ORTHOGONAL POLYNOMIAL APPROXIMATION IN HIGHER DIMENSIONS: APPLICATIONS IN ASTRODYNAMICS

A Dissertation

by

AHMAD HANI ABD ALQADER BANI YOUNES

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

DOCTOR OF PHILOSOPHY

August 2013

Major Subject: Aerospace Engineering

ABSTRACT

We propose novel methods to utilize orthogonal polynomial approximation in higher dimension spaces, which enable us to modify classical differential equation solvers to perform high precision, long-term orbit propagation. These methods have immediate application to efficient propagation of catalogs of Resident Space Objects (RSOs) and improved accounting for the uncertainty in the ephemeris of these objects. More fundamentally, the methodology promises to be of broad utility in solving initial and two point boundary value problems from a wide class of mathematical representations of problems arising in engineering, optimal control, physical sciences and applied mathematics. We unify and extend classical results from function approximation theory and consider their utility in astrodynamics. Least square approximation, using the classical Chebyshev polynomials as basis functions, is reviewed for discrete samples of the to-be-approximated function. We extend the orthogonal approximation ideas to n-dimensions in a novel way, through the use of array algebra and Kronecker operations. Approximation of test functions illustrates the resulting algorithms and provides insight into the errors of approximation, as well as the associated errors arising when the approximations are differentiated or integrated. Two sets of applications are considered that are challenges in astrodynamics. The first application addresses local approximation of high degree and order geopotential models, replacing the global spherical harmonic series by a family of locally precise orthogonal polynomial approximations for efficient computation. A method is introduced which adapts the approximation degree radially, compatible with the truth that the highest degree approximations (to ensure maximum acceleration error $< 10^{-9} ms^{-2}$, globally) are required near the Earths surface, whereas lower degree approximations are required as radius increases. We show that a four order of magnitude speedup is feasible, with both speed and storage efficiency optimized using radial adaptation. The second class of problems addressed includes orbit propagation and solution of associated boundary value problems. The successive Chebyshev-Picard path approximation method is shown well-suited to solving these problems with over an order of magnitude speedup relative to known methods. Furthermore, the approach is parallel-structured so that it is suited for parallel implementation and further speedups. Used in conjunction with orthogonal Finite Element Model (FEM) gravity approximations, the Chebyshev-Picard path approximation enables truly revolutionary speedups in orbit propagation without accuracy loss.

This work is dedicated to my wife *Laiali* and sons *Yamen, Amro* and *Motaz* whose sacrifices, which were realized by our loss of precious time together, were for me the most painful and humbling of all

ACKNOWLEDGEMENTS

God...All that is Holy and Divine, thank YOU for giving me the faith and power to overcome the obstacles I faced during my graduate studies.

Dr. John L. Junkins, I thank you for giving me the opportunity to work with someone as inspiring as yourself on a project as enriching as this. I thank you for your continuous support and faith in me, for your passion, leadership and guidance – for giving me wings to fly!

Thank you, Dr. James D. Turner, for your enormous contributions which had a large impact on my work, for showing insights from your vast experience and research in the field.

Thank you, Dr. Daniele Mortari, for introducing me to the attitude and attitude error world.

Thank you Dr. Rodney D. W. Bowersox for using your leadership and authority in allowing us to use whatever department utility required making my work possible. I also give thanks to the sweetest persons in the department, Karen Knabe and Lisa Jordan; your admisnitrative support has been superb. You were so patient with me and always there to assist me whenever I needed anything. I will cherish that forever.

Thank you Dr. Srinivas Rao Vadali and Dr. Shankar Bhattacharyya for serving on my doctoral committee and for teaching me about dynamics and control.

This work was conducted under the Air Force Office of Scientific Research Contract FA9550-11-1-0279; the support of our program manager Kent Miller is gratefully acknowledged. We are also pleased to acknowledge useful discussions with Ryan

Russell with regard to gravity approximation and for sharing the GRACE spherical harmonic coefficients.

Many thanks to both (LASR) and (MCPI) mafia at Texas A&M University. I have been so blessed to work with such a talented team of friends.

Gratitude from the very bottom of my heart and soul to my amazing and supportive parents, who encouraged, supported and fueled my work. I love you so much, more than you can ever imagine, and more than I can ever demonstrate.

Gratitude to the invisible force that was behind all my triumphs, to the miracle that inspired patience over hardship, to Laiali, I am grateful to you for being the wife that you are, standing by me every step of the way, brightening every dark alley I ended up lost in. Thank you for being so special and for making my life so special. You sure are my four leaf clover; hard to find and lucky to have. Also, I am grateful to you, my dear sons Yamen, Amro and Motaz, for being my deep soul and with my utmost heart felt passion. I deeply thank you, my dear brothers (Mohammad, Abdel Qader, Ammar) and my dear sisters (Nisreen, Shereen, Amani, Enas, and Areen); you are so close.

Last but not least, I would like to gratefully acknowledge the support of all my friends who made the past semesters tolerable during the worst days and both exciting and fun during the best days. I believe that my friends are quiet angels, who lift me to my feet when my wings have trouble remembering how to fly!

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

Efficient, high precision orbit propagation has gained renewed impetus due to the rapidly escalating demands for improved Space Situational Awareness (SSA) and the challenges posed by the Kessler Syndrome, which hypothesizes that every collision of two space objects drastically increases the probability of subsequent collisions. Due to a number of factors, space object catalogs presently contain about 20,000 debris objects ($> 10\ cm$) and will almost certainly increase in the near future, unless effective mitigation efforts are undertaken in the near future. With reference to Figure I.1, and drawing insights from references [1–16], approximately 20,000 objects are presently trackable in Earth orbit using existing sensor systems and this number has escalated rapidly in recent years. Since Figure I.1 was published in 2010, debris tracking has significantly increased the size of the catalog by 30% for objects larger than 10 cm, and to an estimated 500,000 objects greater than 1 cm (smaller than 10 cm cannot be reliably tracked by conventional means. Much of the recent growth is attributed to improved accounting for the debris resulting from the 2007, 2009 Fengyun-1C and Cosmos/Iridium collisions). Conjunction analysis, probability of collision analysis, orbit prediction, and orbit determination require substantial computing resources with thousands of CPU hours presently consumed per week, these computations accelerate at a rate proportional to the number of objects raised to a power greater than 2, therefore the Kessler prediction of exponential growth of the object catalog implies a grand computational challenge. It is mentioned that the competing effects of drag decay and new debris resulting from cascading collisions pose an unsolved problem and the rate of growth implicit in the Kessler Syndrome cannot at present be predicted with confidence.

One key issue near the heart of the challenge is the time required to precisely propagate each object's orbit. Existing methods for solving these problems have not been very successful in exploiting parallel computer architectures.

There exists a fairly small set of "well accepted" for solving the differential equations of celestial mechanics, and the integration methods implemented on parallel machines are only modified versions of the well-proven traditional integration approaches. The conventual methods are robust and stable, but are typically poorly suited for parallelization. Numerical methods for propagating orbits is considered by most to be a rather mature field, with a still widely used method dating from the work of K.F. Guass in the mid-1800s. It is therefore perhaps a surprise that one order of magnitude speedup is possible, relative to the conventional algorithms. However, this is indeed the case, and these methods are one main focus of this dissertation.

This dissertation addresses the following four issues:

1. local approximation of high degree and order geopotential models, replacing the global spherical harmonic series by a family of locally precise orthogonal polynomial approximations for efficient computation,

2. extension of current research results to develop refined methods for efficient orbit propagation, with emphasis on methods that are easily parallized and with systematic error characterization and development of adaptive methods to automate the control of precision and efficiency,

3. evaluation of the new methods with regard to precision, stability, and efficiency, in comparison to existing methods for solution of initial and two point boundary value problems, and

4. implementations of the concepts and algorithms in a parallel computing architecture.

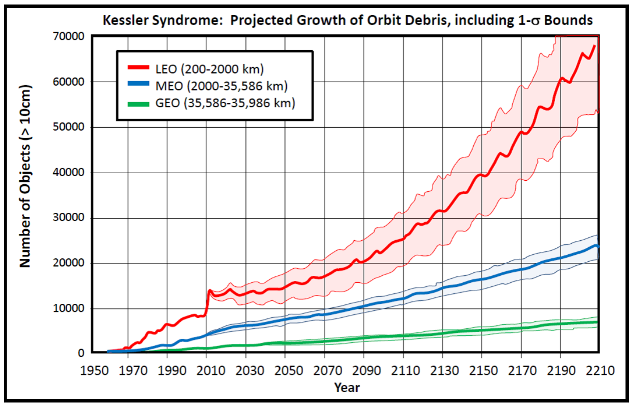


Figure I.1: Estimated Growth of Objects in Earth Orbit Larger Than 10cm (Without Mitigation).

The classical Picard Iteration method has been largely a museum piece, vis-á-vis computing spacecraft orbits, because it converts the usual system of nonlinear ordinary differential equations into a sequence of integrals with a requirement for iterative numerical quadrature in lieu of utilizing the large family of single and/or multi-step methods that have evolved, and have been used successfully for over a century, for numerically solving the corresponding nonlinear system of differential equations. A recent dissertation by Xiaoli Bai [17] presents a novel fusion of orthogonal polynomial approximation and linear algebra developments with Picard

Iteration, called the Modified Chebyshev Picard Iteration (MCPI). MCPI refines a high order orthogonal function approximation of the entire state trajectory over a substantial time span, in contrast to traditional, step-wise integration methods. Bai's version of MCPI is closely related to the historical works of Feagin [18, 19], Shaver [20], and Clenshaw and Norton [21]. The key is that MCPI is well suited to parallelization, whereas the traditional differential equation solvers are poorly suited for parallelization, we show that computation of force functions along each path iteration can be rigorously distributed over many parallel cores with negligible cross communication needed, and this truth opens the door to extremely attractive parallel computing means for propagation of large numbers of high precision orbits over long time intervals. Extensions of the approach to solve the two-point boundary value problems of optimal control, and the preliminary results obtained by Bai et al. [17, 22] are very encouraging. We also mention that research on efficient methods to accurately replace high order gravity fields by piecewise continuous interpolation a la refs [23–27] and in this dissertation indicate that a speedup of over several orders of magnitude in the computation time to compute a state-of-the art gravitational acceleration is possible. Since the gravitational acceleration computation dominates the total acceleration, this speedup, when combined with over one order of magnitude speed up of MCPI gives rise to an opportunity for revolutionary improvement in the efficiency of precision orbit propagation. The fundamental speedups are further enhanced since these efficiency gains are increased substantially as the methods are ideal for computational acceleration through massive parallelization.

I.A. Orthogonal Approximation

There are several treatments of discrete approximation using Chebyshev polynomials [17,24,27–33]. Among the more comprehensive of these are the texts [28,30]. In [27], orthogonal approximation is placed in a broader context of multi-resolution approximation via linear and nonlinear input/output maps. In Appendix A, we summarize a few most relevant aspects of approximation using Chebyshev polynomials that we utilize in this work.

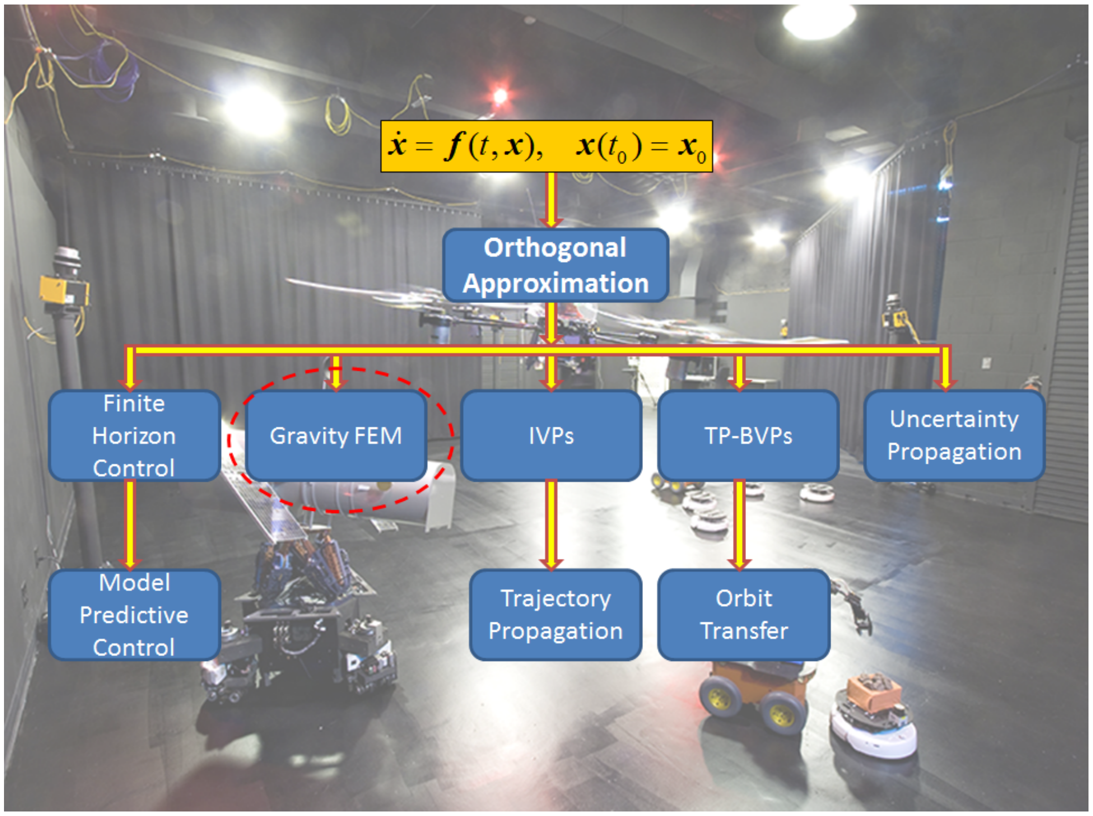


Figure I.2: Flowchart of Orthogonal Approximation Applications.

Figure I.2 shows some orthogonal approximation applications in astrodynamics.

Two sets of applications are considered in this dissertation that are fundamental challenges in astrodynamics. The first application establishes highly efficient local approximation of high degree and order geopotential models, replacing the global spherical harmonic series by a family of locally precise orthogonal polynomial approximations for efficient computation. A method is introduced which adapts the approximation degree radially, compatible with the truth that the highest degree approximations (to ensure, for example, maximum acceleration error $< 10^{-9}\ ms^{-2}$, globally, for high fidelity) are required near the Earth's surface, whereas lower degree approximations are required as the radius increases. The second class of problems is orbit propagation as well as solution of associated boundary value problems. The Chebyshev-Picard path approximation method is shown to be well-suited for solving these problems with over an order of magnitude speedup relative to known methods in a serial processor. Furthermore, the approach is parallel-structured so that it is ideally suited for parallel implementation and further speedups. Used in conjunction with orthogonal Finite Element Model (FEM) gravity approximations, the Chebyshev-Picard path approximation enables truly revolutionary speedups in orbit propagation without accuracy loss.

We first review classical discrete polynomial approximation results for one and two dimensions and introduce a convenient array algebra means to extend the one dimensional orthogonality results to higher dimensions; this path avoids the curse of dimensionality and establishes the results needed for efficient computation. Several simple examples are provided to so that the efficacy and utility of the methodology can be appreciated heuristically. Secondly, we use these ideas to solve the problem of piecewise approximation of high degree and order gravitational potential models, for use in orbit propagation. Specifically, we replace the GRACE [34,35] (156, 156) spherical harmonic model by a global family of local orthogonal polynomial approximations. We also replace the (200, 200) Earth Gravitational Model EGM 2008 [36], using the same FEM orthogonal approximation approach to observe the dependence of the relative computational advantage on higher order gravity terms. Finally, we consider the impact of using the gravitational field approximation models on the efficiency of Chebyshev-Picard methods [17] for solving the corresponding initial value problems.

Interestingly and importantly, the basic methodology researched in this dissertation is of a fundamental nature, and thus of much broader utility than more efficient orbit prediction. It is anticipated these methods still find use wherever we need precise long time interval solutions of nonlinear initial and two point boundary value problems in applied mathematics, physics and engineering. We therefore recommend this approach and related methods be evaluated for solving other physically important problems that have a similar structure.

I.B. Finite Element Representations of the Geopotential

The classical solution to Laplace's equation for gravity is adopted using the globally valid spherical harmonic gravity potential model, where the spherical harmonic (SH) approach is slow and reveals the three main challenges [25,34,37–40]:

1. Choosing a finite upper limit of the series defines the accuracy (the more we know about gravity, the more terms are required and the more it costs to

compute acceleration)

2. Convergence is very inefficient and slow for $n > 2$, so, for the current state of the art, tens of thousands of terms are required to obtain a sufficiently high accuracy global gravity representation

3. The north and south poles represent non-free singularities of the usual spherical coordinates

In view of the slow convergence of global gravity models, we are motivated to truncate the classical expansion at $n = 2$ and introduce a finite element model (FEM) local gravity representation of the higher order perturbation in the anticipation that much lower degree locally valid functions can be used to efficiently model and compute local gravity perturbations. The literature on this subject was initiated with the classical developments [25] and has recently been explored by other others [41–43]. Applicable to both irregular and near-spherical shaped bodies, methods in this class expedite computations by effectively trading computer memory for runtime speed. First proposed by Junkins in 1976 [25], geopotential FEM interpolation methods have been bolstered recently by the extraordinary memory resources of common computers. A variety of approximation techniques and basis functions have been employed for gravity field representation, including weighting functions [23–26, 44], wavelets [45], splines [45, 46], octrees [47], psuedocenters [48] and 3D digital modeling [49]. Each interpolation method balances accuracy with efforts to minimize runtime speed and memory footprint cost while achieving exactness, continuity and smoothness as appropriate.

In the developments herein, we have solved a key historical challenge implicit in this class of methods for geopotential representation: How do we structure the FEM models to render them *radially adaptive* and efficient, so that the resulting algorithms "automatically know" traditional methods about the rapid radial decay of the high frequency terms and more to the point, which terms in the FEM representation to retain, as a function (mainly) of radial distance from geocenter. Addressing this issue herein, we have enabled a much improved efficiency.

I.C. Picard Iteration, Chebyshev Polynomials and Chebyshev-Picard Methods

During the 19th century Emile Picard, a French mathematician, introduced a classical successive path approximation method for solving differential equations of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad \boldsymbol{x}(t_0). \tag{1.1}$$

This can be rearranged without approximation to obtain the following integral equation

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau))\, d\tau. \tag{1.2}$$

Motivated by the exact integral equation form of Eq. (1.2), Picard hypothesized a sequence of trajectory approximations (*Picard Iteration*) generated by

$$\boldsymbol{x}^{i}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}\big(\tau, \boldsymbol{x}^{i-1}(\tau)\big)\, d\tau, \quad i = 1, 2, ... \tag{1.3}$$

Picard also published formal Lipshitz conditions for convergence of this sequence to the solution of Eq. (1.1). The essence of his convergence theorem is that if the function $\boldsymbol{f}(t,\boldsymbol{x})$ and jacobian $[\partial\boldsymbol{f}(t,\boldsymbol{x})/\partial\boldsymbol{x}]$ are continuous and bounded over the finite region $\{\mid t-t_0 \mid \, < \delta, \| \, \boldsymbol{x}^0(t)-\boldsymbol{x}(t) \, \|_\infty < \triangle\}$, then a unique solution of Eq. (1.1) exists. The sequence of trajectories converges to the solution of Eq. (1.1) for some finite bounds $\{\delta, \triangle\}$ defining the finite region of convergence. Furthermore, under these the same conditions on $\boldsymbol{f}(t,\boldsymbol{x})$ and $[\partial\boldsymbol{f}(t,\boldsymbol{x})/\partial\boldsymbol{x}]$, [50–57] establish the conditions under which the Picard Iteration operator is a contraction mapping: the sequence converges to the unique solution if $t-t_0$ is smaller than δ, and the starting trajectory is in the region bounded by $\triangle$. The $(\delta,\triangle)$ bounds for guaranteed convergence are generally difficult to estimate without a computational investigation over the volume of state space where the starting approximations and the unique solution lie. Even difficult to compute, highly conservative bounds on $(\delta,\triangle)$ are of theoretical importance, but they may be of limited computational utility. While means for computing practical convergence bounds that are useful in general-purpose algorithms have proven elusive, as we will show for the case of a linear system where MCPI is implemented, convergence analysis leads to very interesting results and practical convergence insight. When an excellent starting trajectory approximation $\boldsymbol{x}^0(t)$ exists, then the convergent successive trajectories must be close neighbors, and the general nonlinear contraction mapping theory can be replaced approximately by the linear MCPI contraction mapping and convergence analysis. Furthermore, for those problems where a good starting approximation can be generated using prior approximate insight, Picard Iteration is obviously accelerated.

Apparently, and importantly, prior to the work of [17], it was not known that the time interval over which convergence of Picard Iteration is achieved for computing satellite trajectories in Earth orbit approaches $20,000\,\mathrm{s}$. This is well over three periods of a typical LEO orbit. Since all known numerical integrators for high precision solutions cannot take steps of more than a small fraction of an orbit [17], the feasibility of high precision multiple orbit solution arcs via Picard Iteration is surprising (in view of the long legacy of methods developed and implemented for orbit integration). Evidently this is a game-changing truth for many problems of modern interest. The classical Picard Iteration, with regard to orbit integration, has until recently been a "museum piece" of limited computational utility. This is due to the fact that it implicitly trades the well-established family of methods for a numerical solution of nonlinear differential equations with an apparently less well-understood set of methods for resolving the actual convergence domain of the Picard sequence. This of course introduces the need for high precision, efficient and reliable methods to carry out the numerical quadratures of Eq. (1.3).

Several researchers over the past half-century have pursued the goal of rendering Picard Iteration a more practical approach for computing solutions of Eq. (1.1). Some degree of success has been achieved. For example, Parker and Sochacki have studied the use of Picard Iteration to generate solutions of IVPs in the form of a family of local Taylor series [58]. However, convergence of these series is not generally attractive compared with the methods we present below.

Bringing together approximation theory with Picard Iteration has proven a key to recent progress, and the use of orthogonal polynomials as basis functions to approximate both the trajectories $\boldsymbol{x}^i(t)$ and the integrand of Eq. (1.3) along each iterative trajectory have proven to be important steps to achieve both precision and efficiency. Chebyshev polynomials are but one set of orthogonal functions that might be used in an analogous way, however this choice has been widely adopted for function approximation and they have been found especially attractive to approximate the trajectories and integrands in Picard Iteration. When the zeros of Chebyshev polynomials are used as the nodes for polynomial interpolation, the resulting approximation has been shown to minimize the Runge's phenomenon and provide the best approximation under the minimax norm [17]. Many researchers have contributed to the research on using Chebyshev polynomials to solve IVPs and BVPs, but typically not adopting Picard Iteration [25, 31, 37] as the basis for the solution process. Note that the most straightforward approach of parameterizing the trajectory in terms of basis functions leads to a nonlinear programming problem if the trajectory $\boldsymbol{x}(t)$ is expanded in a linear combination of basis functions and substituted into $\boldsymbol{f}(t, \boldsymbol{x}(t))$. Several traditional methods introduce collocation nodes in order to obtain a sufficient number of nonlinear equations to iteratively determine the basis function amplitudes. However, using nonlinear programming to find the required large number of basis function coefficients, while employing this approach, has proven computationally inefficient for higher dimensioned state spaces. In addition, the curse of dimensionality and associated numerical difficulties frequently limits practical convergence. Both these assessments agree with the discussion by Vlassenbroeck and Dooren [59]. Therefore this approach is not considered a viable competitor to traditional existing methods such as the high order Runge-Kutta or multi-step methods for precisely solving problems in celestial mechanics.

As is evident from the literature, with a few exceptions, the classical Picard

Iteration has been realized in very few algorithms for solving IVPs. An important advance that changed the flow of research in this area was made by Clenshaw and Norton [21]. They first proposed solving IVPs and BVPs using both Picard iteration and Chebyshev polynomials (Chebyshev-Picard methods). Their new method approximated both the trajectory *and the integrand along each trajectory* by the same set of discrete Chebyshev polynomials. The basis functions were integrated term-by-term to establish a recursive trajectory approximation technique *that inherently contained the new basis function coefficients* <u>*linearly*</u> *on each iteration, without Taylor-series linearization.* Using Picard Iteration with no necessity for nonlinear programming turns out to allow solution of high order differential equations and thus enables applications to a large class of nonlinear dynamical systems. Their keystone contribution indicates the Chebyshev polynomials, when linearly approximating the integrand along the $(i-1)^{th}$ Picard iterate prior to integration, result in both efficient and accurate approximation of the integrals needed in the Picard Iteration. Notice this process completely avoids differentiation of approximations, and associated precision loss, in contrast to most collocation methods. The feasibility of parallel computation using the Chebyshev-Picard methods has also been studied by Feagin [18, 19], Shaver [20] and Fukushima [60, 61]. Feagin [19] presented a vector-matrix form of the Chebyshev-Picard method that is closely related to MCPI methods refined in [17,22,62]. Shaver [20] did a significant numerical study and made the first serious study of parallel computation with a variant of MCPI. Fukushima implemented a Chebyshev-Picard algorithm on a vector computer [60]. However, for one example problem, the vector code of Fukushima was shown to be slower than the scalar code, thus leading the author to surprisingly conclude that his vectorized code led to additional overhead and an inefficient implementation. In summary, important historical contributions that fused Picard Iteration with approximation theory were made by Clenshaw and Norton [21], Shaver [20], and Feagin [18,19], and Bai [17,22,62]. Our proposed approach builds directly on these important historical formulations. Prior to discussing these developments in detail, we address the state of the art for solving the differential equations of orbit mechanics.

I.D. State of the Art Methods for Numerical Solution of Satellite Orbits: RKN12(10)

We focus on three important sets of competing methods that have been broadly adopted for modern orbit integration. Most of the numerical methods presently in routine use for solving the IVPs can be categorized as either Runge-Kutta type single-step methods, multi-step extrapolation methods, or Taylor series (analytical continuation) methods. All of these methods owe their heritage to Euler's original (late 1700's) first order analytical continuation method and/or Gauss' (mid 1800s) predictor-corrector method. In essence, given a point $\boldsymbol{x}(t_k)$ on the trajectory, the entire family of currently used single step Runge-Kutta methods seek to approximately replace an n^{th} order Taylor series (analytical continuation) to within $O(h^{n+1})$ without the necessity of taking higher derivatives of $\boldsymbol{f}(t, \boldsymbol{x})$. This is accomplished by linearly combining a set of neighboring local evaluations of $\boldsymbol{f}(t, \boldsymbol{x})$ in such a way that produces an approximation of $\boldsymbol{x}(t_k + h)$, which can be shown to match a Taylor series with an error of $\sim O(h^{n+1})$. The currently used multi-step predictor-corrector methods are all descendants of Gauss' original "second sum method" which utilize extrapolations derived from the calculus of finite differences to construct two or more iterative extrapolation formulas based on high order differences and sums formed from a table of immediately previous values of $\boldsymbol{x}(t)$ and $\boldsymbol{f}(t, \boldsymbol{x})$. The most commonly used predictor-corrector method for orbit computation is essentially identical to Gauss' method, but widely known as the Gauss-Jackson [34] algorithm. As discussed in [17], the high order Runge-Kutta methods such as RKN12(10) utilize two orders to implement automatic step size control, (e.g., 10^{th} and 12^{th} order). A detailed discussion is presented in [17]. These step-by-step methods are less efficient for low eccentricity orbits, but are superior for moderate to high eccentricity orbits when compared to the Gauss-Jackson integrator. Until the early 1960's most of the work on Runge-Kutta methods focused on explicit methods which are unsuitable for the solution of stiff equations, but Butcher [63] introduced a framework to study the Implicit Runge-Kutta (IRK) methods, which are robust, adaptive and stable but frequently computationally expensive. The IRK methods require solving a system of algebraic equations at every step, which increases the computational cost considerably. Some of these and other methods are presented in [64–68]. No existing step-by-step method is well-suited for massive parallelization. We will see later that the Chebyshev-Picard Methods are ideally suited to parallelization.

CHAPTER II

ORTHOGONAL APPROXIMATION

II.A. Introduction

We unify and extend classical results from function approximation theory and consider their utility in astrodynamics. Least square approximation, using the classical Chebyshev polynomials as basis functions, is reviewed for discrete samples of the to-be-approximated function. We extend the orthogonal approximation ideas to n-dimensions in a novel way, through the use of array algebra and Kronecker operations. Approximation of test functions illustrates the resulting algorithms and provides insight into the errors of approximation, as well as the associated errors arising when the approximations are differentiated or integrated. We first review classical discrete polynomial approximation results for one and two dimensions and introduce a convenient array algebra means to extend the one dimensional orthogonality results to higher dimensions. This path avoids the curse of dimensionality and establishes the results needed for efficient computation. Several simple examples are provided to enable the efficacy and utility of the methodology to be appreciated heuristically.

II.B. Orthogonal Approximation with One Independent Variable

There are several treatments of discrete approximation using Chebyshev polynomials [17,24,27–33]. Among the more comprehensive of these are the texts [28,30].

In [27], orthogonal approximation is placed in a broader context of multi-resolution approximation via linear and nonlinear input/output maps. In Appendix A, we summarize a few most relevant aspects of approximation using Chebyshev polynomials that we utilize in this dissertation.

Let us first set the context by considering the approximation of a single-valued function of one independent variable, x:

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

To put the problem in a non-dimensional framework, we first introduce a new independent variable ξ such that $\{-1 \leq \xi \leq 1\}$. It is easy to verify the forward and inverse transformations:

$$\begin{gathered} \xi(x) = 2\left(x - x_{\min}\right) / \left(x_{\max} - x_{\min}\right) - 1, \\ \text{and,} \\ x(\xi) = x_{\min} + (\xi + 1)\left(x_{\max} - x_{\min}\right)/2. \end{gathered} \tag{2.2}$$

Substituting the second of Eqs (2.2) into Eq. (2.1), we wish to approximate the function

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

In the case of general basis functions $\phi_n(\xi)$ with $\xi \in \{-1, 1\}$, we seek to approximate $f(\xi)$ as a linear combination of a prescribed set of $N + 1$ linearly independent basis functions $\{\phi_0(\xi), \phi_1(\xi), ..., \phi_N(\xi)\}$ as

$$f(\xi) \approx \sum_{n=0}^{N} a_n \phi_n(\xi). \tag{2.4}$$

For the case of discrete measurement samples, we introduce a set of sample points (nodes) as $\{\xi_0, \xi_1, ..., \xi_M; M \geq N\}$; the residual approximation error at each measurement node is

$$r_j = f(\xi_j) - \sum_{n=0}^{N} a_n \phi_n(\xi_j);\ j = 0, 1, ..., M, \tag{2.5}$$

or in vector-matrix notation

$$\boldsymbol{r} = \boldsymbol{f} - \Phi \boldsymbol{a}, \tag{2.6}$$

where

$$\boldsymbol{f} = \begin{bmatrix} f(\xi_0) \\ f(\xi_1) \\ \vdots \\ f(\xi_M) \end{bmatrix},\ \Phi = \begin{bmatrix} \phi_0(\xi_0) & \phi_1(\xi_0) & \cdots & \phi_N(\xi_0) \\ \phi_0(\xi_1) & \phi_1(\xi_1) & \cdots & \phi_N(\xi_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\xi_M) & \phi_1(\xi_M) & \cdots & \phi_N(\xi_M) \end{bmatrix},\ \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}. \tag{2.7}$$

The method of least squares seeks the coefficient vector ($\boldsymbol{a}$) that minimizes the weighted sum square of the residuals

$$J = \frac{1}{2} (\boldsymbol{f} - \Phi \boldsymbol{a})^T W (\boldsymbol{f} - \Phi \boldsymbol{a})\,;\ W = W^T \text{ (positive definite weight matrix)}. \tag{2.8}$$

It follows [69] that the least square minimization solution for $\boldsymbol{a}$ leads to the *normal equations*

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

Restricting W to be diagonal hereinafter, and choosing a special class of *orthogonal basis functions*, $\Phi^T W \Phi$ can be rendered a diagonal matrix so the matrix inverse in Eq. (2.9) is trivial. So for the orthogonal basis function we obtain

$$\left(\Phi^T W \Phi\right)^{-1} = diag \left\{1/\left(\Phi^T W \Phi\right)_{ii}\right\} \triangleq diag \left\{ \begin{matrix} 1/m_{00} & 1/m_{11} & \cdots & 1/m_{NN} \end{matrix} \right\}. \tag{2.10}$$

The typical element of $\Phi^T W \Phi$ is a *discrete inner product* denoted $m_{\alpha\beta} = m_{\beta\alpha}$ and invoking the requirement 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:

$$m_{\alpha\beta} = m_{\beta\alpha} \triangleq \langle \phi_\alpha(\xi), \phi_\beta(\xi) \rangle \equiv \sum_{j=0}^{M} W_j \phi_\alpha(\xi_j) \phi_\beta(\xi_j) = \left\{ \begin{array}{c} 0,\ for\ \alpha \neq \beta \\ \\ m_{\alpha\alpha} = c_\alpha > 0,\ for\ \alpha = \beta \end{array} \right\}. \tag{2.11}$$

The orthogonality conditions depend jointly on the set of basis functions, the set of node locations and the weight matrix (more generally, $W = W^T$ may be fully populated).

For the case that the above orthogonality conditions are satisfied, the explicit solution for the coefficients of Eq. (2.9) is given by the independent (uncoupled) ratios of inner products as

$$a_\alpha = \frac{\langle \phi_\alpha(\xi), f(\xi) \rangle}{\langle \phi_\alpha(\xi), \phi_\alpha(\xi) \rangle} \equiv \frac{\sum_{j=0}^{M} W_j \phi_\alpha(\xi_j) f(\xi_j)}{\sum_{j=0}^{M} W_j \phi_\alpha^2(\xi_j)} \equiv \frac{1}{c_\alpha} \sum_{j=0}^{M} W_j \phi_\alpha(\xi_j) f(\xi_j), \text{for } \alpha = 0, 1, 2, ..., N. \tag{2.12}$$

An important special case arises when we make a specific choice of orthogonal basis functions, namely $\{\phi_0(\xi), \phi_1(\xi), ..., \phi_N(\xi)\} = \{T_0(\xi), T_1(\xi), ..., T_N(\xi)\}$, i.e., we choose the classical Chebyshev polynomials $\{T_0(\xi), T_1(\xi), ..., T_N(\xi)\}$, as discussed in references [17, 27–30] and the Appendix A as the basis functions. We also choose the $N + 1$ *cosine sample points* (also known [17, 27–30] as the CGL nodes in honor of Chebyshev-Gauss-Lobatto):

$$\xi_j = -\cos(j\pi/M),\ j = 0, 1, 2, ..., M. \tag{2.13}$$

Consistent with the classical orthogonality conditions for Chebyshev polynomials, we adopt the weight matrix $W = diag\left\{\frac{1}{2}, 1, 1, ..., 1, 1, \frac{1}{2}\right\}$. Upon substituting the sample points of Eq. (2.13) and the chosen weight matrix, it is easy to verify that orthogonality conditions of Eqs (2.11) are satisfied and the least square coefficients of Eqs (2.12) are specifically

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

$$= \frac{1}{c_\alpha}\left\{\frac{1}{2}T_\alpha(\xi_0)f(\xi_0) + ... + T_\alpha(\xi_{M-1})f(\xi_{M-1}) + \frac{1}{2}T_\alpha(\xi_M)f(\xi_M)\right\}, \tag{2.14}$$

where the denominators c_α in Eq. (2.14) are the positive constants

$$c_\alpha = \sum_{j=0}^{M} W_j T_\alpha^2(\xi_j) = \left\{\frac{1}{2}T_\alpha^2(\xi_0) + T_\alpha^2(\xi_1) + ... + T_\alpha^2(\xi_{M-1}) + \frac{1}{2}T_\alpha^2(\xi_M)\right\}, \alpha = 0, 1, ..., N, \tag{2.15}$$

More explicitly it can be verified that the denominator inner products reduce to

$$\left.\begin{array}{lllll} c_0 & = & \langle T_0(\xi), T_0(\xi)\rangle & = & M \\ c_\alpha & = & \langle T_\alpha(\xi), T_\alpha(\xi)\rangle & = & M/2, \qquad \alpha = 1, 2, ..., N-1 \\ c_N & = & \langle T_N(\xi), T_N(\xi)\rangle & = & M, \quad \text{if } M = N \text{ (interpolation case)} \\ c_N & = & \langle T_N(\xi), T_N(\xi)\rangle & = & M/2, \quad \text{if } M > N \text{ (least squares case)} \end{array}\right\}. \tag{2.16}$$

Thus the final coefficients for least square approximation are computed directly from the discrete inner products of Eq. (2.14) as

$$\left.\begin{array}{lll}\alpha_0 &= \frac{\langle T_0(\xi), f(\xi)\rangle}{\langle T_0(\xi), T_0(\xi)\rangle} = \frac{1}{M}\left\{\frac{1}{2}T_0(\xi_0)f(\xi_0)+\ldots+T_0(\xi_{M-1})f(\xi_{M-1})+\frac{1}{2}T_0(\xi_M)f(\xi_M)\right\} & \\ \alpha_\alpha &= \frac{\langle T_\alpha(\xi), f(\xi)\rangle}{\langle T_\alpha(\xi), T_\alpha(\xi)\rangle} = \frac{2}{M}\left\{\frac{1}{2}T_\alpha(\xi_0)f(\xi_0)+\ldots+T_\alpha(\xi_{M-1})f(\xi_{M-1})+\frac{1}{2}T_\alpha(\xi_M)f(\xi_M)\right\}, & \alpha=1,2,\ldots,N-1 \\ \alpha_N &= \frac{\langle T_N(\xi), f(\xi)\rangle}{\langle T_N(\xi), T_N(\xi)\rangle} = \frac{1}{c_N}\left\{\frac{1}{2}T_N(\xi_0)f(\xi_0)+\ldots+T_N(\xi_{M-1})f(\xi_{M-1})+\frac{1}{2}T_N(\xi_M)f(\xi_M)\right\}, & \left\{\begin{array}{l} c_N=M,\ M=N \\ c_N=\frac{M}{2},\ M>N \end{array}\right. \end{array}\right\}. \tag{2.17}$$

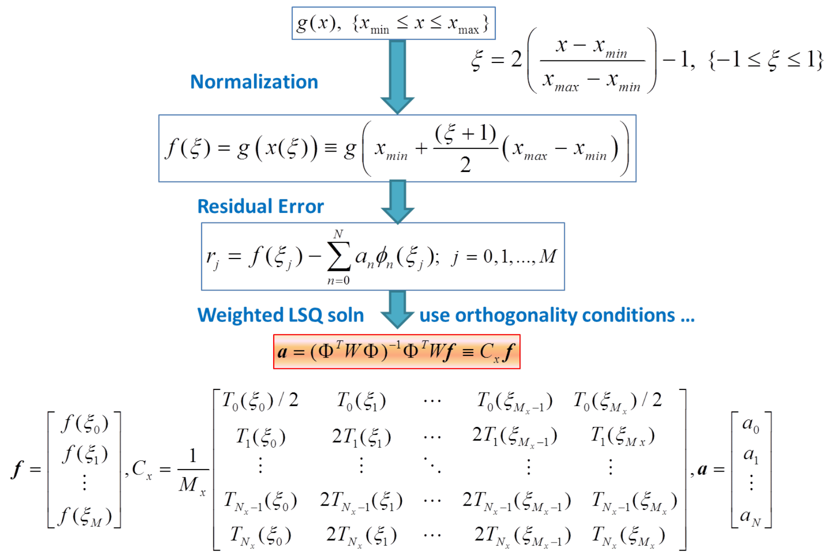
Note that the coefficients of Eq. (2.17) are computed independently of each other, and the absolute value of each coefficient is the maximum contribution of that term – this enables convenient means for obtaining efficient and accurate truncated approximations, as well as insight for adapting the order of the approximation. If a vector-matrix form is desired for the least squares solution for the coefficients, we can rearrange Eqs (2.17) in the form

$$\boldsymbol{a} = C\boldsymbol{f}, \tag{2.18}$$

where the Chebyshev least square operator matrix is simply

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

The flowchart explaining the 1-D approximation is shown in Figure II.1. We mention that the cosine nodes of Eq. (2.13) locate all $N-1$ extrema of the Chebyshev



$g(x),\ \{x_{\min} \le x \le x_{\max}\}$
Normalization
$\xi = 2\left(\frac{x - x_{min}}{x_{max} - x_{min}}\right) - 1,\ \{-1 \le \xi \le 1\}$
$f(\xi) = g(x(\xi)) \equiv g\left(x_{min} + \frac{(\xi+1)}{2}(x_{max} - x_{min})\right)$
Residual Error
$r_j = f(\xi_j) - \sum_{n=0}^{N} a_n \phi_n(\xi_j);\ j = 0, 1, \ldots, M$
Weighted LSQ soln
use orthogonality conditions ...
$\boldsymbol{a} = (\Phi^T W \Phi)^{-1} \Phi^T W \boldsymbol{f} \equiv C_x \boldsymbol{f}$
$\boldsymbol{f} = \begin{bmatrix} f(\xi_0) \\ f(\xi_1) \\ \vdots \\ f(\xi_M) \end{bmatrix}, C_x = \frac{1}{M_x} \begin{bmatrix} T_0(\xi_0)/2 & T_0(\xi_1) & \cdots & T_0(\xi_{M_x-1}) & T_0(\xi_{M_x})/2 \\ T_1(\xi_0) & 2T_1(\xi_1) & \cdots & 2T_1(\xi_{M_x-1}) & T_1(\xi_{Mx}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ T_{N_x-1}(\xi_0) & 2T_{N_x-1}(\xi_1) & \cdots & 2T_{N_x-1}(\xi_{M_x-1}) & T_{N_x-1}(\xi_{M_x}) \\ T_{N_x}(\xi_0) & 2T_{N_x}(\xi_1) & \cdots & 2T_{N_x}(\xi_{M_x-1}) & T_{N_x}(\xi_{M_x}) \end{bmatrix}, \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}$

Figure II.1: 1-D Approximation.

polynomials, as well as the two end points of the approximation interval. Also note the extrema of these polynomials, and therefore the sample points, cluster near the ± 1 boundaries as the degree N of the approximation increases. The first six $T_\alpha(\xi)$ are graphed in Figure A.1 in Appendix A. Note that the particular weight matrix $W = diag\left\{\frac{1}{2}, 1, 1, ..., 1, 1, \frac{1}{2}\right\}$ can be shown to be consistent with the classical Chebyshev polynomials satisfying the orthogonality conditions of Eq (2.11). The choice of an identity matrix, for example, together with the Gramm-Schmidt process [27], gives rise to a related set of orthogonal polynomials. The approximation properties of the Chebyshev polynomials are well-researched and a substantial literature exists related to this choice, therefore we adopt the slight modification of the identity weight matrix. Observe the unit weights apply to all interior maxima and minima, whereas the $\frac{1}{2}$ weights apply to the two boundary points. We also mention that Bai's recent dissertation and some of the related historical literature are consistent with the above, but care must be taken in reading these references due to a factor of $\frac{1}{2}$ applied to the zeroth and/or the N^{th} terms of the summations, depending on whether $M > N$ or $M = N$. In particular, the second of Eqs (2.17) is frequently used [70] to compute all $N + 1$ a_α's, and this is then compensated by the introducing a $\frac{1}{2}$ factor into the zeroth and N^{th} terms of Eq. (2.4). The somewhat unusual "sigma prime and sigma double prime" inner product notations in the literature are eliminated by the notations above. While the competing conventions and inner product definitions (which implicitly incorporate the weights and c_α terms) are not wrong, we believe the above formulation leads to a logical path to generalize the classical weighted least square formulations to the analogous developments for approximating functions of n variables, as we show below.

