REGULARIZATION AND COMPUTATIONAL METHODS FOR PRECISE SOLUTION OF PERTURBED ORBIT TRANSFER PROBLEMS

A Dissertation

by

ROBYN MICHELE WOOLLANDS

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
Committee Members,	John E. Hurtado
	Ahmad Bani Younes
	Shankar Bhattacharyya
Head of Department,	Rodney Bowersox

December 2016

Major Subject: Aerospace Engineering

ABSTRACT

The author has developed a suite of algorithms for solving the perturbed Lambert's problem in celestial mechanics. These algorithms have been implemented as a parallel computation tool that has broad applicability. This tool is composed of four component algorithms and each provides unique benefits for solving a particular type of orbit transfer problem. The first one utilizes a Keplerian solver (a-iteration) for solving the unperturbed Lambert's problem. This algorithm not only provides a "warm start" for solving the perturbed problem but is also used to identify which of several perturbed solvers is best suited for the job. The second algorithm solves the perturbed Lambert's problem using a variant of the modified Chebyshev-Picard iteration initial value solver that solves two-point boundary value problems. This method converges over about one third of an orbit and does not require a Newton-type shooting method and thus no state transition matrix needs to be computed. The third algorithm makes use of regularization of the differential equations through the Kustaanheimo-Stiefel transformation and extends the domain of convergence over which the modified Chebyshev-Picard iteration two-point boundary value solver will converge, from about one third of an orbit to almost a full orbit. This algorithm also does not require a Newton-type shooting method. The fourth algorithm uses the method of particular solutions and the modified Chebyshev-Picard iteration initial value solver to solve the perturbed two-impulse Lambert problem over multiple revolutions. The method of particular solutions is a shooting method but differs from the Newton-type shooting methods in that it does not require integration of the state transition matrix. The mathematical developments that underlie these four algorithms are derived in the chapters of this dissertation. For each of the algorithms, some orbit transfer test cases are included to provide insight on accuracy and efficiency of these individual algorithms. Following this discussion, the combined parallel algorithm, known as the unified Lambert tool, is presented and an explanation is given as to how it automatically selects which of the three perturbed solvers to compute the perturbed solution for a particular orbit transfer. The unified Lambert tool may be used to determine a single orbit transfer or for generating of an extremal field map. A case study is presented for a mission that is required to rendezvous with two pieces of orbit debris (spent rocket boosters). The unified Lambert tool software developed in this dissertation is already being utilized by several industrial partners and we are confident that it will play a significant role in practical applications, including solution of Lambert problems that arise in the current applications focused on enhanced space situational awareness.

DEDICATION

To my parents,

and to all the teachers in my life.

ACKNOWLEDGEMENTS

"The heavens declare the glory of God; the skies proclaim the work of his hands. Day after day they pour forth speech, night after night they reveal knowledge." Psalm 19

It is indeed a privilege to look up to the heavens and gaze in wonder at the intricate workings of the Universe. As a child I yearned to learn about "how things move in space", or what is scientifically known as celestial mechanics. This dream came true when I received an offer from Distinguished Professor John L. Junkins to work on a PhD under his supervision. Dr. Junkins is arguably the most well-known and well-respected researcher in the field, and I knew it was nothing short of a miracle that he had chosen me to be his student.

Working with Dr. Junkins for the last four years has been a wonderful, rewarding and life altering experience. I have been spoiled to enjoy daily conversations with an incredible person who has four decades more experience and wisdom than me. He has been an excellent mentor, teacher, and friend. I cherish all of his advice, his "Junkins War Stories", and his jokes. Dr. Junkins once said, "If getting a PhD was easy, everyone would have one." We have certainly worked hard to find solutions to some challenging and at times daunting problems in Celestial Mechanics. Climbing this PhD mountain was an enormous feat, but it was comforting to know that Dr. Junkins was always there to provide encouragement and reassurance of his "total faith in you" for finding a solution. I have succeeded because each step of the way I have been standing on the shoulders of a giant, Giant Junkins.

Two other mentors also played a significant role in my success: Dr. Ahmad Bani Younes and Dr. John Hurtado. Ahmad was my officemate and a senior PhD student during my first year. He always made time to answer my questions, and I would not be the MCPI expert I am today without the many discussions that we had during that first year. I am truly grateful for his help and his friendship. Dr. Hurtado is another person who invested a significant amount of time in me. I am grateful for our many discussions about dynamics and also for the opportunity to work with him teaching a graduate dynamics class.

During my time as a PhD student I spent two summers interning at NASA's Jet Propulsion Lab. A special thanks to Dr. Martin Lo (JPL adviser) and Dr. Fred Hadaegh for creating these opportunities for me. I am very grateful for the valuable experience I gained from working at JPL.

These acknowledgments would be incomplete without thanking some of my colleagues and dear friends: Julie Read, Kevin Hernandez, Austin Probe, Brent Macomber, Tarek Elgohary, Donghoon Kim, Abhay Masher, Xiaoli Bai, Nathan Budd, Chris Shelton, Vinicius Goecks, Reza Karimi, Travis Swenson, Brian Andersen, Mouath Rashdan, Tim Roorda, Dylan Conway, Bharat Mahajan and Karen Davis. I am grateful for our collaboration, discussion and friendship over the last four years.

Finally, I thank my beloved parents, Robert and Michèle, and my sister Andrea. I would never have made it this far without your continued support, love and prayers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Lambert's problem is the classical two-point boundary value problem (TPBVP) in celestial mechanics that was first posed and solved by Johann Heinrich Lambert in 1761. It is known to have a unique solution for the fractional orbit transfer between prescribed positions with a prescribed "time of flight". Solving this problem requires determining the orbital arc (typically, solving for the initial velocity) connecting prescribed initial and final position vectors, which correspond to the specified flight time. In the modern literature, Richard Battin [1] developed an immortal and the most widely used and general algorithm for solving the unperturbed Lambert problem (Keplerian motion). His universal algorithm generates not only the unique solution for the fractional orbit case, but also admits the multiple solutions associated with multiple revolution orbit transfers and admits hyperbolic orbit arcs.

The most common solution approach for generalizing the Lambert problem to include perturbations is to utilize the state transition matrix sensitivity of the final state with respect to the initial velocity, and iterate via Newton's method on the three components of initial velocity to "hit" the final desired position at the prescribed final time. The unperturbed Lambert solution can be used as an efficiently computable "warm start" to solve the perturbed problem.

One motivation for this research is to respond to the various challenges in Space Situational Awareness (SSA) with a difficult "data association" problem. Short tracks of many newly observed objects, widely separated in time, must be processed to determine orbits and correlate the observations of tracked objects, if possible, with each other and with existing space object data bases. For the case of radar measurements, a full position vector can be formed from measured range vectors at two measured epochs. This naturally gives rise to a Lambert-type problem statement to connect any pair of measurements, conjectured to be the same object, with an orbit arc consistent with the equations of motion.

In the current state of the practice, hundreds of thousands of hypotheses must frequently be tested to find feasible preliminary orbits connecting time-displaced short tracks of unknown space objects. These preliminary orbits and the underlying data associations are taken as the starting conjecture and orbit estimates for further correlation. "Short" tracks may be separated by up to several orbits, so ignoring the effects of perturbations will typically introduce residual errors much larger than the measurement errors, which can obviously corrupt the data association process. In the current state of practice for processes involving radar measurements, data association hypotheses are tested for preliminary orbit estimation using the Keplerian Lambert solutions for sufficiently short arcs, but higher force model precision is needed to accommodate hypothesis testing over longer time intervals where neglecting perturbations can lead to larger propagation errors than measurement errors. It is desirable to have a general and efficient Lambert algorithm that permits a state of the art force model. When more than several hundred thousand hypotheses are tested daily (including perturbations), the computational cost can exceed many CPU days per month. The anticipation of a new radar space fence giving an order of magnitude increase to $\sim 200,000$ more presently un-trackable debris objects (visible to our sensors) means that already high computational costs are about to dramatically increase [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15].

Testing millions of hypotheses will be required to solve the data association problem. Also, "all-on-all" conjunction analysis and probability of collisions will be extremely difficult using existing orbit propagation tools. Thus the issue of finding an optimal solution to a generally perturbed TPBVP lies near the heart of computational challenges in SSA. The inclusion of perturbations in Lambert's problem and the development of more efficient and robust methods are therefore of strong interest to advance SSA.

In addition to the data association problem, which deals with tracking, there is also the problem of debris removal that must be considered. The satellite collision of Iridium and Kosmos, in 2009, demonstrated the seriousness of the orbit debris problem. In an instant hundreds of thousands of small, presently un-trackable fragments with much higher than hypervelocity bullet speeds began orbiting the Earth. This debris is hazardous to operational satellites and reducing the risk of future collisions is possible by rendezvous, capture and de-orbit missions to remove the largest and/or most dangerous derelict objects. There are over 500 USA-launched spent rocket boosters in low Earth orbit. Considering the possibility that a "debris mitigation" spacecraft can be developed to de-orbit spacecraft in neighboring orbits, we have an "orbiting traveling salesman" problem to find the optimal sequence. Determining the globally optimal sequence of maneuvers for retrieving orbital debris can require simulating thousands of feasible transfer trajectories. The Δv cost for each must be computed and displayed in an extremal field map (EFM) in order to effectively distinguish globally optimal from infeasible and sub-optimal orbit maneuver regions, as a function of take-off and arrival time.

The focus of this dissertation is the development of a unified Lambert tool (ULT) that has led to four algorithms, one that solves the Keplerian problem and three that solve the perturbed Lambert's problem. These three perturbed algorithms all make use of modified Chebyshev-Picard iteration (MCPI) in some way, shape of form. The first algorithm, a-iteration or p-iteration, is a variant of a well-known Keplerian Lambert solver that reduces the TPBVP problem to an iteration on the semimajor axis (semilatus rectum) until convergence upon a solution that satisfies the specific boundary conditions with the desired time of flight [16]. This is essentially the classical algorithm adopted as a convenient warm start for the perturbed algorithms. However, any solution of the Keplerian Lambert problem could be used for the warm start. The arc-length or true anomaly angle spanned by the transfer trajectory is the parameter that governs the automated selection of the appropriate perturbed algorithm, and the selection is based on the respective algorithm convergence characteristics. The second algorithm solves the perturbed problem using the standard MCPI-TPBVP algorithm [17, 18], which does not require a Newton-like shooting method, however the domain of convergence is limited to about one third of an orbit. The third algorithm extends this domain of convergence to about ninety percent of a transfer orbit period through regularization with the Kustaanheimo-Stiefel (KS) transformation [19]. This is the next most efficient of the perturbed set of algorithms. The fourth algorithm uses the method of particular solutions (MPS) and the MCPI initial value problem (IVP) algorithm for solving multi-revolution perturbed transfers [20, 21]. This method does require "shooting" but differs from Newton-like shooting methods in that it does not require propagation of a state transition matrix. This leads to efficiency and also permits ease of parallelization. These algorithms accommodate state of the art force models and the ULT as a whole is implemented in C/C++ and in parallel, using message passing interface (MPI), for high performance computation on a 192 core computer cluster. The mathematics underlying each algorithm are developed in this dissertation.

Variations of MCPI play a major role in each of the four methods (except a- and p-iteration) in the ULT, and thus a detailed description of Orthogonal Approximation (Chap 2), the MCPI IVP and BVP Formulations (Chap 3), and Recent Enhancements (Chap 4) are given before developing the algorithms. The following chapters build on these foundational developments starting with KS Regularization

(Chap 5) and MPS (Chap 6). A formulation for solving low thrust transfers orbit transfers using MCPI and MPS is also discussed (Chap 7). Chapter 8 fuses these orbit transfer techniques together into the ULT. This is followed by a demonstration of the parallel implementation of the ULT for generating EFM (Chap 9). Finally this dissertation concludes with a summary of the key contributions of this work and discusses the implications for future research.

2. ORTHOGONAL APPROXIMATION

A derivation for approximating a "general" smooth function of one variable is presented in this chapter. The discrete orthogonality of Chebyshev polynomials is used for efficiently determining the coefficients of the approximated function. The linear combination of basis functions also allows efficient analytical integration to be carried out through the Picard iteration computations presented in Chapter 3.

2.1 Function Approximation

Function approximation is a mathematical procedure whereby a certain *target* function is approximated with another *approximating* function that closely matches the "target/true" function. This is an extremely useful technique when the target function is "difficult to work with". For example, the approximation may be far easier to integrate than the true function. Consider a satellite in orbit about the Earth. It's motion is described by a nonlinear differential equation that is a function of position, velocity, gravitational acceleration and time. Given certain orbital information (i.e. position, velocity, acceleration) at a particular instant in time, one may determine these same parameters again at some future time by integrating the differential equation of motion that describes the orbit. In reality, many situations require the integration to be carried out numerically, and thus obtaining an accurate approximation of the function to be integrated can be crucial to arriving at the correct answer (to within a desired tolerance).

Approximating the true function is ideally done in a way that minimizes the difference (or error) between the true function and the approximate function. Least squares technique is a common method that includes a metric for determining the quality of an approximation. Once a good approximation is achieved, i.e. the error falls below a certain desired threshold, the approximated function may be used in place of the true function to predict future values of the function or its integral with known bounds on approximation accuracy. In general, the desired error threshold value is selected by the user and may be varied depending on the degree of precision required for a particular calculation.

There are an infinite variety of ways to approximate a function. If a linear combination of basis functions is used, one important property is that the basis functions be "complete". Completeness means, for practical purposes, that any smooth function can be replaced to arbitrary accuracy if enough measurements and basis functions are used. An interesting approximation technique is the use of the orthogonal Chebyshev polynomials (Appendix A), developed by the Russian mathematician, Rafnuty Lvovich Chebyshev. Chebyshev polynomials play an important role in the numerical integration technique known as modified Chebyshev-Picard iteration. More on this in the following chapters.

One of the great challenges associated with function approximation is the limitation imposed by computers. Obviously floating point word length ("machine precision") and the frequently competing demands of speed, precision and storage constraints provide an implicit bound on arithmetic precision achievable. Approximating functions of more than one variable, that require hundreds of terms to ensure that the desired error threshold is met, may require excessive computational power and time. If arbitrary basis functions are chosen, one frequently encounters large matrices that must be inverted to compute the coefficients of the approximated function. Furthermore, the larger the number of basis functions, the more expensive the coefficients are to compute. If such calculations are to be performed in real-time, and possibly on-board spacecraft, it is vital that computational efficiency be optimized accordingly. Orthogonality of basis functions leads to diagonalization of the matrix that must be inverted and therefore bypasses the expense and accuracy loss of matrix inversion. The cosine distribution of nodal points are consistent with discrete orthogonality of Chebyshev polynomials that is known to typically lead to near uniform approximation errors. Near uniform approximation errors are very attractive for the applications considered in this dissertation.

2.2 Domain Transformation

The function to be approximated in this section is a single-valued function of one independent variable (x), and is given by $g(x)$ in the following derivation.

$$g(x), \ \{x_{\min} \leq x \leq x_{\max}\} \tag{2.1}$$

The first step is to transform the function from its current domain onto a domain that has lower and upper bounds of -1 and 1 respectively. It is necessary that this transformation be carried out as the classical Chebyshev polynomials are only defined on the interval from -1 to 1. The transformation is done by introducing a new independent variable, ξ, such that $\{-1 \leq \xi \leq 1\}$. The forward and reverse transformations are given in Eqs 2.2 and 2.3 respectively.

$$x(\xi) = x_{\min} + (\xi + 1)\left(x_{\max} - x_{\min}\right)/2 \tag{2.2}$$

$$\xi(x) = 2\left(x - x_{\min}\right) / \left(x_{\max} - x_{\min}\right) - 1 \tag{2.3}$$

There are an infinity of other nonlinear mappings of x onto $\{-1 \leq \xi \leq 1\}$. The above linear transformation is the most widely used, however [22] introduces an alternate asymmetric nonlinear transformation for gravitation field modeling. When $\xi = 1$ is substituted into Eq. 2.2, the result is $x(1) = x_{\max}$, and similarly for $\xi = -1$

the result is $x(-1) = x_{\min}$. As desired, this maps the x values from the function $g(x)$ onto the $[-1, 1]$ domain. All the function approximation calculations are carried out while the data/function is in this form. Once the calculation is complete, the data can be mapped back to the original domain using Eq. 2.3. When $\xi(x) = 1$ is substituted into Eq. 2.3, the result $x_{\max} = x$, and similarly for $\xi(x) = -1$ the result is $x_{\min} = x$. Substituting the forward transformation into Eq. 2.1 gives Eq. 2.4. This is the "transformed" function that is to be approximated.

$$f(\xi) \equiv g(x(\xi)) = g\left(x_{\min} + (\xi + 1)\left(x_{\max} - x_{\min}\right)/2\right) \tag{2.4}$$

2.3 Approximation and Error

The approximation of $f(\xi)$ may be written as the sum of the $N+1$ polynomial terms, where $\{\phi_0(\xi), \phi_1(\xi), ..., \phi_N(\xi)\}$ are linearly independent basis functions, and $\{a_0, a_1, ..., a_N\}$ are the coefficients of these basis functions. Chebyshev polynomials are chosen as the basis functions for most of the remainder of this dissertation. The finite degree polynomials (up to degree N) give the approximation

$$f(\xi) \approx \sum_{\alpha=0}^{N} a_\alpha \phi_\alpha(\xi). \tag{2.5}$$

For discrete measurement samples, a set of sample points or nodes (Section 2.6) are introduced. These are given by $\{\xi_0, \xi_1, ..., \xi_M; M \geq N\}$. The sample points (nodes) are the locations where actual values of the true function are computed. The difference between the true value and the approximated value at each measurement node is known as the residual approximation error and is given by:

$$r_j = f(\xi_j) - \sum_{\alpha=0}^{N} a_\alpha \phi_\alpha(\xi_j);\ j = 0, 1, ..., M. \tag{2.6}$$

Here r_j is the residual at the j^{th} specific sample point, $f(\xi_j)$ is the value of the true function the j^{th} sample point, and the "summation" term is the approximating function.

2.4 Vector-matrix Notation

In vector-matrix notation Eq. 2.6 becomes the linear system

$$\boldsymbol{r} = \boldsymbol{f} - \Phi \boldsymbol{a}, \tag{2.7}$$

where

$$\boldsymbol{f} = \begin{bmatrix} f(\xi_0) \\ f(\xi_1) \\ \vdots \\ f(\xi_M) \end{bmatrix}, \ \Phi = \begin{bmatrix} \phi_0(\xi_0) & \phi_1(\xi_0) & \cdots & \phi_N(\xi_0) \\ \phi_0(\xi_1) & \phi_1(\xi_1) & \cdots & \phi_N(\xi_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\xi_M) & \phi_1(\xi_M) & \cdots & \phi_N(\xi_M) \end{bmatrix}, \ \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}. \tag{2.8}$$

Eq. 2.7 can be rearranged into the more familiar notation $\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{Ax} = 0$ (i.e. $A\boldsymbol{x} = \boldsymbol{b}$), where $\boldsymbol{f}$ is the $\boldsymbol{b}$ vector, A is the Φ matrix, and $\boldsymbol{x}$ is the $\boldsymbol{a}$ vector of coefficients that we wish to solve for. The least squares minimization of $\boldsymbol{r}^T W \boldsymbol{r}$ leads to a solution for $\boldsymbol{a}$ (the normal equations) in Eq. 2.9, where $W = diag\left\{\frac{1}{2}, 1, 1, ..., 1, 1, \frac{1}{2}\right\}$ is a positive definite weight matrix chosen to preserve orthogonality. This weight matrix together with "cosine sampling" discussed below guarantee discrete orthogonality of Chebyshev polynomial approximation with $\Phi^T W \Phi$ diagonalized. A derivation of least squares and a mathematical explanation for the chosen weight are given in Appendix B, sections B.1 and B.2 respectively.

$$\boldsymbol{a} = \left(\Phi^T W \Phi\right)^{-1} \Phi^T W \boldsymbol{f}. \tag{2.9}$$

2.5 Orthogonality Conditions

The explicit solution for the coefficients of Eq. 2.9 is given by the independent/uncoupled ratios of inner products as

$$a_\alpha = \frac{\langle \phi_\alpha(\xi), f(\xi) \rangle}{\langle \phi_\alpha(\xi), \phi_\alpha(\xi) \rangle} \equiv \frac{\sum\limits_{j=0}^{M} W_j \phi_\alpha(\xi_j) f(\xi_j)}{\sum\limits_{j=0}^{M} W_j \phi_\alpha^2(\xi_j)} \equiv \frac{1}{c_\alpha} \sum_{j=0}^{M} W_j \phi_\alpha(\xi_j) f(\xi_j), \text{for} \alpha = 0, 1, 2, ..., N. \tag{2.10}$$

An important special case arises when the Chebyshev polynomials $\{T_0(\xi), T_1(\xi), ..., T_N(\xi)\}$ are used as the basis functions, namely $\{\phi_0(\xi), \phi_1(\xi), ..., \phi_N(\xi)\} = \{T_0(\xi), T_1(\xi), ..., T_N(\xi)\}$. This is discussed by several authors [23, 24], and is explained in detail in this chapter.

2.6 Cosine Sampling

The $M + 1$ sample points are chosen using a *cosine* relationship as shown in Eq. 2.11. This type of node sampling is also known as the CGL nodes in honor of Chebyshev-Gauss-Lobatto:

$$\xi_j = -\cos(j\pi/M),\ j = 0, 1, 2, ..., M. \tag{2.11}$$

The cosine nodes of Eq. 2.11 locate all M–1 extrema of the Chebyshev polynomials, as well as the two end points of the approximation interval. It is important to note that cosine sampling causes the sample points to cluster near the ±1 boundaries as the degree N of the approximation increases. The errors are smaller near the centers of the supporting data for an obvious (qualitative) reason, there are redundant data on either side of the interpolated value. One the other hand, near the ±1 ends of the interval, there is data only on one side of the interpolation point. The cosine concentration of nodes near ±1 serves to compensate for the fact that there are no measurements to the left of the $\xi = -1$ and to the right of $\xi = +1$. It is well known that Chebyshev least square approximation with cosine nodes usually leads to near uniform errors in spite of the compact support on the ± 1 interval. This characteristic is shown graphically in Figure 2.1, along with uniformly spaced samples for comparison. Cosine sampling is puposefully chosen to reduce the runge effect. This effect is clearly illustrated in Figure 2.2. For circumstances where we desire piecewise continuous approximation of a function over long intervals (e.g. a many-revolution propagation orbit via MCPI, see Chapter 4), clearly having the approximation accuracy degrade at interval (segment) boundaries is highly undesirable.

An alternative orthogonal approximation can be based on the following cosine nodes

$$\xi_j = -\cos((j - \frac{1}{2})\pi/M),\ j = 0, 1, 2, ..., M. \tag{2.12}$$

Note that the nodes of Eq. 2.10 include all of the extrema of $T_\alpha(\xi)$ as well as the $\xi = \pm 1$ end points, whereas the alternative of Eq. 2.12 locates the nodes at the zeros of $T_\alpha(\xi)$ and does not include the end points. As known in the literature [25] the nodes of Eq. 2.10 are superior to ensure best approximation accuracy at the ends of the interval. This is needed for best performance of MCPI under the frequent situation where approximation is done in piecewise segments and joined head-to-tail over large domains.

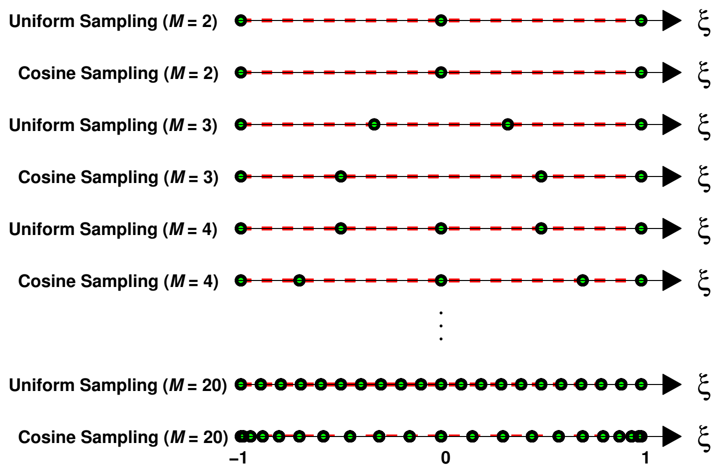

Uniform Sampling ($M = 2$)
Cosine Sampling ($M = 2$)
Uniform Sampling ($M = 3$)
Cosine Sampling ($M = 3$)
Uniform Sampling ($M = 4$)
Cosine Sampling ($M = 4$)
Uniform Sampling ($M = 20$)
Cosine Sampling ($M = 20$)
ξ
–1
0
1

Figure 2.1: Cosine Nodes [22].

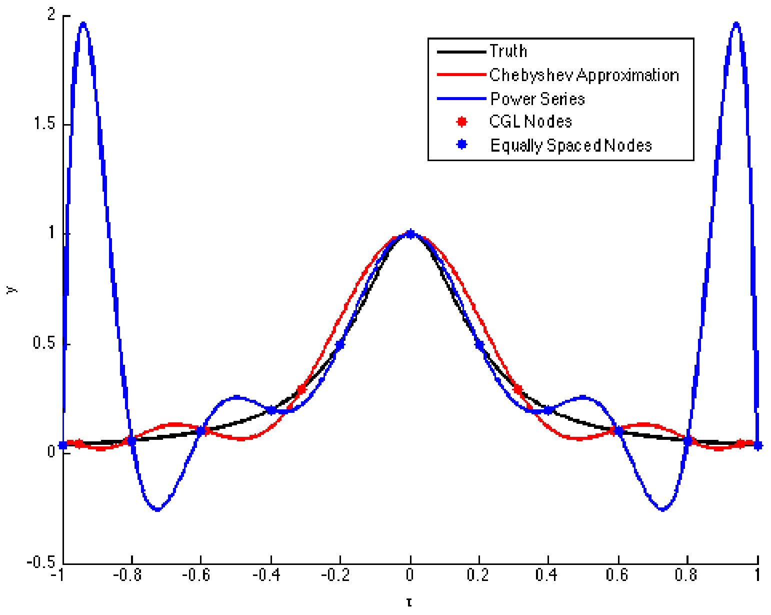


Figure 2.2: The Runge effect is clearly seen in the power series approximation whereas the Chebyshev approximation with cosine nodes has smaller more nearly uniform errors.

2.7 Chebyshev Approximation Coefficients

Upon substituting the sample points of Eq. 2.11 and the chosen weight matrix, it is easy to verify that orthogonality conditions are satisfied and the least square coefficients of Eq. 2.10 are specifically

$$a_\alpha = \frac{1}{c_\alpha}\left\{\sum_{j=0}^{M} W_j T_\alpha(\xi_j) f(\xi_j)\right\}$$

$$= \frac{1}{c_\alpha}\left\{\frac{1}{2}T_\alpha(\xi_0)f(\xi_0) + ... + T_\alpha(\xi_{M-1})f(\xi_{M-1}) + \frac{1}{2}T_\alpha(\xi_M)f(\xi_M)\right\}, \qquad (2.13)$$

where the denominators c_α in Eq. 2.13 are the positive constants

$$c_\alpha = \sum_{j=0}^{M} W_j T_\alpha^2(\xi_j) = \left\{\frac{1}{2}T_\alpha^2(\xi_0) + T_\alpha^2(\xi_1) + ... + T_\alpha^2(\xi_{M-1}) + \frac{1}{2}T_\alpha^2(\xi_M)\right\}, \alpha = 0, 1, ..., N, \qquad (2.14)$$

or, more explicitly it can be verified (Appendix B, Section B.2.3) that the denominator inner products reduce to simply

$$\left.\begin{array}{lll} c_0 & = \langle T_0(\xi), T_0(\xi)\rangle = M & \\ c_\alpha & = \langle T_\alpha(\xi), T_\alpha(\xi)\rangle = M/2, & \alpha = 1, 2, ..., N-1 \\ c_N & = \langle T_N(\xi), T_N(\xi)\rangle = M, & \text{if } M = N \text{ (interpolation case)} \\ c_N & = \langle T_N(\xi), T_N(\xi)\rangle = M/2, & \text{if } M > N \text{ (least squares case)} \end{array}\right\}. \qquad (2.15)$$

It is important to note that for $M \geq N$ the *least squares* method may be used for determining the solution (coefficients) for the system. However, if $M = N$ the least squares Chebyshev approximation reduces to the Chebyshev interpolation.

The final coefficients for least square approximation are computed directly from the discrete inner products of Eq. 2.13 as

$$\left.\begin{array}{lll}
a_0 & = \frac{\langle T_0(\xi), f(\xi)\rangle}{\langle T_0(\xi), T_0(\xi)\rangle} = \frac{1}{M}\left\{\frac{1}{2}T_0(\xi_0)f(\xi_0)+\ldots+T_0(\xi_{M-1})f(\xi_{M-1})+\frac{1}{2}T_0(\xi_M)f(\xi_M)\right\} & \\
a_\alpha & = \frac{\langle T_\alpha(\xi), f(\xi)\rangle}{\langle T_\alpha(\xi), T_\alpha(\xi)\rangle} = \frac{2}{M}\left\{\frac{1}{2}T_\alpha(\xi_0)f(\xi_0)+\ldots+T_\alpha(\xi_{M-1})f(\xi_{M-1})+\frac{1}{2}T_\alpha(\xi_M)f(\xi_M)\right\}, & \alpha=1,2,\ldots,N-1 \\
a_N & = \frac{\langle T_N(\xi), f(\xi)\rangle}{\langle T_N(\xi), T_N(\xi)\rangle} = \frac{1}{C_N}\left\{\frac{1}{2}T_N(\xi_0)f(\xi_0)+\ldots+T_N(\xi_{M-1})f(\xi_{M-1})+\frac{1}{2}T_N(\xi_M)f(\xi_M)\right\}, & \left\{\begin{array}{l} c_N=M,\ M=N \\ c_N=\frac{M}{2},\ M>N \end{array}\right.
\end{array}\right\}. \tag{2.16}$$

Note that the coefficients of Eq. 2.16 are computed independently, and the absolute value of each coefficient is the maximum contribution of that term – this enables convenient means for obtaining efficient and accurate truncated approximations, as well as insight for adapting the order of the approximation.

2.8 Vector Matrix Notation

If a vector-matrix form is desired for the least squares solution for the coefficients, Eq. 2.16 can be expressed as

$$\boldsymbol{a} = \left(\Phi^T W \Phi\right)^{-1} T W \boldsymbol{f} = V T W \boldsymbol{f} = A \boldsymbol{f}. \tag{2.17}$$

This is in the same form as the *least squares minimization* Eq. 2.9, where $\left(\Phi^T W \Phi\right)^{-1} = V = diag\left\{\frac{1}{c_0}, \frac{1}{c_\alpha}, \frac{1}{c_\alpha}, \ldots, \frac{1}{c_\alpha}, \frac{1}{c_\alpha}, \frac{1}{c_N}\right\}$, $T = \Phi^T$ and W and $\boldsymbol{f}$ are the same. The Chebyshev least square operator matrix (A) is simply given by Eq. 2.18 for $M = N$, and Eq. 2.19 for $M > N$. Note that the only difference between the two A matrices is the final row.

For $M = N$,

$$A = \begin{bmatrix} \frac{1}{2}\frac{1}{M}T_0(\xi_0) & \frac{1}{M}T_0(\xi_1) & \cdots & \frac{1}{M}T_0(\xi_{M-1}) & \frac{1}{2}\frac{1}{M}T_0(\xi_M) \\ \frac{1}{2}\frac{2}{M}T_1(\xi_0) & \frac{2}{M}T_1(\xi_1) & \cdots & \frac{2}{M}T_1(\xi_{M-1}) & \frac{1}{2}\frac{2}{M}T_1(\xi_M) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2}\frac{2}{M}T_{N-1}(\xi_0) & \frac{2}{M}T_{N-1}(\xi_1) & \cdots & \frac{2}{M}T_{N-1}(\xi_{M-1}) & \frac{1}{2}\frac{2}{M}T_{N-1}(\xi_M) \\ \frac{1}{2}\frac{1}{M}T_N(\xi_0) & \frac{1}{M}T_N(\xi_1) & \cdots & \frac{1}{M}T_N(\xi_{M-1}) & \frac{1}{2}\frac{1}{M}T_N(\xi_M) \end{bmatrix}. \tag{2.18}$$

For $M > N$,

$$A = \begin{bmatrix} \frac{1}{2}\frac{1}{M}T_0(\xi_0) & \frac{1}{M}T_0(\xi_1) & \cdots & \frac{1}{M}T_0(\xi_{M-1}) & \frac{1}{2}\frac{1}{M}T_0(\xi_M) \\ \frac{1}{2}\frac{2}{M}T_1(\xi_0) & \frac{2}{M}T_1(\xi_1) & \cdots & \frac{2}{M}T_1(\xi_{M-1}) & \frac{1}{2}\frac{2}{M}T_1(\xi_M) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{2}\frac{2}{M}T_{N-1}(\xi_0) & \frac{2}{M}T_{N-1}(\xi_1) & \cdots & \frac{2}{M}T_{N-1}(\xi_{M-1}) & \frac{1}{2}\frac{2}{M}T_{N-1}(\xi_M) \\ \frac{1}{2}\frac{2}{M}T_N(\xi_0) & \frac{2}{M}T_N(\xi_1) & \cdots & \frac{2}{M}T_N(\xi_{M-1}) & \frac{1}{2}\frac{2}{M}T_N(\xi_M) \end{bmatrix}. \tag{2.19}$$

In general, matrix A is an $[N + 1] \times [M + 1]$ matrix. The number of rows is determined by the degree of the polynomial (number of terms), and the number of columns is determined by the number of sample points (ξ). Of course when $M = N$, the matrix A is square and it is given by Eq. 2.18.

2.9 Return to Approximation (Section 2.3)

Having completed the derivation to compute the coefficients ($\boldsymbol{a}$) we return to Eq. 2.5, also shown below on the **L.H.S.** of Eq. 2.20. The approximated function is determined as the inner product of the coefficients ($\boldsymbol{a}_\alpha$) with the basis function ($\phi_\alpha(\xi)$), or in our case Chebyshev polynomials ($T_\alpha(\xi)$). Thus the function $f(\xi)$ on the **L.H.S.** of Eq. 2.20 becomes equivalent to the sum on the **R.H.S.** Using summation notation the function is approximated as follows. Note that in this notation the first element of V and W is the *zero-zero* element.

$$\boldsymbol{f}(\xi) \approx \sum_{\alpha=0}^{N} a_\alpha \phi_\alpha(\xi) = \sum_{\alpha=0}^{N} \left\{ \sum_{j=0}^{M} V_{\alpha j} W_{\alpha j} T_\alpha(\xi_j) f(\xi_j) \right\} T_\alpha(\xi) \tag{2.20}$$

2.10 Summary

A derivation for approximating a "general" function of one variable is presented in this chapter. The first step transforms the function onto the domain $[-1, 1]$ for which the classical Chebyshev are exist. This is followed by a discussion on function approximation using the orthogonal Chebyshev polynomials. The least squares method for determining the coefficients of the Chebyshev polynomials, and the specific weight matrix is discussed. It highlights the orthogonality conditions which dictates that no matrix inversion is required, greatly simplifying the overall calculation. The absence of a matrix inversion together with a complete set of orthogonality functions mean that we have a spectral approximation that can efficiently approach machine precision. With an efficient, robust and adaptive method for computing the coefficients of the approximated function, we proceed on to Chapter 3 and discuss how these come into play when combined with Picard iteration.

3. MCPI IVP AND BVP FORMULATIONS

This chapter provides a detailed mathematical explanation of the MCPI formulation for solving general IVPs and TPBVPs. These methods are extended upon in the latter chapters of this dissertation to solve problems that do not easily fit into the standard MCPI IVP and TPBVP formulations.

3.1 Modified Chebyshev Picard Iteration

MCPI is an attractive numerical method for solving linear or nonlinear differential and integral equations, and it combines the discoveries of two great mathematicians: Emile Picard (Picard Iteration) and Rafnuty Chebyshev (Chebyshev Polynomials). A brief description of each technique is given in the following sub-sections, with the more detailed mathematical description of each given in Appendix A and C.

MCPI differs from the well-known integrators used widely in astrodynamics, such as Gauss-Jackson, Runge-Kutta-Nystrom and ODE45 in that MCPI is an iterative *path approximation* numerical integrator rather than a *time-step* extrapolation integrator. That is, a relatively long state trajectory arc is approximated continuously and can be updated at all time nodes simultaneously on each path iteration. As will be evident below, a slight modification of the MCPI initial value solver makes the algorithm applicable to TPBVPs. The solution of TPBVPs utilizing MCPI *does not require* a local-linearization-based shooting method.

3.1.1 Picard Iteration

Picard iteration was developed during the mid to late 1800s by Émile Picard following a first discovery by Joseph Liouville in 1833. It is a successive path approximation technique that is generally used for proving the existence and uniqueness of solutions to the IVP. In later years Picard extended the capabilities to deal with systems of second order differential equations, and this also allowed natural solutions to the BVP to be computed [26]. A number of other people also worked to develop Picard iteration [27, 28, 29, 30, 31]

Over a large space of time intervals and starting approximations for most dynamical systems, Picard iteration is a contraction mapping (as discussed below) that converges to the solution satisfying both the differential equation and the boundary conditions. Consider the nonlinear first order differential equation

$$\frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad t \in [t_0, t_f], \, \boldsymbol{x} \in R^{n+1}, \, \boldsymbol{f} \in R^{n+1}, \tag{3.1}$$

where $\boldsymbol{f}$ is a smooth differentiable vector function.

Picard observed that the first order differential equation (Eq. 3.1) with an initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, can be rearranged, without approximation, to obtain the following integral equation:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau)) d\tau. \tag{3.2}$$

This re-arrangement does not appear to have made any progress towards finding the solution since the unknown trajectory $\boldsymbol{x}(t)$ is contained within the integrand on the right hand side. However, given some starting estimate $\boldsymbol{x}^0(t)$ for the path, a sequence of approximate solutions $\boldsymbol{x}^i(t)$ (where $i = 1, 2, 3..., \infty$) of the true solution $\boldsymbol{x}(t)$ that satisfies this differential equation and the prescribed boundary conditions, may be obtained through Picard iteration using the following Picard sequence of approximate paths $\{\boldsymbol{x}^0(t), \boldsymbol{x}^1(t), ..., \;\; \boldsymbol{x}^{i-1}(t), \boldsymbol{x}^i(t), ...\}$:

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}^{i-1}(\tau))d\tau, \quad i = 1, 2, ... \tag{3.3}$$

Picard proved an important convergence theorem that essentially states that for smooth, differentiable, single-valued nonlinear functions $\boldsymbol{f}(t, \boldsymbol{x}(t))$, the Picard sequence of trajectories represents a contraction operator that converges to the unique solution of the IVP. That is, if there is a time interval $|t_f - t_0| < \delta$ and a starting trajectory $\boldsymbol{x}^0(t)$ satisfying $\|\boldsymbol{x}^0(t) - \boldsymbol{x}(t)\| < \Delta$, for suitable finite bounds (δ, Δ), then the Picard sequence converges to the unique solution of Eq. 3.1. Bai [18] discusses the literature on estimating the theoretical convergence bounds (δ,Δ). As a practical matter the available theory for estimating (δ, Δ) is known to typically give very conservative estimates and for general nonlinear systems, the convergence domain can be accurately estimated through adaptive numerical methods. Picard did not establish a numerically attractive means to compute the sequence of integrals, so as a numerical method to approximate $\boldsymbol{x}(t)$, Picard iteration had limited adoption until the mid 1900s. The rate of convergence of Picard iteration depends on the particular problem and is typically an (approximately) geometric rate. Insights have been obtained recently that, for the main problem in astrodynamics (orbit propagation), make Picard iteration attractive compared to current state of the practice ordinary differential equation solvers.

3.1.2 Chebyshev Polynomials

In 1857 the Russian mathematician, Rafnuty Lvovich Chebyshev, developed a set of orthogonal polynomials that are now referred to as Chebyshev polynomials [23]. Following this a number of authors made contributions using Chebyshev polynomials: [24, 32, 33, 34, 35, 36, 37, 38]. Orthogonality means that every pair of polynomials in a set of basis functions are orthogonal to each other (on a specific domain and with a specific inner product definition). Orthogonality can be defined as a function of the independent variable τ, over the domain $\{-1 \leq \tau \leq 1\}$ in either the continuous or discretely sampled sense. We will adopt a discrete sampling approach to orthogonality in this dissertation. Orthogonality is extremely important in the field of approximation theory because it means that the coefficients that linearly combine the basis functions to approximate a given function can be computed independently as simple ratios of inner products, with no matrix inversion. The Chebyshev orthogonal polynomials are also a complete set, an arbitrary given continuous function can always be well approximated, to arbitrary precision, by a sufficient number of samples and terms in the Chebyshev series (ignoring finite precision of computation of course).

Chebyshev polynomials are usually generated by the three term recurrence relation as follows:

$$T_0(\tau) = 1, \tag{3.4}$$

$$T_1(\tau) = \tau, \tag{3.5}$$

$$T_{k+1}(\tau) = 2\tau T_k(\tau) - T_{k-1}(\tau),\ k = 1, 2, ... \tag{3.6}$$

The classical Picard iteration method for solving IVPs and BVPs had little adoption for practical computation until the mid-to-late 1900s. The primary challenge, that was eventually overcome, was how to efficiently, accurately and adaptively approximate the integral on each Picard iteration. Clenshaw and Norton [39] developed an approach that satisfactorily addressed this challenge. They developed a variant of Picard iteration based on an approximation of the integrand of Eq. 3.3 using Chebyshev polynomials. First the integrand (e.g. orbital differential equation of motion) in Eq. 3.3 is approximated using the orthogonal Chebyshev polynomials, and then each term is integrated analytically, term-by-term over the previous path approximation. The first one (or two, for the second order generalization) coefficients can be used to satisfy the boundary conditions. Thus each Picard iteration is designed so that it satisfies the known boundary conditions exactly, and the iteration process leads to a high precision satisfaction of the differential equation.

3.2 MCPI Notations

The process to compute the coefficients obtained through orthogonal function approximations in the previous chapter is implemented in the Picard iteration technique in this chapter. The previous chapter dealt with a general function approximation, however in this chapter the focus is specifically on approximating the integrand and carrying out the integration to establish a convergent sequence of trajectory approximations over a given time interval. Notice, assuming we have a problem for which the Picard sequence will converge from a large family of starting paths $\boldsymbol{x}^0(t)$, only the accuracy of the integrals in the Picard iterations on the final two iterations dictates the final convergence accuracy. Said another way, we have found that the accuracy of the force model approximation can be adjusted (if this permits efficiency gains) during the iteration, consistent with the accuracy of the current path iteration.

The symbols used for the MCPI calculation presented in this chapter differ slightly from those used for “general function approximation” technique presented in Chapter 2. In particular:

- The sample independent variable ξ are denoted by τ, a transformed time variable.
- The function to be approximated ($\boldsymbol{f}(\xi)$) is transformed to $\boldsymbol{g}(\tau)$, or more spe-

cifically $\boldsymbol{g}(\tau, \boldsymbol{x}^{i-1}(\tau))$ as specified below in Eq. 3.10, approximated by Eq. 3.12.

- The coefficient vector a_α are replaced by vectors $\boldsymbol{F}_k$ and $\boldsymbol{\beta}_k$ for approximating the integrand and the updated state trajectory of the Picard sequence of Eq. 3.3.

- Finally, the generic basis function $(\phi_k(\xi))$ is replaced specifically with the orthogonal Chebyshev polynomials $(T_k(\tau))$.

These notations are adopted in order to maintain consistency with previous publications that incorporate the MCPI derivation.

3.3 Time Domain Transformation

Eq. 3.7 is an ordinary differential equation where t is the independent variable and $\boldsymbol{x}$ is a vector of dependent variables, typically in seconds. The initial and final times are given by t_0 and t_f respectively, and $\boldsymbol{x}(t = t_0) = \boldsymbol{x}_0$ is the initial condition.

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(t, \boldsymbol{x}) \tag{3.7}$$

In order to use Chebyshev polynomials as the approximating function, the trajectory must be transformed from the time domain (t) onto a domain (τ) ranging from -1 to 1. The linear transformation adopted is given as follows:

$$t = w_1 + w_2\tau, \quad w_1 = (t_f + t_0)/2, \quad w_2 = (t_f - t_0)/2, \quad -1 \leq \tau \leq 1. \tag{3.8}$$

Differentiating t with respect to τ gives

$$\frac{dt}{d\tau} = w_2, \tag{3.9}$$

and multiplying $\frac{dx}{dt}$ from Eq. 3.7 and $\frac{dt}{d\tau}$ from Eq. 3.9 results in the following slightly transformed system of ODEs:

$$\frac{d\boldsymbol{x}}{d\tau} = w_2 \boldsymbol{f}(w_1 + w_2\tau, \boldsymbol{x}) \equiv \boldsymbol{g}(\tau, \boldsymbol{x}). \tag{3.10}$$

Eq. 3.10 is simply Eq. 3.7 transformed from the time domain into the τ domain.

3.4 First Order Picard Method

Picard Iteration for the transformed system is written by analogy with Eq. 3.2 as

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}_0 + \int_{-1}^{\tau} \boldsymbol{g}(s, \boldsymbol{x}^{i-1}(s))ds \quad i = 1, 2, ... \tag{3.11}$$

3.4.1 Computation of R.H.S. Integrand Approximation

Chebyshev polynomials are used to approximate the *integrand* on the **R.H.S.** of Eq. 3.11. It is important to note that the entire trajectory, most generally (or a large segment of the trajectory), is approximated on each Picard iteration step. That is, for each iteration (i) all sample points (τ_j) are used over a large time interval for the trajectory approximation. Since this approximation is done within the integral, the upper index of the summation is only performed to $N-1$ (Eq. 3.12) instead of N, as on the **L.H.S.** Integration increases the degree of the polynomial from $N-1$ to N and therefore care must be taken to ensure that post integration leads to the same degree polynomial on either side of the Picard iteration expression Eq. 3.11.

$$\boldsymbol{g}(s, \boldsymbol{x}^{i-1}(s)) = \sum_{k=0}^{N-1} \boldsymbol{F}_k^{i-1} T_k(s) \equiv \boldsymbol{F}_0^{i-1} T_0(s) + \boldsymbol{F}_1^{i-1} T_1(s) + \boldsymbol{F}_2^{i-1} T_2(s) + ... + \boldsymbol{F}_{N-1}^{i-1} T_{N-1}(s). \tag{3.12}$$

Using the discrete orthogonality property of Chebyshev polynomials derived in Chapter 2, the coefficient vectors $\boldsymbol{F}_k^{i-1}$ can be calculated immediately through

$$\boldsymbol{F}_k^{i-1} = \sum_{j=0}^{M} V_{kj} W_{kj} \boldsymbol{g}(\tau_j, \boldsymbol{x}^{i-1}(\tau_j)) T_k(\tau_j). \tag{3.13}$$

In Eq. 3.13, $W = diag\left\{\frac{1}{2}, 1, 1, ..., 1, 1, \frac{1}{2}\right\}$, and $V = diag\left\{\frac{1}{M}, \frac{2}{M}, \frac{2}{M}, ..., \frac{2}{M}, \frac{2}{M}, \frac{1}{M}\right\}$ or $V = diag\left\{\frac{1}{M}, \frac{2}{M}, \frac{2}{M}, ..., \frac{2}{M}, \frac{2}{M}, \frac{2}{M}\right\}$ for $M = N$ or $M > N$ respectively. Note also that in this notation the first element of W and V is the *zero-zero* element since the summation starts from zero (i.e. W_{00} and V_{00}). In order to verify the statements made in this paragraph the reader should refer to the derivations in the latter sections of Chapter 2.

The coefficients $\{\boldsymbol{F}_0, \boldsymbol{F}_1, ..., \boldsymbol{F}_{N-1}\}$ are computed as the summation of the $N-1$ products that are generated through multiplication of the force function g with the Chebyshev polynomial (T_k), evaluated at the CGL point τ_j (see Section 2.6).

3.4.2 Computation of LHS

The *unknown trajectory* $(\boldsymbol{x}^i(\tau))$ or **L.H.S.** of Eq. 3.11 is also written in terms of an approximating function, the sum of the inner product of coefficients $(\boldsymbol{\beta}_k)$ and Chebyshev polynomials $(T_k(\tau))$. This is done to allow the **L.H.S.** and **R.H.S.** of Eq. 3.11 to be equated as described in the following section.

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\beta}_k^i T_k(\tau) \equiv \boldsymbol{\beta}_0^i T_0(\tau_j) + \boldsymbol{\beta}_1^i T_1(\tau_j) + \boldsymbol{\beta}_2^i T_2(\tau_j) + ... + \boldsymbol{\beta}_N^i T_N(\tau_j),\ j = 0, 1, 2, ..., M. \tag{3.14}$$

3.4.3 Substitution: LHS and RHS

Substituting Eq. 3.12 and Eq. 3.14 into Eq. 3.11 results in Eq. 3.15. Since Picard iteration starts from an initial "transformed time" condition of -1 in the τ

domain, the first term on the **R.H.S.** is $\boldsymbol{x}(-1)$.

