_Z}) & \frac{\partial}{\partial y}(a_{J_Z}) & \frac{\partial}{\partial z}(a_{J_Z}) \end{bmatrix} \tag{B.96}$$

where each zonal accelearation $J_2 - J_6$ is used to find the acceleration, i.e.,

$$a_{J_x} = a_{J_{2_X}} + a_{J_{3_X}} + a_{J_{4_X}} + a_{J_{5_X}} + a_{J_{6_X}} \quad , \quad x \to y, z \tag{B.97}$$

and only the first five zonal terms are considered [54]:

$$J_2 = 1082.63 \times 10^{-6} \tag{B.98}$$

$$J_3 = -2.52 \times 10^{-6} \tag{B.99}$$

$$J_4 = -1.61 \times 10^{-6} \tag{B.100}$$

$$J_5 = -0.15 \times 10^{-6} \tag{B.101}$$

$$J_6 = 0.57 \times 10^{-6} \tag{B.102}$$

The zonal accelerations in terms of inertial coordinates are [54]

$$\mathbf{a}_{J2} = -\frac{3}{2}J_2\left(\frac{\mu}{r^2}\right)\left(\frac{R_e}{r}\right)^2\begin{Bmatrix}\left(1-5\left(\frac{z}{r}\right)^2\right)\frac{x}{r}\\ \left(1-5\left(\frac{z}{r}\right)^2\right)\frac{y}{r}\\ \left(3-5\left(\frac{z}{r}\right)^2\right)\frac{z}{r}\end{Bmatrix} \tag{B.103}$$

$$\mathbf{a}_{J3} = \frac{1}{2}J_3\left(\frac{\mu}{r^2}\right)\left(\frac{R_e}{r}\right)^3\begin{Bmatrix}5\left(7\left(\frac{z}{r}\right)^3-3\left(\frac{z}{r}\right)\right)\frac{x}{r}\\ 5\left(7\left(\frac{z}{r}\right)^3-3\left(\frac{z}{r}\right)\right)\frac{y}{r}\\ 3\left(1-10\left(\frac{z}{r}\right)^2+\frac{35}{3}\left(\frac{z}{r}\right)^4\right)\end{Bmatrix} \tag{B.104}$$

$$\mathbf{a}_{J4} = \frac{5}{8}J_4\left(\frac{\mu}{r^2}\right)\left(\frac{R_e}{r}\right)^4\begin{Bmatrix}\left(3-42\left(\frac{z}{r}\right)^2+63\left(\frac{z}{r}\right)^4\right)\frac{x}{r}\\ \left(3-42\left(\frac{z}{r}\right)^2+63\left(\frac{z}{r}\right)^4\right)\frac{y}{r}\\ \left(15-70\left(\frac{z}{r}\right)^2+63\left(\frac{z}{r}\right)^4\right)\frac{z}{r}\end{Bmatrix} \tag{B.105}$$

$$\mathbf{a}_{J5} = \frac{1}{8}J_5\left(\frac{\mu}{r^2}\right)\left(\frac{R_e}{r}\right)^5\begin{Bmatrix}3\left(35\left(\frac{z}{r}\right)-210\left(\frac{z}{r}\right)^3+231\left(\frac{z}{r}\right)^5\right)\frac{x}{r}\\ 3\left(35\left(\frac{z}{r}\right)-210\left(\frac{z}{r}\right)^3+231\left(\frac{z}{r}\right)^5\right)\frac{y}{r}\\ \left(693\left(\frac{z}{r}\right)^6-945\left(\frac{z}{r}\right)^4+315\left(\frac{z}{r}\right)^2-15\right)\end{Bmatrix} \tag{B.106}$$

$$\mathbf{a}_{J6} = -\frac{1}{16}J_6\left(\frac{\mu}{r^2}\right)\left(\frac{R_e}{r}\right)^6\begin{Bmatrix}\left(35-945\left(\frac{z}{r}\right)^2+3465\left(\frac{z}{r}\right)^4-3003\left(\frac{z}{r}\right)^6\right)\frac{x}{r}\\ \left(35-945\left(\frac{z}{r}\right)^2+3465\left(\frac{z}{r}\right)^4-3003\left(\frac{z}{r}\right)^6\right)\frac{y}{r}\\ \left(245-2205\left(\frac{z}{r}\right)^2+4851\left(\frac{z}{r}\right)^4-3003\left(\frac{z}{r}\right)^6\right)\frac{z}{r}\end{Bmatrix} \tag{B.107}$$