The first four Chebyshev polynomials and the three term recurrence relationship for arbitrary $T_k,\ k > 0$ are given by

$$\begin{aligned} T_0(\xi) &= 1, \\ T_1(\xi) &= \xi, \\ T_2(\xi) &= 2\xi^2 - 1, \\ T_3(\xi) &= 4\xi^3 - 3\xi, \\ &\vdots \\ T_{k+1}(\xi) &= 2\xi T_k(\xi) - T_{k-1}(\xi). \end{aligned} \tag{2.20}$$

The first few Chebyshev polynomials are plotted in Appendix A. As M and N become large, the Chebyshev Polynomials constitute a complete set of basis functions, and therefore, theoretically, a linear combination of these basis functions can represent to arbitrary precision a continuous function $f(\xi)$ on the interval from $\{-1 \leq \xi \leq +1\}$, given a sufficiently high M and N in Eq. (2.4). Some functions “submit” to accurate approximation for a small M and N, and in some unusual cases, very large M and N are required to achieve a small approximation error. Note the absence of a matrix inverse allows great flexibility and efficiency, analogous to other orthogonal approximation techniques, such as Fourier series.

The above Chebyshev polynomial formulation is known to be relatively immune to the so-called *Runge Phenomena* wherein the approximation errors near the end of the data at ± 1 can become unacceptably large. The dense sampling near the ends of the approximation interval associated with Eq. (2.13) implicitly reduces errors near the boundary. Also, the fact that no numerical matrix inversion is required for orthogonal polynomials means that approximation can be robustly computed at any desired or required order. These advantages are best illustrated by numerical examples.

Prior to considering these examples, let us compare the location of the nodes (sample points) of Eq. (2.13), to the most elementary alternative of uniformly spaced samples given by $\xi_j = -1 + 2(i/M),\ i = 0, 1, 2, ..., M$. See Figure II.2 for the cases of $M = 2, 3, 4$, and 20 samples, showing the cosine sample point density of Eq. (2.13) versus the uniform sample density. Note the clustering near the ± 1 ends of the interval and the increasing sparseness as approaching zero (the center of the interval).

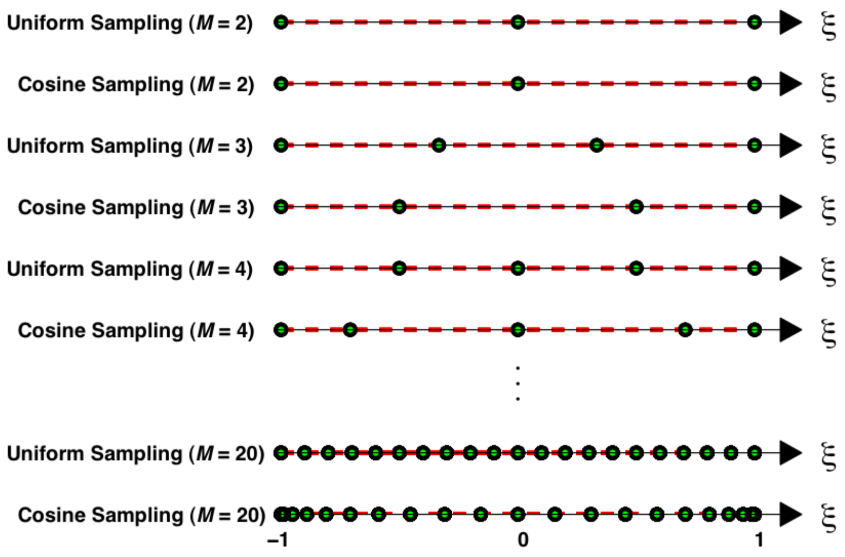

Uniform Sampling ($M = 2$)
Cosine Sampling ($M = 2$)
Uniform Sampling ($M = 3$)
Cosine Sampling ($M = 3$)
Uniform Sampling ($M = 4$)
Cosine Sampling ($M = 4$)
Uniform Sampling ($M = 20$)
Cosine Sampling ($M = 20$)
−1
0
1
ξ

Figure II.2: Cosine Nodes.

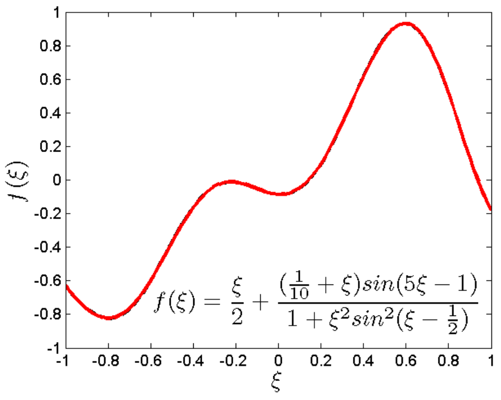
II.C. Numerical Examples for Illustrative Test Functions of One Variable

To appreciate the practical application and utility of the above developments, let us approximate the test function (Test Function 1)

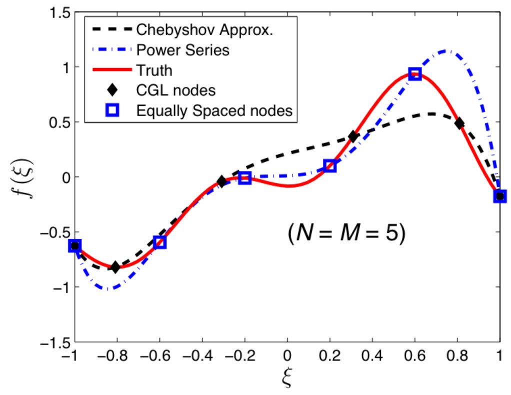
$$f(\xi) \equiv \frac{\xi}{2} + \frac{\left[\left(\frac{1}{10} + \xi\right)\sin\left(5\xi - 1\right)\right]}{\left[1 + \xi^2 \sin^2\left(\xi - \frac{1}{2}\right)\right]}, \tag{2.21}$$

and use Eq (2.13) to generate measurements with either $M = 300$ or $M = N$, with N swept. The true function is shown in Figure II.3a, and Figures II.3b to II.3d, along with Figure II.4, display several approximations. For reference, in addition to the orthogonal Chebyshev orthogonal approximation using the basis functions $\{T_0(\xi), T_1(\xi), ..., T_N(\xi)\}$ and the cosine nodes, we also show least square approximations with the power series polynomial basis functions $\left\{1, \xi, \xi^2, ..., \xi^N\right\}$ of the same (M, N) and uniform nodes.

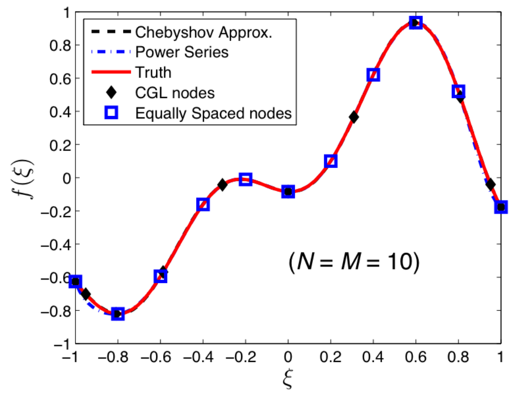
The Runge Phenomena is evident in Figure II.3d (note large boundary errors for the power series approximation versus the more uniform errors of the Chebyshev approximation). Furthermore, we see in Figure II.4, for the case of high degree approximation using Chebyshev polynomials, we approximate Test Function 1 with a residual error approaching a machine zero. All computations are performed using MATLAB® with ~ 16 digit floating point arithmetic.



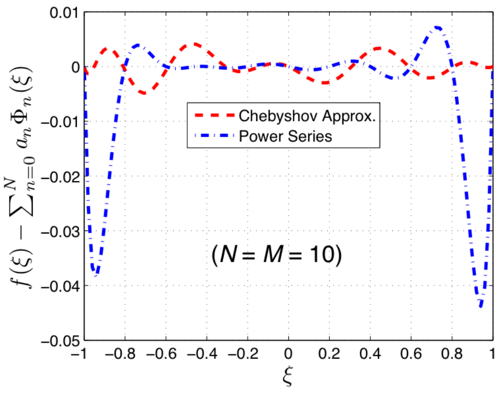
(a) Test Function 1.



(b) Approximations of Test Function 1.



(c) Approximations of Test Function 1.



(d) Approximations of Test Function 1.

Figure II.3: Test Function 1.

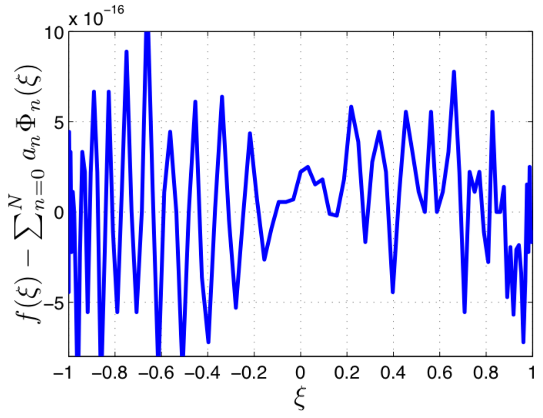
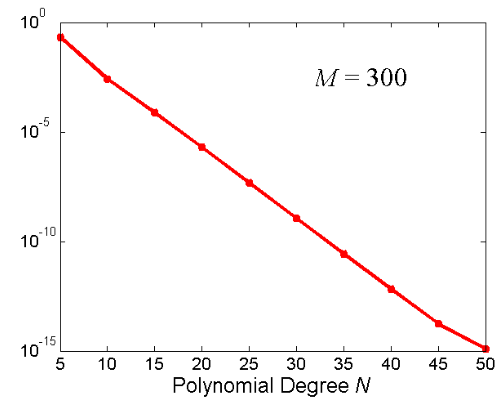


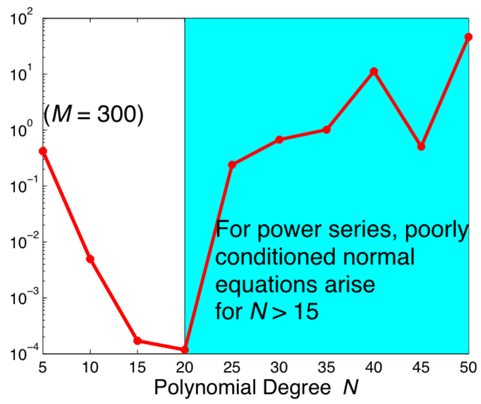
Figure II.4: Approximation Error of Test Function 1 ($N = 50, M = 300$).

Figure II.5a and II.5b show the maximum errors that result from least square approximation when $M = 300$ measurement nodes are used, for the case of the Chebyshev and power series polynomial approximation of Test Function 1. As is evident, convergence of the Chebyshev approximation again approaches machine precision by $N = 50$, with the maximum error decreasing about one order of magnitude when the degree N is increased by $\Delta N = 3$; the linear slope on a log-log scale permits insight on the N required for given accuracy. This behaviour shows that spectral accuracy is obtained, limited only by machine precision. On the other hand, the error slope versus N is much smaller with $N < 15$ for the power series case (due to the Runge Phenomena), and the power series can't be computed accurately for $N > 15$ due to poor conditioning of the normal Eqs (2.9), which must be inverted numerically since power series are non-orthogonal basis functions.

Other issues that frequently arise are the associated accuracy of: (i) differentiation or (ii) integration of the approximating polynomial. The approximation errors associated with these fundamental processes are of obvious importance in many en-



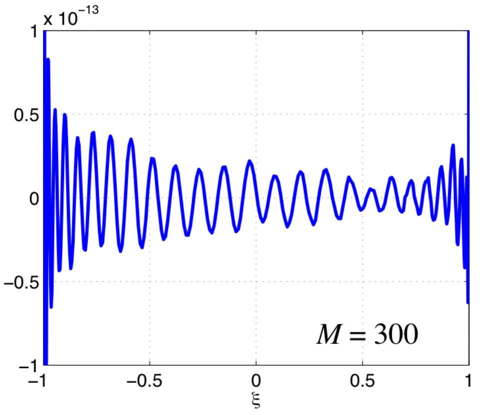
(a) Chebyshev Maximum Approximation Error.



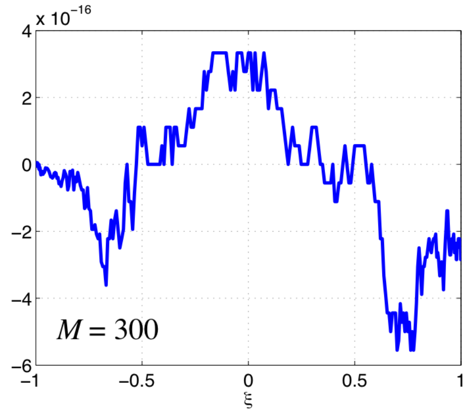
(b) Power Series Maximum Approximation Error.

Figure II.5: Chebyshev and Power Series Approximations.

gineering applications. Figure II.6a shows the differentiation errors (derivative of the approximation minus the analytical derivative of the true Test Function 1); it is evident that the derivative errors are two orders of magnitude larger (10^{-13} derivative approximation error, compared to 10^{-15} function approximation error). On the other hand, with reference to Figure II.6b, the definite integral (starting with a zero left boundary condition) of the approximation minus the true definite integral of Test Function 1, reveals that the integration errors of 10^{-16} are an order of magnitude *smaller* than the zero mean 10^{-15} errors in the underlying function approximation. As is well known, integration is a smoothing process and the zero mean oscillatory errors of least square approximation are averaged out to a degree (an order of magnitude in this case). On the other hand, differentiation invariably amplifies the function approximation error. The "take away message" is clear and important; whenever one has the option, it is better to qualitatively integrate than to differenti-



(a) Chebyshev Differentiation Errors ($N = 50$).



(b) Chebyshev Integration Errors ($N = 50$).

Figure II.6: Differentiation and Integration Errors.

ate, if high precision is sought. The spectral accuracy and integration/differentiation properties extent fully to approximation of multi dimensional functions as is evident in the study below.

While we do not show the corresponding results for power series approximation, the advantage lies with the Chebyshev approximation by several orders of magnitude for intermediate N, and of course, the ~ 16 digit precision approximation of Test Function 1 by power series is not computationally feasible unless one resorts to extended precision arithmetic.

II.D. Orthogonal Approximation: More Than One Variable

Let us consider the approximation of a function of two independent variables

$$g(x,y),\ \{x_{\min} \le x \le x_{\max}\},\ \{y_{\min} \le y \le y_{\max}\}, \tag{2.22}$$

$$\xi(x) = -1+2\left(x - x_{\min}\right) / \left(x_{\max} - x_{\min}\right), \text{and } \eta(y) = -1+2\left(y - y_{\min}\right) / \left(y_{\max} - y_{\min}\right), \tag{2.23}$$

$$x(\xi) = x_{\min} + (\xi + 1)\left(x_{\max} - x_{\min}\right)/2, \text{ and } y(\eta) = y_{\min} + (\eta + 1)\left(y_{\max} - y_{\min}\right)/2. \tag{2.24}$$

Substituting Eqs (2.24) into Eq (2.22), we see that we wish to approximate the function

$$\begin{aligned} f(\xi,\eta) &\triangleq g\left(x(\xi), y(\eta)\right) \\ &\equiv g\left(\underbrace{x_{\min} + (\xi+1)\left(x_{\max} - x_{\min}\right)/2,}_{x(\xi)}\underbrace{y_{\min} + (\eta+1)\left(y_{\max} - y_{\min}\right)/2}_{y(\eta)}\right). \end{aligned} \tag{2.25}$$

In the general case, we seek to approximate $f(\xi,\eta)$ as a linear combination of a prescribed set of linearly independent basis functions of two variables

$$\{\phi_{00}(\xi,\eta), \phi_{01}(\xi,\eta), ..., \phi_{NxNy}(\xi,\eta)\}$$

as

$$f(\xi,\eta) \equiv \sum_{\alpha=0}^{N_x}\sum_{\beta=0}^{N_y} a_{\alpha\beta}\phi_{\alpha\beta}(\xi,\eta). \tag{2.26}$$

For the case of discrete measurement samples, we introduce a set of sample points (nodes) as $\{\xi_0, \xi_1, ..., \xi_{M_x}; M_x \geqslant N_x\}$, $\left\{\eta_0, \eta_1, ..., \eta_{M_y}; M_y \geq N_y\right\}$. The residual approximation error at each measurement node is

$$r_{ij} = f(\xi_i, \eta_j) - \sum_{\alpha=0}^{N_x}\sum_{\beta=0}^{N_y} a_{\alpha\beta}\phi_{\alpha\beta}(\xi_i,\eta_j);\ \{i = 0, 1, ..., M_x; j = 0, 1, ..., M_y\}. \tag{2.27}$$

or in vector-matrix notation, $\boldsymbol{r} = \boldsymbol{f} - \Phi\boldsymbol{a}$, with:

$$\boldsymbol{f}^T = \Big[\ f(\xi_0,\eta_0) \quad f(\xi_0,\eta_1)\ \cdots\ f(\xi_0,\eta_{M_y})\ \vdots\ f(\xi_1,\eta_0) \quad f(\xi_1,\eta_1)\ \cdots\ f(\xi_1,\eta_{M_y})\ \vdots\ \cdots$$
$$\cdots \vdots\ f(\xi_{M_x},\eta_0) \quad f(\xi_{M_x},\eta_1)\ \cdots\ f(\xi_{M_x},\eta_{M_y})\ \Big]$$

$$\boldsymbol{a}^T = \Big[\ a_{00} \quad a_{01} \quad \cdots \quad a_{0N_y} \ \vdots\ a_{10} \quad a_{11} \quad \cdots \quad a_{1N_y} \ \vdots\ \cdots\ \vdots\ a_{N_x1} \quad a_{N_x2} \quad \cdots \quad a_{N_xN_y}\ \Big],$$

$$\Phi = \left[\begin{array}{ccccccc}
\phi_{00}(\xi_0, \eta_0) & \phi_{01}(\xi_0, \eta_0) & \cdots & \phi_{0N_y}(\xi_0, \eta_0) & \phi_{10}(\xi_0, \eta_0) & \phi_{11}(\xi_0, \eta_0) & \cdots \\
\phi_{00}(\xi_0, \eta_1) & \phi_{01}(\xi_0, \eta_1) & \cdots & \phi_{0N_y}(\xi_0, \eta_1) & \phi_{10}(\xi_0, \eta_1) & \phi_{11}(\xi_0, \eta_1) & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\phi_{00}(\xi_0, \eta_{M_y}) & \phi_{01}(\xi_0, \eta_{M_y}) & \cdots & \phi_{0N_y}(\xi_0, \eta_M) & \phi_{10}(\xi_0, \eta_{M_y}) & \phi_{11}(\xi_0, \eta_{M_y}) & \cdots \\
\phi_{00}(\xi_1, \eta_0) & \phi_{01}(\xi_1, \eta_0) & \cdots & \phi_{0N_y}(\xi_1, \eta_0) & \phi_{10}(\xi_1, \eta_0) & \phi_{11}(\xi_1, \eta_0) & \cdots \\
\phi_{00}(\xi_1, \eta_1) & \phi_{01}(\xi_1, \eta_1) & \cdots & \phi_{0N_y}(\xi_1, \eta_1) & \phi_{10}(\xi_1, \eta_1) & \phi_{11}(\xi_1, \eta_1) & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\phi_{00}(\xi_1, \eta_{M_y}) & \phi_{01}(\xi_1, \eta_{M_y}) & \cdots & \phi_{0N_y}(\xi_1, \eta_{M_y}) & \phi_{10}(\xi_1, \eta_{M_y}) & \phi_{11}(\xi_1, \eta_{M_y}) & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\phi_{00}(\xi_{M_x}, \eta_0) & \phi_{01}(\xi_{M_x}, \eta_0) & \cdots & \phi_{0N_y}(\xi_{M_x}, \eta_0) & \phi_{10}(\xi_{M_x}, \eta_0) & \phi_{11}(\xi_{M_x}, \eta_0) & \cdots \\
\phi_{00}(\xi_{M_x}, \eta_1) & \phi_{01}(\xi_{M_x}, \eta_1) & \cdots & \phi_{0N_y}(\xi_{M_x}, \eta_1) & \phi_{10}(\xi_{M_x}, \eta_1) & \phi_{11}(\xi_{M_x}, \eta_1) & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\phi_{00}(\xi_{M_x}, \eta_{M_y}) & \phi_{01}(\xi_{M_x},\eta_{M_y}) & \cdots & \phi_{0N_y}(\xi_{M_x}, \eta_{M_y}) & \phi_{10}(\xi_{M_x}, \eta_{M_y}) & \phi_{11}(\xi_{M_x}, \eta_{M_y}) & \cdots
\end{array} \right.$$

$$\left.\begin{array}{ccccccc}
\cdots & \phi_{1N_y}(\xi_0,\eta_0) & \cdots & \phi_{N_x0}(\xi_0,\eta_0) & \phi_{N_x1}(\xi_0,\eta_0) & \cdots & \phi_{N_xN_y}(\xi_0,\eta_0) \\
\cdots & \phi_{1N_y}(\xi_0,\eta_1) & \cdots & \phi_{N_x0}(\xi_0,\eta_1) & \phi_{N_x1}(\xi_0,\eta_1) & \cdots & \phi_{N_xN_y}(\xi_0,\eta_1) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\cdots & \phi_{1N_y}(\xi_0,\eta_{M_y}) & \cdots & \phi_{N_x0}(\xi_0,\eta_{M^y}) & \phi_{N_x1}(\xi_0,\eta_{M_y}) & \cdots & \phi_{N_xN_y}(\xi_0,\eta_{M_y}) \\
\cdots & \phi_{1N_y}(\xi_1,\eta_0) & \cdots & \phi_{N_x0}(\xi_1,\eta_0) & \phi_{N_x1}(\xi_1,\eta_0) & \cdots & \phi_{N_xN_y}(\xi_1,\eta_0) \\
\cdots & \phi_{1N_y}(\xi_1,\eta_1) & \cdots & \phi_{N_x0}(\xi_1,\eta_1) & \phi_{N_x1}(\xi_1,\eta_1) & \cdots & \phi_{N_xN_y}(\xi_1,\eta_1) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\cdots & \phi_{1N_y}(\xi_1,\eta_{M_y}) & \cdots & \phi_{N_x0}(\xi_1,\eta_{M_y}) & \phi_{N_x1}(\xi_1,\eta_{M_y}) & \cdots & \phi_{N_xN_y}(\xi_1,\eta_{M_y}) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\cdots & \phi_{1N_y}(\xi_{M_x},\eta_0) & \cdots & \phi_{N_x0}(\xi_{M_x},\eta_0) & \phi_{N_x1}(\xi_{M_x},\eta_0) & \cdots & \phi_{N_xN_y}(\xi_{M_x},\eta_0) \\
\cdots & \phi_{1N_y}(\xi_{M_x},\eta_1) & \cdots & \phi_{N_x0}(\xi_{M_x},\eta_1) & \phi_{N_x1}(\xi_{M_x},\eta_1) & \cdots & \phi_{N_xN_y}(\xi_{M_x},\eta_1) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\cdots & \phi_{1N_y}(\xi_{M_x},\eta_{M_y}) & \cdots & \phi_{N_x0}(\xi_{M_x},\eta_{M_y}) & \phi_{N_x1}(\xi_{M_x},\eta_{M_y}) & \cdots & \phi_{N_xN_y}(\xi_{M_x},\eta_{M_y})
\end{array}\right] \qquad (2.28)$$

In general, the weighted least square solution is of the same matrix form as Eq. (2.9), however, the curse of dimensionality is a significant consideration – unless some special structure is present, the normal equations "blow up" quickly in two and higher dimensioned spaces, making the computation of high polynomial degree least square approximation in high dimensioned spaces have to "pass through" a frequently poorly conditioned large matrix inverse. Fortunately, a number of special choices of basis functions and associated sample point patterns exist that render these higher dimensioned approximations tractable.

As a preface to further developments, consider a specific example (from Chapter 1 of Crassidis & Junkins [69] and [31]). Choose the power series as basis functions $\phi_{pq}(\xi,\eta) = \xi^p\eta^q$, so we seek best fitting coefficients a_{pq} in the approximation

$$f(\xi,\eta) \equiv \sum_{p=0}^{N_x}\sum_{q=0}^{N_y} a_{pq}\xi^p\eta^q, \text{ over the region : } \{-1 \leq \xi \leq +1\} \text{ and } \{-1 \leq \eta \leq +1\}. \tag{2.29}$$

In particular, if the x and y degrees are chosen $N_x = 2$ and $N_y = 1$, and if we select a uniform grid of points $\left\{\xi_i = -1 + \frac{2i}{3},\ i = 0,1,2,3;\ \eta_j = -1 + \frac{2j}{3},\ j = 0,1,2,3.\right\}$, then Eqs (2.29) become

$$
f = \begin{bmatrix} f(\xi_0,\eta_0) \\ f(\xi_0,\eta_1) \\ f(\xi_0,\eta_2) \\ f(\xi_0,\eta_3) \\ \cdots\cdots \\ f(\xi_1,\eta_0) \\ f(\xi_1,\eta_1) \\ f(\xi_1,\eta_2) \\ f(\xi_1,\eta_3) \\ \cdots\cdots \\ f(\xi_2,\eta_0) \\ f(\xi_2,\eta_1) \\ f(\xi_2,\eta_2) \\ f(\xi_2,\eta_3) \\ \cdots\cdots \\ f(\xi_3,\eta_0) \\ f(\xi_3,\eta_1) \\ f(\xi_3,\eta_2) \\ f(\xi_3,\eta_3) \end{bmatrix}, \ \Phi = \begin{bmatrix}
1 & \eta_0 & \xi_0 & \xi_0\eta_0 & \xi_0^2 & \xi_0^2\eta_0 \\
1 & \eta_1 & \xi_0 & \xi_0\eta_1 & \xi_0^2 & \xi_0^2\eta_1 \\
1 & \eta_2 & \xi_0 & \xi_0\eta_2 & \xi_0^2 & \xi_0^2\eta_2 \\
1 & \eta_3 & \xi_0 & \xi_0\eta_3 & \xi_0^2 & \xi_0^2\eta_3 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & \eta_0 & \xi_1 & \xi_1\eta_0 & \xi_1^2 & \xi_1^2\eta_0 \\
1 & \eta_1 & \xi_1 & \xi_1\eta_1 & \xi_1^2 & \xi_1^2\eta_1 \\
1 & \eta_2 & \xi_1 & \xi_1\eta_2 & \xi_1^2 & \xi_1^2\eta_2 \\
1 & \eta_3 & \xi_1 & \xi_1\eta_3 & \xi_1^2 & \xi_1^2\eta_3 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & \eta_0 & \xi_2 & \xi_2\eta_0 & \xi_2^2 & \xi_0^2\eta_0 \\
1 & \eta_1 & \xi_2 & \xi_2\eta_1 & \xi_2^2 & \xi_0^2\eta_1 \\
1 & \eta_2 & \xi_2 & \xi_2\eta_2 & \xi_2^2 & \xi_0^2\eta_2 \\
1 & \eta_3 & \xi_2 & \xi_2\eta_3 & \xi_2^2 & \xi_0^2\eta_3 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & \eta_0 & \xi_3 & \xi_3\eta_0 & \xi_3^2 & \xi_3^2\eta_0 \\
1 & \eta_1 & \xi_3 & \xi_3\eta_1 & \xi_3^2 & \xi_3^2\eta_1 \\
1 & \eta_2 & \xi_3 & \xi_3\eta_2 & \xi_3^2 & \xi_3^2\eta_2 \\
1 & \eta_3 & \xi_3 & \xi_3\eta_3 & \xi_3^2 & \xi_3^2\eta_3
\end{bmatrix}, \ a = \begin{bmatrix} a_{00} \\ a_{01} \\ \cdots \\ a_{10} \\ a_{11} \\ \cdots \\ a_{20} \\ a_{21} \end{bmatrix}. \tag{2.30}
$$

It then follows that the identity weighted least square solution in the form $\boldsymbol{f} \cong \Phi \boldsymbol{a}$ with $\boldsymbol{a}$ chosen to minimize $J = \frac{1}{2}\boldsymbol{r}^T\boldsymbol{r} = \frac{1}{2}(\boldsymbol{f} - \Phi\boldsymbol{a})^T(\boldsymbol{f} - \Phi\boldsymbol{a})$ is as before

$$\boldsymbol{a} = \left(\Phi^T\Phi\right)^{-1}\Phi^T\boldsymbol{f}. \tag{2.31}$$

In reflecting on the above, notice that the typical two dimensional basis function is just a simple product of a typical pair of one dimensional basis functions, as in $\phi_{ij}(\xi, \eta) = \phi_i(\xi)\phi_j(\eta)$; this *basis function factorization* property, together with some constraints on the location of the measurements, gives rise to some important opportunities for efficiency and high accuracy. Following Section 1.6.2 of Crassidis & Junkins [69], and in [31], and Appendix B, it turns out that Φ of Eq (2.30) is formed as the Kronecker product of two elementary matrices (where is associated with one dimensional approximation using the same grid intervals

$$\Phi = \begin{bmatrix} 1 & \xi_0 & \xi_0^2 \\ 1 & \xi_1 & \xi_1^2 \\ 1 & \xi_2 & \xi_2^2 \\ 1 & \xi_3 & \xi_3^2 \end{bmatrix} \otimes \begin{bmatrix} 1 & \eta_0 \\ 1 & \eta_1 \\ 1 & \eta_2 \end{bmatrix} = \Phi_x \otimes \Phi_y, \tag{2.32}$$

where the Kronecker matrix product operation is defined as

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{21}B & \cdots & a_{1\beta}B \\ a_{21}B & a_{22}B & \cdots & a_{2\beta}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{\alpha 1}B & a_{\alpha 2}B & \cdots & a_{\alpha\beta}B \end{bmatrix}. \tag{2.33}$$

The Kronecker product $C = A \otimes B$ is implemented in MATLAB® via the command $C = \text{kron}(A, B)$. As discussed in Crassidis and Junkins, the fact that the above Φ

matrix is "Kronecker-factorable" as in Eq (2.33) is immediately verified, because Eq (2.32) generates Eq (2.30). There are important consequences: The un-weighted least squares solution of Eq (2.33) is alternatively computed from a Kronecker product of two smaller least square operator matrices as

$$\boldsymbol{a} = \left(\Phi^T \Phi\right)^{-1} \Phi^T \boldsymbol{f} = \left\{ \left[\left(\Phi_x^T \Phi_x\right)^{-1} \Phi_x^T \right] \otimes \left[\left(\Phi_y^T \Phi_y\right)^{-1} \Phi_y^T \right] \right\} \boldsymbol{f}. \tag{2.34}$$

As proven in Appendix B, Eq (2.34) holds for any Kronecker factorable Φ, as $\Phi = \Phi_x \otimes \Phi_y$, not just the above special case (i.e., Eq (2.34) holds for general maximum rank matrices Φ_x, Φ_y, not merely for the special case definitions evident in Eq. (2.32)). The consequences of Eq (2.34) are immediate: We can solve a larger two dimensional least squares problem by taking Kronecker matrix product of the "least squares operators" for two corresponding one dimensional least squares problems. The dimensions of the matrices that need inverting in Eq (2.34) are qualitatively the "square root" of the dimensions of the matrix that needs inverting in Eq (2.31). While this is generally significant, the implications for orthogonal approximation are even more significant. It can be shown that this result generalizes to higher dimensioned cases, as follows: If $\Phi = \Phi_x \otimes \Phi_y \otimes \Phi_z$ then the generalization of Eq. (2.34) to a three dimensional approximation space (x, y, z) or (ξ, η, ζ) can be verified to be

$$\boldsymbol{a} = \left(\Phi^T \Phi\right)^{-1} \Phi^T \boldsymbol{f} = \left\{ \left[\left(\Phi_x^T \Phi_x\right)^{-1} \Phi_x^T \right] \otimes \left[\left(\Phi_y^T \Phi_y\right)^{-1} \Phi_y^T \right] \otimes \left[\left(\Phi_z^T \Phi_z\right)^{-1} \Phi_z^T \right] \right\} \boldsymbol{f}. \tag{2.35}$$

In using the Kronecker product of three small least square operators to produce the larger least square operator, we are approximately taking the cube root of the size of matrices that have to be inverted. Thus, for once, the power of an idea *increases* dramatically as the dimension (of the space in which we are approximating

f) increases! The generalization to n-dimensions is evident and qualitatively, the n^{th} root of the matrix dimension (we must invert) is taken. Before addressing the further advantages of selecting orthogonal basis functions, judicious weighting and nodes, let us consider how to modify the above to accommodate a positive definite weight matrix. For the case of weighted least squares:

$$\boldsymbol{a} = \left(\Phi^T W \Phi\right)^{-1} \Phi^T W \boldsymbol{f} \equiv C\boldsymbol{f} \equiv \left(\overline{\Phi}^T \overline{\Phi}\right)^{-1} \overline{\Phi}^T \bar{\boldsymbol{f}} \equiv \overline{C}\,\bar{\boldsymbol{f}}. \tag{2.36}$$

Where we introduce the Cholesky square root $W^{\frac{1}{2}}$ of the weight matrix $W = W^{\frac{1}{2}} W^{\frac{1}{2}}$, for the diagonal case $W = diag\left\{ \begin{matrix} w_0 & w_1 & w_2 & \cdots \end{matrix} \right\}$, it is clear that $W^{\frac{1}{2}} = diag\left\{ \begin{matrix} w_0^{\frac{1}{2}} & w_1^{\frac{1}{2}} & w_2^{\frac{1}{2}} & \cdots \end{matrix} \right\}$ and the $\left(C, \overline{C}\right)$ matrices in Eqs (2.36) are defined as

$$C = \left(\Phi^T W \Phi\right)^{-1} \Phi^T W, \;\; \overline{C} \equiv \left(\overline{\Phi}^T \overline{\Phi}\right)^{-1} \overline{\Phi}^T, \tag{2.37}$$

where $\bar{\Phi} \equiv W^{\frac{1}{2}} \Phi, \;\; \bar{\boldsymbol{f}} = W^{\frac{1}{2}} \boldsymbol{f}$.

Thus, if Φ is Kronecker factorable, then essentially the same advantages are enjoyed as in the identity weight matrix case. Now let us consider the weighted version of Eq. (2.35)

$$\begin{gathered}
\boldsymbol{a} = \left(\Phi^T W \Phi\right)^{-1} \Phi^T W \boldsymbol{f} \\
= \left\{ \left[\left(\Phi_x^T W_x \Phi_x\right)^{-1} \Phi_x^T W_x\right] \otimes \left[\left(\Phi_y^T W_y \Phi_y\right)^{-1} \Phi_y^T W_y\right] \otimes \left[\left(\Phi_z^T W_z \Phi_z\right)^{-1} \Phi_z^T W_z\right] \right\} \boldsymbol{f}, \\
\boldsymbol{a} = \left(\overline{\Phi}^T \overline{\Phi}\right)^{-1} \overline{\Phi}^T \bar{\boldsymbol{f}} = \left\{ \left[\left(\overline{\Phi}_x^T \overline{\Phi}_x\right)^{-1} \overline{\Phi}_x^T\right] \otimes \left[\left(\overline{\Phi}_y^T \overline{\Phi}_y\right)^{-1} \overline{\Phi}_y^T\right] \otimes \left[\left(\overline{\Phi}_z^T \overline{\Phi}_z\right)^{-1} \overline{\Phi}_z^T\right] \right\} \bar{\boldsymbol{f}},
\end{gathered} \tag{2.38}$$

$$\text{where } \overline{\Phi} \equiv W^{\frac{1}{2}} \Phi, \overline{\Phi}_x \equiv W_x^{\frac{1}{2}} \Phi, \;\; \overline{\Phi}_y \equiv W_y^{\frac{1}{2}} \Phi, \;\; \overline{\Phi}_z \equiv W_z^{\frac{1}{2}} \Phi, \;\; \bar{\boldsymbol{f}} \equiv W^{\frac{1}{2}} \boldsymbol{f}.$$

The question naturally arises: What is the relationship between the weight matrices $\{W, W_x, W_y, W_z\}$? Consider the diagonal special case:

$$\begin{aligned} W_x &= diag\left\{W_{x0}, W_{x1}, ..., W_{xM_x}\right\} \\ W_y &= diag\left\{W_{y0}, W_{y1}, ..., W_{yM_y}\right\} \\ W_z &= diag\left\{W_{z0}, W_{z1}, ..., W_{zM_z}\right\}. \end{aligned} \tag{2.39}$$

Substitution of Eq (2.39) into Eq (2.38) and some algebra leads to the conclusion that

$$W = diag\left\{W_{x0}W_{y0}W_{z0}, W_{x0}W_{y0}W_{z1}, ..., W_{x0}W_{y0}W_{zM_z} \vdots W_{x0}W_{y1}W_{z0}, W_{x0}W_{y1}W_{z1}, ... \right.$$
$$\left., W_{x0}W_{y1}W_{zM_z} \vdots \cdots \vdots W_{xM_x}W_{yM_y}W_{z0}, W_{xM_x}W_{yM_y}W_{z1}, ..., W_{xM_x}W_{yM_y}W_{zM_z}\right\}$$
$$\text{or} \quad W = W_x \otimes W_y \otimes W_z. \tag{2.40}$$

As an example, if we have the special case (for $M_x = M_y = 3$, anticipating subsequent applications, we use weights and sample locations corresponding to the Chebyshev polynomials):

$$W_x = diag\left\{\frac{1}{2}, 1, 1, \frac{1}{2}\right\}; \; W_y = diag\left\{\frac{1}{2}, 1, 1, \frac{1}{2}\right\};$$
$$W = W_x \otimes W_y = diag\left\{\frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4} \vdots \frac{1}{2}, 1, 1, \frac{1}{2} \vdots \frac{1}{2}, 1, 1, \frac{1}{2} \vdots \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}\right\}. \tag{2.41}$$

With these insights, we now generalize to two dimensions the developments leading up to Eq (2.19), for approximation using Chebyshev orthogonal polynomials. In particular, we seek a least square approximation of the form

$$f(\xi, \eta) \cong \sum_{p=0}^{N_x}\sum_{q=0}^{N_y} a_{pq}\phi_p(\xi)\phi_q(\eta), \text{ over the region : } \{-1 \le \xi \le +1\} \text{ and } \{-1 \le \eta \le +1\}. \tag{2.42}$$

Let the data be denoted $f(\xi_i, \eta_j)$ where $\{-1 \leq \xi_i \leq +1\}$ and $\{-1 \leq \eta_j \leq +1\}$ with the sample points located according to the two dimensional cosine grid point distribution:

$$\{\xi_i = -\cos(i\pi/M_x),\ i = 0, 1, 2, ..., M_x \text{ and } \eta_j = -\cos(j\pi/M_y),\ j = 0, 1, 2, ..., M_y\}\,. \tag{2.43}$$

Then we have

$$\boldsymbol{f} = \begin{bmatrix} f(\xi_0,\eta_0) \\ f(\xi_0,\eta_1) \\ \vdots \\ f(\xi_0,\eta_{M_y}) \\ \cdots\cdots\cdots \\ f(\xi_1,\eta_0) \\ f(\xi_1,\eta_1) \\ \vdots \\ f(\xi_1,\eta_{1M_y}) \\ \cdots\cdots\cdots \\ \vdots \\ \cdots\cdots\cdots \\ f(\xi_{M_x},\eta_0) \\ f(\xi_{M_x},\eta_1) \\ \vdots \\ f(\xi_{M_x},\eta_{M_y}) \end{bmatrix}, \text{ and } \left.\begin{array}{c} \Phi_x = \begin{bmatrix} \phi_0(\xi_0) & \phi_1(\xi_0) & \cdots & \phi_{N_x}(\xi_0) \\ \phi_0(\xi_1) & \phi_1(\xi_1) & \cdots & \phi_{N_x}(\xi_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\xi_{M_x}) & \phi_1(\xi_{M_x}) & \cdots & \phi_{N_x}(\xi_{M_x}) \end{bmatrix} \quad \overline{\Phi}_x = W_x^{\frac{1}{2}}\Phi_x \\ \\ \Phi_x = \begin{bmatrix} \phi_0(\eta_0) & \phi_1(\eta_0) & \cdots & \phi_{N_y}(\eta_0) \\ \phi_0(\eta_1) & \phi_1(\eta_1) & \cdots & \phi_{N_y}(\eta_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\eta_{M_y}) & \phi_1(\eta_{M_y}) & \cdots & \phi_{N_y}(\eta_{M_y}) \end{bmatrix} \quad \overline{\Phi}_y = W_y^{\frac{1}{2}}\Phi_y \\ \\ \Phi = \Phi_x \otimes \Phi_y,\ \overline{\Phi} = W^{\frac{1}{2}}\Phi \\ \\ W_x = diag\left(\tfrac{1}{2}, 1, 1, ..., 1, \tfrac{1}{2}\right),\ \text{an } M_x \times M_x \text{ matrix} \\ \\ W_y = diag\left(\tfrac{1}{2}, 1, 1, ..., 1, \tfrac{1}{2}\right),\ \text{an } M_y \times M_y \text{ matrix} \\ \\ W = W_x \otimes W_y \\ \\ = diag\left\{\tfrac{1}{4}, \tfrac{1}{2}, \tfrac{1}{2}, ...\tfrac{1}{2}, \tfrac{1}{4}, \vdots \tfrac{1}{2}, 1, 1, ..., 1, \tfrac{1}{2} \vdots \tfrac{1}{4}, \tfrac{1}{2}, \tfrac{1}{2}, ..., \tfrac{1}{2}, \tfrac{1}{4}\right\} \end{array}\right\} \tag{2.44}$$

For specificity, we adopt the case that $M_x > N_x, M_y > N_y$.