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\beta}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{F}_k^{i-1} T_k(s) ds. \tag{3.15}$$

Looking at Eq. 3.15 we have "three variables": $T_k(s)$ and $\boldsymbol{F}_k^{i-1}$ are known and $\boldsymbol{\beta}_k^i$ is unknown. Thus equating the coefficients of $T_k(\tau)$ on the **L.H.S.** and **R.H.S.** of Eq. 3.15 allows the $\boldsymbol{\beta}_k^i$ coefficients to be determined in terms of the approximated and known $\boldsymbol{F}_k^{i-1}$ coefficients. Several examples are given in Appendix D to demonstrate this process to derive general formulae for the $\boldsymbol{\beta}_k^i$ coefficients that are independent of the selected polynomial degree and the number of sample points. The result is the following generalized formulae for determining the coefficients from the integration and the free boundary condition:

$$\boldsymbol{\beta}_0^i = \boldsymbol{x}_0 + \sum_{k=1}^{k=N} (-1)^{k+1} \boldsymbol{\beta}_k^i, \tag{3.16}$$

$$\boldsymbol{\beta}_1^i = \frac{1}{2}\left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1}\right), \tag{3.17}$$

$$\boldsymbol{\beta}_k^i = \frac{1}{2k}\left(\boldsymbol{F}_{k-1}^{i-1} - \boldsymbol{F}_{k+1}^{i-1}\right),\ k = 1, 2, ..., N-1, \tag{3.18}$$

$$\boldsymbol{\beta}_{N-1}^i = \frac{\boldsymbol{F}_{N-2}^{i-1}}{2(N-1)}, \tag{3.19}$$

and

$$\boldsymbol{\beta}_N^i = \frac{\boldsymbol{F}_{N-1}^{i-1}}{2N}. \tag{3.20}$$

3.4.4 Vector Matrix Notation

The values that were derived in the previous section can be written in a vector matrix representation as shown in the equations that follow. These developments permit us to collect one-time computations and various inner products in a way that makes Picard iteration efficient for modern computation.

$$\boldsymbol{\beta}^i = \begin{bmatrix} \boldsymbol{x}_0 + \left(\boldsymbol{\beta}_1^i - \boldsymbol{\beta}_2^i + \boldsymbol{\beta}_3^i + \cdots + (-1)^{N+1}\boldsymbol{\beta}_N^i\right) \\ \frac{1}{2}\left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1}\right) \\ \frac{1}{4}\left(\boldsymbol{F}_1^{i-1} - \boldsymbol{F}_3^{i-1}\right) \\ \vdots \\ \frac{1}{2k}\left(\boldsymbol{F}_{k-1}^{i-1} - \boldsymbol{F}_{k+1}^{i-1}\right) \\ \vdots \\ \frac{1}{2(N-1)}\boldsymbol{F}_{N-2}^{i-1} \\ \frac{1}{2N}\boldsymbol{F}_{N-1}^{i-1} \end{bmatrix} \tag{3.21}$$

More specifically, the $\boldsymbol{\beta}^i$ coefficients may be "built" through matrix addition and multiplication as shown in Eq. 3.22. The R matrix is given by $R = diag\left(1, \frac{1}{2k}, ..., \frac{1}{2N}\right)$, the k^{th} $(k = 3, ..., N-1)$ column of the first row of the S matrix is calculated as $S(1,k) = (-1)^k\left(\frac{1}{2(k-2)} - \frac{1}{2k}\right)$, and $\boldsymbol{F}^{i-1}$ is the vector of coefficients calculated through approximation on the **R.H.S.** of Eq. 3.12. This matrix RS is obviously invariant for N specified. $\boldsymbol{F}^{i-1}$ is calculated as $\boldsymbol{F}^{i-1} = A\boldsymbol{g}$, where $A = VTW$ is the matrix obtained in Chapter 2.

$$\boldsymbol{\beta}^{i} = \underbrace{\begin{bmatrix} \boldsymbol{x}_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{X}_0} + \underbrace{\begin{bmatrix} 1 & & & & & & & \\ & \frac{1}{2} & & & & & & \\ & & \frac{1}{4} & & & & & \\ & & & \ddots & & & & \\ & & & & \frac{1}{2\times k} & & & \\ & & & & & \ddots & & \\ & & & & & & \frac{1}{2(N-1)} & \\ & & & & & & & \frac{1}{2N} \end{bmatrix}_{(N+1)\times(N+1)}}_{R} \underbrace{\begin{bmatrix} \frac{1}{2} & -\frac{1}{4} & S(1,3) & S(1,4) & S(1,5) & S(1,6) & \cdots \\ 2 & 0 & -1 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & -1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & -1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & \ddots & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 1 \end{bmatrix}_{(N+1)\times N}}_{S} \underbrace{\begin{bmatrix} \boldsymbol{F}_0^{i-1} \\ \boldsymbol{F}_1^{i-1} \\ \boldsymbol{F}_2^{i-1} \\ \vdots \\ \boldsymbol{F}_{N-2}^{i-1} \\ \boldsymbol{F}_{N-1}^{i-1} \end{bmatrix}_{N\times 1}}_{\boldsymbol{F}}, \tag{3.22}$$

where $\boldsymbol{F}$ is calculated as follows

$$\boldsymbol{F}^{i-1} = A\boldsymbol{g} = \overbrace{\left[\overbrace{\underbrace{\overbrace{\begin{bmatrix} \frac{1}{M} & & & & \\ & \frac{2}{M} & & & \\ & & \ddots & & \\ & & & \frac{2}{M} & \\ & & & & \frac{p}{M} \end{bmatrix}}^{V}_{N\times N} \underbrace{\overbrace{\begin{bmatrix} T_0(\tau_0) & T_0(\tau_1) & \cdots & T_0(\tau_{M-1}) & T_0(\tau_M) \\ T_1(\tau_0) & T_1(\tau_1) & \cdots & T_1(\tau_{M-1}) & T_1(\tau_M) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ T_{N-1}(\tau_0) & T_{N-1}(\tau_1) & \cdots & T_{N-1}(\tau_{M-1}) & T_{N-1}(\tau_M) \end{bmatrix}}^{T}}_{N\times(M+1)} \underbrace{\overbrace{\begin{bmatrix} \frac{1}{2} & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \frac{1}{2} \end{bmatrix}}^{W}}_{(M+1)\times(M+1)}}_{N\times(M+1)}}^{A} \underbrace{\overbrace{\begin{bmatrix} \boldsymbol{g}(\tau_0) \\ \boldsymbol{g}(\tau_1) \\ \vdots \\ \boldsymbol{g}(\tau_{M-1}) \\ \boldsymbol{g}(\tau_M) \end{bmatrix}}^{g}}_{(M+1)\times 1} \right]}^{F}_{N\times 1}. \tag{3.23}$$

where $p = 1$ or $p = 2$ for $M = N$ or $M > N$ respectively.

This $\boldsymbol{F}^{i-1}$ vector is now used in Eq. 3.22 to obtain the vector of $\boldsymbol{\beta}^i$ coefficients. Thus the procedure for computing the $\boldsymbol{F}^{i-1}$ vector of coefficients is: $\boldsymbol{F}^{i-1} = VTW\boldsymbol{g}$. The derivation presented over the last few pages leads to the following equation for calculating the $\boldsymbol{\beta}^i$ coefficients,

$$\boldsymbol{\beta}^i = RS\boldsymbol{F}^{i-1} = RSVTW\boldsymbol{g}. \tag{3.24}$$

Since the product $RSVTW$ is constant (for N fixed), it obviously maps the calculation of the force $\boldsymbol{g}$ directly into the new Chebyshev coefficients of the updated Picard trajectory approximation. The product $RSVTW$ can be computed once and stored as a single matrix, however it is more convenient to group, compute and store the matrix product as the "Chebyshev fit matrix" $C_f = VTW$, and the "Chebyshev integration matrix" $C_{I1} = RS$. The reason for storing the matrix product separately is that some problems require having the intermediate "fit" result available for other computations. Thus the i^{th} position coefficients and position vector are respectively computed as:

$$\boldsymbol{\beta}^i = C_{I1}C_f\boldsymbol{g}(\boldsymbol{X}^{i-1}) + \boldsymbol{X_0} \tag{3.25}$$

$$\boldsymbol{X}^i = C_x\boldsymbol{\beta}^i. \tag{3.26}$$

The "initial condition matrix" is $\boldsymbol{X_0} = col\left[x_0, 0, 0, ..., 0\right] \epsilon R^{M+1}$, the "Chebyshev interpolation matrix" is C_x, and the solution is given by$\boldsymbol{X}^i = col\left\{x^i\left(\tau_0\right), ..., x^i\left(\tau_M\right)\right\}$. The Chebyshev interpolation matrix is essentially $T(\tau)$ where the values of τ are chosen in such a way that the solution is output at the user desired times. This is not necessarily a cosine distribution, in fact when an ephemeris is required the desired output will be uniformly spaced time steps. The result in Eqs 3.25 and 3.26 is the MCPI numerical integration method for solving first order ODEs.

3.5 Second Order Picard Method

A very similar approach is employed for solving second order ODEs. The Picard iteration for solution of a second order system is given in Eq. 3.27, where the acceleration $\boldsymbol{g}(\boldsymbol{x}^{i-1}, \boldsymbol{v}^{i-1})$, that can be a function of position $\boldsymbol{x}^{i-1}$ and velocity $\boldsymbol{v}^{i-1}$ along the previous iteration, is integrated twice.

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}_0 + \int_{-1}^{\tau} \left\{ \boldsymbol{v}_0 + \int_{-1}^{s} \boldsymbol{g}(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q))dq \right\} ds \quad i = 1, 2, ...ds \quad i = 1, 2, ... \tag{3.27}$$

In summation notation this may be written as

$$\boldsymbol{v}^i(\tau) = \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(\tau) = \boldsymbol{v}(-1) + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{F}_k^{i-1} T_k(q)dq, \tag{3.28}$$

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \left\{ \boldsymbol{v}(-1) + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{F}_k^{i-1} T_k(q)dq \right\} ds. \tag{3.29}$$

Note the integrand approximation for Picard iteration is applied to velocity, but the velocity approximation is integrated analytically (without further integrand fitting) to obtain the kinematically consistent position approximation.

Analogous to the first order case the coefficients for the velocity can be computed as follows:

$$\boldsymbol{\beta}_0^i = \boldsymbol{v}_0 + \sum_{k=1}^{k=N-1} (-1)^{k+1} \boldsymbol{\beta}_k^i, \tag{3.30}$$

$$\boldsymbol{\beta}_1^i = \frac{1}{2}\left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1}\right), \tag{3.31}$$

$$\boldsymbol{\beta}_k^i = \frac{1}{2k}\left(\boldsymbol{F}_{k-1}^{i-1} - \boldsymbol{F}_{k+1}^{i-1}\right), \ k = 1, 2, ..., N-1, \tag{3.32}$$

$$\boldsymbol{\beta}_{N-2}^i = \frac{\boldsymbol{F}_{N-3}^{i-1}}{2(N-2)}, \tag{3.33}$$

$$\boldsymbol{\beta}_{N-1}^{i} = \frac{\boldsymbol{F}_{N-2}^{i-1}}{2(N-1)}. \tag{3.34}$$

The coefficients for position can be computed as follows:

$$\boldsymbol{\alpha}_0^i = \boldsymbol{x}_0 + \sum_{k=1}^{k=N} (-1)^{k+1} \boldsymbol{\alpha}_k^i, \tag{3.35}$$

$$\boldsymbol{\alpha}_1^i = \frac{1}{2}\left(2\boldsymbol{\beta}_0^i - \boldsymbol{\beta}_2^i\right), \tag{3.36}$$

$$\boldsymbol{\alpha}_k^i = \frac{1}{2k}\left(\boldsymbol{\beta}_{k-1}^i - \boldsymbol{\beta}_{k+1}^i\right),\ k = 1, 2, ..., N-1, \tag{3.37}$$

$$\boldsymbol{\alpha}_{N-1}^i = \frac{\boldsymbol{\beta}_{N-2}^i}{2(N-1)}, \tag{3.38}$$

$$\boldsymbol{\alpha}_N^i = \frac{\boldsymbol{\beta}_{N-1}^i}{2N}. \tag{3.39}$$

The main difference is that the approximation of the forcing function is done to degree $N-2$ instead of $N-1$ as integration will be performed twice, and instead of just one integration matrix C_{I1}, there are two matrices, one that allows the coefficients of the forcing function (acceleration) to be computed in terms of the coefficients of the velocity C_{I1}, and one that allows the coefficients of the velocity to be computed in terms of the coefficients of the position C_{I2}. The C_{I2} integration matrix is one column larger than C_{I1}. The second order MCPI-IVP can be formulated as follows, where $\boldsymbol{\beta}^i$ are the velocity coefficients, $\boldsymbol{V}_0 = col\,[v_0, 0, 0, ..., 0]\,\epsilon R^{M+1}$ is the "initial condition matrix" for velocity, $\boldsymbol{\alpha}^i$ are the position coefficients, and $\boldsymbol{V}^i$ and $\boldsymbol{X}^i$ are the respective velocity of position solutions. Note that there is no re-fitting of the velocity in this formulation. The position coefficients are computed directly from the coefficients of the velocity, thus maintaining kinematic consistency.

$$\boldsymbol{\beta}^i = C_{I1} C_f \boldsymbol{g}(\boldsymbol{X}^{i-1}, \boldsymbol{V}^{i-1}) + \boldsymbol{V}_0 \tag{3.40}$$

$$\boldsymbol{V}^i = C_v \boldsymbol{\beta}^i. \tag{3.41}$$

$$\boldsymbol{\alpha}^i = C_{I2} \boldsymbol{\beta}^i + \boldsymbol{X}_0 \tag{3.42}$$

$$\boldsymbol{X}^i = C_x \boldsymbol{\alpha}^i. \tag{3.43}$$

For specified Chebyshev polynomial order and node location, the matrices $C_f, C_{I1}, C_{I2}, C_v, C_x$ are constant, so a simple matrix multiplication operates on the new force vector $\boldsymbol{g}(\boldsymbol{X}^{i-1}, \boldsymbol{V}^{i-1})$ to update the trajectories $\boldsymbol{V}^i$ and $\boldsymbol{X}^i$ at all nodes. This calculation is carried out at each i^{th} step of the iteration. Each time the $\boldsymbol{\beta}^i$ and $\boldsymbol{\alpha}^i$ coefficients are updated to define the new trajectory approximation for use in the integrand Eq. 3.11 for the next step $(i+1)$. The the solutions are iteratively improved until some accuracy requirements are satisfied.

3.6 MCPI Boundary Value Problem

The MCPI-TPBVP algorithm is the first of the perturbed solvers that form part of the ULT in this dissertation. The formulation presented herein is similar to the standard MCPI-IVP algorithm in that the trajectory is computed from coefficients, however since the initial velocity is unknown, the first two coefficients in the series must be constructed in a way that bypasses the unknown initial velocity and instead enforces the known initial and final boundary conditions. The entire set of resulting position coefficients does not require a priori knowledge of the initial velocity, thus allowing TPBVPs to be solved without using a Newton-type shooting method [17]. This method is convergent over about one third of an orbit and it may be extended to about ninety percent of an orbit with regularization (see Chap 5 and ref [19]). To solve the Lambert problem over larger arcs a shooting method (such as MPS) must be used (see Chap 6 and [21]), and we must admit multiple local solutions.

We present a new formulation to compute the coefficients that differs from Bai's

[18] formulation, and follows the developments of the kinematically consistent MCPI-IVP Picard iteration developed by Macomber [40]. It is mathematically equivalent to the standard formulation but is attractive as it can be constructed in a sparse matrix that can be easily coded.

The MCPI-TPBVP method is formulated as a "cascade". That is the first integration allows the velocity coefficients ($\boldsymbol{\beta}^i$) to be obtained in terms of the acceleration coefficients ($\boldsymbol{F}^{i-1}$), and the second kinematically consistent, exact integration allows the position coefficients ($\boldsymbol{\alpha}^i$) to be obtained in terms of the velocity coefficients ($\boldsymbol{\beta}^i$). Put another way, the first integration is an integration of the acceleration Chebyshev fit, the second integration is a simple quadrature and is kinematically consistent. The coefficients for these two steps are shown below. Note the repeating pattern that allows these coefficients to be formulated into a sparse matrix for easy coding. Note however, that the unknown initial velocity appears in the coefficients of $\boldsymbol{\alpha}_0^i$, $\boldsymbol{\alpha}_1^i$ and $\boldsymbol{\beta}_0^i$. These two coefficients can be reformulated in terms of the initial and final position vectors and thus bypass the unknown initial velocity. The velocity coefficients are computed as follows:

$$\boldsymbol{\beta}_0^i = \boldsymbol{v}_0 + \sum_{k=1}^{k=N-1} (-1)^{k+1} \boldsymbol{\beta}_k^i, \tag{3.44}$$

$$\boldsymbol{\beta}_1^i = \frac{1}{2}\left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1}\right), \tag{3.45}$$

$$\boldsymbol{\beta}_k^i = \frac{1}{2k}\left(\boldsymbol{F}_{k-1}^{i-1} - \boldsymbol{F}_{k+1}^{i-1}\right),\ k = 1, 2, ..., N-1, \tag{3.46}$$

$$\boldsymbol{\beta}_{N-2}^i = \frac{\boldsymbol{F}_{N-3}^{i-1}}{2(N-2)}, \tag{3.47}$$

$$\boldsymbol{\beta}_{N-1}^i = \frac{\boldsymbol{F}_{N-2}^{i-1}}{2(N-1)}. \tag{3.48}$$

The position coefficients are computed as follows:

$$\boldsymbol{\alpha}_0^i = \boldsymbol{x}_0 + \sum_{k=1}^{k=N} (-1)^{k+1} \boldsymbol{\alpha}_k^i, \tag{3.49}$$

$$\boldsymbol{\alpha}_1^i = \frac{1}{2} \left(2\boldsymbol{\beta}_0^i - \boldsymbol{\beta}_2^i \right), \tag{3.50}$$

$$\boldsymbol{\alpha}_k^i = \frac{1}{2k} \left(\boldsymbol{\beta}_{k-1}^i - \boldsymbol{\beta}_{k+1}^i \right), \; k = 1, 2, ..., N-1, \tag{3.51}$$

$$\boldsymbol{\alpha}_{N-1}^i = \frac{\boldsymbol{\beta}_{N-2}^i}{2(N-1)}, \tag{3.52}$$

$$\boldsymbol{\alpha}_N^i = \frac{\boldsymbol{\beta}_{N-1}^i}{2N}. \tag{3.53}$$

The first two position coefficients ($\boldsymbol{\alpha}_0^i$ and $\boldsymbol{\alpha}_1^i$) that contain the unknown initial velocity are computed using the known initial and final position as follows:

$$\boldsymbol{x}(-1) = \sum_{k=0}^{k=N} \boldsymbol{\alpha}_k^i T_k(-1) \tag{3.54}$$

$$\boldsymbol{x}(1) = \sum_{k=0}^{k=N} \boldsymbol{\alpha}_k^i T_k(1) \tag{3.55}$$

This leads to

$$\boldsymbol{\alpha}_0^i - \boldsymbol{\alpha}_1^i + \boldsymbol{\alpha}_2^i + ... + (-1)^N \boldsymbol{\alpha}_N^i = \boldsymbol{x}(-1) \tag{3.56}$$

$$\boldsymbol{\alpha}_0^i + \boldsymbol{\alpha}_1^i + \boldsymbol{\alpha}_2^i + ... + \boldsymbol{\alpha}_N^i = \boldsymbol{x}(1) \tag{3.57}$$

The first two coefficients can be recovered as shown.

$$\boldsymbol{\alpha}_0^i = \boldsymbol{x}(1) + \boldsymbol{x}(-1) - \left(\boldsymbol{\alpha}_2^i + \boldsymbol{\alpha}_4^i + \boldsymbol{\alpha}_6^i + ... \right) \tag{3.58}$$

$$\boldsymbol{\alpha}_1^i = \boldsymbol{x}(1) - \boldsymbol{x}(-1) - \left(\boldsymbol{\alpha}_3^i + \boldsymbol{\alpha}_5^i + \boldsymbol{\alpha}_7^i + ...\right) \tag{3.59}$$

The position solution is obviously obtained by multiplying the position coefficients ($\boldsymbol{\alpha}^i$) by the Chebyshev polynomials, where the values of τ are selected by the user in order to and provide the solution at desired discrete time locations. A "pseudo" velocity ("Δv" integral of acceleration) can be determined from the current velocity coefficients ($\boldsymbol{\beta}^i$) in a similar manner, however it will be offset by the unknown initial velocity constant. This constant must first be solved for from the position solution. The procedure for doing this is to equate the two $\boldsymbol{\alpha}_0^i$ coefficients shown in Eq. 3.57 and Eq. 3.58. The first one is essentially the "IVP" coefficient and the second is the "BVP" coefficient. Equating these clearly reveals that the unknown initial velocity appears linearly and thus can be easily computed. Once this constant is added to the "Δv" pseudo velocity coefficients ($\boldsymbol{\beta}^i$) the true velocity may be obtained.

The matrix representation of the MCPI-TPBVP is similar to that of the MCPI-IVP but differs in that the initial velocity must be reconstructed for the known final boundary condition. The "pseudo" velocity coefficients are computed as

$$\boldsymbol{\beta}^i = C_{I1} C_f \boldsymbol{g}(\boldsymbol{X}^{i-1}, \boldsymbol{V}^{i-1}), \tag{3.60}$$

where $\boldsymbol{g}(\boldsymbol{X}^{i-1}, \boldsymbol{V}^{i-1})$ is the forcing function (acceleration), C_f is the "Chebyshev fit matrix", and C_{I1} is a sparsely populated "integration matrix". The position coefficients are computed as

$$\boldsymbol{\alpha}^i = C_{B2} C_{I1} C_f \boldsymbol{g}(\boldsymbol{X}^{i-1}, \boldsymbol{V}^{i-1}) + \boldsymbol{X}_{0f}. \tag{3.61}$$

C_{B2} is the sparsely populated integration matrix. $\boldsymbol{X}_{0f}$ is a vector with the first two components being $\boldsymbol{x}_f + \boldsymbol{x}_0$ and $\boldsymbol{x}_f - \boldsymbol{x}_0$ respectively. The remaining elements in this vector are zeros. The position solution is computed as

$$\boldsymbol{X^i} = C_x \boldsymbol{\alpha}^i, \tag{3.62}$$

and the velocity solution is computed as

$$\boldsymbol{V^i} = C_x \boldsymbol{\beta}^i + C_x \boldsymbol{V}_0. \tag{3.63}$$

The "pseudo velocity" $C_x \boldsymbol{\beta}^i$ is the integrand of acceleration and differs from the velocity only through the velocity initial condition. $\boldsymbol{V}_0$ is a vector where the first element is $\boldsymbol{v}_0$ and the remaining elements are zeros. $\boldsymbol{v}_0$ is obtained from equating the $\boldsymbol{\alpha}_0^i$ coefficients as described above.

3.7 Chapter Summary

The formulations for the MCPI-IVP and MCPI-BVP were presented in this chapter. An obvious, but simultaneously subtle truth, even though the differential equation is nonlinear, is that in this implementation of Picard iteration the constants of integration are *always* rigorously linearly contained as un-determined coefficients in the $\boldsymbol{\alpha}_0^i$, $\boldsymbol{\alpha}_1^i$, $\boldsymbol{\beta}_0^i$ that can be computed to rigorously satisfy the boundary conditions on each iteration. The fact that a TPBVP can be solved with iteration (on non-linearly contained parameters) is an important feature of this approach. It turns out that there are exactly the required number ($2n$ for a system of n second order differential equations, n for a system of n first order differential equations) of linearly contained free constants on each iteration to specify any combination of initial position and velocity or terminal position and velocity as boundary conditions. As a consequence, any combination of initial and terminal linear boundary conditions can be imposed exactly, *on each Picard iteration.* So the distinction between solving an

IVP and a TPBVP reduces to how the first one or two linearly contained coefficient vectors in the Chebyshev series are determined as a function of boundary conditions. This truth enables TPBVPs to be solved without a shooting method (based on local linearizations with respect to initial conditions). The specified boundary conditions are enforced on the solution on each iteration. This is accomplished for TPBVPs using a close cousin of the same Picard algorithm used to solve IVPs. Picard iterations converge analogously to refine path approximations whether solving an IVP or a TPBVP. However, as reported by [18], and consistent with our results in Chapter 5, the details of how the boundary constants enter the Picard formulation *does affect the time interval over which the Picard sequence converges* to the path satisfying the governing differential equations. The IVP version of Picard iteration typically has about one order of magnitude larger maximum time interval for convergence than that for the corresponding TPBVP. Quite apart from numerical experiments, on theoretical grounds, we can anticipate that Picard iteration methods will necessarily fail for multi-revolution Lambert problems (known to have multiple solutions) because the Picard convergence theorem requires the *unknown solution to be the unique solution of the differential equations that satisfies the prescribed boundary conditions.* It is well known that multi-revolution TPBVP in celestial mechanics have multiple solutions for prescribed boundary position vectors and time of flight, while Lambert's theorem tells us that *the solution is unique for the fractional orbit case.* Chapter 5 sheds much more light on this subject and ultimately leads to an important set of algorithms.

4. RECENT MCPI ENHANCEMENTS

A brief overview of some recent enhancements made to the MCPI algorithm by the research group at Texas A&M is presented in this chapter. Much of the work presented in this chapter is a summary of the following two papers [41, 42], to which Woollands made significant contributions as a co-author. Some of this work also formed part of Bani Younes and Macomber's PhD dissertations [22, 40]. These enhancements to the general MCPI algorithm are further refined in this dissertation and thus deserve a brief explanation before proceeding.

4.1 Historical Overview

The original fusion of orthogonal approximation theory and Picard iteration was introduced by Clenshaw and Norton in 1963 [39]. Feagin also contributed in this area; his PhD dissertation on Picard Iteration using Chebyshev Approximation established the first vector-matrix version of Picard iteration utilizing orthogonal basis functions, in 1973 [43]. In 1980, Shaver wrote a related dissertation giving insights on parallel computation in an early parallel computer architecture using Picard Iteration and Chebyshev approximation [44]. Fukushima [45] also addressed parallelization of Picard iteration and again in a particular software and computer architecture, however his parallel implementation paradoxically did not result in a computational speedup.

Bai and Junkins revisited the Picard-Chebyshev approach and developed improved algorithms for solving IVPs and TPBVPs [46, 17]. They established new convergence insights (much less conservative that classical bounds) and optimized the solution of IVPs utilizing vector-matrix formulations that are published in Bai's PhD dissertation [18]. Bai and Junkins [46, 17] also showed that MCPI, when applied to the state/co-state differential equations derived from Pontryagin's principle, led to an accurate and efficient means for indirect calculus of variations based on trajectory optimization. Their trajectory optimization solution, however, converged only over about $1/3$ of an orbit and was based on an unusual performance metric. The unique indirect optimal control problem posed by Bai involved a hybrid impulsive/continuous admissible control and the performance index penalized *only* the boundary impulsive velocity increments - the result was a boundary value problem with the unknown initial costates contained linearly in the MCPI-BVP formulation, and being uniquely related to the velocity boundary conditions. The main drawback in Bai's optimal control formulation is the relatively small interval over which the resulting TPBVP associated with the stiff state/co-state differential equations are solvable using Picard iteration. This MCPI optimal control algorithm was compared to a pseudo spectral solution of the same example problem and showed improved accuracy, efficiency and robustness of convergence. These results used polar coordinates. Woollands redid the problem solved by Bai using Cartesian coordinates in three dimensions and found analogous convergence properties [47]. Other coordinates and associated differential equations require further investigation. We also mention that, in general, when an admissible control and performance index is used, we find that the initial costates are not always linearly contained in MCPI. If the costates are contained in a nonlinear fashion, a shooting method is needed. Thus a Picard-Chebyshev approach for a general indirect optimal control problem remains an elusive problem.

Following this work, Bani Younes and Junkins developed methods to include high order gravity perturbations [22, 48, 49, 50] to more efficiently compute the motion of satellites orbiting in the vicinity of the Earth. Their work was aimed at efficient solution of IVPs in the presence of general perturbations. Bani Younes gave special attention to finite element gravity models to speed up gravity computation in MCPI.

Since then further enhancements have been made by several investigators at Texas A&M. Recent work includes variable fidelity and radially adaptive gravity approximations [42, 51, 41], segmentation/order tuning [52], multi-orbit accuracy and numerical stability studies [53], Kustaanheimo-Stiefel regularization [54], and using the method of particular solutions [20] to solve TPBVPs [21]. These studies also include significant benchmarks of MCPI algorithms as compared to traditional high order integrators such as RK(12)10 and Gauss-Jackson 8^{th} order (perhaps the most important state-of-the-practice algorithm for celestial mechanics).

4.2 Segmentation

While MCPI converges over a large time domain, more than an orbit for the perturbed case [22] and up to three orbits for the unperturbed case [18], multi-day, week, or longer time intervals obviously require multiple time segments for the low Earth orbit (LEO) case. Even though accurate Picard convergence can be achieved over fairly large intervals, using the longest convergent interval is not necessarily the most efficient approach for solving these problems. As the domain over which we seek convergence increases, so does the number of nodes required for achieving the desired level of accuracy. It is easy to verify that the rate of convergence of Picard iterations decreases approximately linearly as the time interval of the solution segment is increased, and ultimately fails at some maximum time interval. To minimize computational expense, while constraining accuracy, it necessary to allow both the time span of the segments and the number of nodes per segment to be subject to optimization.

Optimal segmentation and node density for the perturbed two-body problem depends on a number of parameters including eccentricity (e), perigee radius (r_p), the model adopted for perturbing accelerations, and accuracy of the desired solution. The following papers published by the research group at Texas A&M [52, 42, 41] present detailed studies to gain approximate insight into the optimal segmentation scheme for the perturbed two-body problem. One result is a multidimensional surface which can be accessed with the initial conditions (mapped into the initially osculating perigee radius and eccentricity), the maximum degree of the gravity model, and the desired number of significant figures in the final trajectory. We can find the interpolated output for the orbit segmentation break points (in time) and the number of nodes for the application of MCPI. This tuning approach is designed to be conservative and enforces symmetry with an odd number of segment break points per orbit (1,3,5,...). A key issue is to maintain approximately uniform accuracy across all segments. This tuning scheme also ensures that sub-optimal nodal density is consistent with the physics to avoid redundant and superfluous nodes near apogee that may otherwise occur. Obviously, following a segment accurate to (say) 10^{-8} relative error by a more accurate segment locally accurate to (say) 10^{-10} relative error is simply wasting computation time. We can never regain in subsequent segments accuracy sacrificed in a poorly tuned earlier segment. We mention, that when the force model differs from that used to tune the segment breaks and node density, we can expect this tuning to be sub-optimal. We may therefore need further adaption to efficiently maintain a prescribed relative error.

Figure 4.1 shows a Molniya orbit (12 hour period) with five segments. The multi-segment solutions are straightforward, and the end state of the m^{th} segment becomes the initial state for the $(m+1)^{th}$ segment. The number of nodes per segment is held constant in this example. Note that the segments closer to perigee span a shorter time duration and also have a higher spatial node density. This is a result of symmetric segment breaks as a function of eccentricity consistent with Kepler's second law - time breaks are optimized as symmetric true anomaly breaks associated with the

Keplerian orbit that osculates at perigee of each revolution. This automatically generates higher node density at perigee where the object reaches maximum velocity and the nonlinear variations in the gravity field are most significant. Properly tuned, this approach leads to approximately uniform errors on a given time segment and also, approximately identical segment-to-segment error norms.

The study [40] generated a database of sub-optimal segmentation parameters corresponding to various osculating two-body trajectories. With the initial conditions one may simply extract the nominal segmentation scheme from the database prior to integrating the equations of motion. In a way, this is similar to GJ and RKN(12)10 where the pre-calculated table of differences and interpolation coefficients and user insight are used to prescribe the nominal time step at the beginning, prior to integration. For example, for all satellites in the vicinity of the Earth, i.e. for space situational awareness studies, a one off data base can be generated that spans the range of all possible orbits to permit near optimal performance. For other types of trajectories, i.e. interplanetary missions, other approaches for determining optimal segmentation must be utilized.

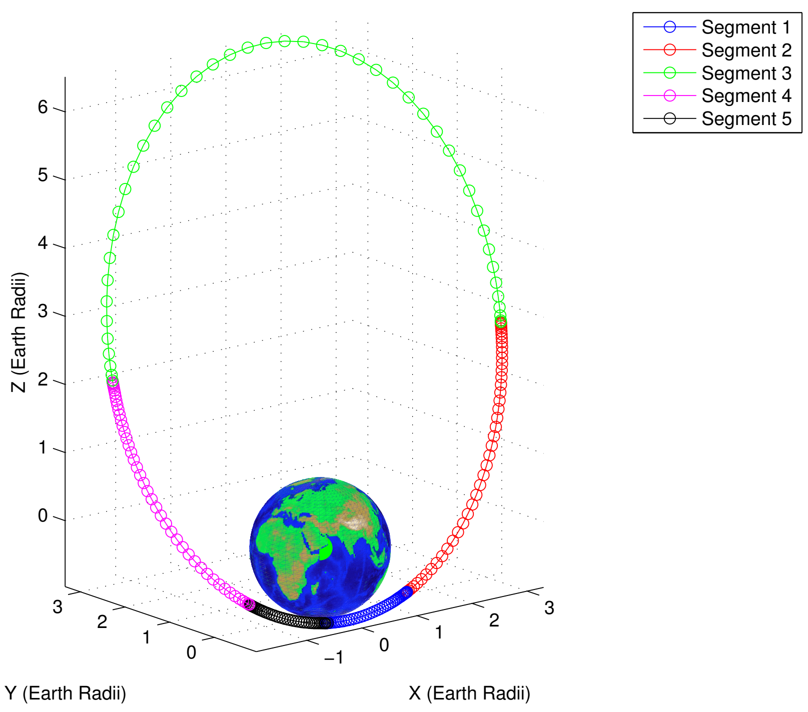


Figure 4.1: A 12 hour Molniya orbit is propagated using MCPI in five segments.

4.3 Warm/Hot Starts

A two-body *warm start*, computed by means of Battin's [1] analytical *F&G* solution (IVP) or *a*-iteration (TPBVP), is used to start the first MCPI iteration. We note that the *F&G* solution here refers to the exact transcendental function analytical expression [1] for *F&G* not the, perhaps more familiar, poorly converging time power series. This warm start immediately brings the relative accuracy down to the $< 1 \times 10^{-3}$ relative error (region for segments $\leq$ 1 orbit) and greatly reduces the number of Picard iterations required for convergence. A hot start is used on all subsequent orbits (taking dominant zonal perturbations into account), bringing the relative accuracy down another two orders of magnitude prior to iterating with the high degree and order gravity model. The hot start is computed using the converged solution on the previous orbit. At the end of the previous orbit, the difference between the converged solution and the two-body analytical warm start solution is calculated. This is then added to the two-body (warm start) on the following orbit to produce a hot start. This hot start is conceptually an approximate Encke displacement and is the net departure motion due to all non-two-body perturbations on the previous orbit. It is most effective for LEO because of the high correlation of the orbit-to-next-orbit perturbations, and is implemented generally in the MCPI algorithm. For the case of zonal perturbations and static atmosphere rotating uniformly with the Earth, this approach turns out to be an excellent hot start, with approximation errors usually to the right of the 6^{th} significant digit. Even in the most general gravity field and drag model, for low eccentricity LEO orbits, the errors are usually to the right of the 5^{th} significant digit. The warm/hot start concept is illustrated heuristically in Figure 4.2. This hot start is but one hot start method, we can also make use of Brouwer theory to approximate the effect of the zonal harmonics and cannonball drag.

4.4 Gravity Approximations

A spacecraft in an atmosphere-skimming orbit with a perigee radius of about 1.04 Earth radii experiences far greater perturbative gravitational effects compared with those experienced at an apogee radius of (say) 7 Earth radii. Close to the Earth the gravitational acceleration is affected much more by the higher degree and order gravity terms, and also changes rapidly along an eccentric orbit due to the more rapid motion near perigee. To compute gravitational acceleration with significant figure accuracy at 1.04 Earth radii we require degree and order > 200 in the spherical harmonic series, whereas at most degree and order 6 is typically required for

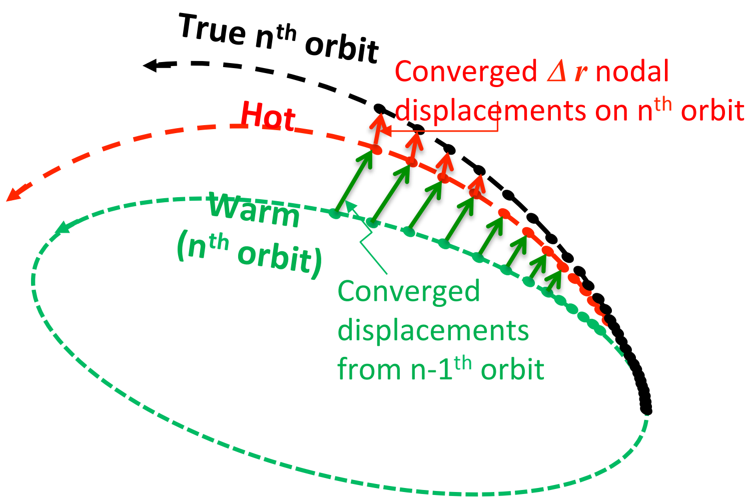

True n^th orbit
Converged Δ r nodal displacements on n^th orbit
Hot
Warm (n^th orbit)
Converged displacements from n-1^th orbit

Figure 4.2: Schematic demonstrating the Warm/Hot Start approach [41].

geostationary orbits (GEO). Only degree and order 10 is required to compute 16 digit acceleration at 7 Earth radii. Obviously we must distinguish arithmetic precision from physical accuracy. Thus it is essential to consider the full, computationally more expensive and more "dimpled" gravity field, gravity model. Since the strength of the gravitational field decreases by $\frac{1}{r^n}$, the perturbative effects are far smaller at apogee and a lower fidelity gravity model is required. We take advantage of this physical truth and apply radial adaptation as illustrated qualitatively in Figure 4.3. In Figure 4.4 we present a Molniya orbit that shows the equivalent gravity spherical harmonic degree required for approximating a (40,40) degree and order gravity field. This curve is just for illustration; it has the same qualitative behavior for $(200, 200)$ gravity models. Note that near apogee the spacecraft is far enough from the Earth that the same level of accuracy is obtained by performing the calculation with a much lower degree model. If only a physical gravitational acceleration accuracy of 8 digits is desired then only a 4^{th} degree expansion is required at GEO. Radial adaptation ensures that the appropriate fidelity model is used to achieve the required accuracy in the minimum possible computation time; it avoids the computation of terms that can be known a priori to be negligible.

In addition to applying radial adaptation it is also not necessary to evaluate gravity, even the computationally cheaper radially adapted gravity, on every single iteration. Since Picard iteration requires multiple path approximations we can tune the fidelity of the force model to match the evolving accuracy associated with the rate of Picard convergence. Also, the final few iterations are extremely near the previous iterations, and this motivates the use of local force approximations in the very near vicinity of the nodes of the previous iterations.

The numerical process adopted begins by employing a simplified model that includes the analytical two-body solution and the $J2$ to $J6$ zonals [1]. After a specific tolerance is met (1×10^{-3}) the radially adapted gravity is called and one iteration is completed with this high fidelity gravity (*EGM2008*). At this time the local *difference* (Δg) at each node, between high fidelity gravity and two-body plus $J2$ to $J6$ is computed. Subsequent iterations are carried out by once again calling the simple two-body plus $J2$-$J6$ model and adding the difference (Δg). Every fourth iteration the high fidelity model is called again and the same subtraction procedure is performed. Each time bringing the trajectory closer to convergence until the final tolerance is met. While this process is heuristic, we have validated it and showed we achieve both high accuracy and efficiency. It can be shown to be the first term in the local Taylor series about each node of previous iteration. In general, about three high fidelity gravity calls are made at the nodes of a segment spanning a third of a LEO orbit to achieve a machine precision Hamiltonian. These local approximations together with resulting variable fidelity gravity model create significant speedups as documented by [42].

Figure 4.5 depicts not only the variable fidelity gravity model, but also demonstrates the nature of the warm/hot start. Convergence patterns are plotted for each of the five segments of the Molniya orbit, for a duration of two orbits. The warm and hot starts are applied to the first and second orbit respectively. Each curve corresponds to a particular segment of the orbit. The blue, green and magenta dots represent the $J2$-$J6$, low and high fidelity gravity, respectively. Note the reduced starting relative tolerance for the red trajectories (second orbit) with respect to the black trajectories (first orbit), due to implementation of the hot start. As is evident, only one "full" gravity computation is required on each segment for the hot-start orbits, thus drastically reducing multi-revolution propagation cost. We have verified that negligible accuracy loss is incurred compared to using full-gravity on all force computations. These enhancements reduce the cost by over one order of magnitude. Multi-orbit propagation is where the hot start gives MCPI a significant advantage.

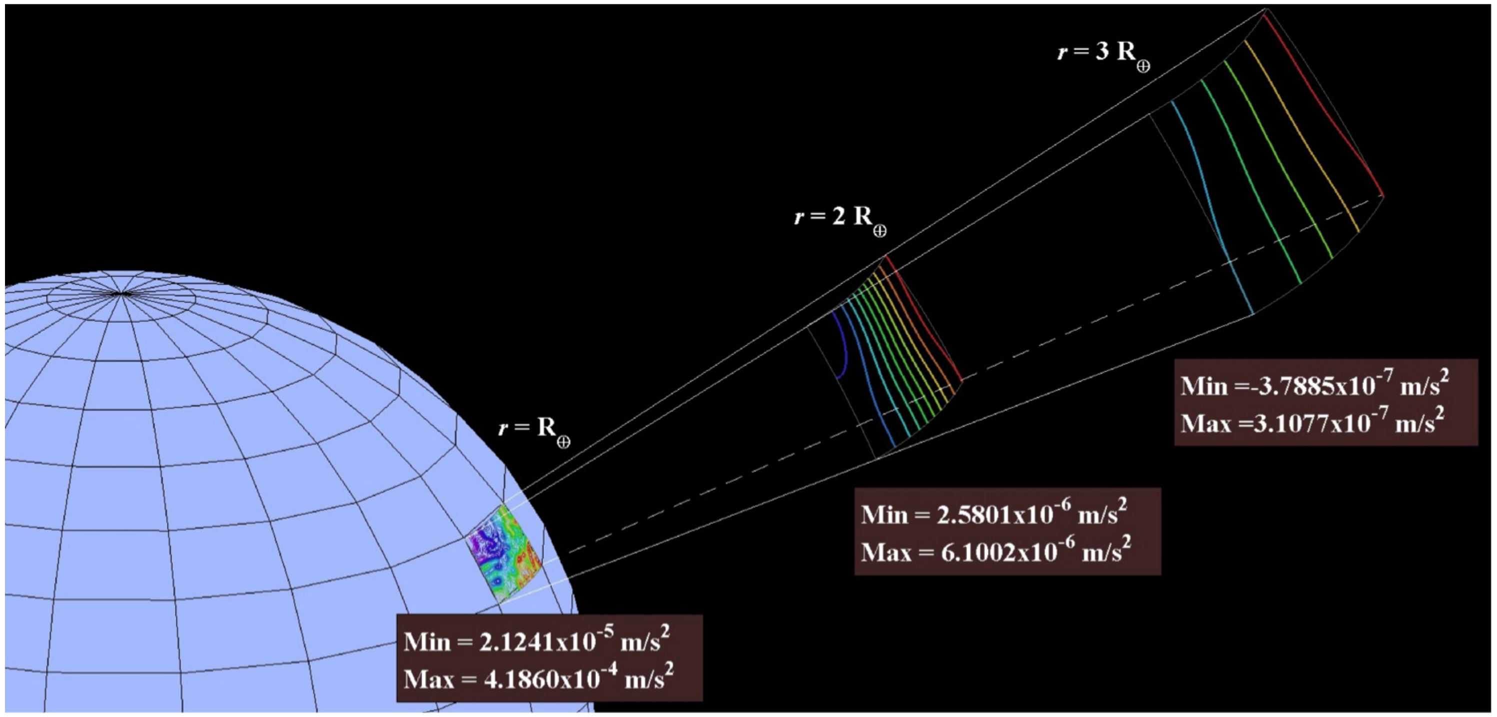

$r = 3\ R_\oplus$
$r = 2\ R_\oplus$
$r = R_\oplus$
Min =-3.7885x10^{-7} m/s^2
Max =3.1077x10^{-7} m/s^2
Min = 2.5801x10^{-6} m/s^2
Max = 6.1002x10^{-6} m/s^2
Min = 2.1241x10^{-5} m/s^2
Max = 4.1860x10^{-4} m/s^2

Figure 4.3: Schematic depicting the decrease in the intensity of the gravity field as distance from the Earth increases [22].

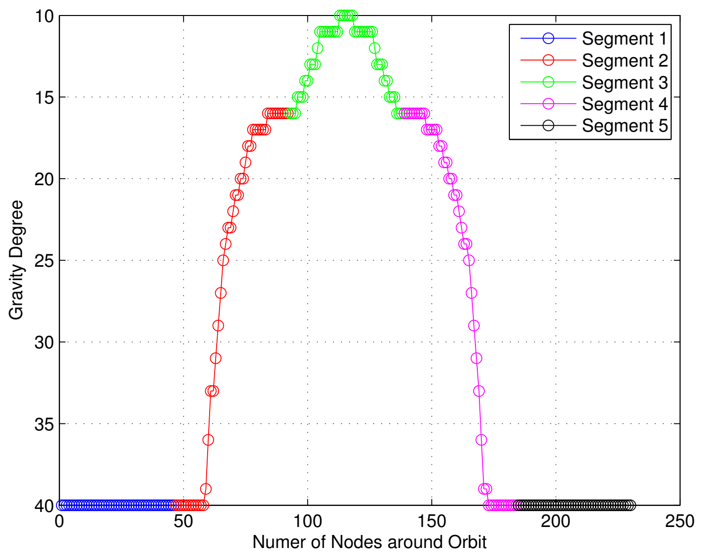


Figure 4.4: Gravity degree required over a Molniya orbit to produce a machine precision gravity acceleration.

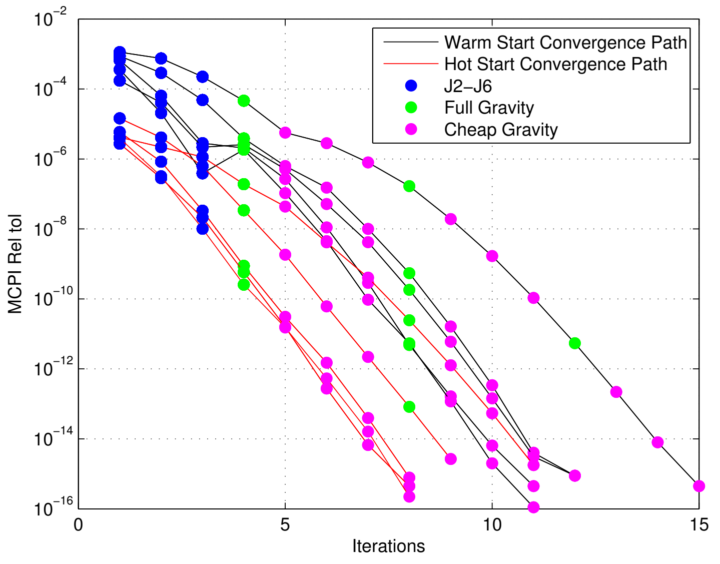


Figure 4.5: The number and type of iterations required for MCPI convergence.

4.5 Computational Speedup

It is important to make the distinction between the present implementation of MCPI, and other related attempts to fuse orthogonal approximation and Picard iteration, that claim poor performance statistics compared with various other numerical integrators [55]. The algorithm utilized in the present study incorporates insights from [42, 53, 41, 52] that collectively separate it from the independently programmed "MCPI" algorithms utilized in [55]. One of the key speedups is achieved through local force approximation that takes full advantage of the fixed-point nature of Picard iteration; the approximation nodes quickly converge to the close neighborhood of fixed points in the force field. Most fundamentally, some care is required for integration methods (including MCPI) that have two or more parameters that control efficiency, accuracy and stability. For example, simply holding fixed the time interval for path approximation as one orbit period will lead to badly sub-optimal results. The algorithms in [52, 42, 51, 41] address the proper tuning and demonstrate excellent stability, precision, and efficiency as compared to state of the practice algorithms.

The MCPI method is a fixed point algorithm, thus every nodal point ultimately converges to a fixed point in the force field. The terminal iterations are in the very near vicinity of a fixed point and following [22], we know local gravity can be accurately approximated with very inexpensive local algorithms. As a consequence of the local force approximations, variable fidelity gravity models as described above can be introduced that are several orders of magnitude less expensive than brute force re-computation of the globally valid spherical harmonic series on subsequent neighboring Picard iterations.

4.6 Chapter Summary

A brief overview of some recent enhancements made to the MCPI algorithm by the research group at Texas A&M was presented in this chapter. Much of the work presented in this chapter was published in the following papers [41, 42], to which Woollands made a significant contributed as a co-author. This work formed part of Macomber's and Bani Younes' respective PhD dissertations. These insights more than compensate for the geometric convergence rate of Picard iteration (by making most of the Picard iterations "very cheap") and lead to an efficient as well as accurate means for orbit propagation. These enhancements to the general MCPI algorithm are utilized and extended in the following chapters of this dissertation.

5. REGULARIZATION*

A new approach is presented here for solving TPBVPs and IVPs using the KS transformation and MCPI. The first section introduces the transformation and gives the main insights and steps for deriving the perturbed two-body equations of motion in KS variables. A full derivation is presented in Appendix E. The second section discusses a special coordinate system that is utilized to avoid certain ambiguities that arise as a result of the non-unique mapping of KS to Cartesian variables. Boundary conditions for this special coordinate choice also permits important insights needed to resolve a consistency and uniqueness issue that arises in redundant KS coordinates. Following this we present a section on an analytical KS Lambert solver that is used as a warm start for solving the perturbed problem. The sections following these developments present key issues with regard to convergence advantages afforded by the KS boundary value formulation, and the numerical results of a study conducted reveal relative merits of three methods for solving the perturbed Lambert's problem. This work was presented at the 37th Annual AAS Guidance & Control Conference [54], and was also published in the Journal of Guidance Control and Dynamics [19] (the special issue in honor of the late R.H. Battin).

5.1 KS Regularization Transformation

Our interest in the KS regularized Lambert's problem is motivated by the results in two classical papers from the 1970s [56] and [57]. Kritz [57] considered the Keplerian unperturbed problem, and Engels and Junkins [56] developed an approx-

*Reprinted with permission from "New Solutions for the Perturbed Lambert Problem Using Regularization and Picard Iteration" by R. Woollands, A. Bani Younes, J. Junkins, 2015. Journal of Guidance, Control and Dynamics, v 38, p 1548-1562, Copyright 2015 by the American Institute of Aeronautics and Astronautics.

imate solution of the $J2$-perturbed Lambert problem in KS variables. The present developments go further, considering general perturbations and novel algorithms for efficiently computing the numerical solutions of generally perturbed Lambert problems in KS variables. The previous perturbed Lambert solvers sought approximate solutions to $O(J_2)$. When extending the formulations to seek high precision Lambert solutions that accommodate general perturbations we encountered and solved certain subtle issues associated with prescribing consistent boundary conditions in the redundant KS coordinates. Furthermore, by avoiding the approximations of the earlier papers and taking advantage of the MCPI developments, near machine precision solution of the two-point boundary-value-problem can be obtained.