The individual Jacobian G components for the zonal harmonics case, where $\phi = \sin^{-1}\left(\frac{z}{r}\right)$, and the comments (on each line following the %) indicate which portion of the expression corresponds to which zonal term are

$$
\begin{aligned}
G_{11} = \frac{\partial}{\partial x}(a_{J_X}) &= \frac{5x^2(7z^2 + 2\sin^2\phi r^2 - r^2)(-\frac{3}{2}J_2\mu {R_e}^2)}{r^9} \quad \%J_2 \\
&- \frac{15x^2(21z^3 - 7r^2z + 7\sin^3\phi r^3 - \sin\phi r^3)(-\frac{1}{2}J_3\mu {R_e}^3)}{r^{11}} \quad \%J_3 \\
&- \frac{21x^2(33z^4 - 18r^2z^2 + 12\sin^4\phi r^4 - 4\sin^2\phi r^4 + r^4)(-\frac{5}{8}J_4\mu {R_e}^4)}{r^{13}} \quad \%J_4 \\
&- \frac{21x^2(429z^5 - 330r^2z^3 + 45r^4z + 165\sin^5\phi r^5 - 90\sin^3\phi r^5 + 5\sin\phi r^5)(-\frac{J_5}{8}\mu {R_e}^5)}{r^{15}} \quad \%J_5 \\
&+ \frac{63x^2(715z^6 - 715r^2z^4 + 165r^4z^2 + 286\sin^6\phi r^6 - 220\sin^4\phi r^6 + 30\sin^2\phi r^6 - 5r^6)(\frac{J_6}{16}\mu {R_e}^6)}{r^{17}} \quad \%J_6
\end{aligned}
\tag{B.108}
$$

$$
\begin{aligned}
G_{12} = \frac{\partial}{\partial y}(a_{J_X}) &= \frac{5xy(7z^2 + 2\sin^2\phi r^2 - r^2)(-\frac{3}{2}J_2\mu {R_e}^2)}{r^9} \quad \%J_2 \\
&- \frac{15xy(21z^3 - 7r^2z + 7\sin^3\phi r^3 - \sin\phi r^3)(-\frac{1}{2}J_3\mu {R_e}^3)}{r^{11}} \quad \%J_3 \\
&- \frac{21xy(33z^4 - 18r^2z^2 + 12\sin^4\phi r^4 - 4\sin^2\phi r^4 + r^4)(-\frac{5}{8}J_4\mu {R_e}^4)}{r^{13}} \quad \%J_4 \\
&- \frac{21xy(429z^5 - 330r^2z^3 + 45r^4z + 165\sin^5\phi r^5 - 90\sin^3\phi r^5 + 5\sin\phi r^5)(-\frac{J_5}{8}\mu {R_e}^5)}{r^{15}} \quad \%J_5 \\
&+ \frac{63xy(715z^6 - 715r^2z^4 + 165r^4z^2 + 286\sin^6\phi r^6 - 220\sin^4\phi r^6 + 30\sin^2\phi r^6 - 5r^6)(\frac{J_6}{16}\mu {R_e}^6)}{r^{17}} \quad \%J_6
\end{aligned}
\tag{B.109}
$$

$$
\begin{aligned}
G_{13} = \frac{\partial}{\partial z}(a_{J_X}) = {} & \frac{5x(7\sqrt{y^2+x^2}z^3 + 2\cos\phi\sin\phi r^2 z^2 - r^2\sqrt{y^2+x^2}z - 2\cos\phi\sin\phi r^4)(-\frac{3}{2}J_2\mu {R_e}^2)}{r^9\sqrt{y^2+x^2}} \quad \%J_2 \\
& - \frac{15xz(21z^3 - 7r^2z + 7\sin^3\phi r^3 - \sin\phi r^3)(-\frac{1}{2}J_3\mu {R_e}^3)}{r^{11}} \quad \%J_3 \\
& \frac{z}{r}\left(\frac{x\left(\frac{(84z^2)}{r^3} - \frac{(252z^4)}{r^5}\right)(-\frac{5}{8}J_4\mu {R_e}^4)}{r^7} - \frac{7x\left(\frac{(63z^4)}{r^4} - \frac{(42z^2)}{r^2} + 3\right)(-\frac{5}{8}J_4\mu {R_e}^4)}{r^8}\right) \\
& - \frac{sin\phi(252\cos\phi\sin^3\phi - 84\cos\phi\sin\phi)xz(-\frac{5}{8}J_4\mu {R_e}^4)}{\cos\phi r^9} \quad \%J_4 \\
& \frac{z}{r}\left(\frac{3x\left(\frac{(-1155z^5)}{r^6} + \frac{(630z^3)}{r^4} - \frac{(35z)}{r^2}\right)(-\frac{J_5}{8}\mu {R_e}^5)}{r^8} - \frac{24x\left(\frac{(231z^5)}{r^5} - \frac{(210z^3)}{r^3} + \frac{(35z)}{r}\right)(\frac{-J_5}{8}\mu {R_e}^5)}{r^9}\right) \\
& - \frac{3\sin\phi(1155\cos\phi\sin^4\phi - 630\cos\phi\sin^2\phi + 35\cos\phi)xz(-\frac{J_5}{8}\mu {R_e}^5)}{\cos\phi r^{10}} \quad \%J_5 \\
& \frac{z}{r}\left(\frac{x\left(\frac{(18018z^6)}{r^7} - \frac{(13860z^4)}{r^5} + \frac{(1890z^2)}{r^3}\right)(\frac{J_6}{16}\mu {R_e}^6)}{r^9}\right. \\
& \left. - \frac{9x\left(\frac{(-3003z^6)}{r^6} + \frac{(3465z^4)}{r^4} - \frac{(945z^2)}{r^2} + 35\right)(\frac{J_6}{16}\mu {R_e}^6)}{r^{10}}\right) \\
& - \frac{\sin\phi(-18018\cos\phi\sin^5\phi + 13860\cos\phi\sin^3\phi - 1890\cos\phi\sin\phi)xz(\frac{J_6}{16}\mu {R_e}^6)}{\cos\phi r^{11}} \quad \%J_6
\end{aligned}
\tag{B.110}
$$