Note that the least squares solution is given by any of Eqs (2.36)-(2.38), with or without making use of the Kronecker factorization. However, these numerical solutions take no advantage of the orthogonality of the solution process, and the one dimensional least square operators assume especially attractive forms for the case of orthogonal basis functions. In particular, it is evident from considering all of the above that the least square solution in two dimensions is "constructed" with no matrix inverse from orthogonal least square operators corresponding to two one-dimensional Chebyshev least square operators as follows:

$$\boldsymbol{a} = C\boldsymbol{f} \equiv \underbrace{\left(\Phi^T W \Phi\right)^{-1} \Phi^T W}_{\equiv C} \boldsymbol{f} = \underbrace{\left(\overline{\Phi}^T \overline{\Phi}\right)^{-1} \overline{\Phi}^T}_{\equiv \overline{C}} \overline{\boldsymbol{f}} = \overline{C}\, \overline{\boldsymbol{f}}, \tag{2.45}$$

where $\overline{\boldsymbol{f}} = W^{\frac{1}{2}} \boldsymbol{f}$ and the Kronceker product recipe for constructing the two dimensional Chebyshev least square operator matrix is the right-most of the following equations:

$$\overline{C} \equiv \left(\overline{\Phi}^T \overline{\Phi}\right)^{-1} \overline{\Phi}^T = \left(\Phi^T W \Phi\right)^{-1} \Phi^T W^{\frac{1}{2}} = C W^{-\frac{1}{2}} \equiv \overline{C}_x \otimes \overline{C}_y, \tag{2.46}$$

and where $\overline{C}_x = \left(\overline{\Phi}_x^T \overline{\Phi}_x\right)^{-1} \overline{\Phi}_x^T \equiv \left(\Phi_x^T W_x \Phi_x\right)^{-1} \Phi_x^T W_x^{\frac{1}{2}} \equiv C_x W_x^{-\frac{1}{2}}$, C_x is from Eq (2.19), leading to:

$$\overline{C}_x = \frac{1}{M_x} \begin{bmatrix} \frac{1}{2}T_0(\xi_0) & T_0(\xi_1) & \cdots & T_0(\xi_{M_x-1}) & \frac{1}{2}T_0(\xi_{M_x}) \\ T_1(\xi_0) & 2T_1(\xi_1) & \cdots & 2T_1(\xi_{M_x-1}) & T_1(\xi_{M_x}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ T_{N_x-1}(\xi_0) & 2T_{N_x-1}(\xi_1) & \cdots & 2T_{N_x-1}(\xi_{M_x-1}) & T_{N_x-1}(\xi_{M_x}) \\ T_{N_x}(\xi_0) & 2T_{N_x}(\xi_1) & \cdots & 2T_{N_x}(\xi_{M_x-1}) & T_{N_x}(\xi_{M_x}) \end{bmatrix} \times$$

$$\begin{bmatrix} \frac{1}{2} & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & \frac{1}{2} \end{bmatrix}^{-\frac{1}{2}}$$

or finally

$$\overline{C}_x = \frac{1}{M_x}\begin{bmatrix} \frac{\sqrt{2}}{2}T_0(\xi_0) & T_0(\xi_1) & \cdots & T_0(\xi_{M_x-1}) & \frac{\sqrt{2}}{2}T_0(\xi_{M_x}) \\ \sqrt{2}T_1(\xi_0) & 2T_1(\xi_1) & \cdots & 2T_1(\xi_{M_x-1}) & \sqrt{2}T_1(\xi_{M_x}) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sqrt{2}T_{N_x-1}(\xi_0) & 2T_{N_x-1}(\xi_1) & \cdots & 2T_{N_x-1}(\xi_{M_x-1}) & \sqrt{2}T_{N_x-1}(\xi_{M_x}) \\ \sqrt{2}T_{N_x}(\xi_0) & 2T_{N_x}(\xi_1) & \cdots & 2T_{N_x}(\xi_{M_x-1}) & \sqrt{2}T_{N_x}(\xi_{M_x}) \end{bmatrix} \quad (2.47)$$

$$\text{with} \begin{cases} x \longrightarrow y \\ \xi \longrightarrow \eta \end{cases},$$

$$\overline{\Phi}_x = W_x^{\frac{1}{2}}\Phi_x,\ \overline{\Phi}_y = W_y^{\frac{1}{2}}\Phi_y,\ \overline{\Phi} = \overline{\Phi}_x \otimes \overline{\Phi}_y. \quad (2.48)$$

The final row in the matrix $\overline{C}_x$ of Eq (2.48) holds for the over-determined (least square) case ($M > N$), alternatively, and for the determined (square) $M = N$ case, the last row of $\overline{C}_x$ is replaced by

$$\begin{bmatrix} \frac{\sqrt{2}}{2}T_{N_x}(\xi_0) & T_{N_x}(\xi_1) & \cdots & T_{N_x}(\xi_{M_x-1}) & \frac{\sqrt{2}}{2}T_{N_x}(\xi_{M_x}) \end{bmatrix}. \quad (2.49)$$

We mention, without going through the details, that this formulation readily extends to n-dimensions, for example for approximating a function in three dimensions, Eq. (2.46) are simply

$$\overline{C} = \left(\Phi^T W \Phi\right)^{-1}\Phi^T W = \overline{C}_x \otimes \overline{C}_y \otimes \overline{C}_z,\ W \equiv W_x \otimes W_y \otimes W_z,\ \bar{\boldsymbol{f}} = W^{\frac{1}{2}}\boldsymbol{f}, \quad (2.50)$$

where $\overline{C}_z$ has the same form as Eqs (2.48,2.48) and the coefficients are given by Eq. (2.34). The flowchart explaining the 2-D approximation is given in Figure II.7

In general, let us consider a function of n independent variables

$$g(x_1, x_2, \cdots, x_n),\ \{x_{i_{\min}} \le x_i \le x_{i_{\max}}\},\ \{i = 1, 2, \cdots, n\}, \quad (2.51)$$

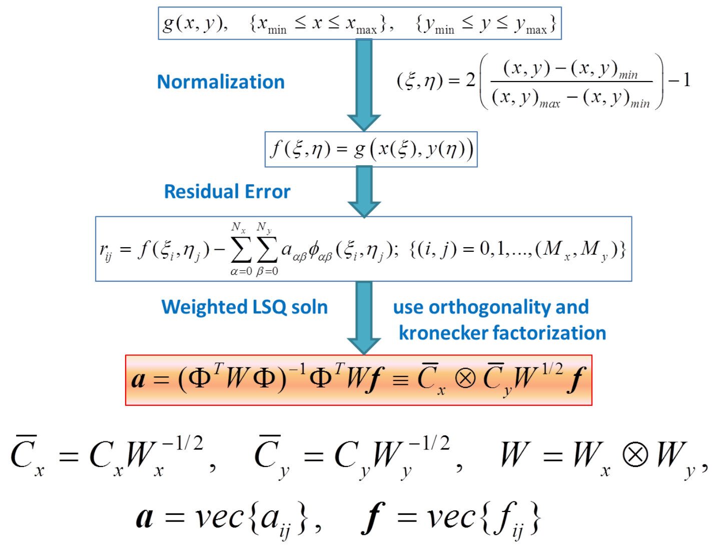

$g(x,y), \quad \{x_{\min} \le x \le x_{\max}\}, \quad \{y_{\min} \le y \le y_{\max}\}$
Normalization
$(\xi,\eta) = 2\left(\frac{(x,y)-(x,y)_{min}}{(x,y)_{max}-(x,y)_{min}}\right)-1$
$f(\xi,\eta) = g\left(x(\xi), y(\eta)\right)$
Residual Error
$r_{ij} = f(\xi_i,\eta_j) - \sum_{\alpha=0}^{N_x}\sum_{\beta=0}^{N_y} a_{\alpha\beta}\phi_{\alpha\beta}(\xi_i,\eta_j); \; \{(i,j)=0,1,\dots,(M_x,M_y)\}$
Weighted LSQ soln
use orthogonality and kronecker factorization
$\boldsymbol{a} = (\Phi^T W \Phi)^{-1}\Phi^T W\boldsymbol{f} \equiv \bar{C}_x \otimes \bar{C}_y W^{1/2}\boldsymbol{f}$
$\bar{C}_x = C_x W_x^{-1/2}, \quad \bar{C}_y = C_y W_y^{-1/2}, \quad W = W_x \otimes W_y,$
$\boldsymbol{a} = vec\{a_{ij}\}, \quad \boldsymbol{f} = vec\{f_{ij}\}$

Figure II.7: 2-D Approximation.

$$\xi_i(x_i) = -1 + 2\left(x_i - x_{i_{\min}}\right) / \left(x_{i_{\max}} - x_{i_{\min}}\right), \tag{2.52}$$

$$x_i(\xi_i) = x_{i_{\min}} + (\xi_i + 1)\left(x_{i_{\max}} - x_{i_{\min}}\right)/2. \tag{2.53}$$

Substituting Eqs (2.53) into Eq (2.51), we see that we wish to approximate the function

$$\begin{aligned} f(\xi_1, \xi_2, \cdots, \xi_n) &\triangleq g\left(x_1, x_2, \cdots, x_n\right) \\ f(\xi_i) &\triangleq g\left(x_i(\xi_i)\right) \equiv g\left(\underbrace{x_{i_{\min}} + (\xi_i + 1)\left(x_{i_{\max}} - x_{i_{\min}}\right)/2}_{x_i(\xi_i)}\right). \end{aligned} \tag{2.54}$$

In the general case, we seek to approximate $f(\xi_i)$ as a linear combination of a prescribed set of linearly independent basis functions of two variables

$$\left\{\phi_{00\cdots0}(\xi_i), \phi_{00\cdots1}(\xi_i), \ldots, \phi_{N_{x_1}N_{x_2}\cdots N_{x_n}}(\xi_i)\right\}$$

as

$$f(\xi_i) = f(\xi_1, \xi_2, \cdots, \xi_n) \equiv \sum_{\alpha_1=0}^{N_{x_1}}\sum_{\alpha_2=0}^{N_{x_2}} \cdots \sum_{\alpha_n=0}^{N_{x_n}} a_{\alpha_1\alpha_2\cdots\alpha_n}\phi_{\alpha_1\alpha_2\cdots\alpha_n}(\xi_1, \xi_2, \cdots, \xi_n). \tag{2.55}$$

For the case of discrete measurement samples, we introduce a set of sample points (nodes) as $\left\{\xi_{i_0}, \xi_{i_1}, ..., \xi_{i_{M_{x_i}}}; M_{x_i} \geqslant N_{x_i}\right\}$. The residual approximation error at each measurement node is

$$r_{i_k} = r_{1_k 2_k \cdots n_k} = f(\xi_{1_k}, \xi_{2_k}, \cdots, \xi_{n_k}) - \sum_{\alpha_1=0}^{N_{x_1}} \sum_{\alpha_2=0}^{N_{x_2}} \cdots \sum_{\alpha_n=0}^{N_{x_n}} a_{\alpha_1 \alpha_1 \cdots \alpha_n} \phi_{\alpha_1 \alpha_2 \cdots \alpha_n}(\xi_{1_k}, \xi_{2_k}, \cdots, \xi_{n_k}), \tag{2.56}$$

where the indices $\{i_k = (1_k, 2_k, \cdots, n_k) = 0, 1, ..., (M_{x_1}, M_{x_2}, \cdots, M_{x_n})\}$ and k indicates the k^{th} measurement for the i^{th} variable. Eq. (2.56) is alternatively written in vector-matrix notation, $\boldsymbol{r} = \boldsymbol{f} - \Phi \boldsymbol{a}$. The weighted least squares solution is computed from a Kronecker product of n smaller least square operator matrices as:

$$\boldsymbol{a} = \left(\Phi^T W \Phi\right)^{-1} \Phi^T W \boldsymbol{f} \equiv \overline{C}_{x_1} \otimes \overline{C}_{x_2} \otimes \cdots \otimes \overline{C}_{x_n} W^{1/2} \boldsymbol{f}, \tag{2.57}$$

where $\overline{C}_{x_i} = C_{x_i} W_{x_i}^{-1/2}$, $W = W_{x_1} \otimes W_{x_2} \otimes \cdots \otimes W_{x_n}$, and $\boldsymbol{a} = vec\{a_{1_k 2_k \cdots n_k}\}$, and $\boldsymbol{f} = vec\{f_{1_k 2_k \cdots n_k}\}$. The flowchart explaning the n-D approximation is shown in Figure II.8

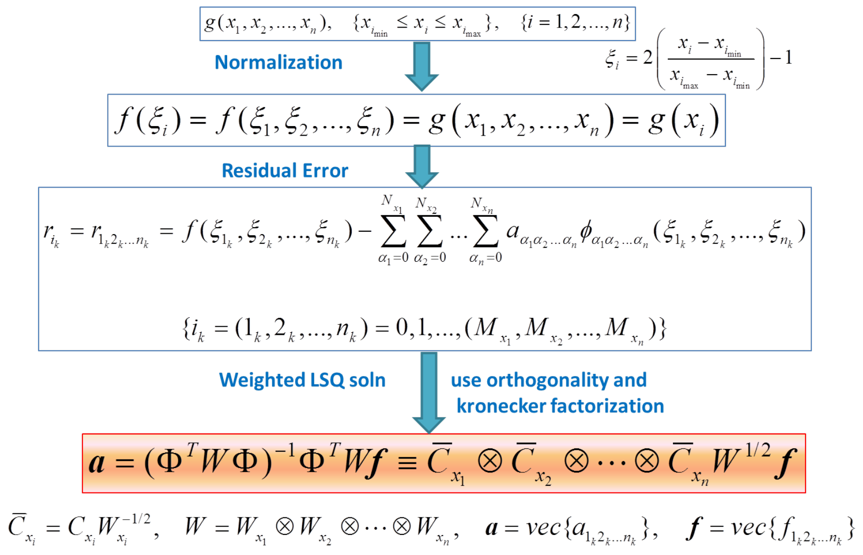

$g(x_1, x_2, \ldots, x_n), \quad \{x_{i_{\min}} \le x_i \le x_{i_{\max}}\}, \quad \{i = 1, 2, \ldots, n\}$
Normalization
$\xi_i = 2\left(\dfrac{x_i - x_{i_{\min}}}{x_{i_{\max}} - x_{i_{\min}}}\right) - 1$
$f(\xi_i) = f(\xi_1, \xi_2, \ldots, \xi_n) = g(x_1, x_2, \ldots, x_n) = g(x_i)$
Residual Error
$r_{i_k} = r_{1_k 2_k \ldots n_k} = f(\xi_{1_k}, \xi_{2_k}, \ldots, \xi_{n_k}) - \sum_{\alpha_1=0}^{N_{x_1}} \sum_{\alpha_2=0}^{N_{x_2}} \cdots \sum_{\alpha_n=0}^{N_{x_n}} a_{\alpha_1 \alpha_2 \ldots \alpha_n} \phi_{\alpha_1 \alpha_2 \ldots \alpha_n}(\xi_{1_k}, \xi_{2_k}, \ldots, \xi_{n_k})$
$\{i_k = (1_k, 2_k, \ldots, n_k) = 0, 1, \ldots, (M_{x_1}, M_{x_2}, \ldots, M_{x_n})\}$
Weighted LSQ soln
use orthogonality and kronecker factorization
$\boldsymbol{a} = (\Phi^T W \Phi)^{-1} \Phi^T W \boldsymbol{f} \equiv \bar{C}_{x_1} \otimes \bar{C}_{x_2} \otimes \cdots \otimes \bar{C}_{x_n} W^{1/2} \boldsymbol{f}$
$\bar{C}_{x_i} = C_{x_i} W_{x_i}^{-1/2}, \quad W = W_{x_1} \otimes W_{x_2} \otimes \cdots \otimes W_{x_n}, \quad \boldsymbol{a} = vec\{a_{1_k 2_k \ldots n_k}\}, \quad \boldsymbol{f} = vec\{f_{1_k 2_k \ldots n_k}\}$

Figure II.8: n-D Approximation.

II.E. Numerical Examples for Illustrative Test Functions of Two Variables

To construct some two dimensional test cases that relate closely to the above one dimensional examples, we define Test Function 2: $f(\xi, \eta) \equiv G(\xi)G(\eta)$ where

$$G(x) = \frac{x}{2} + \left(\frac{\left[\left(\frac{1}{10} + x\right) sin(5x - 1)\right]}{\left[1 + x^2 sin^2\left(x - \frac{1}{2}\right)\right]} \right). \tag{2.58}$$

Below in Figure II.9 is an illustration of the cosine nodal distribution in one, two and three dimensional spaces; the generalization to a hypercube is straightforward. We now consider several cases analogous to the one dimensional case, but we omit detailed but straightforward discussions and focus on approximation ideas to vitally important problems in astrodynamics. Similar experiments as in the Test Function 1 case are performed using Eq (2.42) to generate measurements with either $M_x = M_y = M = 80$ or $M_x = M_y = M = N$, with N swept. The true function is shown in Figure II.10 and in Figures II.11 and II.12 are several approximations. The Runge Phenomena as evident in Figure II.12b and II.12d (note large boundary errors for the power series approximation versus the more uniform Chebyshev approximation, which is concentrated in the center) can be expected to generalize for higher dimensioned cases. We approximate Test Function 2 (Figure II.10) with a residual error approaching a machine zero. All computations are performed using MATLAB® with 16 digit floating point arithmetic.

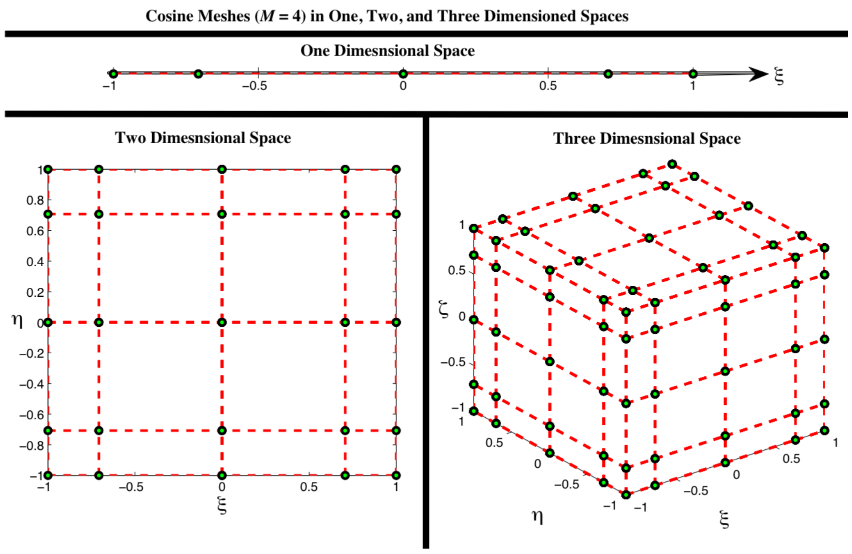


Figure II.9: Multidimensional Cosine Meshes for Discrete Orthogonality Chebyshev Polynomials in n Dimensional Approximation.

Figure II.11 shows the approximation results for the Chebyshev and power series polynomial approximation of Test Function 2 for ($M = N = 5, 10, 30$). The power series experienced large Runge errors near the boundary and the least square solutions "died" altogether due to ill-conditioning around $N \sim 15$. Figure II.12a to

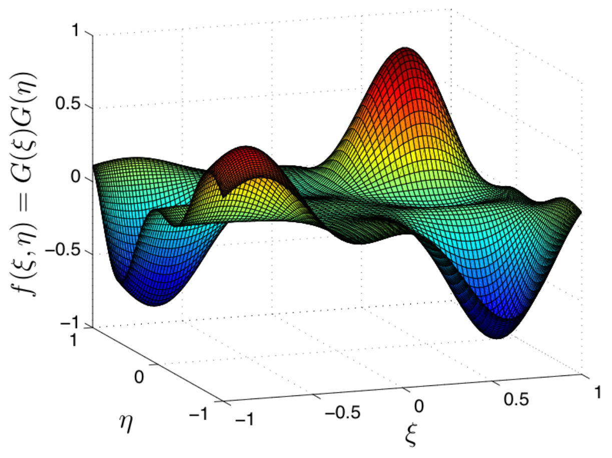


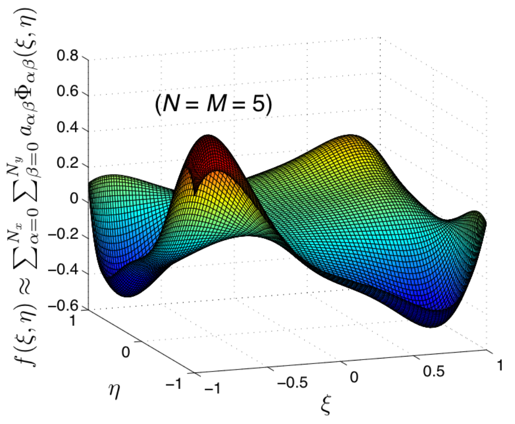
Figure II.10: Test Function 2.

II.12d show the approximation error for the Chebyshev and power series polynomial approximation of Test Function 2 for ($M = N = 10, 30$). Note for low degree approximation that the power series works fairly well in the center of the interval, but encounters large errors near the boundary (see Figures II.12b, II.12d). The maximum errors are shown in Figure II.12e and II.12f that result from least square approximation when $M = 80$ measurement nodes are used, for the case of the Chebyshev and power series polynomial approximation of Test Function 2. The Chebyshev approximations converged to 8 digit accuracy around $N = 20$, and ~ 15 digit accuracy is obtained (essentially a machine zero approximation error) around $N = 50$.

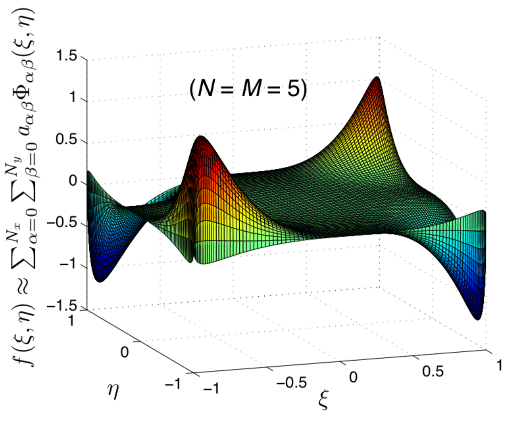
The uniform convergence of the Chebyshev approximation again approaches machine precision by $N = 50$, with the maximum error decreasing about one order of magnitude every time the degree N is increased by $\Delta N = 3$. On the other hand, the slope is much less for $N < 15$ for the power series case (due to the Runge Phenomena), and the power series cannot be computed accurately above $N \sim 15$ due to poor conditioning of the normal Eqs (2.9), which must be inverted numerically for the case of non-orthogonal basis functions.

The numerical examples for illustrative test functions of three variables, Test Function 3, is discussed in Appendix D. Test Function 3 is defined as $f(\xi, \eta, \zeta) \equiv G(\xi)G(\eta)G(\zeta)$ where

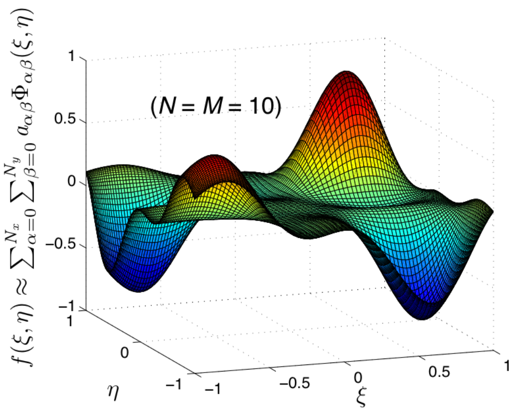
$$G(x) = \frac{x}{2} + \left(\frac{\left[\left(\frac{1}{10} + x\right) sin(5x - 1)\right]}{\left[1 + x^2 sin^2\left(x - \frac{1}{2}\right)\right]} \right).$$



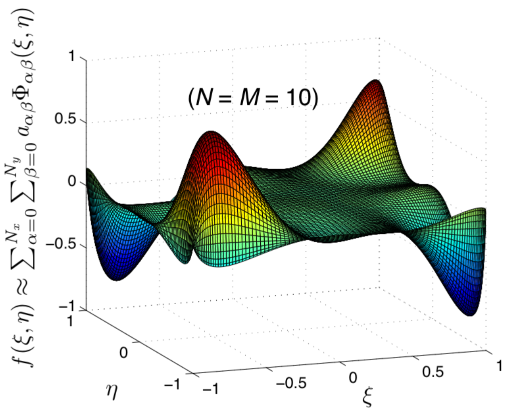
(a) Chebyshev Approximation.



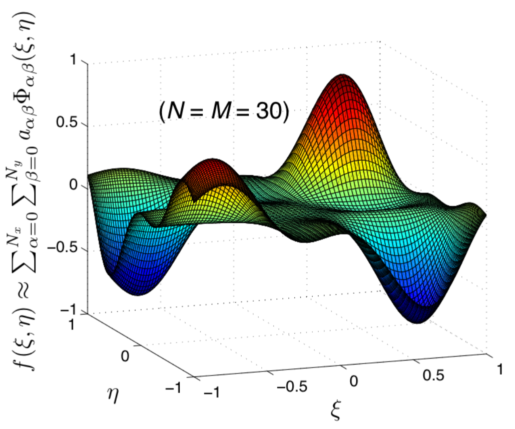
(b) Power Series Approximation.



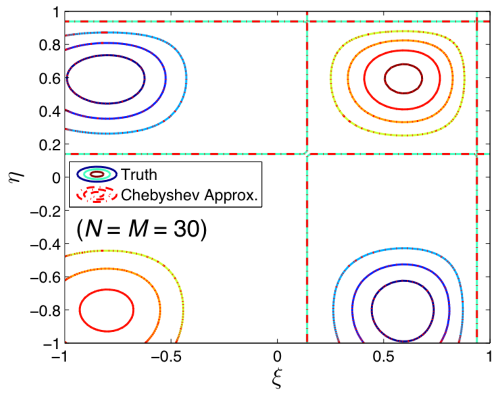
(c) Chebyshev Approximation.



(d) Power Series Approximation.

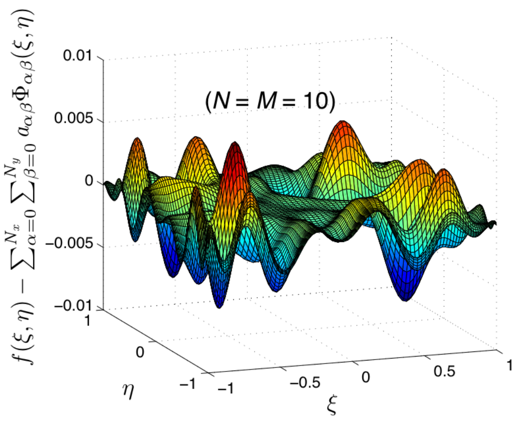


(e) Chebyshev Approximation.

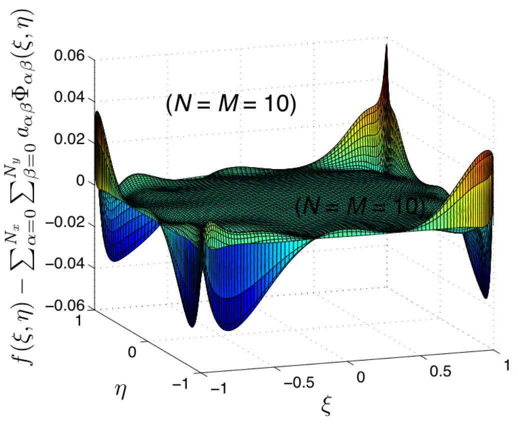


(f) Contours of Chebyshev Approximation Superimposed on True Contours

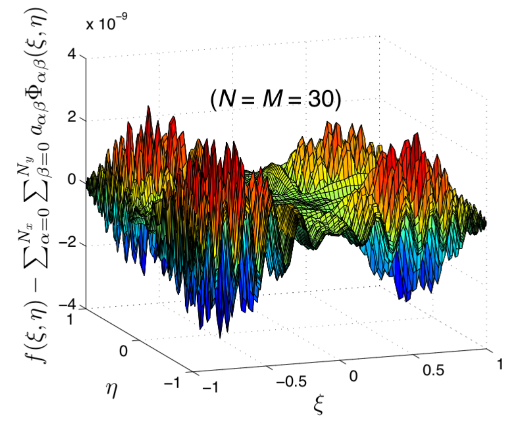
Figure II.11: Approximation of Test Function 2.



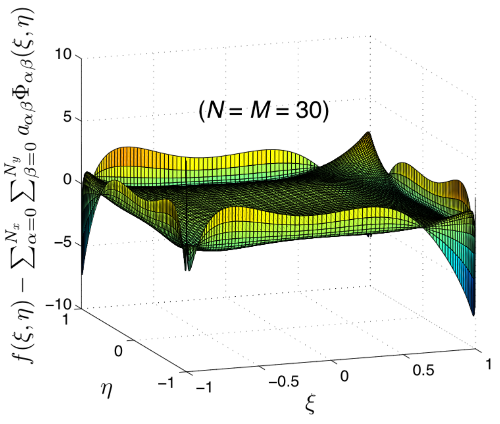
(a) Chebyshev Error.



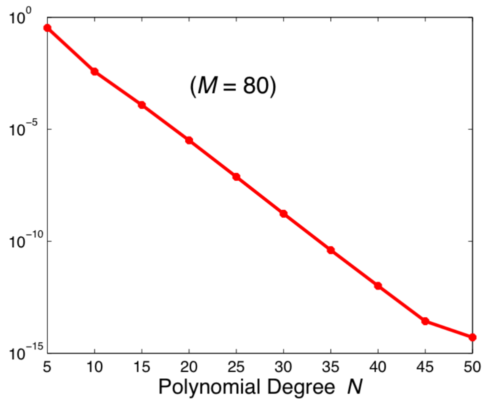
(b) Power Series Error.



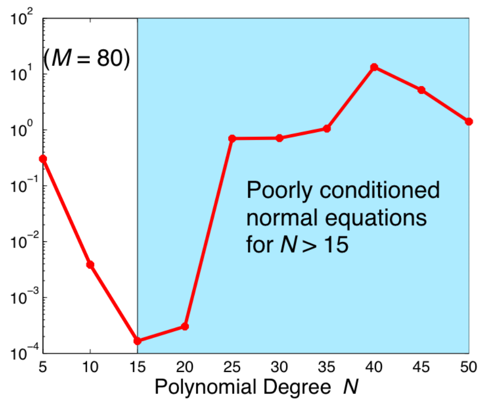
(c) Chebyshev Error.



(d) Power Series Error.



(e) Chebyshev Maximum Approximation Error.



(f) Power Series Maximum Approximation Error.

Figure II.12: Approximation Error of Test Function 2.

CHAPTER III

ORTHOGONAL FINITE ELEMENT REPRESENTATIONS OF THE GEOPOTENTIAL

III.A. Introduction

In the following discussion we first consider the construction of an orthogonal FEM approximation to the gravity potential field model determined from the Gravity Recovery And Climate Experiment (GRACE). The GRACE Gravity Model has been publicly released [34,35,71]. Access to the model's coefficients and other descriptive files about GRACE were obtained from [35,71]. After presenting the results for the GRACE gravity model, we then consider the analogous FEM approximation of the $(200, 200)$ EGM 2008 gravity model [36], in order to see the effects of including the higher order gravitational anomalies.

The classical solution to Laplace's equation for gravity is adopted using the globally valid spherical harmonic gravity potential model, defined by [25,34,37]:

$$U(r, \lambda, \phi) \equiv \frac{\mu}{r}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\left(\frac{R_{\oplus}}{r}\right)^{n} P_{n}^{m}(\sin\phi)\left[C_{n}^{m}\cos m\lambda + S_{n}^{m}\sin m\lambda\right], \tag{3.1}$$

where the coordinate r is the geocentric radius (i.e. distance from the Earth's center to the typical point near the Earth), λ and ϕ are the geocentric (geographic) latitude and longitude respectively, and $\mu = GM$ is the Earth's gravitational-mass constant, and $R_{\oplus}$ is the Earth equator radius, and C_{n}^{m} and S_{n}^{m} are spherical harmonic gravity coefficients, P_{n}^{m} are the fully normalized associated Legendre polynomials of degree n and order m. The acceleration coordinate systems can be obtained from the potential using the classical gradient relationships in spherical or rectangular coordinates as follows.

$$\begin{array}{ccc} \underline{\text{Spherical}} & & \underline{\text{Rectangular}} \\ \left\{ \begin{array}{c} \text{South}: \ G_S = -\frac{1}{r}\frac{\partial U}{\partial \phi} \\ \text{East}: \ G_E = -\frac{1}{r\,cos\phi}\frac{\partial U}{\partial \lambda} \\ \text{Radial}: \ G_R = \frac{\partial U}{\partial r} \end{array} \right\} & \Longleftrightarrow & \left\{ \begin{array}{c} G_x = \frac{\partial U}{\partial x} \\ G_y = \frac{\partial U}{\partial y} \\ G_z = \frac{\partial U}{\partial z} \end{array} \right\} \end{array}$$

The spherical harmonic gravity model has obvious utility, but if used to represent the gravity field to high precision, one encounters three main challenges:

1. Choosing a finite upper limit of the series defines the accuracy (the more we know about gravity, the more terms are required and the more it costs to compute acceleration)

2. Convergence is very inefficient and slow for $n > 2$, so tens of thousands of terms are frequently required to obtain a sufficiently high accuracy representation

3. The north and south poles represent non-free singularities of the usual spherical coordinates (longitude is undefined at the north and south poles)

In view of the slow convergence of global gravity models, we are motivated to introduce a finite element model (FEM) local gravity representations in the anticipation that much lower degree functions can be used to efficiently model and compute local gravity. The literature on this subject initiated with Junkins' classical developments [24–26, 44] and has recently been explored by other others [41–43]. In our developments herein, we have solved a key historical challenge implicit in this class of methods for geopotential representation: How do we structure the FEM models to render them *radially adaptive* and efficient, so that the resulting algorithms "automatically know" about the rapid radial decay of the high frequency terms and more to the point, which terms in the FEM representation to retain, as a function (mainly) of radial distance from geocenter. Addressing this issue herein, we show below that a much improved efficiency can be achieved.

We consider the total gravity potential model split into reference and disturbance gravity terms, where, as the most usual example, the global reference gravity term includes the O(1) 2-body and the O(10^{-3}) J_2 oblateness terms whereas "everything else" (all the higher degree and order terms) are considered pertubative gravity disturbance to the reference model. The potential and the acceleration are

$$\underbrace{U(r,\lambda,\phi)}_{Total} = \underbrace{U_{2B}(r,\lambda,\phi) + U_{J_2}(r,\lambda,\phi)}_{Reference} + \underbrace{\triangle U(r,\lambda,\phi)}_{Disturbance},$$

$$\ddot{\boldsymbol{r}} = \underbrace{-\frac{\partial U}{\partial \boldsymbol{r}}}_{Total} = \underbrace{-\frac{\partial}{\partial \boldsymbol{r}}\left\{\frac{GM}{r}\left[1-\frac{3}{2}J_2\left(\frac{R_\oplus}{r}\right)^2\left(3\sin^2(\phi)-1\right)\right]\right\}}_{Reference} \underbrace{-\frac{\partial}{\partial \boldsymbol{r}}\{\triangle U\}}_{Disturbance}. \qquad (3.2)$$

Since the reference potential is compact and efficient, and contains the macroscopic global gravity model, our finite element model approximation is applied only to the perturbative disturbance gravity.

III.B. Finite Element Model

As a specific example FEM grid, a sphere of radius $R_\oplus$ is covered by a 2-D mesh (4×4) degree $(\lambda, \phi) : 0 \le \lambda \le 360°; -88° \le \phi \le 88°$ cellular grid, except for the polar caps of angular radius 2 degrees. We mention that this is a *for example FEM grid* for illustration purposes. At arbitrary $r\,\{r_{\min} = R_\oplus \le r \le r_{\max} = 7R_\oplus\}$, a large family of spherical shells is sampled using the cosine distribution Eq.(3.3), (3.4). Let the gravity data $U(r = \text{constant}, \lambda, \phi)$ on a given spherical shell be transformed into $U(\xi = \text{constant}, \zeta_i, \eta_j)$ where $\{-1 \leq \zeta_i \leq +1\}$ and $\{-1 \leq \eta_j \leq +1\}$ with the sample points located according to the cosine distribution. We can conceive of the the gravity modeling as representing gravity $U(\zeta, \eta)$ accurately on a "sufficiently dense" set of concentric spherical surfaces. The radial coordinate variation is unique (compared to λ, ϕ) because of the $1/r^n$ terms.

III.B.1. Radial Smart Sampling

The transformed position (r) obeys smart "cosine-like" transformation as a function of the transformed radial variable ξ:

$$r = r_{\min} + (r_{\max} - r_{\min}) \left[1 - \cos\left(\frac{\pi}{4}(1+\xi)\right)\right], \ \xi = \frac{4}{\pi}\cos^{-1}\left(1 - \frac{r - r_{\min}}{r_{\max} - r_{\min}}\right) - 1, \tag{3.3}$$

$$\text{where} \left\{ \begin{array}{c} -1 \leq \xi \leq +1 \\ r_{\min} \leq r \leq r_{\max} \end{array} \right\}$$

where and $r_{\min} = R_{\oplus}$ and $r_{\max} = 7R_{\oplus}$. This transformation is intended, for uniform samples, to generate much denser r samples on the left (near $r_{\min}$) and less dense on the right (near $r_{\max}$) than the classical cosine sampling, see Figure III.1. The dense sampling near $\xi = -1$, $r = r_{\min} = R_{\oplus}$ ensures more dense measurements where the gravity perturbations are maximum and have the largest local significant differential changes. Since the gravity anomalies "die out" rapidly with increasing radius, less dense sampling is anticipated for increasing r. The transformed radius variable ξ of Eq. 3.3 was developed heuristically and is adopted in lieu of r as the independent variable. In order to satisfy the Chebyshev orthogonality conditions for radial approximations, ξ is actually sampled non-uniformly using the further cosine transformation:

$$\xi_j = -\cos(j\pi/2M), \; j = 0, 1, 2, ..., M. \tag{3.4}$$

III.B.2. Radial Adaption

It is important to determine the required polynomial order, as a function of radius, to adaptively maintain an approximation error tolerance. Radial order adaptation enables enormous speedups in the computation of the state of the art gravity models. For this insight, the required acceleration error tolerance is determined as a function of the polynomial order N. For instance, a (4×4) degree square area at the Earth's surface is sampled using 2D Chebyshev distribution. The convergence error is defined by the maximum absolute error between the truth (GRACE or EGM2008 model) and the approximation acceleration. It is chosen to be constrained by a maximum approximation error of $10^{-9}\,\mathrm{m\,s^{-2}}$.

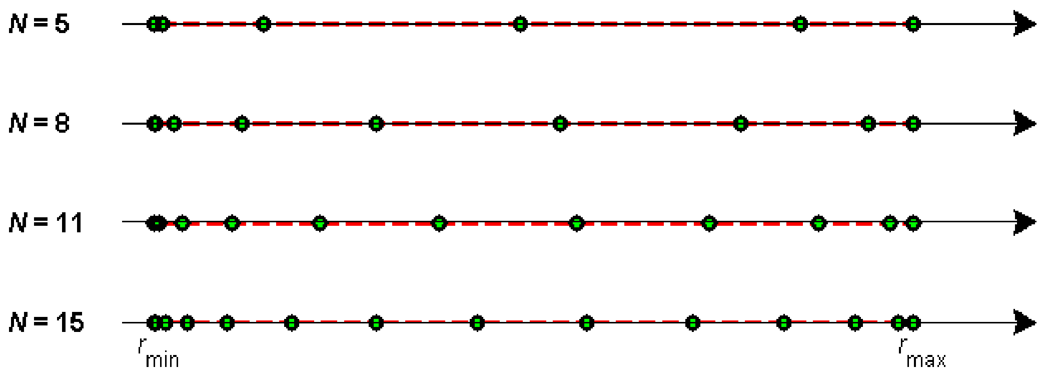


Figure III.1: Cosine-like Sampling for the Radial Distance. Note Density of Nodes is Highest Near r_{min}.

III.B.3. Orthogonal Approximation of the Gravitational Acceleration

Let the gravity data $U(r, \lambda, \phi)$ be transformed into $U(\xi_i, \zeta_j, \eta_k)$ where $-1 \leq \xi_i, \zeta_j, \eta_k \leq +1$ with the sample points located according to the cosine distribution. Note that the transformed position (r) obeys smart cosine-like transformation as a function of the transformed radial variable (ξ). We first review the approximation of the three components of the gravity acceleration:

$$\begin{aligned}
g_x(\xi_i, \zeta_j, \eta_k) &\cong \sum_{\alpha=0}^{N_x} \sum_{\beta=0}^{N_y} a_{\alpha\beta}(\xi_i)\phi_{\alpha\beta}(\zeta_j, \eta_k) = \Psi_x^T(\zeta_j, \eta_k)\boldsymbol{a}_x(\xi_i) \\
g_y(\xi_i, \zeta_j, \eta_k) &\cong \sum_{\alpha=0}^{N_x} \sum_{\beta=0}^{N_y} a_{\alpha\beta}(\xi_i)\phi_{\alpha\beta}(\zeta_j, \eta_k) = \Psi_y^T(\zeta_j, \eta_k)\boldsymbol{a}_y(\xi_i) \qquad (3.5) \\
g_z(\xi_i, \zeta_j, \eta_k) &\cong \sum_{\alpha=0}^{N_x} \sum_{\beta=0}^{N_y} a_{\alpha\beta}(\xi_i)\phi_{\alpha\beta}(\zeta_j, \eta_k) = \Psi_z^T(\zeta_j, \eta_k)\boldsymbol{a}_z(\xi_i)
\end{aligned}$$

where $(i, j, k)= 0, 1, \cdots, (M_r, M_\lambda, M_\phi)$. If we assume that we use the same basis functions for the acceleration components; i.e. $\Psi = \Psi_x = \Psi_y = \Psi_z$, then Eq. (3.5) is written in the following compact form:

$$\boldsymbol{g}(\xi_i, \zeta_j, \eta_k) = \begin{bmatrix} g_x(\xi_i, \zeta_j, \eta_k) \\ g_y(\xi_i, \zeta_j, \eta_k) \\ g_z(\xi_i, \zeta_j, \eta_k) \end{bmatrix} = \underbrace{\Big[(\Psi(\zeta_j, \eta_k) \otimes I_n) \otimes I_n\Big]}_{[\Upsilon]} \underbrace{\begin{bmatrix} \boldsymbol{a}_x(\xi_i) \\ \boldsymbol{a}_y(\xi_i) \\ \boldsymbol{a}_z(\xi_i) \end{bmatrix}}_{\boldsymbol{\rho}} \qquad (3.6)$$

or

$$\boldsymbol{g}(\xi_i, \zeta_j, \eta_k) = [\Upsilon(\zeta_j, \eta_k)]\boldsymbol{\rho}(\xi_i) \qquad (3.7)$$

III.B.4. Orthogonal Approximation of the Gravitational Potential

Here we consider approximating the gravity potential instead of the acceleration vector. We use the approximated potential to generate the higher derivatives (including the acceleration). Let us first discuss the consequences of this approach:

1. Accuracy: Approximation error increases by performing derivatives. This issue is treated by imposing a tolerance on the coefficients approximation. However, this is not a solution.