The Kustaanheimo-Stiefel (KS) transformation [58] is a method for rigorously linearizing, without local approximation, the TPBVP through a judicious coordinate transformation. We begin by writing the classical differential equations of orbital motion in the most familiar rectangular Cartesian coordinates:

$$\frac{d^2\boldsymbol{r}}{dt^2} = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{F} \tag{5.1}$$

where $\boldsymbol{r} = [X\ Y\ Z\ 0]^T, r = \mid \boldsymbol{r} \mid$. The KS transformation involves transforming both the position coordinates and the independent time variable. The position transformation can be written compactly in matrix form as

$$\boldsymbol{r} = \begin{Bmatrix} X \\ Y \\ Z \\ 0 \end{Bmatrix} = L(\boldsymbol{u})\boldsymbol{u}, \text{ where } L(\boldsymbol{u}) = \begin{bmatrix} u_1 & -u_2 & -u_3 & u_4 \\ u_2 & u_1 & -u_4 & -u_3 \\ u_3 & u_4 & u_1 & u_2 \\ u_4 & -u_3 & u_2 & -u_1 \end{bmatrix}, \quad \boldsymbol{u} = \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}. \tag{5.2}$$

The most frequently used (non-unique) inverse transformation [54, 56, 57] is given by:

$$\boldsymbol{u} = \left\{ \begin{array}{c} \left(\frac{(r+X)}{2}\right)^{\frac{1}{2}} \\ \frac{Y}{(2(r+X))^{\frac{1}{2}}} \\ \frac{Z}{(2(r+X))^{\frac{1}{2}}} \\ 0 \end{array} \right\}. \tag{5.3}$$

The operator $L(\boldsymbol{u})$ has many interesting properties [58, 56, 54] including

$$L^{-1}(\boldsymbol{u}) = \frac{1}{r}L^T(\boldsymbol{u}), \text{ and } L(\boldsymbol{u})\boldsymbol{v} = L(\boldsymbol{v})\boldsymbol{u}. \tag{5.4}$$

It is important to note that the second of Eqs 5.4 is only valid for the planar case for arbitrary $\boldsymbol{u}$ and $\boldsymbol{v}$. Given a 4D $\boldsymbol{u}$-vector, $L(\boldsymbol{u})\boldsymbol{v} = L(\boldsymbol{v})\boldsymbol{u}$ holds only if $\boldsymbol{v}$ satisfies $L_4(\boldsymbol{u})\boldsymbol{v} = 0$, i.e. $\boldsymbol{v}$ is perpendicular to the fourth row of $L(\boldsymbol{u})$. This gives rise to [59] the bilinear constraint ($\phi = L_4(\boldsymbol{u})\boldsymbol{u}' = 0$) and $\phi = 0$ can be verified to be an exact integral of the KS equations of motion developed below. In order for the motion to be physically plausible, the bilinear constraint must be satisfied initially and for all time. This means geometrically that any physically admissible velocity, $\boldsymbol{u}'$, in KS variables will at all times remain orthogonal to $L_4(\boldsymbol{u})$. Also, since $\phi = 0$ is an exact integral of the governing differential equations, the bilinear constraint is (in theory) automatically satisfied for the entire IVP trajectory once the initial position consistent with Eq 5.2 and "orthogonal" initial velocity in KS variables have been chosen. However, it is not automatically satisfied for the TPBVP. If geometrically feasible initial and final $\boldsymbol{u}$ vectors are chosen at random we must think carefully. There are an infinity of choices of feasible position $\boldsymbol{u}$ boundary conditions, so how do we ensure "dynamical feasibility" in the sense that the initial and final $\boldsymbol{u}$'s are actually the initial and final points that lie on the particular trajectory of Eq. 5.11?

Due to the redundancy of the $\boldsymbol{u}$-space, two sets of admissible initial and final boundary conditions exist for any prescribed Cartesian position, and also, an infinity of inertial frames can be chosen! As we show below, a one parameter family of initial and final $\boldsymbol{u}$-points belong to each set (for a given choice on the initial Cartesian axes), satisfying Eqs 5.2 at the initial and final time. It turns out there is a one-to-one correspondence of points (initial and final) in feasible dynamically consistent $\boldsymbol{u}$-space. Mapping the Cartesian coordinates into consistent initial and final $\boldsymbol{u}$-vectors is not trivial, because we find that each point in the feasible initial set flows into a specific dynamically consistent point in the final set. Resolving the ambiguity introduced because an infinite set of $\boldsymbol{u}$'s geometrically correspond to a given (X, Y, Z) is a key aspect of the KS Lambert algorithm, in order to correctly specify terminal points that lie on the same dynamical path in $\boldsymbol{u}$-space. The final boundary conditions reside on a "fiber" [59] with a constant radius $(\sqrt{r}_f)$ locating a point on the geodesic on the four-dimensional (4D) sphere. The feasible final boundary condition on this fiber can not be determined analytically, as we find it is a function of the unique perturbed space through which the trajectory travels, following the selection of the particular geometrically consistent initial conditions. While we can resolve this problem completely for the case of planar motion where the space curve fiber degenerates into two distinct points (because for the planar case only the first two elements of $\boldsymbol{u}$ are needed, the transformation is not redundant), we must resort to numerical methods for general 4D KS dynamics to implicitly select the proper *dynamically consistent* final boundary condition. More on this is given in the following section.

Another interesting property of the KS transformation is: $r = \boldsymbol{u}^T\boldsymbol{u}$, which is easily proven by squaring both sides of Eq 5.2. Obviously, quadratic combinations of the elements of $\boldsymbol{u}$ produce both the rectangular coordinates and the radial distance.

The mapping from $\boldsymbol{u}$ to (X, Y, Z) is unique, for all $\boldsymbol{u}$. However, the inverse from (X, Y, Z) to $\boldsymbol{u}$ (the right most of Eqs 5.3) *is not unique*, and the given inverse transformation is the most popular of the set of inverse mappings.

It is well known that any transformed time coordinate that is linearly proportional to r, together with Eqs 5.2, maps the nonlinear differential equations (Eqs 5.1) into 4 oscillators in $\boldsymbol{u}$-space. The nonlinearities vanish identically as $\boldsymbol{F} \to 0$. The $\boldsymbol{F}$ vector here is the Cartesian perturbing acceleration components augmented with a zeroth fourth element. We restrict attention in the present discussion to the case of perturbed elliptic orbits for which the instantaneous Keplerian energy ($\alpha = 1/a$) is positive, where $a = a(t)$ and we adopt the following implicit time transformation $E \to t$.

$$\frac{dt}{dE} = \left(\frac{1}{\sqrt{\mu\alpha}}\right) r, \quad \alpha = \frac{2}{r} - \frac{\dot{\boldsymbol{r}}^T \dot{\boldsymbol{r}}}{\mu}. \tag{5.5}$$

Note that $\frac{d}{dE} = \frac{d}{dt}\frac{dt}{dE}$, so that, after some manipulation, we can show

$$\boldsymbol{u}' = \frac{d\boldsymbol{u}}{dE} = \frac{1}{2}[\sqrt{\mu\alpha}]^{-1/2} L^T(\boldsymbol{u})\dot{\boldsymbol{r}}. \tag{5.6}$$

After considerable algebra, we derive the the resulting differential equations:

$$\frac{d^2\boldsymbol{u}}{d\boldsymbol{E}^2} + \frac{1}{4}\boldsymbol{u} = \frac{r}{2\mu\alpha}\left[I + \frac{4}{r}\frac{d\boldsymbol{u}}{d\boldsymbol{E}}\frac{d\boldsymbol{u}}{d\boldsymbol{E}}^T\right] L^T(\boldsymbol{u})\boldsymbol{F}, \tag{5.7}$$

and time is related to the perturbed change in eccentric anomaly from Eq. 5.5 through the integral:

$$t = t_0 + \int_0^E \left(\frac{r(\phi)}{\sqrt{\mu\alpha(\phi)}}\right) d\phi. \tag{5.8}$$

Finally, for $\boldsymbol{F} \neq 0$, it can be shown that [54] α satisfies the variation-of-parameters differential equation

$$\frac{d\alpha}{dE} = -\frac{4}{\mu}\frac{d\boldsymbol{u}}{d\boldsymbol{E}}^T L^T(\boldsymbol{u})\boldsymbol{F}, \tag{5.9}$$

and also, the osculating energy in the classical Keplerian energy form of Eq. 5.5 can be transformed to the new form of the osculating energy constraint

$$\alpha = \frac{2}{r}\left[1 + \frac{4}{r}\frac{d\boldsymbol{u}}{dE}^T \frac{d\boldsymbol{u}}{dE}\right]^{-1}. \tag{5.10}$$

The first expression for α (Eq. 5.5) is the Cartesian form of the Keplerian energy equation, whereas the second expression (Eq. 5.10) is the KS transformed Keplerian energy equation. This equation holds in the presence of perturbations for $\alpha(t)$ as an *osculation constraint* in the variation of parameters. Thus α in Eq. 5.7 does not have to be solved by a differential equation (i.e. Eq. 5.9) as is frequently done. Rather it is a known function of the instantaneous KS state variables, given in Eq. 5.10. Notice E is the *perturbed change* in eccentric anomaly, not the eccentric anomaly itself.

Substitution of Eq. 5.10 into Eq. 5.7 gives the new and elegant form for the generally perturbed differential equation of motion in the KS variables:

$$\frac{d^2\boldsymbol{u}}{dE^2} + \frac{1}{4}\boldsymbol{u} = \frac{r^2}{4\mu}\left[1 + \frac{4}{r}\frac{d\boldsymbol{u}}{dE}^T \frac{d\boldsymbol{u}}{dE}\right]\left[I + \frac{4}{r}\frac{d\boldsymbol{u}}{dE}\frac{d\boldsymbol{u}}{dE}^T\right] L^T(\boldsymbol{u})\boldsymbol{F}. \tag{5.11}$$

This form Eq. 5.11 of the KS transformed differential equations is an original contribution from the research leading to this dissertation. Since the spherical harmonic series first non-zero term has a multiplicative factor of, $\frac{1}{r^2}$, and all higher order terms contain $\frac{1}{r^n}$, the multiplication by r^2 on the RHS of Eq. 5.11 simply reduces by 2 the denominator powers of r in all the spherical harmonic series representation of the gravitational perturbations. In particular, for the second zonal harmonic, r^2 in the denominator of this second zonal harmonic perturbation is canceled. The formulation therefore has a regularizing effect not only on the unperturbed problem but also on the dominant gravitational perturbation.

For an arbitrary force, the differential equations Eq. 5.5 - Eq. 5.11 must be solved numerically, however for small forces, they represent a weakly-coupled, weakly-nonlinear oscillator description of orbital motion and these equations are attractive from several points of view. From the work of Bai and Junkins [46, 17] and the classical Picard literature, we know that the convergence of the Picard method is a function of the "strength" of the dominant terms of the differential equation. Therefore, the $\frac{1}{4}$ coefficient of Eqs. 5.7 and Eqs. 5.11 suggests a basis for optimism that significant convergence advantages will be achieved in these transformed differential equations, compared to Eqs. 5.1, for reducing the number of Picard iterations and also increasing the maximum interval over which the Picard contraction mapping iterations will converge. We can anticipate these will be advantages for both the IVP and for the TPBVP. As will be evident below, these heuristic expectations are consistent with numerical reality and represent a significant computational advantage, especially for the Lambert problem.

In this section we have derived a *new* form for the *transformed* equation of motion (Eq. 5.11) in KS variables. By using the osculating constraint for α, we eliminate the need to solve the work/energy differential equation (Eq. 5.9) for α as is frequently done (with time variables that differ from E) in the classical references [56, 57]. Thus our KS differential equations are of order *nine*, not *ten* as is the case in the usual KS developments. In the following section we formulate the Keplerian Lambert problem in KS variables. This is somewhat analogous to Battin's classical Lambert solution, but in new variables. While formally analogous, our formulation here is not as general as Battin's. He considers all species of conics (elliptical, parabolic and hyperbolic), and for the elliptic case, he considers the multiple revolution transfers as well. In this dissertation we only consider the elliptical case. The relatively straightforward universal generalizations are left for future developments. The present developments apply to the majority of practical applications and the level of abstraction associated with the universal variable treatment is avoided for this "first rendition" treatment to increase qualitative insight. However, and importantly, we do generalize (in this dissertation) the Keplerian Lambert results to consider arbitrary smooth perturbations and demonstrate the formulation to efficiently solve perturbed TPBVPs using a $(40, 40)$ spherical harmonic gravity model.

5.2 Special Inertial Cartesian Coordinate System

Now let us consider the case of general perturbations. The TPBVP turns out to have some subtleties that arise due to the infinity of $\boldsymbol{u}$ vectors that correspond to given Cartesian coordinates. First, consider the three scalar equations implicit in 5.2:

$$\begin{aligned} X &= u_1^2 - u_2^2 - u_3^2 + u_4^2, \\ Y &= 2(u_1u_2 - u_3u_4), \\ Z &= 2(u_1u_3 + u_2u_4). \end{aligned} \tag{5.12}$$

Also squaring Eq. 5.2 allows us to establish

$$r = u_1^2 + u_2^2 + u_3^2 + u_4^2. \tag{5.13}$$

Adding the first of Eq. 5.12 and 5.13 yields the nice result

$$u_1^2 + u_4^2 = R^2 = \frac{r + X}{2}. \tag{5.14}$$

So it is clear that (u_1, u_4) lie on a circle of radius $R = \sqrt{(r + X)/2}$, and so all infinity of possible (u_1, u_4) pairs can be parameterized by the angle $\phi = \tan^{-1}(\frac{u_4}{u_1})$ or,

$$u_1 = R\cos\phi, \quad u_4 = R\sin\phi, \quad 0 \leq \phi < 2\pi. \tag{5.15}$$

Eliminating (u_1, u_4) as a function of ϕ, we can solve the second pair of equations 5.12 simultaneously for (u_2, u_3) as

$$u_2 = \frac{1}{2R}(Y\cos\phi + X\sin\phi), \quad u_3 = \frac{1}{2R}(-Y\sin\phi + Z\cos\phi). \tag{5.16}$$

Eqs 5.14- 5.16 define a four dimensional space curve or "fiber" that is a geodesic that lies on the surface of the four dimensional sphere of radius $\sqrt{r}$. Sweeping ϕ over the 2π range generates all infinity of points along the fiber of geometrically feasible u_i coordinates corresponding to the given (X, Y, Z). There is one additional subtlety that turns out to offer a way to simplify the most general case. As is well known, the same Cartesian form of the equations of motion $\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{F}$ holds for an infinity of inertial Cartesian coordinate system choices, therefore, we are free to choose the fixed inertial coordinate system orientation, so long as we are careful to project the force Cartesian components and boundary conditions appropriately into this chosen inertial frame. Notice at either initial or final time, that the boundary conditions and therefore, the above equations simplify significantly if one of the inertial Cartesian axes is aligned with either the initial or final position vector (Figures 5.1 and 5.2). It is also useful to rotate the inertial frame about the fixed axis so that the XY-plane lies in the unperturbed plane of motion. We can either use the initial position and velocity vector to define the inertial plane, or we can use (for TPBVPs) the prescribed initial and final position vectors to define the inertial plane. For both cases, and for small elapsed time, we can expect the out-of-plane motions to be small and in some circumstances, this truth is useful to guide heuristics in applications.

As one example, if the inertial system is chosen such that the initial or final position vector aligns with the X-axis, such that $X = r$, $Y = Z = 0$, at that point, then $R = \sqrt{r}$ and the above space curve assumes its simplest form as

$$u_1^2 + u_4^2 = r, \quad \{u_1 = \sqrt{r}\cos\phi, \quad u_4 = \sqrt{r}\sin\phi\} \quad \text{and} \quad \{u_2 = 0,\ u_3 = 0\}. \tag{5.17}$$

See Figure 5.3 for a geometric interpretation of the mapping from Cartesian coordinates into the infinite set of $\{u_1, u_2, u_3, u_4\}$ coordinates, for this specific example. In this case the locus of feasible vectors is a circle of radius $\sqrt{r}$ in the (u_1, u_4) plane. This is one of the few examples where one can fully visualize a space curve in a four-dimensional space!

We mention, Eq. 5.14 is a projection of the 4D space curve into the (u_1, u_4) plane. It is easy to verify, by subtraction of the first of Eqs 5.12 from 5.13 that the projected curve in the (u_2, u_3) plane is another circle:

$$u_2^2 + u_3^2 = R^2 = \frac{r - X}{2}, \tag{5.18}$$

which can be used to derive an alternate form for the space curve:

$$u_2 = R\cos\psi, \quad u_3 = R\sin\psi, \quad 0 \leq \psi < 2\pi, \tag{5.19}$$

$$u_1 = \frac{1}{2R}(Y\cos\psi + Z\sin\psi), \quad u_4 = \frac{1}{2R}(-Y\sin\psi + Z\cos\psi). \tag{5.20}$$

Notice, even though the infinite set of $\boldsymbol{u}$ vector boundary conditions exist for any given $\{X, Y, Z\}$, once $\{X, Y, Z\}$ and any particular consistent $\boldsymbol{u}$ are selected, the corresponding velocity forward and inverse mapping of $(\dot{X}, \dot{Y}, \dot{Z})$ to from $\boldsymbol{u}'$ is unique:

$$\left\{ \begin{array}{c} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ 0 \end{array} \right\} = \frac{2\sqrt{\mu\alpha}}{r} L(\boldsymbol{u})\boldsymbol{u}' \quad \Longleftrightarrow \quad \boldsymbol{u}' = \frac{r}{2\sqrt{\alpha\mu}} L^T(\boldsymbol{u}) \left\{ \begin{array}{c} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ 0 \end{array} \right\}. \tag{5.21}$$

The above developments can be used to resolve a key issue: Once an initial $\boldsymbol{u}$ vector is selected from among the infinity of possibilities, the corresponding velocity is computed from the second of Eqs 5.21 and then solution of the KS differential Eq. 5.8 leads to a corresponding unique trajectory $(\boldsymbol{u}(\tau), \boldsymbol{u}'(\tau))$. Even if the initial $(\boldsymbol{u}(\tau), \boldsymbol{u}'(\tau))$ state is such that Eq. 5.2 gives the correct final Cartesian $\{X_f, Y_f, Z_f\}$, we need to focus on the truth that the arrival $\boldsymbol{u}(\tau(E_f))$ will correspond to only one of the infinity of $\boldsymbol{u}$ vectors consistent with $\{X_f, Y_f, Z_f\}$. So, when we attempt to prescribe an initial and final $\boldsymbol{u}$ vector to solve a TPBVP in $\boldsymbol{u}$-space, if we hold the initial $\boldsymbol{u}$ fixed, we must only determine the single unknown variable ϕ to correctly prescribe the terminal boundary condition for $\boldsymbol{u}(\tau(E_f))$ that makes the arrival state of the unique MCPI Lambert solution of the KS differential equation 5.11, consistent with the boundary condition imposed at the initial time. Thus a one-dimensional search over the set of feasible $\boldsymbol{u}$ boundary conditions is required to find ϕ that generates the particular $\boldsymbol{u}(\tau(E_f))$. However, through a judicious choice of the inertial frame, and using a warm start from Keplerian motion, we have been able to construct a reasonably efficient solution process having $\phi = 0$ if perturbations vanish, and that evidently ensures the desired root for ϕ for the perturbed case is typically small.

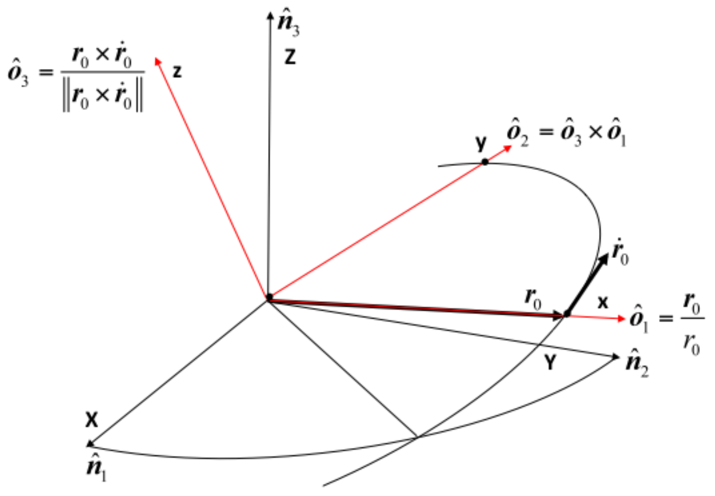
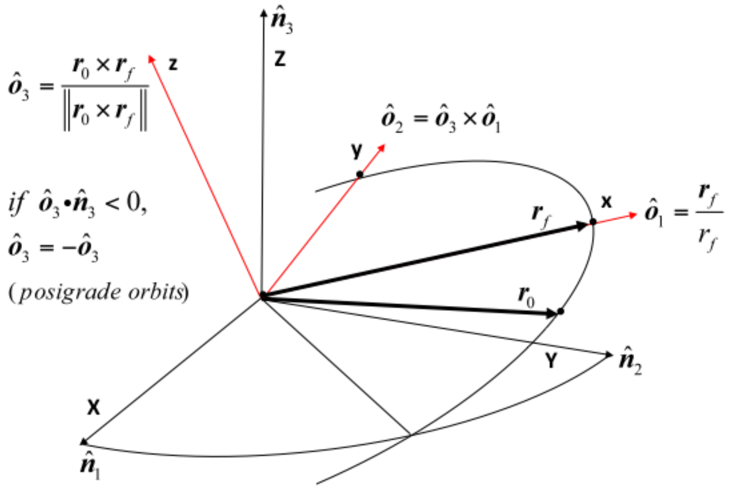

$\hat{o}_3 = \frac{r_0 \times \dot{r}_0}{\|r_0 \times \dot{r}_0\|}$
z
$\hat{n}_3$
Z
y
$\hat{o}_2 = \hat{o}_3 \times \hat{o}_1$
$\dot{r}_0$
r_0
x
$\hat{o}_1 = \frac{r_0}{r_0}$
Y
$\hat{n}_2$
X
$\hat{n}_1$

Figure 5.1: Inclined inertial plane defined by initial position vector and the initial velocity vector [19].

This small ϕ enables algorithms to quickly find the ϕ such that the initial and final boundary conditions in $\boldsymbol{u}$-space are the desired terminal states along the unique solution of the KS differential equations and correspond to the specified Cartesian terminal coordinates.

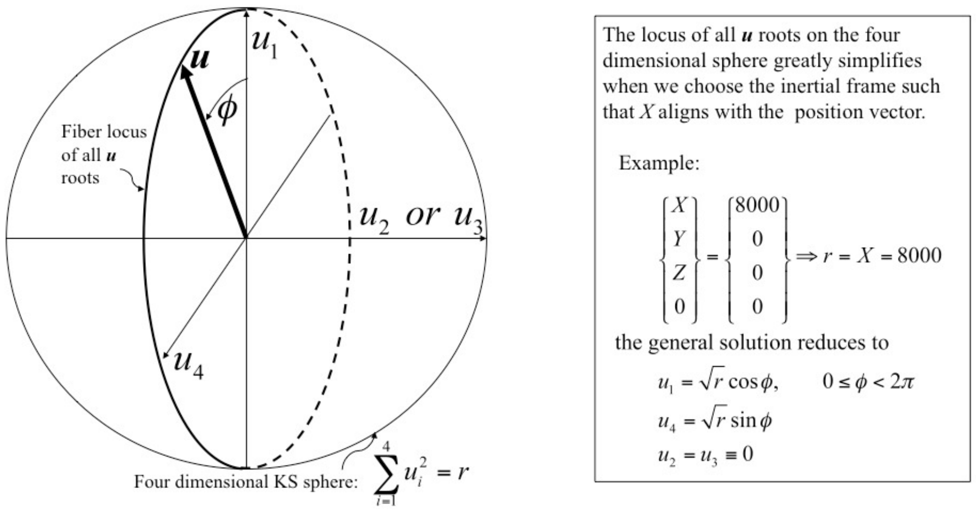
5.3 Analytical Lambert Solver

It is useful to first consider solving the planar KS Keplerian special case (which has an analytical solution), before discussing the general three dimensional Lambert problem in KS variables. The upper left 2×2 sub-matrix of $L(\boldsymbol{u})$ is the needed subset of the position transformation and the resulting equations simplify to the classical Levi-Civita transformation [1, 60] discovered in 1920, some forty years prior



$\hat{n}_3$
Z
$\hat{o}_3 = \frac{r_0 \times r_f}{\|r_0 \times r_f\|}$
z
$\hat{o}_2 = \hat{o}_3 \times \hat{o}_1$
y
$if\ \hat{o}_3 \bullet \hat{n}_3 < 0,$
$\hat{o}_3 = -\hat{o}_3$
$(posigrade\ orbits)$
r_f
x
$\hat{o}_1 = \frac{r_f}{r_f}$
r_0
Y
$\hat{n}_2$
X
$\hat{n}_1$

Figure 5.2: Inclined inertial plane defined by the initial position vector and the final position vector [19].



u_1
u
ϕ
Fiber locus of all u roots
$u_2\ or\ u_3$
u_4
Four dimensional KS sphere: $\sum_{i=1}^{4} u_i^2 = r$

The locus of all u roots on the four dimensional sphere greatly simplifies when we choose the inertial frame such that X aligns with the position vector.

Example:

$$\begin{Bmatrix} X \\ Y \\ Z \\ 0 \end{Bmatrix} = \begin{Bmatrix} 8000 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \Rightarrow r = X = 8000$$

the general solution reduces to

$$u_1 = \sqrt{r}\cos\phi, \qquad 0 \le \phi < 2\pi$$
$$u_4 = \sqrt{r}\sin\phi$$
$$u_2 = u_3 \equiv 0$$

Figure 5.3: The feasible $\boldsymbol{u}$ - locus is simply a circle of radius $\boldsymbol{u}$ in the $(\boldsymbol{u}_1,\boldsymbol{u}_4)$ plane [19].

to the more general KS result [58]. Since the planar KS transformation reduces to a two-to-two mapping $(X, Y) \Longleftrightarrow (u_1, u_2)$, the root structure is greatly simplified and the fiber of solutions in 4D reduce in 2D to fixed points (roots) as discussed below.

Restricting the motion to the plane ($Z(t) = 0$), the general KS transformation simplifies as follows:

$$\begin{Bmatrix} X \\ Y \end{Bmatrix} = L(\boldsymbol{u})\boldsymbol{u}, \quad \text{where} \quad L(\boldsymbol{u}) = \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}. \tag{5.22}$$

The inverse mapping is

$$\text{for } X \geq 0: \;\; \boldsymbol{u} = \pm \begin{Bmatrix} \left(\frac{(r+X)}{2}\right)^{\frac{1}{2}} \\ \frac{Y}{(2(r+X))^{\frac{1}{2}}} \end{Bmatrix} \text{ or for } X < 0: \;\; \boldsymbol{u} = \pm \begin{Bmatrix} \frac{Y}{(2(r-X))^{\frac{1}{2}}} \\ \left(\frac{(r-X)}{2}\right)^{\frac{1}{2}} \end{Bmatrix}. \tag{5.23}$$

As is evident above in Eqs. 5.2 and the planar special case of Eq. 5.22, the mapping from $\boldsymbol{u}$-space to Cartesian space is unique. As expected, the inverse mapping is not unique, but it is greatly simplified to a 2-to-2 mapping for the planar case. In fact for the planar case ($Z(t) = 0$), for (X, Y) specified, only two real points in $\boldsymbol{u}$-space exist, given in Eq. 5.23. For IVPs, as long as we avoid the potential division by zero at $x = \pm r$ (by following the sign of rules evident in Eq 5.23), the solution of these equations is very well-behaved. For IVPs we only need to use these equations once at initial time, and the inverse mapping in Eq. 5.22 (or more generally Eq. 5.2) is non-singular and unique everywhere. For TPBVP, however, we have to resolve the sign ambiguities carefully, otherwise we may accidentally request the algorithm to look for a one-and-a-fraction orbit transfer instead of a fractional orbit transfer. Note that two revolutions occur in Cartesian space for each revolution in $\boldsymbol{u}$-space, quite analogous to quaternion representation of rotational motion.

For a given set of boundary conditions, there are two possible solutions that depend on the sign of the final position in $\boldsymbol{u}$-space. If the angle between the initial and final boundary conditions is less than π, and the number of revolutions is odd (a fraction of the first orbit is considered an odd revolution and a fraction of the second orbit is even etc.), then there is no sign change on the final position. However, in the odd orbits, when there is more than an angle of π between the initial and final position then there is a sign change on the final boundary condition. The opposite sign convention occurs for an even orbit. For retrograde orbits, the above sign convention holds, but it is all reversed.

Using the planar KS transformation (Levi-Civita transformation) it is evident that solving the four uncoupled harmonic oscillators of Eq. 5.7 or Eq. 5.11 (where $\boldsymbol{F} = 0$) has an analytical solution simply given by

$$\boldsymbol{u} = \boldsymbol{u}_0 \cos\frac{E}{2} + 2 \left.\frac{d\boldsymbol{u}}{dE}\right|_0 \sin\frac{E}{2}, \quad \frac{d\boldsymbol{u}}{dE} = -\frac{1}{2}\boldsymbol{u}_0 \sin\frac{E}{2} + \left.\frac{d\boldsymbol{u}}{dE}\right|_0 \cos\frac{E}{2}. \tag{5.24}$$

Or, in state transition matrix form:

$$\left\{ \begin{array}{c} \boldsymbol{u} \\ \frac{d\boldsymbol{u}}{dE} \end{array} \right\} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \left\{ \begin{array}{c} \boldsymbol{u}_0 \\ \left.\frac{d\boldsymbol{u}}{dE}\right|_0 \end{array} \right\}, \tag{5.25}$$

where the sub-matrices are $\Phi_{11} = \cos\frac{E}{2}I$, $\Phi_{12} = 2\sin\frac{E}{2}I$, $\Phi_{21} = -\frac{1}{2}\sin\frac{E}{2}I$, $\Phi_{22} = \cos\frac{E}{2}I$.

The integral of Eq. 5.8 can be carried out analytically for the $\boldsymbol{F} = 0$ case to obtain a form of Kepler's equation

$$\alpha^{\frac{3}{2}}\sqrt{\mu}\,(t - t_0) = E - (1 - \alpha r_0)\sin E - \alpha^{\frac{1}{2}}\sigma_0\,(\cos E - 1)\,, \quad \sqrt{\mu}\sigma_0 \equiv \boldsymbol{r}_0 \cdot \dot{\boldsymbol{r}}_0. \tag{5.26}$$

E denotes the *change* in the classical eccentric anomaly from initial conditions to the current state, i.e. herein E is not referenced to perigee, but rather to initial position. There is a single unique orbit for the fractional orbit transfer case, except for the co-linear position case in which the orbit plane is not unique). The singularity structure for the Keplerian special case has been found to carry over to the gravitationally perturbed generalization of this two-point boundary value problem. The preservation of singularity structure in the presence of perturbations is a consequence of the geometric truth that the osculating orbit plane is undetermined when the initial and final position vectors are co-linear.

From Eq. 5.22, we can eliminate initial velocity in $\boldsymbol{u}$-space as a function of the final boundary conditions

$$\left.\frac{d\boldsymbol{u}}{dE}\right|_0 = \frac{1}{2\sin\frac{E_f}{2}}\left(\boldsymbol{u}_f - \boldsymbol{u}_0\cos\frac{E_f}{2}\right). \tag{5.27}$$

We now outline the completion of the solution of the Keplerian Lambert's problem in KS variables. Using the energy equation

$$\alpha = \frac{2}{r_0} - \left.\frac{d\boldsymbol{r}^T}{dt}\right|_{t_0}\left.\frac{d\boldsymbol{r}}{dt}\right|_{t_0} = \frac{2}{r_0}\left[1 + \frac{4}{r_0}\left.\frac{d\boldsymbol{r}^T}{dE}\right|_{t_0}\left.\frac{d\boldsymbol{r}}{dE}\right|_{t_0}\right]^{-1}, \tag{5.28}$$

and also using Eq. 5.28, we can eliminate α and $\sigma_0 = \frac{r_0 \cdot \dot{r}_0}{\sqrt{\mu}}$ in Eq. 5.26 as a function of $(\boldsymbol{u}_0, \boldsymbol{u}_f, E_f)$, leaving only E_f as an unknown. Then the modified Eq. 5.26 with all unknowns on the RHS eliminated except E_f can be iterated via a Newton/secant method to converge on the correct final eccentric anomaly (E_f) for a given final time (t_f). An initial guess for the final eccentric anomaly is computed from true anomaly, using the dot product of the initial and final position vectors.

This method is well-behaved and convergence is reliable. For our study in this chapter we consider only fractional orbit cases. Solving Lambert's problem analytically not only provides the final eccentric anomaly (corresponding to the final time), but it also provides a *warm start* solution approximation for solving the perturbed problem.

It may seem that the *warm start* developed above would only work for planar orbits, however, with a bit of thought, we can use this for any inclined orbit in 3D Cartesian space or 4D $\boldsymbol{u}$-space. We construct a plane defined by the initial and final positions and use the corresponding constant direction cosine matrix to project the initial Cartesian coordinates from the equatorial frame into this special inertial frame. The warm start is computed in this special orbit frame where $z = 0$ and where there are no u_3 or u_4 components.

Our KS Lambert solver is somewhat analogous to Battin's classical Lambert solution [1], but in new variables. Although, for geometrical clarity and to address the most common applications, we have developed these results for the elliptic (positive α case). It is clear that by following the pattern of reference [58] and especially reference [56], a universal analogy of the above developments can be developed that is applicable to all species of elliptical, parabolic and hyperbolic motion. We leave this generalization for future work.

The present KS special case analytical solution of the Keplerian Lambert problem can virtually certainly be improved upon. It is computationally competitive with p-iteration which is not generally preferred over Battin's algorithm in terms of the iteration efficiency and universal generality. However, our ultimate goal here is the development of generalized Lambert methods that accommodate rather general perturbations for which the classical Lambert algorithms do not apply. The present KS Lambert Keplerian solution should be viewed as a demonstration and not as the final word. It sets the stage for the perturbed Lambert algorithms below in KS space.

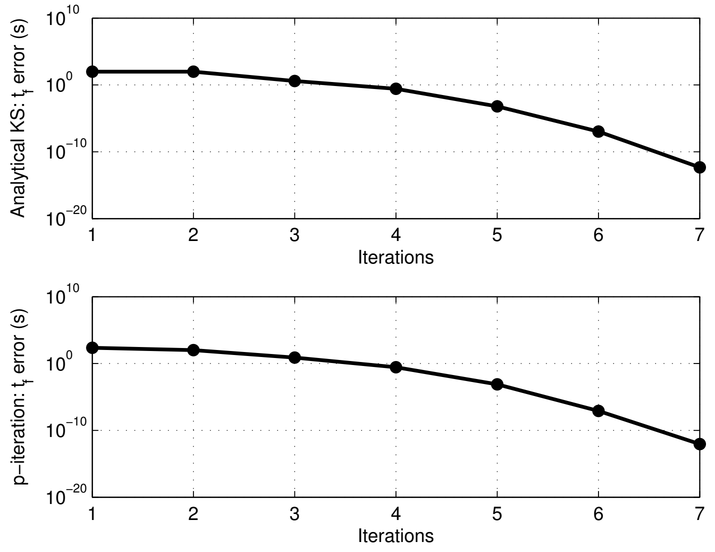


Figure 5.4: Keplerian Lambert solver comparison to p-iteration [19].

5.4 Theoretical Convergence

The domain over which the MCPI-IVP will converge is finite, and this gives rise to easily solvable challenges when the desired time interval over which a solution is sought is greater than the domain of convergence. For the TPBVP, Bai and Junkins [18] also studied MCPI convergence, and found that the interval was typically about one order of magnitude smaller for the TPBVP compared to the corresponding IVP. Bai and Junkins [17] did an MCPI convergence analysis using a simple linear oscillator to provide some insight on the matter. We reiterate the key points of their study here, for reader convenience, and extend the developments to investigate the convergence domain (time of flight interval between the specified terminal positions) specifically for the KS transformed equations of motion.

Consider a TPBVP described by the following equation, where c is a scalar:

$$\frac{d^2\boldsymbol{x}(t)}{dt^2} = -c\boldsymbol{x}(t), \quad t\epsilon[t_0, t_f], \quad \boldsymbol{x}(t_0), \quad \boldsymbol{x}(t_f) = \boldsymbol{x}_f. \tag{5.29}$$

This can be solved iteratively using the equation below, which is obtained from Eqs 3.61 and 3.62 in Chapter 3, where $\boldsymbol{g}(\boldsymbol{X}^{i-1}, \boldsymbol{V}^{i-1})$ has already been transformed from the time domain into the τ domain.

$$\boldsymbol{X^i} = C_x C_{B2} C_{I1} C_f \boldsymbol{g}(\boldsymbol{X}^{i-1}, \boldsymbol{V}^{i-1}) + C_x \boldsymbol{X}_{0f}. \tag{5.30}$$

More specifically, for Eq. 5.29

$$\boldsymbol{X^i} = -cw_2^2 C_x C_{B2} C_{I1} C_f \boldsymbol{X}^{i-1}(t) + C_x \boldsymbol{X}_{0f}, \quad w_2 = \frac{dt}{d\tau} = \frac{t_f - t_0}{2}. \tag{5.31}$$

It is known from linear systems theory that this sequence is convergent to a fixed point only if all the eigenvalues of the matrix $[cw_2^2 C_x C_{B2} C_{I1} C_f]$ are contained within the unit circle. That is, Eq. 5.31 must be a contraction mapping for convergence to a fixed point. Furthermore, the largest norm eigenvalue dictates the rate of Picard iteration convergence. The eigenvalues of $[C_x C_{B2} C_{I1} C_f]$ for various N are shown in Figure 5.5. In contrast [17] to the complex eigenvalues for the IVP that are attracted to fixed points on a small circle near the origin, inside the unit sphere, the real eigenvalues of $[C_x C_{B2} C_{I1} C_f]$ for the case of a TPBVP *are attracted to fixed points on a straight line inside the unit circle.* The maximum magnitude of eigenvalues of $\lambda_{max}\left([C_x C_{B2} C_{I1} C_f]\right)$ is almost invariant for large N approaches a constant $\lambda_{max}\left([C_x C_{B2} C_{I1} C_f]\right) \approx 0.4053$ with respect to increasing the Chebyshev order.

The necessary condition for convergence is given by

$$cw_2^2\lambda_{max}\left([C_x C_{B2} C_{I1} C_f]\right) \leq 1. \tag{5.32}$$

Rearranging this with the unknown quantities on the left and known quantities on the right gives the following.

$$c(t_f - t_0)^2 \leq \frac{4}{\lambda_{max}\left([C_x C_{B2} C_{I1} C_f]\right)} \leq \frac{4}{0.4053} \approx 9.87. \tag{5.33}$$

It is clear that the time interval and the linear coefficient c will dictate the domain over which the method will converge. Satisfying this condition guarantees that Picard iteration will converge, for a fixed N, but it does not guarantee that N is sufficiently large to accurately approximate the solution. That is, while the solution will converge to some specific approximation, more nodes may be required to accurately capture the system dynamics with desired precision. While still theoretically convergent, as the LHS of Eqs 5.32 and 5.33 approach the RHS, the eigenvalues approach the stability boundary of the unit circle and very near the unit circle, the convergence may be too slow to be considered practically convergent.

Performing the same analysis for the KS transformed equations of motion results in the following necessary condition for convergence.

$$\frac{1}{4}(E_f - E_0)^2 \leq \frac{4}{0.4053} \approx 9.87. \tag{5.34}$$

Note the c in Eq. 5.33 is replaced by $\frac{1}{4}$ in Eq. 5.34. This dictates $(E_f - E_0) \leq 6.28 \approx 2\pi$, so this (remarkably!) shows that the theoretical upper bound is one orbit period; this bound can only be approached because convergence rate will approach zero as $(E_f - E_0) \to 6.28$. Since Lambert's theorem says that there is a unique solution only

Table 5.1: Test case orbits [19].

Orbit type	Semimajor axis a (km)	Eccentricity e
LEO	8,000	0.125
MEO	10,963	0.4
GTO	26,352	0.6
HEO	26,554	0.72

for up to one orbit case, it is immediately evident that no other coordinate choice can possibly lead to a Picard iteration with a large convergence interval! To investigate how this $(E_f - E_0) \leq 2\pi$ theoretical bound maps into enhanced convergence, we computed the theoretical domain of convergence for the four test case orbits given in Table 5.1, for Cartesian and KS equations of motion.

Although this convergence analysis is formulated for linear systems, we use it approximately on the nonlinear Cartesian equations of motion (Eq. 5.35) simply to provide some insight on what we can expect to see with regard to numerical convergence. The Keplerian orbit equations in Cartesian and KS variables are given below.

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r}, \tag{5.35}$$

$$\boldsymbol{u}'' = -\frac{1}{4}\boldsymbol{u}. \tag{5.36}$$

As an example, we compute the approximate *theoretical* domain of convergence (in # orbits) at perigee for the LEO orbit given in Table 5.1. The local linear approximation of the equation means that the local c will vary around any non-circular the orbit. The smallest value for the domain of convergence would coincide with perigee, and thus should provide the most conservative guess.

At perigee, $t_f - t_0 \leq \sqrt{\frac{4}{c\lambda_{max}\left(\left[C_x C_{B2} C_{I1} C_f\right]\right)}}$, where for a low eccentricity orbit with $r_0 = 7000$ km, we adopt the approximation, $c = \frac{\mu}{7000^3} = 1.1621 \times 10^{-6}$. This results in a maximum time-of-flight of $t_f - t_0 \leq 34745\ s$. The approximate domain of convergence can then be computed as $(t_f - t_0)/P \leq 0.4$ orbits, where $P = 2\pi\sqrt{\frac{a^3}{\mu}}$. For an elliptical orbit, we find basing this approximation on $r_0 = \textit{perigee radius}$, while using $a = \textit{actual semimajor axis}$ proves conservative. If we do the same computation for the KS problem we get $E_f - E_0 \leq \sqrt{\frac{4}{\lambda_{max}\left(\left[C_x C_{B2} C_{I1} C_f\right]\right)}}$, where $c = \frac{1}{4}$. Therefore $E_f - E_0 \leq 6.28\ \textit{radians} = 2\pi$, and the *theoretical* domain of convergence when integrating KS variables is 1 orbit.

The above computation is done for the other test case orbits, and all the theoretical results are displayed in Figure 5.6. Note that the theoretical convergence upper bound on transfer time, when solving the Keplerian Lambert problem by MCPI in Cartesian variables decreases with increasing eccentricity. This is not the case in KS variables, and in fact all orbits theoretically have the same MCPI domain of convergence (about one orbit), regardless of eccentricity. Obviously invariance with respect to the eccentricity variations is a fundamental advantage.

Of course, this discussion is qualitative, since equations of motion in Cartesian coordinates are nonlinear and rigorous local linearization will cause the effective c to vary around the orbit, and the theoretical value computed above does not rigorously represent the attainable convergence. However, we have done numerical studies that show that these theoretical bounds are usually optimistic and agree within about $10 - 15\%$ with the actual upper bound time interval over which MCPI can achieve practical convergence. In KS coordinates, the motion is linear *without approximation* and $c = \frac{1}{4}$ is rigorously constant. Based on this theoretical analysis it is clear that the KS transformation is extremely powerful when the resulting regularized differential equations are solved by MCPI and give the maximum theoretical convergence limit of one orbit. Clearly, the elimination of the eccentricity dependence of the domain of MCPI convergence is "better than useful!".

Having considered the MCPI-TPBVP, we conduct the same analysis for the MCPI-IVP. The necessary condition for convergence is shown below, with the maximum eigenvalues of the $[C_x C_{I2} C_{I1} C_f]$ displayed in Figure 5.7.

$$c(t_f - t_0)^2 \leq \frac{4}{\lambda_{max}\left([C_x C_{I2} C_{I1} C_f]\right)} \leq \frac{4}{0.003} \approx 1300. \tag{5.37}$$

The theoretical domain of convergence for the IVP is presented in Figure 5.8. It is interesting to note that the theoretical domain of convergence for the MCPI-IVP is much greater than that for the MCPI-TPBVP, and in KS variables the domain of convergence is increased for both the IVP and TPBVP compared with that achievable in Cartesian variables. Once again, we see that in KS coordinates, the theoretical domain of convergence does not degrade with increasing eccentricity. In the following sections we compute the domain of convergence numerically and anticipate that these insightful theoretical results will be an optimistic estimate of the truth.

5.5 Keplerian Lambert Problem Solver by MCPI: Cartesian vs KS

To study the numerical domain of convergence for Lambert's problem, we use the MCPI-TPBVP implementation to integrate the Keplerian equations of motion in both Cartesian and KS variables. The four test cases given in Table 5.1, with varying semimajor axis and eccentricity span the region of interest. We anticipate that the Keplerian MCPI results will provide an indication of the ideal range of convergence associated with perturbations, and these results will also show how much the introduction of perturbations "costs" in terms of reduction of the convergence domain. In all cases the Hamiltonian is preserved to machine precision.

Figure 5.9 shows the results for these four orbits. In each test case the thin solid line and thick solid line represent the domain of convergence for the problem solved using KS and Cartesian variables respectively. As anticipated, the practical conver-

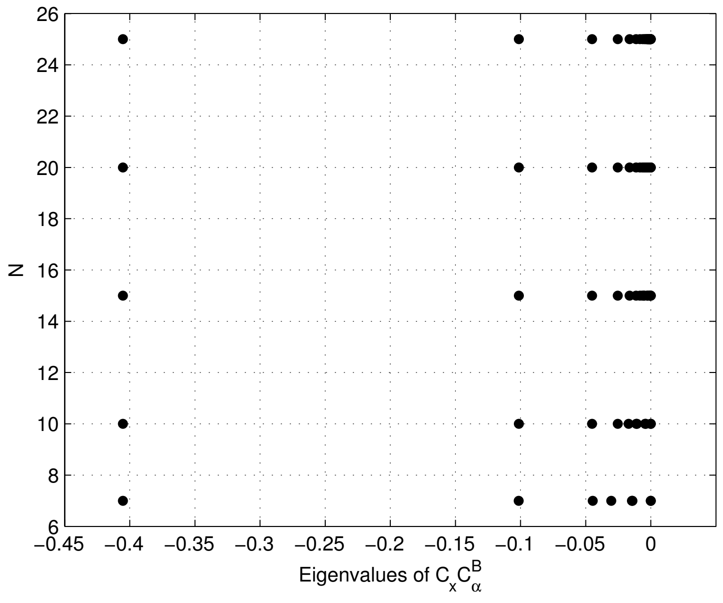


Figure 5.5: TPBVP: eigenvalues of $[C_xC_{B2}C_{I1}C_f]$ [19].

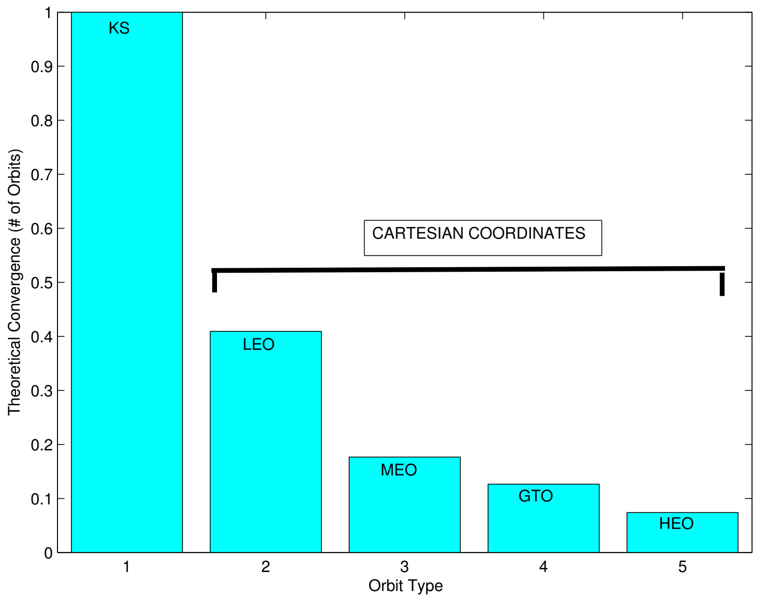


Figure 5.6: TPBVP theoretical convergence [19].

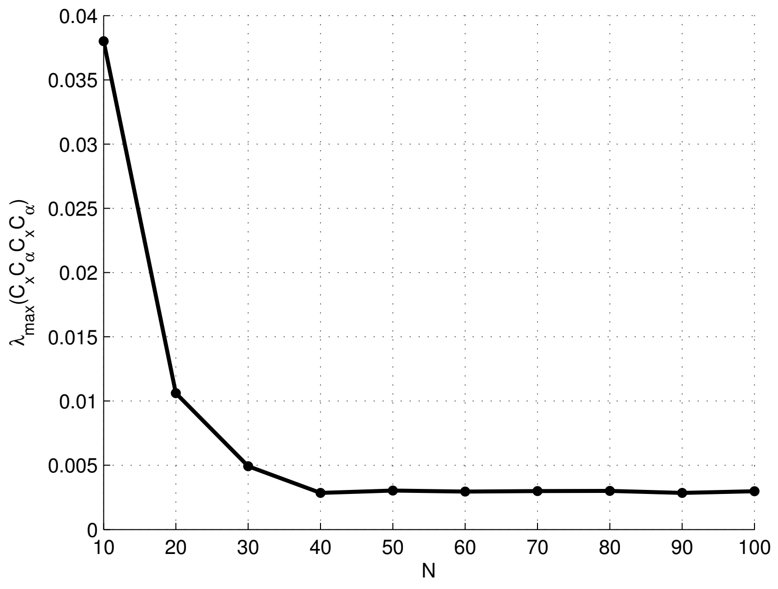


Figure 5.7: IVP: eigenvalues of $[C_x C_{I2} C_{I1} C_f]$ [19].