$$
\begin{aligned}
G_{21} = \frac{\partial}{\partial x}(a_{J_Y}) = {} & \frac{5xy(7z^2 + 2\sin\phi^2 r^2 - r^2)(\frac{-3}{2}J_2\mu {R_e}^2)}{r^9} \quad \%J_2 \\
& - \frac{15xy(21z^3 - 7r^2 z + 7\sin\phi^3 r^3 - \sin\phi r^3)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^{11}} \quad \%J_3 \\
& - \frac{21xy(33z^4 - 18r^2 z^2 + 12\sin\phi^4 r^4 - 4\sin\phi^2 r^4 + r^4)(\frac{-5}{8}J_4\mu {R_e}^4)}{r^{13}} \quad \%J_4 \\
& - \frac{21xy(429z^5 - 330r^2 z^3 + 45r^4 z + 165\sin\phi^5 r^5 - 90\sin\phi^3 r^5 + 5\sin\phi r^5)(\frac{-J_5}{8}\mu {R_e}^5)}{r^{15}} \quad \%J_5 \\
& + \frac{63x(715yz^6 - 715r^2 yz^4 + 165r^4 yz^2 - 5r^6 y + 286\sin\phi^6 r^6 x - 220\sin\phi^4 r^6 x + 30\sin\phi^2 r^6 x)(\frac{J_6}{16}\mu {R_e}^6)}{r^{17}} \quad \%J_6
\end{aligned}
\tag{B.111}
$$

$$
\begin{aligned}
G_{22} = \frac{\partial}{\partial y}(a_{J_Y}) = {} & \frac{5y^2(7z^2 + 2\sin\phi^2 r^2 - r^2)(\frac{-3}{2}J_2\mu {R_e}^2)}{r^9} \quad \%J_2 \\
& - \frac{15y^2(21z^3 - 7r^2 z + 7\sin\phi^3 r^3 - \sin\phi r^3)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^{11}} \quad \%J_3 \\
& - \frac{21y^2(33z^4 - 18r^2 z^2 + 12\sin\phi^4 r^4 - 4\sin\phi^2 r^4 + r^4)(\frac{-5}{8}J_4\mu {R_e}^4)}{r^{13}} \quad \%J_4 \\
& - \frac{21y^2(429z^5 - 330r^2 z^3 + 45r^4 z + 165\sin\phi^5 r^5 - 90\sin\phi^3 r^5 + 5\sin\phi r^5)(\frac{-J_5}{8}\mu {R_e}^5)}{r^{15}} \quad \%J_5 \\
& \frac{(45045y^2 z^6)(\frac{J_6}{16}\mu {R_e}^6)}{r^{17}} - \frac{(45045y^2 z^4)(\frac{J_6}{16}\mu {R_e}^6)}{r^{15}} + \frac{(10395y^2 z^2)(\frac{J_6}{16}\mu {R_e}^6)}{r^{13}} - \frac{(315y^2)(\frac{J_6}{16}\mu {R_e}^6)}{r^{11}} \\
& + \frac{(18018\sin\phi^6 xy)(\frac{J_6}{16}\mu {R_e}^6)}{r^{11}} - \frac{(13860\sin\phi^4 xy)(\frac{J_6}{16}\mu {R_e}^6)}{r^{11}} + \frac{(1890\sin\phi^2 xy)(\frac{J_6}{16}\mu {R_e}^6)}{r^{11}} \quad \%J_6
\end{aligned}
\tag{B.112}
$$