2. Memory: Approximating one variable (gravity potential) is much easier (and requires less memory) than approximating three variables (acceleration vector).

3. Speed: Fast approximation (as less coefficients are needed).

The approximation of the gravity potential is given by

$$U(\xi_i, \zeta_j, \eta_k) \cong \sum_{\alpha=0}^{N_x} \sum_{\beta=0}^{N_y} a_{\alpha\beta}(\xi_i)\phi_{\alpha\beta}(\zeta_j, \eta_k) = \Psi^T(\zeta_j, \eta_k)\boldsymbol{a}(\xi_i) \tag{3.8}$$

where the coefficients $a_{\alpha\beta}(\xi_i)$ are also fitted in the radial direction using the following approximation

$$a_{\alpha\beta}(\xi_i) \cong \sum_{\gamma=0}^{N_z} b_\gamma^{\alpha\beta}\phi_\gamma(\xi_i) = \Phi^T(\xi_i)\boldsymbol{b}^{\alpha\beta} \tag{3.9}$$

Equation (3.9) is written as

$$U(\xi_i, \zeta_j, \eta_k) \cong \sum_{\alpha=0}^{N_x} \sum_{\beta=0}^{N_y} \left\{ \sum_{\gamma=0}^{N_z} b_\gamma^{\alpha\beta}\phi_\gamma(\xi_i) \right\} \phi_{\alpha\beta}(\zeta_j, \eta_k) = \Psi^T(\zeta_j, \eta_k)\Phi^T(\xi_i)\boldsymbol{b}^{\alpha\beta} \tag{3.10}$$

Then higher derivatives are obtained by differentiating the approximation/polynomials in Eq. (3.10). For example, the acceleration coordinate systems are obtained from the potential using the classical gradient relationships in spherical or rectangular coordinates as

$$\begin{array}{ccc} \underline{\text{Spherical}} & & \underline{\text{Rectangular}} \\ \left\{ \begin{array}{c} \text{South}: \ G_S = -\frac{1}{r}\frac{\partial U}{\partial \phi} \\ \text{East}: \ G_E = -\frac{1}{r\, cos\phi}\frac{\partial U}{\partial \lambda} \\ \text{Radial}: \ G_R = \frac{\partial U}{\partial r} \end{array} \right\} & \Longleftrightarrow & \left\{ \begin{array}{c} G_x = \frac{\partial U}{\partial x} \\ G_y = \frac{\partial U}{\partial y} \\ G_z = \frac{\partial U}{\partial z} \end{array} \right\} \end{array}$$

III.C. GRACE Finite Element Representations

The Gravity Recovery And Climate Experiment (GRACE) has been publicly released [34, 35, 71]. Access to the model's coefficients and other descriptive files about GRACE were obtained from [35, 71]. For high precision orbit computation, we consider all $12,246$ terms out to $(m, n) = (156, 156)$ in the GRACE spherical harmonic model. We note that the gradient of Eq. (3.2), when computed using the spherical harmonic series for ΔU, generates almost $40,000$ terms to evaluate each local acceleration, and this is the motivation for local approximations. We further note, when we consider the $(200, 200)$ EGM 2008 model, $20,100$ terms are required to model the potential, whereas, representing the three components of acceleration requires over $60,000$ terms, an increase of 50% compared to the Grace Model.

As mentioned above, we selected a conservative maximum approximation error of $10^{-9}\,\mathrm{m\,s^{-2}}$ as the tolerance for errors in replacing the high degree and order gravity model by FEM approximation. Figure III.2 below shows maximum absolute error of the approximated "disturbance" acceleration (x, y, and z components) as function of Chebyshev polynomial order N. In this case, the reference gravity potential is simply the point mass term and the J_2 perturbation, everything else is approximated and plotted in (Figure III.4). The $(156, 156)$ GRACE model is adopted as the truth and the errors between the FEM approximation error are reduced by adjusting the degree of the Chebyshev polynomials to achieve convergence to an error smaller than the tolerance $10^{-9}\,\mathrm{m\,s^{-2}}$. At the Earth's surface, with the (4×4) degree FEM cell size, this is achievable at $N = 10$, whereas at $r_{\max} = 7R_{\oplus}$, the worst case error $10^{-9}\,\mathrm{m\,s^{-2}}$ is achieved with only a first degree $(N = 1)$ model for the gravity disturbance acceleration (as an additive local correction to the reference global model). This indicates that the local gravity perturbation potential at the Earth's surface is approximated by 121 orthogonal polynomial terms, whereas only a linear approximation of local disturbance acceleration is required at $r_{\max} = 7R_{\oplus}$. It is to be expected that the required polynomial order decreases monotonically as we move from the Earth's surface $r_{\min} = R_{\oplus}$ out to $r_{\max} = 7R_{\oplus}$, outside of the GEO radius: $r_{_{GEO}} = 6.623\,R_{\oplus}$; we found that only first degree polynomials are required at $r_{\max} = 7R_{\oplus}$, and thus only 4 polynomial terms are needed for all three components of acceleration. On a serial machine, the FEM approach is 2 orders of magnitude more computationally efficient at the Earth's surface than the $(156, 156)$ spherical harmonic expansion, and due to the radial adaptation feature of this approach, the FEM computational cost is, remarkably, reduced an additional ~ 2 orders of magnitude for $1.02\,R_{\oplus} < r < r_{\max} = 7\,R_{\oplus}$. This means for routine exo-atmosphere orbit calculations, the FEM gravity model is computed with 9 to 10 digit accuracy with a 4 order of magnitude reduction in CPU time, relative to using the correspondingly accurate spherical harmonic representation. Radial adaptation of the FEM computation is readily implemented by a one-time a priori computational process (at the time the FEM model is established) to find the maximum degree to maintain a prescribed accuracy tolerance, as a function of radial displacement through each FEM element. For redundant least squares representation using the Chebyshev methods developed earlier in chapter II, the residuals associated with approximation of gravity on the family of spherical shells, together with the numerical size of the coefficients, are readily exploited to establish $N(r)$ that guarantees the prescribed accuracy when computing acceleration from formulas analogous to Eq (2.42). The results for are shown in Figures III.5a and III.5b, so the maximum degree required quickly decreases from 10 at the surface of the Earth down to 2 at radial distances greater than $1.04\, R_{\oplus}$.

The $\sim$ 4 order of magnitude computational speedup by this approach can be even further enhanced by introduction of parallelization in conjunction with the Chebyshev-Picard methods discussed below where many gravitational acceleration evaluations at judicious nodal points along a known approximate path can be simultaneously computed in an iterative path approximation algorithm. In most cases, these path approximations are found to converge over 2 to 3 orbits and therefore allow 2 or more orders magnitude additional speedup. Supercomputer orbit computation performance with a desktop computer is therefore possible by using a fusion of adaptive orthogonal FEM gravity approximation and the Chebyshev-Picard orbit path approximation method.

Figure III.3 shows radial disturbance acceleration on the Earth's surface from a FEM representation with 4×4 degree square using $N = 10^{\text{th}}$ degree approximation. The FEM model agrees with 10 significant digits everywhere with the parent

(156, 156) spherical harmonic model and the GRACE coefficients. Figure III.4 is the corresponding FEM disturbance potential on the Earth's surface which replaces the parent spherical harmonic series everywhere with > 10 accurate digits.

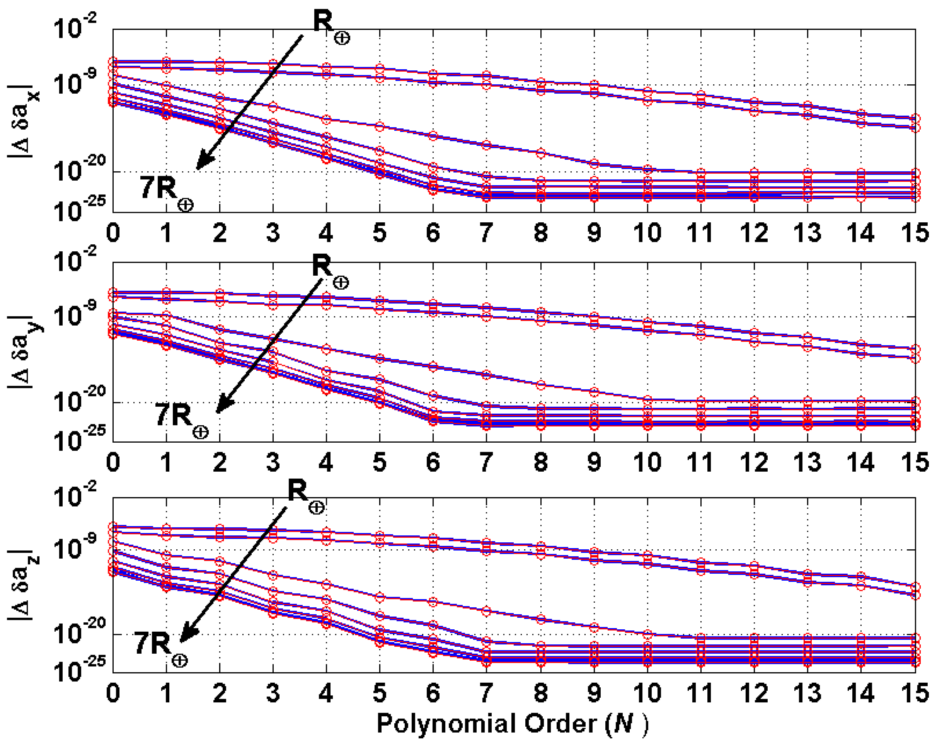


Figure III.2: (GRACE 156 × 156) Maximum Error of Chebyshev FEM Gravity Approximation ($\mathrm{m\,s^{-2}}$) as a Function of Polynomial Order N, for Various Radial Distances.

We discuss some other details of the FEM representation. As previously mentioned, a key step is using the classical cosine distribution for the transformed radius variable ξ, which is then mapped through the smart densification formula into r to satisfy the Chebyshev orthogonality conditions for radial approximations. This

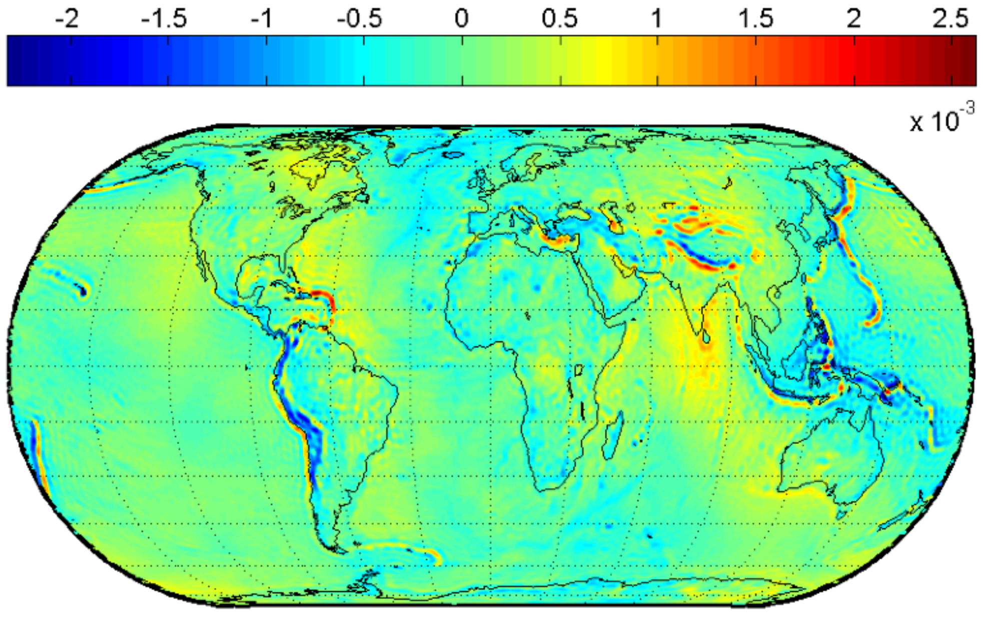


Figure III.3: (GRACE 156×156) Radial Perturbative FEM Gravity Approximation at the Earth's Surface ($\mathrm{m\,s^{-2}}$).

step is highly desirable to efficiently and accurately compute Chebyshev polynomial coefficients fits $a_{ij}(\xi(r))$ as a function of radius using the standard equations for the orthogonal Chebyshev functions. Observing Figure III.2, since, the gravity field within the range of interest $[r_{\min} = R_\oplus$ up to $r_{\max} = 7\,R_\oplus]$ has significant variation as r varies, it becomes useful to divide the model into two concentric spherical shell regions for the sake of FEM representation:

Region I Atmospheric region $r \in [R_\oplus, 1.02\,R_\oplus]$

Region II The mostly exo-atmospheric region $r \in [1.02\,R_\oplus, 7\,R_\oplus]$.

To establish FEM gravity modeling, each region is sampled on a family of spherical shells in each region using cosine-like measurement distribution of radii and a

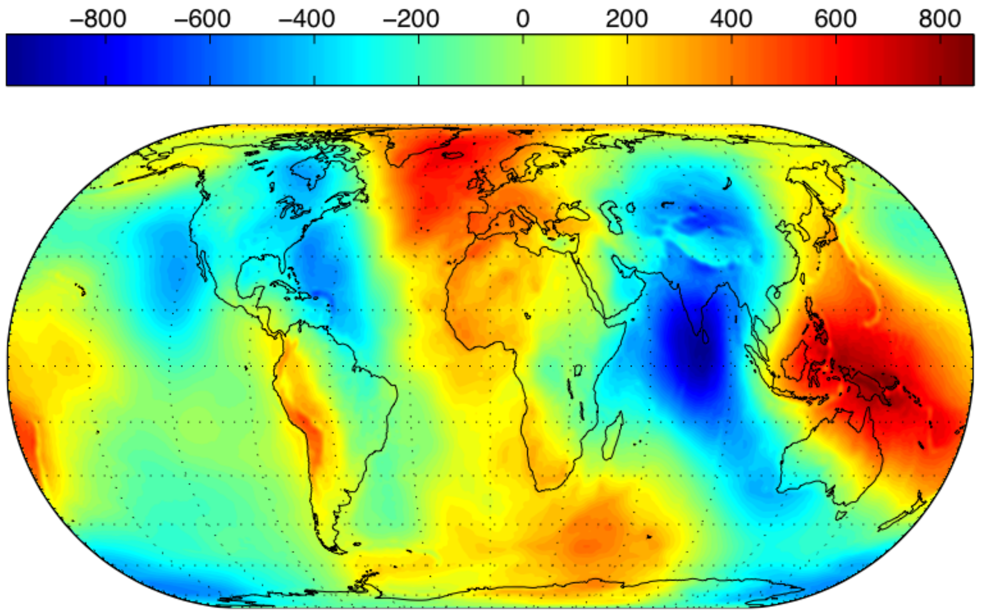


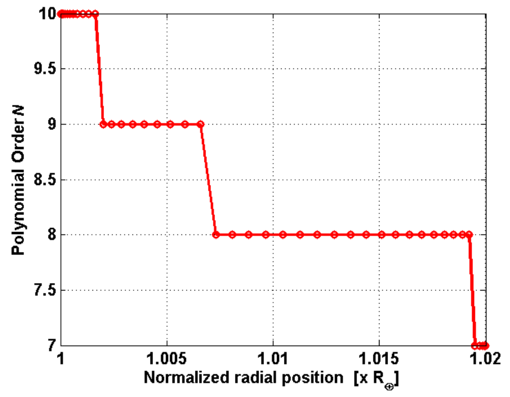
Figure III.4: (GRACE 156×156) Global FEM Gravity Potential Approximation ($\mathrm{m}^2\,\mathrm{s}^{-2}$).

family of $4^\circ \times 4^\circ$ (λ, ϕ) FEM elements are approximated over each spherical corresponding to a given r. The coefficients of the (λ, ϕ) FEM approximations of gravitational acceleration on each spherical surface boundary are considered a function of r, and this gives rise to a natural way to accomplish radial adaptation.

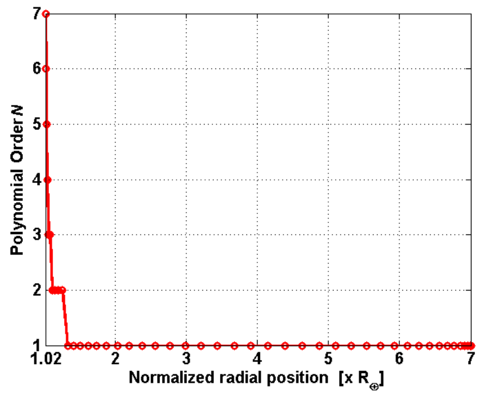
Figures III.5a and III.5b show the polynomial order required as a function of spherical shell radius r, over each of the two regions, to achieve maximum approximation error of $10^{-9}\,\mathrm{m\,s^{-2}}$. For Region I, the maximum polynomial order is found sufficient in the most anomalous spherical shell at the Earth's surface, while in Region II the maximum polynomial order $N = N_{\max} = 7$ is needed nearest the Earth, but a much smaller N is required at large r. To compute acceleration from the FEM model, we use the maximum polynomial order that ensures consistency with the full polynomial model, but retain for orbit computation only the $N_{rqrd}(r) < N_{\max}$ terms that contribute at that r. So for larger r, an order of magnitude of further computational speed improvement is achieved by only including the significantly non-zero terms. The inherent cosine sampling of the radial position allows us to use the standard equations for orthogonal least square Chebyshev approximation to obtain the polynomial coefficients. To demonstrate by example, Figures III.5e and III.5f show the polynomial coefficient $a_{00}(r)$ as a function of the radial position, @ $\lambda = 120°, \phi = 0°$. This first term follows the same behavior as the average disturbance acceleration x, y, and z components respectively, since all other basis functions have an average value of zero, see Figures III.5c and III.5d. All coefficients exhibit smooth behavior and are fit easily with low-polynomial order functions of ξ.

The computational speed of the FEM versus typical spherical harmonic GRACE $(156, 156)$ [35, 71] favors the highly accurate FEM approximation by about four orders of magnitude. Figure III.6 shows the more detailed computational comparison between the two representations of gravity. It is obvious that the computational speed of the FEM decreases as the required polynomial order decreases (i.e. further dramatic computational reduction as r increases).

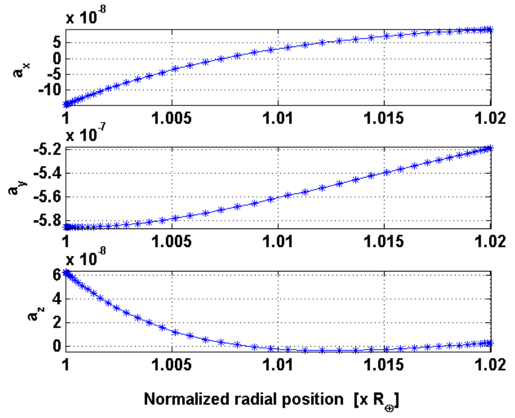
Alternative to approximate the gravity acceleration, one can interpolate the geopotential using the same approach, as discussed in subsection III.B.4. The GRACE 156×156) Global FEM Gravity Potential Approximation ($\mathrm{m}^2\,\mathrm{s}^{-2}$), at the earth's surface, is shown in Figures III.4 and C.1.



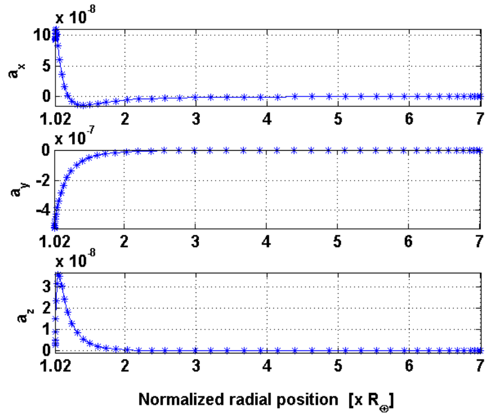
(a) Polynomial Order N versus r.



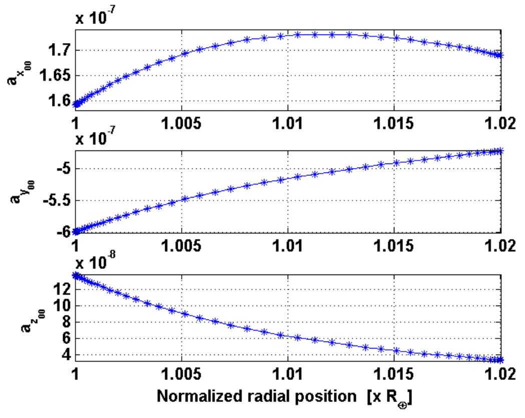
(b) Polynomial Order N versus r.



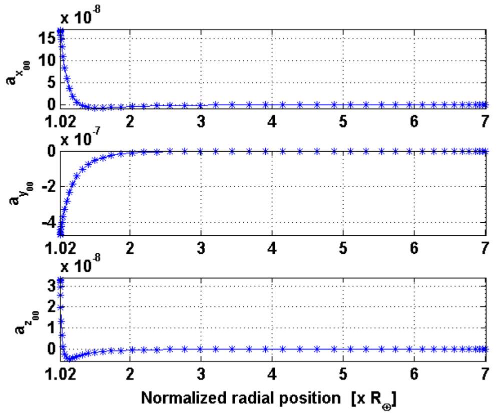
(c) Approx. Acceleration Components.



(d) Approx. Acceleration Components.

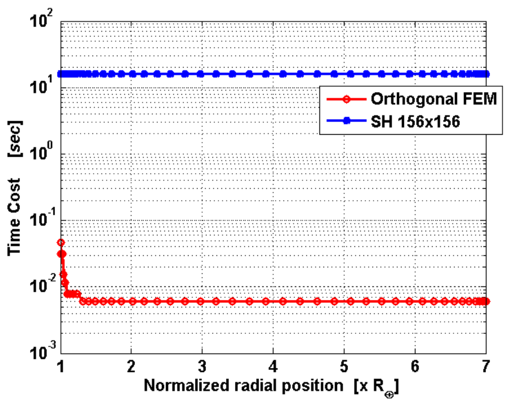


(e) Polynomial Coefficients a_{00} versus r. $r \in (R_{\oplus}, 1.02\,R_{\oplus})$.

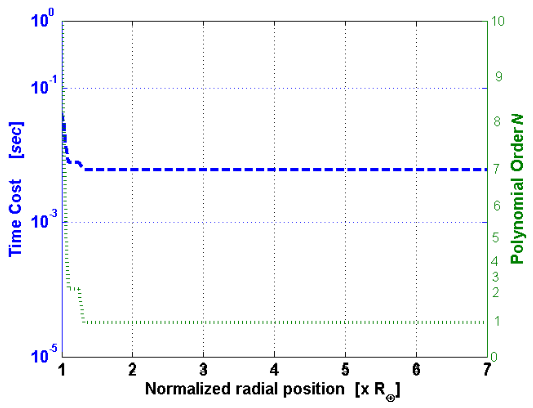


(f) Polynomial Coefficients a_{00} versus r. $r \in (1.02\,R_{\oplus}, 7\,R_{\oplus})$.

Figure III.5: (GRACE 156 × 156) Global FEM Gravity and Associated Polynomial Coefficients Approximation.



(a) Computational Speed.



(b) Computational Speed.

Figure III.6: (GRACE 156 × 156) Computation Speed of the FEM versus Spherical Harmonic.

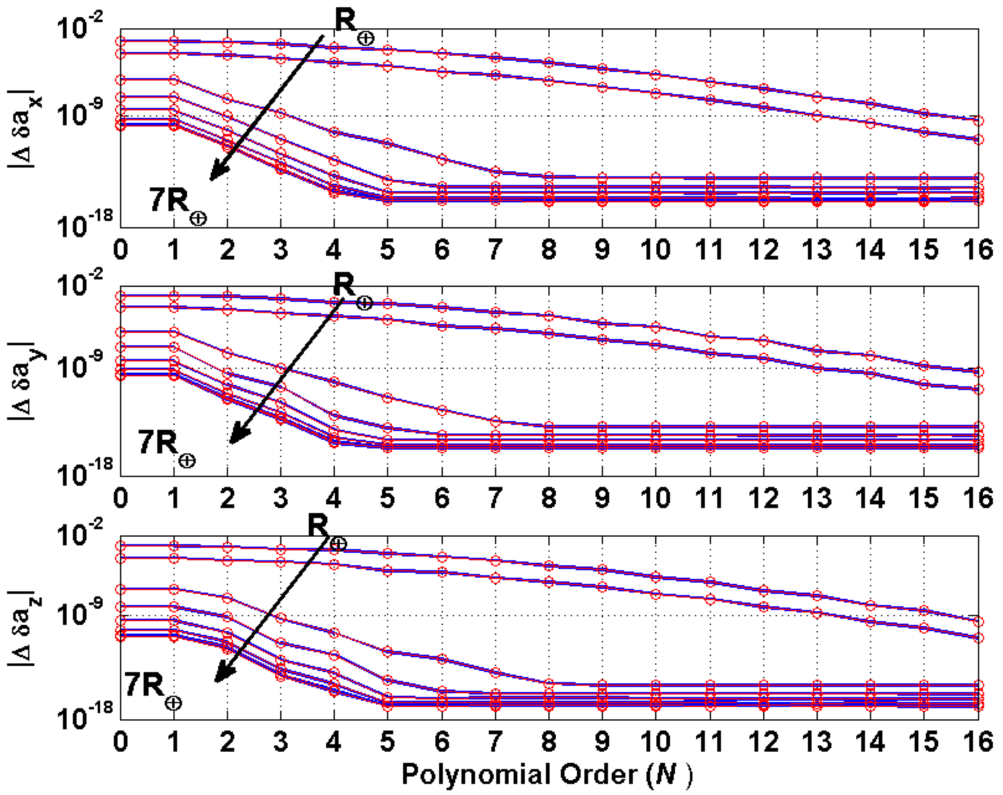


Figure III.7: (EGM2008 200 × 200) Maximum Error of Chebyshev FEM Gravity Approximation ($\mathrm{m\,s^{-2}}$) as a Function of Polynomial Order N, for various Radial Distances.

III.D. The EGM2008 Finite Element Representations

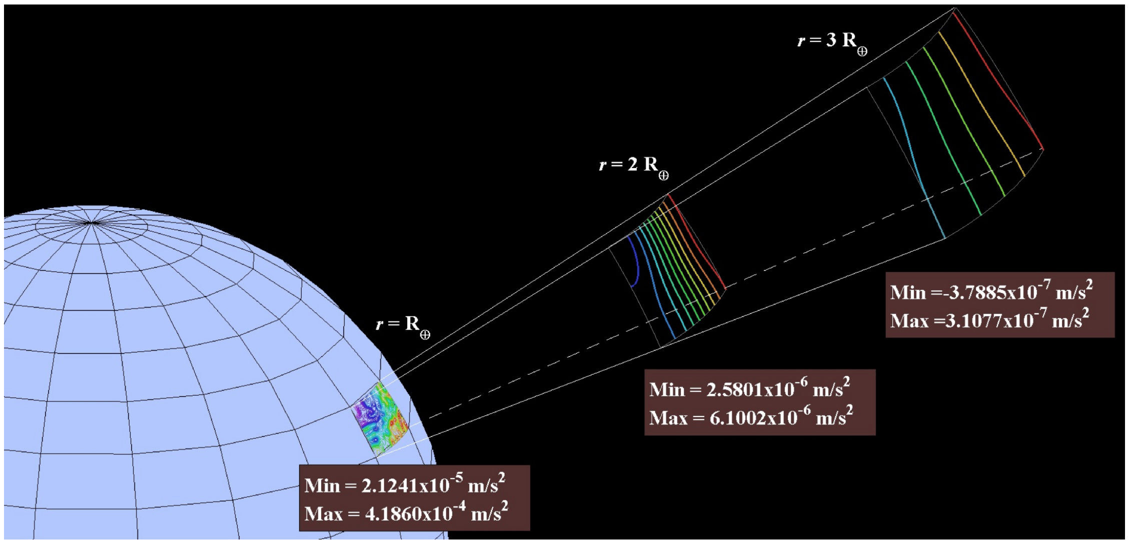
As another example, the Earth Gravity Model EGM2008 $(200, 200)$ FEM model has also been generated and the results, including detailed comparisons, are presented in Figures III.7-III.11. The official Earth Gravitational Model EGM2008 has been publicly released by the National Geospatial-Intelligence Agency (NGA) EGM Development Team. This gravitational model is complete to spherical harmonic degree and order 2159, and contains additional coefficients extending to degree 2190 and order 2159 [36]. Full access to the model's coefficients and other descriptive files about EGM2008 are obtained using the MATLAB® Aerospace toolbox. The $(200, 200)$ EGM2008 model was adopted as the truth and the errors between the FEM approximation error are reduced by adjusting the degree of the Chebyshev polynomials to achieve convergence to an error smaller than the tolerance $10^{-9}\,\mathrm{m\,s^{-2}}$. Figure III.7 below shows maximum absolute error of the approximated "disturbance" acceleration (x, y, and z components) as function of Chebyshev polynomial order N. At the Earth's surface, with the (4×4) degree FEM cell size, this is achievable at $N = 16$, whereas at $r_{\max} = 7\,R_{\oplus}$, the worst case error $10^{-9}\,\mathrm{m\,s^{-2}}$ is achieved with only a second degree ($N = 2$) model for the gravity disturbance acceleration. This indicates that the local gravity perturbation of the point mass potential at the Earth's surface is approximated by 289 orthogonal polynomial terms, whereas only a quadratic approximation is required at $r_{\max} = 7\,R_{\oplus}$. Also, we found that only $N = 2$ second degree polynomials are required at $r_{\max} = 7\,R_{\oplus}$, and thus only 9 polynomial terms are needed for all three components of acceleration. For redundant least squares representation using the Chebyshev methods developed earlier in this dissertation, the residuals associated with approximation of gravity on the family of spherical shells, together with the numerical size of the coefficients can be readily exploited to establish $N(r)$ to guarantee the prescribed accuracy when computing acceleration from formulas analogous to Eq (2.42). The results for are shown in Figures III.10a and III.10b, so the maximum degree quickly decreases from 16 at the surface of the Earth down to 2 at radial distances greater than $1.04\,R_{\oplus}$. Figure III.9

shows the global FEM gravity perturbative acceleration on the Earth's surface using $N = 16$. Of course, as r increases, the gravitational topography quickly becomes much smoother.

Observing Figure III.8, since, the gravity field within the range of interest $[r_{\min} = R_{\oplus}$ up to $r_{\max} = 7\,R_{\oplus}]$ results in significant differential changes along the radial position r, it becomes elegant to subdivide the problem into spherical shells based on the spectral content of gravity perturbations in each region. For instance, nearest the earth and within the Earth's atmosphere ($r \leq 1.02\,R_{\oplus}$), the gravity perturbations are strongest and due to the atmospheric drag, the overwhelming majority of orbit computations occur outside this spherical shell. This observation motivates the adopting of two spherical shells: (1) Atmospheric region $r \in [R_{\oplus}, 1.02\,R_{\oplus}]$, and (2) exo-atmospheric region $r \in [1.02\,R_{\oplus}, 7\,R_{\oplus}]$. For modeling the gravity perturbations, each region is sampled using cosine-like distribution to define a family of spherical shells and on each $r =$ constant shell a family of 4×4 degree FEM shells are approximated over each surface. While we can conceive of the number of such surfaces must approach infinity, we find that modeling surface gravity a finite number of spherical shells (< 50) and then using radial interpolation of the surface gravity coefficients provides excellent 3D approximation of gravity, approaching machine precision. We adopted 10 digit accuracy tolerance for the computations described below. Figures III.10a and III.10b show the polynomial order required for each region to achieve maximum approximation error of $10^{-9}\,\mathrm{m\,s^{-2}}$. For region I, the maximum polynomial order $N = N_{\max} = 16$ is used, while in region II the maximum polynomial order $N = N_{\max} = 13$ is used. We remark that the two dimensional approximations on each surface are "promoted" to three dimensional by considering their coefficients to be a function of radius and representing each coefficient as an orthogonal Chebyshev function of radius. However, as radius increases there are only some lower-order terms that are contributing when $N_{\mathrm{reqd}}(r) < N_{\max}$. This provides the motivation for *radial adaption* to retain only those terms that contribute significantly and enable another important computational speed improvement. The inherent cosine sampling of the radial position allows us to use the previously derived equations for orthogonal approximation of each the polynomial coefficients as a function of (transformed) radius. To capture qualitatively the rapid radial decay of the gravity anomalies in a typical finite element, refer to Figure III.8, where we show the radial gravitational acceleration perturbation contoured on three surface slices of a typical finite element, where the three spherical shell surface slices have radii $r = 1, 2$, and $3\,R_{\oplus}$. As a further illustration, Figures III.10e and III.10f show the free term polynomial coefficient $a_{00}(r)$ as a function of the radial position, @ $\lambda = 120°, \phi = 0°$. This free term is found to qualitatively follow the same asymptotic behavior of the average disturbance acceleration x, y, and z components respectively, see Figures III.10c and III.10d. All higher non-zero coefficients $a_{\alpha\beta}$ have been found to be analogous smooth functions that are fit easily with low-polynomial order orthogonal functions of $\xi(r)$. Observe that not only the irregularity of the gravity variations decays rapidly, but also, the norm of the gravity decays rapidly. This means, for example, that at most 3 significant figure approximation of the small radial gravity perturbations evident at $r = 3\,R_{\oplus}$ are required to achieve the $10^{-9}\,\mathrm{m\,s^{-2}}$ accuracy tolerance.

The computational speed of the FEM versus typical spherical harmonic EGM2008

(200, 200) [36] favors the FEM by about five orders of magnitude. Figure III.11 shows the comparison between the two computational methods. It is obvious that the computational efficiency of the FEM increases as the required polynomial order decreases. As is evident, comparing Figures III.6 and III.11, the relative advantage of using the FEM representation is greater by almost one order of magnitude for the (200, 200) model than for the (156, 156), due to the expense of evaluating 60, 000 terms of the (200, 200) model versus 30, 000 terms of the (156, 156) model.



$r = 3\ R_\oplus$
$r = 2\ R_\oplus$
$r = R_\oplus$
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 III.8: (EGM2008 200 × 200) Radial Gravity Contoured on Three Spherical Shells.

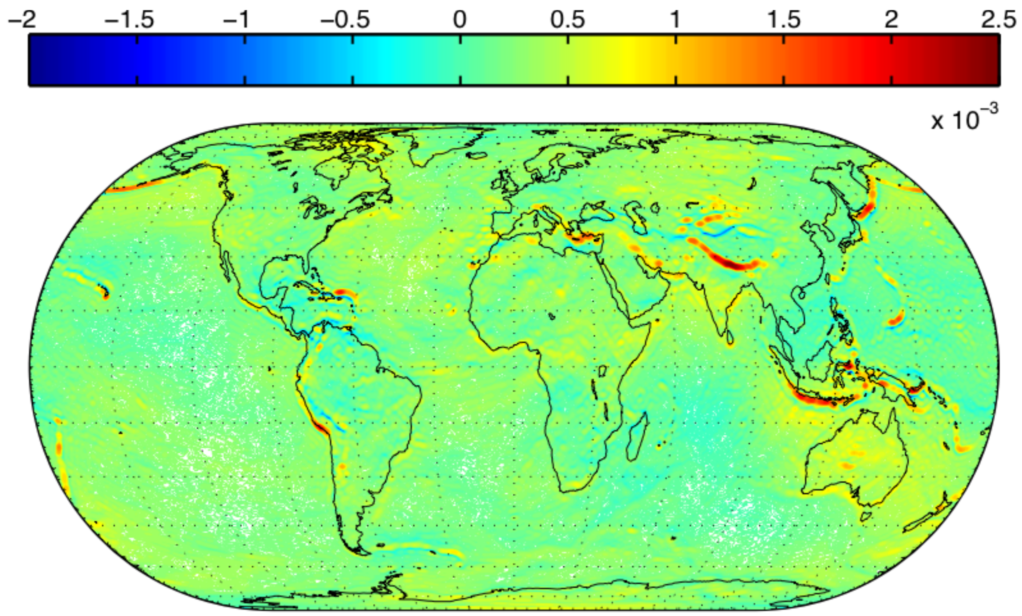
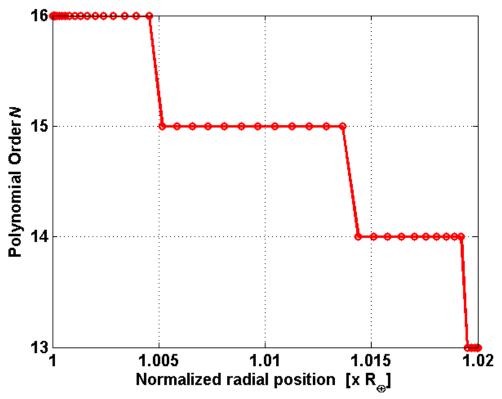
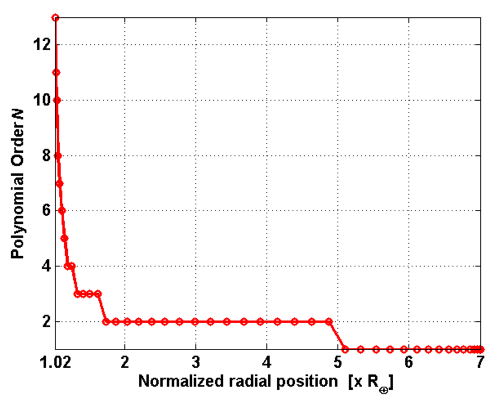
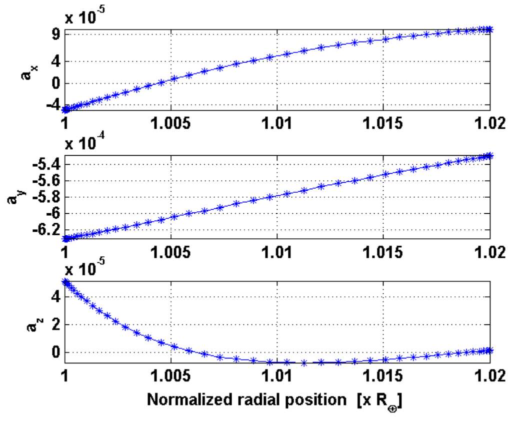
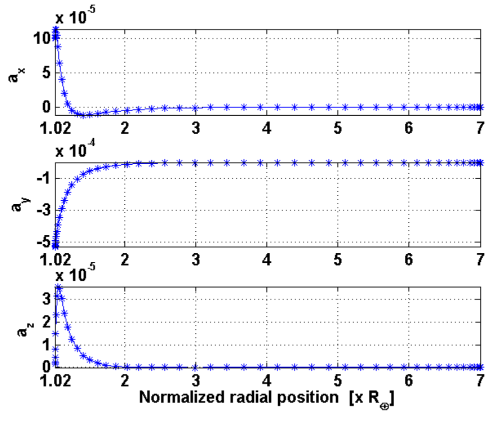


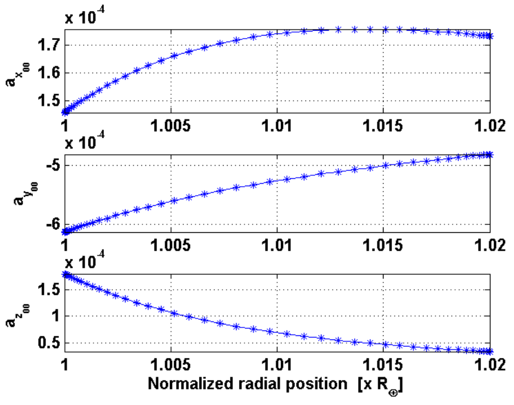
Figure III.9: (EGM2008 200 × 200) Radial Perturbative FEM Gravity Approximation at the Earth's Surface $[\mathrm{m\,s^{-2}}]$.


(a) Polynomial Order N versus r.


(b) Polynomial Order N versus r.


(c) Approx. Acceleration Components.


(d) Approx. Acceleration Components.


(e) Polynomial Coefficients a_{00} versus r, $r \in (R_{\oplus}, 1.02\, R_{\oplus})$.

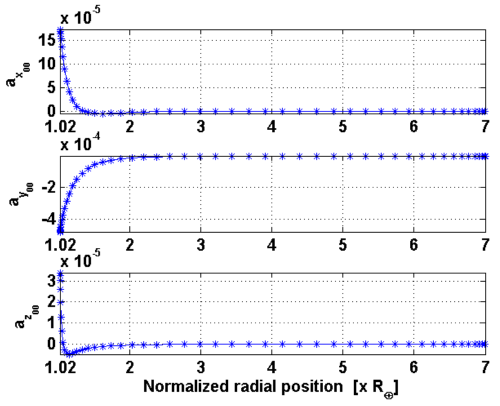
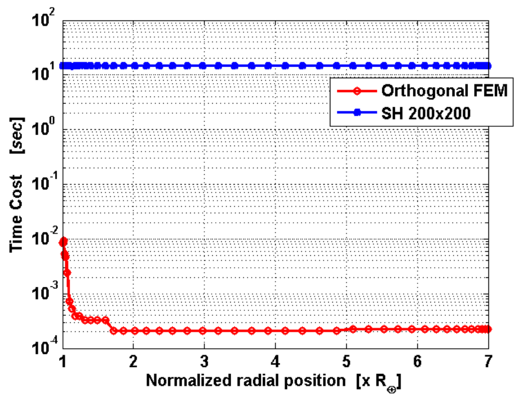
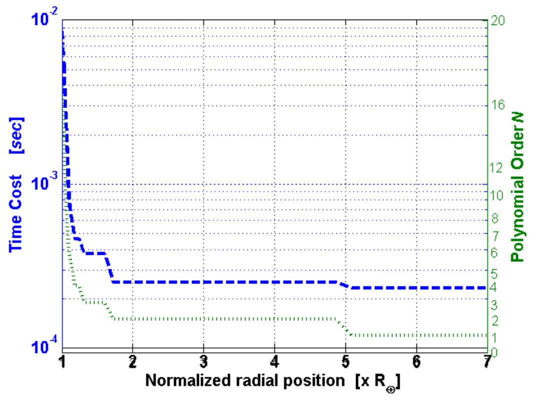

(f) Polynomial Coefficients a_{00} versus r, $r \in (1.02\, R_{\oplus}, 7\, R_{\oplus})$.

Figure III.10: (EGM2008 200×200) Global FEM Gravity and Associated Polynomial Coefficients Approximation.