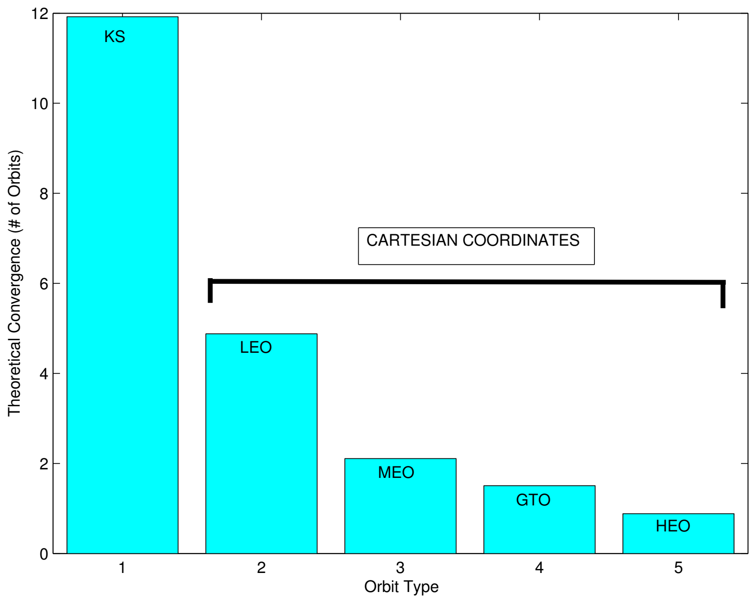


Figure 5.8: IVP theoretical convergence [19].

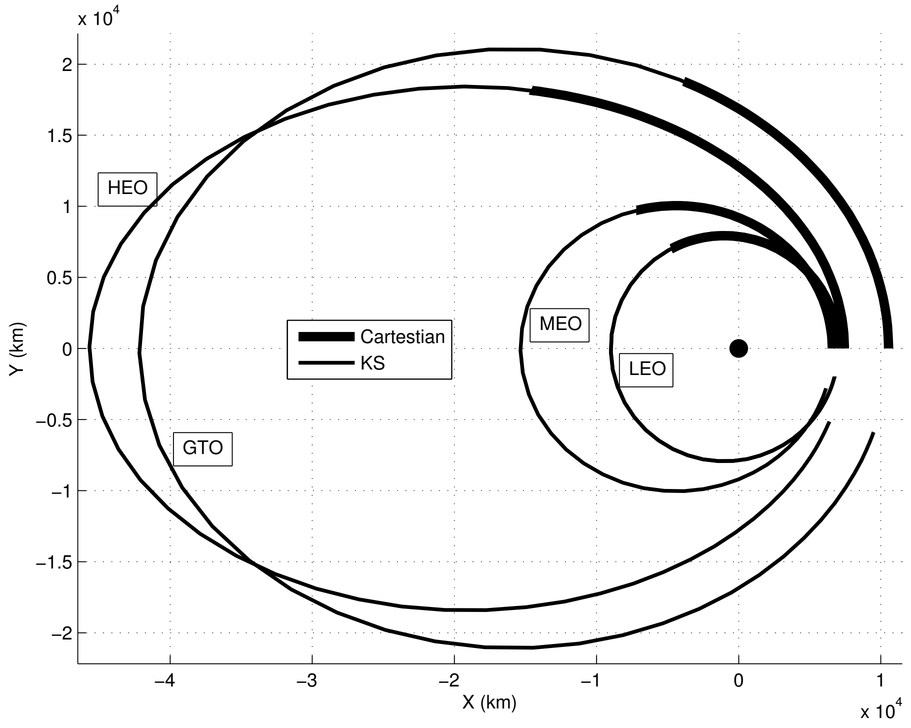


Figure 5.9: MCPI domain of convergence comparison: Cartesian vs KS [19].

gence is a bit less than the theoretical one-orbit bound. It is clear that implementing the KS transformation has enabled the domain of convergence to be vastly improved relative to the usual Cartesian coordinate formulation.

These results simply demonstrate the superiority, with regard to convergence, of the MCPI-TPBVP implementation in KS variables compared with that in Cartesian variables. Since we already have analytic methods (except for iteration of a single transcendental equation) for solving the Keplerian Lambert problem, for example Battin's method, a-iteration [16], p-iteration [61], and our KS Lambert solver presented in the previous section, there is no evident advantage to solve the Keplerian problem using MCPI. However, when considering the perturbed problem, which is discussed below, integrating the perturbed KS equations of motion becomes an extremely useful and efficient compared with other methods. A final observation: using segmentation to establish piecewise continuous approximation, the MCPI algorithm for the IVP can propagate orbits over arbitrarily long time intervals. However, for the BVP, if we wish to avoid shooting methods, the maximum interval over which MCPI converges is a very important issue. It is evident that, for up to 95% of an orbit, all fractional orbit Keplerian Lambert transfers can be solved by MCPI (no linearization based shooting method is required), and in the developments below, we show this also holds true for the gravitationally perturbed Lambert problem with a modest reduction in the maximum time interval for convergence. Even with the KS formulation, for the multi-revolution orbit transfers, we must resort to nonlinear shooting algorithms.

5.6 Perturbed Lambert Problem: KS vs Cartesian

We now extend the formulation to consider perturbations. As mention in a previous section, the MCPI-TPBVP is not a *Newton-like* shooting method and is advantageous from the point of efficiency. However, solving a TPBVP requires specifying both the initial and final positions. These are usually known in Cartesian space and can be converted to KS space. However, as mentioned earlier, it is readily verified that the final boundary condition in the perturbed $\boldsymbol{u}$-space is not unique. This presents challenges that we discuss below, as well as our method and algorithm for solving this important problem.

At every instantaneous point along a Cartesian trajectory $(X(t), Y(t), Z(t))$ there is an instantaneous sphere of radius $\sqrt{\boldsymbol{r}} = \boldsymbol{u}^T\boldsymbol{u}$ in $\boldsymbol{u}$-space that contains the geodesic of Eq 5.17. This instantaneous geodesic contains all the feasible $\boldsymbol{u}$ points consistent with the projection $\boldsymbol{r} = L(\boldsymbol{u})\boldsymbol{u}$. A particular unique trajectory for $(\boldsymbol{u}(E), \boldsymbol{u}'(E))$

ensues from solving the KS differential equations 5.11 with the initial conditions

$$\boldsymbol{u}(0) = \frac{1}{r_0} L^T(\boldsymbol{u}(0))\boldsymbol{r}_0 \text{ and } \boldsymbol{u}'(0) = \frac{r_0}{2\sqrt{\mu\alpha_0}} L^T(\boldsymbol{u})\dot{\boldsymbol{r}}_0. \tag{5.38}$$

Note, $\boldsymbol{u}'(0)$ cannot be chosen independently of $\boldsymbol{u}(0)$, because admissible $\boldsymbol{u}'(0)$ must satisfy $L_4(\boldsymbol{u}(0))\boldsymbol{u}'(0) = 0$. Associated with $\boldsymbol{u}(E)$ there is a specific $\phi(E)$ consistent with the instantaneous $\boldsymbol{u}_1(E)$, $\boldsymbol{u}_4 E$, viz $\tan(\phi(\mathrm{E})) = \frac{\boldsymbol{u}_4(\mathrm{E})}{\boldsymbol{u}_1(\mathrm{E})}$.

Thus the time varying sphere of radius $\sqrt{\boldsymbol{r}(E)}$ of feasible $\boldsymbol{u}$ vectors sweeps out a 4D tube in $\boldsymbol{u}$-space that initiates with a radius $\sqrt{r_0}$ and terminates with a final radius $\sqrt{r_f}$. Also, for any specific inertial frame orientation choice, a geodesic curve gives the feasible $\boldsymbol{u}(E)$ corresponding to $(X(t), Y(t), Z(t))$, located at a point on the instantaneous sphere of radius $\sqrt{r}$ completely defined by the angle $\phi(0) = \tan^{-1}(\frac{\boldsymbol{u}_4(0)}{\boldsymbol{u}_1(0)})$, an angle in the $\boldsymbol{u}_1(0)$, $\boldsymbol{u}_4(0)$ plane. Traditionally the specific choice of $\phi_0 = 0$ is made. Note for general 3D perturbed motion, we cannot constrain $\phi(E) = 0$ along the $\boldsymbol{u}$ trajectory, because $\phi(E) = \tan^{-1}(\frac{\boldsymbol{u}_4(\mathrm{E})}{\boldsymbol{u}_1(\mathrm{E})})$ must always hold true. All of this is leading up to the key points: The desired solution $(\boldsymbol{u}(E),\boldsymbol{u}'(E))$ for a TPBVP has a specific $\phi(E)$ history which initiates with some arbitrary $\phi(0)$ and arrives with some specific $\phi(E_f) = \phi_f$. We will not know ϕ_f a priori. A key issue is finding ϕ_f so $\boldsymbol{u}(E_f)$ can be computed such that it lies on the solution of the KS differential equations that initiates at $\phi_0 = 0$.

Our method to solve this problem is a hybrid use of the the MCPI-KS-TPBVP algorithm and the MCPI-KS-IVP. We solve the TPBVP in $\boldsymbol{u}$-space using the boundary conditions computed by transforming the Cartesian boundary conditions into $\boldsymbol{u}$-space.

$$\{\boldsymbol{u}_{_{TPBVP}}(E), \boldsymbol{u}'_{_{TPBVP}}(E), \phi_f\} = KS_{_{TPBVP}}(\boldsymbol{u}_0, \boldsymbol{u}(E_f)), \tag{5.39}$$

where $\boldsymbol{u}(E_f)$ lies on the geodesic admissible fiber of final boundary conditions given by Eqs 5.14 to 5.16, a point on the circle of radius $\sqrt{(r_f)}$ in Figure 5.3:

$$\boldsymbol{u}(E_f) = g(X_f, 0, 0, \phi_f), \tag{5.40}$$

where g represents the u_1, u_2, u_3 and u_4 given in Figure 5.3. That is, it is the space curve for the special coordinate system, generated by sweeping ϕ_f.

Since we have a weakly perturbed problem, we anticipate that the warm start will be near the final solution; this produces an initial $\boldsymbol{u}$-space velocity estimate ($\boldsymbol{u}'$) that is close to the ultimate desired final velocity but it slightly violates the bilinear constraint. Although the trajectory computed by the MCPI-KS-TPBVP meets the final boundary conditions, we find when converted back to Cartesian space the vector has a small 4^{th} component, which is not physically possible and is of course due to the bilinear constraint not being exactly satisfied. In order to trim this 4^{th} component we convert the $\boldsymbol{u}'$ into Cartesian velocity and just set the small 4^{th} component equal to zero. We then convert this physically admissible Cartesian velocity back to the $\boldsymbol{u}$-space and input this now physically feasible KS velocity ($\boldsymbol{u}'$) and the initial position boundary condition in $\boldsymbol{u}$-space into the MCPI-KS-IVP as shown below.

$$\{\boldsymbol{u}_{_{IVP}}(E), \boldsymbol{u}'_{_{IVP}}(E), \phi(E_f)\} = KS_{_{IVP}}(\boldsymbol{u}_0, \boldsymbol{u}'_0), \tag{5.41}$$

where

$$\boldsymbol{u}'_{_{TPBVP}}(0) = \frac{\boldsymbol{r}_0}{2\sqrt{\mu\alpha_0}} L(\boldsymbol{u}(0))\dot{\boldsymbol{r}}(0), \tag{5.42}$$

and

$$\phi_f = \frac{\boldsymbol{u}_{4_{IVP}}(E_f)}{\boldsymbol{u}_{1_{IVP}}(E_f)}. \tag{5.43}$$

We evaluate the converged $\boldsymbol{u}$ trajectory for the desired time interval and find that the final boundary condition in $\boldsymbol{u}$-space is slightly perturbed from that which we originally prescribed on the plane. Figure 5.10 illustrates the geodesic curves and the initial and final boundary conditions in perturbed $\boldsymbol{u}$-space. We use this new final boundary condition to compute ϕ_f, which we use to compute an updated feasible final position in$\boldsymbol{u}$-space. The MCPI-KS-TPBVP is run one more time with this new final boundary condition. Remarkably, the resulting solution has been found to reliably meet the initial and final boundary conditions in Cartesian space and also preserve the Hamiltonian to machine precision. While we have no theoretical guarantee of this convergence behavior, it has been consistent over a family of numerical tests.

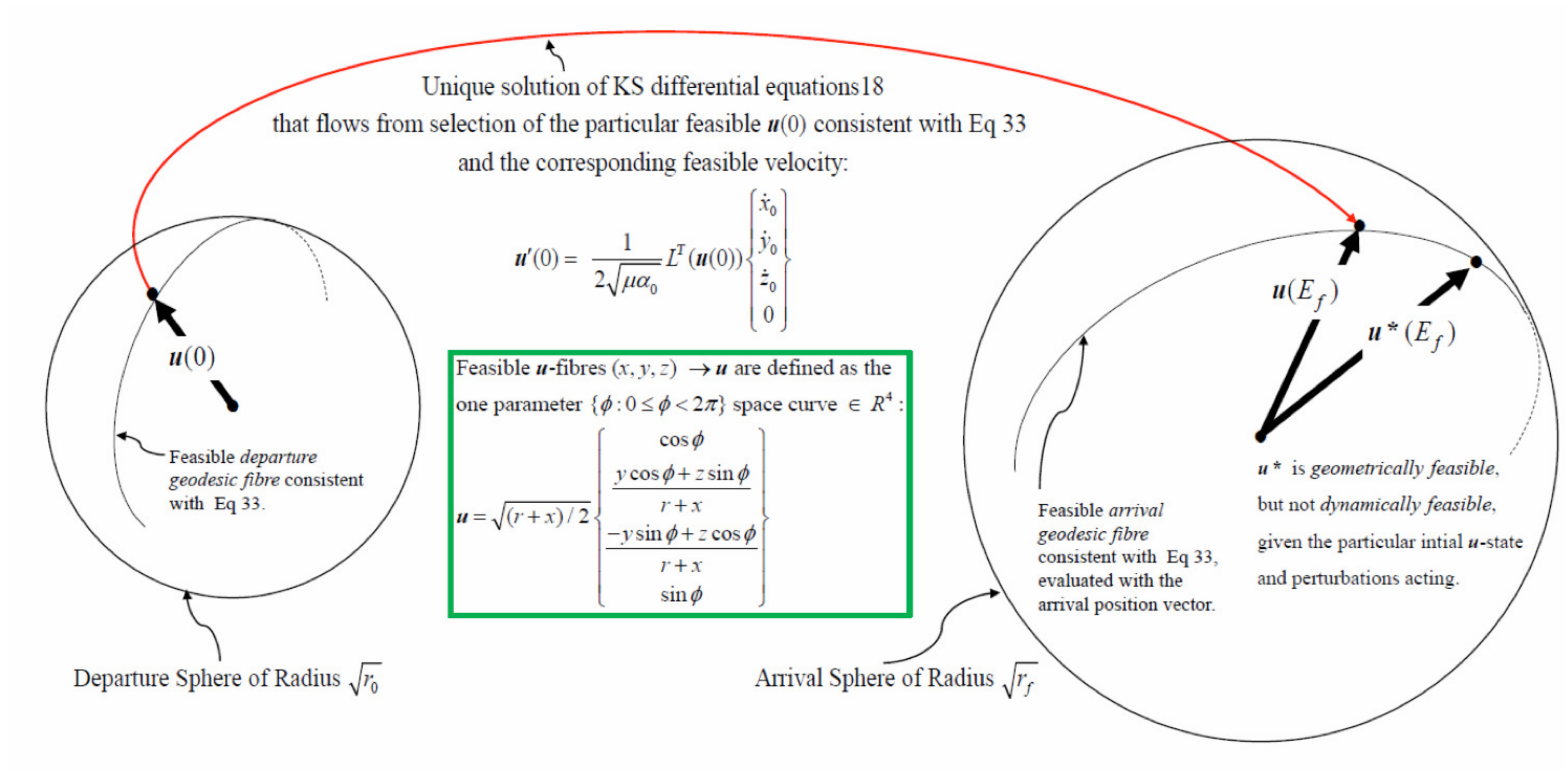

Unique solution of KS differential equations18
that flows from selection of the particular feasible $\boldsymbol{u}(0)$ consistent with Eq 33
and the corresponding feasible velocity:
$\boldsymbol{u}'(0) = \frac{1}{2\sqrt{\mu\alpha_0}} L^T(\boldsymbol{u}(0)) \begin{Bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{z}_0 \\ 0 \end{Bmatrix}$
Feasible $\boldsymbol{u}$-fibres $(x, y, z) \rightarrow \boldsymbol{u}$ are defined as the
one parameter $\{\phi : 0 \le \phi < 2\pi\}$ space curve $\in R^4$:
$\boldsymbol{u} = \sqrt{(r+x)/2} \begin{Bmatrix} \cos\phi \\ \frac{y\cos\phi + z\sin\phi}{r+x} \\ \frac{-y\sin\phi + z\cos\phi}{r+x} \\ \sin\phi \end{Bmatrix}$
$\boldsymbol{u}(0)$
Feasible *departure* *geodesic fibre* consistent with Eq 33.
Departure Sphere of Radius $\sqrt{r_0}$
$\boldsymbol{u}(E_f)$
$\boldsymbol{u}^*(E_f)$
Feasible *arrival* *geodesic fibre* consistent with Eq 33, evaluated with the arrival position vector.
$\boldsymbol{u}^*$ is *geometrically feasible*, but not *dynamically feasible*, given the particular intial $\boldsymbol{u}$-state and perturbations acting.
Arrival Sphere of Radius $\sqrt{r_f}$

Figure 5.10: Four dimensional departure and arrival spheres with feasible geodesics [19].

The perturbed Lambert's problem is simulated with gravity perturbations, using the EGM2008 spherical harmonic gravity model [22]. As with the unknown final boundary condition, discussed in the previous paragraphs, the final eccentric anomaly (E_f) corresponding to the final time (t_f) is also no longer analytically computable. Thus we make use of the two-body final time (and final eccentric anomaly), computed with our analytical KS Lambert solver, as an initial guess for solving the perturbed problem. The analytical KS Lambert solver also provides the two-body trajectory solution that is used as a *warm start* for solving the perturbed problem. It is advantageous to use this regularized KS *warm start* as opposed to a *warm start* generated using the (Cartesian) analytical F&G solution, but either will work. Note the node spacing should be chosen as eccentric anomaly changed mapped onto -1 to 1, using cosine sampling, in order to facilitate efficient convergence of MCPI. Once the Picard iterations have converged, the final time (and final eccentric anomaly) is updated using the secant method. This Picard/secant sequence of iterations is repeated until the final solution satisfies both the specified *time* and *Picard* tolerances. Typically about four or five secant iterations are required for convergence to the desired final time.

All of our perturbed MCPI-TPBVP algorithms make use of a radially adaptive gravity approximation [51, 41], and a variable fidelity force model [42, 41]. While maintaining a machine precision Hamiltonian, these enhancements greatly increase the overall efficiency of our algorithms by reducing the number of "full" force function evaluations required for convergence. Figure 5.11 shows the typical variable fidelity convergence pattern, where the large dots represent the usual ("full") force function evaluations and the crosses represent the low fidelity force function evaluations. Refer to Chapter 4 for more details on the gravity approximations.

To demonstrate the efficiency of our algorithm, we solve the test case orbits in Ta-

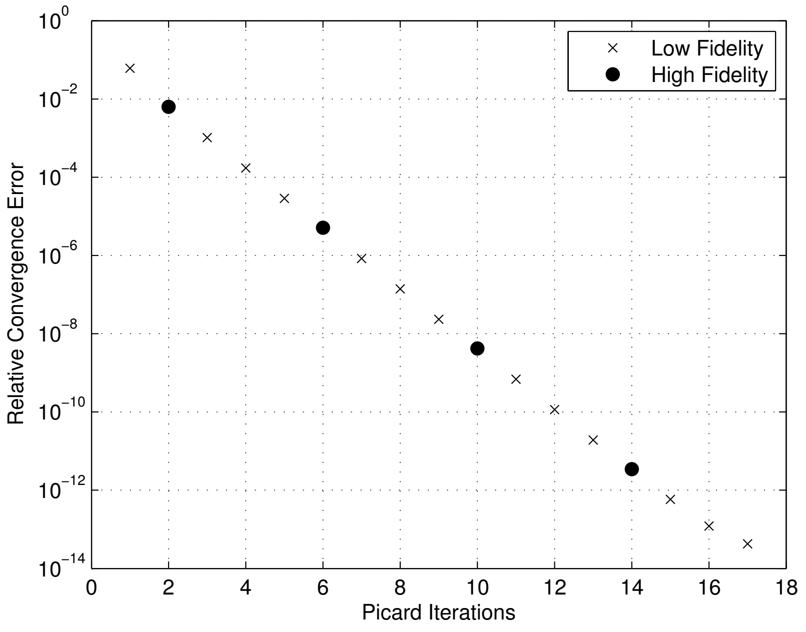


Figure 5.11: Variable fidelity gravity convergence pattern [19].

ble 5.1 using MATLAB's *fsolve* where *Runge-Kutta-Nystrom 12(10)* and *Gauss-Jackson 8* are the adopted integrators. In all cases the methods were tuned to maintain a machine precision Hamiltonian. The results reveal that our regularized MCPI-TPBVP/IVP algorithm is the most efficient method for solving the perturbed Lambert's problem over this interval. That is, over an interval that falls outside of the range attainable using the MCPI-TPBVP in Cartesian variables (about one third of an orbit) and inside of the range attainable using the MCPI-TPBVP in KS variables (just short of 1 orbit). Figures 5.12 to 5.15 shows these results.

5.7 Initial Value Problem: Cartesian vs KS Equations of Motion Solved via MCPI

The KS transformation and Picard iteration can also be applied to solving IVPs, much the way the classical Cartesian differential equations have been solved by MCPI [40]. Similarly, the final two-body eccentric anomaly is determined analytically using

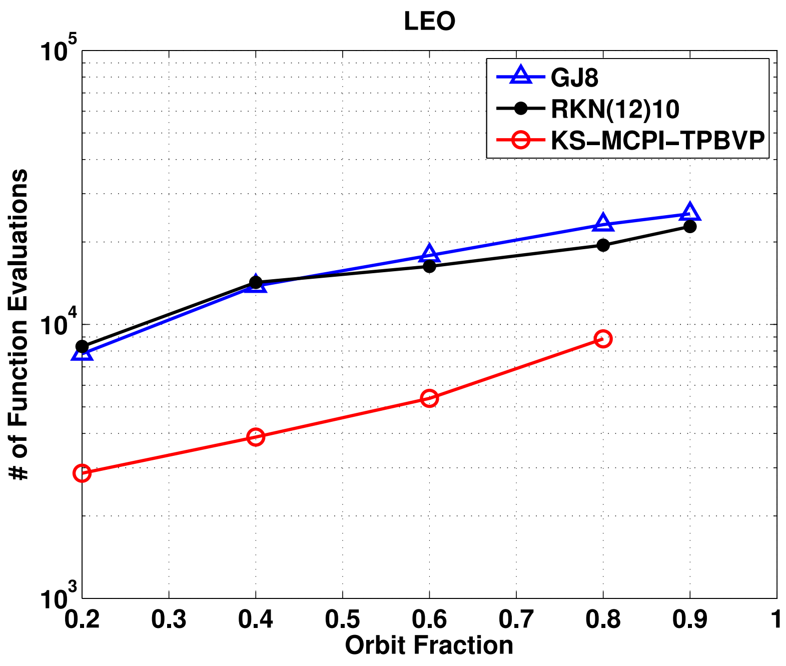


Figure 5.12: Function evaluation comparison of three methods for solving the perturbed Lambert's problem, for a low Earth orbit [19].

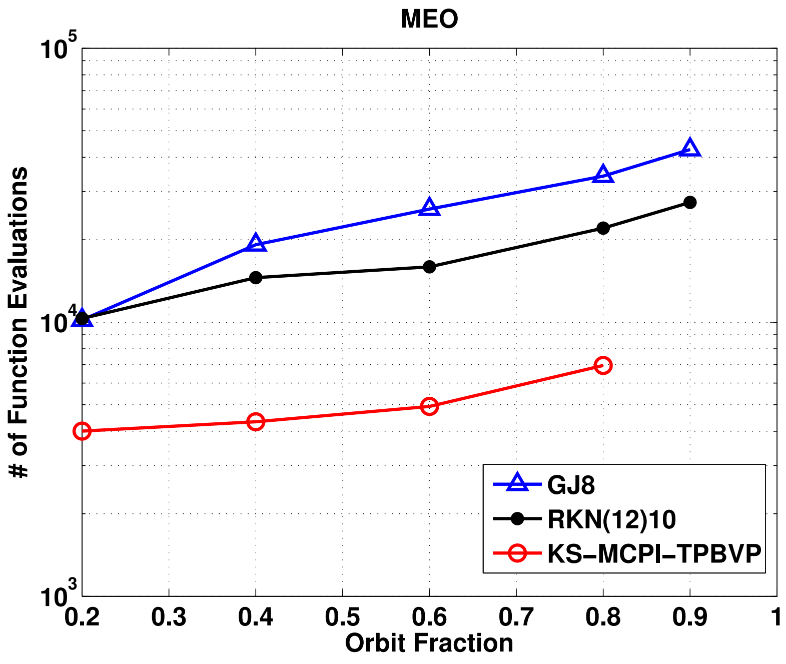


Figure 5.13: Function evaluation comparison of three methods for solving the perturbed Lambert's problem, for a medium Earth orbit [19].

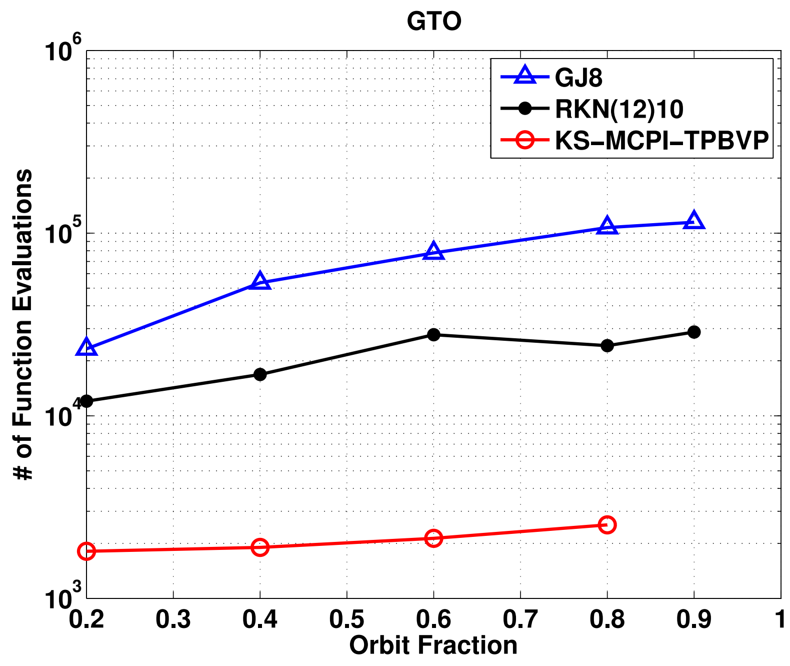


Figure 5.14: Function evaluation comparison of three methods for solving the perturbed Lambert's problem, for a geostationary-transfer orbit [19].

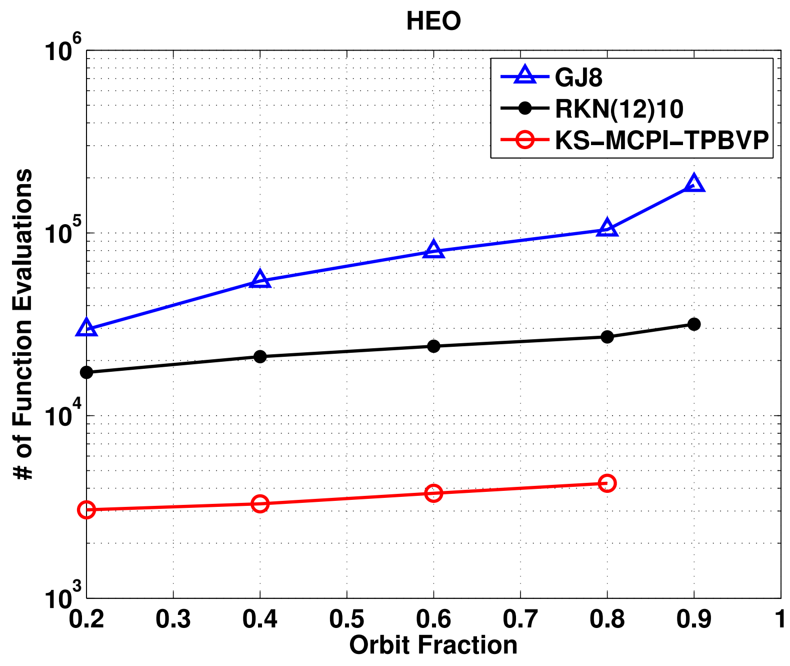


Figure 5.15: Function evaluation comparison of three methods for solving the perturbed Lambert's problem, for a Molniya orbit [19].

the classical Kepler equation for the change in eccentric anomaly given the time. As expected, we find that the domain of convergence achievable for the IVP in KS variables is greatly increased compared with that in Cartesian variables. The KS implementation achieves practical convergence over 8 orbits with no dependence on eccentricity orbits compared with the Cartesian implementation that ranges from about 1.5 to 4 orbits, depending on the eccentricity of the orbit. These numbers agree relatively well with the theoretical values computed in a previous section, more so for the Cartesian implementation than for KS. The theoretical results predict about 12 orbits for the KS implementation, however, our experiments revealed that 2000 nodes are required for the trajectory to converge over 8 orbits. That is, while Picard's method theoretically converges over such long time arcs, the number of basis functions required (> 2000) are usually not attractive. Numerical challenges begin to arise and integrating the motion in "one segment" is not particularly efficient due to slowness of convergence as the eigenvalues approach the unit circle.

Fractional orbit segmentation should be used in practice to reduce the number of iterations (because shorter segments) converge faster due to the approximately linear relationship between number of Picard iterations and segment length. This picture is complicated, however, due to the dependency of convergence on the number of nodes for $N \lesssim 45$, and also the dependency of accuracy on both the number of nodes and the segment length. However, we usually find the optimum number of nodes $30 \leq N \leq 70$. A recent study by Macomber et al. [52] shows that 3 to 5 segments per orbit, with the number of nodes/segments being tuned for each segment length and force model, typically leads to optimal efficiency for the Cartesian equations of motion. Typically, fewer than ten Picard iterations are needed with a warm start over optimal fraction of an orbit segments. Only one or two of these Picard iterations will require the "full" force model. The fixed point nature of convergence permits use of local force approximations to make most of the local force computations extremely cheap compared to the full force model. Thus the number of Picard iteration is not nearly as important as the number of effective full force computations at each node. See Chapter 4 for more details.

All the gravity approximations and optimal segmentation enhancements are included in the MCPI-KS-IVP algorithm. Unlike the perturbed Lambert problem, we do *not* require a secant method to solve for the final eccentric anomaly corresponding to the desired final time. Instead we solve the perturbed KS IVP with the two-body final eccentric anomaly (E_f) plus 2% (we found this conservative and ensures we integrate past the final time of interest). After convergence, it is easy to isolate any "real" E_f interior point by algebraic interpolation/iteration of the Chebyshev orbit approximation. A qualitative difference between this approach and that used for the perturbed Lambert's problem is that a repetitive Picard/secant iteration scheme is not required in each iteration. For the IVP the secant method is only required once, after the Picard iterations have converged, to isolate the final time of interest.

Looking at Figures 5.16 and 5.17 we see that the KS transformation does have a slight advantage over the Cartesian IVP implementation, but it is not as significant as it is for Lambert's problem. When a variable fidelity gravity model is applied, as was done for the perturbed Lambert's problem, we find that the number of full force evaluations required for the solution to converge to machine precision is the same for both the Cartesian and KS implementations. Thus the benefit that the KS implementation provides is a saving of perhaps one or two low fidelity force function evaluations per node.

Using the KS transformation and MCPI for solving IVPs produces nice academic results, however, in reality these do not apparently provide a significant advantage over the already very efficient Cartesian IVP implementation of MCPI [42].

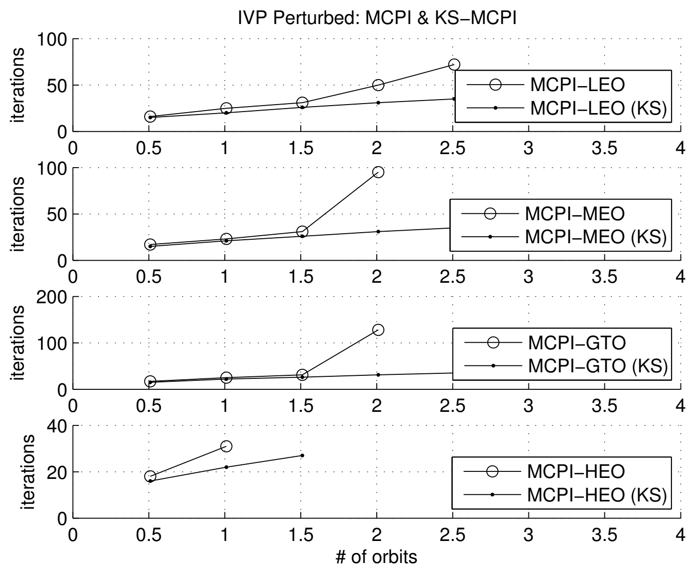


Figure 5.16: MCPI-IVP number of iterations required for convergence: Cartesian vs KS [19].

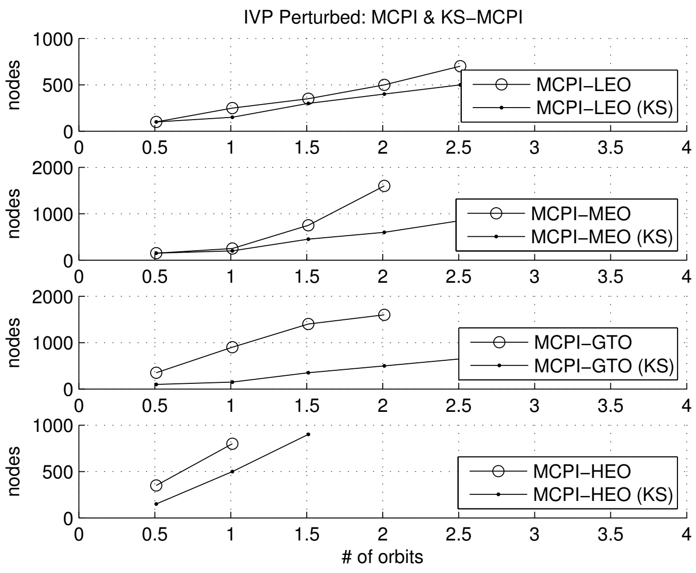


Figure 5.17: MCPI-IVP number of nodes required for convergence: Cartesian vs KS [19].

Pursuing this KS IVP analysis involved considerable effort and enabled us to gain valuable insight about this problem. It is important that this (not what we anticipated or desired) outcome be documented in the event that others consider pursuing this same avenue. While the KS MCPI transformed equations of motion did not show an advantage for solving the IVP, *the advantage remains very significant for solving the perturbed Lambert problem.*

5.8 Summary

A new approach was presented in this chapter for solving TPBVPs and IVPs using the Kustaanheimo-Stiefel transformation and MCPI. The first section introduced the transformation that is used for deriving the perturbed two-body equations of motion in KS variables. A special coordinate system is discussed in the second section that is utilized to avoid certain ambiguities that arise as a result of the non-unique mapping of KS to Cartesian variables. Following this we presented the analytical KS Lambert solver that is used as a warm start for solving the perturbed problem. The sections following outlined the theoretical domain of convergence advantages afforded by KS, and the numerical results of a study conducted to compare three methods for solving the perturbed Lambert's problem. The conclusion of this study is that regularizing the perturbed two-body equations of motion using the KS transformation considerably extends the domain over which the MCPI-TPBVP implementation will converge. We also show significant reductions in the computational costs are achieved by using MCPI-KS-TPBVP for solving Perturbed Lambert's problems.

We note that Figure 5.10 is for heuristic purposes only. The vector $\boldsymbol{u}(E)$ locates a point on a space curve which resides on a time-varying ($E = E(t)$) 4D sphere of radius $\sqrt{\boldsymbol{r}(t)} = \sqrt{\boldsymbol{r}(E(t))}$. The radius is variable in a fairly general fashion in the presence of perturbations but will pass it's minimum at perigee and maximum at apogee (we note the Keplerian motions or perigee and apogee must be generalized for perturbed motion to correspond to the extrema of the perturbed motion.) In reality, all of the infinity of 4D spheres swept out as $|\boldsymbol{u}(E(t))| = \sqrt{\boldsymbol{r}(E(t))}$ varies share a common origin. It is therefore difficult to draw the space curve in $\boldsymbol{u}$-space, and hence, we separated the origins of initial and final spheres in the heuristic sketch of Figure 5.10.

6. METHOD OF PARTICULAR SOLUTIONS*

The method of particular solutions [20] is the third perturbed Lambert algorithm that forms part of the ULT, and it is the focus of this chapter. MPS is a broadly applicable shooting-type method for solving nonlinear two-point boundary value problems but it differs from the much-better-known Newton-shooting method in that integration of the state transition matrix is not required. Unlike the previous two chapters in which algorithms were presented for solving the perturbed Lambert problem over a limited domain of convergence (less than one orbit), MPS converges over multiple revolutions but at the price of increased computational cost. Although less efficient than the previous two algorithms, our studies reveal that MPS is more efficient than the Newton-shooting method, especially when used in conjunction with MCPI. The first two sections of this chapter provide a mathematical description of the method, and this is followed by a section on a-iteration (Keplerian Lambert solver) that is used as a warm start for solving the perturbed problem with MPS. In multi-revolution cases there is not a unique solution to Lambert's problem and this issue is also discussed. Finally, to demonstrate the MPS algorithm, we present results from a simulation to rendezvous with two spent rocket boosters. This work was presented at the 25th Space Flight Mechanics Conference [21], and is accepted for publication in the Journal of the Astronautical Sciences [62].

*Reprinted with permission from "Multiple Revolution Solutions for the Perturbed Lambert Problem using the Method of Particular Solutions and Picard Iteration by R. Woollands, J. Read, A. Probe, J. Junkins, 2016. Journal of Astronautical Sciences, Copyright 2016 by the American Astronautical Society.

6.1 Classical Shooting Method

Consider a natural second order differential equation of the form:

$$\ddot{\boldsymbol{r}} = \mathbf{g}\left(t, \boldsymbol{r}, \dot{\boldsymbol{r}}, \boldsymbol{u}\right), \tag{6.1}$$

where $\boldsymbol{r}$ is an n-vector of "position" coordinates, $\dot{\boldsymbol{r}}$ is an n-vector of "velocity" coordinates, and $\boldsymbol{u}$ is a prescribed function of time m-vector of "control inputs". In this chapter we consider only impulse maneuvers, so we set $\boldsymbol{u} = 0$ during coasting transfer orbits.

The classical solution to the generalized (perturbed) Lambert's problem, for a general right hand side of Eq. 6.1, involves a Newton iteration method whereby we require the $n \times n$ partial derivatives of the terminal position with respect to the initial velocity $[\partial \mathbf{r}\left(t_f\right)/\partial \mathbf{v}\left(t_0\right)]$ to satisfy the $2n \times 2n$ differential equations Eq. 6.2. $[\partial \mathbf{r}\left(t_f\right)/\partial \mathbf{v}\left(t_0\right)]$ is the final time computation of the lower right $n \times n$ sub-matrix of the state transition matrix $\Phi(t, t_0)$.

$$\dot{\Phi}(t, t_0) = \begin{bmatrix} 0 & I \\ \partial\mathbf{g}/\partial\mathbf{r} & \partial\mathbf{g}/\partial\dot{\mathbf{r}} \end{bmatrix} \Phi(t, t_0),\ \Phi(t_0, t_0) = I. \tag{6.2}$$

To implement the classical shooting method, these $4n^2$ differential equations (Eq. 6.2) couple to the $2n$ nonlinear equations of Eq. 6.1. These $4n^2 + 2n$ differential equations can be solved iteratively with the initial velocity updated following each iteration using the Newton iteration shown below. In Eq. 6.3 $\Phi_{21}(t, t_0) = \frac{\partial r(t)}{\partial r(t_0)}$ is the 3×3 $(2, 1)$ sub-matrix of $\Phi(t, t_0)$ from solution of Eq. 6.2.

$$\dot{\boldsymbol{r}}_{k+1}(t_0) = \dot{\boldsymbol{r}}_k(t_0) + \Phi_{21_k}^{-1}(t_f, t_0)\left\{\boldsymbol{r}_k(t_f) - \boldsymbol{r}_f\right\};\ k = 1, 2, ... \tag{6.3}$$

The k subscript denotes the solution of Eqs. 6.1 and 6.2 with the k^{th} iterative trial value for the unknown initial velocity vector $\dot{\boldsymbol{r}}_0(t_0) = \boldsymbol{v}_0$.

For the Lambert problem of orbital dynamics, $\boldsymbol{r}$ is a 3×1 vector of inertial Cartesian coordinates, and $\boldsymbol{v} = \dot{\boldsymbol{r}}$ is the corresponding 3×1 inertial velocity vector. For sufficiently close starting iteration, one can expect convergence in a single digit number of iterations. However, obtaining a sufficiently close starting estimate is not always easy and that is part the challenge for each physical problem. This is especially acute for the case of multi-revolution orbit transfers where there are more than one feasible transfer orbit. In many cases, approximate versions of the governing differential equations permit, for example Battin's solution [1] of the two-body Lambert's problem, to start iterations for the fully perturbed system. Battin's formulation generates all solutions, including the multiple revolution local roots. Utilizing the most attractive of the Keplerian starting iteratives is virtually always good enough for weakly perturbed orbit mechanics problems. When the perturbations are too large for convergence with the analytical two-body Lambert starting approximation, a homotopic method can frequently be designed that sweeps some embedded parameter such that the solution departs from a known solution to ultimately arrive at the solution of the fully perturbed problem at hand. However, in the presence of the most general right hand side of Eq. 6.1, and a poor starting estimate, the problem is more challenging and each iteration has to pay the overhead associated with computing the partial derivatives by solving Eq. 6.2. This is a burden that must be paid to use this method.

We discuss below an alternative local linearity-based approach that is more attractive in that there is no need to compute the state transition matrix and thereby, we avoid solving the $n \times n$ differential equations (Eq. 6.2). This alternate shooting technique is known as the method of particular solutions, as developed in [20].

We combine MPS with the MCPI-IVP integration method and solve the perturbed two-body orbital equations of motion.

6.2 Method of Particular Solutions

The Method of Particular Solutions [20] makes use of a reference trajectory $\boldsymbol{r}_{ref}(t)$, $\dot{\boldsymbol{r}}_{ref}(t), \ddot{\boldsymbol{r}}_{ref}(t)$ and all neighboring solutions of Eq. 6.1 can be re-formulated exactly in terms of a departure motion $\Delta\boldsymbol{r}(t)$ as

$$\boldsymbol{r}(t) = \boldsymbol{r}_{ref}(t) + \triangle\boldsymbol{r}, \;\; \dot{\boldsymbol{r}}(t) = \dot{\boldsymbol{r}}_{ref}(t) + \triangle\dot{\boldsymbol{r}}, \;\; \ddot{\boldsymbol{r}}(t) = \ddot{\boldsymbol{r}}_{ref}(t) + \triangle\ddot{\boldsymbol{r}}. \tag{6.4}$$

From Eq. 6.4 and Eq. 6.1, we can write the exact departure motion differential equation:

$$\triangle\ddot{\boldsymbol{r}} = \boldsymbol{g}\left(t, \boldsymbol{r}_{ref}(t) + \triangle\boldsymbol{r}, \dot{\boldsymbol{r}}_{ref}(t) + \triangle\dot{\boldsymbol{r}}\right) + \boldsymbol{u} - \ddot{\boldsymbol{r}}_{ref}(t). \tag{6.5}$$

Now consider the circumstance that $\ddot{\boldsymbol{r}}_{ref}(t)$ is a solution of the differential equation which satisfies "good" initial boundary conditions, in this case, the Δ's can be expected to be small, $\ddot{\boldsymbol{r}}_{ref}(t) = \boldsymbol{g}\left(t, \boldsymbol{r}_{ref}(t), \dot{\boldsymbol{r}}_{ref}(t)\right)$ and to a linear approximation, the exact nonlinear Eq. 6.5 could be replaced by an approximate linear equation of the form

$$\triangle\ddot{\boldsymbol{r}} = A\triangle\boldsymbol{r} + B\triangle\dot{\boldsymbol{r}} + O(\triangle^2). \tag{6.6}$$

where A, B are time varying Jacobians of $\boldsymbol{g}$ with respect to $\boldsymbol{r}, \dot{\boldsymbol{r}}$ evaluated along $\boldsymbol{r}_{ref}(t)$. To within the accuracy with which the linear terms of Eq. 6.6 approximates the exact departure motion of Eq. 6.5, we can consider the departure motion linear. Note, the reference trajectory is not generally held invariant, the initial conditions can be iteratively updated to reflect improved knowledge. Consider the case that the reference motion satisfies the known left boundary position coordinates exactly $\boldsymbol{r}_{ref}(t_0) = \boldsymbol{r}_0$, and the initial velocity $\dot{\boldsymbol{r}}_{ref}(t_0)$ represents the current best estimate of the unknown initial velocity. For this given (or just computed) $\boldsymbol{r}_{ref}(t)$, consider three neighboring variant trajectories obtained by varying the initial velocity by small linearly independent (typically orthogonal) perturbations. Typically, as a rule of thumb, the initial perturbations should occur in the last three significant digits of the motion; for example if the numerical solution is accurate to 10 digits, the norm of the independent initial condition perturbations could be $|\Delta\dot{\boldsymbol{r}}_j(t_0)| \approx 10^{-7}\,|\dot{\boldsymbol{r}}_{ref}(t_0)|$ to obtain the neighboring initial velocities

$$\boldsymbol{r}_j(t_0) = \boldsymbol{r}_{ref}(t_0) = \boldsymbol{r}_0; \;\; \dot{\boldsymbol{r}}_j(t_0) = \dot{\boldsymbol{r}}_{ref}(t_0) + \triangle\dot{\boldsymbol{r}}_j(t_0); \;\; j = 1, 2, 3. \tag{6.7}$$

Now solve the differential Eq. 6.1 for each of the 3 particular solutions $\boldsymbol{r}_j(t)$. Now we can compute the exact departure motions

$$\triangle\boldsymbol{r}_j(t) = \boldsymbol{r}_j(t) - \boldsymbol{r}_{ref}(t). \tag{6.8}$$

These exact departure motions are particular solutions and conjectured to approximately satisfy the linear differential equation in Eq. 6.6. Since independent velocity initial conditions were used, it is assumed that these trajectories span the space of interest and all neighboring trajectories of interest that also satisfy the linear departure motion Eq. 6.6. The linear combination of any particular solution of a linear differential equation satisfies the differential equation as well, and the general solution as a linear combination of three (in general n) departure motions can be written in the form:

$$\triangle\boldsymbol{r}(t) \approx \sum_{j=1}^{3}\alpha_j\triangle\boldsymbol{r}_j(t) \Rightarrow \boldsymbol{r}(t) \approx \boldsymbol{r}_{ref}(t) + \sum_{j=1}^{3}\alpha_j\triangle\boldsymbol{r}_j(t). \tag{6.9}$$

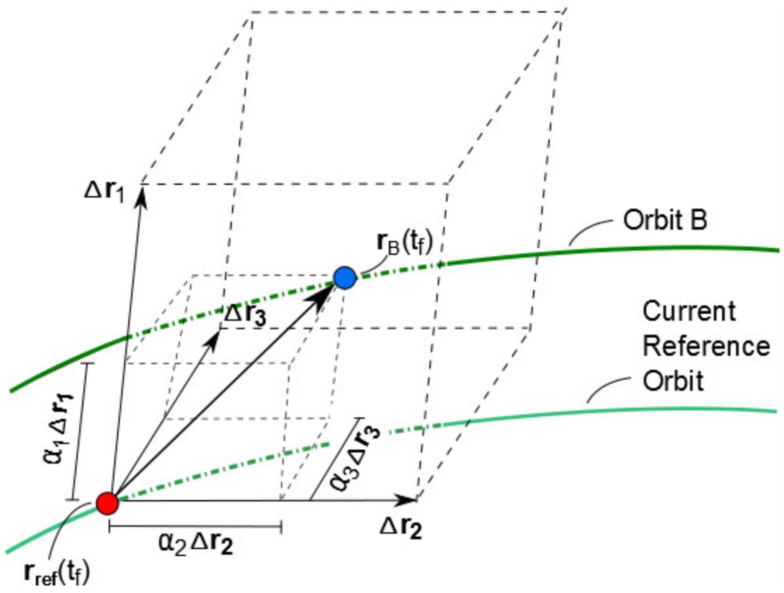


Figure 6.1: A schematic showing the departure motion space for the method of particular solutions [62].

If the Δ's were rigorously in the linear domain, of course, Eq. 6.9 would hold with negligible error. Here we consider only 3 variant trajectories because the admissible initial variations are only the unknown initial velocity coordinates. That is, we contain $\boldsymbol{r}_j(t_0) = \boldsymbol{r}_{ref}(t_0)$ and $\Delta \boldsymbol{r}_j(t_0) = 0$.

Figure 6.1 shows a conceptual example of the departure motion for the three particular solutions. Notice that $\boldsymbol{r}_B(t_f)$, the target position, lies at the vertex of the 3-D parallelpiped, which is a scaled version of the space spanned by $\Delta \boldsymbol{r}_j$; the vector $\boldsymbol{r}_B(t_f)$ is approximated (to within the assumption of linearity) from the linear combination of the three $\alpha_j \Delta \boldsymbol{r}_j$. For the case shown, all three α_j's are less than 1, but the requirement is that the current miss vector $\Delta \boldsymbol{r}_B = \boldsymbol{r}_B(t_f) - \boldsymbol{r}_{ref}(t_f)$ lies in the region approximated by Eq. 6.6.