$$
\begin{aligned}
G_{23} = \frac{\partial}{\partial z}(a_{J_Y}) &= \frac{5y(7\sqrt{y^2+x^2}z^3 + 2\cos\phi\sin\phi r^2 z^2 - r^2\sqrt{y^2+x^2}z - 2\cos\phi\sin\phi r^4)(\frac{-3}{2}J_2\mu {R_e}^2)}{r^9\sqrt{y^2+x^2}} \quad \%J_2 \\
&- \frac{15yz(21z^3 - 7r^2z + 7\sin^3\phi r^3 - \sin\phi r^3)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^{11}} \quad \%J_3 \\
&\frac{z}{r}\left(\frac{y\left(\frac{(84z^2)}{r^3} - \frac{(252z^4)}{r^5}\right)(\frac{-5}{8}J_4\mu {R_e}^4)}{r^7} - \frac{7y\left(\frac{(63z^4)}{r^4} - \frac{(42z^2)}{r^2} + 3\right)(\frac{-5}{8}J_4\mu {R_e}^4)}{r^8}\right) \\
&- \frac{\sin\phi(252\cos\phi\sin^3\phi - 84\cos\phi\sin\phi)yz(\frac{-5}{8}J_4\mu {R_e}^4)}{cos\phi r^9} \quad \%J_4 \\
&- \frac{21yz(429z^5 - 330r^2z^3 + 45r^4z + 165\sin^5\phi r^5 - 90\sin^3\phi r^5 + 5\sin\phi r^5)(\frac{-J_5}{8}\mu {R_e}^5)}{r^{15}} \quad \%J_5 \\
&\frac{z}{r}\left(\frac{y\left(\frac{(18018z^6)}{r^7} - \frac{(13860z^4)}{r^5} + \frac{(1890z^2)}{r^3}\right)(\frac{J_6}{16}\mu {R_e}^6)}{r^9} - \frac{9y\left(-\frac{(3003z^6)}{r^6} + \frac{(3465z^4)}{r^4} - \frac{(945z^2)}{r^2} + 35\right)(\frac{J_6}{16}\mu {R_e}^6)}{r^{10}}\right) \\
&- \frac{\sin\phi(-18018\cos\phi\sin^5\phi + 13860\cos\phi\sin^3\phi - 1890\cos\phi\sin\phi)xz(\frac{J_6}{16}\mu {R_e}^6)}{\cos\phi r^{11}} \quad \%J_6
\end{aligned}
\tag{B.113}
$$

$$
\begin{aligned}
G_{31} = \frac{\partial}{\partial x}(a_{J_Z}) &= \frac{5xz(7z^2 + 2\sin^2\phi r^2 - 3r^2)(\frac{-3}{2}J_2\mu {R_e}^2)}{r^9} \quad \%J_2 \\
&+ \frac{5x(63z^4 - 42r^2z^2 + 28\sin^4\phi r^4 - 12\sin^2\phi r^4 + 3r^4)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^{11}} \quad \%J_3 \\
&+ \frac{7xz(99z^4 - 90r^2z^2 + 36\sin^4\phi r^4 - 20\sin^2\phi r^4 + 15r^4)(\frac{-5}{8}J_4\mu {R_e}^4)}{r^{13}} \quad \%J_4 \\
&+ \frac{21x(429z^6 - 495r^2z^4 + 135r^4z^2 + 198\sin^6\phi r^6 - 180\sin^4\phi r^6 + 30\sin^2\phi r^6 - 5r^6)(\frac{-J_5}{8}\mu {R_e}^5)}{r^{15}} \quad \%J_5 \\
&- \frac{63xz(715z^6 - 1001r^2z^4 + 385r^4z^2 + 286\sin^6\phi r^6 - 308\sin^4\phi r^6 + 70\sin^2\phi r^6 - 45r^6)(\frac{J_6}{16}\mu {R_e}^6)}{r^{17}} \quad \%J_6
\end{aligned}
\tag{B.114}
$$

$$
\begin{aligned}
G_{32} = \frac{\partial}{\partial y}(a_{J_Z}) &= \frac{5yz(7z^2 + 2\sin^2\phi r^2 - 3r^2)(\frac{-3}{2}J_2\mu {R_e}^2)}{r^9} \quad \%J_2 \\
&+ \frac{(315yz^4)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^{11}} - \frac{(210yz^2)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^9} + \frac{(140\sin^4\phi y)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^7} - \frac{(60\sin^2\phi y)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^7} \\
&+ \frac{(15y)(\frac{-1}{2}J_3\mu {R_e}^3)}{r^7} \quad \%J_3 + \frac{7yz(99z^4 - 90r^2z^2 + 36\sin^4\phi r^4 - 20\sin^2\phi r^4 + 15r^4)(\frac{-5}{8}J_4\mu {R_e}^4)}{r^{13}} \quad \%J_4 \\
&+ \frac{21y(429z^6 - 495r^2z^4 + 135r^4z^2 + 198\sin^6\phi r^6 - 180\sin^4\phi r^6 + 30\sin^2\phi r^6 - 5r^6)(\frac{-J_5}{8}\mu {R_e}^5)}{r^{15}} \quad \%J_5 \\
&- \frac{63yz(715z^6 - 1001r^2z^4 + 385r^4z^2 + 286\sin^6\phi r^6 - 308\sin^4\phi r^6 + 70\sin^2\phi r^6 - 45r^6)(\frac{J_6}{16}\mu {R_e}^6)}{r^{17}} \quad \%J_6
\end{aligned}
\tag{B.115}
$$