(a) Computational Speed.



(b) Computational Speed.

Figure III.11: (EGM2008 200×200) Computation Speed of the FEM versus Spherical Harmonic.

III.E. Memory Saving

One way to generate the finite element volumes on a sphere is to begin with uniform $(\Delta\lambda, \Delta\Phi)$ region using the classical spherical coordinate system. Figure III.12 shows uniform $(\Delta\lambda, \Delta\Phi)$ patches and the coordinate system on which the spherical coordinates are based. Note that ϕ is constrained to vary over π, i.e. the difference between ϕ_{max} and ϕ_{min} is no more than π. In a similar manner, λ is constrained over 2π and as such the difference between λ_{max} and λ_{min} is no more than 2π.

The spherical patch size $(\Delta\phi \times \Delta\lambda)$ of the finite-element base over the Earth's surface together with the accuracy tolerance, determine the number of the polynomial order required. This indicates that the number of coefficients in each cell is also a function of the cell size. Figure III.13 shows the required polynomial order N

to maintain a certain approximation error tolerance for a finite element size. It is clear that at certain error tolerance, the number of approximating coefficients increases as the size of the finite-element cell increases. This is basically the key factor to determine how many coefficients for each cell are required to approximate the gravity model (truth), maintaining the maximum approximation error below a specific tolerance. Therefore, we require more memory for the coefficients as we decrease the approximation tolerance. Note that near the poles, the finite element size is smaller in the east-west direction than near the equator. For example, if we set the size at the equator to produce approximation error less than a specific tolerance, then the approximation error will further decrease as we move north or south. In other words, near the poles we may need not need to use the same number of coefficients as we use near the equators. For the finite element model, the total number of cells over the Earth's is $90 \times 44 = 3960$, if the classical spherical coordinate systems is used. To save memory, this number of cells needs be reduced, while the required approximation tolerance is maintained. This motivated us to explore the following two directions: (1) Map equalled-area cells all over the earth's surface, and (2) Perform north/South adaption.

III.E.1. Equalled-Area Shells

A new method for partitioning a unit sphere into regions of equal area is presented by Leopardi [72]. His paper describes the recursive zonal equal area (EQ) partition of the unit sphere $S^d \subset R^{d+1}$, where d is the dimension. The unit sphere

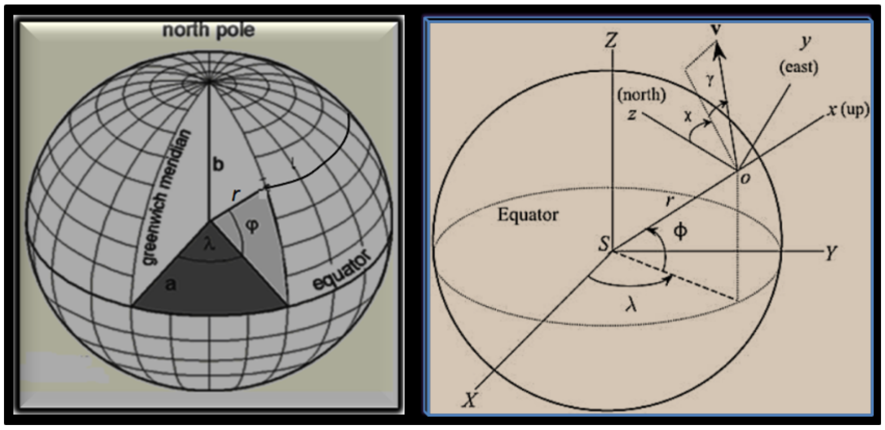


Figure III.12: Coordinate System.

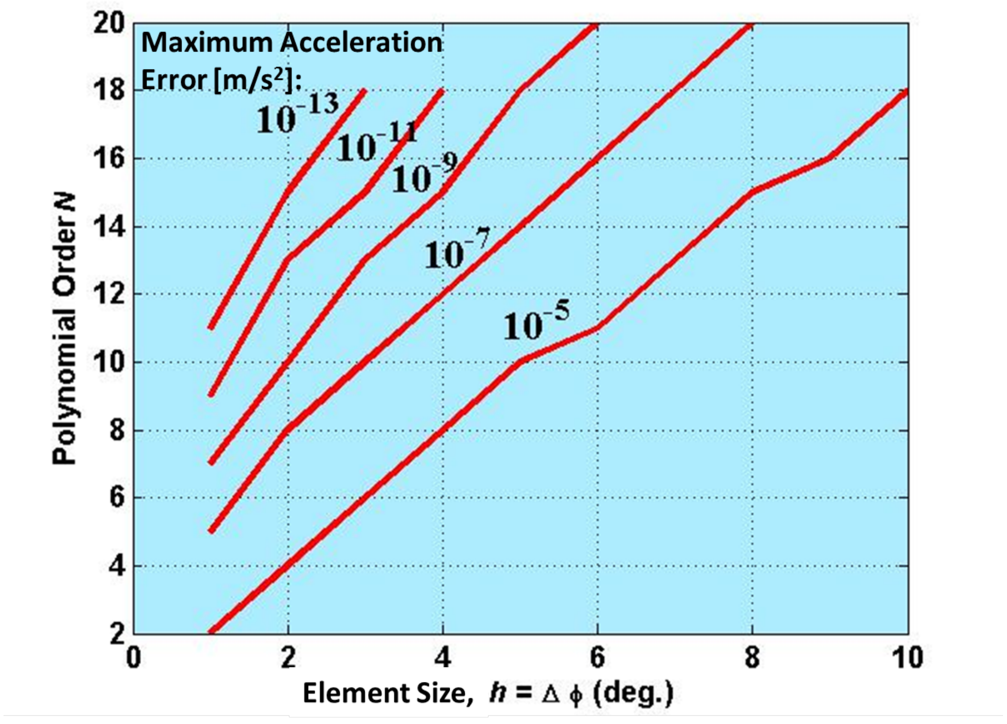
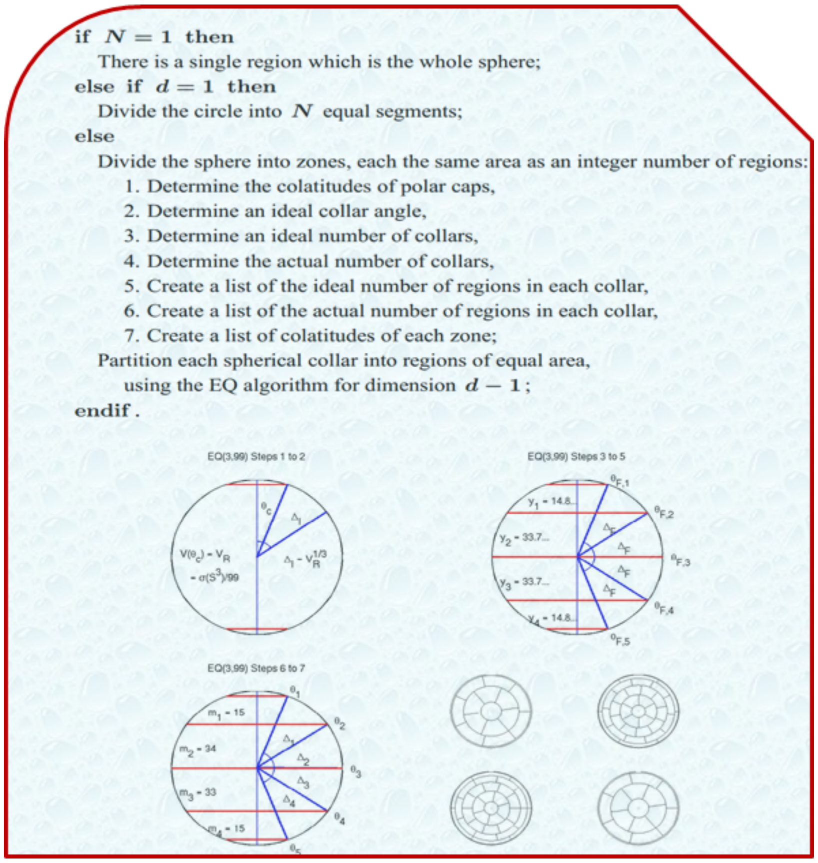


Figure III.13: (EGM2008 200×200): Polynomial Order N versus Finite Element Size at Various Approximation Tolerances.

S^d is defined as

$$S^d = \{x \in R^{d+1} \ \mid \sum_{k=1}^{d+1} x_k^2 = 1\} \tag{3.11}$$

The partition EQ(d;N) is a partition of the unit sphere S^d into N regions of equal area and small diameter. The EQ algorithm is shown in Figure III.14. Examples of



```
if  $N = 1$  then
    There is a single region which is the whole sphere;
else  if  $d = 1$  then
    Divide the circle into  $N$  equal segments;
else
    Divide the sphere into zones, each the same area as an integer number of regions:
        1. Determine the colatitudes of polar caps,
        2. Determine an ideal collar angle,
        3. Determine an ideal number of collars,
        4. Determine the actual number of collars,
        5. Create a list of the ideal number of regions in each collar,
        6. Create a list of the actual number of regions in each collar,
        7. Create a list of colatitudes of each zone;
    Partition each spherical collar into regions of equal area,
        using the EQ algorithm for dimension  $d - 1$ ;
endif .
```

Figure III.14: Outline of the EQ Algorithm.

equalled shells are shown in Figure III.15.

Note that the sphere is divided into equal area collars in the latitude direction ($\frac{88\ +\ 88\ degree}{4\ degree}$ = 44 collar + North Pole cap + South Pole cap = 46). Each collar is divided into equal areas cells that are equal to theareaof the North Pole cap

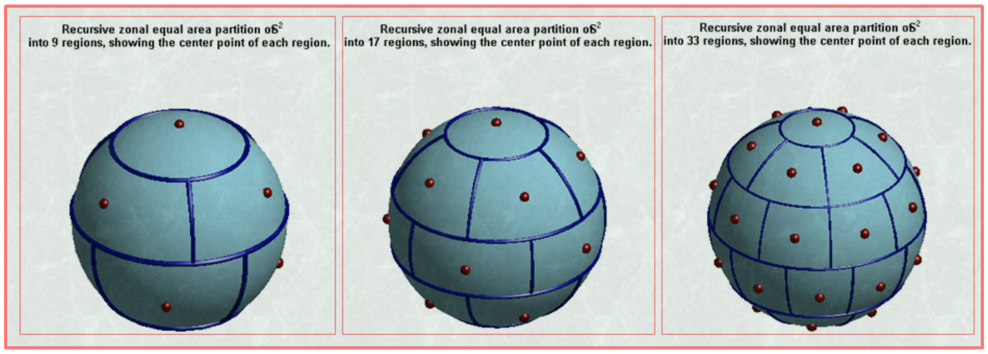

Recursive zonal equal area partition of S^2 into 9 regions, showing the center point of each region.
Recursive zonal equal area partition of S^2 into 17 regions, showing the center point of each region.
Recursive zonal equal area partition of S^2 into 33 regions, showing the center point of each region.

Figure III.15: Examples of EQ Shells for $N = 9,\ N = 17,\ N = 33$.

(and South Pole cap). So, each cell on the sphere occupies the same area as the North Pole cap (and South Pole cap).We modified Leopardi's [72] algorithm to align the south/north cell boundary with the Greenwich meridian, at $\lambda = 0$. Equal area mapping versus classical spherical mapping is shown in Figure III.17. In order to evaluate the memory saving, we assume that the spherical size of the finite-element cell near the equator in the Earth's surface is chosen to be ($\Delta\phi \times \Delta\lambda$, $4\ degree \times 4\ degree$). This results in the number of finite elements at the equator being $360/4 = 90$. This number of finite elements will decrease gradually as latitude increases. The number of cells in each collar is plotted in Figure III.16. Thus, the total number of cells to cover the earth's surface is 2578.

III.E.2. North-South Adaption

Similar to the radial adaption, we can perform North-South adaption, where the required polynomial order decreases in these directions. A simple test is performed at the greenwich meridian for cells between $\lambda = 0$ to 4 degrees. Figure III.18 shows the number of required polynomial order N_ϕ as a function of the latitude direction

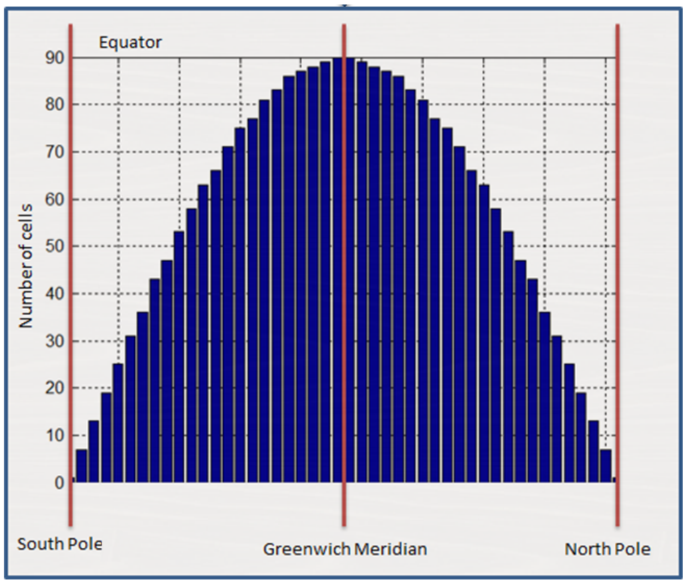


Figure III.16: Number of Cells in Each Collar.

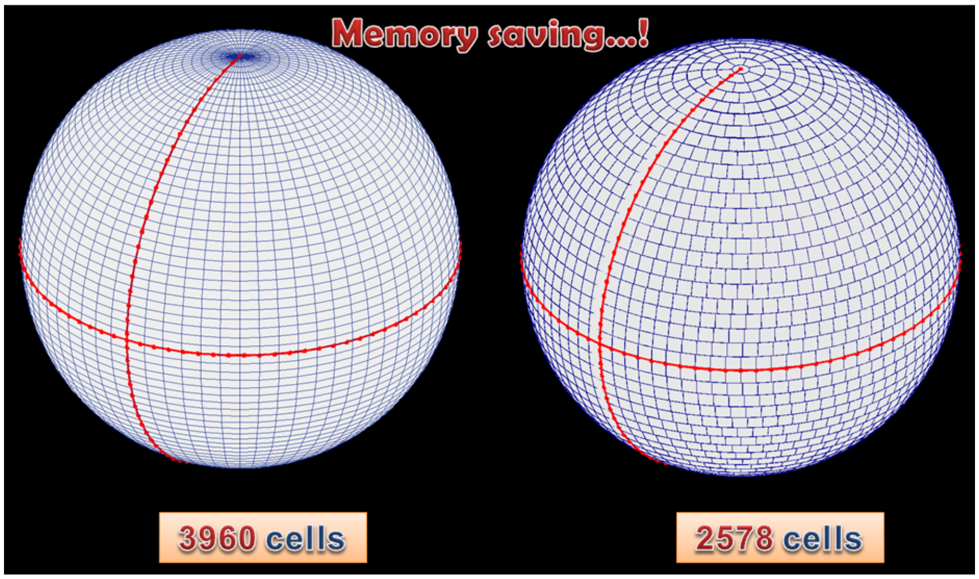


Figure III.17: Equal Areas Mapping versus Classical Spherical Mapping.

at the earth's surface. It is obvious that the polynomial order varies along the north-south direction. This implies that less coefficients are needed to get the same approximation accuracy as we move towards the poles.

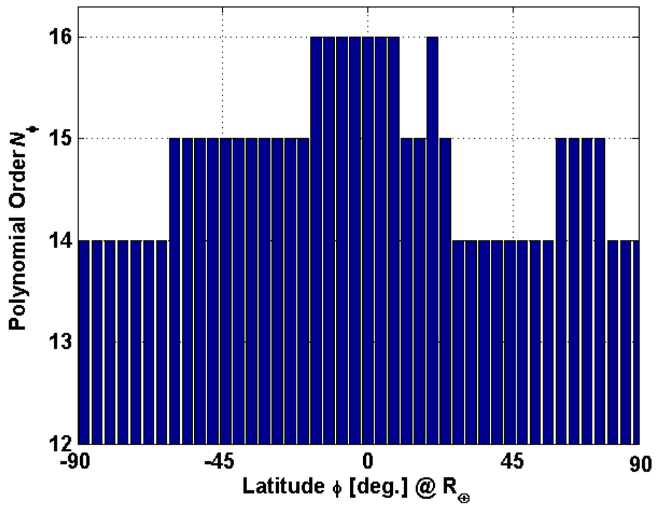


Figure III.18: Number of Required Polynomial Order N_ϕ as a Function of the Latitude at the Earth's Surface.

III.F. Accessing the FEM Coefficients

After performing the finite element approximation, we save the radial coefficients, the ones that we fit the Longitude/Latitude coefficients, $a_{\alpha\beta}(\xi_i)$. For example, for the EGM2008 FEM we use $N_\lambda = N_\phi = 16$; which indicates that we need $(16+1)^2 = 289$ coefficients to fit the gravitational accelerations over the earth surface. For the region (I): $r \in [R_\oplus, 1.02R_\oplus]$, the required radial polynomial to maintain the approximation error tolerance 10^{-9} is $N_r = 9$. For the region (II): $r \in [1.02R_\oplus, 7R_\oplus]$, the required radial polynomial to maintain the approximation error tolerance 10^{-9} is $N_r = 37$. Thus the total number of coefficients required to be saved for each acceleration component is calculated as follows:

1. For Region(I): Number Of Coefficients = (90 × 44 Cells) × (17 × 17 Coefficients/Cell) × (10 Radial Coefficients)

2. For Region(II): Number Of Coefficients = (90 × 44 Cells) × (14 × 14 Coefficients/Cell) × (38 Radial Coefficients)

The radial coefficients are stored in a block form (as shown in Figure III.19), where at each cell (λ, ϕ) on the earth's surface, we store the radial coefficients; 10 for region (I), and 38 for region (II). This structure enables us to easily look up the corresponding coefficients at each 3-D location. For example, for a given position

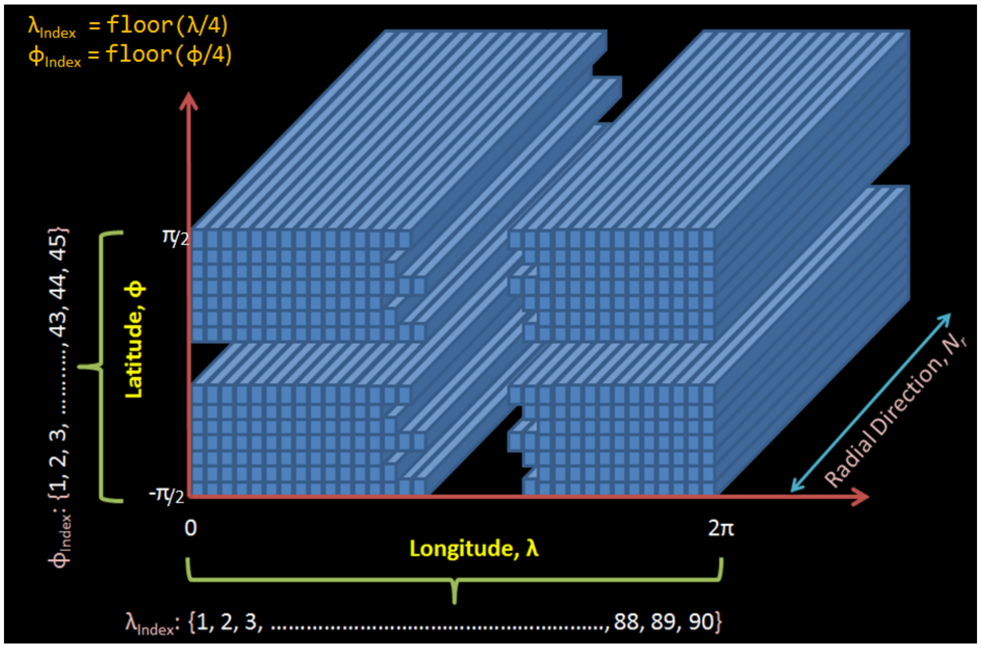

λ_{Index} = floor(λ/4)
ϕ_{Index} = floor(ϕ/4)
π/2
-π/2
Latitude, φ
ϕ_{Index}: {1, 2, 3, ..., 43, 44, 45}
0
2π
Longitude, λ
λ_{Index}: {1, 2, 3, ..., 88, 89, 90}
Radial Direction, N_r

Figure III.19: Radial Coefficients Look-up Data Structure.

$[X, Y, Z]$, the corresponding $[r, \lambda, \phi]$ is calculated. This is used to determine the finite cell "storing" the required coefficients for the approximation. $[\lambda_{Index}, \phi_{Index}]$ are identified by applying the following formula (note that the finite element size is 4 degree $\times$ 4 degree):

$$\lambda_{Index} = \texttt{floor}(\frac{\lambda}{4})$$
$$\phi_{Index} = \texttt{floor}(\frac{\phi}{4})$$

where the `floor`(a) is a function that rounds object (a) to the nearest integer in the direction of negative infinity.

III.G. Post-processing the FEM Coefficients

It is clear that interpolating the gravitational acceleration trades large memory compared to interpolating the geopotential. Our goal is to efficiently trade higher memory for faster runtime. In addition to the methods discussed in III.E, one can perform a screening check to remove noncontributing coefficients; i.e. removing all the coefficients that have absolute values less than a given tolerance. If *tol.* $= 10^{-9}$ is considered, then there is about a 79% coefficient saving for each component in the first region, $r \in [R_\oplus, 1.02\, R_\oplus]$, and about an 88% coefficient saving for each component in the second region, $r \in [1.02\, R_\oplus, 7\, R_\oplus]$. The total number of coefficient saving is shown in Figure III.20 and Tables III.1 and III.1. This is a huge memory saving when only 21% of the original set of coefficients for first region, and 12% for the second region are required producing an approximation that satisfies the desired accuracy level.

The benefit of this optimization is efficiently reducing the memory size associated with the interpolating coefficients, and subsequently reducing the runtime cost. The

Table III.1: Number of Coefficients Saved, $r \in [R_\oplus, 1.02\, R_\oplus]$.

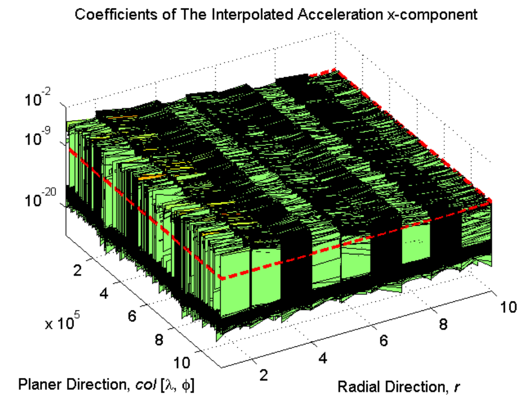
Total Number of Coefficients	x component	y component	z component
All coefficients:	11444400	11444400	11444400
Coefficients $< 10^{-9}$:	8987021	8991224	8980165
Coefficients $\geq 10^{-9}$	2457379	2453176	2464235
Saving %	78.53	78.56	78.47

Table III.2: Number of Coefficients Saved, $r \in [1.02\, R_\oplus, 7\, R_\oplus]$.

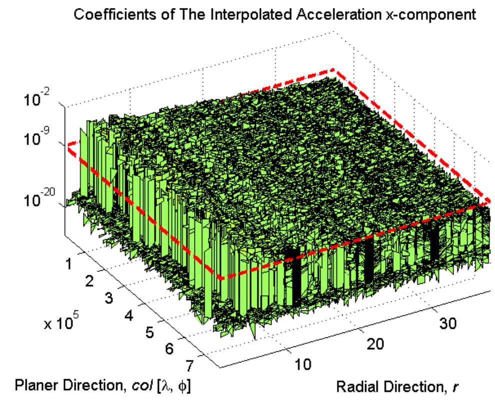
Total Number of Coefficients	x component	y component	z component
All coefficients	29494080	29494080	29494080
Coefficients $< 10^{-9}$	25850543	25862276	25832960
Coefficients $\geq 10^{-9}$	3643537	3631804	3661120
Saving %	87.65	87.69	87.59

coefficients storage is reduced to 136 MB. The plots shown in Figure III.20 exhibit the interpolating coefficients viewed in the longitude/latitude direction versus radial direction. The figures show that a big portion of the interpolating coefficients has an absolute magnitude less than 10^{-9}, which is represented by dotted red line plane. The coefficients in the first region can be interpolated by chebyshev polynomials of order nine. In the second region, the approximation can be truncated at $N_r = 35$. All extra coefficients are removed form the final *look up* table.

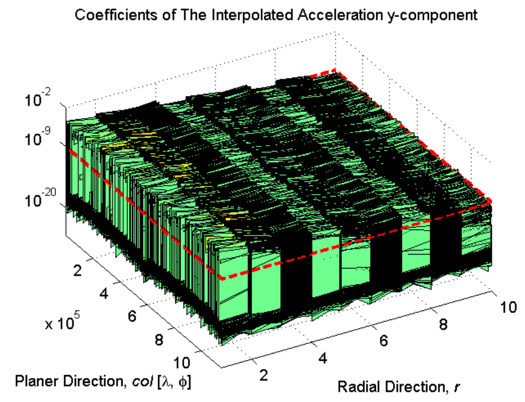
Figures III.21, III.22, and III.23 show the distribution of radial coefficients that interpolate the x-component, y-component, and z-component acceleration, respectively, in the region $r \in (R_\oplus, 1.02\,R_\oplus)$. Figures III.24, III.25 and III.26 show the distribution of radial coefficients that interpolate the x-component, y-component, and z-component acceleration, respectively, in the region $r \in (1.02\,R_\oplus, 7\,R_\oplus)$. Each FEM cell is plotted in the (λ, ϕ) coordinate. As previously mentioned, the $N_\lambda \times N_\phi$ coefficients within each cell are fitted in the radial direction. Although the location of each cell is spatially dependent on λ and ϕ, the value of required coefficients to approximate the radial direction is not spatial dependent. Thus the color coding in each cell represents the variance of the required number of coefficients at all points tested within that cell for approximating the radial direction. This ranges from blue (minimum number of coefficients) at the lower left of each cell to red (maximum number of coefficients) at the upper right, hence forming a surface displaced in each cell.



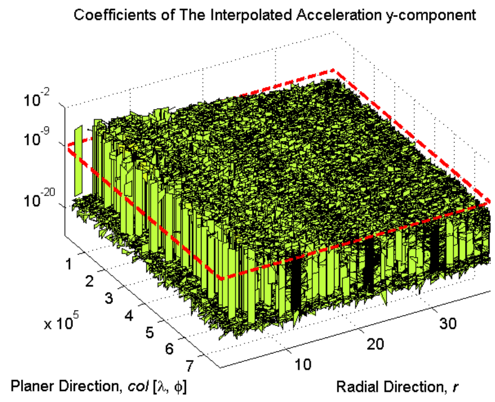
(a) x-component.



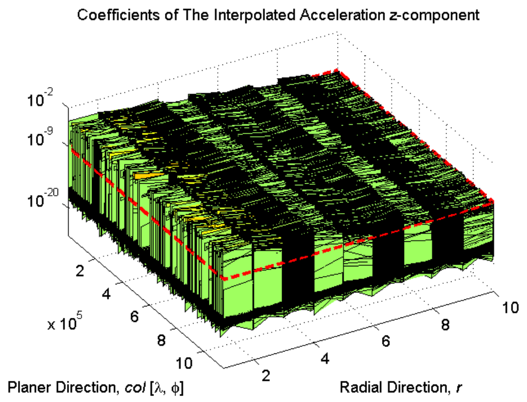
(b) x-component.



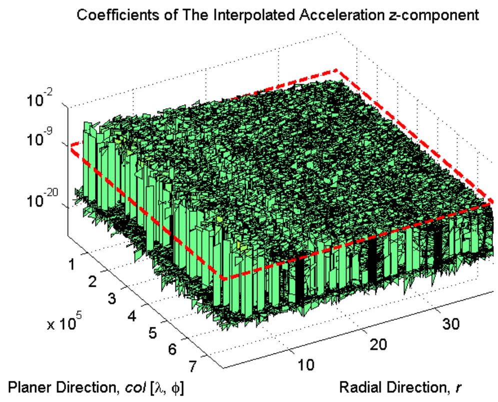
(c) y-component.



(d) y-component.



(e) z-component.
$r \in (R_\oplus, 1.02\, R_\oplus)$



(f) z-component.
$r \in (1.02\, R_\oplus, 7\, R_\oplus)$

Figure III.20: Number of Polynomial Coefficients.

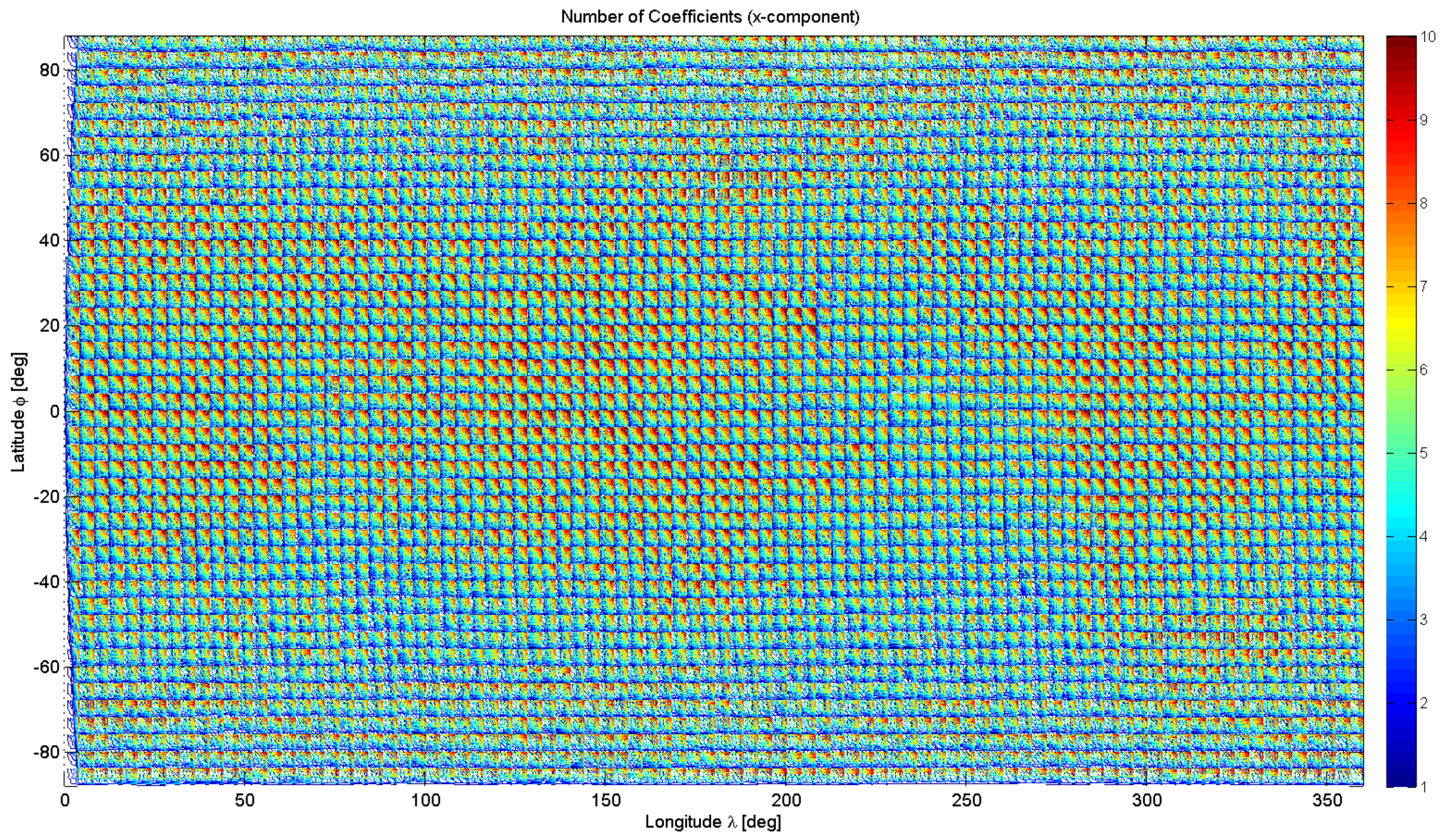


Figure III.21: Distribution of Number of Radial Coefficients Interpolating the x Component Acceleration $r \in (R_\oplus, 1.02\,R_\oplus)$.

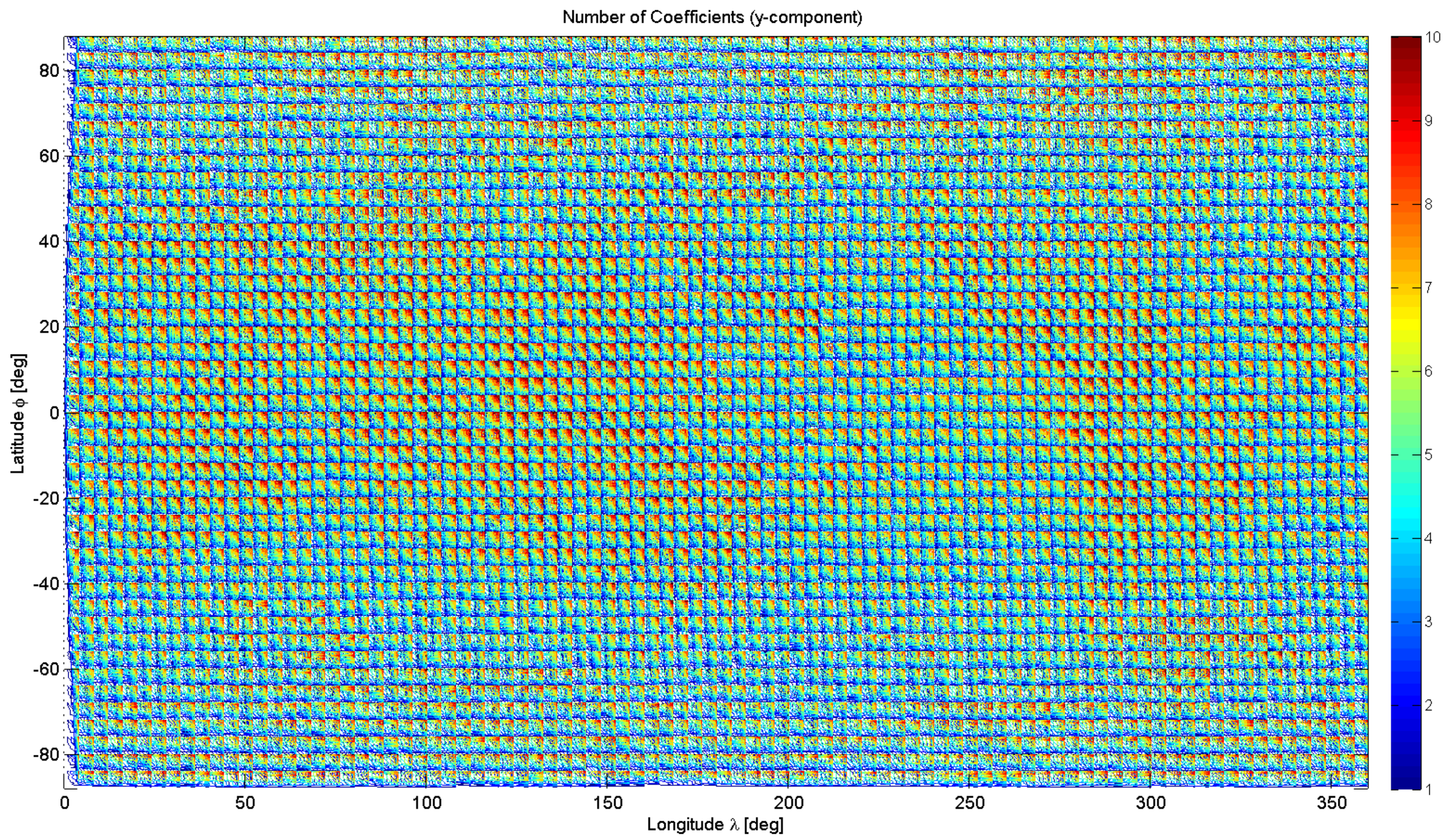


Figure III.22: Distribution of Number of Radial Coefficients Interpolating the y Component Acceleration $r \in (R_\oplus, 1.02\, R_\oplus)$.

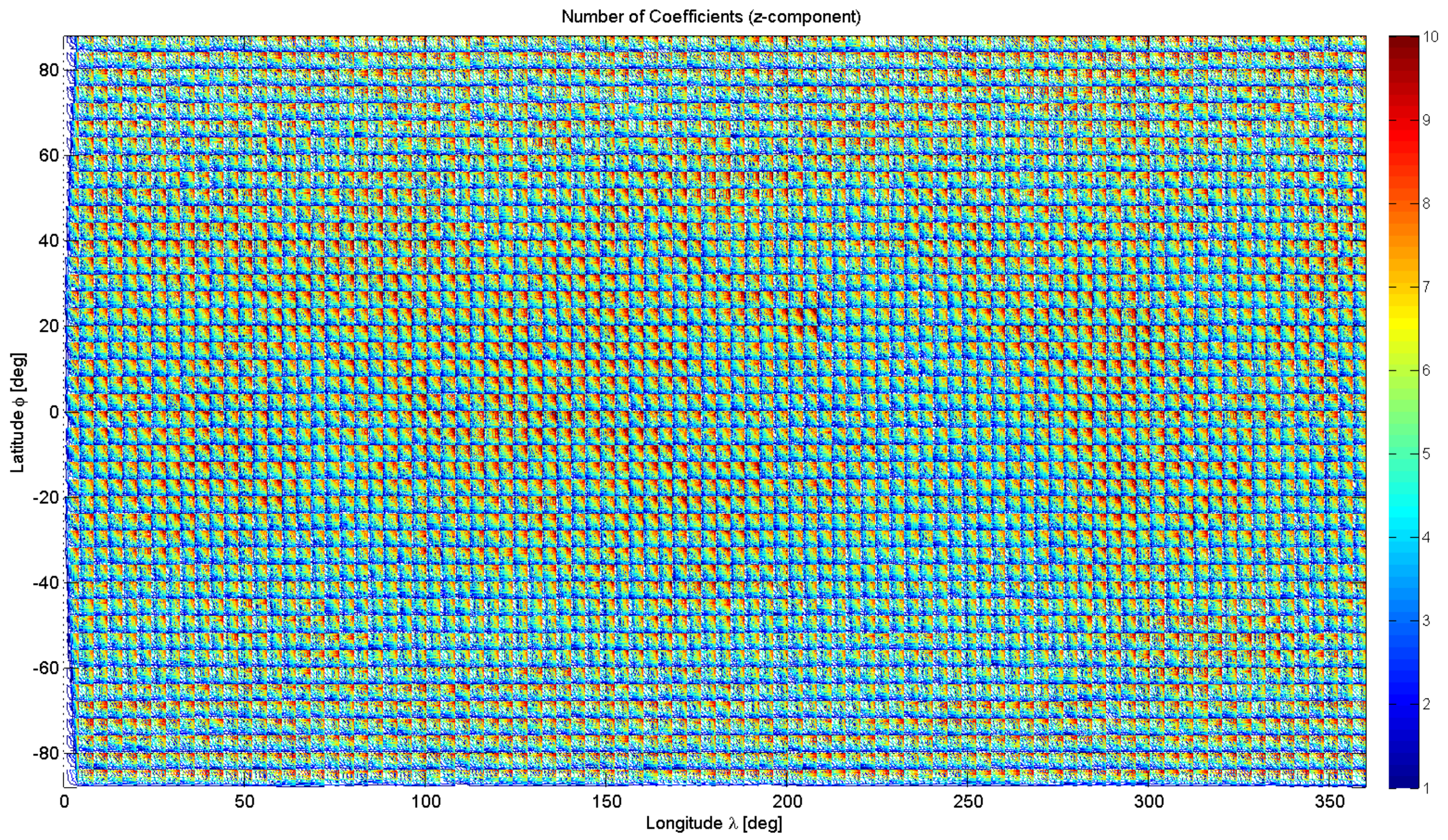


Figure III.23: Distribution of Number of Radial Coefficients Interpolating the z Component Acceleration

$r \in (R_\oplus, 1.02\,R_\oplus)$.

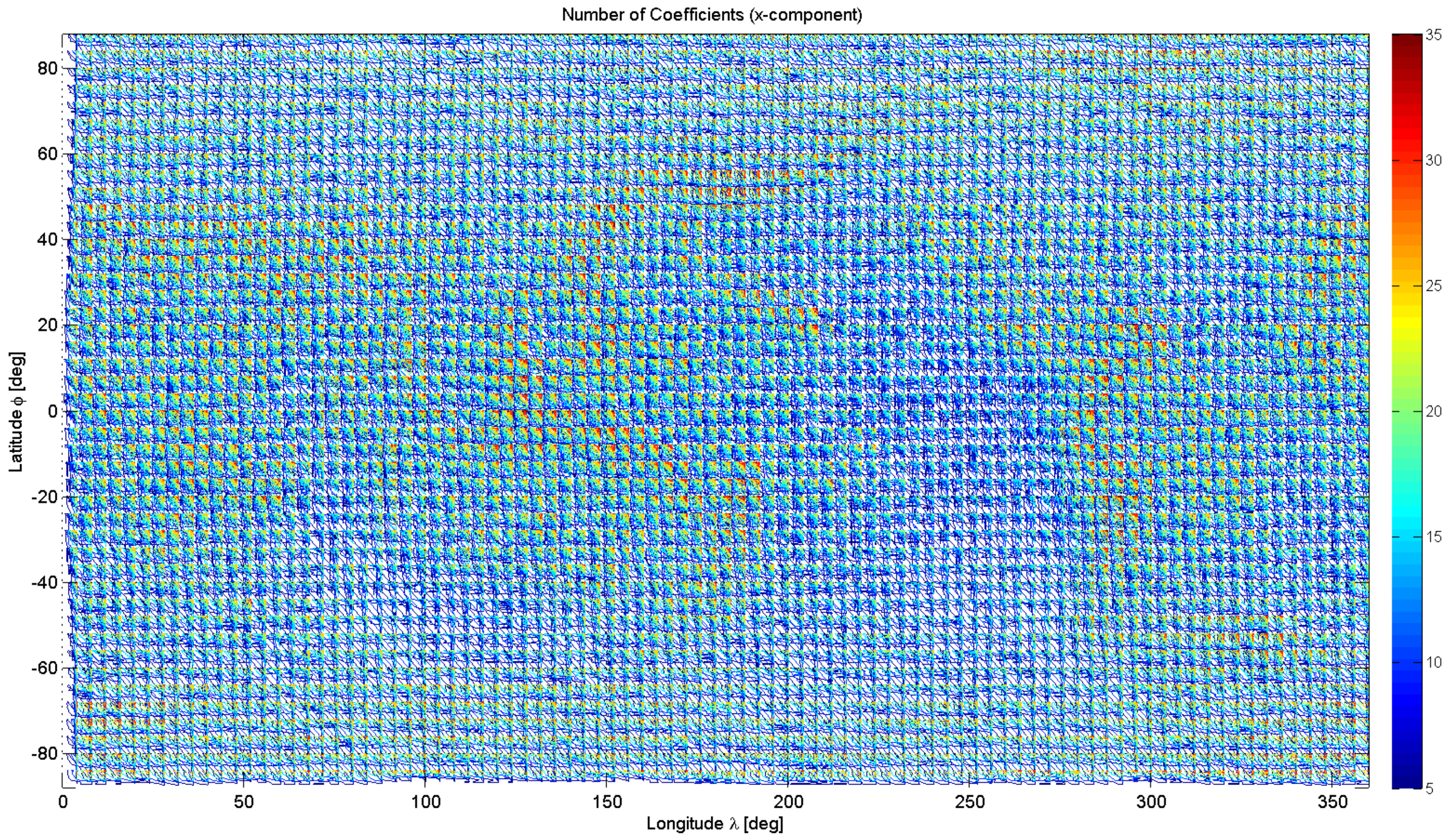


Figure III.24: Distribution of Number of Radial Coefficients Interpolating the x Component Acceleration $r \in (1.02\,R_{\oplus}, 7\,R_{\oplus})$.