Evaluating Eq. 6.9 at the final time and imposing the desired result that $\boldsymbol{r}(t_f) = \boldsymbol{r}_f$, leads to the solution for the coefficients of linear combination

$$\begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{Bmatrix} \approx \begin{bmatrix} \triangle\boldsymbol{r}_1(t_f) & \triangle\boldsymbol{r}_2(t_f) & \triangle\boldsymbol{r}_3(t_f) \end{bmatrix}^{-1} \{\boldsymbol{r}_B(t_f) - \boldsymbol{r}_{ref}(t_f)\}. \tag{6.10}$$

Given the α_is, we can compute the departure $\triangle\boldsymbol{r}(t)$ and the time derivatives at any time t.

$$\triangle\boldsymbol{r}(t_f) = \alpha_1\triangle\boldsymbol{r}_1(t_f) + \alpha_2\triangle\boldsymbol{r}_2(t_f) + \alpha_3\triangle\boldsymbol{r}_3(t_f), \tag{6.11}$$

$$\triangle\dot{\boldsymbol{r}}(t_f) = \alpha_1\triangle\dot{\boldsymbol{r}}_1(t_f) + \alpha_2\triangle\dot{\boldsymbol{r}}_2(t_f) + \alpha_3\triangle\dot{\boldsymbol{r}}_3(t_f). \tag{6.12}$$

The velocity departure equation obviously holds at time t_0, so the time derivative of Eq. 6.10, evaluated at time t_0, allows a new estimate for the initial velocity to be calculated.

$$\dot{\boldsymbol{r}}_{new}(t_0) = \dot{\boldsymbol{r}}_{ref}(t_0) + \sum_{j=1}^{3}\alpha_j\triangle\dot{\boldsymbol{r}}_j(t_0). \tag{6.13}$$

We mention that occasionally the volume shown in Figure 6.1 can collapse into a plane, and in extreme cases into a line, as a consequence of the particular sensitivities of the local position variations at certain times and also due to the orthogonal velocity variations at t_0. While rare, these rank deficiencies can be overcome in several ways. The decision to use orthogonal independent velocity variations at t_0 is heuristically reasonable but it is not a constraint. Instead, we can introduce sets of random initial velocity variations. We are also not constrained to use only three particular solutions if a rank deficiency in Eq. 6.13 is encountered.

Given an invertible matrix in Eq. 6.10, Eq. 6.1 can now be re-solved with the reference trajectory's initial velocity replaced by $\dot{\boldsymbol{r}}_{new}(t_0)$. The procedure is repeated to compute three reference neighboring trajectories, which will result in new α's from Eq. 6.13. We iterate for improved α's using Eq. 6.10 and 6.13, analogous Newton's method, but without the necessity of the partials $\frac{\partial \mathbf{r}}{\partial \dot{\mathbf{r}}_0}$ (from computing a state transition matrix).

Any numerical integrator can be used for solving two-point boundary problems with MPS, however using MCPI affords an avenue for increased efficiency that is not available with other step-by-step integrators. We take advantage of the path approximation nature of MCPI (that is, nodes iteratively converge to fixed points in space) and utilize a variable fidelity force model for propagating the reference trajectory as described in Macomber's dissertation [40]. We use the following acceleration computations to approximate a full spherical harmonic gravity model, where TB indicates an unperturbed two-body acceleration; $\mathbf{a}_{low_{ref}}$ and the approximation of the full gravity, $\mathbf{a}_{approx_{ref}}$, are computed first for the reference trajectory:

$$\mathbf{a}_{low_{ref}} = \mathbf{a}_{TB_{ref}} + \mathbf{a}_{(J_2+J_3+J_4+J_5+J_6)_{ref}} \tag{6.14}$$

$$\mathbf{a}_{full_{ref}} = \mathbf{a}_{(40\times 40)}\ spherical\,harmonic\,gravity \tag{6.15}$$

$$\mathbf{a}_{approx_{ref}} = \mathbf{a}_{TB_{ref}} + \mathbf{a}_{(J_2+J_3+J_4+J_5+J_6)_{ref}} + (\mathbf{a}_{full_{ref}} - \mathbf{a}_{low_{ref}}) \tag{6.16}$$

The particular solutions are then assumed to lie close to the reference trajectory, and their accelerations may be approximated using

$$\mathbf{a}_{approx_{particular}} = \mathbf{a}_{TB_{particular}} + \mathbf{a}_{(J_2+J_3+J_4+J_5+J_6)_{particular}} + (\mathbf{a}_{full_{particular}} - \mathbf{a}_{low_{particular}}) \tag{6.17}$$

Remarkably, we demonstrate that computing the particular solutions with only low fidelity approximation function evaluations, that is, *two-body* plus *zonal perturbations* plus the *difference between the full force evaluation and two-body plus zonal perturbations* on the *reference trajectory* , greatly increases the efficiency of the algorithm while maintaining machine precision accuracy. Later we show that solving the perturbed Lambert's problem using MPS with MCPI is about an order of magnitude faster compared with the classical shooting method and a tenth-twelfth order Runga-Kutta (RK(12)10) integrator [63, 64].

6.3 a-iteration

We now discuss an analytical method for solving the Keplerian TPBVP [16]. For the TPBVP, the initial position vectors ($\boldsymbol{r}_1$ and $\boldsymbol{r}_2$) and time of flight ($t_{desired}$) are known, but what is unknown is the initial velocity ($\dot{\boldsymbol{r}}_1$) that is required to reach the final position in the desired time interval. General TPBVP shooting methods require a guess for the initial velocity to propagate the trajectory forward over the desired time interval. The *miss distance* between the current final position and the desired final position is used to refine the initial velocity guess, and the propagation is repeated iteratively until convergence. In contrast, a-iteration requires an initial guess for a (semimajor axis), and assuming two-body motion, the resulting trajectory will always terminate at the desired final position. However, the time of arrival is uncertain and it is unlikely that the initial guess for a would allow the spacecraft to arrive at the desired time. The error in the flight time, based on its sensitivity to a, is what must be iterated in order to determine the correct value of a that corresponds to the desired time of flight.

Prussing [16] provides a comprehensive mathematical explanation of Lambert's problem, and over several pages of algebra leads us to the following transcendental equation that must be iterated via Newton's method to solve for the value of a corresponding to $t_{desired}$.

$$\sqrt{\mu}t_{des} = a^{3/2}\left(\alpha - \beta - \left(\sin\left(\alpha\right) - \sin\left(\beta\right)\right)\right), \tag{6.18}$$

where α and β are given by Eq. 6.19 and Eq. 6.20 respectively, and μ is the gravitational parameter. The parameters c and s are the chord and semiperimeter and are defined in [16].

$$\sin\left(\frac{\alpha}{2}\right) = \left(\frac{s}{2a}\right)^{1/2}. \tag{6.19}$$

$$\sin\left(\frac{\beta}{2}\right) = \left(\frac{s-c}{2a}\right)^{1/2}. \tag{6.20}$$

When solving Lambert's problem there exists a minimum semimajor axis $\left(a_m = \frac{s}{2}\right)$ associated with the minimum energy orbit transfer, and any transfer with a semimajor axis less that a_m will result in an orbit that does not have enough energy to hit the desired target position. Thus a good initial guess for starting the iterations would be $a = 1.001a_m$.

The time of flight associated with this minimum energy transfer is computed as follows:

$$\sqrt{\mu}t_m = \left(\frac{s^3}{8}\right)^{1/2}\left(\alpha_m - \beta_m + \sin\left(\beta_m\right)\right), \tag{6.21}$$

where $\alpha_m = \pi$ and $\sin\left(\frac{\beta_m}{2}\right) = \left(\frac{s-c}{s}\right)^{1/2}$. For $0 \leq \theta \leq \pi$, $\beta_m = \beta_{m_0}$ and for $\pi \leq \theta \leq 2\pi$, $\beta_m = -\beta_{m_0}$. In addition to t_m (transfer time corresponding to minimum energy ellipse) there is also the minimum time in which the elliptic transfer can be made. This is denoted as t_p, or the parabolic transfer time.

$$\sqrt{\mu}t_p = \frac{\sqrt{2}}{3}\left(s^{3/2} - sgn(\sin(\theta))(s-c)^{3/2}\right), \tag{6.22}$$

where the *signum function*, which is defined in Eq. 6.23, takes care of the sign change for transfer angles of $\theta < \pi$ to $\theta > \pi$.

$$sgn(x) = \begin{cases} 1 & for\ x > 0 \\ -1 & for\ x < 0 \end{cases} . \tag{6.23}$$

To summarize, Eq. 6.18 can be used to compute all elliptic transfers in the range $0 \leq \theta \leq 2\pi$ where the values of α and β are determined from the principle values as follows:

$$0 \leq \theta \leq 2\pi, \quad \beta = \beta_0, \tag{6.24}$$

$$\pi \leq \theta \leq 2\pi, \quad \beta = -\beta_0, \tag{6.25}$$

$$t_{des} \leq t_m, \quad \alpha = \alpha_0, \tag{6.26}$$

$$t_{des} > t_m, \quad \alpha = 2\pi - \alpha_0. \tag{6.27}$$

To demonstrate the algorithm, two example transfers are computed between two circular orbits that represent Earth and Mars (Figure 6.2). Both transfers sweep through an angle of 70° but in a different specified time of flight. Figure 6.3 shows a dashed curve that is a Eq. 6.18 swept for various a values. Note the two branches that are separated by t_m. Note that the two orbit transfers shown in Figure 6.2 lie on different branches of the curve shown in Figure 6.3. The solution on the lower branch was computed with $\alpha = \alpha_0$ whereas the solution on the upper branch was computed with $\alpha = 2\pi - \alpha_0$.

6.4 Multiple Revolution Solutions

When the desired time of flight is long enough for the transfer orbit to make one or multiple complete revolutions of the focus ($\theta \geq 2\pi$) we find that the solution to

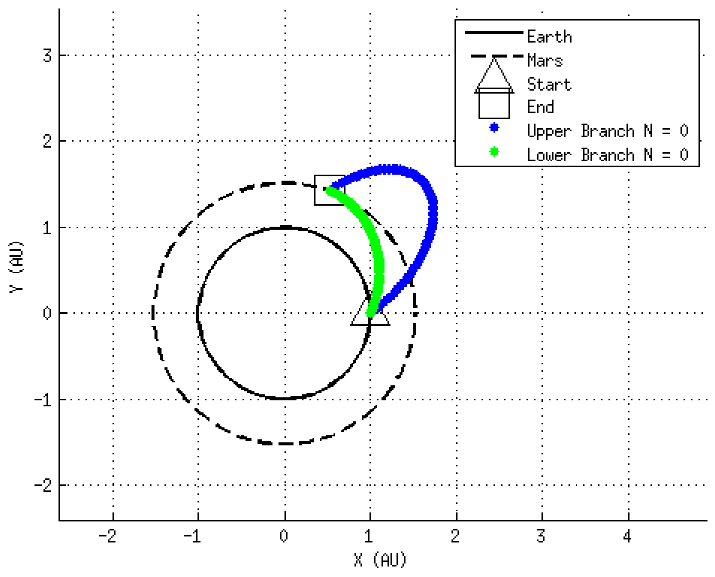


Figure 6.2: Two example transfers computed between two circular orbits representing Earth and Mars.

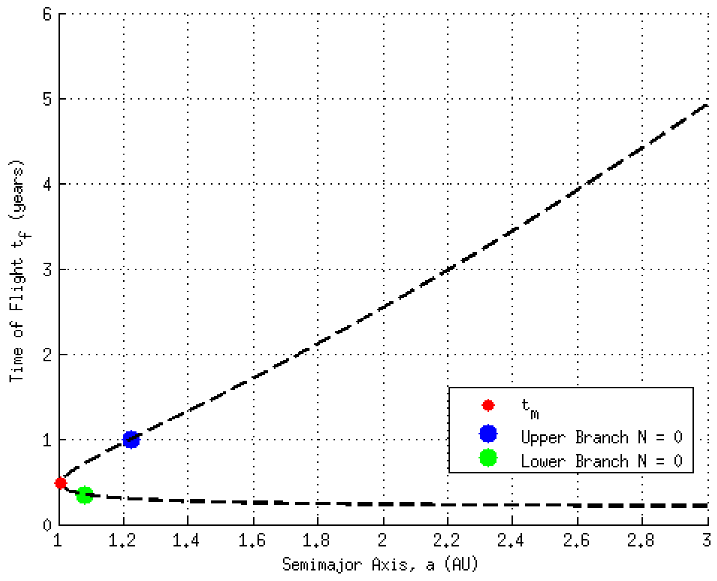


Figure 6.3: Transfer time as a function of semimajor axis for transfers with an angle of $\theta = 70°$ between two circular orbits at 1 AU radii to 1.524 AU.

Lambert's problem is no longer unique, and in fact there are $2N+1$ distinct solutions to the problem. As before, the semimajor axis is related to the desired time of flight through the following transcendental equation that differs only from Eq. 6.18 in that there is an additional term of $2N\pi$ added in the parenthesis.

$$\sqrt{\mu}t_{des} = a^{3/2}\left(2N\pi + \alpha - \beta - \left(\sin\left(\alpha\right) - \sin\left(\beta\right)\right)\right) \tag{6.28}$$

Figure 6.5 shows the time of flight as a function of sweeping the semimajor axis over a certain range for different values of N. Note that on each curve for which $N > 0$ there is a minimum possible transfer time (t_{minN}) which is marked by the blue dots. As expected, there is no minimum time-of-flight for the $N = 0$ case because as the time is decreased the orbit will eventually switch from an elliptic transfer to a parabolic transfer.

The value of t_{m_N} (Eq. 6.29) that corresponds to the minimum energy orbit with semimajor axis $\left(a_m = \frac{s}{2}\right)$ is marked by the red dots in Figure 6.5. This value is critical to the solution process for isolating the multiple roots because it separates each curve into and upper and lower branch, each of which contains a solution.

$$\sqrt{\mu}t_{m_N} = \left(\frac{s}{2}\right)^{3/2}\left(\left(2N+1\right)\pi - \beta_m + \sin\left(\beta_m\right)\right). \tag{6.29}$$

To start the solution process the value of N_{max} must be determined. This is done by root solving $f(a) = 0$ (Eq. 6.30) with different values of $N > 0$. The function $f(a)$ is given by

$$f\left(a\right) = \left(6N\pi + 3\left(\alpha - \beta\right) - \left(\sin\left(\alpha\right) - \sin\left(\beta\right)\right)\right) * \left(\sin\left(\alpha - \beta\right) + \left(\sin\left(\alpha\right) - \sin\left(\beta\right)\right)\right) ...$$

$$-8\left(1 - \cos\left(\alpha - \beta\right)\right), \tag{6.30}$$

where the derivative, which is required for Newton's method, is given by

$$f'(a) = \tfrac{\partial f}{\partial a} = ((6N\pi + 3\xi - \eta)(\cos(\xi) + \cos(\alpha)) \, ...$$

$$+ (3 - \cos(\alpha))(\sin(\xi) + \eta) - 8\sin(\xi))\left(-\tfrac{1}{a}\tan\left(\tfrac{\alpha}{2}\right)\right) ...$$

$$+ ((6N\pi + 3\xi - \eta)(-\cos(\xi) - \cos(\alpha)) \, ...$$

$$+ (-3 - \cos(\beta))(\sin(\xi) + \eta) + 8\sin(\xi))\left(-\frac{1}{a}\tan\left(\frac{\beta}{2}\right)\right), \qquad (6.31)$$

and $\xi \equiv \alpha - \beta$ and $\eta = \sin(\alpha) - \sin(\beta)$. A good initial guess is $a = 1.001a_m$ since the converged value will be bigger than a_m. Once the value of a is found for a particular $N > 0$, t_{min_N} can be computed using Eq. 6.28. If t_{des} is less than say t_{min3}, then $N_{max} = 2$ and there are 5 solutions which must be found. If $t_{des} = t_{min_{N_{max}}}$ then the two solutions on the N_{max} branch are equal and there are a total of 4 unique solutions.

Once the number of solutions is determined, another Newton iteration is require for determining the values of these solutions which are the a's that correspond to the specified t_{des}. That is, Eq. 6.32 must be satisfied iteratively, where Δt is the current transfer time estimate on a particular iteration.

$$g(a) = t_{des}(a) - \Delta t = 0. \qquad (6.32)$$

The derivative, which is also required for Newton's method, is given by

$$\frac{\partial \Delta t}{\partial a} = \frac{\frac{1}{2}\left(\frac{a}{\mu}\right)^{\frac{1}{2}}}{\sin(\alpha - \beta) + (\sin(\alpha) - \sin(\beta))} f(a), \qquad (6.33)$$

where the values of α and β are computed using Eqs. 6.19 and 6.20 respectively. If $\Delta t \leq t_{m_N}$ then $\alpha = \alpha_0$ is used and the solution falls on the lower branch; if

$\Delta t > t_{m_N}$ then $\alpha = 2\pi - \alpha_0$ and the solution falls on the upper branch. If $t_{min_{N_{max}}} \leq \Delta t \leq t_{m_{N_{max}}}$, that is the time of flight on the current iteration is greater than the minimum possible time in which to make the transfer, but smaller than the time of flight corresponding to the minimum energy transfer ellipse, then the situation is a little more challenging as both solutions fall on the lower branch of the curve and thus $\alpha = \alpha_0$ for both solutions. In this case the solutions lie on opposite sides of a_{tmin}, the semimajor axis associated with the minimum possible transfer time, and this information can be used to pick an appropriate initial guess for a.

Figures 6.5 and 6.4 show an example of the multiple solutions that exist for making a transfer between Earth and Mars, through an angle of $\theta = 270°$, in a specified time of flight. Note the five colored dots in Figure 6.5 that represent the five unique orbits shown in Figure 6.4. Each of these transfer trajectories has an associated Δv cost that is require for making the transfer between these orbits.

6.5 Terminal Velocity Vectors

Once the value of a is known the terminal velocity vectors can be computed. These may be written as a set of skewed unit vectors that are co-linear to the local radius and the chord respectively.

$$\boldsymbol{u}_1 \equiv \frac{\boldsymbol{r}_1}{r_1}, \tag{6.34}$$

$$\boldsymbol{u}_2 \equiv \frac{\boldsymbol{r}_2}{r_2}. \tag{6.35}$$

$$\boldsymbol{u}_c \equiv \frac{(\boldsymbol{r}_2 - \boldsymbol{r}_1)}{c}. \tag{6.36}$$

Prussing states that the initial velocity vector ($\boldsymbol{v}_1$) can be expressed as

$$\boldsymbol{v}_1 = (B + A)\,\boldsymbol{u}_c + (B - A)\,\boldsymbol{u}_1 \tag{6.37}$$

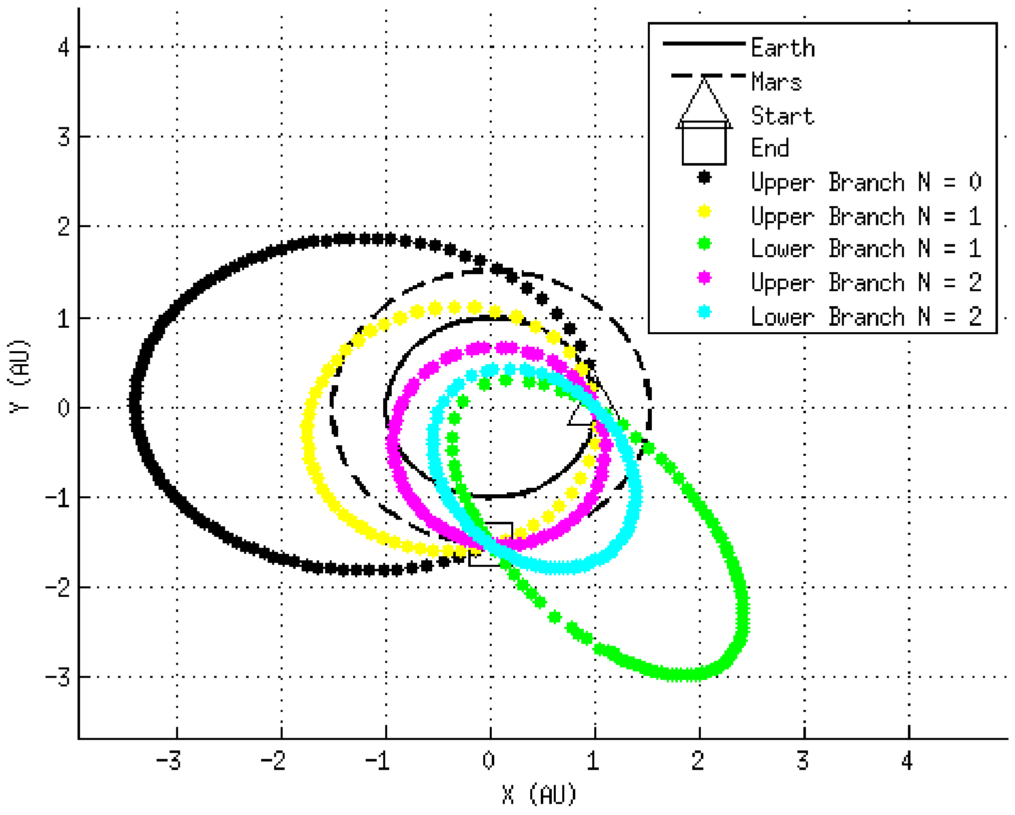


Figure 6.4: Multiple solutions for transferring between two circular orbits, representing Earth and Mars, in the same specified time of flight through an angle of $\theta = 270°$.

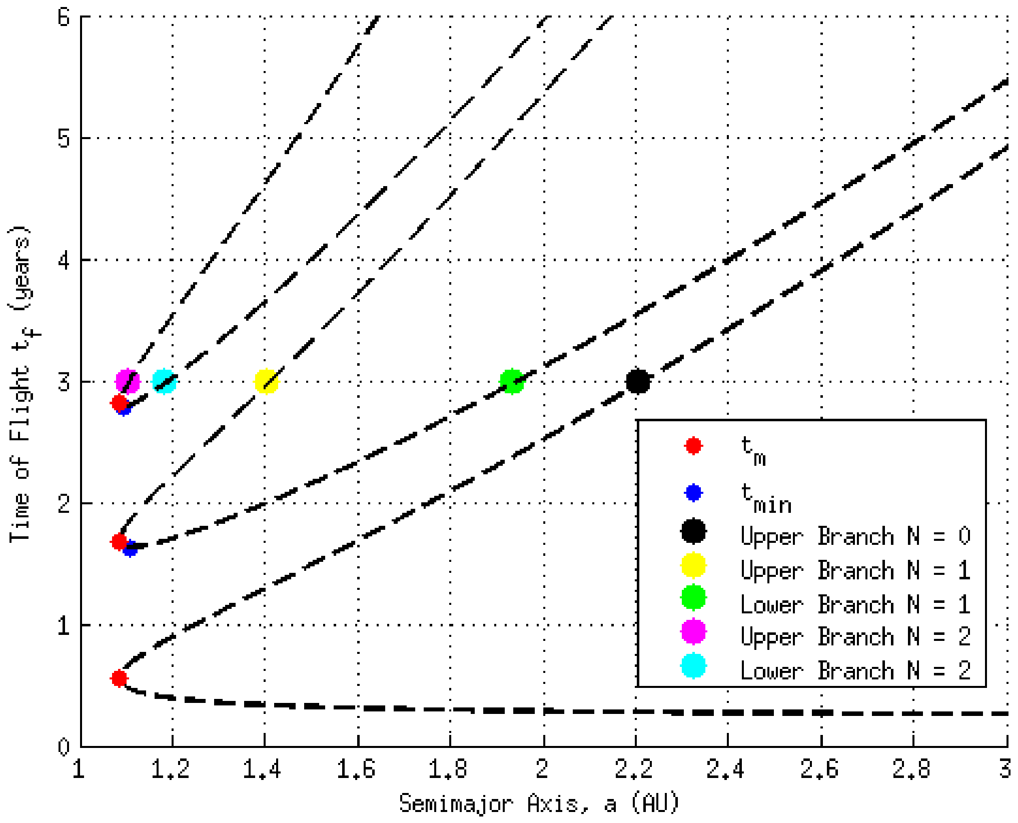


Figure 6.5: Transfer time as a function of semimajor axis for transfers with an angle of $\theta = 270°$ between two circular orbits at 1 AU radii to 1.524 AU. All transfers occur in the same time of flight. The three different dashed black curves represent solutions for the $N = 0$ case (lower curve), $N = 1$ (middle curve) and $N = 2$(upper curve).

where

$$A = \left(\frac{\mu}{4a}\right)^{1/2} \cot\left(\frac{\alpha}{2}\right), \tag{6.38}$$

$$B = \left(\frac{\mu}{4a}\right)^{1/2} \cot\left(\frac{\beta}{2}\right), \tag{6.39}$$

and the values of α and β are determined from Eqs. 6.19 and 6.20 respectively. The final velocity vector ($\boldsymbol{v}_2$) is computed as follows:

$$\boldsymbol{v}_2 = (B + A)\,\boldsymbol{u}_c - (B - A)\,\boldsymbol{u}_2. \tag{6.40}$$

Knowing the value for the initial and final velocity on the transfer orbit allows a Δv value for the specific transfer to be computed. This cost metric is used later in the dissertation to quantify the solution with respect to other possible solutions for making the transfer.

It is important to note that in general, not all the possible solutions are feasible. Some will collide with the Earth, some exceed escape velocity, and others, near π and multiples of π, are undefined as a result of the plane ambiguity that is associated with a 180° transfer in Lambert's problem. In addition, if the transfer orbit Δv exceeds an upper limit imposed by the mission then that solution will also be treated as infeasible.

6.6 Algorithm Efficiency: MPS vs Newton-Shooting

To compare the computational efficiency of MPS to the Newton-shooting method, where RK(12)10 is used as the integrator, we perform three example orbit transfers spanning LEO, MEO and GTO. For each simulation the orbit transfer is computed considering a (40 × 40) degree and order EGM2008 gravity model in a compiled C code environment. A two-body initial guess was used to start each simulation, and in all three test cases MCPI-MPS outperforms the Newton-RK(12)10 shooting method

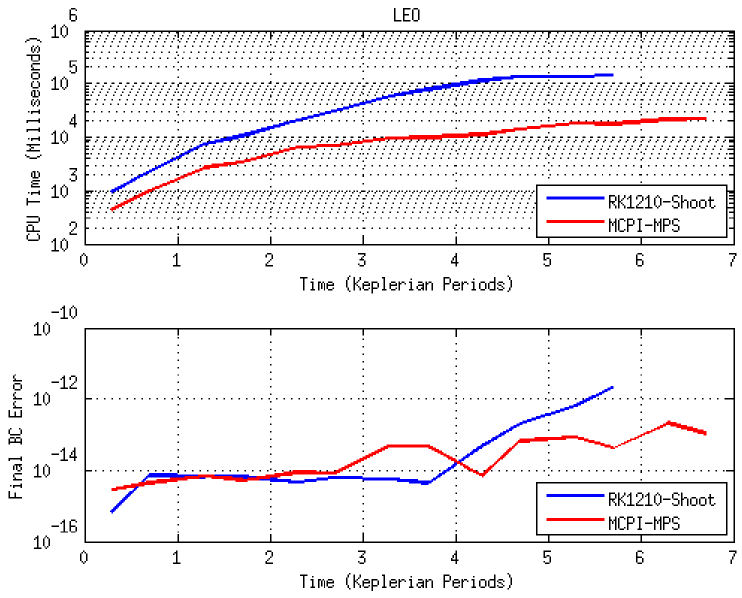


Figure 6.6: Timing comparison for varying time-of-flight LEO transfers using MCPI-MPS and Newton-RK(12)10-shooting [62].

to obtain a machine precision orbit transfer solution in less time. Figure 6.6 shows the computation time in milliseconds for transfers between LEO orbits. Similarly, Figures 6.7 and 6.8 show the computation times for orbit transfers in MEO and GTO.

6.7 Chapter Summary

The method of particular solutions, which is the third perturbed Lambert algorithm that forms part of the ULT was presented in this chapter. MPS is a shooting-type method for solving nonlinear two-point boundary value problems but it differs from the well-known Newton-shooting method in that integration of the state transition matrix is not required. The first two sections of this chapter provided a mathematical description of the method, and this was followed by a section on a-iteration (Keplerian Lambert solver) that is used as a warm start for solving the

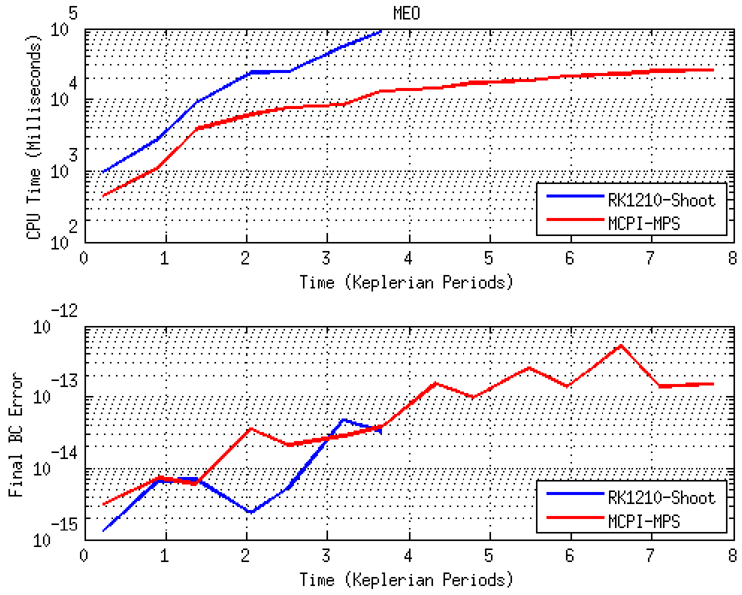


Figure 6.7: Timing comparison for varying time-of-flight MEO transfers using MCPI-MPS and Newton-RK(12)10-shooting [62].

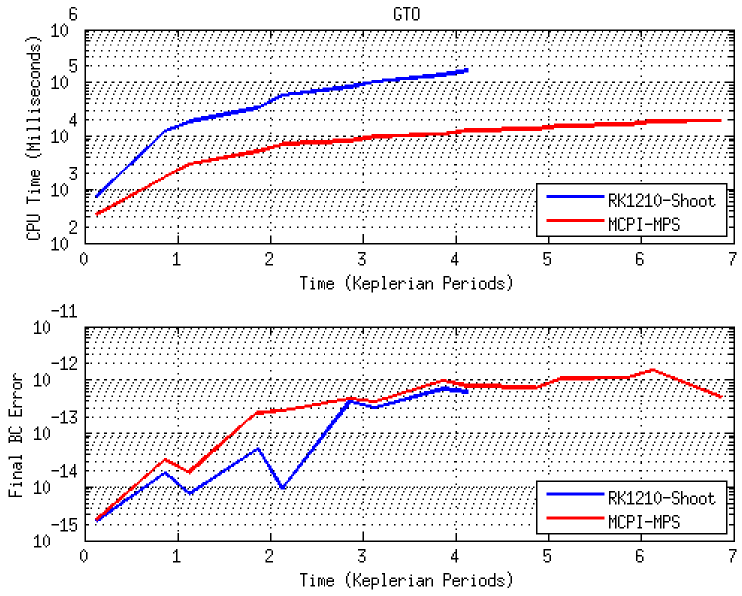


Figure 6.8: Timing comparison for varying time-of-flight Geosynchronous-Transfer orbit transfers using MCPI-MPS and Newton-RK(12)10-shooting [62].

perturbed problem with MPS. In many cases there is not a unique solution to Lambert's problem and the nature of the multiple solutions was discussed in detail in Section 6.4. Finally a numerical study showed the superiority of MPS compared to the Newton-shooting method with regard to efficiency.

7. LOW-THRUST SUB-OPTIMAL TRANSFERS

The mathematical formulation for the Low-Thrust Sub-optimal (LTSO) transfers is presented in this chapter. The method makes use of MPS and the MCPI-IVP to iteratively solve for the coefficients of the Chebyshev polynomial that parameterize each of two steering angles orienting the thrust vector. This unique implementation of minimum norm direct optimization is attractive in that it does not require partial derivatives, yet we have shown that we can converge efficiently using a relatively high dimensional parameterization of the control variables.

7.1 Low-Thrust Sub-optimal Formulation

Consider the situation that a spacecraft is initially in *Orbit A* and it is desired for it to make a transfer to *Orbit B*. The object in *Orbit A* may be considered a "conjectured-to-be-maneuverable" resident space object (RSO) and the object in *Orbit B* as a valuable space asset. The objective is to determine the optimal transfer or set of transfers from *Orbit A to Orbit B*, that will allow for a continuous-magnitude-variable-direction low thrust maneuver to be carried out in near-minimal time. The value of the constant low thrust being applied is constrained by the assumed thrust capabilities of the low thrust engine on board the spacecraft. Typical low thrust values are less than 1 N [65].

The problem is formulated in classical orbital element space, where a is the semi-major axis, e is the eccentricity, i is the orbit inclination, ω is the argument of perigee, Ω is the right ascension of the ascending node, f is the true anomaly, and M is the mean anomaly (see Figure 7.1). Formulating the problem in element space provides several advantages, one being that the first five initial elements $(a_i, e_i, i_i, \Omega_i, w_i)$ and first five final elements $(a_f, e_f, i_f, \Omega_f, w_f)$ can be specified and the time dependent final elements (f_f, M_f) can be left free. That is, the problem can be solved so that the spacecraft arrives at any position in the final orbit, and if desired, slight period miss-match additional phasing can be done to inexpensively rendezvous with the target object. If the problem were to be formulated in Cartesian variables, then both the final position and velocity could be satisfied (six constraints rather than five) in order to be injected into the desired final orbit without additional Δf requirements. This would also mean that only one specific point in that orbit could be targeted rather than admitting any phasing in the orbit. In addition to the above mentioned attributes, formulating the problem in element space also increases the domain over which the MCPI-IVP will converge, by about one order of magnitude [66]. The nominally constant elements are "slow variables" in the presence of low thrust. The elements undergo a much smaller time rate of change than the "fast variables", Cartesian position and velocity. Fast variables are typically changing one order of magnitude faster than the osculating elements.

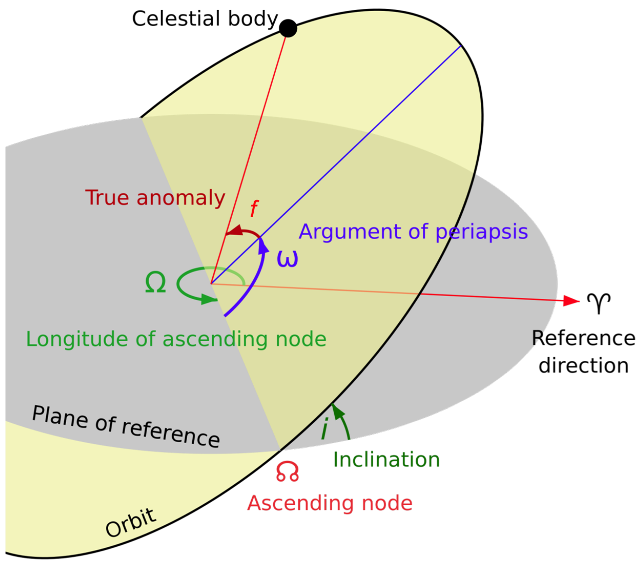


Figure 7.1: Orbit elements [67].

Gauss' Variational Equations, which present the time derivatives of the classical orbit elements in the presence of an arbitrary force, are shown in Eqs 7.1 through 7.7 , where p is the semilatus rectum, h is the magnitude of the angular momentum vector, r is the magnitude of the radius vector, and a_r, a_θ, a_h are the perturbing accelerations in the three orthogonal directions (radial, transverse, normal). The perturbing accelerations in these equations can be just the spherical harmonic gravity perturbation components, or just the components of low thrust perturbations, or the sum of both plus all other perturbations.

$$\frac{da}{dt} = \frac{2a^2}{h}\left(e\sin(f)\,a_r + \frac{p}{r}a_\theta\right), \tag{7.1}$$

$$\frac{de}{dt} = \frac{1}{h}\left(p\sin(f)\,a_r + \left((p+r)\cos(f) + re\right)a_\theta\right), \tag{7.2}$$

$$\frac{di}{dt} = \frac{r\cos(\theta)}{h}a_h, \tag{7.3}$$

$$\frac{d\Omega}{dt} = \frac{r\sin(\theta)}{h\sin i}a_h, \tag{7.4}$$

$$\frac{dw}{dt} = \frac{1}{he}\left(-p\cos(f)\,a_r + (p+r)\sin(f)\,a_\theta\right) - \frac{r\sin(\theta)\cos(i)}{h\sin(i)}a_h, \tag{7.5}$$

$$\frac{df}{dt} = \frac{h}{r^2} + \frac{1}{he}\left(p\cos(f)\,a_r - (p+r)\sin(f)\,a_\theta\right), \tag{7.6}$$

$$\frac{dM}{dt} = n + \frac{b}{ahe}\left((p\cos(f) - 2re)\,a_r - (p+r)\sin(f)\,a_\theta\right). \tag{7.7}$$

The control vector is given by $\boldsymbol{u}(t) = T\left[a_r \;\; a_\theta \;\; a_h\right]$, which can also be written as $\boldsymbol{u}(t) = S\left[\cos(\alpha)\cos(\delta)\,,\; \sin(\alpha)\cos(\delta)\,,\; \sin(\delta)\right]$, where S is the non-dimensionalized magnitude of the constant low thrust induced acceleration, α is the in-plane ($\boldsymbol{i}_r$,$\boldsymbol{i}_\theta$) azimuthal steering angle , and δ is the out-of-plane (elevation) steering angle. These angles are measured relative to the moving unit vectors ($\boldsymbol{i}_r$, $\boldsymbol{i}_\theta$, $\boldsymbol{i}_h$) which osculate the instantaneous position and velocity. For orbit transfers in the vicinity of the Earth, which are required for SSA applications, a range of non-dimensional thrust values are considered from 0.05 N ($S = 8.5 \times 10^{-6}$) to 1 N ($S = 1.7 \times 10^{-4}$), for a 600 kg spacecraft. The non-dimensional distance unit is 1 Earth radii ($DU = 6,738,137$ m) and the corresponding time unit is computed as $TU = \sqrt{DU^3/\mu_\oplus} = 806.8104$ seconds.

In the previous chapter MPS was used to update the initial velocities required for hitting the target position in the desired time of flight. In this chapter MPS is used to solve a minimum norm direct optimization problem by adjusting the low order Chebyshev coefficients that parameterize the steering angles along the trajectory. The essential idea is that a $2m$ family of independent neighboring quasi-linear variations can be generated by introducing $2m$ variations in parametric representation of $\alpha(t)$ and $\delta(t)$. The in-plane and out-of-plane steering angles are approximately parameterized as follows: $\alpha = \sum_{i=0}^{m} \eta_i T_i(\tau);\ m < 10$, and $\delta = \sum_{i=0}^{m} \kappa_i T_i(\tau);\ m < 10$, where η and κ are the respective independent Chebyshev coefficients and T are the Chebyshev polynomials.

Consider the problem in the general form:

$$\begin{gathered}\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{p})\,; \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \quad \boldsymbol{x}(t_f) = \boldsymbol{x}_f, \\ \boldsymbol{p} = [\eta_0, \kappa_0; \eta_1, \kappa_1; ...; \eta_m, \kappa_m]\,; \quad \text{with } \boldsymbol{x} \in R^n; \ \ \boldsymbol{p} \in R^{2m}; \ \ \boldsymbol{p} \geq n; \ \ m \geq n.\end{gathered} \tag{7.8}$$

The augmented system is defined as $\boldsymbol{z} = \left\{ \begin{array}{c} z_1 \\ z_2 \end{array} \right\} = \left\{ \begin{array}{c} \boldsymbol{x} \\ \boldsymbol{p} \end{array} \right\} \Rightarrow \dot{\boldsymbol{z}} = \boldsymbol{F}(t, \boldsymbol{z})$, where $\boldsymbol{p}$ is the vector containing the Chebyshev coefficients that parameterize the steering angles. Suppose there exists some preliminary initial estimate $\boldsymbol{p}_c$ and that a small variation is applied to each element of $\boldsymbol{p}$ in turn. That is $\boldsymbol{p}_i = \boldsymbol{p}_c + \Delta\boldsymbol{p}_i$ where

$$\Delta\boldsymbol{p}_0 = \begin{bmatrix} \Delta\eta_0 \\ \eta_1 \\ \vdots \\ \eta_m \\ \kappa_0 \\ \kappa_1 \\ \vdots \\ \kappa_m \end{bmatrix};\ \Delta\boldsymbol{p}_1 = \begin{bmatrix} \eta_0 \\ \Delta\eta_1 \\ \vdots \\ \eta_m \\ \kappa_0 \\ \kappa_1 \\ \vdots \\ \kappa_m \end{bmatrix};\ldots,\ \Delta\boldsymbol{p}_{m+1} = \begin{bmatrix} \eta_0 \\ \eta_1 \\ \vdots \\ \eta_m \\ \Delta\kappa_0 \\ \kappa_1 \\ \vdots \\ \kappa_m \end{bmatrix};\ldots,\ \Delta\boldsymbol{p}_{2m+1} = \begin{bmatrix} \eta_0 \\ \eta_1 \\ \vdots \\ \eta_m \\ \kappa_0 \\ \kappa_1 \\ \vdots \\ \Delta\kappa_m \end{bmatrix}. \tag{7.9}$$

This means that for each MPS iteration the Gauss' variational equations are integrated 21 times, once for the reference trajectory and once for each of the particular solutions (variation of coefficient). This process could obviously be easily parallelized. Notice that the $\eta_k T_k(\tau)$ and $\kappa_k T_k(\tau)$ parameterize angles measured in radians. The norms of (η_k, κ_k) coefficients are therefore the maximum that the k^{th} term contributes to (α, δ). Thus, it is easy to choose $\Delta\eta_k$ and $\Delta\kappa_k$ to be small but large enough to make a computationally significant variation in $\alpha(t)$ and $\delta(t)$ in the near-linear range. In this case, all of the Δ's were set to 10^{-8}, after experimentation. The $2(m+1)$ resulting trajectories from solving $\dot{\boldsymbol{z}} = \boldsymbol{F}(t, \boldsymbol{z})$ with the i^{th} of the $2(m+1)$ tweaked $\boldsymbol{p}$-vectors are

$$\boldsymbol{z}_i(t) = f(t, \boldsymbol{p}) = f(t, \boldsymbol{p}_i + \Delta\boldsymbol{p}_i) = \boldsymbol{z}_c(t) + \Delta\boldsymbol{z}_i(t). \tag{7.10}$$

All small neighboring departures $\Delta\boldsymbol{z}(t) = \left\{ \begin{array}{c} \Delta\boldsymbol{x}(t) \\ \Delta\boldsymbol{p} \end{array} \right\}$ are approximated as a linear combination combination as $\Delta \boldsymbol{x}(t) = \sum_{i=1}^{m} \gamma_i \Delta x_i(t)$. A particular neighboring $\Delta \boldsymbol{x}(t)$ is desired such that the final departure from $\Delta \boldsymbol{x}_c(t)$ satisfies

$$\begin{aligned}
\Delta \boldsymbol{x}(t_f) &= \boldsymbol{x}_f - \boldsymbol{x}_c(t_f) \\
&= \sum_{i=1}^{m} \gamma_i \Delta \boldsymbol{x}_i(t_f) \\
&= \begin{bmatrix} \Delta x_1(t_f) & \Delta x_2(t_f) & \cdots & \Delta x_m(t_f) \end{bmatrix} \begin{Bmatrix} \gamma_1 \\ \vdots \\ \gamma_m \end{Bmatrix} \\
&\equiv [\Delta X(t_f)]\, \Gamma.
\end{aligned} \tag{7.11}$$

Minimizing $\{\Gamma\}^T \{\Gamma\}$ leads to

$$\Gamma = [\Delta X(t_f)]^T \left([\Delta X(t_f)] \, [\Delta X(t_f)]^T \right)^{-1} \Delta \boldsymbol{x}(t_f). \tag{7.12}$$

Obviously Eq. 7.10 represents an assumption that $\Delta \boldsymbol{x}(t_f)$ lies in a local linear region. LTSO requires a starting iteration. We found a very simple process appears to converge universally. If the final orbit has a larger semimajor axis than the initial orbit, then the thrust vector is initially aligned with the velocity vector ($\boldsymbol{a}_{thrust} = S(\dot{\boldsymbol{r}}/\left|\dot{\boldsymbol{r}}\right|)$) and the equations or motion are integrated until the first time (t^*) that the osculating semimajor axis exceeds the target time. This amounts to a "gravity turn" trajectory. This is the starting trajectory, where the corresponding $\alpha(t)$ and $\delta(t)$ are easily found and used to compute the initial guess for the control.

Once the solution converges for the user specified final time, the procedure is repeated with a reduced final time until the time is reduced to the point where any smaller final time specified prevents the algorithm from converging i.e. no feasible minimum norm solution for the terminal constraints. Obviously if final time is specified smaller than the actual minimum time, then the final state is not reachable. This leads to a near-minimum time trajectory that satisfies the initial and final boundary conditions in the minimum time-of-flight. If m is sufficiently large this sub-optimal solution will typically be graphically identical to the actual minimum time maneuver.

7.2 Algorithm Performance

The domain over which the LTSO algorithm will converge is limited by two factors: the domain of convergence of the MCPI-IVP when integrating Gauss' variational equations, and the domain over which MPS will converge. [66] showed the the domain of convergence of the MCPI-IVP is as much as 50 orbits when integrating Gauss' variational equations. This is substantially more compared to integrating the perturbed two-body orbit equations of motion in Cartesian variables. Although the domain of convergence is large, it does not lead to optimal computational efficiency to integrate such large arcs, as was discussed in Chap 4, and segmentation is used with the average segment size being about one orbit period of the departure orbit (close enough to optimal for this study). These segments are patched together and thus the MCPI-IVP domain of convergence is no longer a limiting factor on the convergence of LTSO.

The domain over which MPS will converge is dependent on the initial starting elements, the thrust level, and the specified time-of-flight. Two examples of low thrust maneuvers that were computed with LTSO are shown in Figures 7.2 (LEO) and 7.3 (MEO). The respective direction components of the normalized thrust vector are shown in Figures 7.4 and 7.5. The components vary smoothly with time implying that the motion of the thruster is smooth and not chaotic. Figures 7.6, 7.7, 7.8 and 7.9 show the variation in the orbital elements for these two transfers. It is clear that the elements also vary slowly over time and that to make a large orbit change would take many spirals. If the target elements exceed the MPS domain of convergence

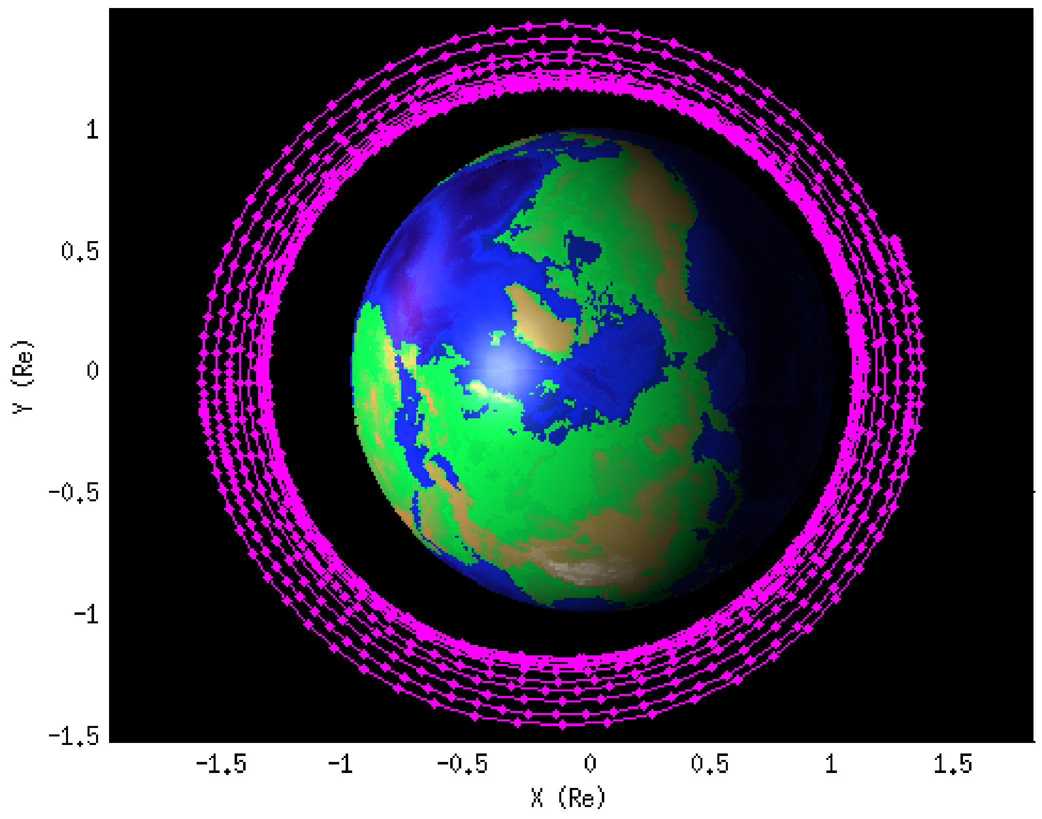


Figure 7.2: Low thrust transfer departing low Earth orbit.

then the problem must be solved using a multiple-shooting method. In the current work multiple-shooting is not considered and is left for future work. It is also evident that the present approach will solve a large family of practical low-thrust transfer problems.