$$
\begin{aligned}
G_{33} = \frac{\partial}{\partial z}(a_{J_Z}) &= \frac{5z(7\sqrt{y^2+x^2}z^3 + 2\cos\phi\sin\phi r^2 z^2 - 3r^2\sqrt{y^2+x^2}z - 2\cos\phi\sin\phi r^4)(\frac{-3}{2}\mu R_e^2 J_2)}{r^9\sqrt{y^2+x^2}} \quad \%_0 J_2 \\
&\frac{z}{r}\left(\frac{3\left(\frac{(140z^4)}{(3r^5)} - \frac{(20z^2)}{r^3}\right)(\frac{-1}{2}\mu R_e^3 J_3)}{r^5} - \frac{15\left(\frac{(-35z^4)}{(3r^4)} + \frac{(10z^2)}{r^2} - 1\right)(\frac{-1}{2}\mu R_e^3 J_3)}{r^6}\right) \\
&- \frac{3\sin\phi\left(20\cos\phi\sin\phi - \frac{1}{3}(140\cos\phi\sin^3\phi)\right)z(\frac{-1}{2}\mu R_e^3 J_3)}{\cos\phi r^7} \quad \%_0 J_3 \\
&\frac{z}{r}\left(\frac{z\left(\frac{(252z^4)}{r^5} - \frac{(140z^2)}{r^3}\right)(\frac{-5}{8}\mu R_e^4 J_4)}{r^7} - \frac{7z\left(\frac{(-63z^4)}{r^4} + \frac{(70z^2)}{r^2} - 15\right)(\frac{-5}{8}\mu R_e^4 J_4)}{r^8})\right) \\
&- \frac{\sin\phi(140\cos\phi\sin\phi - 252\cos\phi\sin^3\phi)z^2(\frac{-5}{8}\mu R_e^4 J_4)}{\cos\phi r^9} \quad \%_0 J_4 \\
&\frac{z}{r}\left(\frac{\left(\frac{(4158z^6)}{r^7} - \frac{(3780z^4)}{r^5} + \frac{(630z^2)}{r^3}\right)(\frac{-1}{8}\mu R_e^5 J_5)}{r^7} - \frac{7\left(-\frac{(693z^6)}{r^6} + \frac{(945z^4)}{r^4} - \frac{(315z^2)}{r^2} + 15\right)(\frac{-1}{8}\mu R_e^5 J_5)}{r^8}\right) \\
&- \frac{\sin\phi(-4158\cos\phi\sin^5\phi + 3780\cos\phi\sin^3\phi - 630\cos\phi\sin\phi)z(\frac{-1}{8}\mu R_e^5 J_5)}{\cos\phi r^9} \quad \%_0 J_5 \\
&\frac{z}{r}\left(\frac{z\left(\frac{(-18018z^6)}{r^7} + \frac{(19404z^4)}{r^5} - \frac{(4410z^2)}{r^3}\right)(\frac{1}{16}\mu R_e^6 J_6)}{r^9} - \frac{9z\left(\frac{(3003z^6)}{r^6} - \frac{(4851z^4)}{r^4} + \frac{(2205z^2)}{r^2} - 315\right)(\frac{1}{16}\mu R_e^6 J_6)}{r^{10}}\right) \\
&- \frac{\sin\phi(18018\cos\phi\sin^5\phi - 19404\cos\phi\sin^3\phi + 4410\cos\phi\sin\phi)z^2(\frac{1}{16}\mu R_e^6 J_6)}{\cos\phi r^{11}} \quad \%_0 J_6
\end{aligned}
\tag{B.116}
$$

B.4 Jacobian Matrix G for Two-Body Gravity

This Jacobian may be used for low-fidelity models (for example, the two-body STM may be used in a shooting method [66]). The two-body gravity model provides a simple expression for the G matrix:

$$G = \frac{\mu}{r^5}\begin{bmatrix} 3x^2 - r^2 & 3xy & 3xz \\ 3xy & 3y^2 - r^2 & 3yz \\ 3xz & 3yz & 3z^2 - r^2 \end{bmatrix} \tag{B.117}$$

B.5 Partials of Associated Legendre Functions

Computation of the STM for the spherical harmonic gravity case (see Chapter 4, Section 4.3 and also Sections B.1 and B.2 of this appendix) requires the partial derivative of P_{nm} with respect to ϕ, where C_{nm} and S_{nm} are the normalized Stokes coefficients determined from satellite motion observations, and N_{nm} is a scale factor. The first and second partials of the ALFs are incorporated in the calculations for the partials of the gravity potential, U, through the computation of the corresponding ALFs and the appropriate scale factors [52]. The following known relationships are used for these derivations [34]:

$$\frac{\partial}{\partial u}A_{nm}(u) = A_{n,m+1}(u) \tag{B.118}$$

$$\frac{\partial^2}{\partial u^2}A_{nm}(u) = A_{n,m+2}(u) \tag{B.119}$$

$$\frac{\partial}{\partial \phi}A_{nm}(\sin\phi) = \frac{\partial A_{nm}(\sin\phi)}{\partial \sin\phi}\frac{\partial \sin\phi}{\partial \phi} = \frac{\partial A_{nm}(\sin\phi)}{\partial \sin\phi}\cos\phi \tag{B.120}$$