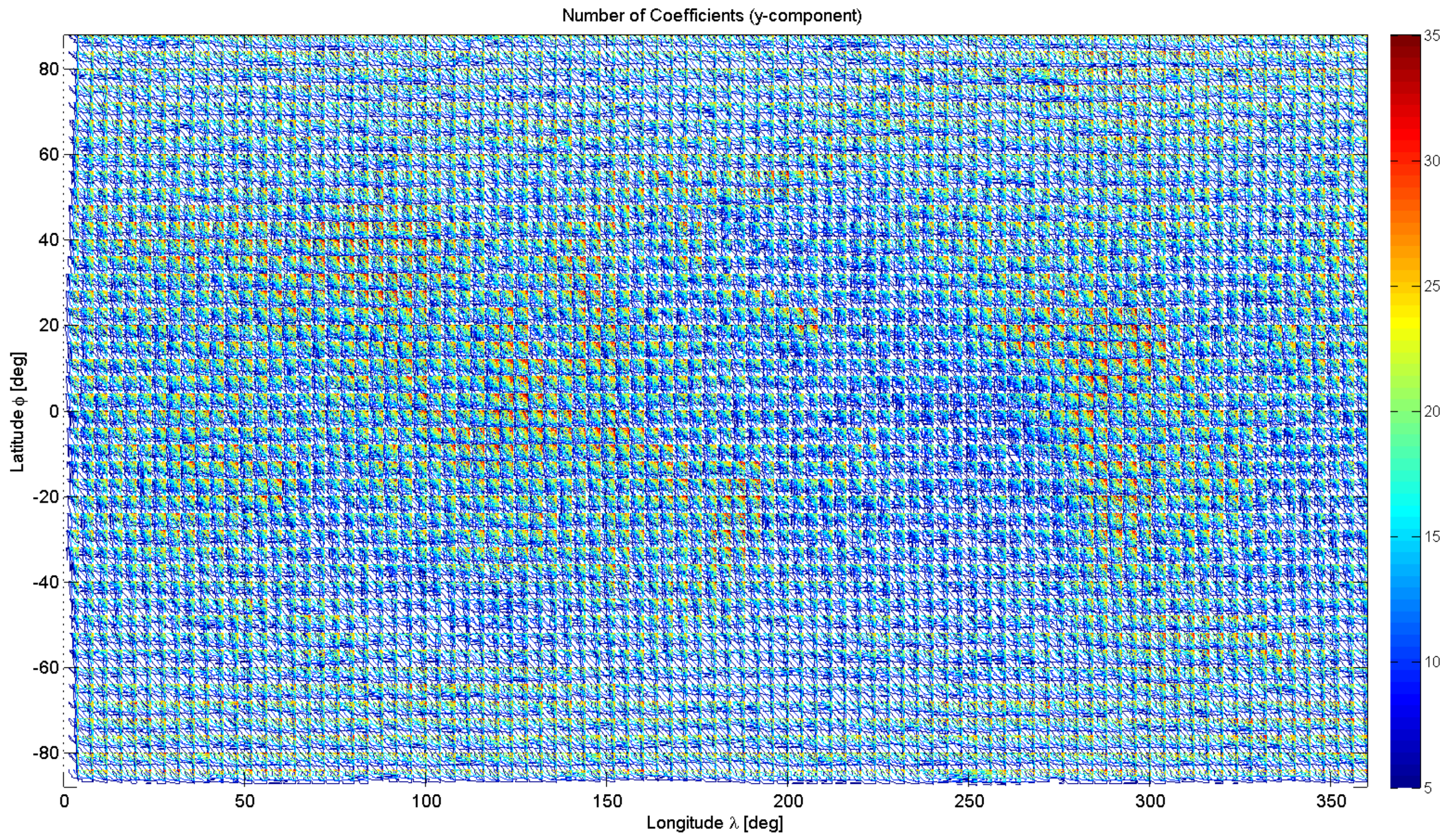


Figure III.25: Distribution of Number of Radial Coefficients Interpolating the y Component Acceleration $r \in (1.02\, R_{\oplus}, 7\, R_{\oplus})$.

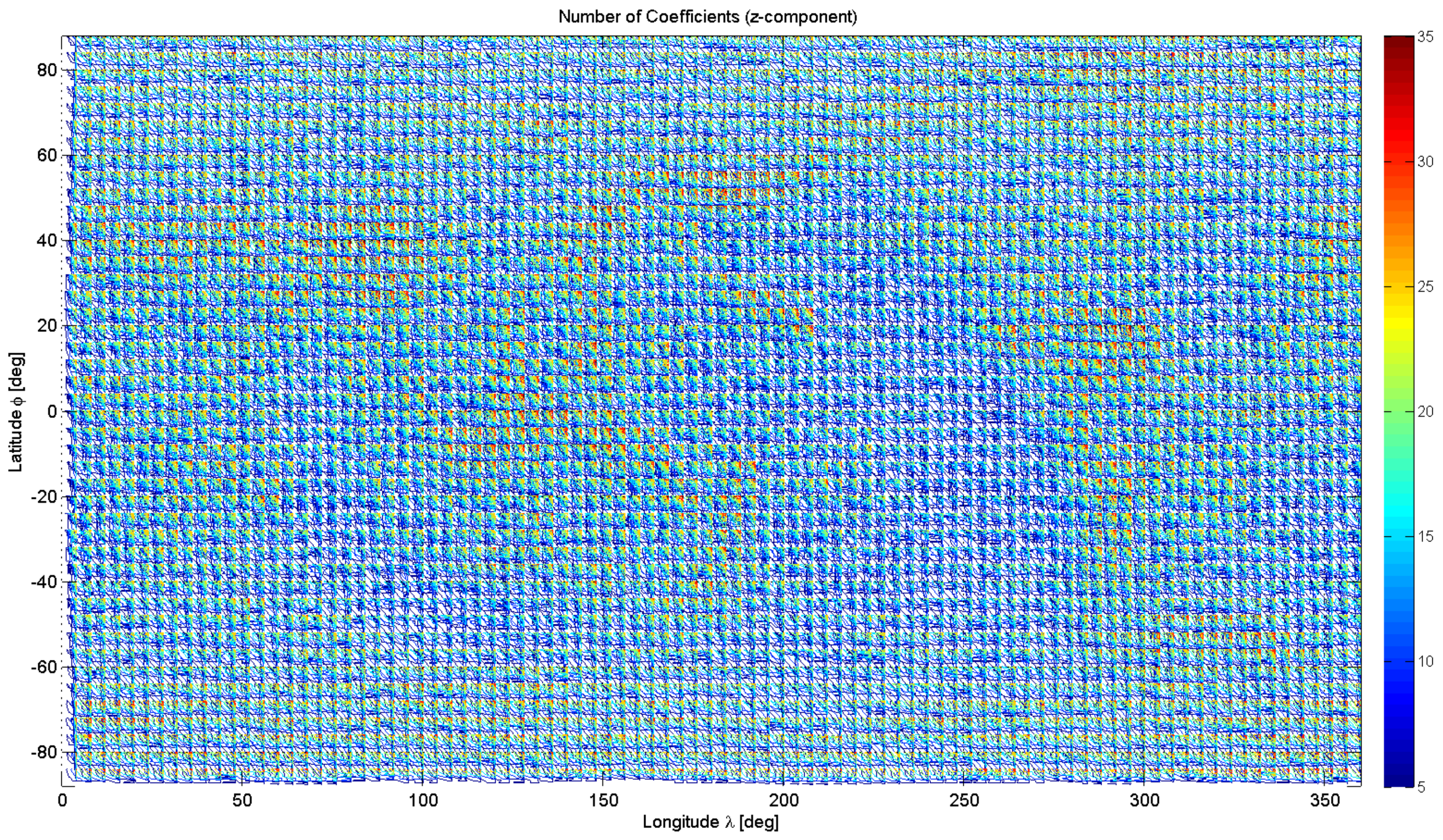


Figure III.26: Distribution of Number of Radial Coefficients Interpolating the z Component Acceleration $r \in (1.02\,R_{\oplus}, 7\,R_{\oplus})$.

CHAPTER IV

PICARD ITERATION, CHEBYSHEV POLYNOMIALS AND CHEBYSHEV-PICARD METHODS

IV.A. Introduction

During the 19th century Emile Picard, a French mathematician, introduced a classical successive path approximation method for solving differential equations of the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad \boldsymbol{x}(t_0). \tag{4.1}$$

This can be rearranged without approximation to obtain the following integral equation

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau))\, d\tau. \tag{4.2}$$

Motivated by the exact integral equation form of Eq. (4.2), Picard hypothesized a sequence of trajectory approximations (*Picard Iteration*) generated by

$$\boldsymbol{x}^{i}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}\left(\tau, \boldsymbol{x}^{i-1}(\tau)\right) d\tau, \quad i = 1, 2, ... \tag{4.3}$$

Picard also published formal Lipshitz conditions for convergence of this sequence to the solution of Eq. (4.1). The essence of his convergence theorem is that if the function $\boldsymbol{f}(t, \boldsymbol{x})$ and it's Jacobian $[\partial \boldsymbol{f}(t, \boldsymbol{x})/\partial \boldsymbol{x}]$ are continuous and bounded over the finite region $\{\mid t - t_0 \mid \; < \delta, \| \; \boldsymbol{x}^0(t) - \boldsymbol{x}(t) \; \|_\infty < \triangle\}$, then a unique solution of Eq (4.1) exists and the sequence of Eq. (4.3) converges to the unique solution. The sequence of trajectories from Eq. (4.3) converges to the solution of Eq. (4.1) above for some finite bounds $\{\delta, \triangle\}$ defining the finite region of guaranteed convergence. Furthermore, under these the same conditions on $\boldsymbol{f}(t, \boldsymbol{x})$ and $[\partial \boldsymbol{f}(t, \boldsymbol{x})/\partial \boldsymbol{x}]$, [50–57] establish the conditions under which the Picard Iteration operator is a contraction mapping: the sequence converges to the unique solution if $t - t_0$ is smaller than δ, and the starting trajectory is in the region bounded by $\triangle$. The $(\delta, \triangle)$ bounds for guaranteed convergence of the Picard sequence are generally difficult to estimate without extensive computational investigation over the volume of state space where the starting approximations and the unique solution lie. Even though difficult to compute, and frequently, the $(\delta, \triangle)$ bounds are highly conservative, they remain of theoretical importance. While means for computing practical convergence bounds that are useful in general-purpose algorithms have proven elusive, as we will show for the case of a linear system where MCPI is implemented, convergence analysis leads to very interesting results and practical convergence insight. When an excellent starting trajectory approximation $\boldsymbol{x}^0(t)$ exists, then the successive trajectories are close neighbors, and the general nonlinear contraction mapping theory can be replaced approximately by the linear MCPI contraction mapping and convergence analysis. Furthermore, for those problems where a good starting approximation can be generated using prior approximate insight, Picard Iteration is obviously accelerated.

IV.B. OCPI: Orthogonal Chebyshev Polynomial Integrator

Before we discuss the Modified Chebyshev Picard Iteration (MCPI), let us first introduce a conventional orthogonal polynomial integrator, where no Picard iteration is performed. The Orthogonal Chebyshev Polynomial Integrator (OCPI) is used when the force function, $\boldsymbol{g}(\tau)$, only depends on time; i.e. no trajectory dependance. Let us define our first order differential equation as

$$\frac{d\boldsymbol{x}}{d\tau} = \boldsymbol{g}(\tau) \tag{4.4}$$

where τ is defined on the valid range (the closed interval $[-1, 1]$) of Chebyshev polynomials. The force function can be approximated by N order of orthogonal polynomial

$$\boldsymbol{g}(\tau) \cong \sum_{n=0}^{N} a_n \phi_n(\tau) \tag{4.5}$$

where a_n is the n^{th} approximation coefficient. The basis function $\phi(\tau)$ is chosen to be the classical Chebyshev polynomial, namely

$$\Big\{\phi_0(\tau), \phi_1(\tau), \cdots, \phi_N(\tau)\Big\} = \Big\{T_0(\tau), T_1(\tau), \cdots, T_N(\tau)\Big\} \tag{4.6}$$

The Chebyshev polynomial of degree k is denoted by $T_k(\tau)$ and the $(M + 1)$ discrete nodes that are used to approximate the force function are the Chebyshev-Gauss-Lobatto (CGL) nodes are given by

$$\tau_j = -cos(j\pi/M), \quad j = 0, 1, 2, ..., M. \tag{4.7}$$

Substituting Eq. (4.6) in Eq. (4.5) yields

$$\boldsymbol{g}(\tau) \cong \sum_{n=0}^{N} a_n T_n(\tau) \tag{4.8}$$

Using the discrete orthogonality property of Chebyshev polynomials, the coefficient a_n can be calculated immediately through the weighted inner product

$$a_n = \frac{1}{c_n} \sum_{k=0}^{M} w_k \boldsymbol{g}(\tau_k) T_n(\tau_k), \tag{4.9}$$

where $c_0 = N$; $\{c_n = N/2;$ for $n = 1, 2, ...N\}$; $w_0 = w_N = \frac{1}{2}$; $\{w_k = 1;$ for $k = 1, 2, ...N\}$.

The solution is now obtained by integrating the approximation, which is expressed by

$$\boldsymbol{x}(\tau) \cong constant + \sum_{n=0}^{N} a_n \int T_n(\tau) d\tau \tag{4.10}$$

where $\int T_n(\tau)d\tau$ is calculated using the following identity

$$\int T_n(\tau)d\tau = \begin{cases} \frac{1}{2}\left(\frac{T_{n+1}(\tau)}{n+1} - \frac{T_{n-1}(\tau)}{n-1}\right) + constant & \text{if } n \geq 2 \\ \frac{1}{4}T_2(\tau) + constant & \text{if } n = 1 \\ T_1(\tau) + constant & \text{if } n = 0 \end{cases} \tag{4.11}$$

IV.B.1. Example: Ballistic Projectile Problem

In this example, the attitude dynamics, described by the pitch (θ) and the yaw (ψ), of an inertially and aerodynamically symmetric projectile are modeled by the following algebraic equations [69]

$$\begin{aligned} \theta(t) &= \sum_{i=1}^{3} k_i e^{\lambda_i t} cos(\omega_i t + \delta_i) + k_4, \\ \dot{\theta}(t) &= \sum_{i=1}^{3} k_i e^{\lambda_i t} \{\lambda_i cos(\omega_i t + \delta_i) - \omega_i sin(\omega_i t + \delta_i)\} \\ \psi(t) &= \sum_{i=1}^{3} k_i e^{\lambda_i t} sin(\omega_i t + \delta_i) + k_5, \\ \dot{\psi}(t) &= \sum_{i=1}^{3} k_i e^{\lambda_i t} \{\lambda_i sin(\omega_i t + \delta_i) + \omega_i cos(\omega_i t + \delta_i)\} \end{aligned} \tag{4.12}$$

where t is given time in seconds [0, 25], which is replaced by the normalized $\tau \in [-1, 1]$. The constants $k_i, \omega_i, \lambda_i,$ and δ_i are related to the aerodynamic and mass characteristics of the projectile and to the initial motion conditions. In this test, these constants are selected randomly to generate an arbitrary trajectory that only depends on time. The pitch and yaw projectiles are shown in Figure IV.1. The goal here is to approximate ($\dot{\theta}(t)$ and $\dot{\psi}(t)$) using the Orthogonal Chebyshev Polynomial Integrator (OCPI) to approximate the derivatives and then integrate the approximation to compare with the exact ($\theta(t)$ and $\psi(t)$). A machine error is achieved for both the function approximation and its integration, for $N = 50$, as shown in Figure IV.2. We show only the pitch trajectory approximation θ and $\dot{\theta}$. The yaw trajectory approximation ψ and $\dot{\psi}$ give exactly analogous results.

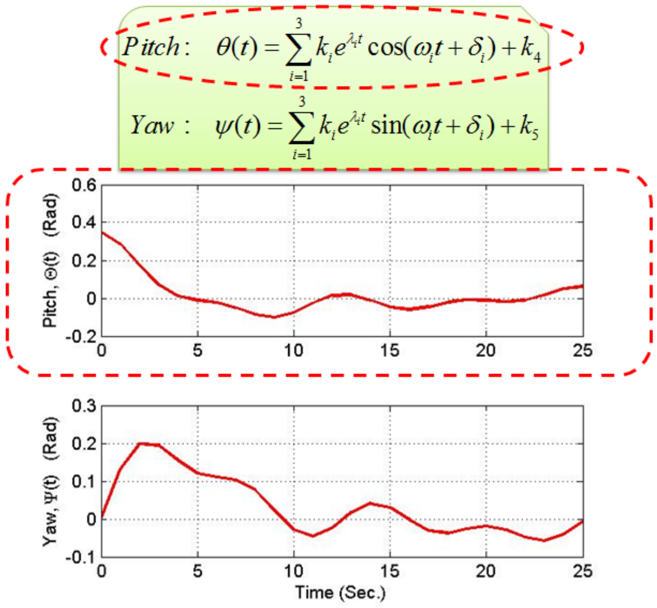


Figure IV.1: Ballistic Projectile Problem.

This example is implemented in C/C++, for the serial computation, and CUDA, for the parallel computation, see Figure IV.3a. It is intended also to understand the computation runtime cost when the code is run on a serial processor against parallel processor. The parallel structure is performed by carrying out each node calculation on a GPU thread, then passing the individual results into the host device (CPU).

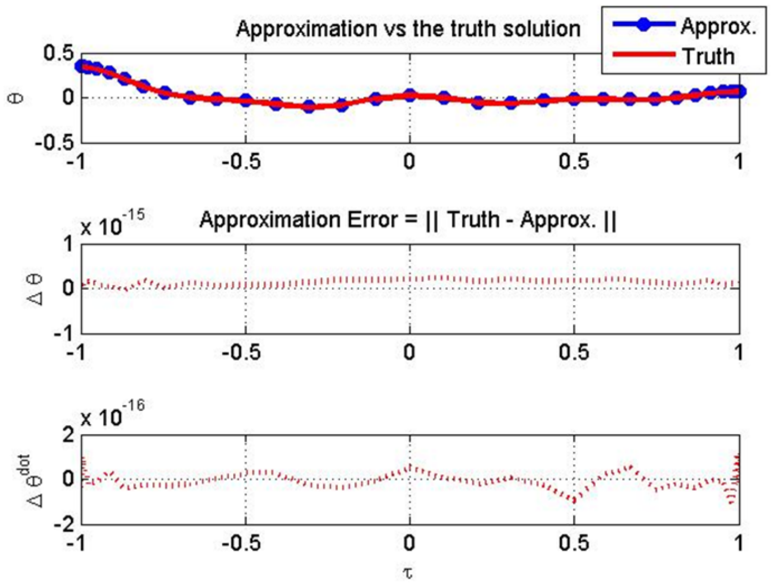


Figure IV.2: Approximation Error ($N = 50$).

Speedup over Serial is about one order of magnitude for small N and increases to over 2 orders of magnitude for large N, as shown in Figure IV.3b. In this case, the results for large N are simply for illustration of efficiency of parallel computation versus N, since $N = 50$ gives machine precision accuracy.

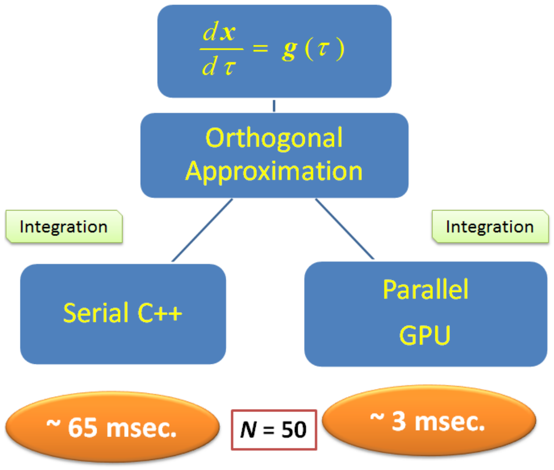
IV.C. MCPI: Modified Chebyshev Picard Iteration

Following the approach of [17], the first step of MCPI methods is to transform the generic independent variable t to a new variable τ, which is defined on the valid range (the closed interval $[-1, 1]$) of Chebyshev polynomials.

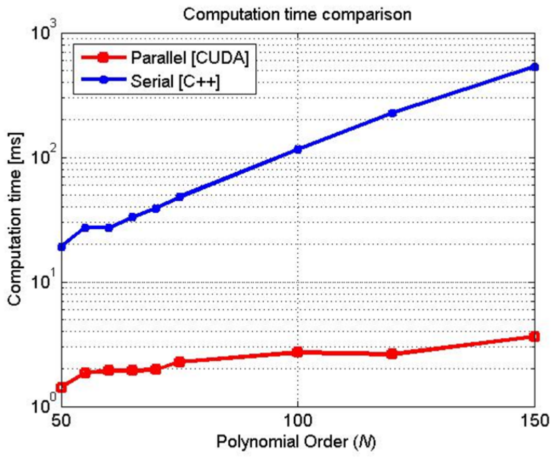
$$t = w_1 + w_2\tau, \quad w_1 = (t_f + t_0)/2, \quad w_2 = (t_f - t_0)/2, \quad -1 \leq \tau \leq 1 \tag{4.13}$$

Introducing this time transformation of Eq. (4.13) into Eq. (4.1), it is re-written as

$$\frac{d\boldsymbol{x}}{d\tau} = \boldsymbol{g}(\tau, \boldsymbol{x}) \equiv w_2 \boldsymbol{f}(w_1 + w_2\tau, \boldsymbol{x}), \tag{4.14}$$



(a) Computation Flowchart.



(b) Computation Time.

Figure IV.3: Computation Cost Serial vs Parallel.

and Picard Iteration for this transformed system is written by analogy with Eq. (4.3) as

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}_0 + \int_{-1}^{\tau} \boldsymbol{g}(s, \boldsymbol{x}^{i-1}(s))ds \quad i = 1, 2, ... \tag{4.15}$$

Now, we introduce Chebyshev polynomial approximations of *both the unknown trajectory and the integrand of Eq. (4.15) along each trajectory* $\boldsymbol{x}^i(t)$. The path approximation sequence $x^i(t)$ is frequently convergent under surprisingly large $(t_f - t_0)$ intervals; in fact intervals exceeding one orbit period for low earth orbits are typical. The Chebyshev polynomial of degree k is denoted by $T_k(\tau)$ and the $(N+1)$ discrete nodes that are used to approximate the states are the Chebyshev-Gauss-Lobatto (CGL) nodes are given by

$$\tau_j = -cos(j\pi/N), \quad j = 0, 1, 2, ..., N. \tag{4.16}$$

Assume the force function vector is approximated by an N^{th} order Chebyshev polynomial

$$\boldsymbol{g}(\tau,\boldsymbol{x}^{i-1}(\tau)) = \sum_{k=0}^{N-1} \boldsymbol{F}_k^{i-1} T_k(\tau) \equiv \boldsymbol{F}_0^{i-1}T_0(\tau)+\boldsymbol{F}_1^{i-1}T_1(\tau)+\boldsymbol{F}_2^{i-1}T_2(\tau)+...+\boldsymbol{F}_{N-1}^{i-1}T_{N-1}(\tau). \tag{4.17}$$

Note that in Eq. (4.17) the summation is to $N-1$ rather than to N. This is because the summation is within the integral and after integration the order of the polynomial will be increased from $N-1$ to degree N as a convergance of analytical integration $T_{N-1}(\tau)$. This is a slight modification from Bai's PhD dissertation [17].

Using the discrete orthogonality property of Chebyshev polynomials, the coefficient vectors $\boldsymbol{F}_k^{i-1}$ are calculated immediately through

$$\boldsymbol{F}_n^{i-1} = \frac{1}{c_n}\sum_{k=0}^{N} w_k \boldsymbol{g}(\tau_k,\boldsymbol{x}^{i-1}(\tau_k))T_n(\tau_k), \tag{4.18}$$

where $c_0 = N;\ \{c_n = N/2;\ \text{for } n = 1, 2, ...N\}\,;\ w_0 = w_N = \frac{1}{2};\ \{w_k = 1;\ \text{for } k = 1, 2, ...N\}$.

Notice each coefficient of $\boldsymbol{F}_k^{i-1}$ is obtained through the summation of $(N+1)$ independent terms, each of which is an inner product of the force function $\boldsymbol{g}(\tau,\boldsymbol{x}(\tau))$ and the Chebyshev polynomials $T_k(\tau)$ evaluated at the CGL points of Eq. (4.16). Furthermore, all the coefficient vectors are independent of each other, and can therefore be computed in parallel processors. Also, and most importantly, for problems where calculating the force vector function $\boldsymbol{g}(\tau,\boldsymbol{x}(\tau))$ is time consuming, significant time performance improvement is achieved by simultaneous computation of $\boldsymbol{g}(\tau_j,\boldsymbol{x}(\tau_j))$ at the nodes of Eq. (4.16) on $N+1$ parallel processors. Assuming the solution at the i^{th} path approximation is denoted $\boldsymbol{x}^i(t)$, the Picard Iteration provides the recursion to calculate $\boldsymbol{x}^i(t)$ as a Chebyshev polynomial approximation over the entire time interval as

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}_0 + \sum_{r=0}^{N-1} \boldsymbol{F}_r^{i-1} \int_{-1}^{\tau} T_r(s)ds \equiv \sum_{k=0}^{N} \beta_k^i T_k(\tau). \tag{4.19}$$

The coefficient vectors for this new trajectory expressed below in Eq. (4.20), are obtained directly from recollecting the term-by-term analytical integration of Eq. (4.19), and imposing the initial boundary conditions.

$$\beta_r^i = \tfrac{1}{2r}(\boldsymbol{F}_{r-1}^{i-1} - \boldsymbol{F}_{r+1}^{i-1}),\ r = 1, 2, ..., N-1,$$

$$\beta_0^i = \boldsymbol{x}_0 + \textstyle\sum_{k=1}^{N}(-1)^{k+1}\beta_k^i,$$

$$\beta_{N-1}^i = \tfrac{\boldsymbol{F}_{N-2}^{i-1}}{2(N-1)}$$

and

$$\beta_N^i = \frac{1}{2N}\boldsymbol{F}_{N-1}^{i-1}. \tag{4.20}$$

The third equation of Eqs. (4.20) is added to the original set published by Bai [17]. This incorporates the $N-1$ summation change mentioned previously.

The updated coefficient vectors define the new trajectory approximation for use in the integrand for the next step $(i+1)$. Figure IV.4 displays an overview of the algorithm. Thus the solutions are iteratively improved until some accuracy requirements are satisfied.

To account for the occasional non uniform convergence errors, a conservative stopping criterion we choose is to require both the maximum difference (among all the $N+1$ CGL nodes) between the solutions $\boldsymbol{x}^i(\tau)$, $\boldsymbol{x}^{i-1}(\tau), and$ the maximum difference between the solutions $\boldsymbol{x}^i(\tau)$, $\boldsymbol{x}^{i+1}(\tau)$ are less than some tolerance.

Instead of term by term scalar process to solve for the state value at the $(N+1)$ CGL nodes, the $(N+1)$ Chebyshev coefficients, and the updated $(N+1)$ Chebyshev coefficients, we have developed a compact matrix-vector approach to implement MCPI methods, which is shown in Figure IV.5. The position vector is written in matrix-vector form as follows:

$$X^i = TC_\alpha G(X^{i-1}) + T\Theta_{x0}, \tag{4.21}$$

where $X^i = col\left\{x^i(\tau_0), ..., x^i(\tau_N)\right\}$, and G is the vector representation of the forcing function $\boldsymbol{g}$. The initial conditions are contained in the vector $\Theta_{x0} = col\left[x_0, 0, 0, ..., 0\right]$ ϵR^{N+1}.

The derivation of the matrices is found in Bai's dissertation [17]. The basis functions and constants arising from the process are collected in the matrices

$$T = \begin{bmatrix} T_0(\tau_0) & T_1(\tau_0) & \cdots & T_N(\tau_0) \\ T_0(\tau_1) & T_1(\tau_1) & \cdots & T_N(\tau_1) \\ \vdots & \vdots & \vdots & \vdots \\ T_0(\tau_N) & T_0(\tau_N) & \cdots & T_0(\tau_N) \end{bmatrix}, \quad C_\alpha = RSTV,$$

where the matrices are defined as

$$S = \begin{bmatrix} 1 & -\frac{1}{2} & S(1,3) & S(1,4) & S(1,5) & \cdots & 0 \\ 1 & 0 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

and

$$\left\{ \begin{array}{c} R = diag\left(\left[1, \frac{1}{2\times r}, ...\right]\right),\ r = 1, 2, ..., N \\ \\ V = diag\left(\left[\frac{1}{N}, \frac{2}{N}, ..., \frac{2}{N}, \frac{1}{N}\right]\right) \end{array} \right\}. \tag{4.22}$$

The k^{th} $(k = 3, ..., N)$ column of the 1^{st} row of S has the form: $S(1,k) = (-1)^k \left(\frac{1}{k-2} - \frac{1}{k}\right)$. Notice that the T and C_α matrices and the product TC_α are constant (once N is selected), and so all computations of inner products are efficiently implemented. The eigenstructure of the matrix product TC_α is of crucial importance in analyzing convergence of the Picard Iteration. For the linear case, the "complete story" on convergence can be obtained readily based on the eigenvalue analysis of TC_α. The structure of these eigenvalues is remarkably elegant and is discussed in the convergence analysis that follows.

IV.C.1. Some Remarks, for Perspective:

We emphasize that the proposed MCPI methods are different from the collocation Implicit Runge-Kutta (IRK) methods [17], although both utilize discrete nodes and frequently, orthogonal functions, to approximate the state trajectories. In the collocation IRK methods, $N+1$ collocation points are chosen, and an N^{th} order polynomial approximation that satisfies the differential equations to some tolerance at the collocation points is sought by solving $N+1$ nonlinear algebraic equations. The derivative of the trajectory approximation is substituted on the L.H.S of Eq. (4.1), and the trajectory is substituted into the argument of the R.H.S of Eq. (4.1). When evaluated at the nodes, this provides the system of nonlinear algebraic equations to iteratively solve for the basis function coefficients. However, for the MCPI methods, the same polynomial basis function set (Chebyshev polynomials in the current discussion) is used to approximate both the solutions of the differential equations and the force functions along each trajectory approximation. Note, the approximation is never differentiated in MCPI, having the force function approximated as linear combination of basis functions (along the previous trajectory approximation), permits the Picard integrals of Eq (4.3) to be done term by term, and as a consequence, there is no nonlinear iteration involved. In the case of the collocation methods, including IRK algorithms, local linearization in the parameter space of the basis function amplitudes is required to generate the successive approximation iterations. In the case of MCPI, no linearization is required even though the differential equation is nonlinear and the new amplitudes appear linearly, with no further approximation. The sole basis for successive iterations is the Picard Iteration itself.

Furthermore, since all unknowns appear linearly with no further approximation beyond adopting the Picard Iteration, and an appropriate number of free constants arise in the integration to linearly impose all boundary conditions, *no local Jacobian need be evaluated and no shooting method type iteration is required.* Further, other than the $N + 1$ function evaluations that can be performed in parallel, all other operations are only vector-matrix multiplication operations with most of the matrices computed only once. Additionally, although currently for an N^{th} order Chebyshev polynomial we use $N + 1$ CGL nodes to approximate both the solution and the integrand, the fundamental MCPI algorithm only requires that the number of nodes to be no less than the order of the polynomials and located at points consistent with the discrete orthogonality conditions. This is due to our choice to use the discrete least-square formulation to approximate the functions, which offers some flexibility for choosing nodes and polynomial orders.

Fox [30] proved that if a continuous function $f(\tau)$ is approximated by a n^{th} order Chebyshev polynomial $p_n(\tau) = \sum\limits_{k=0}^{N} \alpha_k T_k(\tau)$, as a consequence of the orthogonality of the basis functions for the selected nodes, the error $e_n(\tau) = f(\tau) - p_n(\tau)$ is guaranteed to satisfy the discrete least-squares criterion $S = \sum\limits_{r=0}^{N} e^2(\tau_j) = minimum$, and the explicit error bound is $S_{min} = \sum\limits_{j=0}^{N} \mid f^2(\tau_j) - \sum\limits_{r=0}^{N} a_r^2 T_r^2(\tau_j) \mid$.

IV.D. Second Order MCPI Approach

Consider a second order problem:

$$\frac{d^2\boldsymbol{x}}{dt^2} = \boldsymbol{f}(t, \boldsymbol{x}(t), \dot{\boldsymbol{x}}(t)), \quad \boldsymbol{x}(t) \epsilon R^n, \quad \{\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \ \& \ \dot{\boldsymbol{x}}(t_0) = \dot{\boldsymbol{x}}_0.\} \tag{4.23}$$

The first order Picard Iteration formulation presented in the last section can solve this problem after we introduce a new state variable $z = col\{\boldsymbol{x}, \dot{\boldsymbol{x}}\} \epsilon R^{2n}$ to transform Eq. (4.23) to a system of $2n$ first order equations. Bai [17] developed a straightforward cascaded MCPI formulation, i.e., a second order MCPI approach, that is generalized to systems described by higher order differential equations. Importantly, this approach solves problems with differential equations dependent on the velocity, whereas many of the most efficient RKN methods cannot. Thus in lieu of Eqs (4.14)-(4.15), we have the transformed version of Eq. (4.23):

$$\frac{d\boldsymbol{v}}{d\tau} = \boldsymbol{g}(\tau, \boldsymbol{x}, \boldsymbol{v}), \quad \frac{d\boldsymbol{x}}{d\tau} \equiv \boldsymbol{v}. \tag{4.24}$$

Given a starting iterative $\{\boldsymbol{x}^0(\tau), \boldsymbol{v}^0(\tau)\}$, Picard Iteration for this pair of equations is by analogy given as

$$\boldsymbol{v}^i(\tau) = \boldsymbol{v}_0 + \int_{-1}^{\tau} \boldsymbol{g}(s, \boldsymbol{x}^{i-1}(s), \boldsymbol{v}^{i-1}(s))ds, \quad i = 1, 2, ... \tag{4.25}$$

Similar to Eq. (4.15), the acceleration along the $(i-1)^{th}$ trajectory approximation is integrated to velocity from Eq. (4.25). Importantly, the position vector is obtained, not from Picard Iteration, but by direct integration of $\boldsymbol{v}^i(\tau)$ to position by using exact kinematic equation {the 2^{nd} of Eq. (4.25)}:

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(t_0) + \int_{-1}^{\tau} \boldsymbol{v}^i(s)ds. \tag{4.26}$$

Notice the approximation errors incur only at the velocity level in Eq. (4.26). When we expand the velocity trajectory and the integrand of Eqs. (4.25) in Chebyshev basis functions and we carry out the integrals of Eq. (4.26) term by term, then no further approximation is required. An exactly kinetically consistent position approximation is derivable from term by term integration of the linearly contained velocity in the integrand of Eq. (4.23). In the equivalent first order Picard Iteration, the position is obtained by modeling the position and velocity independently by linearly combining Chebyshev polynomials resulting in $2n$ approximations in the integrand of Eq. (4.15) with $\{\{\boldsymbol{x}\}\,\epsilon R^n \Longrightarrow \{\mathbf{z}\}\,\epsilon R^{2n}\}$, constrained through the state variable definitions. Thus the velocity approximations implicitly know that they are the derivative of position (more importantly, the position implicitly knows it is the integral of velocity!). However, in the above cascaded formulation the velocity vector approximation directly dictates the corresponding coefficients for the position vector through the kinematic constraint implicit in taking the integral of Eq (4.23). In the above second order MCPI approach, when we introduce the Chebyshev approximations of all variables, computation of position approximation from velocity simply amounts to a matrix multiplication with and addition of invariant (computed once) coefficient matrices T , C_α as follows:

$$\overrightarrow{V^i} = TC_\alpha G\left(\overrightarrow{V}^{i-1}\right) + T\Theta_{v0} \tag{4.27}$$

and

$$\overrightarrow{X^i} = \frac{t_f - t_0}{2} TC_\alpha \left(\overrightarrow{V^i}\right) + T\Theta_{x0}, \tag{4.28}$$

where the i^{th} step Picard iterate evaluated at the $N+1$ CGL nodes are represented by vectors
$\overrightarrow{X^i} = col\left[x(\tau_0), x(\tau_1), x(\tau_2), ..., x(\tau_N)\right]$ and $\overrightarrow{V^i} = col\left[\mathbf{v}(\tau_0), \mathbf{v}(\tau_1), \mathbf{v}(\tau_2), ..., \mathbf{v}(\tau_N)\right]$. The initial conditions are contained in the vector $\Theta_{v0} = col\left[v_0, 0, 0, ..., 0\right] \epsilon R^{N+1}$.

The difference between using (i) an Picard Iteration simultaneously for both position and velocity or (ii) a cascaded Picard Iteration for velocity Eqs (4.23 and 4.26) and subsequent integration to get position {Eqs (4.27), (4.25)} may at first blush look minor, however, this approach is computationally much more attractive because the dimensionality of the approximation space is reduced by a factor of two. It is also more accurate. Note – this efficiency and accuracy advantage is enjoyed on each step, and therefore the entire Picard Iteration efficiency and accuracy are accelerated accordingly.

IV.E. MCPI Convergence Analysis

The accumulation of round off and approximation errors during the iterations (when a finite order of Chebyshev polynomial is used to approximate solutions) lead to the convergence domain of MCPI methods being different from the ideal conditions under which Picard iteration theoretically converges (Lipschitz continuity). However, the general bounds in the literature have been found very conservative in predicting the actual convergence domain; establishing the rigorous convergence domain of MCPI methods applicable for general nonlinear systems is not possible by any known approach. To obtain some essential insight we first use a linear scalar problem as an example to show that the global convergence of MCPI methods is not generally guaranteed, and we then address the practical issue of checking the convergence, and importantly, various approaches to enlarge the convergence domain. Obviously, the Picard Iteration is not proposed for actually solving linear problems, this exercise is presented simply to establish insights on convergence in the ideal case. Consider a scalar linear dynamic system

$$\frac{dx(t)}{dt} = cx(t). \tag{4.29}$$

The i^{th} step Picard iterate evaluated at the $N+1$ CGL nodes is represented by a vector $\Theta_0 = col\,[2x_0, 0, 0, ..., 0]\,\epsilon R^{N+1}$.