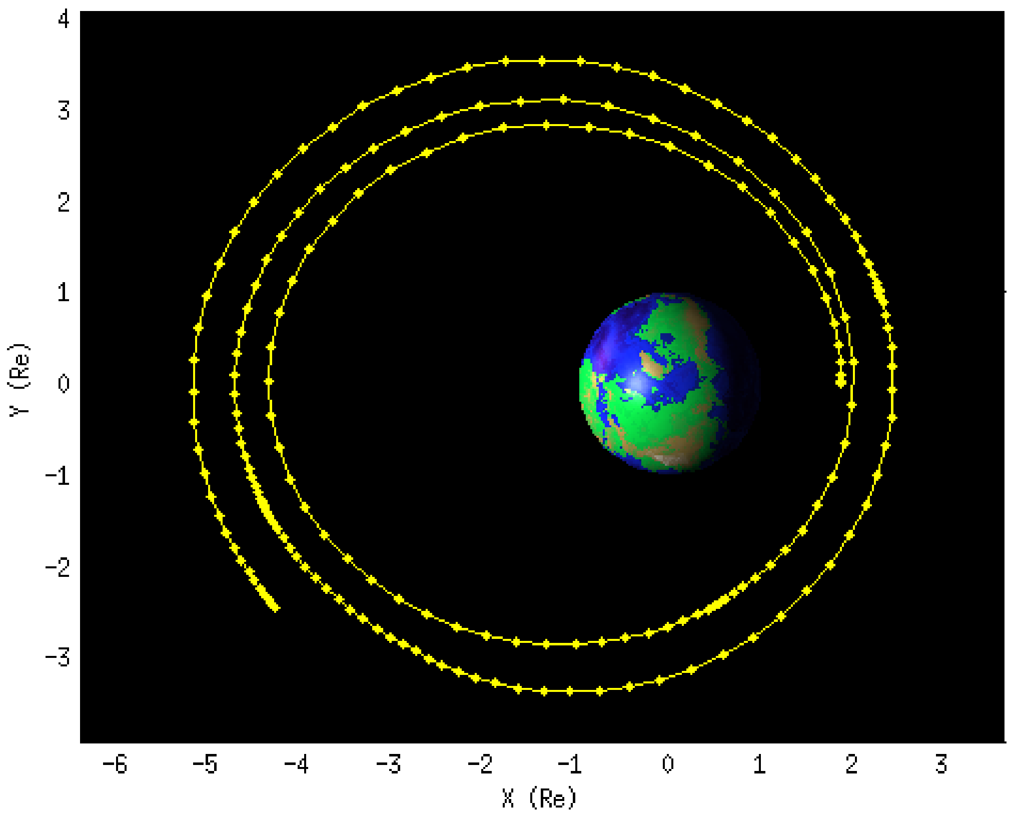


Figure 7.3: Low thrust transfer departing medium Earth orbit.

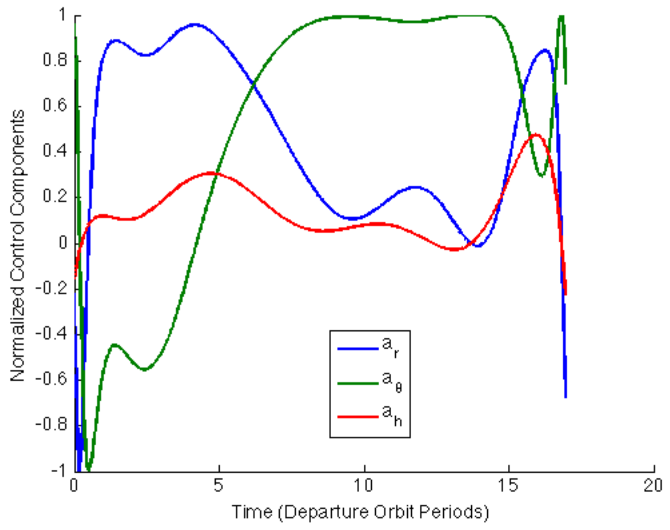


Figure 7.4: Time history of the normalized thrust direction vector for the trajectory shown in Figure 7.2.

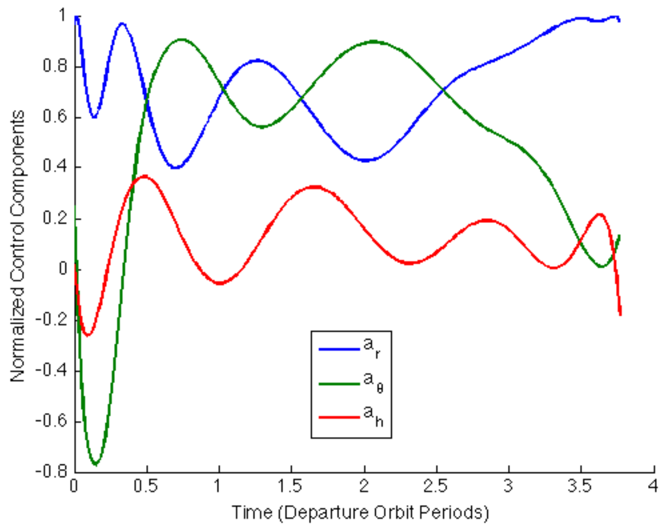


Figure 7.5: Time history of the normalized thrust direction vector for the trajectory shown in Figure 7.3.

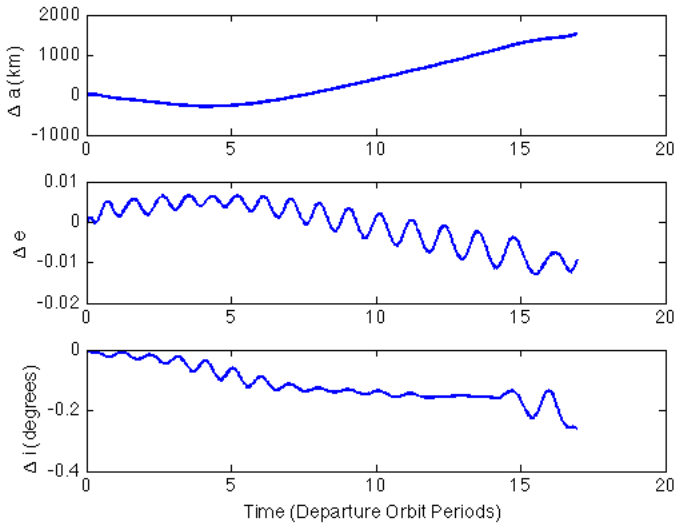


Figure 7.6: Variation in the classical orbit elements (a, e, i) for the low thrust transfer departing low Earth orbit.

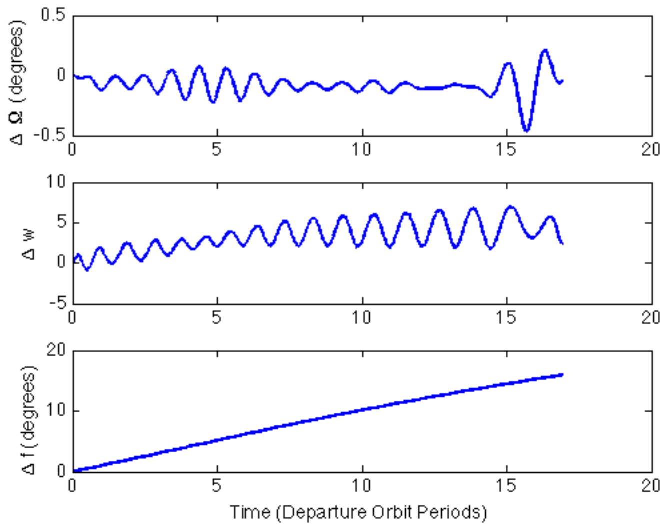


Figure 7.7: Variation in the classical orbit elements (Ω, w, f) for the low thrust transfer departing low Earth orbit.

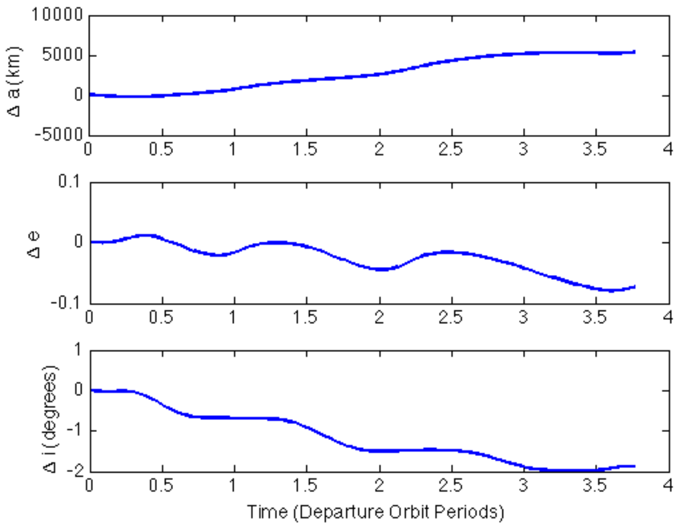


Figure 7.8: Variation in the classical orbit elements (a, e, i) for the low thrust transfer departing medium Earth orbit.

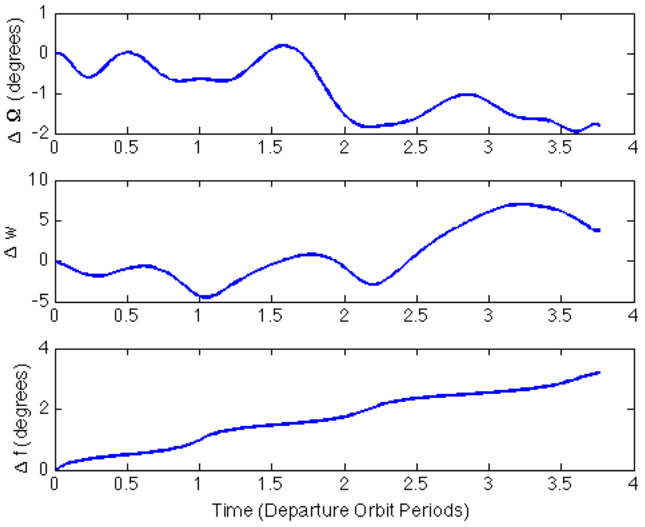


Figure 7.9: Variation in the classical orbit elements $(\Omega,\ w,\ f)$ for the low thrust transfer departing medium Earth orbit.

7.3 Chapter Summary

The LTSO transfer algorithm makes use of MPS and the MCPI-IVP to iteratively solve for the coefficients that parameterize the steering angles of the control vector. This unique implementation of minimum norm direct optimization is attractive in that it does not require solution of auxiliary different equations to compute partial derivatives. The problem is formulated and solved using the classical orbital elements, however, these exhibit singularities for zero eccentricity and zero inclination orbits. There are several ways to overcome this challenge that are mentioned here but will be left for future work (e.g. use of equinoctual elements [66]).

8. UNIFIED LAMBERT TOOL

The ULT is a numerical tool developed by the author in C/C++ that combines the Keplerian Lambert solver (a-iteration) and three perturbed Lambert solvers (MCPI-BVP, MCPI-KS-TPBVP, MPS) into an accurate and efficient means for solving Lambert's problem. The ULT is also implemented in parallel using Message passing Interface (MPI) and when required is run on the Texas A&M LASR Lab SSA Cluster. Many situations require solving hundreds of thousands of Lambert problems and this is where the parallel implementation is particularly useful. The need for solving hundreds of thousands of trajectories arises when one considers solving the challenging SSA data association problem, or for the generation of EFMs to determine an optimal Δv rendezvous maneuver. A case study for these two problems is presented in the latter chapters of this dissertation to demonstrate the capability and power of the algorithm.

8.1 Sub-Algorithms

The ULT consists of four Lambert algorithms written in a C/C++ environment and a suit of MATLAB post-processing tools. Each Lambert problem that the ULT is tasked with solving is first computed using a-iteration [16], the Keplerian Lambert algorithm, and then if a perturbed trajectory is required the ULT automatically selects the best suited perturbed algorithm for the job. There are three perturbed algorithm choices, and the arc length or true anomaly angle spanned by the Keplerian transfer trajectory is the parameter that governs the automated selection of the appropriate perturbed algorithm. This selection is based on the algorithm convergence characteristics of the respective perturbed solvers.

The first perturbed algorithm solves the Lambert problem using the standard MCPI-TPBVP. This algorithm does not require a Newton-like shooting method and is the most efficient of the perturbed solvers presented herein, however the domain of convergence is limited to about a third of an orbit. The second perturbed algorithm extends the domain of convergence of the MCPI-TPBVP solver to about 90% of an orbit, through regularization with the KS transformation. This is the second most efficient out of the perturbed set of algorithms. The third perturbed algorithm uses MPS and the MCPI-IVP for solving multi-revolution perturbed transfers. This method does require "shooting" but differs from Newton-like shooting methods in that it does not require propagation of a state transition matrix. A detailed description of the algorithms and test cases to demonstrate the respective performance were presented in the preceding chapters.

8.2 Parallelization

The ULT is implemented in parallel, using MPI, on the 192 core space situational awareness computer cluster at the LASR Lab, Texas A&M University. The nature of the parallelization is general in the sense that ULT can compute multiple trajectories at any instant in time if they are independent, but it is specific in the sense that a unique "front end" to the ULT is required for solving different types of problems.

For example, to solve the data association problem with radar measurements, the ULT will accept a configuration file with a user specified number of candidate paired initial and final positions (typically hundreds or thousands) and the corresponding user specified times-of-flight. The paired initial and final positions must be pre-computed from the radar measurements, and the time-of-flight corresponds to the time between measurements. In a brute force manner the ULT would consider each pair of position points and attempt to find a solution (initial velocity) that would allow for a feasible transfer between the points in the specified time-of-flight. The only limitation to this computation is the number of compute nodes available for performing the task.

If for example, an EFM is required then the parallelization is slightly different because each trajectory is not independent of each other. That is, information from the previous trajectory (true anomaly transfer angle) is used for computing the next trajectory. There are two possible layers of parallelization that may be utilized. The first layer involves the length of the transfer (single or multi-revolution). The second layer of parallelization is related to each "vertical row" (increasing time-of-flight) on the EFM. As a result the speedup provided by this implementation is only limited by the number of compute nodes available on the LASR Cluster.

8.3 User Input

The user input required by the ULT differs slightly for solving different types of problems. Below is a brief description of the initial conditions and simulation parameters required in the configuration file(s). Figure 8.1 also shows a flow diagram that demonstrates how the ULT operates for generating EFMs.

1. Keplerian OR perturbed final solution?
2. Single trajectory OR EFM?
    (a) Single trajectory
        i. Specify initial and final position (Cartesian or classical elements).
        ii. Specify the desired time-of-flight.
    (b) Data Association
        i. Specify *all* initial and final positions (Cartesian or classical elements).
        ii. Specify *all* the corresponding times-of-flight.

   (c) EFM

      i. Specify departure and arrival orbit (Cartesian or classical elements).

      ii. Specify EFM dimensions.

         A. Maximum time-of-flight (y-axis)

         B. Maximum time past arbitrary starting point (x-axis)

3. Gravity Model (EGM2008 or GMAT)

   (a) Specify spherical harmonic degree and order

   (b) Specify atmospheric drag parameters

   (c) Specify third body effects

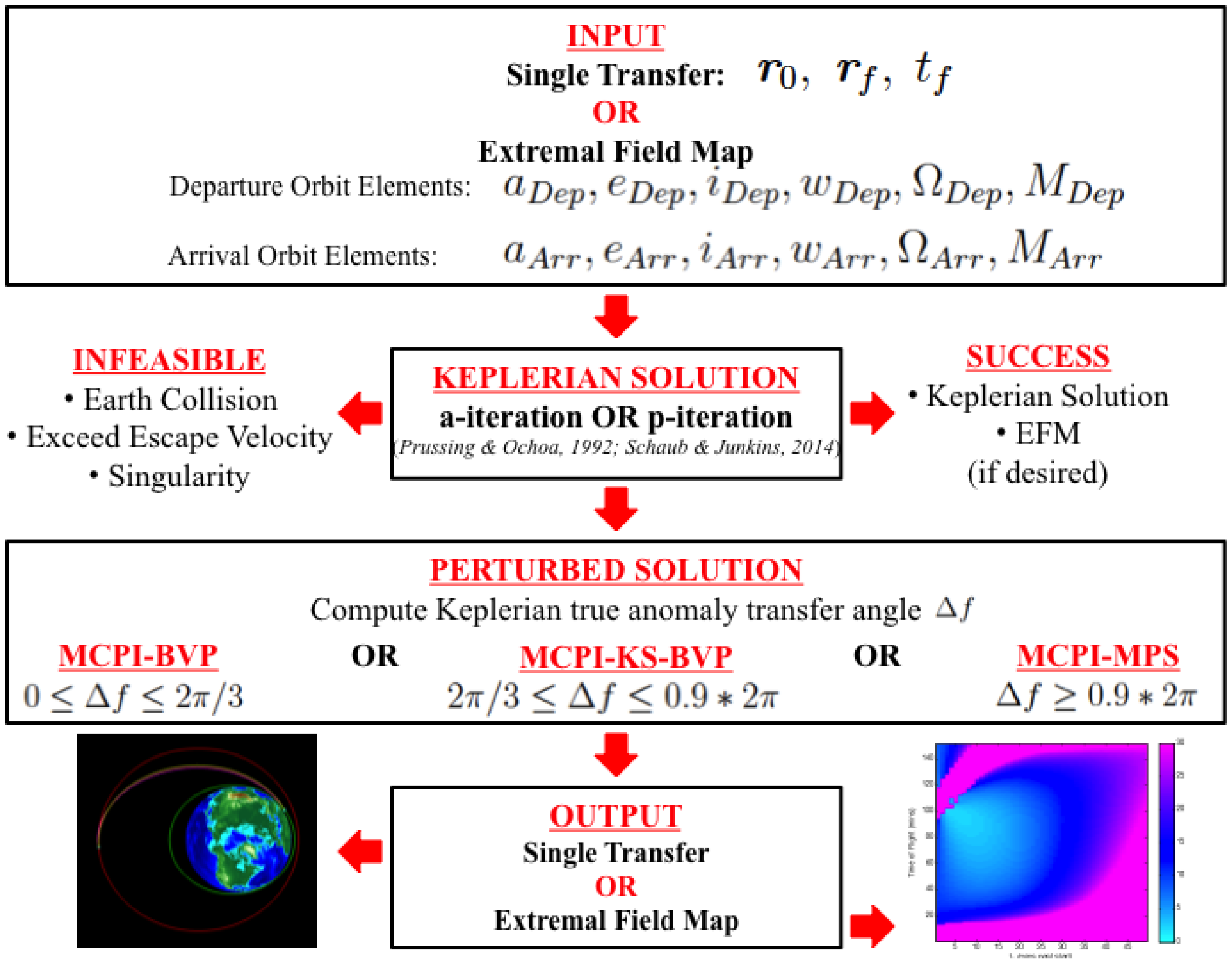

INPUT
Single Transfer: $r_0,\ r_f,\ t_f$
OR
Extremal Field Map
Departure Orbit Elements: $a_{Dep}, e_{Dep}, i_{Dep}, w_{Dep}, \Omega_{Dep}, M_{Dep}$
Arrival Orbit Elements: $a_{Arr}, e_{Arr}, i_{Arr}, w_{Arr}, \Omega_{Arr}, M_{Arr}$
INFEASIBLE
• Earth Collision
• Exceed Escape Velocity
• Singularity
KEPLERIAN SOLUTION
a-iteration OR p-iteration
(Prussing & Ochoa, 1992; Schaub & Junkins, 2014)
SUCCESS
• Keplerian Solution
• EFM
(if desired)
PERTURBED SOLUTION
Compute Keplerian true anomaly transfer angle Δf
MCPI-BVP
$0 \leq \Delta f \leq 2\pi/3$
OR
MCPI-KS-BVP
$2\pi/3 \leq \Delta f \leq 0.9 * 2\pi$
OR
MCPI-MPS
$\Delta f \geq 0.9 * 2\pi$
OUTPUT
Single Transfer
OR
Extremal Field Map

Figure 8.1: Flow diagram showing the unified Lambert tool for generating extremal field maps.

8.4 EFM Output

The output from the ULT consists of several arrays of data that can be plotted to generate EFMs using the MATLAB software in the post-processing folder. There is an option to plot two-body EFMs or perturbed EFMs for departure Δv , arrival Δv, and combined Δv (departure plus arrival), each for a specific number of revolutions. The EFMs shown in this chapter were computed using p-iteration as the Keplerian solver in the ULT. For simplicity only the combined Δv perturbed EFMs are plotted (Figures 8.2 to 8.6). Following this the combined perturbed EFM is shown in Figure 8.7. Notice that towards the top of the EFM there are several "holes" indicating that the perturbed algorithm (MPS) failed to converge. The two-body (not shown) and perturbed EFMs for this test case the were essentially the same to graphical precision (without the holes). Over long time intervals, especially for highly eccentric orbits, the EFM for the perturbed case will differ graphically from the Keplerian case of course.

In the Figures 8.2 through 8.7 the magenta represents the infeasible transfer regions. These may be considered infeasible for a number of reasons, for example the Δv exceeds a limit that is user specified and reflects the physical capability of the engine; the transfer trajectory collides with the Earth; the transfer trajectory is hyperbolic (exceeds escape velocity); the algorithm was unable to converge due to numerical limitations like a transfer through an angle of 180°. The blue regions represent feasible transfer regions, with lighter shades representing lower Δv transfers and darker regions representing more expensive transfers. Notice that a region in the EFM that is infeasible for a transfer trajectory through a specified number of revolutions (i.e. $0 \leq t_f \leq 2\pi$, Figure 8.2) may be feasible for a different number of revolutions (i.e. $2\pi \leq t_f \leq 4\pi$, Figure 8.3). In addition, an expensive transfer for a certain number of revolutions may be cheaper for a different number of revolutions. The appearance of these EFM will of course change based on the initial phasing of the orbiting objects and the shapes of the orbits.

The final plot shows which algorithm performed the computations for generating the perturbed trajectory. Red represents the MCPI-BVP which is the fastest algorithm of the three but has the smallest domain of convergence, yellow represents the MCPI-KS-TPBVP which is the next fastest algorithm, and blue represents the MCPI-MPS which converges over multiple revolutions.

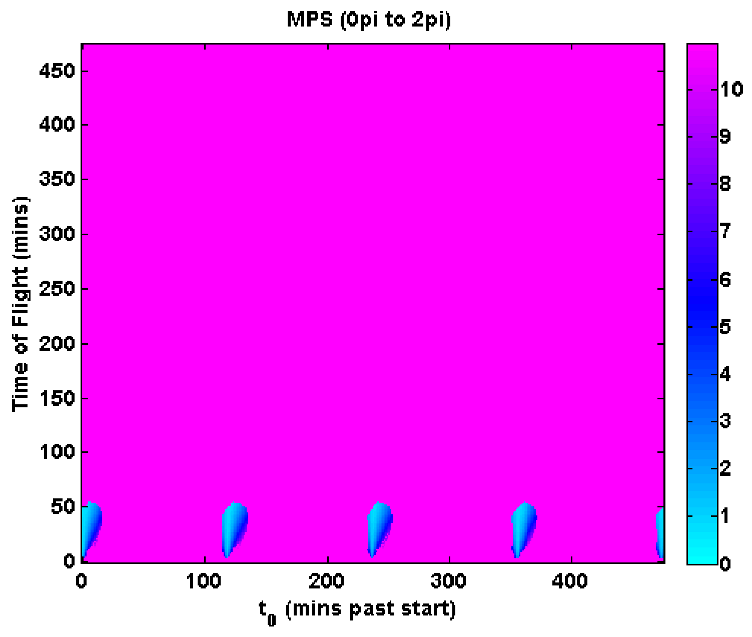


Figure 8.2: Extremal field map for transfers with a true anomaly angle between 0 and 2π. The color bar is Δv squared in $\mathrm{km}^2/\mathrm{s}^2$.

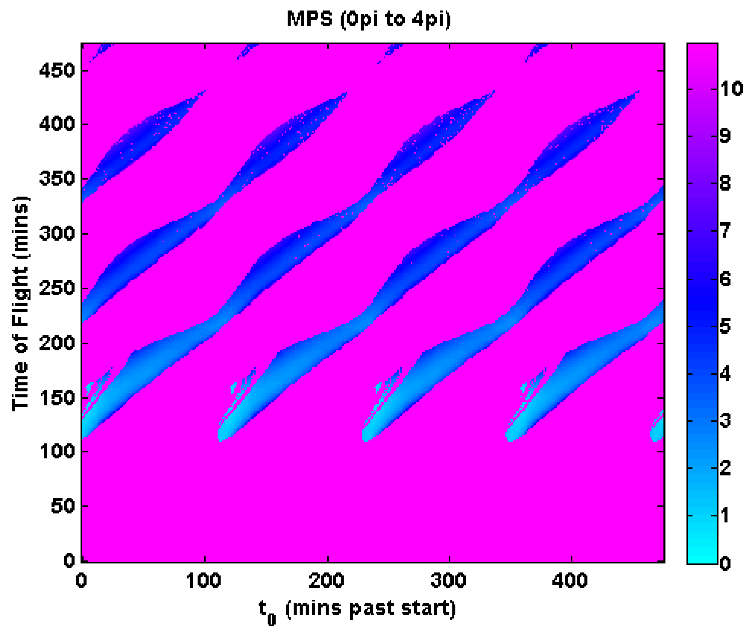


Figure 8.3: Extremal field map for transfers with a true anomaly angle between 0 and 4π. The color bar is Δv squared in $\mathrm{km}^2/\mathrm{s}^2$.

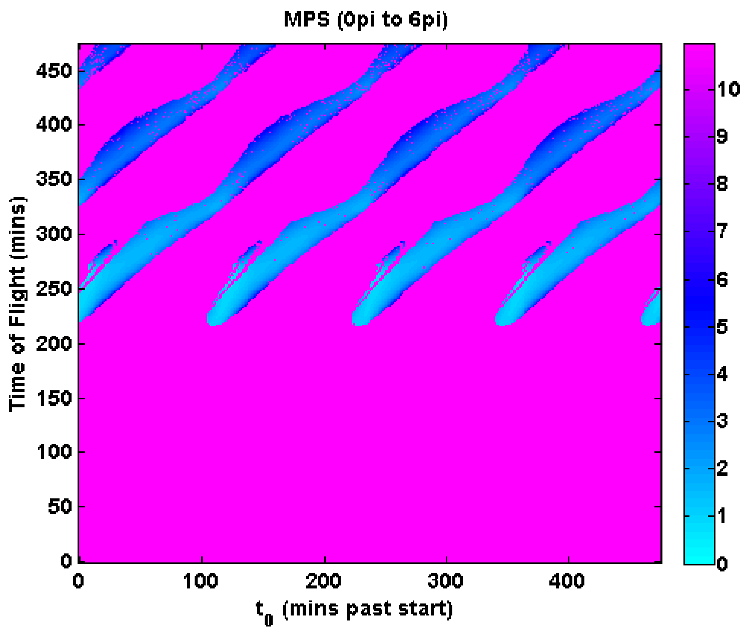


Figure 8.4: Extremal field map for transfers with a true anomaly angle between 0 and 6π. The color bar is Δv squared in $\mathrm{km}^2/\mathrm{s}^2$.

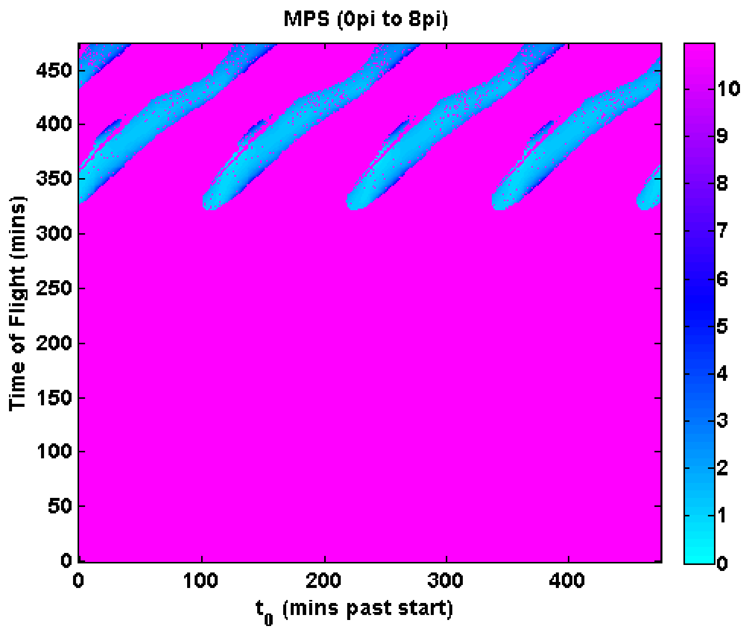


Figure 8.5: Extremal field map for transfers with a true anomaly angle between 0 and 8π. The color bar is Δv squared in $\mathrm{km}^2/\mathrm{s}^2$.

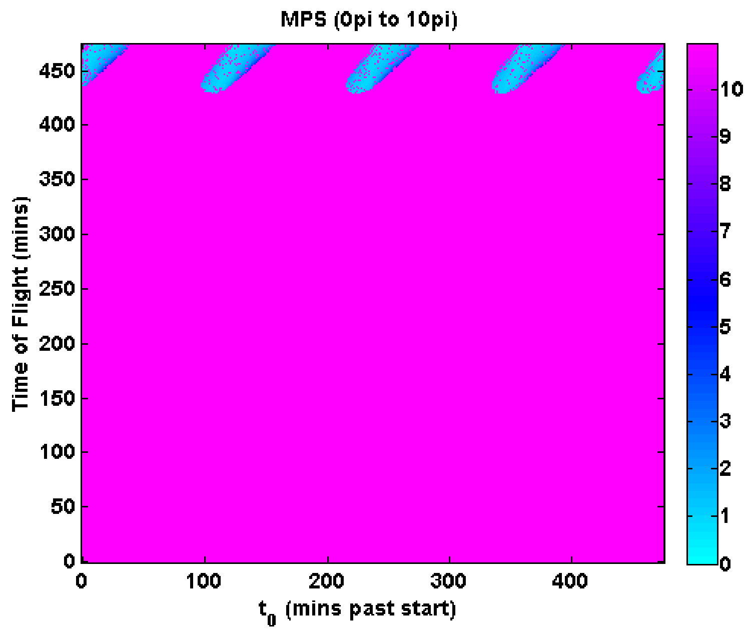


Figure 8.6: Extremal field map for transfers with a true anomaly angle between 0 and 10π. The color bar is Δv squared in km^2/s^2.

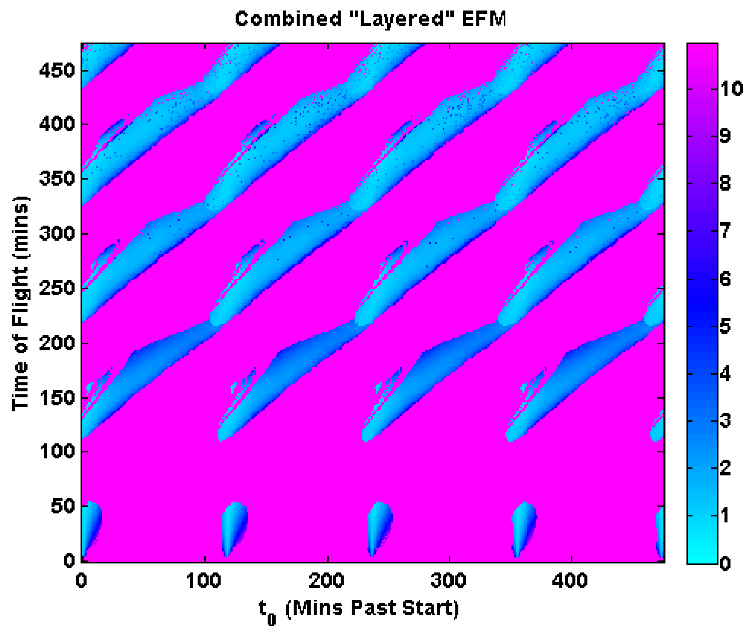


Figure 8.7: Combined perturbed extremal field map showing the minimum cost for all the multi-revolution "layers". The color bar is Δv squared in km^2/s^2.

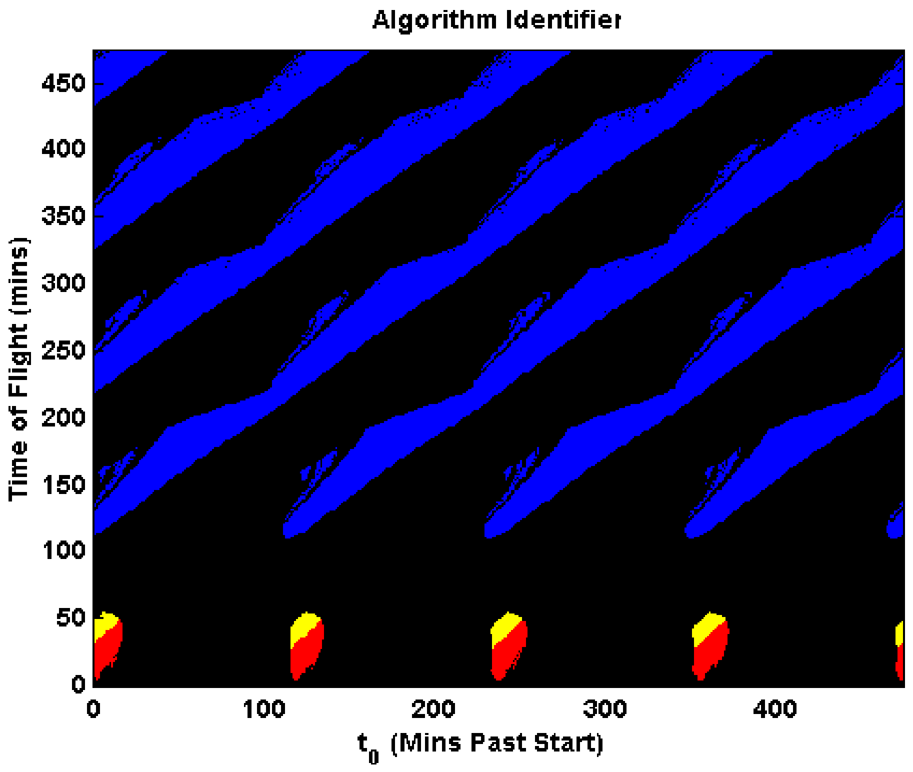


Figure 8.8: Extremal field map showing which perturbed algorithm was utilized for performing the computations. Red represents orbit transfers computed using the MCPI-BVP, yellow represents orbit transfers using the MCPI-KS-BVP, and blue represents orbit transfers using MCPI-MPS.

9. PARALLEL GENERATION OF EXTREMAL FIELD MAPS

There are over 500 USA-launched and over 1000 Soviet-launched spent boosters in low Earth orbit. The combined mass of these upper-stage boosters represent approximately half of all the mass of debris objects orbiting the Earth. Orbital debris is hazardous to operational satellites and reducing the danger is possible by orbit rendezvous, capture and de-orbit missions directed at the most high priority debris objects. Determining the optimal maneuver sequence for rendezvous with these large derelict objects requires simulating hundreds of thousands of feasible transfer trajectories. In this chapter the ULT is used for generating EFMs in order to determine the optimal maneuver sequence for rendezvous with two spent boosters.

9.1 Optimal Maneuver Sequence

We simulate a "retrieval" spacecraft in a reference orbit and require it to rendezvous with two pieces of debris in different LEO orbits (Orbit 1 and Orbit 2 in Table 9.1). Each piece of debris is assumed to be a spent Delta Upper Stage (mass approx 6000 kg). The ULT (in particular p-iteration [61]) is used to compute the two-body transfer trajectory for each feasible solution, and the Δv values are shown in an EFM.

The mass was selected based on the Δv values required to place the debris in an re-entry orbit with a perigee radius of 100 km. We perform a "representative" calculation to determine the approximate mass of fuel required for conducting the mission, assuming a specific impulse of 358 s ($\approx$ kerosene or RP1) and a final mass of no less than 20% of the original mass of the retrieval vehicle. That is, 80% of the retrieval vehicle is assumed to be fuel. This rough computation provides some numbers in the relevant domain for conducting the simulation, however, more accurate computations would of course be required for the engine, propellant and spacecraft considered for a specific mission.

The mission requirement is to carry out the maneuvers and rendezvous with both pieces of debris using the minimum Δv. Whether to rendezvous with object 1 first and then fly to object 2, or vice versa, is unknown. To determine this we generate an EFM for transfers from the Reference to Orbit 1 (Figure 9.1). The transfer trajectory that corresponds to the point on the EFM with the minimum Δv is selected as the first phase in this transfer sequence. A second EFM is then generated for the transfers from Orbit 1 to Orbit 2 (Figure 9.2). The trajectory that resulted in the minimum Δv transfer from Orbit 1 to Orbit 2 is selected as the second phase in the sequence. Following this the total Δv for the sequence is computed. The same procedure is performed for computing the minimum Δv to transfer from the Reference to Orbit 2 (Figure 9.3), and then to Orbit 1 (Figure 9.4). The EFMs generate the local extrema for a large family of take-off and arrival times for each transfer so that the global extremal can be found.

Figure 9.1 is the EFM for transfers from the Reference orbit to Orbit 1. The maximum Δv cost displayed on the plot is 3.5 km/s. The color bar reveals how the cost varies for different starting positions past perigee (horizontal axis) and varying time-of-flight (vertical axis). The black represents all infeasible regions, and regions where the cost exceeds 3.5 km/s. It is interesting to note that towards the top of the figure some yellow regions meet with red regions. Each of these represent a different type of transfer (lob, first multi-revolution, second multi-revolution and so on). In certain circumstances, as one type of transfer may become infeasible, i.e. the multi-revolution transfer may collide with the Earth, another more expensive type (perhaps the lob) will become the transfer option (red).

Looking at Figure **??** it appears that the optimal transfer would be one of the

Table 9.1: Orbital elements [62].

Elements	Reference Orbit	Orbit 1	Orbit 2
Semimajor axis (A)	7500 km	7000 km	8000 km
Eccentricity (e)	0.1	0.05	0.07
Inclination (i)	28.5°	33.5°	23.5°
RA of Ascending Node (Ω)	0°	0°	0°
Argument of Periapses (w)	0°	0°	0°

Table 9.2: Δv Maneuvers [62].

Maneuvers	Sequence 1	Sequence 2
Acceleration Δv_1	0.5658 km^2/s^2	0.3798 km/s
Decceleration Δv_1	0.3532 km^2/s^2	0.2977 km/s
De-orbit Δv_1	0.0495 km^2/s^2	0.2468 km/s
Acceleration Δv_2	0.7329 km^2/s^2	0.6842 km/s
Decceleration Δv_2	0.6556 km^2/s^2	0.7268 km/s
De-orbit Δv_2	0.2468 km^2/s^2	0.0495 km/s
TOTAL Δv	2.6040 km^2/s^2	2.3850 km/s

multi-revolution cases where the shading is yellow. The magenta star at the top right is the global minimum on this EFM, and this trajectory is selected for making the transfer between the Reference orbit and Orbit 1.

Figure 9.2 is the EFM for all feasible transfers from Orbit 1 to Orbit 2. The global minimum is shown with the magenta star. Figure 9.3 is the EFM for all feasible transfers from the Reference orbit to Orbit 2, with the global minimum depicted by the magenta star. Finally, Figure 9.4 is the EFM for all the feasible transfers from Orbit 2 to Orbit 1. Again the global minimum is marked with the magenta star.

Figures 9.5 and 9.6 show the transfer sequences from the *Reference to Orbit 1 to Orbit 2* and from the *Reference to Orbit 2 to Orbit 1* respectively. For the first case

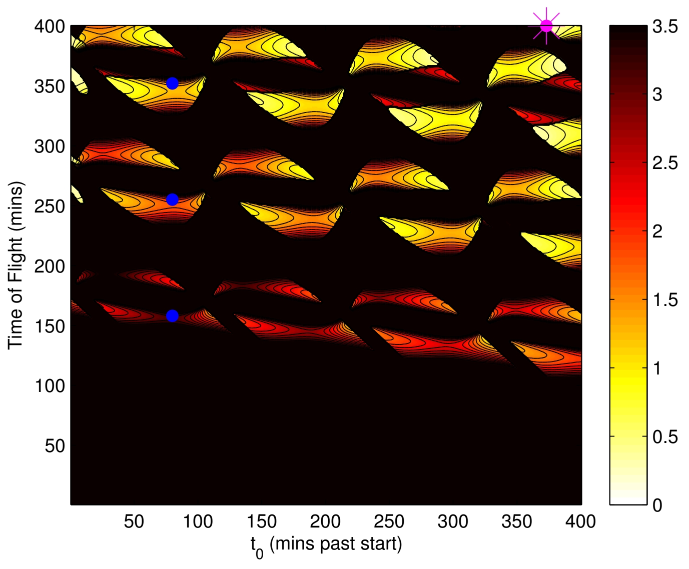


Figure 9.1: Two-body EFM showing the minimum velocity orbit transfer maneuvers between the Reference orbit and Orbit 1. The maximum allowable Δv is $3.5\text{km}^2/\text{s}^2$ for this 5° inclination plane change. The magenta star marks the global minimum Δv transfer. The blue dots represent the example transfer trajectories shown in Figure **??** [62].

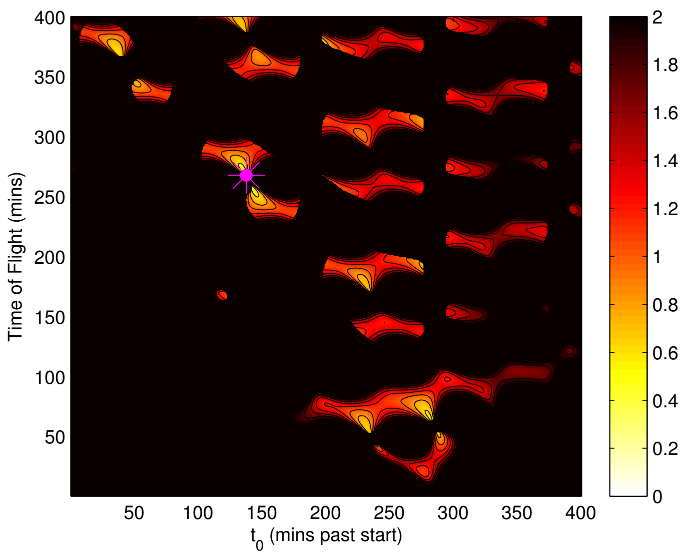


Figure 9.2: Two-body EFM showing the minimum velocity orbit transfer maneuvers between the Orbit 1 and Orbit 2. The maximum allowable Δv is $2\text{km}^2/\text{s}^2$ for this 10° inclination plane change. The magenta star marks the global minimum Δv transfer [62].

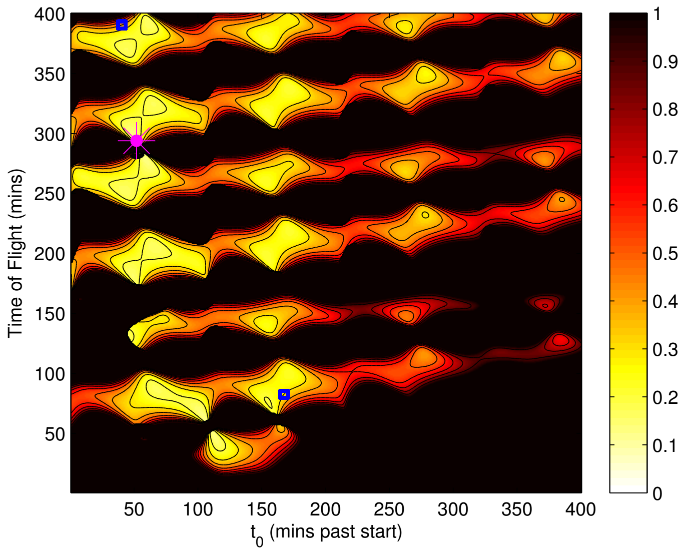


Figure 9.3: Two-body EFM showing the minimum velocity orbit transfer maneuvers between the Reference and Orbit 2. The maximum allowable Δv is $1 \text{km}^2/\text{s}^2$ for this 5° inclination plane change. The magenta star marks the global minimum Δv transfer [62].

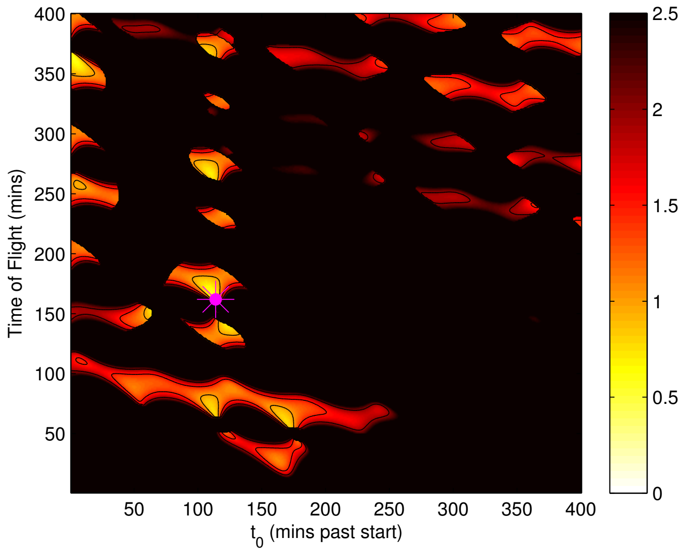


Figure 9.4: Two-body EFM showing the minimum velocity orbit transfer maneuvers between the Orbit 2 and Orbit 1. The maximum allowable Δv is $2.5 \text{km}^2/\text{s}^2$ for this 10° inclination plane change. The magenta square star the global minimum Δv transfer [62].

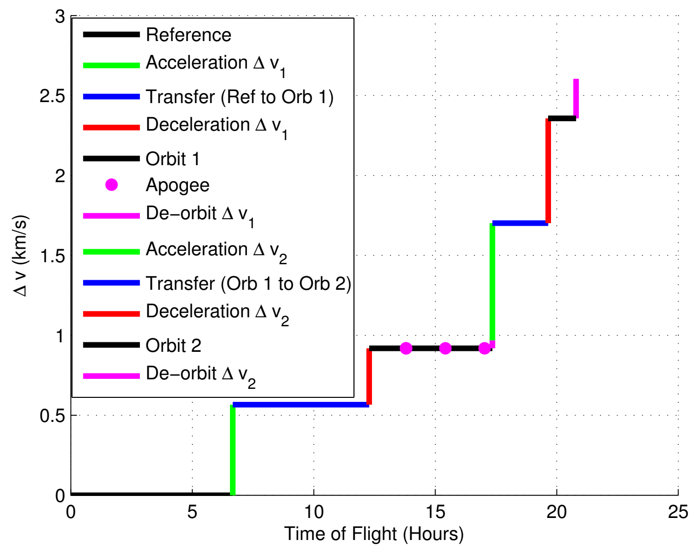


Figure 9.5: The total Δv budget as a function of time for the transfer sequence from Reference to Orbit 1 to Orbit 2 [62].

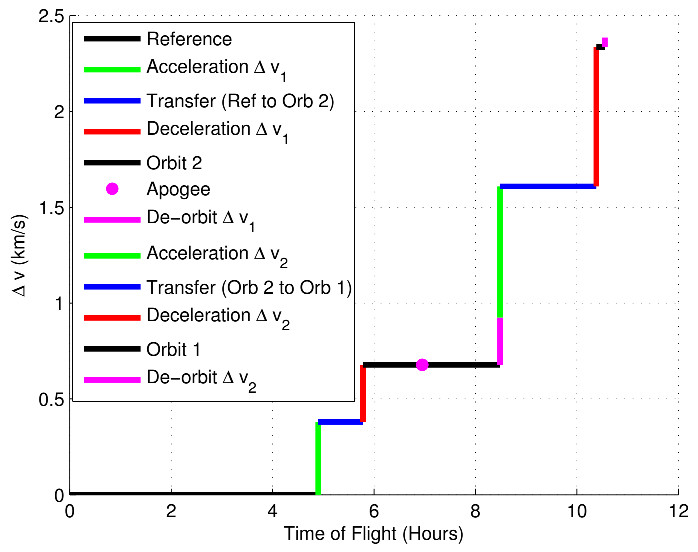


Figure 9.6: The total Δv budget as a function of time for the transfer sequence from Reference to Orbit 2 to Orbit 1. [62].

the optimal Δv sequence is to remain in the Reference orbit for about 7 hours (black line), then apply a Δv (green line) to get onto the first transfer orbit (blue line). The transfer between the Reference and Orbit 1 takes about 5 hours. Following this a Δv (red line) is applied to rendezvous with the debris in Orbit 1. The two docked spacecraft (debris removal craft and debris) remain in Orbit 1 for about 5 hours (black line). During this time the vehicle passes through apogee three times (magenta dots). All of these apogee passages are ideal times to release the debris onto a re-entry orbit. The Δv for this re-entry orbit (assuming a perigee radius of 100 km) is given by the small vertical magenta line at the end of this black line. At about 17 hours the debris removal vehicle applies a Δv (green line) to move onto the second transfer trajectory (blue line). This transfer takes about 3 hours, after which a Δv is applied to rendezvous with the second piece of debris in Orbit 2. After rendezvous the vehicle travels for a short time in Orbit 2 before reaching apogee. At this time a Δv is applied to send the second piece of debris in to a re-entry orbit, also with a perigee radius of 100 km. The total time for this sequence of maneuvers is about 21 hours, with a total Δv of about 2.6 km/s. The exact numbers are shown in Table 9.2.

A similar sequence of maneuvers is applied for the second case, the Reference to Orbit 2 to Orbit 1. Here the total time-of-flight is just short of 11 hours, with a total Δv of about 2.4 km/s. Based on this analysis, the optimal Δv sequence of maneuvers for conducting this orbit debris removal mission would be to transfer from the Reference to Orbit 2 and then to Orbit 1.

At some future time (after many revolutions), an optimal transfer region on the two-body EFM may not be optimal compared with a "perturbed" EFM. Thus, if high precision transfers are desired it is essential to include perturbations and drag in the simulation as was demonstrated in Chapter 8. However, an important point to note is that even though a trajectory may be simulated with high precision, in reality the uncertainty in the achieved Δv from the physical maneuver (i.e. the rocket engine) may be greater than the errors of the simulated solution. Thus the level of accuracy needed for each situation requires several considerations. Presently, these mission-specific considerations will dictate whether or not perturbations and drag should be included when generating the EFM.