The derived ALFs are related to the conventional Legendre functions in terms of latitute ϕ through [34]:

$$P_{nm}(\sin\phi) = \cos^m\phi A_{nm}(\sin\phi) \tag{B.121}$$

Starting with the above equation and using the product rule,

$$\frac{\partial P_{nm}(\sin\phi)}{\partial\phi} = \cos^m\phi\frac{\partial A_{nm}(\sin\phi)}{\partial\phi} - m\sin\phi\cos^{m-1}\phi A_{nm}(\sin\phi) \tag{B.122}$$

This equation is rewritten using Eq. (B.121) as

$$\begin{aligned}\frac{\partial P_{nm}(\sin\phi)}{\partial\phi} &= \cos^m\phi\frac{\partial A_{nm}(\sin\phi)}{\partial\phi} - m\tan\phi\cos^m\phi A_{nm}(\sin\phi)\\ &= \cos^m\phi\frac{\partial A_{nm}(\sin\phi)}{\partial\phi} - m\tan\phi P_{nm}(\sin\phi)\end{aligned} \tag{B.123}$$

Using Equations (B.118) and (B.121), the final equation becomes

$$\frac{\partial P_{nm}(\sin\phi)}{\partial\phi} = P_{n,m+1}(\sin\phi) - m\tan\phi P_{nm}(\sin\phi) \tag{B.124}$$

Similarly, an expression for $\frac{\partial^2 P_{nm}}{\partial\phi^2}$ is derived.

$$\frac{\partial^2 P_{nm}(\sin\phi)}{\partial\phi^2} = \frac{\partial P_{n,m+1}(\sin\phi)}{\partial\phi} - m\sec^2\phi P_{nm}(\sin\phi) - m\tan\phi\frac{\partial P_{nm}(\sin\phi)}{\partial\phi} \tag{B.125}$$

Substituting Eq. (B.124) into this expression gives

$$\frac{\partial^2 P_{nm}(\sin\phi)}{\partial\phi^2} = \frac{\partial P_{n,m+1}(\sin\phi)}{\partial\phi} - m\sec^2\phi P_{nm}(\sin\phi) - m\tan\phi\big[P_{n,m+1}(\sin\phi) - m\tan\phi P_{nm}(\sin\phi)\big] \tag{B.126}$$

Eq. (B.124) is used to write

$$\frac{\partial P_{n,m+1}(\sin\phi)}{\partial\phi} = P_{n,m+2}(\sin\phi) - (m+1)\tan\phi P_{n,m+1}(\sin\phi) \tag{B.127}$$

Therefore, the final expression for the second partial is

$$\begin{aligned}\frac{\partial^2 P_{nm}(\sin\phi)}{\partial\phi^2} &= P_{n,m+2}(\sin\phi) - (2m+1)\tan\phi P_{n,m+1}(\sin\phi) \\ &\quad + m(m\tan^2\phi - \sec^2\phi)P_{nm}(\sin\phi)\end{aligned} \tag{B.128}$$

In the code, the term $P_{n,m+1}(\sin\phi)$ is multiplied by the scale factor F_{nm}, and the term $P_{n,m+2}(\sin\phi)$ is multiplied by the scale factor $F_{n,m+1}$ to compensate for the original normalization of the ALFs (see Chapter 4, Section 4.3). The final, normalized equations are then

$$\frac{\partial P_{nm}(\sin\phi)}{\partial\phi} = P_{n,m+1}(\sin\phi)F_{nm} - m\tan\phi P_{nm}(\sin\phi) \tag{B.129}$$

$$\begin{aligned}\frac{\partial^2 P_{nm}(\sin\phi)}{\partial\phi^2} &= P_{n,m+2}(\sin\phi)F_{n,m+1} - (2m+1)\tan\phi P_{n,m+1}(\sin\phi)F_{nm} \\ &\quad + m(m\tan^2\phi - \sec^2\phi)P_{nm}(\sin\phi)\end{aligned} \tag{B.130}$$

APPENDIX C

ENERGY JACOBI INTEGRAL

Jacobi's integral is simply the classical energy integral, expressed in rotating coordinates and may be used to indicate the accuracy level of a numerical solution [54]. For conservative forces (i.e., gravity), it should remain constant over time. However, for nonconservative forces (i.e., drag or solar radiation pressure), this integral does not remain constant and a different method must be used to verify a trajectory's solution such as round-trip closure. The Jacobi integral is given here for the different cases used to verify problem solutions throughout this dissertation; for the two-body case it is known as the Hamiltonian and may be developed analogously to the Jacobi integral for the restricted three-body problem as given in [68].