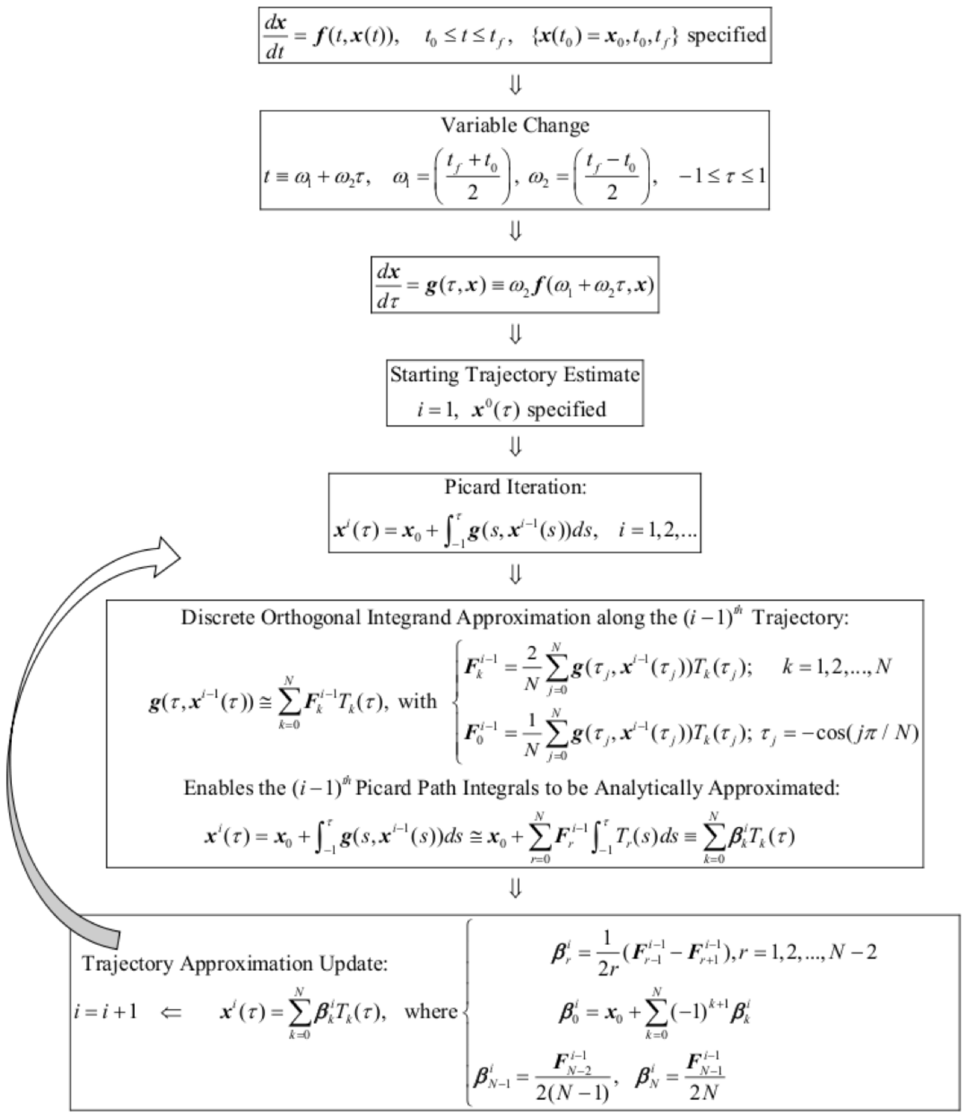

$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad t_0 \le t \le t_f, \quad \{\boldsymbol{x}(t_0) = \boldsymbol{x}_0, t_0, t_f\}$ specified
Variable Change
$t \equiv \omega_1 + \omega_2 \tau, \quad \omega_1 = \left(\frac{t_f + t_0}{2}\right), \ \omega_2 = \left(\frac{t_f - t_0}{2}\right), \quad -1 \le \tau \le 1$
$\frac{d\boldsymbol{x}}{d\tau} = \boldsymbol{g}(\tau, \boldsymbol{x}) \equiv \omega_2 \boldsymbol{f}(\omega_1 + \omega_2 \tau, \boldsymbol{x})$
Starting Trajectory Estimate
$i = 1, \ \boldsymbol{x}^0(\tau)$ specified
Picard Iteration:
$\boldsymbol{x}^i(\tau) = \boldsymbol{x}_0 + \int_{-1}^{\tau} \boldsymbol{g}(s, \boldsymbol{x}^{i-1}(s))ds, \quad i = 1, 2, ...$
Discrete Orthogonal Integrand Approximation along the $(i-1)^{th}$ Trajectory:
$\boldsymbol{g}(\tau, \boldsymbol{x}^{i-1}(\tau)) \cong \sum_{k=0}^{N} \boldsymbol{F}_k^{i-1} T_k(\tau)$, with $\begin{cases} \boldsymbol{F}_k^{i-1} = \frac{2}{N} \sum_{j=0}^{N} \boldsymbol{g}(\tau_j, \boldsymbol{x}^{i-1}(\tau_j)) T_k(\tau_j); \quad k = 1, 2, ..., N \\ \boldsymbol{F}_0^{i-1} = \frac{1}{N} \sum_{j=0}^{N} \boldsymbol{g}(\tau_j, \boldsymbol{x}^{i-1}(\tau_j)) T_k(\tau_j); \ \tau_j = -\cos(j\pi / N) \end{cases}$
Enables the $(i-1)^{th}$ Picard Path Integrals to be Analytically Approximated:
$\boldsymbol{x}^i(\tau) = \boldsymbol{x}_0 + \int_{-1}^{\tau} \boldsymbol{g}(s, \boldsymbol{x}^{i-1}(s))ds \cong \boldsymbol{x}_0 + \sum_{r=0}^{N} \boldsymbol{F}_r^{i-1} \int_{-1}^{\tau} T_r(s)ds \equiv \sum_{k=0}^{N} \boldsymbol{\beta}_k^i T_k(\tau)$
Trajectory Approximation Update:
$i = i + 1 \quad \Leftarrow \quad \boldsymbol{x}^i(\tau) = \sum_{k=0}^{N} \boldsymbol{\beta}_k^i T_k(\tau)$, where $\begin{cases} \boldsymbol{\beta}_r^i = \frac{1}{2r}(\boldsymbol{F}_{r-1}^{i-1} - \boldsymbol{F}_{r+1}^{i-1}), r = 1, 2, ..., N-2 \\ \boldsymbol{\beta}_0^i = \boldsymbol{x}_0 + \sum_{k=0}^{N} (-1)^{k+1} \boldsymbol{\beta}_k^i \\ \boldsymbol{\beta}_{N-1}^i = \frac{\boldsymbol{F}_{N-2}^{i-1}}{2(N-1)}, \quad \boldsymbol{\beta}_N^i = \frac{\boldsymbol{F}_{N-1}^{i-1}}{2N} \end{cases}$

Figure IV.4: MCPI Iterations for Solution of Initial Problems.

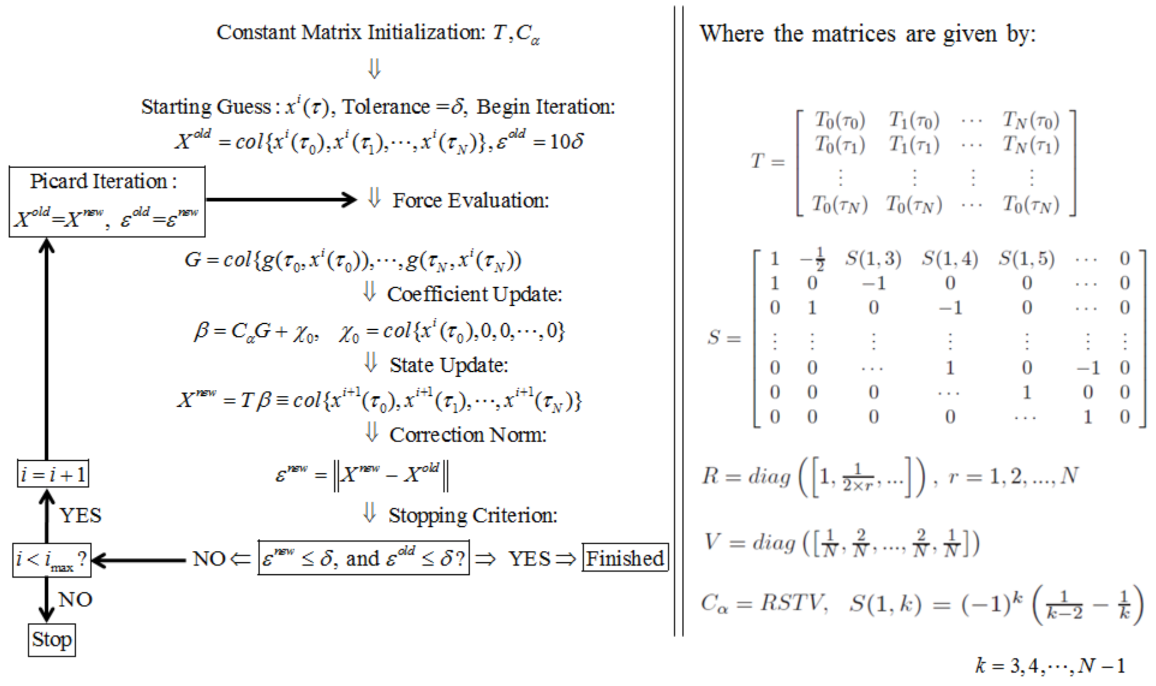

Constant Matrix Initialization: T, C_α
Starting Guess: $x^i(\tau)$, Tolerance $=\delta$, Begin Iteration:
$X^{old} = col\{x^i(\tau_0), x^i(\tau_1), \cdots, x^i(\tau_N)\}, \varepsilon^{old} = 10\delta$
Picard Iteration: $X^{old} = X^{new}$, $\varepsilon^{old} = \varepsilon^{new}$
Force Evaluation:
$G = col\{g(\tau_0, x^i(\tau_0)), \cdots, g(\tau_N, x^i(\tau_N))$
Coefficient Update:
$\beta = C_\alpha G + \chi_0$; $\chi_0 = col\{x^i(\tau_0), 0, 0, \cdots, 0\}$
State Update:
$X^{new} = T\beta \equiv col\{x^{i+1}(\tau_0), x^{i+1}(\tau_1), \cdots, x^{i+1}(\tau_N)\}$
Correction Norm:
$\varepsilon^{new} = \|X^{new} - X^{old}\|$
Stopping Criterion:
NO
$\varepsilon^{new} \le \delta$, and $\varepsilon^{old} \le \delta$?
YES
Finished
$i < i_{max}$?
YES
$i = i + 1$
NO
Stop
Where the matrices are given by:
$T = \begin{bmatrix} T_0(\tau_0) & T_1(\tau_0) & \cdots & T_N(\tau_0) \\ T_0(\tau_1) & T_1(\tau_1) & \cdots & T_N(\tau_1) \\ \vdots & \vdots & \vdots & \vdots \\ T_0(\tau_N) & T_0(\tau_N) & \cdots & T_0(\tau_N) \end{bmatrix}$
$S = \begin{bmatrix} 1 & -\frac{1}{2} & S(1,3) & S(1,4) & S(1,5) & \cdots & 0 \\ 1 & 0 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$
$R = diag\left(\left[1, \frac{1}{2\times r}, ...\right]\right)$, $r = 1, 2, ..., N$
$V = diag\left(\left[\frac{1}{N}, \frac{2}{N}, ..., \frac{2}{N}, \frac{1}{N}\right]\right)$
$C_\alpha = RSTV$, $S(1,k) = (-1)^k\left(\frac{1}{k-2} - \frac{1}{k}\right)$
$k = 3, 4, \cdots, N-1$

Figure IV.5: Flowchart of the Matrix-vector Form of the MCPI Algorithm.

The matrix-vector form of MCPI (Figure 4.1) leads to the recursive solution

$$\overrightarrow{X^i} = \left[\frac{t_f - t_0}{2} cTC_\alpha\right] \overrightarrow{X}^{i-1} + T\Theta_0. \tag{4.30}$$

It is known from linear system theory that this sequence is convergent to a fixed point only if all the eigen-values of the matrix $[((t_f - t_0)c/2)\, TC_\alpha]$ are within a unit circle (i.e., Eq. (IV.5) must be a contraction mapping, to converge to a fixed point). Notice that the scalars $\{(t_f - t_0)/2, c\}$ appear multiplicatively and therefore simply scale the maximum eigenvalues of TC_α and lead to an attractive analysis for this simplest case of a linear problem. Thus the convergence of the MCPI method is dependent on both the dynamical system characteristics c, the length of the time interval $t_f - t_0$, and the matrix TC_α, which is only dependent on the order of the Chebyshev polynomials used. For convergence, we require

$$\mid \left(\frac{t_f - t_0}{2}\right) c\lambda_{max}\left(TC_\alpha\right) \mid < 1, \; or \; \mid t_f - t_0 \mid < \frac{2}{\mid c\lambda_{max}(TC_\alpha) \mid}. \tag{4.31}$$

Remarkably, we see that this identical invariant (given N) matrix TC_α appears multiplicatively in the nonlinear generalization in vector-matrix notation (see Figure IV.5). Therefore during the terminal iterations of a convergent solution process, we can expect this type of eigenanalysis to give approximate behavior useful in a more general setting. In fact, for a vector time varying nonlinear system, the above bound changes only with the scalar c being replaced by the infinity norm of the Jacobian - an elegant result. Notice that TC_α depends solely on the choice of Chebyshev basis functions, the nodal pattern selected, and the degree of the approximation. Therefore TC_α is invariant, can be computed once, and the eigenanalysis may be studied once for all N and applies in all subsequent MCPI solutions.

The eigenvalues of TC_α are shown in Figures IV.6-IV.7 and the maximum eigenvalues are shown in Figure IV.8. We found for small N ($N < 40$), this value decays approximately from 0.7 to 0.054, almost linearly on a log-log scale. Thereafter, for $N > 40$, this value remains approximately constant at 0.054. This gives rise to the maximum interval length as $(t_f - t_0)_{max} = (2/c)\,(1/\lambda_{max}(TC_\alpha)) = 37/c$. While this condition guarantees convergence of the Picard iterations, for a fixed N, it does not guarantee that N is sufficiently high to give an accurate approximation of the solution. It is fortunate, as is evident in Figure IV.8, that convergence does not degrade for large N, or put another way, $N > 40$ can be adjusted to achieve high solution precision *without affecting the rate of Picard iteration convergence.* Thus, the most fundamental truth, we are guaranteed a significant finite time interval over which Picard iteration will converge. For longer time intervals, this suggests "patching" converged sub arcs together.

The remarkable asymptotic behavior of the maximum eigenvalue of TC_α is related to an even more unusual behavior of the locus of eigenvalues of TC_α with increasing N. As is evident in Figures IV.6 and IV.7, we show the eigenvalue locus results that are new and are reported in this proposal for the first time. For $N < 100$, as seen in Figure IV.6, all eigenvalues are comfortably within the unit circle, and distributed along elliptical shaped loci that decrease in size and eccentricity as N increases. For each N, there are two sets, one set near the origin, and another locus that loops to the right in Figure IV.6. As N increases, a very surprising and beautiful pattern emerges. Note in Figure IV.7 that the eigenvalues are apparently attracted to distinct locations along a circle of radius $\tilde{0}.027$ which touches the origin.

Note further that the right most eigenvalues lie at distinct points on this circle for $N > 100$, while the remaining eigenvalues of TC_α, including all "new" ones as N increases and the $(N+1) \times (N+1)$ dimensions of TC_α increases - all cluster on the circular locus near the origin. Based on this root locus study, we hypothesize that as $N \to \infty$, an infinite number of eigenvalues cluster approaching zero separation tangent to the circle at the origin. Thus the new eigenvalues for $N > 40$ cluster ever nearer the origin, while the larger eigenvalues converge to distinct positions on the right-most part of the circular locus in Figure IV.7. The right most pair's convergence to their position with increasing N is responsible for the asymptotic behavior in Figure IV.8. We have studied these remarkable loci carefully and confirmed this behavior, but explaining the cause of these beautiful loci has proven elusive to this point. In addition to gaining fundamental insights on the eigenstructure, we seek to learn how these characteristics depend upon the choice of the Chebyshev polynomials and the CGL non-uniform nodes.

For convergence insight for the case of 2nd order differential equations, we consider $\frac{d^2x(t)}{dr^2} = cx(t)$. We find the velocity update equation is similar to Eq (4.28), whereas the position updated equation is

$$\overrightarrow{X}^i = c\left(\frac{t_f - t_0}{2}\right)^2 TC_\alpha TC_\alpha \overrightarrow{X}^{i-1} + \frac{t_f - t_0}{2} TC_\alpha T\Theta_{v0} + T\Theta_{x0}. \tag{4.32}$$

We found out that the maximum eigenvalues of $TC_\alpha TC_\alpha$ decreases from 0.038 to 0.003 as N increases from 10 to 40, and analogous to the case for the first order system, for all the $N > 40$, the maximum $TC_\alpha TC_\alpha$ eigenvalues are asymptotically approach about 0.003. Thus the convergence condition for Picard Iteration (MCPI) is approximately $c(t_f - t_0)^2 < 4/(0.003) \approx 1333$; we mention the significant truth that this represents a two order of magnitude increase over the size of the classical estimate of the convergence region $\{c(t_f - t_0)^2 < 12\}$ [17], i.e. the classical estimate is highly conservative.

Although this linear analysis tells us that the Chebyshev-Picard iteration algorithm only converges on a finite interval, we can anticipate using a piecewise approach over longer intervals to solve a significant family of IVPs over an arbitrarily large time domain. The initial conditions on the subsequent segments should be the final state values from the previous segment. This may sound similar to the concept of the step size control used in forward integration methods such as Runge-Kutta, or analytical continuation methods. However, the step size used by MCPI methods is typically a much larger finite interval than the steps used by the typical numerical methods, as will be shown in the examples. Furthermore, compared with the forward integration methods in which the integration errors are typically increasing with time in a secular unstable fashion, we anticipate that better stability/accuracy can be achieved from using MCPI methods because, qualitatively the largest errors from MCPI methods usually appear in the middle of the interval and the smallest errors are at the ends where adjacent (successive) segments are joined. The fundamental reason for this special characteristic of MCPI methods is a result of the chosen Chebyshev basis functions and CGL nodes that are denser at the boundaries and sparser in the middle.

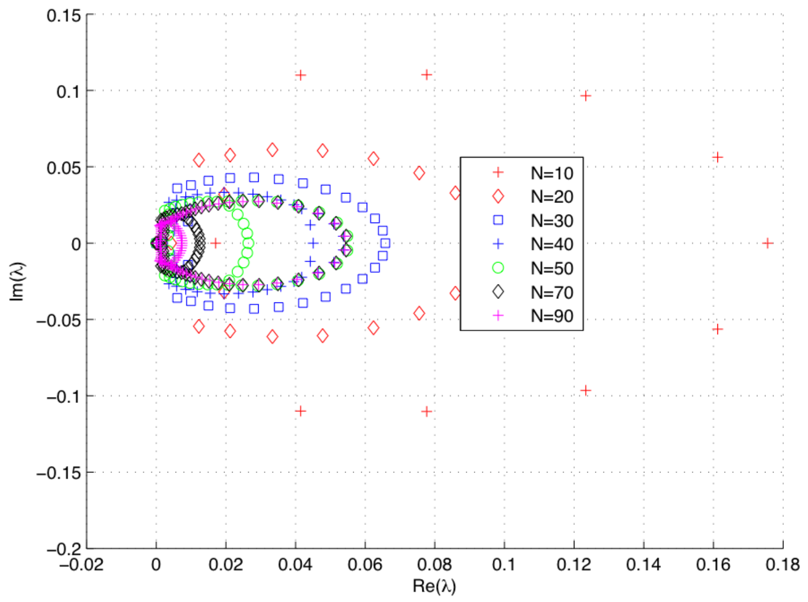


Figure IV.6: Eigenvalue Locus of TC_α, $N < 100$.

IV.F. ACPI: Adaptive Chebyshev Picard Iteration

Chebyshev polynomials are used to approximate both the state trajectory and the integrand, on the L.H.S. and R.H.S. of Eq. (4.19), respectively. It is important to note that the entire trajectory is approximated at each iteration. That is, for each iteration i all sample points τ_j are used for the trajectory approximation. Since the approximation of the R.H.S. of Eq. (4.19) is done within the integral, the upper index of the summation is only performed to $N-1$ instead of N, as on the L.H.S. Integration increases the degree of the polynomial 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. (4.19). In Bai's dissertation [17] the summation was performed from 0 to N for both the unknown trajectory (R.H.S.) and the integrand

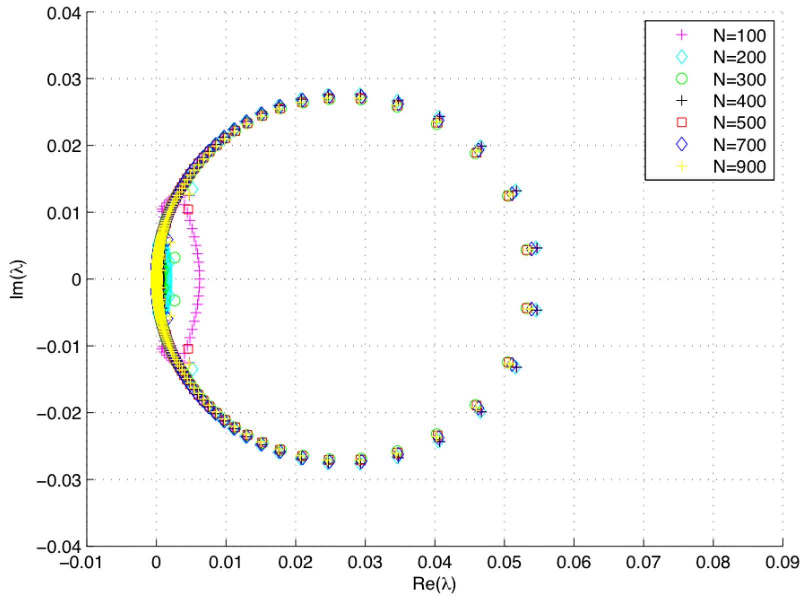


Figure IV.7: Maximum Eigenvalue Locus of TC_α, $N >= 100$.

(L.H.S.). After performing several examples using both definitions, we do not see any significant impact on the solution accuracy; specially for large N.

The confusion associated with these settings led us to consider a more relaxed, controllable and adaptive approach that could eliminate the confusion and still retain the same principals. This approach, Adaptive Chebyshev Picard Iteration (ACPI), is introduced in this section. The key is utilizing the least squares solution of the orthogonal chebyshev polynomial approximation at both sides of the picard equation. Let M represent the number of sample points (nodes), $N_{L.H.S}$ represent the degree of the chebyshev polynomial to approximate the trajectory $\boldsymbol{x}(\tau)$, and $N_{R.H.S}$ represent the degree of the chebyshev polynomial to approximate the force function $\boldsymbol{g}(\tau, \boldsymbol{x}(\tau))$. Note that $N_{L.H.S}$ and $N_{R.H.S}$ are not required to be equal, but both should take on

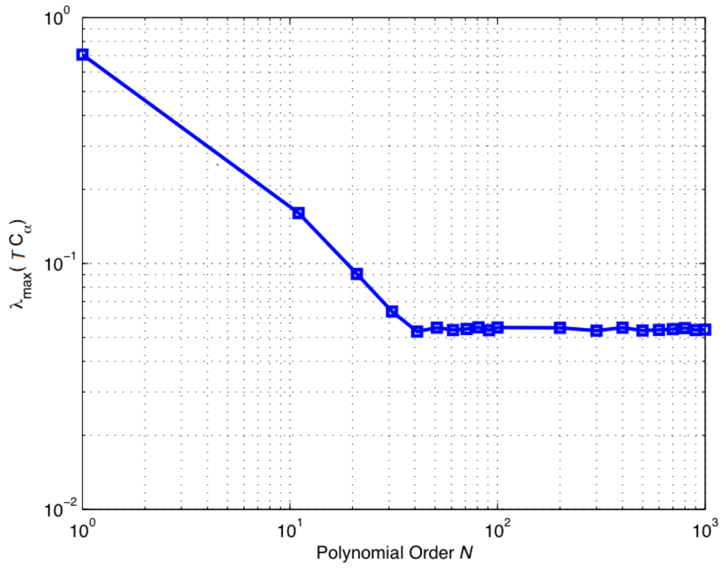


Figure IV.8: Maximum Eigenvalue Locus of TC_α versus Polynomial Order N.

a form where $N_{L.H.S} \leq M$ and $N_{R.H.S} \leq M$. Eq. (4.19) is written as

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N_{L.H.S}} \beta_k^i T_k(\tau) \equiv \boldsymbol{x}_0 + \sum_{r=0}^{N_{R.H.S}} \left\{ \Big[\int_{-1}^{\tau} T_r(s) ds \Big] F_r^{i-1} \right\} \tag{4.33}$$

where τ is the normalized time which is sampled using $\tau_j = -cos(j\pi/M), \quad j = 0, 1, 2, \cdots, M$. The chebyshev integration term between the square bracket [ ] is evaluated by

$$\Omega_r(\tau) = \int_{-1}^{\tau} T_r(s) ds = \begin{cases} \frac{1}{2}\Big[\frac{T_{r+1}(\tau)-T_{r+1}(-1)}{r+1} - \frac{T_{r-1}(\tau)-T_{r-1}(-1)}{r-1}\Big] & \text{if } r \geq 2 \\ \frac{1}{2}(\tau^2 - 1) & \text{if } r = 1 \\ \tau + 1 & \text{if } r = 0 \end{cases} \tag{4.34}$$

Eq. (4.33) becomes

$$\sum_{k=0}^{N_{L.H.S}} \beta_k^i T_k(\tau) = \boldsymbol{x}_0 + \sum_{r=0}^{N_{R.H.S}} F_r^{i-1} \Omega_r(\tau) \tag{4.35}$$

Using the discrete orthogonality property of Chebyshev polynomials, the coefficient F_r^{i-1} can be calculated immediately through

$$F_r^{i-1} = \frac{1}{c_r}\sum_{j=0}^{M} w_j \boldsymbol{g}(\tau_j, \boldsymbol{x}^{i-1}(\tau_j)) T_r(\tau_j) \tag{4.36}$$

where $c_0 = M$; $\{c_r = M/2;\ \text{for } r = 1, 2, ...N_{R.H.S} - 1\}$; $\{c_{N_{R.H.S}} = M;\ \text{if } N_{R.H.S} = M\}$; $\{c_{N_{R.H.S}} = M/2;\ \text{if } N_{R.H.S} < M\}$; $w_0 = w_M = \frac{1}{2}$; $\{w_j = 1;\ \text{for } j = 1, 2, ...M\}$. Notice each coefficient F_r^{i-1} is obtained through the summation of $(M + 1)$ independent terms, each of which is an inner product of the force function $\boldsymbol{g}(\tau, \boldsymbol{x}(\tau))$ and the Chebyshev polynomials $T_r(\tau)$ evaluated at the CGL points. Furthermore, all the coefficient vectors are independent of each other, and can therefore be computed in parallel processors. Also, and most importantly, for problems where calculating the force vector function $\boldsymbol{g}(\tau, \boldsymbol{x}(\tau))$ is time consuming, significant time performance improvement can be achieved by simultaneous computation of $\boldsymbol{g}(\tau_j, \boldsymbol{x}(\tau_j))$ at the nodes on $M + 1$ parallel processors.

Note that the two terms of the R.H.S of Eq. (4.35) are known at *every* previous iteration $(i - 1)$. Therefore, we can define the variable $G^{i-1}(\tau)$ to represent those two terms at $(i - 1)^{th}$ iteration. Then Eq. (4.35) is written as

$$\begin{aligned}
\sum_{k=0}^{N_{L.H.S}} \beta_k^i T_k(\tau) &= \underbrace{\boldsymbol{x}_0 + \sum_{r=0}^{N_{R.H.S}} F_r^{i-1}\Omega_r(\tau)}_{G^{i-1}(\tau)} \\
\sum_{k=0}^{N_{L.H.S}} \beta_k^i T_k(\tau) &= G^{i-1}(\tau) \qquad (4.37)
\end{aligned}$$

Since $G^{i-1}(\tau)$ is calculated at the $(i - 1)^{th}$, then we can calculate the coefficient β_k^i using the discrete orthogonality property of Chebyshev polynomials

$$\beta_k^i = \frac{1}{c_k}\sum_{j=0}^{M} w_j T_k(\tau_j) G^{i-1}(\tau_j) \tag{4.38}$$

Not only does this approach eliminate the confusion regarding the degree of the force function approximation, N versus $N-1$, but it also allows *freedom* to choose different polynomial degree to approximate the trajectory and the force function; i.e. $N_{L.H.S} \leq M$ can be different than $N_{R.H.S} \leq M$. Since both of the polynomial degree $N_{L.H.S}$ and $N_{R.H.S}$ can be less than the time sampling points M, it results in the vectorized T and C_α having a smaller size compared to the original MCPI approach [17]. This reduces the computation cost as less operations are performed during iteration due to the reduced matrix sizes. In addition, the different choices of the approximation degree can be utilized to adapt the iteration process by tuning $N_{L.H.S}$ and $N_{R.H.S}$. The cost benefit of this approach depends strongly on the problem that we solve. Some examples showed that a speed up of greater than 1.5X can be achieved compared with the original MCPI approach [17].

IV.F.1. VACPI: Vectorized Adaptive Chebyshev Picard Iteration

A compact vector-matrix approach, initially described by Bai [17] can be used to reformulate the Adaptive Picard Chebyshev Iteration to make it compatible to parallel implementation. We start by vectorizing the trajectory and the force function approximation, as given in Eq. (4.19) and Eq. (4.17) respectively,

$$\boldsymbol{x}^i(\tau) = \sum_{k=0}^{N_{L.H.S}} \beta_k^i T_k(\tau) \equiv \boldsymbol{\Phi}_{L.H.S}^T \boldsymbol{\beta}^i \tag{4.39}$$

$$\boldsymbol{g}(\tau, \boldsymbol{x}^{i-1}(\tau)) = \sum_{r=0}^{N_{R.H.S}} F_r^{i-1} T_r(\tau) \equiv \boldsymbol{\Phi}_{R.H.S}^T \boldsymbol{F}^{i-1} \tag{4.40}$$

where

$$\begin{aligned}
\boldsymbol{\beta}^i &= \begin{bmatrix} \beta_0^i & \beta_1^i & \beta_2^i & \cdots & \beta_{N_{L.H.S}}^i \end{bmatrix}^T && (4.41)\\
\boldsymbol{F}^{i-1} &= \begin{bmatrix} F_0^{i-1} & F_1^{i-1} & F_2^{i-1} & \cdots & F_{N_{R.H.S}}^{i-1} \end{bmatrix}^T && (4.42)\\
\boldsymbol{\Phi}_{L.H.S} &= \begin{bmatrix} T_0(\tau) & T_1(\tau) & T_2(\tau) & \cdots & T_{N_{L.H.S}}(\tau) \end{bmatrix}^T && (4.43)\\
\boldsymbol{\Phi}_{R.H.S} &= \begin{bmatrix} T_0(\tau) & T_1(\tau) & T_2(\tau) & \cdots & T_{N_{R.H.S}}(\tau) \end{bmatrix}^T && (4.44)
\end{aligned}$$

Using the discrete orthogonality property of Chebyshev polynomials, the coefficient $\boldsymbol{F}^{i-1}$ can be calculated through

$$\begin{aligned}
\boldsymbol{F}^{i-1} &= (\boldsymbol{\Phi}_{R.H.S}^T W \boldsymbol{\Phi}_{R.H.S})^{-1} \boldsymbol{\Phi}_{R.H.S}^T W \boldsymbol{g}(\tau, \boldsymbol{x}^{i-1}(\tau))\\
&= (\boldsymbol{\Phi}_{R.H.S}^T W \boldsymbol{\Phi}_{R.H.S})^{-1} \boldsymbol{\Phi}_{R.H.S}^T W \boldsymbol{g}^{i-1} && (4.45)
\end{aligned}$$

Consistent with the classical orthogonality conditions, presented in Chapter II, for the chebyshev polynomials, we adopt the weight matrix $W = diag\{\frac{1}{2}, 1, 1, \cdots, 1, 1, \frac{1}{2}\}$. The typical element of $(\boldsymbol{\Phi}_{R.H.S}^T W \boldsymbol{\Phi}_{R.H.S})$ is a *discrete inner product* denoted $m_{\alpha\beta}$. The typical pair of orthogonal basis functions' inner products obey

$$m_{\alpha\beta} = \langle \phi_\alpha, \phi_\beta \rangle \equiv \sum_{j=0}^{M} w_j \phi_\alpha(\tau_j) \phi_\beta(\tau_j) = \begin{cases} 0 & , \; for \; \alpha \neq \beta \\ c_\alpha & , \; for \; \alpha = \beta \end{cases} \qquad (4.46)$$

The orthogonal basis functions (chebyshev polynomials) reduces the matrix inverse to the division of their scalar diagonal terms; leading to

$$V_{R.H.S} = (\boldsymbol{\Phi}_{R.H.S}^T W \boldsymbol{\Phi}_{R.H.S})^{-1} = diag\Big\{ 1/c_0, \;\; 1/c_1, \;\; , \cdots, \;\; 1/c_{N_{R.H.S}} \Big\} \qquad (4.47)$$

Therefore Eq. (4.45) becomes

$$\boldsymbol{F}^{i-1} = V_{R.H.S} \boldsymbol{\Phi}_{R.H.S}^T W \boldsymbol{g}^{i-1} \qquad (4.48)$$

Vectorizing the second term in Eq. (4.35)

$$\sum_{r=0}^{N_{R.H.S}} F_r^{i-1}\Omega_r(\tau) \equiv \boldsymbol{\Omega}^T \boldsymbol{F}^{i-1} \tag{4.49}$$

where

$$\boldsymbol{\Omega} = \Big[\Omega_0(\tau) \;\; \Omega_1(\tau) \;\; \Omega_2(\tau) \;\; \cdots \;\; \Omega_{N_{R.H.S}}(\tau)\Big]^T \tag{4.50}$$

Substituting Eq.(4.48) into Eq. (4.49) yields

$$\begin{aligned} \sum_{r=0}^{N_{R.H.S}} F_r^{i-1}\Omega_r(\tau) &= \boldsymbol{\Omega}^T \boldsymbol{F}^{i-1} && (4.51)\\ &= \boldsymbol{\Omega}^T V_{R.H.S} \boldsymbol{\Phi}_{R.H.S}^T W \boldsymbol{g}^{i-1} && (4.52) \end{aligned}$$

Analogous to the MCPI development, let us define the matrix $A_\alpha = \boldsymbol{\Omega}^T V_{R.H.S} \boldsymbol{\Phi}_{R.H.S}^T W$. Then Eq. (4.52) becomes

$$\sum_{r=0}^{N_{R.H.S}} F_r^{i-1}\Omega_r(\tau) = A_\alpha \boldsymbol{g}^{i-1} \tag{4.53}$$

Substituting Eq. (4.39) and Eq. (4.53) into Eq. (4.35)

$$\boldsymbol{\Phi}_{L.H.S}^T \boldsymbol{\beta}^i = \boldsymbol{x}_0 + A_\alpha \boldsymbol{g}^{i-1} \tag{4.54}$$

For $N_{L.H.S} = M$, Eq. (4.54) leads to the original MCPI derivation. However, for general *adaptive* case, when $N_{L.H.S} \leq M$, we calculate the coefficient vector $\boldsymbol{\beta}^i$ using the discrete orthogonality property of Chebyshev polynomials. Note that the R.H.S terms are known at the $(i-1)^{th}$ iteration. The coefficient vector $\boldsymbol{\beta}^i$ is calculated through

$$\boldsymbol{\beta}^i = V_{L.H.S} \boldsymbol{\Phi}_{L.H.S}^T W \Big\{\boldsymbol{x}_0 + A_\alpha \boldsymbol{g}^{i-1}\Big\} \tag{4.55}$$

where

$$V_{L.H.S} = (\boldsymbol{\Phi}_{L.H.S}^T W \boldsymbol{\Phi}_{L.H.S})^{-1} = diag\Big\{1/c_0, \;\; 1/c_1, \;\; ,\cdots, \;\; 1/c_{N_{L.H.S}}\Big\} \tag{4.56}$$

Similarly, let us define the matrix $A_x = V_{L.H.S}\boldsymbol{\Phi}^T_{L.H.S}W$. Then Eq. (4.55) becomes

$$\boldsymbol{\beta}^i = A_x\Big\{\boldsymbol{x}_0 + A_\alpha \boldsymbol{g}^{i-1}\Big\} \tag{4.57}$$

Thus, the trajectory algorithm follows as

$$\begin{aligned} \boldsymbol{x}^i(\tau) &= \boldsymbol{\Phi}^T_{L.H.S}\boldsymbol{\beta}^i && (4.58)\\ &= \boldsymbol{\Phi}^T_{L.H.S}A_x\Big\{\boldsymbol{x}_0 + A_\alpha \boldsymbol{g}^{i-1}\Big\} && (4.59)\end{aligned}$$

Figure IV.9 illustrates the flowchart of the matrix-vector form of the ACPI algorithm. The flowchart takes the same flow as the MCPI except the coefficients are different here. This indicates the implementation is very similar. Note that the coefficients calculations are computed once and then used during the iteration process.

This new approach can be also utilized in modifying the current MCPI [17] to solve the Boundary Value Problems. The benefit would be improving the convergence speed.

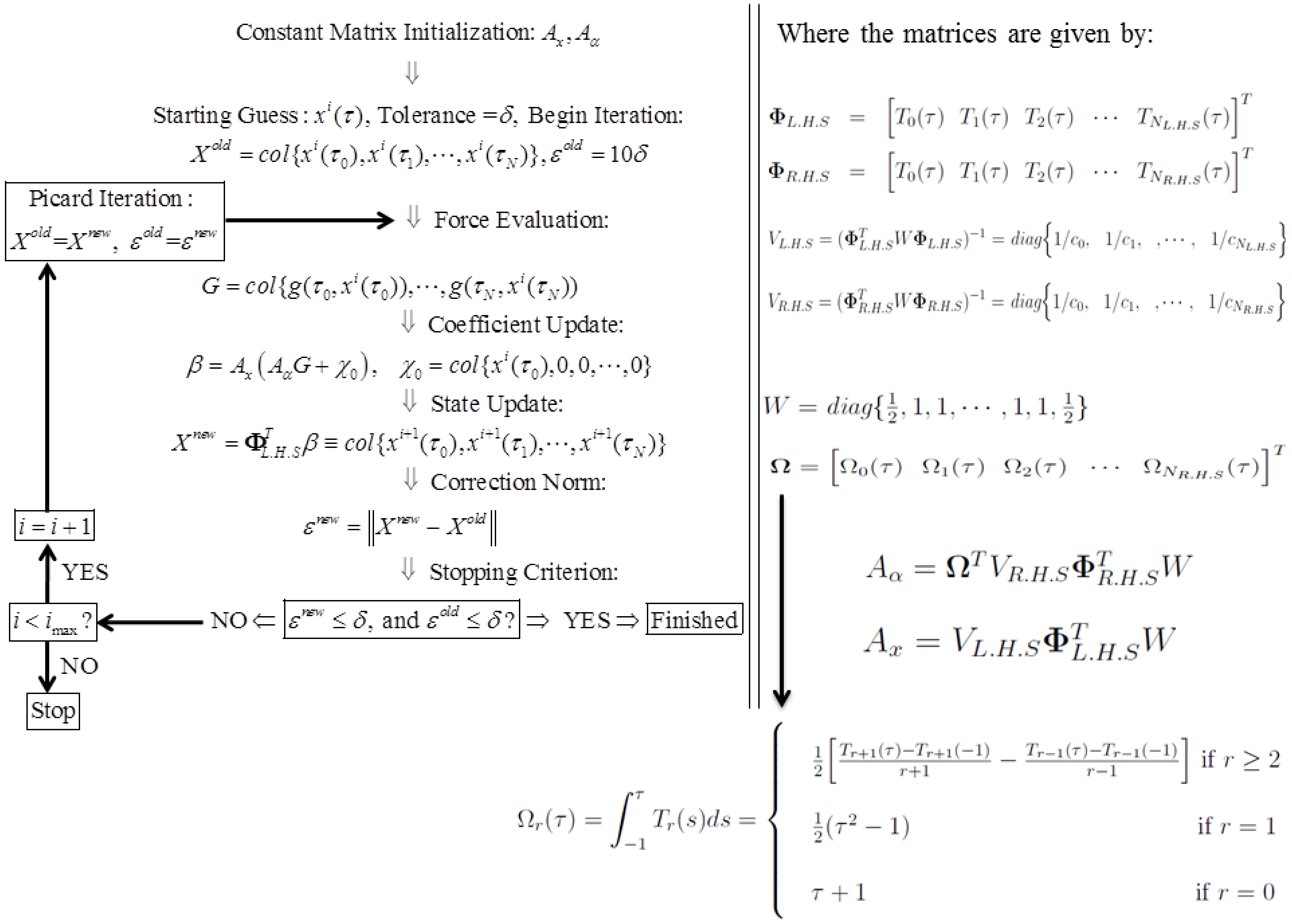

Constant Matrix Initialization: A_x, A_α
Starting Guess : $x^i(\tau)$, Tolerance $=\delta$, Begin Iteration:
$X^{old} = col\{x^i(\tau_0), x^i(\tau_1), \cdots, x^i(\tau_N)\}, \varepsilon^{old} = 10\delta$
Picard Iteration :
$X^{old} = X^{new}, \ \varepsilon^{old} = \varepsilon^{new}$
Force Evaluation:
$G = col\{g(\tau_0, x^i(\tau_0)), \cdots, g(\tau_N, x^i(\tau_N))$
Coefficient Update:
$\beta = A_x(A_\alpha G + \chi_0), \quad \chi_0 = col\{x^i(\tau_0), 0, 0, \cdots, 0\}$
State Update:
$X^{new} = \Phi^T_{L.H.S}\beta \equiv col\{x^{i+1}(\tau_0), x^{i+1}(\tau_1), \cdots, x^{i+1}(\tau_N)\}$
Correction Norm:
$\varepsilon^{new} = \|X^{new} - X^{old}\|$
Stopping Criterion:
NO
$\varepsilon^{new} \le \delta$, and $\varepsilon^{old} \le \delta$?
YES
Finished
$i < i_{\max}$?
YES
$i = i + 1$
NO
Stop
Where the matrices are given by:
$\Phi_{L.H.S} = \begin{bmatrix} T_0(\tau) & T_1(\tau) & T_2(\tau) & \cdots & T_{N_{L.H.S}}(\tau) \end{bmatrix}^T$
$\Phi_{R.H.S} = \begin{bmatrix} T_0(\tau) & T_1(\tau) & T_2(\tau) & \cdots & T_{N_{R.H.S}}(\tau) \end{bmatrix}^T$
$V_{L.H.S} = (\Phi^T_{L.H.S} W \Phi_{L.H.S})^{-1} = diag\left\{1/c_0, \ 1/c_1, \ , \cdots, \ 1/c_{N_{L.H.S}}\right\}$
$V_{R.H.S} = (\Phi^T_{R.H.S} W \Phi_{R.H.S})^{-1} = diag\left\{1/c_0, \ 1/c_1, \ , \cdots, \ 1/c_{N_{R.H.S}}\right\}$
$W = diag\{\frac{1}{2}, 1, 1, \cdots, 1, 1, \frac{1}{2}\}$
$\boldsymbol{\Omega} = \begin{bmatrix} \Omega_0(\tau) & \Omega_1(\tau) & \Omega_2(\tau) & \cdots & \Omega_{N_{R.H.S}}(\tau) \end{bmatrix}^T$
$A_\alpha = \boldsymbol{\Omega}^T V_{R.H.S} \Phi^T_{R.H.S} W$
$A_x = V_{L.H.S} \Phi^T_{L.H.S} W$
$\Omega_r(\tau) = \int_{-1}^{\tau} T_r(s)ds = \begin{cases} \frac{1}{2}\left[\frac{T_{r+1}(\tau) - T_{r+1}(-1)}{r+1} - \frac{T_{r-1}(\tau) - T_{r-1}(-1)}{r-1}\right] & \text{if } r \ge 2 \\ \frac{1}{2}(\tau^2 - 1) & \text{if } r = 1 \\ \tau + 1 & \text{if } r = 0 \end{cases}$

Figure IV.9: Flowchart of the Matrix-vector Form of the ACPI Algorithm.

IV.G. Numerical Examples

Computations for the first five numerical examples (other than the parallel computations) are performed on a conventional PC. The settings of the computer and the development environment used are the following

- Intel(R) Pentium(R) D CPU 3.4GHz, 3.4GHz, 2.0GB of RAM
- Windows XP Operating System
- MATLAB R2009b
- NVIDIA GeForce 9400GT Graphics Card
- Microsoft Visual Studio 2005

IV.G.1. Review Bai's Examples

In this section we review a few simply examples that were previously presented in Bai's work [17,22,62,73]. These are reintroduced here in order to set a framework for the more complex analysis that follows in the latter part of this chapter.

IV.G.1.a. Example 1: A First Order Nonlinear System

Consider a dynamic equation

$$\frac{dy}{dt} = f(t,y) = cos(t + \varepsilon y),\ t_0 = 0,\ t_f = 256\pi,\ y(t_0) = 1,\ \varepsilon = 0.001. \qquad (4.60)$$

Fukushima [60] suggested this problem, that has an analytical solution, as a benchmark with a known true value for conducting convergence accuracy studies. We first solve an IVP for Eq (4.1) by comparing the MCPI method implemented in MATLAB, and a “garden variety” solver ODE45 (i.e. a Runge-Kutta $4-5$ method also implemented in MATLAB). For more significant nonlinear problems we use more sophisticated (and efficient) integrators as the basis for comparison. The results are shown for this first example in Figure IV.10. The immediately following conclusions are drawn.