For this simple problem with only two pieces of debris considered, we needed to generate 4 EFMs, for an additional piece of debris 15 EFMs would be required to seek an optimal solution - the resulting "orbit transfer traveling salesman problem". The problem would become even more complicated if, instead of selecting the minimum Δv in each EFM, we took into account the fact that the sum of the minimum Δv's may not be the absolute optimal way to make the transfer. That is, a moderately low Δv for the first maneuver could lead to a "super low" Δv for the second maneuver. The total may be less than the procedure adopted for the above example. In addition, if there is a finite time constraint and a fuel constraint then a different outcome could also be observed. However, independent of the degree or number of constraints, in all cases it is anticipated that the generation of EFMs is essential to give "global visibility" of the family of feasible transfers and therefore, a crucial tool to find the optimal (or desirable sup-optimal) maneuver sequence.

10. CONCLUSION

The author developed a suite of algorithms for solving the perturbed Lambert problem in celestial mechanics. These algorithms were implemented as a parallel computation tool that has broad applicability. This tool is composed of four sub-algorithms and each provides unique benefits for solving a particular type of orbit transfer problem. The first algorithm utilizes a Keplerian solver (a-iteration or p-iteration) for solving the unperturbed Lambert's problem. This algorithm not only provides a "warm start" for solving the perturbed problem but also helps to identify which of the several perturbed solvers is best suited for the job.

The second algorithm solves the two-point boundary value problem using a variant of the modified Chebyshev-Picard iteration approach to solve for two-impulse Lambert transfers. This method converges over about one third of an orbit and does not require a Newton-type shooting method; no state transition matrix needs to be computed.

The third algorithm makes use of regularization of the differential equations through the Kustaanheimo-Stiefel transformation and extends the domain of convergence over which the modified Chebyshev-Picard iteration two-point boundary value problem will converge, from about one third of an orbit to almost a full orbit. This algorithm also does not require a Newton-type shooting method. The fourth algorithm uses the method of particular solution and the modifed Chebyshev-Picard iteration initial value solver to solve the perturbed two-impulse Lambert problem over multiple revolutions. The method of particular solutions is a shooting method but differs from the Newton-type shooting methods in that it does not require integration of the state transition matrix.

The mathematical developments that underlie these four sub-algorithms were derived in the chapters of this dissertation. For each of the algorithms, some orbit transfer test cases are included to provide insight on accuracy and efficiency of these individual algorithms. Following this discussion, the combined parallel algorithm, known as the unified Lambert tool, was presented and an explanation was given as to how it automatically selects which of the three sub-algorithms to use for computing the perturbed solution for a particular orbit transfer. The unified Lambert tool may be used to determine a single orbit transfer or for the generation of extremal field maps. A case study was presented for a mission that was required to rendezvous with two pieces of orbit debris (spent rocket boosters).

The results of this dissertation can be used for mission planning, orbit transfer, and space situational awareness applications. The extremal field maps permit graphical and numerical "what if" questions to be quickly posed and answered. For future research, it is recommended that the sub-optimal continuous thrust orbit transfer techniques be used to initiate an iteration of the two-point boundary value problems associated with the indirect, calculus of variations approach to trajectory optimization. Specifically the minimum time optimal continuous thrust transfer problem. Since the take-off time is free, there are an infinite family of these, and given a range of feasible take-off times, we could find the take-off time that results in minimum time of flight (which is also minimum fuel). We could also vary the thrust level and generate an extremal field map showing take-off times, arrival times and thrust levels.

The unified Lambert tool software developed in this dissertation is already being utilized by several industrial partners and we are confident that it will play a significant role in practical applications, including solution of Lambert problems that arise in the current applications focused on enhanced space situational awareness.

REFERENCES

[1] Battin, R., *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA Education Series, 1999.

[2] Ailor, W., "Collision Avoidance and Improving Space Surveillance," *Astropolis*, Vol. 2, 2004, pp. 107–120.

[3] Anselmo, L., Cordelli, R., and Rossi, A., "New Results of the upgraded SDM space debris modeling software," *Space Debris (Science and Technology Series) American Astronautical Society*, 1999.

[4] Gaposchkin, E., von Braun, C., and Sharma, J., "Space-based surveillance with the Space-Based Visible," *Journal of Guidance, Control and Dynamics*, Vol. 23, 2000, pp. 148–152.

[5] Johnson, N., Krisko, P., Liou, J., and Anz-Meador, P., "NASA's new breakup model of evolve 4.0," *Advances in Space Research*, Vol. 28, 2001, pp. 1377–1384.

[6] Kessler, D. and Stansbery, E., "A Computer-Based Orbital Debris Environment Model for Spacecraft Design and Observation in Low Earth Orbit," *Tech. Rep. NASA-TM-104825, NASA-Johnson Space Center*, 1996.

[7] Kessler, D. and Anz-Meador, P., "Critical number of spacecraft in low Earth orbit: using satellite fragmentation data to evaluate the stability of he orbital debris environment," *Space Debris ESA Special Publication*, Vol. 473, 2001.

[8] Liou, J., Matney, M., Anz-Meador, P., Kessler, D., Jansen, M., and Theall, J., "The New NASA Orbital Debris Engineering Model OR-DEM2000," *NASA STI/Recon Technical Report N*, Vol. 2, 2002.

[9] Liou, J., Hall, D., Krisko, P., and Opiela, J., "LEGEND - a three-dimensional LEO-to-GEO debris evolutionary model," *Advances in Space Research*, Vol. 34,

2004, pp. 981–986.

[10] Liou, J., “Collision activities in the future orbital debris environment,” *Advances in Space Research*, Vol. 38, 2006, pp. 2102–2106.

[11] Naka, F., Canvana, G., Clinton, G., Judd, R., and Pensa, A., “Space Surveillance, Asteroids and Comets, and Space Debris. Volume 1: Space Surveillance,” *Tech. Rep. SAB-TR-96-04, USAF Advisory Board - DTIC Document*, Vol. 19, 1997.

[12] Oswald, M., Wegener, P., Stabroth, S., Wiedemann, C., Rosebrock, J., Martin, C., Klinkrad, H., and Vorsmann, P., “The Master 2005 Model,” *Fourth European Conference on Space Debris*, Vol. 587, 2005, pp. 235.

[13] Sdunnus, H., Bendisch, J., and Klinkrad, H., “The ESA MASTER’99 Space Debris and Meteoroid Reference Model,” *Space Debris*, Vol. 473, 2001, pp. 299–307.

[14] Sharma, J., Stokes, G., von Braun, C., Zollinger, G., and Wisemann, A., “Toward operational space-based space surveillance,” *Lincoln Laboratory Journal*, Vol. 13, 2002, pp. 309–334.

[15] Wright, D., “Space Debris,” *Physics Today*, Vol. 60, 2007, pp. 35–40.

[16] Prussing, J. and Ochoa, S., “Multiple Revolution Solutions to Lambert’s Problem,” *AAS/AIAA Spaceflight Mechanics Meeting*, 1992.

[17] Bai, X. and Junkins, J., “Modified Chebyshev-Picard Iteration Methods for Solution of Boundary Value Problems,” *Advances in the Astronautical Sciences*, Vol. 140, 2011, pp. 381–400.

[18] Bai, X., “Modified Chebyshev-Picard Iteration for Solution of Initial Value and Boundary Value Problems,” *PhD. Dissertation, Texas A&M, College Station, Texas, USA*, 2010.

[19] Woollands, R., Bani Younes, A., and Junkins, J., “New Solutions for the Per-

turbed Lambert Problem Using Regularization and Picard Iteration," *Journal of Guidance, Control and Dynamics*, Vol. 38, 2015, pp. 1548–1562.

[20] Miele, A. and Iyer, R., "General Technique for Solving Nonlinear, Two-Point Boundary value Problems Via the Method of Particular Solutions," *Journal of Optimization Theory and Applications*, Vol. 5, No. 5, 1970, pp. 392–399.

[21] Woollands, R., Read, J., Macomber, B., Probe, A. Bani Younes, A., and Junkins, J., "Method of Particular Solutions and Kustaanheomi-Stiefel Regularized Picard Iteration for Solving Two-Point Boundary Value Problems," *AAS/AIAA Space Flight Mechanics Meeting, Williamsburg, Virginia*, 2015.

[22] Bani Younes, A., "Orthogonal Polynomial Approximation in Higher Dimensions: Applications in Astrodynamics," *PhD. Dissertation, Texas A&M, College Station, Texas, USA*, 2013.

[23] Chebyshev, P. L., "Thèorie des mècanismes connus sous le nom de parallèlogrammes," *Mèmoires des Savants ètrangers prèsentès liAcadèmie de Saint-Pètersbourg*, Vol. 7, 1857, pp. 539–586.

[24] Fox, L. and Parker, I. B., *Chebyshev Polynomials in Numerical Analysis*, London, UK: Oxford University Press, 1972.

[25] Boyd, J., *Chebyshev and Fourier Spectral Methods*, Dover Publications Inc., 2001.

[26] Picard, E., "Sur l'application des methodes d approximations successives b latude de certaines bquatioiis differentielles or- dinaires," *J. de Math*, Vol. 9, 1983, pp. 217–271.

[27] Picard, E., *Trait d'analyse*, Gauthier-Villars, Paris, France, 3rd ed., 1922.

[28] Craats, J., "On the region of convergence of Picard's iteration," *Journal of Applied Mathematics and Mechanics*, Vol. 52, 1971, pp. 487–491.

[29] Lettenmeyer, F., "Ober di von einem punkt ausgehenden itegralkurven einer

differentialgleichung 2. ordnung," *Deutsche Math*, Vol. 7, 1944, pp. 56–74.

[30] Agarwal, R., "Nonlinear two-point boundary value problems," *Indian Journal of Pure and Applied Mathematics*, Vol. 4, 1973, pp. 757–769.

[31] Coles, W. and Sherman, T., "Convergence of successive approximations for nonlinear two-point boundary value problems," *SIAM Journal of Applied Mathematics*, Vol. 15, 1967, pp. 426–433.

[32] Clenshaw, C. W., "The numerical solution of linear differential equations in Chebyshev series," *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 53, 1957, pp. 134–149.

[33] Scraton, R., "The solution of linear differential equations in Chebyshev series," *The Computer Journal*, Vol. 8, 1965, pp. 57–61.

[34] Wright, K., "Chebyshev collocation methods for ordinary differential equations in Chebyshev series," *The Computer Journal*, Vol. 7, 1964, pp. 358–365.

[35] Norton, H., "The iterative solution of non-linear ordinary differential equations in Chebyshev series," *The Computer Journal*, Vol. 7, 1964.

[36] Urabe, M., "Galerkin's procedure for nonlinear periodic systems," *Archive for Rational Mechanics and Analysis*, Vol. 20, 1965, pp. 120–152.

[37] Urabe, M. and Reiter, A., "Numerical computation of nonlinear forced oscillations by Galerkin's procedure," *Journal of Mathematical Analysis and Application*, Vol. 14, 1966, pp. 107–140.

[38] Chen, B., Carca-Bolsb, R., Jdarb, L., and Rosellb, M., "Chebyshev polynomial approximations for nonlinear differential initial value problems," *Nonlinear Analysis*, Vol. 63, 2005, pp. 629–637.

[39] Clenshaw, C. W. and Norton, H. J., "The Solution of Nonlinear Ordinary Differential Equations in Chebyshev Series," *The Computer Journal*, Vol. 6, 1963, pp. 88–92.

[40] Macomber, B., "Enhancements of Chebyshev-Picard Iteration Efficiency for Generally Perturbed Orbits and Constrained Dynamics Systems," *PhD. Dissertation, Texas A&M University, College Station, Texas, USA*, 2015.

[41] Macomber, B., Probe, A., Woollands, R., Read, J., and Junkins, J., "Enhancements of Modified Chebyshev-Picard Iteration Efficiency for Perturbed Orbit Propagation," *Computational Modelling in Engineering & Sciences*, Vol. 111, 2016, pp. 29–64.

[42] Probe, A., Macomber, B., Kim, D., Woollands, R., and Junkins, J., "Terminal Convergence Approximation Modified Chebyshev Picard Iteration for Efficient Numerical Integration of Orbital Trajectories," *Advanced Maui Optical Space Surveillance Technologies Conference, Maui, Hawaii*, 2014.

[43] Feagin, T. and Nacozy, P., "Matrix Formulation of the Picard Method for Parallel Computation," *Celestial Mechanics and Dynamical Astronomy*, Vol. 29, 1983, pp. 107–115.

[44] Shaver, J., "Formulation and Evaluation of Parallel Algorithms for the Orbit Determination Problem," *Ph.D. Dissertation, Department of Aeronautics and Astronautics, MIT, Cambridge, MA*, 1980.

[45] Fukushima, T., "Vector Integration of Dynamical Motions by the Picard-Chebyshev Method," *The Astronomical Journal*, Vol. 113, 1997, pp. 2325–2328.

[46] Bai, X. and Junkins, J., "Modified Chebyshev-Picard Iteration Methods for Solution of Initial Value Problems," *Advances in the Astronautical Sciences*, Vol. 139, 2011, pp. 345–362.

[47] Woollands, R., Bani Younes, A., Macomber, B., Bai, X., and Junkins, J., "Optimal Continuous Thrust Maneuvers for Solving 3D Orbit Transfer Problems," *38th Annual AAS Guidance & Control Conference, Breckenridge, Colorado*, 2015.

[48] Junkins, J., Bani Younes, A., Woollands, R., and Bai, X., "Orthogonal Approximation in Higher Dimensions: Applications in Astrodynamics," *AAS 12-634, JN Juang Astrodynamics Symp*, 2012.

[49] Junkins, J., Bani Younes, A., Woollands, R., and Bai, X., "Picard Iteration, Chebyshev Polynomial and Chebyshev Picard Methods: Application in Astrodynamics," *Journal of the Astronautical Sciences*, 2013.

[50] Junkins, J., Bani Younes, A., Woollands, R., and Bai, X., "Efficient and Adaptive Orthogonal Finite Element Representation of the Geopotential," *Journal of the Astronautical Sciences*, accepted 2016.

[51] Probe, A., Macomber, B., Read, J., Woollands, R., and Junkins, J., "Radially Adaptive Evaltuation of the Spherical Harmonic Gravity Series for Numerical Orbital Propagation," *AAS/AIAA Space Flight Mechanics Meeting, Williamsburg Virginia*, 2015.

[52] Macomber, B., Probe, A., Woollands, R., and Junkins, J., "Automated Tuning Parameter Selection for Orbit Propagation with Modified Chebyshev Picard Iteration," *AAS/AIAA Space Flight Mechanics Meeting, Williamsburg Virginia*, 2015.

[53] Woollands, R., Bani Younes, A., Macomber, B., Probe, A., Kim, D., and Junkins, J., "Validation of Accuracy and Efficiency of Long-Arc Orbit Propagation Using the Method of Manufactured Solutions and the Round-Trip-Closure Method," *Advanced Maui Optical Space Surveillance Technologies Conference, Maui, Hawaii*, 2014.

[54] Woollands, R., Bani Younes, A., and Junkins, J., "A New Solution for the General Lambert Problem," *37th Annual AAS Guidance & Control Conference*, 2014.

[55] Koblick, D. and Shankar, P., "Evaluation of the Modified Picard-Chebyshev

Method for High-Precision Orbit Propagation," *Journal of Aerospace Engineering*, 2014.

[56] Engels, R. and Junkins, J., "The Gravity-Perturbed lambert Problem: A KS Variation of Parameters Approach," *Celestial Mechanics*, 1981.

[57] Kriz, J., "A Uniform Solution of the Lambert Problem," *Celestial Mechanics*, Vol. 14, 1976, pp. 509–513.

[58] Kustaanheimo, P. and Stiefel, E., "Perturbation theory of Kepler motion based on spinor regularzation," *Journal fur die Reine und Angewandt Mathematik*, Vol. 218, 1965, pp. 204–219.

[59] Stiefel, E. and Scheifele, G., *Linear and Regular Celestial Mechanics*, Springer-Verlag Berlin Heidelberg New York, 1971.

[60] Levi-Civita, T., "Sur la regularization du problem des trois corps," *Acta Mathematica*, Vol. 42, 1920.

[61] Schaub, H. and Junkins, J., *Analytical Mechanics of Aerospace Systems*, AIAA Education Series, 3rd ed., 2014.

[62] Woollands, R., Read, J., Probe, A., and Junkins, J., "Multiple Revolution Solutions for the Perturbed Lambert Problem using the Method of Particular Solutions and Picard Iteration," *Journal of Astronautical Sciences*, 2016.

[63] Feagin, T., "High-Order m-symmetric Runge-Kutta Methods," *Proceeding of the 23rd Biennial Conference on Numerical Analysis*, 2009.

[64] Feagin, T., "High-Order m-symmetric Runge-Kutta Methods," *Neural, Parallel and Scientific Computations*, Vol. 20, 2012, pp. 437–458.

[65] Prussing, J. and Conway, B., *Orbital Mechanics*, Oxford University Press, 1993.

[66] Read, J., Bani Younes, A., and Junkins, J., "Efficient Orbit Propagation of Orbital Elements Using Modified Chebyshev-Picard Iteration Method," *Computational Modelling in Engineering & Sciences*, Vol. 111, 2016, pp. 65–82.

[67] “Orbital Elements,” $https://en.wikipedia.org/wiki/Orbital_elements$, 2016.

[68] “Chebyshev Polynomials,” $https://en.wikipedia.org/wiki/Chebyshev_polynomials$, 2016.

[69] Crassidis, J. L. and Junkins, J. L., *Optimal Estimation of Dynamic Systems, Second Edition.*, New York, NY: CRC Press - Taylor and Francis. ISBN 978-1-4398-3985-0., 2011.

[70] Ince, E. L., *Ordinary Differential Equations*, New York, NY: Dover Publications, 1956.

APPENDIX A

CHEBYSHEV POLYNOMIALS

In 1857 the Russian mathematician, Rafnuty Lvovich Chebyshev, developed a series of orthogonal polynomials that are now referred to as Chebyshev polynomials [23]. There are two types of Chebyshev polynomials and these are distinguished as follows: Chebyshev polynomials of the first kind denoted by (T_k), and Chebyshev polynomials of the second kind denoted by (U_k). For simplicity, throughout this dissertation the term Chebyshev polynomials is used to refer to only the first kind of Chebyshev polynomials.

In order to define Chebyshev polynomials and derive the recurrence relation (a similar procedure was outlined in Bai's dissertation [18]) we start with the following identity:

$$\cos((n+1)\theta) = 2\cos(\theta)\cos(n\theta) - \cos((n-1)\theta). \tag{A.1}$$

This identity is a rearrangement of Eq. (B.21) and can be proven using the "Sums-to-Products" identity as follows.

$$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta \tag{A.2}$$

Substitute $\alpha = n\theta$ and $\beta = \theta$ in first to form one version of the equation, and then substitute $\alpha = n\theta$ and $\beta = -\theta$ to form a second version. Add the two together as follows:

$$\cos(n\theta+\theta) = \cos(n\theta)\cos\theta - \sin(n\theta)\sin\theta \tag{A.3}$$

and

$$\cos(n\theta - \theta) = \cos(n\theta)\cos(-\theta) - \sin(n\theta)\sin(-\theta). \tag{A.4}$$

Adding these results in

$$\cos(n\theta + \theta) + \cos(n\theta - \theta) = 2\cos(n\theta)\cos\theta, \tag{A.5}$$

and simple rearrangement leads to Eq. (A.1) as desired. The next step is to write $\cos(n\theta)$ as a summed function of itself. This is shown below, along with some verification to prove the identity:

$$\cos(n\theta) = \sum_{i=0}^{n} c_i \cos^i(\theta) \tag{A.6}$$

The validity of the preceding formula is tested using $n = 0, 1, 2, 3$. The c_i are to be determined functions $c_i(\theta)$. For $n = 0$,

$$1 = c_0. \tag{A.7}$$

For $n = 1$,

$$\cos\theta = 1 + c_1 \cos\theta \Rightarrow c_1 = \frac{\cos(\theta) - 1}{\cos(\theta)}. \tag{A.8}$$

For $n = 2$,

$$\cos(2\theta) = 1 + (\cos(\theta) - 1) + c_2\cos^2(\theta) \Longrightarrow c_2 = \frac{\cos(2\theta) - \cos(\theta)}{\cos^2(\theta)}. \tag{A.9}$$

For $n = 3$,

$$\cos(3\theta) = 1+(\cos(\theta)-1)+(\cos(2\theta)-\cos(\theta))+c_3\cos(3\theta) \Longrightarrow c_3 = \frac{\cos(3\theta)-\cos(2\theta)}{\cos^3(\theta)}. \tag{A.10}$$

Substituting the coefficients c_0, c_1, c_2, c_3 in to the the equation for $n = 3$ leads to $\cos(3\theta) = \cos(3\theta)$, thus proving by induction that Eq. (A.6) holds true. Eq. (A.6) and the first three c_i also suggests that $\cos(n\theta)$ is a polynomial in $\cos(\theta)$, and thus for a fixed n the the n^{th} Chebyshev polynomial is defined as

$$\cos(n\theta) = T_n \cos\theta. \tag{A.11}$$

If $\tau = \cos\theta$ we obtain the following:

$$T_n(\tau) = \cos(n \arccos(\tau)), \quad \text{for} \ \ \tau \ \ \text{in} \ \ [-1, 1] \tag{A.12}$$

Chebyshev polynomials are actually cosine curves with a somewhat disturbed horizontal scale, but the vertical scale has been untouched. Eq. (A.12) can be extended to Eq. (A.13) when necessary:

$$T_m(T_n(\tau)) = T_{nm}(\tau), \tag{A.13}$$

since

$$\cos(m \arccos(\cos(n \arccos(\tau)))) = \cos(mn \arccos(\tau)). \tag{A.14}$$

Referring to Eq. (A.1) is is clear that the recurrence relation for cosines leads to the recurrence relation for Chebyshev polynomials as shown below.

$$T_0(\tau) = 1 \tag{A.15}$$

$$T_1(\tau) = \tau \tag{A.16}$$

$$T_{k+1}(\tau) = 2\tau T_k(\tau) - T_{k-1}(\tau). \tag{A.17}$$

The continuous orthogonality of Chebyshev polynomials satisfies

$$\int_{-1}^{1} T_n(\tau)T_m(\tau)\frac{1}{\sqrt{1-\tau^2}} = \begin{cases} 0 : n \neq m \\ \pi : n = m = 0 \\ \frac{\pi}{2} : n = m \neq 0. \end{cases} \tag{A.18}$$

The discrete orthogonality of the Chebyshev polynomials using the GCL nodes is given as

$$\sum_{k=0}^{k=N}{''} T_n(\tau_k)T_m(\tau_k) = \begin{cases} 0 : n \neq m \\ N : n = m = 0 \\ \frac{N}{2} : n = m \neq 0, \end{cases} \tag{A.19}$$

where $''$ conveys that both the first and last terms in the summation are multiplied by a half, similar to the weight matrix (W) notation discussed in Chapter 2. The number of CGL nodes ($N+1$) for the Nth order Chebyshev polynomials are computed using

$$\tau_k = cos\left(\frac{k\pi}{N}\right). \tag{A.20}$$

The integration of the Chebyshev polynomials have the following property:

$$\int T_k(\tau)ds = \frac{1}{2}\left(\frac{T_{k+1}}{k+1} - \frac{T_{k-1}}{k-1}\right). \tag{A.21}$$

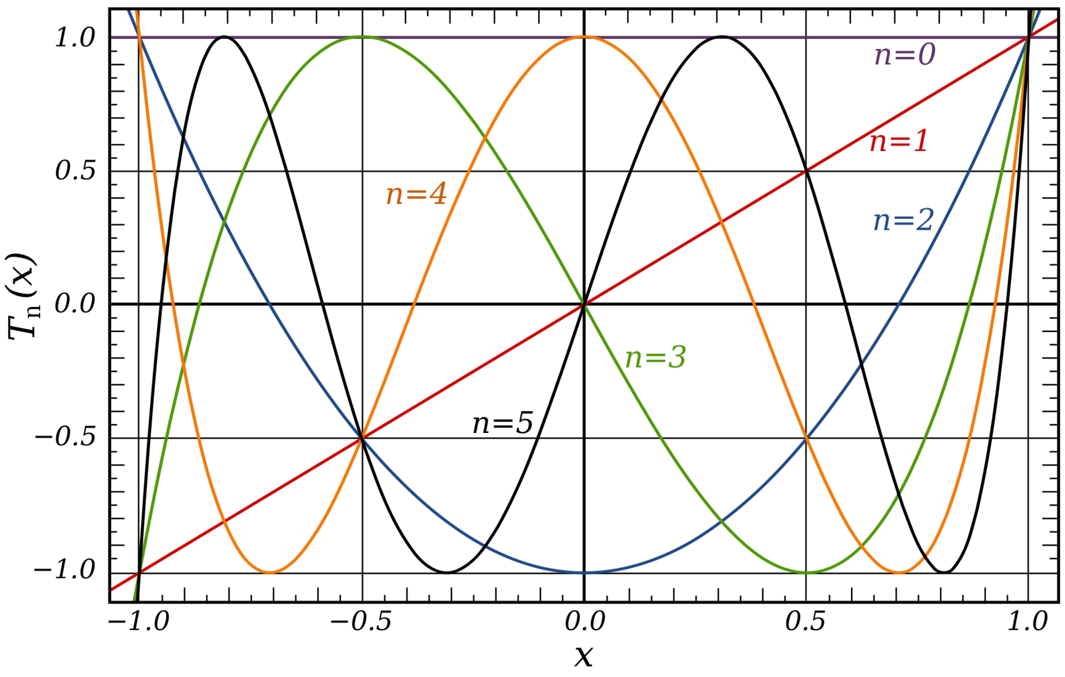


Figure A.1: The first six Chebyshev Polynomials of the first kind [68].

Chebyshev polynomials of the first kind are related to Chebyshev polynomials of the second kind through

$$\frac{dT_k}{d\tau} = kU_{k-1}. \tag{A.22}$$

It is important to note that another set of polynomials also exist that are the zeros of the Chebyshev polynomials. These are give defined by

$$\tau_k = cos\left(\frac{(2k+1)\pi}{2N}\right), k = 0, 1, ...N. \tag{A.23}$$

Fox [24] performed a comparison of these two formula and concluded that first have theoretical advantages for cases of slow convergence. They are also economic in that when the number of points is doubled, the old matching points may be reused. Figure A.1, displays the first six Chebyshev polynomials.

APPENDIX B

ORTHOGONAL APPROXIMATION

The material in this appendix supplements that presented in Chapter 2. MCPI uses *least squares approximation* and for completeness we derived this in Section B.1. This is followed by a derivation of the particular choice of weight matrix that preserves orthogonality and allows an expensive matrix inversion to be avoided (Section B.2). Finally this appendix includes examples for computing the coefficients of the least squares approximation (Section B.2.3).

B.1 Least Squares Derivation

In vector-matrix notation Eq. 2.6 from Section 2.3 becomes the linear system

$$\boldsymbol{r} = \boldsymbol{f} - \Phi \boldsymbol{a}, \tag{B.1}$$

where

$$\boldsymbol{f} = \begin{bmatrix} f(\xi_0) \\ f(\xi_1) \\ \vdots \\ f(\xi_M) \end{bmatrix}, \ \Phi = \begin{bmatrix} \phi_0(\xi_0) & \phi_1(\xi_0) & \cdots & \phi_N(\xi_0) \\ \phi_0(\xi_1) & \phi_1(\xi_1) & \cdots & \phi_N(\xi_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\xi_M) & \phi_1(\xi_M) & \cdots & \phi_N(\xi_M) \end{bmatrix}, \ \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}. \tag{B.2}$$

Eq. B.1 can be rearranged into the more familiar notation $\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} = 0$ (i.e. $A\boldsymbol{x} = \boldsymbol{b}$), where $\boldsymbol{f}$ is the $\boldsymbol{b}$ vector, A is the Φ matrix, and $\boldsymbol{x}$ is the $\boldsymbol{a}$ vector of coefficients that we wish to solve for. Depending on the dimensions of Φ we can either solve to obtain an exact solution, or find the "best/closest" solution using the *least squares* or *minimum norm* methods.

If M and N are equal (i.e. # of unknowns = # of equations) then an exact solution can be found. If $M < N$ (i.e. # of unknowns < # of equations) then the *minimum norm* method must be used, and if $M > N$ (i.e. # of unknowns > # of equations) then the *least squares* method must be employed. Since both M, the number of sample points, and N, the degree of the approximating polynomial, are selected/determined manually we anticipate that solving using *minimum norm* methods can be avoided. Therefore only the *least squares* method is considered and derived in this appendix.

Starting with

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{B.3}$$

in the usual least squares fashion, the error that we wish to minimize is given by

$$\boldsymbol{e} = \boldsymbol{b} - A\boldsymbol{x}. \tag{B.4}$$

Traditionally the least squares cost function, J given in Eq. B.5, must be minimized. The cost function may be thought of as the "quality or goodness" of the fit.

$$J = \min \frac{\boldsymbol{e}^T \boldsymbol{e}}{2}, \tag{B.5}$$

$$J = \frac{1}{2}\boldsymbol{e}^T \boldsymbol{e} = \frac{1}{2}\left(\boldsymbol{b} - A\boldsymbol{x}\right)^T \left(\boldsymbol{b} - A\boldsymbol{x}\right). \tag{B.6}$$

Distributing the transpose produces

$$J = \frac{1}{2}\left(\boldsymbol{b}^T - \boldsymbol{x}^T A^T\right)\left(\boldsymbol{b} - A\boldsymbol{x}\right). \tag{B.7}$$

Expanding the above equation leads to the following:

$$J = \frac{1}{2}\boldsymbol{b}^T\boldsymbol{b} - \frac{1}{2}\boldsymbol{b}^T A\boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^T A^T\boldsymbol{b} + \frac{1}{2}\boldsymbol{x}^T A^T A\boldsymbol{x}. \tag{B.8}$$

Simplifying results in

$$J = \frac{1}{2}\boldsymbol{b}^T\boldsymbol{b} - \boldsymbol{b}^T A\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T A^T A\boldsymbol{x}. \tag{B.9}$$

To minimize J we differentiate and set it equal to zero.

$$\frac{\partial J}{\partial x} = 0 - A^T\boldsymbol{b} + A^T A\boldsymbol{x} = 0 \tag{B.10}$$

Thus,

$$A^T A\boldsymbol{x} = A^T\boldsymbol{b} \tag{B.11}$$

and we have the usual "normal equations" for the least squares approximation [61]

$$\boldsymbol{x} = \left(A^T A\right)^{-1} A^T\boldsymbol{b}. \tag{B.12}$$

Eq. B.12 shows that the vector of coefficients of the approximating polynomial ($\boldsymbol{x}$) may be determined in terms of the known quantities $\boldsymbol{b}$ and A. If desired, a positive definite weight matrix may also be included in the cost function. This would lead to Eq. B.6 becoming $J = \frac{1}{2}\boldsymbol{e}^T W\boldsymbol{e} = \frac{1}{2}\left(\boldsymbol{b} - A\boldsymbol{x}\right)^T W\left(\boldsymbol{b} - A\boldsymbol{x}\right)$, and finally the normal equations Eq. (B.12) are generalized to $\boldsymbol{x} = \left(A^T W A\right)^{-1} A^T W\boldsymbol{b}$.

For MCPI the cost function is specifically

$$J = \frac{1}{2}\left(\boldsymbol{f} - \Phi\boldsymbol{a}\right)^T W\left(\boldsymbol{f} - \Phi\boldsymbol{a}\right), \tag{B.13}$$

where $W = W^T$ is a positive definite weight matrix, and the least squares minimization solution for $\boldsymbol{a}$ leads to the normal equations in Eq. 2.9. For more details refer to Crassidis & Junkins [69].

$$\boldsymbol{a} = \left(\Phi^T W \Phi\right)^{-1} \Phi^T W \boldsymbol{f}. \tag{B.14}$$

B.2 Weight Matrix

If W is restricted to be a diagonal matrix, and a special class of orthogonal functions (Chebyshev polynomials) is chosen such that the matrix of $\Phi^T W \Phi$ is also diagonal orthogonal, then the inverse of the diagonal matrix $\Phi^T W \Phi$ is trivial. That is, the inverse of a diagonal matrix is simply the matrix with the reciprocal of each element on the diagonal. This is given in Eq B.15.

$$\left(\Phi^T W \Phi\right)^{-1} = diag\left\{1/\left(\Phi^T W \Phi\right)_{ii}\right\} \equiv diag\left\{\begin{array}{cccc} 1/m_{00} & 1/m_{11} & \cdots & 1/m_{NN} \end{array}\right\} \tag{B.15}$$

B.2.1 Elements of $\Phi^T W \Phi$

The typical element of $\Phi^T W \Phi$ is a *discrete inner product* denoted $m_{\alpha\beta} = m_{\beta\alpha}$, and invoking the condition that $\Phi^T W \Phi$ be a diagonal matrix directly gives rise to the *orthogonality conditions*, requiring the typical pair of orthogonal basis functions' inner products obey:

$$m_{\alpha\beta} = m_{\beta\alpha} \equiv \langle \phi_\alpha(\xi), \phi_\beta(\xi) \rangle \equiv \sum_{j=0}^{M} W_j \phi_\alpha(\xi_j)\phi_\beta(\xi_j) = \left\{\begin{array}{c} 0,\ for\ \alpha \neq \beta \\ m_{\alpha\alpha} = c_\alpha > 0,\ for\ \alpha = \beta \end{array}\right\}, \tag{B.16}$$

where $m_{\alpha\beta}$ is the typical term in $\Phi^T W \Phi$. More specifically (for $M = 2$),

$$m_{\alpha\beta} = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} = \left(\Phi^T W \Phi\right), \tag{B.17}$$

where $\left(\Phi^T W \Phi\right)$ is given by,

$$\left(\Phi^T W \Phi\right) = \begin{bmatrix} \phi_0(\xi_0) & \phi_0(\xi_1) & \phi_0(\xi_2) \\ \phi_1(\xi_0) & \phi_1(\xi_1) & \phi_1(\xi_2) \\ \phi_2(\xi_0) & \phi_2(\xi_1) & \phi_2(\xi_2) \end{bmatrix} \begin{bmatrix} W_0 & 0 & 0 \\ 0 & W_1 & 0 \\ 0 & 0 & W_2 \end{bmatrix} \begin{bmatrix} \phi_0(\xi_0) & \phi_1(\xi_0) & \phi_2(\xi_0) \\ \phi_0(\xi_1) & \phi_1(\xi_1) & \phi_2(\xi_1) \\ \phi_0(\xi_2) & \phi_1(\xi_2) & \phi_2(\xi_2) \end{bmatrix}. \tag{B.18}$$

The matrix multiplication computation leads to the following:

$$\begin{aligned} m_{00} &= W_0\phi_0^2(\xi_0) + W_1\phi_0^2(\xi_1) + W_2\phi_0^2(\xi_2), \\ m_{11} &= W_0\phi_1^2(\xi_0) + W_1\phi_1^2(\xi_1) + W_2\phi_1^2(\xi_2), \\ m_{22} &= W_0\phi_2^2(\xi_0) + W_1\phi_2^2(\xi_1) + W_2\phi_2^2(\xi_2), \\ m_{01} = m_{10} &= W_0\phi_0(\xi_0)\phi_1(\xi_0) + W_0\phi_0(\xi_0)\phi_1(\xi_0) + W_2\phi_0(\xi_2)\phi_1(\xi_2), \\ m_{02} = m_{20} &= W_0\phi_0(\xi_0)\phi_2(\xi_0) + W_1\phi_0(\xi_1)\phi_2(\xi_1) + W_2\phi_0(\xi_2)\phi_2(\xi_2), \\ m_{12} = m_{21} &= W_0\phi_1(\xi_0)\phi_2(\xi_0) + W_1\phi_1(\xi_1)\phi_2(\xi_1) + W_2\phi_1(\xi_2)\phi_2(\xi_2). \end{aligned} \tag{B.19}$$

If $\{\phi_0(\xi), \phi_1(\xi), \phi_2(\xi)\}$ are an orthogonal set, then $m_{01} = m_{10} = 0$, $m_{02} = m_{20} = 0$, and $m_{12} = m_{21} = 0$, and all that is left are the diagonal terms of the m matrix.

The orthogonality conditions depend jointly on the set of basis functions, the set of node locations and the weight matrix ($W = W^T$). Consistent with the classical orthogonality conditions for Chebyshev polynomials, the weight matrix is defined as follows $W = diag\left\{\frac{1}{2}, 1, 1, ..., 1, 1, \frac{1}{2}\right\}$.

B.2.2 Why Half?

The choice of a half for the first and last elements in the weight matrix is discussed in [24], pages 25 through 32. At the fundamental level, once the basis functions and nodes are chosen, the weights are "whatever that have to be" to ensure the orthogonality conditions are satisfied. Their discussion and derivation is summarized in this section. Note that the interior nodes are at the extrema of the N Chebyshev basis functions, where the boundary nodes are not extrema.

The orthogonal Chebyshev polynomials may be calculated using Eq. A.15 to Eq. A.17 or using the cosine trigonometric identity given in Appendix A, also shown below, where $k = 0, ..., N$.

$$\phi_k(\xi_j) = \cos(k \ \arccos(\xi_j)),\ \xi \in [-1, 1], \tag{B.20}$$

As part of the least squares analysis, that was discussed in previous section, the sum in Eq. B.16 must be computed. This is the product of two basis functions using the form shown in Eq. B.20, $\phi_\alpha(\xi) = \cos(\alpha x)$ and $\phi_\beta(\xi) = \cos(\beta x)$, where $x_j = arccos(\xi_j)$. Since $\xi_j = -cos\left(\frac{i\pi}{M}\right)$, $x = \arccos(\xi_j) = \arccos\left(-cos\left(\frac{i\pi}{M}\right)\right) = \frac{(M\pi - j\pi)}{M}$. Making use of the "products-to-sums" cosine trigonometric identity leads to the following expression:

$$\cos(\alpha x)\cos(\beta x) = \frac{1}{2}\left(\cos((\alpha+\beta)x) + \cos((\alpha-\beta)x)\right). \tag{B.21}$$

This implies that the product of the two basis functions within the "summation" may be written purely as a sum of cosine terms. Looking at an example of $M = 3$, and selecting α and β to be 2 and 3 respectively results in the following.

$$\sum_{j=0}^{M} \phi_\alpha(\xi_j)\phi_\beta(\xi_j) = \sum_{j=0}^{M} \frac{1}{2}\left(\cos\left(\left(\alpha+\beta\right)x_j\right) + \cos\left(\left(\alpha-\beta\right)x_j\right)\right), \tag{B.22}$$

which becomes

$$\begin{aligned} \sum_{j=0}^{3} \phi_2(\xi_j)\phi_3(\xi_j) = \frac{1}{2}\big(&\cos\left((2+3)\,x_0\right) + \cos\left((2-3)\,x_0\right) + ... \\ &... + \cos\left((2+3)\,x_1\right) + \cos\left((2-3)\,x_1\right) + ... \\ &... + \cos\left((2+3)\,x_2\right) + \cos\left((2-3)\,x_2\right) + ... \\ &+ \cos\left((2+3)\,x_3\right) + \cos\left((2-3)\,x_3\right)\big), \end{aligned} \tag{B.23}$$

More specifically,

$$\begin{aligned} \sum_{j=0}^{3} \phi_2(\xi_j)\phi_3(\xi_j) = \frac{1}{2}\Bigg(&\cos\underbrace{\left((2+3)\frac{(3\pi-0\pi)}{3}\right)}_{=0} + \cos\underbrace{\left((2-3)\frac{(3\pi-0\pi)}{3}\right)}_{=0} + ... \\ &... + \cos\underbrace{\left((2+3)\frac{(3\pi-\pi)}{3}\right)}_{=\frac{10\pi}{3}\equiv\theta_1} + \cos\underbrace{\left((2-3)\frac{(3\pi-\pi)}{3}\right)}_{=-\frac{2\pi}{3}\equiv\theta_2} + ... \\ &... + \cos\underbrace{\left((2+3)\frac{(3\pi-2\pi)}{3}\right)}_{=\frac{5\pi}{3}=2\theta_1} + \cos\underbrace{\left((2-3)\frac{(3\pi-2\pi)}{3}\right)}_{=-\frac{\pi}{3}=2\theta_2} + ... \\ &... + \cos\underbrace{\left((2+3)\frac{(3\pi-3\pi)}{3}\right)}_{=5\pi=3\theta_1} + \cos\underbrace{\left((2-3)\frac{(3\pi-3\pi)}{3}\right)}_{=-\pi=3\theta_2}\Bigg) \end{aligned} \tag{B.24}$$

The purpose of the "lower" braces and $\theta' s$ will become apparent shortly.

For orthogonality, the inner product of the two basis functions (ϕ_2 and ϕ_3) must be zero. Thus Eq. (B.24) must equal zero. In order to achieve this, now consider another trigonometric identity that is a sum of cosine terms shown in Eq. (B.21). It is clear that when $\theta = M\pi$ the **R.H.S.** becomes zero.

$$\frac{1}{2} + cos\left(\theta\right) + cos\left(2\theta\right) + ... + cos\left(\left(M-1\right)\theta\right) + \frac{1}{2}cos\left(M\theta\right) = \frac{1}{2}sin\left(M\theta\right)cot\left(\frac{1}{2}\theta\right) \tag{B.25}$$

This trigonometric identity is similar to the cosine sum that results from the inner product in Eq. B.24, if θ is set equal to $\left(\alpha+\beta\right)\frac{\pi}{3}$. For $\theta = \left(\alpha-\beta\right)\frac{\pi}{3}$ we have another form of Eq. B.21, so in fact Eq. B.24 is the sum of twice Eq. B.21, one with $\theta_1 = \left(\alpha+\beta\right)\frac{\pi}{3}$ and the other with $\theta_2 = \left(\alpha-\beta\right)\frac{\pi}{3}$. This is shown in Eq. B.26 with the θ_1 terms colored in red to aid in identifying the two series.

$$\frac{1}{2} + \frac{1}{2} + \cos\theta_1 + \cos\theta_2 + \cos 2\theta_1 + \cos 2\theta_2 + \frac{1}{2}\cos 3\theta_1 + \frac{1}{2}\cos 3\theta_2 = 0. \tag{B.26}$$

Independently comparing the red (or **black**) parts of Eq. B.26 to Eq. B.21, it is clear that the only difference is that the first and last terms of the identity (Eq. B.21) are $\frac{1}{2}$ the size of those in Eq. B.26. Thus in order to achieve orthogonality the **R.H.S.** of Eq. (B.24) must take on the form of the **L.H.S.** of Eq. (B.26). To achieve this we must apply a $\frac{1}{2}$ weight to the first and last boundary terms, while all the rest are unity (these nodes locating the extrema of the M Chebyshev sample points). It is for this reason that the we adopt the particular weight matrix of $W = diag\left\{\frac{1}{2}, 1, 1, ..., 1, 1, \frac{1}{2}\right\}$, to ensure orthogonality.

B.2.3 Coefficients

A step-by-step verification of Eq. 2.15, in Section 2.7, is given for a few select example cases.

B.2.3.1 Verification of c_0

Consider the case for c_0 and M= 4. From Eq. 2.15

$$c_0 = \langle T_0(\xi), T_0(\xi)\rangle, \tag{B.27}$$

and from Eq. 2.14

$$c_0 = \frac{1}{2}T_0^2(\xi_0) + T_0^2(\xi_1) + T_0^2(\xi_2) + T_0^2(\xi_3) + \frac{1}{2}T_0^2(\xi_4). \tag{B.28}$$

The zeroth Chebyshev polynomial is given by

$$T_0 = 1, \tag{B.29}$$

and therefore

$$c_0 = \left\{\frac{1}{2} + 1 + 1 + 1 + \frac{1}{2} = 4\right\} = M. \tag{B.30}$$

B.2.3.2 Verification of $c_\alpha = c_1$

Consider the case for $N = 1$, i.e. ($c_\alpha = c_1$), and $M = 4$. From Eq. 2.15

$$c_1 = \langle T_1(\xi), T_1(\xi)\rangle, \tag{B.31}$$

and from Eq. 2.14

$$c_1 = \frac{1}{2}T_1^2(\xi_0) + T_1^2(\xi_1) + T_1^2(\xi_2) + T_1^2(\xi_3) + \frac{1}{2}T_1^2(\xi_4). \tag{B.32}$$

The first Chebyshev polynomial is given by

$$T_1 = \xi, \tag{B.33}$$

and the cosine sample points are calculated as follows:

$$\xi_0 = -\cos(\frac{0\pi}{4}) = -1, \tag{B.34}$$

$$\xi_1 = -\cos(\frac{\pi}{4}) \cong -0.7071, \tag{B.35}$$

$$\xi_2 = -\cos(\frac{2\pi}{4}) = 0, \tag{B.36}$$

$$\xi_3 = -\cos(\frac{3\pi}{4}) \cong 0.7071, \tag{B.37}$$

$$\xi_4 = -\cos(\frac{4\pi}{4}) = 1. \tag{B.38}$$

Substituting these sample points back into Eq. B.33 gives

$$T_1^2(\xi_0) = (-1)^2, \tag{B.39}$$

and similarly,

$$T_2^2(\xi_1) = (-0.7071)^2,\ T_2^2(\xi_2) = 0,\ T_2^2(\xi_3) = (0.7071)^2,\ T_2^2(\xi_3) = 1. \tag{B.40}$$

Therefore, as expressed in Eq. 2.15

$$c_\alpha = \left\{ c_1 = \frac{1}{2} + \frac{1}{2} + 0 + \frac{1}{2} + \frac{1}{2} = 2 \right\} = \frac{M}{2}. \tag{B.41}$$

B.2.3.3 Verification of $c_\alpha = c_2$

Consider the case for $N = 2$, i.e. ($c_\alpha = c_2$), and $M = 4$. From Eq. 2.15

$$c_2 = \langle T_2(\xi), T_2(\xi) \rangle, \tag{B.42}$$

and from Eq. 2.14

$$c_2 = \frac{1}{2} T_2^2(\xi_0) + T_2^2(\xi_1) + T_2^2(\xi_2) + T_2^2(\xi_3) + \frac{1}{2} T_2^2(\xi_4). \tag{B.43}$$

The second Chebyshev polynomial is given by

$$T_2 = 2\xi^2 - 1, \tag{B.44}$$

and the cosine sample points are the same as for the c_1 calculation. Substituting these sample points back into Eq. (B.44) gives

$$T_2^2(\xi_0) = 1, \tag{B.45}$$

and similarly,

$$T_2^2(\xi_1) = 0,\ T_2^2(\xi_2) = 1,\ T_2^2(\xi_3) = 0,\ T_2^2(\xi_3) = 1. \tag{B.46}$$

Therefore, as expressed in Eq. 2.15

$$c_\alpha = \left\{ c_2 = \frac{1}{2} + 0 + 1 + 0 + \frac{1}{2} = 2 \right\} = \frac{M}{2}. \tag{B.47}$$

B.2.3.4 Verification of $c_\alpha = c_3$

Consider the case for $N = 3$, i.e. ($c_\alpha = c_3$), and $M = 4$. From Eq. 2.15

$$c_3 = \langle T_3(\xi), T_3(\xi) \rangle, \tag{B.48}$$

and from Eq. 2.14

$$c_3 = \frac{1}{2}T_3^2(\xi_0) + T_3^2(\xi_1) + T_3^2(\xi_2) + T_3^2(\xi_3) + \frac{1}{2}T_3^2(\xi_4). \tag{B.49}$$

The third Chebyshev polynomial is given by

$$T_3 = 4\xi^3 - 3\xi, \tag{B.50}$$

and the cosine sample points are the same as for the c_1 calculation. Substituting these sample points back into Eq. (B.50) gives

$$T_3^2(\xi_0) = 1, \tag{B.51}$$

and similarly,

$$T_3^2(\xi_1) = \frac{1}{2},\ T_3^2(\xi_2) = 0,\ T_3^2(\xi_3) = \frac{1}{2},\ T_3^2(\xi_3) = 1. \tag{B.52}$$

Therefore, as expressed in Eq. 2.15

$$c_\alpha = \left\{ c_3 = \frac{1}{2} + \frac{1}{2} + 0 + \frac{1}{2} + \frac{1}{2} = 2 \right\} = \frac{M}{2}. \tag{B.53}$$

B.2.3.5 Verification of $c_N = c_4$ where $N = M$

Consider the case for $N = 4$, i.e. $(c_\alpha = c_4)$, and $M = 4$. From Eq. 2.15

$$c_4 = \langle T_4(\xi), T_4(\xi) \rangle, \tag{B.54}$$

and from Eq. 2.14

$$c_4 = \frac{1}{2} T_4^2(\xi_0) + T_4^2(\xi_1) + T_4^2(\xi_2) + T_4^2(\xi_3) + \frac{1}{2} T_4^2(\xi_4). \tag{B.55}$$

The fourth Chebyshev polynomial is given by

$$T_4 = 8\xi^4 - 8\xi^2 + 1, \tag{B.56}$$

and the cosine sample points are the same as for the c_1 calculation. Substituting these sample points back into Eq. B.56 gives

$$T_4^2(\xi_0) = 1, \tag{B.57}$$

and similarly,

$$T_4^2(\xi_1) = 1,\ T_4^2(\xi_2) = 1,\ T_4^2(\xi_3) = 1,\ T_4^2(\xi_3) = 1. \tag{B.58}$$

Therefore, as expressed in Eq. 2.15

$$c_\alpha = \left\{ c_4 = \frac{1}{2} + 1 + 1 + 1 + \frac{1}{2} = 4 \right\} = M. \tag{B.59}$$

B.2.3.6 Verification of $c_{\scriptscriptstyle N}$ where $M > N$

Consider the present interpolation case for $N = 4$, i.e. $(c_N = c_4)$, and $M = 5$. From Eq. 2.15

$$c_4 = \langle T_4(\xi), T_4(\xi) \rangle, \tag{B.60}$$

and from Eq. 2.14

$$c_4 = \frac{1}{2}T_4^2(\xi_0) + T_4^2(\xi_1) + T_4^2(\xi_2) + T_4^2(\xi_3) + T_4^2(\xi_4) + \frac{1}{2}T_4^2(\xi_5). \tag{B.61}$$

The fourth Chebyshev polynomial is given by

$$T_4 = 8\xi^4 - 8\xi^2 + 1, \tag{B.62}$$

and the cosine sample points are calculated as follow:

$$\xi_0 = -\cos(\frac{0\pi}{5}) = -1, \tag{B.63}$$

$$\xi_1 = -\cos(\frac{\pi}{5}) \cong -0.8090, \tag{B.64}$$

$$\xi_2 = -\cos(\frac{2\pi}{5}) \cong -0.3090, \tag{B.65}$$

$$\xi_3 = -\cos(\frac{3\pi}{5}) \cong 0.3090. \tag{B.66}$$

$$\xi_4 = -\cos(\frac{4\pi}{5}) \cong 0.8090. \tag{B.67}$$

$$\xi_5 = -\cos(\frac{5\pi}{5}) = 1. \tag{B.68}$$

Substituting these sample points back into Eq. B.62 gives

$$T_4^2(\xi_0) = 1, \tag{B.69}$$

and similarly,

$$T_4^2(\xi_1) = 0.6545,\ T_4^2(\xi_2) = 0.0955,\ T_4^2(\xi_3) = 0.0955\ T_4^2(\xi_4) = 0.6545,\ T_4^2(\xi_5) = 1. \tag{B.70}$$

Therefore, as expresses in Eq. 2.15

$$c_{N} = \left\{ c_3 = \frac{1}{2} + 0.6545 + 0.0955 + 0.0955 + 0.6545 + \frac{1}{2} = \frac{5}{2} \right\} = \frac{M}{2}. \tag{B.71}$$

The computations in this section verify Eq. 2.15, in Section 2.7.