C.1 Jacobi Integral for Zonal Harmonic Gravity

In this case, Jacobi's integral is

$$T + V = const \tag{C.1}$$

For Earth, the zonal harmonics are given by

$$J_2 = 1082.63 \times 10^{-6} \tag{C.2}$$

$$J_3 = -2.52 \times 10^{-6} \tag{C.3}$$

$$J_4 = -1.61 \times 10^{-6} \tag{C.4}$$

$$J_5 = -0.15 \times 10^{-6} \tag{C.5}$$

$$J_6 = -0.57 \times 10^{-6} \tag{C.6}$$

The kinetic energy for two-body motion perturbed with zonal harmonics $J_2 - J_6$ is simply a function of the velocity:

$$T = \frac{1}{2}v^2 \tag{C.7}$$

The Potential energy is derived from the gravity potential for the two-body expression plus the first six harmonics [54]:

$$V = -\frac{\mu}{r} + \frac{J_2}{2}\frac{\mu}{r}\left(\frac{R_e}{r}\right)^2\left(3\sin^2\phi - 1\right) \tag{C.8}$$

$$+ \frac{J_3}{2}\frac{\mu}{r}\left(\frac{R_e}{r}\right)^3\left(5\sin^3\phi - 3\sin\phi\right) \tag{C.9}$$

$$+ \frac{J_4}{8}\frac{\mu}{r}\left(\frac{R_e}{r}\right)^4\left(35\sin^4\phi - 30\sin^2\phi + 3\right) \tag{C.10}$$

$$+ \frac{J_5}{8}\frac{\mu}{r}\left(\frac{R_e}{r}\right)^5\left(63\sin^5\phi - 70\sin^3\phi + 15\sin\phi\right) \tag{C.11}$$

$$+ \frac{J_6}{16}\frac{\mu}{r}\left(\frac{R_e}{r}\right)^6\left(231\sin^6\phi - 315\sin^4\phi + 105\sin^2\phi - 5\right) \tag{C.12}$$

where μ is the gravitational constant, R_e is the Earth's radius, r is the radius of the spacecraft, and ϕ is the third angle expressing spherical coordinates (r, θ, ϕ) using the right-hand rule. Notice that the longitude λ appears because this potential corresponds to a body of revolution about the z-axis. Thus, the difference between

the between the ECI and the ECEF frames do not need to be distinguished when taking the gradient of the "zonal only" potential function.

C.2 Jacobi Integral for Spherical Harmonic Gravity

For spherical harmonic gravity, Jacobi's integral is

$$T + V + A = const \tag{C.13}$$

The kinetic energy for two-body motion perturbed with an arbitrary order and degree spherical harmonic gravity is simply a function of the velocity:

$$T = \frac{1}{2}v^2 \tag{C.14}$$

The spherical harmonic gravity potential is expressed in Earth-fixed coordinates as (see Chapter 4 for more details):

$$V = U(r, \phi, \lambda) = \frac{\mu}{r}\left[1 + \sum_{n=2}^{\infty}\sum_{m=0}^{n}\left(\frac{R_e}{r}\right)^n P_{nm}(\sin\phi)\left[C_{nm}\cos(m\lambda) + S_{nm}\sin(m\lambda)\right]\right] \tag{C.15}$$

The additional term, A, is computed as a function of the Earth's rotation $\omega = 7.2921 \times 10^{-5}$ rad/s, and the Earth-fixed x and y components of the position vector.

$$A = -\frac{1}{2}\omega^2(r_x^2 + r_y^2) \tag{C.16}$$

APPENDIX D

CANONICAL UNITS FOR CARTESIAN COORDINATES

Canonical units are used for the p-iteration and MPS shooting methods as described in Chapter 6, as well as for a suboptimal control formulation as described in Chapter 7, though they could be used in practically any simulation given in this dissertation. Additionally, the conversion to/from canonical units for the State Transition Matrix (STM) corresponding with Cartesian coordinates is given in Chapter 4, Section 4.7.2.

Using canonical (non-dimensional) coordinates normalizes values associated with a reference orbit so that the number of significant figures is relatively the same. In many cases, for instance, using metric units leads to position terms that are several orders of magnitude larger than velocity terms. These non-dimensional units are mainly used to "bring order" to the equations and make both position and velocity coordinates be of order 1 in the vicinity of the reference orbit. In addition, using canonical units means that the values are much smaller than when using physical units, which allows for more significant figures to the right of the decimal point such that a digital computer (with a finite amount of storage space) can give a more precise solution than is possible using larger metric units.

To convert to canonical units, distances and time are scaled with respect to reference distances and time scales that are more appropirate for celestial motion. The choice of these scaling factors depends upon traditions associated with the central celestial body as the "primary" in the particular discussion, as described in the next few sections [54].

Consider the perturbed two-body problem,

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{a}_d \tag{D.1}$$

As shown in Chapter 4, Section 4.7.2, if a non-dimensional set of coordinates are introduced by defining

$$\vec{\mathcal{R}} \equiv \frac{\mathbf{r}}{r_\oplus} \tag{D.2}$$

and

$$|\vec{\mathcal{R}}| = \mathcal{R} \equiv \frac{r}{r_\oplus} \tag{D.3}$$

then the non-dimensional two-body equations of motion can be written as

$$\frac{d^2\vec{\mathcal{R}}}{d\tau^2} = -\frac{1}{\mathcal{R}^3}\vec{\mathcal{R}} + \mathbf{A}_d \tag{D.4}$$

with

$$\tau = \sqrt{\frac{\mu_\oplus}{r_\oplus^3}}t \tag{D.5}$$

$$d\tau = \sqrt{\frac{\mu_\oplus}{r_\oplus^3}}dt \tag{D.6}$$

D.1 Earth-Centered Motion

When a spacecraft is orbiting the Earth, or any solar body other than the Sun, a distance unit DU is used to normalize motion variables. This DU is typically equal to the planet's equatorial distance r_{eq}. For the Earth,

$$1DU_{\oplus} = 6378.14km \tag{D.7}$$

The reference velocity is the circular orbit speed at 1 DU, and is the planet's critical speed, again for Earth:

$$v_{ref} = 1\frac{DU_{\oplus}}{TU_{\oplus}} = 7.9054km/s \tag{D.8}$$

The Earth's TU measurement in metric units is

$$TU = \frac{1DU_{\oplus}}{v_{ref}} \approx 806.8117s \tag{D.9}$$

The gravitational constant for the Earth is then set to unity:

$$\mu_{\oplus} = 1\frac{DU_{\oplus}^3}{TU_{\oplus}^2} \tag{D.10}$$

D.2 Heliocentric Motion

For heliocentric motion, the reference distance used is an astronautical unit, or AU. This distance is approximately the semimajor axis of the Earth's approximately circular orbit about the Sun.

$$1AU = 149,597,870.691 \quad km \tag{D.11}$$

Normalizing all distances by 1 AU relates all distances to the Earth/sun distance, allowing for more intuitive planetary motion simulations.

Next, the canonical time unit TU is chosen such that the velocity in a circular orbit with a radius equal to 1 AU assumes a value of 1. In other words, during 1 TU an object moves through one radian along a circular orbit with a radius of 1 AU.

This means that

$$v_{ref} = 1\frac{AU}{TU_{\odot}} \approx 29.785km/s \tag{D.12}$$

or

$$TU_{\odot} = \frac{1AU}{v_{ref}} = 5.02264 \times 10^{6}s \tag{D.13}$$

For the heliocentric case, the gravitational constant of the sun is unity as well:

$$\mu_{\odot} = 1\frac{AU^3}{TU_{\odot}^2} \tag{D.14}$$

D.3 Conversion Script

To convert from cartesian to canonical units, the following computations are performed, where the user inputs the values for μ and DU for the given position $\mathbf{r}$, velocity $\mathbf{v}$, acceleration $\mathbf{a}$, and vector of time steps t.

$$TU = \sqrt{\frac{DU^3}{\mu}} \tag{D.15}$$

$$\mathbf{pos} = \frac{\mathbf{r}}{DU} \tag{D.16}$$

$$\mathbf{vel} = \frac{\mathbf{v}}{DU}TU \tag{D.17}$$

$$\mathbf{accel} = \frac{\mathbf{a}}{DU}TU^2 \tag{D.18}$$

$$T = \frac{t}{TU} \tag{D.19}$$

Similarly, to convert from canonical to cartesian units, the following computations are performed.

$$TU = \sqrt{\frac{DU^3}{\mu}} \tag{D.20}$$

$$\mathbf{r} = \mathbf{pos}DU \tag{D.21}$$

$$\mathbf{v} = \mathbf{vel}\frac{DU}{TU} \tag{D.22}$$

$$\mathbf{a} = \mathbf{accel}\frac{DU}{TU^2} \tag{D.23}$$

$$T = tTU \tag{D.24}$$

APPENDIX E

COMPUTE CLUSTER AT THE LAND, AIR, AND SPACE ROBOTICS LABORATORY (LASR) FOR SPACE SITUATIONAL AWARENESS (SSA)

The studies performed during this dissertation are or may be implemented on a compute cluster using parallel computation. This compute cluster has one front-end machine with 15 compute nodes. Each node has 64 GB RAM and 24 processor cores. The front-end node provides the user a connection to the cluster and also starts/monitors jobs. The jobs are run on the compute nodes and are managed by Sun Grid Engine (SGE), which is a resource management software that allows cluster resources to be used effectively.

The full specifications for the compute cluster are as follows [36]. The ideal computational capability of the LASR SSA Compute Cluster is 2 TFLOPS.

Head Node:

- 2x Intel Xeon® Processor E5-2630 v2
  - 6 Cores
  - 2.6 GHz
  - 3.1 GHz Turbo Boost
- 64 GB DDR3 1600 Memory
- 500 GB Raid 5 Hard Drive
- 6x Gigabit Ethernet

- Redundant Power Supply with 3000VA UPS
- PCI Express 2.0 expansion slots for accelerator processors (GPUs)

Compute Nodes (x15):

- 2x Intel Xeon® Processor E5-2630 v2
  - 6 Cores
  - 2.6 GHz
  - 3.1 GHz Turbo Boost
- 64 GB DDR3 1600 Memory
- 500 GB Hard Drive
- 6x Gigabit Ethernet
- PCI Express 2.0 expansion slots for accelerator processors (GPUs)

Software:

- Centos 6.5 Operating System
- MPICH 3 Message Passing Interface
- Slurm (Simple Linux Utility for Resource Management)
- GNU C/C++/Fortran compilers
- Locally hosted yum package manager repository