APPENDIX C

PICARD ITERATION

The equation below, with an of initial condition $x(t_0) = x_0$, is a first order differential equation for a scalar case.

$$\frac{dx}{dt} = f(x,t) \tag{C.1}$$

A sequence of approximate solutions, $x^i(t)(i = 1, 2, ..., \infty)$, to this differential equation may be obtained through Picard iteration using the following recursion formula;

$$x^i(t) = x(t_0) + \int_{t_0}^{t} f\left(x^{i-1}, s\right) ds. \tag{C.2}$$

This exists in a time domain D surrounding the point $(t_0, x_0(t))$, and is defined in a function space by the inequalities

$$\mid t - t_0 \mid\leq a, \mid x(t) - x_0(t) \mid\leq b. \tag{C.3}$$

If $f(x(t),t)$ is a single-valued continuous function of x and t, and if $f(x(t),t)$ satisfies the Lipschitz condition, then the sequence produced by Eq. C.2 will converge to a unique and continuous solution satisfying the differential equation (Eq. C.1). The Lipschitz condition is a smoothness condition that restricts the increase of a function. For a function $f(x(t),t)$, if $(x(t),t)$ and $(X(t),t)$ are two points in the domain D, then

$$\mid f(x(t),t) - f(X(t),t) \mid< K \mid x(t) - X(t) \mid, \tag{C.4}$$

where the Lipschitz contant of the function f is given by K. It has been shown by [70] that the continuity of $f(x(t), t)$ is not necessary and that the requirements for $f(x(t), t)$ are that it is bounded. Also, all the integrals of the form $\int_0^t f(x^i, s) ds$ do exist.

For a system of equations that exhibit the first order form as shown below, with m dependent variables,

$$\frac{dx_1}{dt} = f_1(x_1(t), x_2(t), ..., x_m(t), t) \tag{C.5}$$

$$\frac{dx_2}{dt} = f_2(x_1(t), x_2(t), ..., x_m(t), t) \tag{C.6}$$

$$\vdots$$

$$\frac{dx_m}{dt} = f_m(x_1(t), x_2(t), ..., x_m(t), t), \tag{C.7}$$

[70] demonstrated that if over the $m+1$ variables in the augmented list, $f_1, f_2, ..., f_m$ are single-valued, continuous functions that are restricted to lie in the domain D defined by

$$\mid t - t_0 \mid \leq a, \mid x(t) - x_0(t) \mid \leq b_1, ..., \mid x(t) - x_m(t) \mid \leq b_m; \tag{C.8}$$

and if M is the greatest of the upper bounds of $f_1, f_2, ..., f_m$ in the domain D; and if h is the least of $\frac{b_1}{M}, \frac{b_2}{M}, ..., \frac{b_m}{M}$; and let t be further restricted if necessary by $\mid t - t_0 \mid < h$; the Lipschitz condition has the form

$$| f_r(X_1(t), X_2(t), ..., X_m(t), t) - f_r(x_1(t), x_2(t), ..., x_m(t), t) | < ...$$

$$... K_1 | X_1(t) - x_1(t) | + K_2 | X_2(t) - x_2(t) | + ... + K_m | X_m(t) - x_m(t) |, \quad \text{(C.9)}$$

where $r = 1, 2, ..., m$. The iteration then having the form

$$x_r^i(t) = x_r(t_0) + \int_0^t f_r\left(x_1^{i-1}, x_2^{i-1}, ..., x_m^{i-1}, s\right) ds \quad \text{(C.10)}$$

will converge to the unique and continuous solution shown in Eq. C.5 through Eq. C.7.

APPENDIX D

MCPI COEFFICIENT DERIVATION

This appendix provides some additional details that were not included in Chapter 3, but are useful for understanding how the MCPI algorithm is derived. The following developments continue the formulations from Sections 3.4, 3.5 and 3.6. One distinction that must be made clear is that these developments present a subtle difference compared with those presented in Bai's PhD dissertation [18]. Macomber's dissertation [40] also mentioned this subtle difference, referencing in-house tutorial notes prepared by Woollands. Those in-house tutorial notes have become the contents of Chapter 3 and this appendix. This appendix is, in some sense, a "brute force" approach wherein all Chebyshev polynomials are replaced by their equivalent power series of degree N in τ. This permits ease of integration and collection of terms. This approach, while not necessary to derive MCPI, has appeal on heuristic grounds.

D.1 First Order MCPI-IVP

In Chapter 3 some steps were shown that resulted in a set of general formulae for determining the coefficients ($\boldsymbol{\beta}$) of the "solution" trajectory. In this section we provide a more detailed derivation for the coefficients of the first order MCPI-IVP using two example cases for $N = 5$ and $N = 6$ respectively. Recall that the coefficients of the forcing function $\boldsymbol{F}$ were derived in Chapter 2 and Appendix B, and that they were obtained through least squares approximation. The $\boldsymbol{\beta}$ coefficients may then be determined in terms of these known $\boldsymbol{F}$ coefficients.

D.1.1 $N = 5$

Note that the upper limit of the summation on the **R.H.S.** is $N-1$, not N, because integration increases the degree of the polynomial by one. In Eq. D.2 through Eq. D.9 and the coefficient summary tables, the superscript i has been left off to prevent clutter.

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\beta}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{F}_k^{i-1} T_k(s) ds. \tag{D.1}$$

L.H.S.

$$\boldsymbol{\beta}_0 T_0 + \boldsymbol{\beta}_1 T_1 + \boldsymbol{\beta}_2 T_2 + \boldsymbol{\beta}_3 T_3 + \boldsymbol{\beta}_4 T_4 + \boldsymbol{\beta}_5 T_5 = R.H.S. \tag{D.2}$$

Substitute the Chebyshev polynomials $T_n(\tau)$ as a power series in τ from Eq. (3.4) to Eq. (3.6).

$$\boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \tau + \boldsymbol{\beta}_2 (2\tau^2 - 1) + \boldsymbol{\beta}_3 (4\tau^3 - 3\tau) + \boldsymbol{\beta}_4 (8\tau^4 - 8\tau^2 + 1) + \boldsymbol{\beta}_5 (16\tau^5 - 20\tau^3 + 5\tau) = R.H.S. \tag{D.3}$$

$$\boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 \tau + 2\boldsymbol{\beta}_2 \tau^2 - \boldsymbol{\beta}_2 + 4\boldsymbol{\beta}_3 \tau^3 - 3\boldsymbol{\beta}_3 \tau + ...$$

$$... + 8\boldsymbol{\beta}_4 \tau^4 - 8\boldsymbol{\beta}_4 \tau^2 + \boldsymbol{\beta}_4 + 16\boldsymbol{\beta}_5 \tau^5 - 20\boldsymbol{\beta}_5 \tau^3 + 5\boldsymbol{\beta}_5 \tau = R.H.S \tag{D.4}$$

Collect the coefficients for τ^0 to τ^5.

Degree	**Coefficient**
τ^0	$\boldsymbol{\beta}_0 - \boldsymbol{\beta}_2 + \boldsymbol{\beta}_4$
τ^1	$\boldsymbol{\beta}_1 - 3\boldsymbol{\beta}_3 + 5\boldsymbol{\beta}_5$
τ^2	$2\boldsymbol{\beta}_2 - 8\boldsymbol{\beta}_4$
τ^3	$4\boldsymbol{\beta}_3 - 20\boldsymbol{\beta}_5$
τ^4	$8\boldsymbol{\beta}_4$
τ^5	$16\boldsymbol{\beta}_5$

Table D.1: Collecting the L.H.S. coefficients of τ ($N = 5$).

R.H.S.

$$L.H.S. = \boldsymbol{x}_0 + \boldsymbol{F}_0 \int_{-1}^{\tau} T_0(s)ds + \ \boldsymbol{F}_1 \int_{-1}^{\tau} T_1(s)ds + \ \boldsymbol{F}_2 \int_{-1}^{\tau} T_2(s)ds + ...$$

$$... + \ \boldsymbol{F}_3 \int_{-1}^{\tau} T_3(s)ds + \ \boldsymbol{F}_4 \int_{-1}^{\tau} T_4(s)ds \tag{D.5}$$

Substitute the Chebyshev polynomials.

$$L.H.S. = \boldsymbol{x}(-1) + \boldsymbol{F}_0 \int_{-1}^{\tau} 1ds + \ \boldsymbol{F}_1 \int_{-1}^{\tau} sds + \ \boldsymbol{F}_2 \int_{-1}^{\tau} \left(2s^2 - 1\right) ds + ...$$

$$... + \ \boldsymbol{F}_3 \int_{-1}^{\tau} \left(4s^3 - 3s\right) ds + \ \boldsymbol{F}_4 \int_{-1}^{\tau} \left(8s^4 - 8s^2 + 1\right) ds \tag{D.6}$$

Integration yields the following:

$$L.H.S. = \boldsymbol{x}(-1) + \boldsymbol{F}_0 \left[s\right]_{-1}^{\tau} + \boldsymbol{F}_1 \left[\frac{1}{2}s^2\right]_{-1}^{\tau} + \boldsymbol{F}_2 \left[\frac{2}{3}s^3 - s\right]_{-1}^{\tau} + ...$$

$$... + \boldsymbol{F}_3 \left[s^4 - \frac{3}{2}s^2\right]_{-1}^{\tau} + \boldsymbol{F}_4 \left[\frac{8}{5}s^5 - \frac{8}{3}s^3 + s\right]_{-1}^{\tau} . \tag{D.7}$$

Evaluate the terms from -1 to τ to get

$$L.H.S. = \boldsymbol{x}(-1) + \boldsymbol{F}_0\left[\tau+1\right] + \boldsymbol{F}_1\left[\frac{1}{2}\tau^2 - \frac{1}{2}\right] + \boldsymbol{F}_2\left[\frac{2}{3}\tau^3 - \tau - \left(-\frac{2}{3} - (-1)\right)\right] + ...$$

$$... + \boldsymbol{F}_3\left[\tau^4 - \frac{3}{2}\tau^2 - \left(1 - \frac{3}{2}\right)\right] + \boldsymbol{F}_4\left[\frac{8}{5}\tau^5 - \frac{8}{3}\tau^3 + \tau - \left(-\frac{8}{5} + \frac{8}{3} - 1\right)\right]. \quad \text{(D.8)}$$

Expand and simplify as follows:

$$L.H.S. = \boldsymbol{x}(-1) + \boldsymbol{F}_0\tau + \boldsymbol{F}_0 + \frac{1}{2}\boldsymbol{F}_1\tau^2 - \frac{1}{2}\boldsymbol{F}_1 + \frac{2}{3}\boldsymbol{F}_2\tau^3 - \boldsymbol{F}_2\tau - \frac{1}{3}\boldsymbol{F}_2 + ...$$

$$... + \boldsymbol{F}_3\tau^4 - \frac{3}{2}\boldsymbol{F}_3\tau^2 + \frac{1}{2}\boldsymbol{F}_3 + \frac{8}{5}\boldsymbol{F}_4\tau^5 - \frac{8}{3}\boldsymbol{F}_4\tau^3 + \boldsymbol{F}_4\tau - \frac{1}{15}\boldsymbol{F}_4. \quad \text{(D.9)}$$

Collect the coefficients for τ^0 to τ^6.

Degree	**Coefficient**
τ^0	$\boldsymbol{x}(-1) + \boldsymbol{F}_0 - \frac{1}{2}\boldsymbol{F}_1 - \frac{1}{3}\boldsymbol{F}_2 + \frac{1}{2}\boldsymbol{F}_3 - \frac{1}{15}\boldsymbol{F}_4$
τ^1	$\boldsymbol{F}_0 - \boldsymbol{F}_2 + \boldsymbol{F}_4$
τ^2	$\frac{1}{2}\boldsymbol{F}_1 - \frac{3}{2}\boldsymbol{F}_3$
τ^3	$\frac{2}{3}\boldsymbol{F}_2 - \frac{8}{3}\boldsymbol{F}_4$
τ^4	$\boldsymbol{F}_3$
τ^5	$\frac{8}{5}\boldsymbol{F}_4$

Table D.2: Collecting the R.H.S. coefficients of τ ($N = 5$).

Equate the terms from the **L.H.S.** and the **R.H.S.** starting with $\boldsymbol{\beta}_5$.

$$\boldsymbol{\beta}_5^i = \frac{1}{10}\boldsymbol{F}_4^{i-1} \quad \text{(D.10)}$$

$$\boldsymbol{\beta}_4^i = \frac{1}{8}\boldsymbol{F}_3^{i-1} \quad \text{(D.11)}$$

$$\boldsymbol{\beta}_3^i = \frac{1}{6}\left(\boldsymbol{F}_2^{i-1} - \boldsymbol{F}_4^{i-1}\right) \tag{D.12}$$

$$\boldsymbol{\beta}_2^i = \frac{1}{4}\left(\boldsymbol{F}_1^{i-1} - \boldsymbol{F}_3^{i-1}\right) \tag{D.13}$$

$$\boldsymbol{\beta}_1^i = \frac{1}{2}\left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1}\right) \tag{D.14}$$

To solve for $\boldsymbol{\beta}_0$ substitute $\tau = -1$ into Eq. D.4 and Eq. D.5. Equating leads to the following result.

$$\boldsymbol{\beta}_0^i = \boldsymbol{x}(-1) + \boldsymbol{\beta}_1^i - \boldsymbol{\beta}_2^i + \boldsymbol{\beta}_3^i - \boldsymbol{\beta}_4^i + \boldsymbol{\beta}_5^i. \tag{D.15}$$

D.1.2 Example N = 6

In this section we perform the same computations as above, to determine the $\boldsymbol{\beta}$ coefficients as a function of the $\boldsymbol{F}$ coefficients, but with $N = 6$.

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\beta}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{F}_k^{i-1} T_k(\mathbf{s}) d\mathbf{s}. \tag{D.16}$$

Degree	**Coefficient**
τ^0	$\boldsymbol{\beta}_0 - \boldsymbol{\beta}_2 + \boldsymbol{\beta}_4 - \boldsymbol{\beta}_6$
τ^1	$\boldsymbol{\beta}_1 - 3\boldsymbol{\beta}_3 + 5\boldsymbol{\beta}_5$
τ^2	$2\boldsymbol{\beta}_2 - 8\boldsymbol{\beta}_4 + 18\boldsymbol{\beta}_6$
τ^3	$4\boldsymbol{\beta}_3 - 20\boldsymbol{\beta}_5$
τ^4	$8\boldsymbol{\beta}_4 - 48\boldsymbol{\beta}_6$
τ^5	$16\boldsymbol{\beta}_5$
τ^6	$32\boldsymbol{\beta}_6$

Table D.3: Collecting the L.H.S. coefficients of τ ($N = 6$)

Degree	**Coefficient**
τ^0	$\boldsymbol{x}(-1) + \boldsymbol{F}_0 - \frac{1}{2}\boldsymbol{F}_1 - \frac{1}{3}\boldsymbol{F}_2 + \frac{1}{2}\boldsymbol{F}_3 - \frac{1}{15}\boldsymbol{F}_4 - \frac{1}{6}\boldsymbol{F}_5$
τ^1	$\boldsymbol{F}_0 - \boldsymbol{F}_2 + \boldsymbol{F}_4$
τ^2	$\frac{1}{2}\boldsymbol{F}_1 - \frac{3}{2}\boldsymbol{F}_3 + \frac{5}{2}\boldsymbol{F}_5$
τ^3	$\frac{2}{3}\boldsymbol{F}_2 - \frac{8}{3}\boldsymbol{F}_4$
τ^4	$\boldsymbol{F}_3 - 5\boldsymbol{F}_5$
τ^5	$\frac{8}{5}\boldsymbol{F}_4$
τ^6	$\frac{8}{3}\boldsymbol{F}_5$

Table D.4: Collecting the R.H.S. coefficients of τ $(N = 6)$.

Equate the terms from the **L.H.S.** and the **R.H.S.** starting with $\boldsymbol{\beta}_6$.

$$\boldsymbol{\beta}_6^i = \frac{1}{12}\boldsymbol{F}_5^{i-1}, \tag{D.17}$$

$$\boldsymbol{\beta}_5^i = \frac{1}{10}\boldsymbol{F}_4^{i-1}, \tag{D.18}$$

$$\boldsymbol{\beta}_4^i = \frac{1}{8}\left(\boldsymbol{F}_3^{i-1} - \boldsymbol{F}_5^{i-1}\right), \tag{D.19}$$

$$\boldsymbol{\beta}_3^i = \frac{1}{6}\left(\boldsymbol{F}_2^{i-1} - \boldsymbol{F}_4^{i-1}\right), \tag{D.20}$$

$$\boldsymbol{\beta}_2^i = \frac{1}{4}\left(\boldsymbol{F}_1^{i-1} - \boldsymbol{F}_3^{i-1}\right), \tag{D.21}$$

$$\boldsymbol{\beta}_1^i = \frac{1}{2}\left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1}\right), \tag{D.22}$$

$$\boldsymbol{\beta}_0^i = \boldsymbol{x}(-1) + \boldsymbol{\beta}_1^i - \boldsymbol{\beta}_2^i + \boldsymbol{\beta}_3^i - \boldsymbol{\beta}_4^i + \boldsymbol{\beta}_5^i - \boldsymbol{\beta}_6^i. \tag{D.23}$$

D.1.3 Coefficients Summarized

Looking at the previous two examples it is easy to see that a set of general formulae may be derived for the $\boldsymbol{\beta}$ coefficients as a function of the $\boldsymbol{F}$ coefficients. These formulae are given below and are the same as those given in Eqs 3.16 through 3.20 in Chapter 3. These coefficients for $\boldsymbol{\beta}$ can be arranged in a vector matrix form as shown in Eqs 3.21 through 3.23 in Chapter 3, thus allowing a one time computation of the matrices prior to integration.

$$\boldsymbol{\beta}_0^i = \boldsymbol{x}_0 + \sum_{k=1}^{k=N} (-1)^{k+1} \boldsymbol{\beta}_k^i, \tag{D.24}$$

$$\boldsymbol{\beta}_1^i = \frac{1}{2} \left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1} \right), \tag{D.25}$$

$$\boldsymbol{\beta}_k^i = \frac{1}{2k} \left(\boldsymbol{F}_{k-1}^{i-1} - \boldsymbol{F}_{k+1}^{i-1} \right), \; k = 1, 2, ..., N-1, \tag{D.26}$$

$$\boldsymbol{\beta}_{N-1}^i = \frac{\boldsymbol{F}_{N-2}^{i-1}}{2(N-1)}, \tag{D.27}$$

and

$$\boldsymbol{\beta}_N^i = \frac{\boldsymbol{F}_{N-1}^{i-1}}{2N}. \tag{D.28}$$

D.2 Second Order MCPI-IVP

The formulation for the second order case is similar to that of the first order case but since integration is performed twice, the method can be formulated in a cascade fashion where the result from the first integration (velocity) is directly integrated to obtain the second result (position). This approach preserves kinematic consistency. Note that for the first integration the upper limit of the summation for the coefficients approximating the forcing function is $N-2$, and in the second integration the upper limit of the summation is increased to $N-1$. The upper limits are applied because integration increases the degree of the polynomial by one. The approximation of the solution trajectory on the **L.H.S** is of course summed to N.

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \left\{ \boldsymbol{v}(-1) + \int_{-1}^{s} \sum_{k=0}^{N-2} \boldsymbol{F}_k^{i-1} T_k(q)dq \right\} ds. \quad \text{(D.29)}$$

Solving for $\boldsymbol{\alpha}$ directly in terms of $\boldsymbol{F}$ is possible but it leads to a set of complicated formulae that do not exhibit any obvious pattern, thus making it very challenging to code the method for arbitrary N. Instead we consider formulating the coefficients of $\boldsymbol{\beta}$ in terms of $\boldsymbol{F}$, and then the coefficients of $\boldsymbol{\alpha}$ in terms of $\boldsymbol{\beta}$. This results in two nice sets of general formulae the resemble those derived for the first order case.

D.2.1 Velocity Approximation

Consider the following velocity approximation for $N = 6$,

$$\boldsymbol{v}^i(\tau) = \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(\tau) = \boldsymbol{v}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-2} \boldsymbol{F}_k^{i-1} T_k(s)ds. \quad \text{(D.30)}$$

Substituting in the Chebyshev polynomials, simplifying and equating leads to the following set of equations:

$$\boldsymbol{\beta}_5^i = \frac{1}{10}\boldsymbol{F}_4^{i-1}, \quad \text{(D.31)}$$

$$\boldsymbol{\beta}_4^i = \frac{1}{8}\boldsymbol{F}_3^{i-1}, \quad \text{(D.32)}$$

$$\boldsymbol{\beta}_3^i = \frac{1}{6}\left(\boldsymbol{F}_2^{i-1} - \boldsymbol{F}_4^{i-1}\right), \quad \text{(D.33)}$$

$$\boldsymbol{\beta}_2^i = \frac{1}{4}\left(\boldsymbol{F}_1^{i-1} - \boldsymbol{F}_3^{i-1}\right), \quad \text{(D.34)}$$

$$\boldsymbol{\beta}_1^i = \frac{1}{2}\left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1}\right), \quad \text{(D.35)}$$

$$\boldsymbol{\beta}_0^i = \boldsymbol{x}(-1) + \left(\boldsymbol{\beta}_1^i - \boldsymbol{\beta}_2^i + \boldsymbol{\beta}_3^i - \boldsymbol{\beta}_4^i + \boldsymbol{\beta}_5^i\right). \quad \text{(D.36)}$$

These can be constructed as a set of general formulae as follows:

$$\boldsymbol{\beta}_0^i = \boldsymbol{x}_0 + \sum_{k=1}^{k=N-1} (-1)^{k+1} \boldsymbol{\beta}_k^i, \tag{D.37}$$

$$\boldsymbol{\beta}_1^i = \frac{1}{2}\left(2\boldsymbol{F}_0^{i-1} - \boldsymbol{F}_2^{i-1}\right), \tag{D.38}$$

$$\boldsymbol{\beta}_k^i = \frac{1}{2k}\left(\boldsymbol{F}_{k-1}^{i-1} - \boldsymbol{F}_{k+1}^{i-1}\right),\ k = 1, 2, ..., N-3, \tag{D.39}$$

$$\boldsymbol{\beta}_{N-2}^i = \frac{\boldsymbol{F}_{N-3}^{i-1}}{2(N-2)}, \tag{D.40}$$

and

$$\boldsymbol{\beta}_{N-1}^i = \frac{\boldsymbol{F}_{N-2}^{i-1}}{2(N-1)}. \tag{D.41}$$

D.2.2 Position Approximation

Consider the following position approximation for $N = 6$,

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\alpha}_k^i T_k(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{\beta}_k^i T_k(s) ds. \tag{D.42}$$

Substituting in the Chebyshev polynomials, simplifying and equating leads to the following set of equations:

$$\boldsymbol{\alpha}_6^i = \frac{1}{12}\boldsymbol{\beta}_5^i \tag{D.43}$$

$$\boldsymbol{\alpha}_5^i = \frac{1}{10}\boldsymbol{\beta}_4^i \tag{D.44}$$

$$\boldsymbol{\alpha}_4^i = \frac{1}{8}\left(\boldsymbol{\beta}_3^i - \boldsymbol{\beta}_5^i\right) \tag{D.45}$$

$$\boldsymbol{\alpha}_3^i = \frac{1}{6}\left(\boldsymbol{\beta}_2^i - \boldsymbol{\beta}_4^i\right) \tag{D.46}$$

$$\boldsymbol{\alpha}_2^i = \frac{1}{4}\left(\boldsymbol{\beta}_1^i - \boldsymbol{\beta}_3^i\right) \tag{D.47}$$

$$\boldsymbol{\alpha}_1^i = \frac{1}{2}\left(2\boldsymbol{\beta}_0^i - \boldsymbol{\beta}_2^i\right) \tag{D.48}$$

$$\boldsymbol{\alpha}_0^i = \boldsymbol{x}_0 + \left(\boldsymbol{\alpha}_1^i - \boldsymbol{\alpha}_2^i + \boldsymbol{\alpha}_3^i - \boldsymbol{\alpha}_4^i + \boldsymbol{\alpha}_5^i - \boldsymbol{\alpha}_6^i\right) \tag{D.49}$$

These can be constructed as a set of general formulae as follows:

$$\boldsymbol{\alpha}_0^i = \boldsymbol{x}_0 + \sum_{k=1}^{k=N}(-1)^{k+1}\boldsymbol{\alpha}_k^i, \tag{D.50}$$

$$\boldsymbol{\alpha}_1^i = \frac{1}{2}\left(2\boldsymbol{\beta}_0^i - \boldsymbol{\beta}_2^i\right), \tag{D.51}$$

$$\boldsymbol{\alpha}_k^i = \frac{1}{2k}\left(\boldsymbol{\beta}_{k-1}^i - \boldsymbol{\beta}_{k+1}^i\right),\ k = 1, 2, ..., N-1, \tag{D.52}$$

$$\boldsymbol{\alpha}_{N-1}^i = \frac{\boldsymbol{\beta}_{N-2}^i}{2(N-1)}, \tag{D.53}$$

and

$$\boldsymbol{\alpha}_N^i = \frac{\boldsymbol{\beta}_{N-1}^i}{N}. \tag{D.54}$$

The sets of general formulae that are derived in this section can be represented in vector matrix notation as shown in Eqs 3.40 through 3.43 in Chapter 3. These matrices can be computed and stored for arbitrary values of N and then called when required by the MCPI algorithms.

D.3 Second Order MCPI-TPBVP

To derive the MCPI-TPBVP the same approach is used and the final result only differs in the way the first two $\boldsymbol{\alpha}$ coefficients are computed as these contain the unknown constant of integration for the first integration (i.e. the unknown initial velocity). The necessary details for deriving the MCPI-TPBVP are already given in Chapter 3, Section 3.6. Also presented is the approach for extraction of the unknown initial velocity post integration.

APPENDIX E

KS DERIVATION

Considerable algebra is required to derive the perturbed two-body equations of motion (Eq. 5.11) shown in Chapter 5. In this appendix we present a step-by-step derivation for the planar (two-dimensional) case starting with the Cartesian perturbed two-body equations of motion.

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{F}, \tag{E.1}$$

and for $Z(t) = 0$ this reduces to

$$\begin{Bmatrix} \ddot{X} \\ \ddot{Y} \end{Bmatrix} = -\frac{\mu}{r^3}\begin{Bmatrix} X \\ Y \end{Bmatrix} + \boldsymbol{F}. \tag{E.2}$$

Restricting the motion to the plane we know that the following relationship is true

$$\begin{Bmatrix} X \\ Y \end{Bmatrix} = L(\boldsymbol{u})\boldsymbol{u}, \quad \text{where} \quad L(\boldsymbol{u}) = \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}. \tag{E.3}$$

Eq. E.2 now becomes

$$\begin{Bmatrix} \ddot{X} \\ \ddot{Y} \end{Bmatrix} = -\frac{\mu}{r^3}\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} + \boldsymbol{F}. \tag{E.4}$$

Taking the time derivative of Eq. E.3 leads to

$$\begin{Bmatrix} \dot{X} \\ \dot{Y} \end{Bmatrix} = \begin{bmatrix} \dot{u}_1 & -\dot{u}_2 \\ \dot{u}_2 & \dot{u}_1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} + \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} \dot{u}_1 \\ \dot{u}_2 \end{Bmatrix}. \tag{E.5}$$

Note that $\frac{d}{dE} = \frac{d}{dt}\frac{dt}{dE}$ and from Eq. 5.5 $\frac{dt}{dE} = \left(\frac{1}{\sqrt{\mu\alpha}}\right) r$, where $\alpha = \frac{2}{r} - \frac{\dot{r}^T\dot{r}}{\mu}$. To simplify notation we adopt the following convention for derivatives $\dot{(\,)} = \frac{\sqrt{\mu\alpha}}{r}(\,)'$, where the dot represents derivatives with respect to time and the prime represents derivatives with respect to eccentric anomaly. The right hand side of Eq. E.5 can now be written as

$$\begin{Bmatrix} \dot{X} \\ \dot{Y} \end{Bmatrix} = \frac{\sqrt{\mu\alpha}}{r} \begin{bmatrix} u_1' & -u_2' \\ u_2' & u_1' \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} + \frac{\sqrt{\mu\alpha}}{r} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}, \tag{E.6}$$

which can be further simplified to

$$\begin{Bmatrix} \dot{X} \\ \dot{Y} \end{Bmatrix} = \frac{2\sqrt{\mu\alpha}}{r} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}. \tag{E.7}$$

Differentiating with respect to time again leads to

$$\begin{Bmatrix} \ddot{X} \\ \ddot{Y} \end{Bmatrix} = \frac{-2\sqrt{\mu\alpha}}{r^2}\dot{r} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix} + \sqrt{\frac{\mu}{\alpha}}\frac{\dot{\alpha}}{r} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix} + \cdots$$

$$\cdots + \frac{2\mu\alpha}{r^2} \begin{bmatrix} u_1' & -u_2' \\ u_2' & u_1' \end{bmatrix} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix} + \frac{2\mu\alpha}{r^2} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix}. \tag{E.8}$$

In Eq. E.8, $\dot{r} = \frac{\sqrt{\mu\alpha}}{r} r'$ and $\dot{\alpha}$ is the time derivative of $\alpha = \frac{2}{r} - \frac{\dot{r}^T \dot{r}}{\mu}$ (more on this later in the derivation). Since $r = u_1^2 + u_2^2$, $r' = 2(u_1 u_1' + u_2 u_2')$, and thus $\dot{r} = \frac{2\sqrt{\mu\alpha}}{r}(u_1 u_1' + u_2 u_2')$. Substituting these into Eq. E.8 results in

$$\left\{ \begin{array}{c} \ddot{X} \\ \ddot{Y} \end{array} \right\} = \frac{-4\sqrt{\mu\alpha}}{r^3} \begin{bmatrix} u_1^2 u_1' + u_1 u_2 u_2' & -u_1 u_1' u_2 - u_2^2 u_2' \\ u_1 u_1' u_2 + u_2^2 u_2' & u_1^2 u_1' + u_1 u_2 u_2' \end{bmatrix} \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} + \cdots$$

$$\cdots + \sqrt{\frac{\mu}{\alpha}} \frac{\dot{\alpha}}{r} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} + \frac{2\mu\alpha}{r^2} \begin{bmatrix} u_1' & -u_2' \\ u_2' & u_1' \end{bmatrix} \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} + \cdots$$

$$\cdots + \frac{2\mu\alpha}{r^2} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \left\{ \begin{array}{c} u_1'' \\ u_2'' \end{array} \right\}. \tag{E.9}$$

The first and third terms can be combined using the following substitution, $1 = \frac{r}{r} = \frac{u_1^2 + u_2^2}{r}$, which allows a common denominator to be created.

$$\left\{ \begin{array}{c} \ddot{X} \\ \ddot{Y} \end{array} \right\} = \frac{-4\mu\alpha}{r^3} \left\{ \begin{array}{c} u_1^2 u_1'^2 + u_1 u_1' u_2 u_2' - u_1 u_1' u_2 u_2' - u_2^2 u_2'^2 \\ u_1 u_1'^2 u_2 + u_1' u_2^2 u_2' + u_1^2 u_1' u_2' + u_1 u_2 u_2'^2 \end{array} \right\} + \cdots$$

$$\cdots + \sqrt{\frac{\mu}{\alpha}} \frac{\dot{\alpha}}{r} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} + \frac{2\mu\alpha}{r^3} \left\{ \begin{array}{c} u_1^2 u_1'^2 + u_1'^2 u_2^2 - u_1^2 u_2'^2 - u_2^2 u_2'^2 \\ 2u_1^2 u_1' u_2' + 2u_2^2 u_1' u_2' \end{array} \right\} + \cdots$$

$$\cdots + \frac{2\mu\alpha}{r^2} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \left\{ \begin{array}{c} u_1'' \\ u_2'' \end{array} \right\}. \tag{E.10}$$

Combining the first and third terms leads to

$$\left\{ \begin{array}{c} \ddot{X} \\ \ddot{Y} \end{array} \right\} = \tfrac{\mu\alpha}{r^3} \left\{ \begin{array}{c} -4u_1^2u_1'^2 + 4u_2^2u_2'^2 + 2u_1^2u_1'^2 + 2u_1'^2u_2^2 - 2u_1^2u_2'^2 - 2u_2^2u_2'^2 \\ -4u_1u_2\left(u_1'^2 + u_2'^2\right) \end{array} \right\} + \cdots$$
$$\cdots + \sqrt{\frac{\mu}{\alpha}}\frac{\dot{\alpha}}{r} \left[\begin{array}{cc} u_1 & -u_2 \\ u_2 & u_1 \end{array} \right] \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} + \frac{2\mu\alpha}{r^2} \left[\begin{array}{cc} u_1 & -u_2 \\ u_2 & u_1 \end{array} \right] \left\{ \begin{array}{c} u_1'' \\ u_2'' \end{array} \right\}. \tag{E.11}$$

Further simplification leads to

$$\left\{ \begin{array}{c} \ddot{X} \\ \ddot{Y} \end{array} \right\} = \tfrac{2\mu\alpha}{r^3} \left\{ \begin{array}{c} -u_1^2u_1'^2 + u_2^2u_2'^2 + u_1'^2u_2^2 - u_1^2u_2'^2 \\ -2u_1u_2\left(u_1'^2 + u_2'^2\right) \end{array} \right\} + \cdots$$
$$\cdots + \sqrt{\frac{\mu}{\alpha}}\frac{\dot{\alpha}}{r} \left[\begin{array}{cc} u_1 & -u_2 \\ u_2 & u_1 \end{array} \right] \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} + \frac{2\mu\alpha}{r^2} \left[\begin{array}{cc} u_1 & -u_2 \\ u_2 & u_1 \end{array} \right] \left\{ \begin{array}{c} u_1'' \\ u_2'' \end{array} \right\}. \tag{E.12}$$

Even further simplification leads to

$$\left\{ \begin{array}{c} \ddot{X} \\ \ddot{Y} \end{array} \right\} = \tfrac{2\mu\alpha}{r^3} \left\{ \begin{array}{c} -\left(u_1^2 - u_2^2\right)u_1'^2 - \left(u_1^2 - u_2^2\right)u_2'^2 \\ -2u_1u_2\left(u_1'^2 + u_2'^2\right) \end{array} \right\} + \cdots$$
$$\cdots + \sqrt{\frac{\mu}{\alpha}}\frac{\dot{\alpha}}{r} \left[\begin{array}{cc} u_1 & -u_2 \\ u_2 & u_1 \end{array} \right] \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} + \frac{2\mu\alpha}{r^2} \left[\begin{array}{cc} u_1 & -u_2 \\ u_2 & u_1 \end{array} \right] \left\{ \begin{array}{c} u_1'' \\ u_2'' \end{array} \right\}. \tag{E.13}$$

Even further simplification leads to

$$\left\{ \begin{array}{c} \ddot{X} \\ \ddot{Y} \end{array} \right\} = \tfrac{-2\mu\alpha}{r^3} \left\{ \begin{array}{c} \left(u_1^2 - u_2^2\right)\left(u_1'^2 + u_2'^2\right) \\ 2u_1u_2\left(u_1'^2 + u_2'^2\right) \end{array} \right\} + \cdots$$

$$\cdots + \sqrt{\frac{\mu}{\alpha}}\frac{\dot{\alpha}}{r}\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}\begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix} + \frac{2\mu\alpha}{r^2}\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}\begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix}. \tag{E.14}$$

Even further simplification leads to

$$\begin{Bmatrix} \ddot{X} \\ \ddot{Y} \end{Bmatrix} = \tfrac{-2\mu\alpha}{r^3}\left(u_1'^2 + u_2'^2\right)\begin{Bmatrix} u_1^2 - u_2^2 \\ 2u_1u_2 \end{Bmatrix} + \cdots$$

$$\cdots + \sqrt{\frac{\mu}{\alpha}}\frac{\dot{\alpha}}{r}\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}\begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix} + \frac{2\mu\alpha}{r^2}\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}\begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix}. \tag{E.15}$$

Recall

$$\begin{Bmatrix} \ddot{X} \\ \ddot{Y} \end{Bmatrix} = -\frac{\mu}{r^3}\begin{Bmatrix} u_1^2 - u_2^2 \\ 2u_1u_2 \end{Bmatrix} + \boldsymbol{F}. \tag{E.16}$$

Equating Eq. E.15 and Eq. E.16 and canceling $\frac{\mu}{r^3}$ leads to the following

$$-2\alpha\left(u_1'^2 + u_2'^2\right)\begin{Bmatrix} u_1^2 - u_2^2 \\ 2u_1u_2 \end{Bmatrix} + \tfrac{\dot{\alpha}r^2}{\sqrt{\mu\alpha}}\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}\begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix} + \cdots$$

$$\cdots + 2\alpha r\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}\begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix} = -\begin{Bmatrix} u_1^2 - u_2^2 \\ 2u_1u_2 \end{Bmatrix} + \frac{r^3}{\mu}\boldsymbol{F}. \tag{E.17}$$

Rearrange as follows

$$\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix} = \tfrac{-\left(1-2\alpha\left(u_1'^2+u_2'^2\right)\right)}{2\alpha r} \begin{Bmatrix} u_1^2 - u_2^2 \\ 2u_1u_2 \end{Bmatrix} + \cdots$$

$$\cdots + \frac{r^2}{2\mu\alpha}\boldsymbol{F} - \frac{\dot{\alpha}r}{2\alpha\sqrt{\mu\alpha}} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}. \tag{E.18}$$

Note that

$$L(\boldsymbol{u})L^T(\boldsymbol{u}) = rI = r \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } L^T(\boldsymbol{u}) = rL^{-1}(\boldsymbol{u}), \tag{E.19}$$

which leads to

$$\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}^{-1} = \frac{1}{r} \begin{bmatrix} u_1 & -u_2 \\ -u_2 & u_1 \end{bmatrix}. \tag{E.20}$$

Using Eq. E.20, u_1'' and u_2'' can be written as

$$\begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix} = \tfrac{-\left(1-2\alpha\left(u_1'^2+u_2'^2\right)\right)}{2\alpha r^2} \begin{bmatrix} u_1 & u_2 \\ -u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1^2 - u_2^2 \\ 2u_1u_2 \end{Bmatrix} + \tfrac{r}{2\mu\alpha} \begin{bmatrix} u_1 & u_2 \\ -u_2 & u_1 \end{bmatrix} \boldsymbol{F} + \cdots$$

$$\cdots - \frac{\dot{\alpha}r}{2\alpha\sqrt{\mu\alpha}} \begin{bmatrix} u_1 & u_2 \\ -u_2 & u_1 \end{bmatrix}^{-1} \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}. \tag{E.21}$$

Further simplification leads to

$$\begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix} = \frac{-\left(1-2\alpha\left(u_1'^2+u_2'^2\right)\right)}{2\alpha r^2} \begin{Bmatrix} u_1\left(u_1^2+u_2^2\right) \\ u_2\left(u_1^2+u_2^2\right) \end{Bmatrix} + \frac{r}{2\mu\alpha} L^T(\boldsymbol{u})\boldsymbol{F} - \frac{\dot{\alpha} r}{2\alpha\sqrt{\mu\alpha}} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}. \tag{E.22}$$

Simplifying further leads to

$$\begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix} = \frac{-\left(1-2\alpha\left(u_1'^2+u_2'^2\right)\right)}{2\alpha r} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} + \frac{r}{2\mu\alpha} L^T(\boldsymbol{u})\boldsymbol{F} - \frac{\dot{\alpha} r}{2\alpha\sqrt{\mu\alpha}} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}. \tag{E.23}$$

Note $(u_1'^2+u_2'^2) = \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}^T \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}$, but from Eq. E.7

$$\begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix} = \frac{1}{2\sqrt{\mu\alpha}} \begin{bmatrix} u_1 & u_2 \\ -u_2 & u_1 \end{bmatrix} \begin{Bmatrix} \dot{X} \\ \dot{Y} \end{Bmatrix}. \tag{E.24}$$

So

$$(u_1'^2+u_2'^2) = \tfrac{1}{4\mu\alpha} \begin{Bmatrix} \dot{X} \\ \dot{Y} \end{Bmatrix}^T \underbrace{\begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix} \begin{bmatrix} u_1 & u_2 \\ -u_2 & u_1 \end{bmatrix}}_{\left(u_1^2+u_2^2\right)I=rI} \begin{Bmatrix} \dot{X} \\ \dot{Y} \end{Bmatrix} \cdots$$

$$= \frac{r}{4\mu\alpha}\left(\dot{X}^2+\dot{Y}^2\right) = \frac{r}{4\mu\alpha}v^2. \tag{E.25}$$

Substituting Eq. E.25 into Eq. E.23 gives

$$\begin{bmatrix} u_1'' \\ u_2'' \end{bmatrix} = \frac{-\left(1 - 2\alpha\frac{r}{4\mu\alpha}v^2\right)}{2\alpha r} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} + \frac{r}{2\mu\alpha} L^T(\boldsymbol{u})\boldsymbol{F} - \frac{\dot{\alpha} r}{2\alpha\sqrt{\mu\alpha}} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}. \tag{E.26}$$

Further simplification leads to

$$\begin{Bmatrix} u_1'' \\ u_2'' \end{Bmatrix} = -\frac{1}{2\alpha}\left(\frac{1}{r} - \frac{v^2}{2\mu}\right) \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} + \frac{r}{2\mu\alpha} L^T(\boldsymbol{u})\boldsymbol{F} - \frac{\dot{\alpha} r}{2\alpha\sqrt{\mu\alpha}} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}. \tag{E.27}$$

Recall that the energy integral is given as

$$\alpha = \frac{1}{a} = \frac{2}{r} - \frac{v^2}{\mu} = 2\left(\frac{1}{r} - \frac{v^2}{2\mu}\right). \tag{E.28}$$

Substituting the energy integral into the first term of Eq. E.27 leads to

$$\begin{bmatrix} u_1'' \\ u_2'' \end{bmatrix} = -\frac{1}{4} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} + \frac{r}{2\mu\alpha} L^T(\boldsymbol{u})\boldsymbol{F} - \frac{\dot{\alpha} r}{2\alpha\sqrt{\mu\alpha}} \begin{Bmatrix} u_1' \\ u_2' \end{Bmatrix}. \tag{E.29}$$

Note that

$$\alpha = \frac{2}{r} - \frac{\dot{\boldsymbol{r}}^T \dot{\boldsymbol{r}}}{\mu}, \tag{E.30}$$

where $\dot{\boldsymbol{r}} = 2L(\boldsymbol{u})\frac{d\boldsymbol{u}}{dt} = \frac{2\sqrt{\alpha\mu}}{r}L(\boldsymbol{u})\boldsymbol{u}'$.

The Cartesian velocity dot-product can be written as

$$\dot{\boldsymbol{r}}^T\dot{\boldsymbol{r}} = \frac{4\alpha\mu}{r^2}\boldsymbol{u}'^T L^T(\boldsymbol{u})L(\boldsymbol{u})\boldsymbol{u}'. \tag{E.31}$$

This leads to an alternate form of the vis-viva equation in KS variables

$$\alpha = \frac{2}{r}\left[1 + \frac{4}{r}\boldsymbol{u}'^T\boldsymbol{u}'\right]^{-1}. \tag{E.32}$$

Differentiating to obtain $\dot{\alpha}$, and using $r^2 = \boldsymbol{r}^T\boldsymbol{r}$ and the derivative $2r\dot{r} = 2\dot{\boldsymbol{r}}^T\boldsymbol{r}$ in Eq. E.30, gives

$$\dot{\alpha} = -\frac{2}{\mu}\dot{\boldsymbol{r}}^T\boldsymbol{F} = -\frac{2}{\mu}\frac{2\sqrt{\mu\alpha}}{r}\boldsymbol{u}'^T L^T(\boldsymbol{u})\boldsymbol{F} = \frac{-4}{r}\sqrt{\frac{\alpha}{\mu}}\boldsymbol{u}'^T L^T(\boldsymbol{u})\boldsymbol{F}. \tag{E.33}$$

Substituting this into Eq. E.29 gives

$$\left\{\begin{array}{c} u_1'' \\ u_2'' \end{array}\right\} = -\frac{1}{4}\boldsymbol{u} + \frac{r}{2\mu\alpha}L^T(\boldsymbol{u})\boldsymbol{F} + \frac{2}{\mu\alpha}\boldsymbol{u}'^T L^T(\boldsymbol{u})\boldsymbol{F}\boldsymbol{u}'. \tag{E.34}$$

The third term of Eq. E.34 can be rearranged as follows:

$$\text{term 3} = \frac{2}{\mu\alpha}\left\{\begin{array}{c} u_1' \\ u_2' \end{array}\right\}^T \left[\begin{array}{cc} u_1 & u_2 \\ -u_2 & u_1 \end{array}\right]\left\{\begin{array}{c} f_1 \\ f_2 \end{array}\right\}\left\{\begin{array}{c} u_1' \\ u_2' \end{array}\right\}. \tag{E.35}$$

Further rearranging leads to

$$\text{term } 3 = \frac{2}{\mu\alpha} \left\{ \begin{array}{c} (u_1 f_1 + u_2 f_2)\, u_1' \\ (-u_2 f_1 + u_1 f_2)\, u_2' \end{array} \right\} \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\}, \tag{E.36}$$

and then

$$\text{term } 3 = \frac{2}{\mu\alpha} \left\{ \begin{array}{c} u_1' \left(u_1 u_1' - u_2 u_2'\right) f_1 + u_1' \left(u_2 u_1' + u_1 u_2'\right) f_2 \\ u_2' \left(u_1 u_1' - u_2 u_2'\right) f_1 + u_2' \left(u_2 u_1' + u_1 u_2'\right) f_2 \end{array} \right\}. \tag{E.37}$$

Factorizing out the perturbing forces f_1 and f_2 gives

$$\text{term } 3 = \frac{2}{\mu\alpha} \left[\begin{array}{cc} u_1' \left(u_1 u_1' - u_2 u_2'\right) & u_1' \left(u_2 u_1' + u_1 u_2'\right) \\ u_2' \left(u_1 u_1' - u_2 u_2'\right) & u_2' \left(u_2 u_1' + u_1 u_2'\right) \end{array} \right] \left\{ \begin{array}{c} f_1 \\ f_2 \end{array} \right\}. \tag{E.38}$$

Still further rearranging results in

$$\text{term } 3 = \frac{2}{\mu\alpha} \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} \left\{ \begin{array}{c} u_1 u_1' - u_2 u_2' \\ u_2 u_1' + u_1 u_2' \end{array} \right\}^T \boldsymbol{F}, \tag{E.39}$$

and then

$$\text{term } 3 = \frac{2}{\mu\alpha} \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\} \left\{ \begin{array}{c} u_1' \\ u_2' \end{array} \right\}^T \left[\begin{array}{cc} u_1 & u_2 \\ -u_2 & u_1 \end{array} \right] \boldsymbol{F}. \tag{E.40}$$

This leaves us with a nice form for the third term

$$\text{term 3} = \frac{2}{\mu\alpha}\boldsymbol{u}'\boldsymbol{u}'^T L^T(\boldsymbol{u})\boldsymbol{F}. \tag{E.41}$$

Substituting Eq. E.41 back into Eq. E.34 and factorizing out the L matrix and the perturbing forces produces

$$\left\{\begin{array}{c} u_1'' \\ \\ u_2'' \end{array}\right\} = -\frac{1}{4}\boldsymbol{u} + \frac{r}{2\mu\alpha}\left[\boldsymbol{I} + \frac{4}{r}\boldsymbol{u}'\boldsymbol{u}'^T\right] L^T(\boldsymbol{u})\boldsymbol{F}. \tag{E.42}$$

Finally, substituting α from Eq. E.32 into Eq. E.42 gives the following elegant two-dimensional equations of motion.

$$\left\{\begin{array}{c} u_1'' \\ \\ u_2'' \end{array}\right\} + \frac{1}{4}\boldsymbol{u} = \frac{r^2}{4\mu}\left[1 + \frac{4}{r}\boldsymbol{u}'^T\boldsymbol{u}'\right]\left[\boldsymbol{I} + \frac{4}{r}\boldsymbol{u}'\boldsymbol{u}'^T\right] L^T(\boldsymbol{u})\boldsymbol{F}. \tag{E.43}$$

These equations for the two-dimensional case are easily extended into four dimensions by including the u_3 and u_4 components into the $\boldsymbol{u}$ vector. The resulting equations are the same as those shown in Eq. 5.11 from Chapter 5.