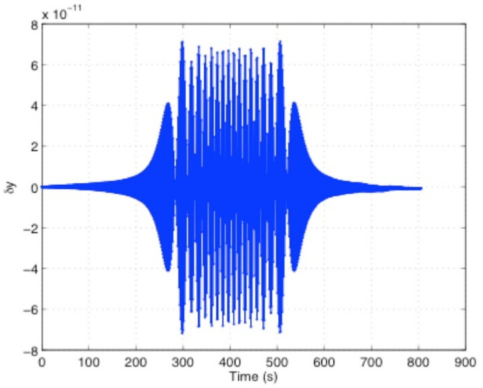
1. For this tuning, the MCPI solutions have about one order of magnitude better accuracy than the ODE45 solutions. The CPU time using ODE45 is about 40 times slower than the CPU time using MCPI methods. We further note that orders (N) up to of several thousand are feasible with MCPI, without numerical difficulty, owing to the orthogonality (no matrix inverses) and, especially, highly efficient recursions based on simple inner products. The optimal order is typically much less, but obviously *high order approximation* in numerical integration now takes on a new meaning.

2. As we show later, these solutions have not taken advantage of the fact that the long intervals are subdivided which will reduce the order of the required polynomials, thus more speedup is achieved by the MCPI methods when implemented on a serial machine. Furthermore, if parallel computation environment is available, further speedup is obtained because the uncoupled function evaluations matrix operations are distributed to different processors.

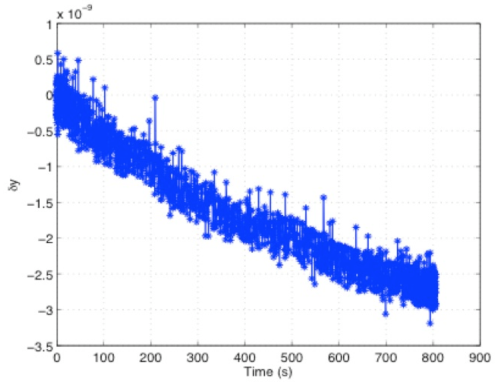
3. While the errors from the reference ODE45 solution display a typical secular increase, which is a pattern common to all forward integration methods, the errors using MCPI method have the maximum values near the middle of the in-

terval and the smallest errors at the boundaries. To graphical precision on a log scale there is negligible secular error growth in this example. The fundamental reason is due to the positioning of the CGL nodes, dense at the boundaries and sparse in the middle. Notice this special feature makes MCPI methods more attractive than the forward integration methods in reducing the global errors for long time integrations, where different segments have to be patched together at the terminal points of each solution interval where the errors are typically smallest.

4. We note convergence is obtained up to some problem dependent-maximum final time. For linear problems this is determined. For nonlinear problems, approximation or adaptive tuning is required. The interval for practical convergence is greater than 256π (~ 128 oscillation periods).



(a) MCPI.



(b) ODE45.

Figure IV.10: Integration Errors for Example 1. The CPU Time is 0.042 s and 1.722 s Respectively.

For qualitative purpose, we note that expanding the dynamic equation in ϵ leads to the approximate equation $\frac{dy}{dt} = cos(t) - \varepsilon sin(t)y + ...$ so the linear (in y) coefficient is bounded by $\pm\epsilon$. Though not rigorous, we estimate from the above analysis of the analogous constant coefficient linear system that convergence might be expected if $H < \frac{2}{\varepsilon}\frac{1}{\lambda_{max}(C_x C_\alpha)}$. Thus with the polynomial order $N > 100$, the idealized convergence analysis suggests that H should be less than $\frac{2}{(0.001)(0.05)} \sim 40,000\,\mathrm{s}$. These approximations are typically useful for starting estimates. In this case we verified excellent convergence for the nonlinear system was actually achieved if $H < 800 \times 2\pi \approx 5026.5$, so the linear estimate is optimistic in this example. Perhaps the most striking feature of this example is that high precision is achieved over long time periods including many main period oscillations of a nonlinear system, whereas many time steps per period are required by all step-by-step solvers known to achieve comparable precision.

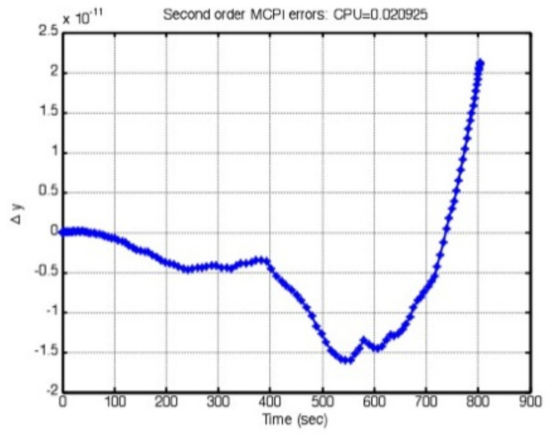
IV.G.1.b. Example 2: Second Order Nonlinear System

The following second order differential equation has the same analytical solution as the above first order example, but allows us to conduct convergence studies for integrators designed for second order systems.

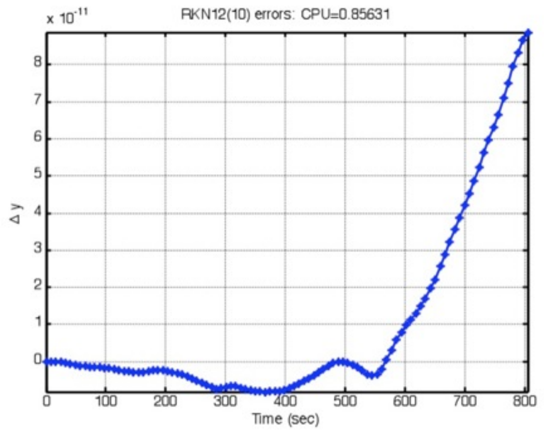
$$\frac{d^2y}{dt^2} = f(t,y) = -sin(t+\varepsilon y) - \frac{1}{2}\varepsilon sin(2t+2\varepsilon y), \quad t_0 = 0, \quad t_f = 256\pi, \quad y(t_0) = 1, \quad \dot{y}(t_0) = 1 \tag{4.61}$$

Among the many convergent possibilities, we have tuned the second order MCPI methods to use a Chebyshev polynomial of order 130 to approximate the solution over an interval length of 16π (8 periods of unperturbed oscillations), and found convergent solutions on the sixteen segments of 16π duration that are patched together to generate the final solution. At the starting iteration, all the positions and velocities at the $N+1$ CGL nodes are simply chosen as the straight line ensuing from the initial position and the initial velocity, thus a very poor starting guess is provided for the MCPI methods so that the timing results are very conservative. To provide a more meaningful comparison vis-a-vis relative efficiency, we adopt the 12^{th} order Runge-Kutta-Nystrom algorithm RKN12(10) with adaptive step size control. The errors of MCPI and RKN12(10) are shown in Figures IV.11a, IV.11b. We can see the slightly better accuracy achieved by the MCPI solution. MCPI also obtained a speedup of about 32X relative to RKN12(10). This speedup is in spite of the vector-matrix nature of the MCPI algorithm. As the force model becomes complicated, this speedup advantage on a serial machine might be expected to be smaller. The RKN12(10) algorithm calls the function evaluation routine $14,974$ times, whereas totally MCPI takes 113 Picard iterations, which leads to $113 \times (130 + 1) = 14,803$ function evaluations (remarkably, almost the same number in this case). Thus on a serial machine, even with a very poor starting solution estimate, MCPI requires essentially the same number of function evaluations as does RKN12(10). However, it is vitally important to recognize that the MCPI acceleration evaluations are independent, since the entire path approximation is available at once on each Picard path approximation iteration, so in an ideal parallel environment where we can distribute the function evaluations on the N+1 CGL nodes onto $N+1$ processors, the theoretical speedup a factor is 131. We can approach that limit if 131X or more cores are available since little shared memory is involved. Additionally, comparing the computational time and accuracy of this tuned second order MCPI with the previous first order MCPI using one segment, we see the benefit to use the second order formulation and also the potential for even better accuracy and more speedup when careful tuning is applied to the MCPI methods.

Figures IV.11a and IV.11b show the errors are in the 11^{th} significant figure for both solutions, although the MCPI solution has about $\frac{1}{4}$ the error norm of the RKN12(10) solution. The speedup achieved on a serial processor was 32X. The theoretical speedup on a parallel processor with over 130 cores is two additional orders of magnitude for this problem. Impressive potential obviously exists, if these results for "toy" idealized problems extend to the problems of orbit mechanics. In the results presented below, the test cases to date indicate that these speedups are typical for the more nonlinear problems of central practical interest.



(a) MCPI Error History, Serial CPU Time Cost = 0.028 s.



(b) RKN12(10) Error History, Serial CPU Time Cost = 0.856 s.

Figure IV.11: Second Order MCPI and RKN12(10) Error History.

IV.G.1.c. Example 3: Integration of Unperturbed Keplerian Motion (Natural Second Order System Example)

The two-body problem is a classic example problem that has been used to frequently quantify and compare the performance of ODE solvers [21,59,74–76], because while nonlinear, it has an exact analytical solution and it is a physically important problem. We choose to use an example that integrates a three dimensional near circular orbit for one week to allow us draw more practical insight. The dynamic equations are

$$\frac{d^2x}{dt^2} = -\frac{\mu}{r^3}x; \;\; \frac{d^2y}{dt^2} = -\frac{\mu}{r^3}y; \;\; \frac{d^2z}{dt^2} = -\frac{\mu}{r^3}z; \;\; r^2 = x^2 + y^2 + z^2. \tag{4.62}$$

We look at a low eccentricity problem, see Table IV.1 as well as a high eccentricity problem, see Table IV.2. The six classical unperturbed orbital elements are:

1. Both the MCPI methods and RKN12(10) are tuned such that sub-millimeter position accuracy, relative to the exact analytical solution, is achieved for the whole week. A Chebyshev polynomial of order 40 is chosen for the MCPI method and the (convergent) segment length is selected to be 5400 s (about one orbit period). The classical F&G solution [77] provides an analytical truth that is used to calculate the solution relative errors, which are smaller than 10^{-11}. Several observations are summarized in this section. The computational time is 0.2639 s for MCPI and 1.8882 s for RKN12(10). Thus with slightly better accuracy in both position and velocity, MCPI achieved a speedup factor of seven.

Table IV.1: Classical Orbital Elements for Low Eccentricity Orbit.

EPOCH STATE			EPOCH ELEMENTS		
r_x	-464.856	Km	a	6644.754	Km
r_y	6667.880	Km	e	0.01	
r_z	574.231	Km	i	68	deg
v_x	-2.8381186	Km/sec	ω	-160	deg
v_y	-0.7871898	Km/sec	Ω	92	deg
v_z	7.0830275	Km/sec	M	164	deg
		PERIOD, T_p	90	mins	

2. RKN calls the differential equations 29662 times. Using an initial starting solution that the position and velocity at all the CGL nodes are the same as the initial position and velocity, MCPI completed 2465 iterations in total. Thus on a serial machine, the ratio of function evaluation of RKN over MCPI is $29662/(2465 \times 41) = \ 0.3$. However, in an ideal parallel environment where we can distribute the function evaluation on the $N + 1$ CGL nodes to $N + 1$ processors, the ratio is $29662/2465 = \ 12$.

3. The reason for the speedup of MCPI over RKN in a serial implementation lies in two aspects. The first is that the matrix-vector form of the MCPI approach is computationally very efficient, and the second attributes to the large step size that can be used with MCPI. For RKN12(10), the maximum is 629.4089 s, and the mean is 363.4615 s. This is approximately 7% of the one full orbit step size

that is used during MCPI, and thus a significant qualitative difference exists in approximating a long time interval dynamical path versus taking small steps along it!

4. We have gained some preliminary insight about how to tune the polynomial order and the segment length. Figures IV.12a, IV.12b show the computational time and accuracy for the MCPI method when the orders are chosen from 40 to 300, and the segment lengths are chosen from about 10% of the orbit periods up to 2.2 of the orbit period. The computational time is shown in Figure IV.12a and its contour plot is displayed in Figure IV.12b. The minimum computation time is 0.1847 s, which is obtained with a choice of $N = 50$ and segment length of 10260 s (about 1.9 orbits). Notice, although we currently do not have an efficient way to find this optimal setting for the minimum computational time, we have a large region (time intervals and approximation areas) where we can obtain sub-optimal solutions (the region where the computational time is less than one second, which is still significantly faster than RKN12(10)). Another way of looking at this issue, the very flat surface indicates a large family of near optimal tunings exist. The most time consuming settings are the cases where an unnecessary high order Chebyshev polynomial is used with many small time segments to reach a multi-orbit final time.

We also characterize the solution errors as the maximum global relative error

$$e = max\left(\frac{\mid \boldsymbol{r}(MCPI,t) - \boldsymbol{r}(FG,t) \mid}{\mid \boldsymbol{r}(FG,t) \mid}\right) + max\left(\frac{\mid \boldsymbol{v}(MCPI,t) - \boldsymbol{v}(FG,t) \mid}{\mid \boldsymbol{v}(FG,t) \mid}\right), \tag{4.63}$$

where the first argument indicates the method to compute the solution, and FG denotes the classical two-body analytical solution. The significant figures shown in Figure IV.12c are defined here as $-log_{10}(e)$.

Looking at Figures IV.12a, IV.12b it is clear that there is a large region where more than eleven significant digits, in less than one second of computational time, may be obtained. The irregularity of the 11^{th} and especially the 12^{th} significant digit contour is a consequence of the solution accuracy approaching the noisy precision limit associated with finite word arithmetic, which in turn are associated with a machine precision of 16 digit arithmetic. The RKN12(10) algorithm also experiences similar bumpy convergence when it approaches 12 digit accuracy, but only the step size tolerance was available for tuning. In this case, we have the choice over a large space of interval lengths and orders to achieve 12 digit accuracy, but of course, 9 digit accuracy for orbit problems is typically considered sufficient for "engineering accuracy" since this already corresponds to cm precision. For runtime efficiency, the optimal region in this tuning space for serial machines is as near the top left boundary of Figures IV.12b, IV.12d as accuracy allows. However, for parallel machines, it is near the top boundary but further to the right. We accept the longest practical convergence interval, since the order is adjusted, by moving right for larger N equal to the number of cores available (so the number $N + 1$ of function evaluations along each iterative trajectory is carried out simultaneously to achieve a theoretical speedup of $(N + 1)$. The flatness of the efficiency and accuracy surfaces and their large overlapping sweet spots permit a large space for adjustment to take full advantage of various parallel architectures.

IV.G.1.d. MCPI Preliminary Results for Propagating a Family of Perturbed Orbits

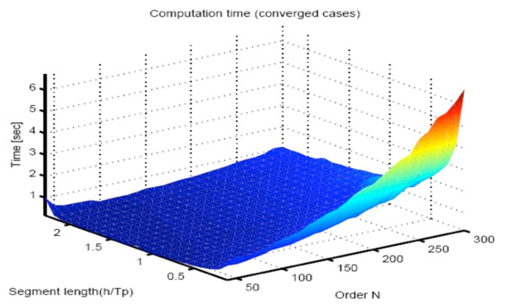
The ability to propagate satellite motion quickly and accurately is one of the major factors that affect the performance for tasks such as collision avoidance. For these tasks, numerical integration of the satellite motion with even more accurate and complicated perturbation force models has become necessary. It is possible that a degree 200 and order 200 (or higher) gravity model, and a time-varying atmospheric density model is required to adequately model perturbation accelerations. In the following preliminary studies we include only the zonal harmonic perturbation forces up to the fifth order in the dynamic models, and we investigate how the performance of the algorithms change as the force model becomes more complicated. Including zonal harmonic perturbations up to the order of k leads to

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r} + \sum_{i=2}^{k} \boldsymbol{a}_d^i, \tag{4.64}$$

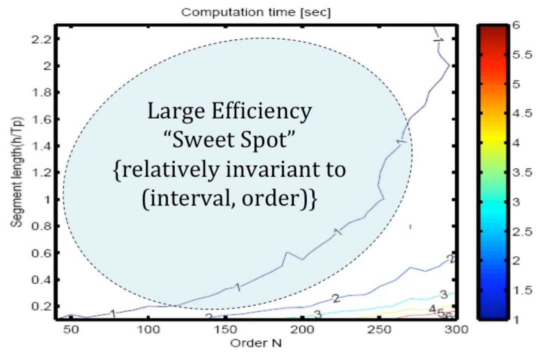
where $\mathbf{a}_d^i$ is the i^{th} of k gravity spherical harmonic perturbation terms. We compare the computational time and the number of function evaluations when using MCPI and RKN12(10) respectively. Both low eccentricity and high eccentricity problems are examined using the four perturbed force models below.

- Inverse-square gravity force + J2 perturbation
- Inverse-square gravity force + J2 perturbation + J3 perturbation
- Inverse-square gravity force + J2 perturbation + J3 perturbation + J4 perturbation
- Inverse-square gravity force + J2 perturbation + J3 perturbation + J4 perturbation + J5 perturbation

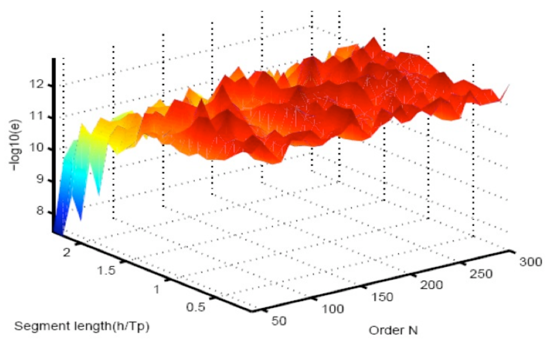
The purpose of considering the perturbations in this way is to assess the role that model complexity plays on the relative efficiency and accuracy of MCPI in comparison to existing methods. The eccentricity is varied for orbits near circular to very eccentric, in order to assess the degree to which rapidly varying nonlinearity impacts the relative merits of several algorithms. MCPI results are presented for both low and high eccentricity orbits.



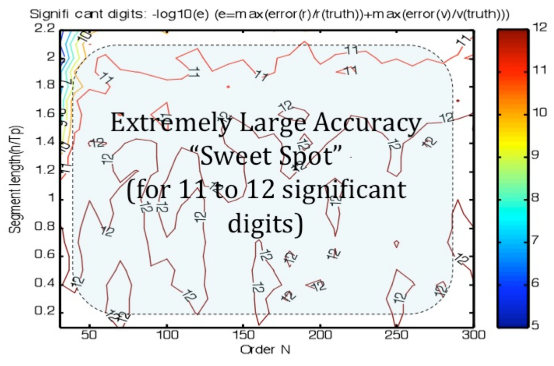
(a) MCPI CPU Computational Time.



(b) Time Efficiency Sweet Spot.



(c) MCPI Significant Figures.



(d) Accuracy Sweet Spot.

Figure IV.12: Computation Time and Significant Figures for MCPI.

IV.G.1.e. Example 4: Integration of Perturbed Orbits with Low Eccentricity (e = 0.01)

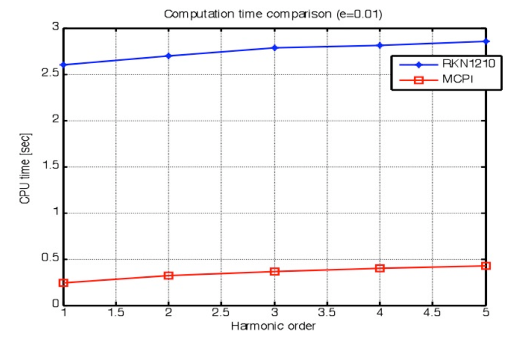
The initial conditions for this example are the same as those used in Example 2. For the MCPI method, the Chebyshev polynomial order is 40 and the segment length is 5400 seconds, which have been tested in the unperturbed problem to provide sub-millimeter position accuracy. For the four perturbed cases, although no analytical solutions are available, we have verified that the relative energy changes (kinetic & potential) for both methods are in the range of 10^{-13}. The computational times for the two methods are shown in Figure IV.13a and the comparison results are shown in Figures IV.13b-IV.13f. Using the energy check avoids the necessarily of interpolating the solution to match the points, if the absolute position and velocity error integrated by the two methods is calculated. The order "1" case is the unperturbed Example 2 that we studied before and we include it here to illustrate the performance trend with respect to the complication level of the perturbation models. Figure IV.13b shows that the MCPI method achieved six to eleven speedup over RKN12(10). Figure IV.13c shows that RKN12(10) calls about 30% of the number of function evaluations required by the MCPI method. Figure IV.13d shows that in an ideal parallel computation environment where we can distribute the force evaluation on the $(N + 1)$ CGL nodes onto $(N + 1)$ processors, RKN12(10) calls about twelve times of the number of function evaluations required by the MCPI method. Although the speedup achieved by the MCPI is shown to be decreasing on Figure IV.13b in a serial implementation, we anticipate that the trend will change to be beneficial to the MCPI method on an advanced parallel machine due to the following three reasons:

1. Figure IV.13a and IV.13b show that as more perturbation terms are included, the function call ratio of RKN12(10) over MCPI is increasing.

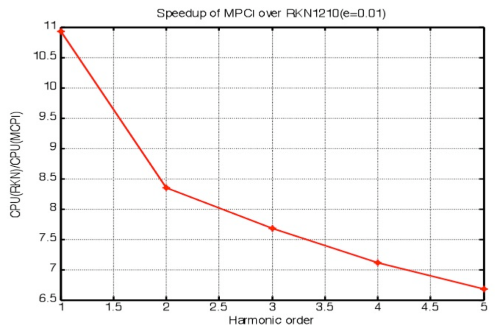
2. Figures IV.13e and IV.13f show some preliminary results regarding the speedups obtained from the GPU-accelerated MCPI over the MATLAB MCPI when used with INVIDIA GeForce 9400GT. These demonstrate that the speedup achieved by the parallel MCPI code increases significantly, as either the perturbation forces become more complicated or higher order polynomials are used.

3. As we discussed earlier, the parameters for the MCPI method used here are not claimed as the optimal settings; further optimizations are possible. The current MCPI implementation does not utile step size adaptation. However, there could be a possible implementation where the time segment (step size) can be adapted in a such way to optimize the computation and maintain a specific tolerance.

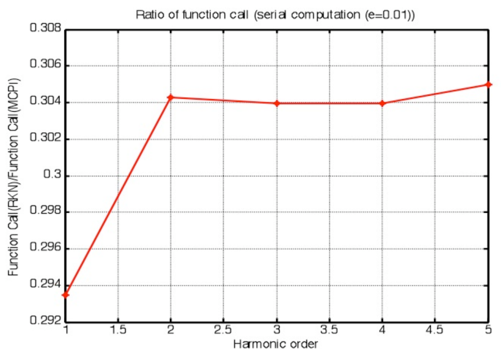
As mentioned before, these examples were previously presented in Bai's work [17, 22, 62, 73]. These are reintroduced here in order to set a framework for the more complex analysis that follows in the sext sections.



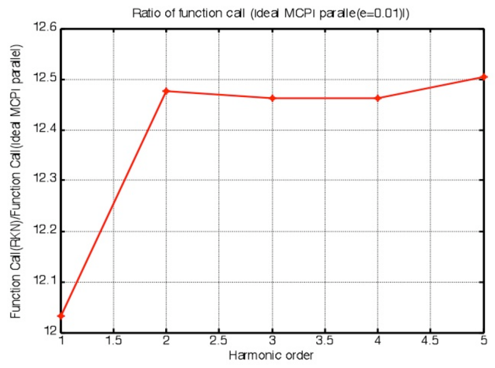
(a) Computational Time of MCPI and RKN12(10) ($e = 0.01$).



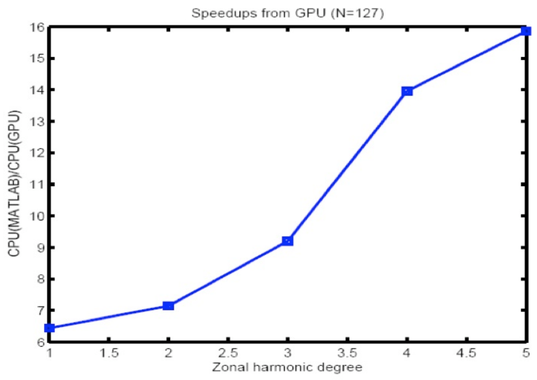
(b) Speedup of MCPI over RKN12(10) ($e = 0.01$).



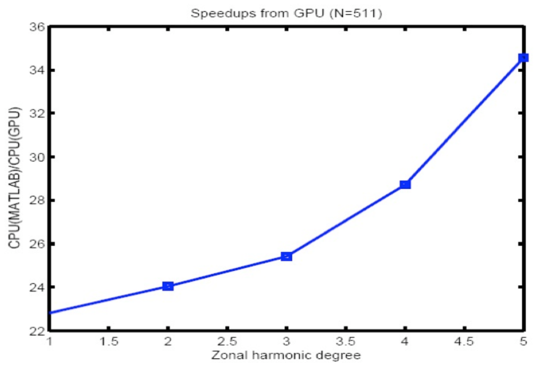
(c) Ratio of Function Calls in a Serial Computer ($e = 0.01$).



(d) Ratio of Function Calls in an Ideal Parallel Architecture ($e = 0.01$).

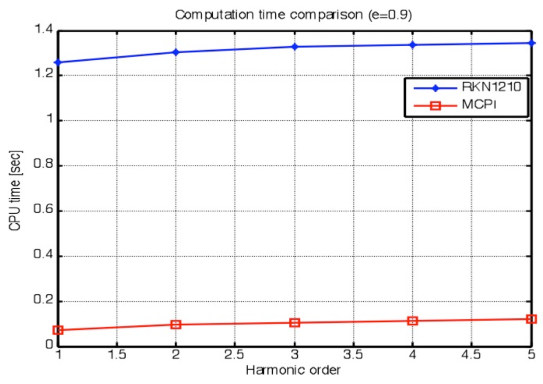


(e) Speedup of GPU-MCPI ($N = 127$).

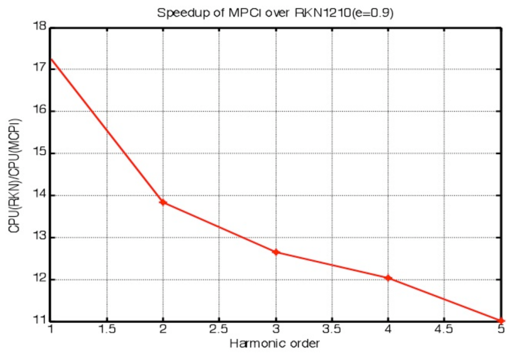


(f) Speedup of GPU-MCPI ($N = 511$).

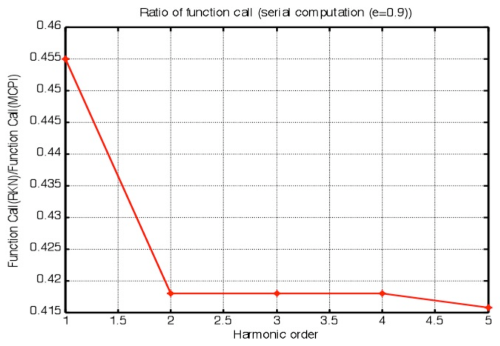
Figure IV.13: MCPI and RKN12(10) Results.



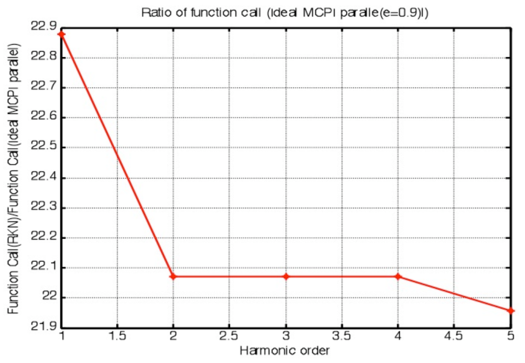
(a) Computational Time ($e = 0.9$).



(b) Speedup of MCPI ($e = 0.9$).



(c) Ratio of Function Calls in a Serial Computer ($e = 0.9$).



(d) Ratio of Function Calls in an Ideal Parallel Architecture ($e = 0.9$).

Figure IV.14: MCPI vs ODE45.

IV.G.1.f. Example 5: Integration of Perturbed Orbits with High Eccentricity (e=0.9)

For this highly eccentric orbit, the six classical orbital elements is shown in Table IV.2. For the MCPI method, the Chebyshev polynomial order is chosen as 45, with the exception of the segment passing perigee during which a polynomial of order 110 is utilized. The segment length is assumed to be 1/20 of the orbit, which has been tested in the unperturbed problem to provide sub-millimeter position accuracy.

Table IV.2: Classical Orbital Elements for High Eccentricity Orbit.

EPOCH STATE			EPOCH ELEMENTS		
r_x	1034.404	Km	a	65000	Km
r_y	-6086.687	Km	e	0.9	
r_z	-2032.917	Km	i	68	deg
v_x	3.673094	Km/sec	ω	-160	deg
v_y	3.795599	Km/sec	Ω	92	deg
v_z	-9.413550	Km/sec	M	164	deg
		PERIOD, T_p	45.811	hrs	

For the perturbed cases, we have verified that the relative energy changes for both methods are in the range of 10^{-13}. The computational time for the two methods are shown in Figure IV.14a and the comparison results are shown in Figures IV.14b-IV.14d. Figure IV.14b shows that the MCPI method achieved more than one order magnitude of speedup over RKN12(10), and this speedup is higher than the one achieved for the low eccentric case. Figure IV.14c shows that RKN12(10) calls about 40% of the number of function evaluations required by the MCPI method, whereas Figure IV.14d shows that in an ideal parallel computation environment where we can distribute the force evaluation on the $(N + 1)$ CGL nodes onto $(N + 1)$ processors, RKN12(10) calls about twenty times of the number of function evaluations required by the MCPI method.

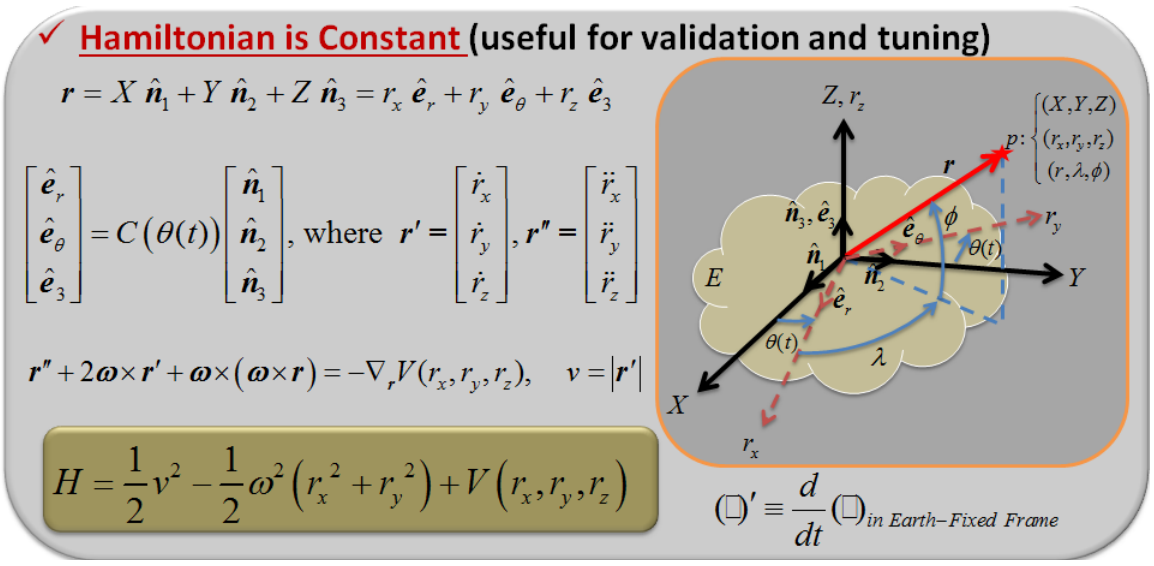
We remark that the tuning of the MCPI algorithm is ad hoc, and even without optimization, the order of magnitude speedup and high precision relative to the competing RKN12(10) algorithm suggests a strong basis for optimism. This should become even more apparent as the highly parallel architecture and adaptive tuning features are fully developed and exploited.

IV.G.2. Energy Jacobi Integral

In order to investigate the accuracy level in the numerical solution, we develop an energy integral formula (actually, the Hamiltonian, which can be proven to be constant in the absence of drag or other non-conservative forces) of the perturbed two body solution in the rotating body frame (derived quite analogously to the classical Jacobi integral of the restricted three body problem). Therefore, this rigorous "motion constant" accuracy check is carried out based on degree to which the Hamiltonian is constant rather than simply comparing one slightly incorrect numerical solution to the other. The fact that this integral exists for the most elaborate gravity models does not appear to be widely appreciated, but it is a quite powerful referee when validating orbit propagators accuracy and especially, judging relative efficiency for a given accuracy.

Consider the Earth rotation angle is defined as: $\theta(t) = \theta(t_{mid-night}) + \omega(t - t_{mid-night})$ where ω is the earth angular velocity. To develop the Jacobi Integral, we express the inertial position vector r in a rotating reference frame $F : \{\hat{\boldsymbol{e}}_r, \hat{\boldsymbol{e}}_\theta, \hat{\boldsymbol{e}}_3\}$, see Figure IV.15:

$$\boldsymbol{r} = X\hat{\boldsymbol{n}}_1 + Y\hat{\boldsymbol{n}}_2 + Z\hat{\boldsymbol{n}}_3 = r_x\hat{\boldsymbol{e}}_r + r_y\hat{\boldsymbol{e}}_\theta + r_z\hat{\boldsymbol{e}}_3 \tag{4.65}$$



✓ Hamiltonian is Constant (useful for validation and tuning)

$$\boldsymbol{r} = X\,\hat{\boldsymbol{n}}_1 + Y\,\hat{\boldsymbol{n}}_2 + Z\,\hat{\boldsymbol{n}}_3 = r_x\,\hat{\boldsymbol{e}}_r + r_y\,\hat{\boldsymbol{e}}_\theta + r_z\,\hat{\boldsymbol{e}}_3$$

$$\begin{bmatrix} \hat{\boldsymbol{e}}_r \\ \hat{\boldsymbol{e}}_\theta \\ \hat{\boldsymbol{e}}_3 \end{bmatrix} = C(\theta(t)) \begin{bmatrix} \hat{\boldsymbol{n}}_1 \\ \hat{\boldsymbol{n}}_2 \\ \hat{\boldsymbol{n}}_3 \end{bmatrix}, \text{ where } \boldsymbol{r}' = \begin{bmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \end{bmatrix}, \boldsymbol{r}'' = \begin{bmatrix} \ddot{r}_x \\ \ddot{r}_y \\ \ddot{r}_z \end{bmatrix}$$

$$\boldsymbol{r}'' + 2\boldsymbol{\omega}\times\boldsymbol{r}' + \boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{r}) = -\nabla_r V(r_x, r_y, r_z), \quad v = |\boldsymbol{r}'|$$

$$H = \frac{1}{2}v^2 - \frac{1}{2}\omega^2\left(r_x^2 + r_y^2\right) + V(r_x, r_y, r_z)$$

$$(\square)' \equiv \frac{d}{dt}(\square)_{in\ Earth\text{-}Fixed\ Frame}$$

Figure IV.15: Illustration of the Earth Rotating Frame.

The coordinate mapping

$$\begin{bmatrix} \hat{\boldsymbol{e}}_r \\ \hat{\boldsymbol{e}}_\theta \\ \hat{\boldsymbol{e}}_3 \end{bmatrix} = C(\theta(t)) \begin{bmatrix} \hat{\boldsymbol{n}}_1 \\ \hat{\boldsymbol{n}}_2 \\ \hat{\boldsymbol{n}}_3 \end{bmatrix} \tag{4.66}$$

Note that the angular velocity vector of the F frame relative to some inertial frame $\boldsymbol{\omega} = \omega\hat{\boldsymbol{e}}_3$. Let the time derivative as seen by the F frame can be labeled as:

$$\frac{^F d}{dt}\boldsymbol{x} = \boldsymbol{x}' \tag{4.67}$$

then the velocity and acceleration vectors as seen by F are given by

$$\boldsymbol{r}' = \begin{bmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \end{bmatrix}_F \qquad \boldsymbol{r}'' = \begin{bmatrix} \ddot{r}_x \\ \ddot{r}_y \\ \ddot{r}_z \end{bmatrix}_F \tag{4.68}$$

by differentiating the position vector and applying the transport theorem, the inertial velocity and acceleration vectors are expressed as:

$$\begin{aligned} \boldsymbol{v} &= \frac{{}^{N}d}{dt}\boldsymbol{r} = \frac{{}^{F}d}{dt}\boldsymbol{r} + \boldsymbol{\omega}\times\boldsymbol{r} \\ &= \boldsymbol{r}' + \boldsymbol{\omega}\times\boldsymbol{r} = \begin{bmatrix} \dot{r}_x - \omega r_y \\ \dot{r}_y + \omega r_x \\ \dot{r}_z \end{bmatrix} \end{aligned} \tag{4.69}$$

$$\begin{aligned} \boldsymbol{a} &= \frac{{}^{N}d}{dt}\boldsymbol{v} = \frac{{}^{F}d^2}{dt^2}\boldsymbol{r} + 2\boldsymbol{\omega}\times\frac{{}^{F}d}{dt}\boldsymbol{r} + \boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{r}) \\ &= \boldsymbol{r}'' + 2\boldsymbol{\omega}\times\boldsymbol{r}' + \boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{r}) = -\nabla_{\boldsymbol{r}}V(r_x,r_y,r_z) = -\begin{bmatrix} \frac{\partial V}{\partial r_x} \\ \frac{\partial V}{\partial r_y} \\ \frac{\partial V}{\partial r_z} \end{bmatrix} \end{aligned} \tag{4.70}$$

Eq. (4.70) can be written as

$$\begin{aligned} \boldsymbol{r}'' + 2\boldsymbol{\omega}\times\boldsymbol{r}' &= -\nabla_{\boldsymbol{r}}V(r_x,r_y,r_z) - \boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{r}) \\ \begin{bmatrix} \ddot{r}_x - 2\omega\dot{r}_y \\ \ddot{r}_y + 2\omega\dot{r}_x \\ \ddot{r}_z \end{bmatrix} &= -\begin{bmatrix} \frac{\partial V}{\partial r_x} - \omega^2 r_x \\ \frac{\partial V}{\partial r_y} - \omega^2 r_y \\ \frac{\partial V}{\partial r_z} \end{bmatrix} \end{aligned} \tag{4.71}$$

Assume $U(r_x,r_y,r_z) = V(r_x,r_y,r_z) - \frac{1}{2}\omega^2(r_x^2 + r_y^2)$ then Eq. (4.71) becomes

$$\boldsymbol{r}'' + 2\boldsymbol{\omega}\times\boldsymbol{r}' = \begin{bmatrix} \ddot{r}_x - 2\omega\dot{r}_y \\ \ddot{r}_y + 2\omega\dot{r}_x \\ \ddot{r}_z \end{bmatrix} = -\nabla_{\boldsymbol{r}}U(r_x,r_y,r_z) \tag{4.72}$$

By performing the vector dot product of Eq. (4.72) by $\boldsymbol{r}'$, we find the following perfect differential equation:

$$(\boldsymbol{r}'' + 2\boldsymbol{\omega}\times\boldsymbol{r}')\cdot\boldsymbol{r}' = \boldsymbol{r}''\cdot\boldsymbol{r}' = \frac{1}{2}\frac{{}^{F}d}{dt}(\boldsymbol{r}'\cdot\boldsymbol{r}') = -\nabla_{\boldsymbol{r}}U\cdot\boldsymbol{r}' = -\frac{\partial U}{\partial \boldsymbol{r}}\cdot\boldsymbol{r}' = -\frac{{}^{F}d}{dt}U \tag{4.73}$$

Integrating this equation with respect to time yields a perfect integral of the relative equations of motion:

$$v^2 = \boldsymbol{r}' \cdot \boldsymbol{r}' = -2U(r_x, r_y, r_z) - C$$

$$\text{or} \tag{4.74}$$

$$\frac{1}{2}v^2 - \frac{1}{2}\omega^2(r_x^2 + r_y^2) + V(r_x, r_y, r_z) = -C$$

where the scalar constant C is determined through the initial conditions. The constant C can be thought of as a relative energy measure. This perfect integral of the relative equations of motion is used to investigate the accuracy of a numerical integration.

As another interpretation, we use the Lagrange's equation and the Hamiltonian formula in the rotating frame to prove the same formula. Lagrange's Equation is defined by

$$L = T(\boldsymbol{q}, \dot{\boldsymbol{q}}) - V(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{4.75}$$

where T is the kinetic energy and V is the potential energy of the system. The generalized coordinate is chosen to be the position vector in the rotating frame

$$\boldsymbol{q} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}_F \qquad \dot{\boldsymbol{q}} = \begin{bmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \end{bmatrix}_F \tag{4.76}$$

The kinetic energy per unit mass is calculated by

$$\begin{aligned} T(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \frac{1}{2}({}^N\dot{\boldsymbol{r}} \cdot {}^N\dot{\boldsymbol{r}}) = \frac{1}{2}({}^F\dot{\boldsymbol{r}} + \boldsymbol{\omega} \times {}^F\boldsymbol{r}) \cdot ({}^F\dot{\boldsymbol{r}} + \boldsymbol{\omega} \times {}^F\boldsymbol{r}) \\ &= \frac{1}{2}({}^F\dot{\boldsymbol{r}} \cdot {}^F\dot{\boldsymbol{r}} + 2{}^F\dot{\boldsymbol{r}} \cdot (\boldsymbol{\omega} \times {}^F\boldsymbol{r}) + (\boldsymbol{\omega} \times {}^F\boldsymbol{r}) \cdot (\boldsymbol{\omega} \times {}^F\boldsymbol{r})) \\ &= \frac{1}{2}(\dot{\boldsymbol{q}} \cdot \dot{\boldsymbol{q}} + 2\dot{\boldsymbol{q}} \cdot (\boldsymbol{\omega} \times \boldsymbol{q}) + (\boldsymbol{\omega} \times \boldsymbol{q}) \cdot (\boldsymbol{\omega} \times \boldsymbol{q})) \\ &= \frac{1}{2}((\dot{r}_x - \omega r_y)^2 + (\dot{r}_y + \omega r_x)^2 + \dot{r}_z^2) \end{aligned} \tag{4.77}$$

The potential energy per unit mass is the gravity potential computed in the rotating frame, which is given by

$$V(\boldsymbol{q}, \dot{\boldsymbol{q}}) = V({}^F\boldsymbol{r}) = V(r_x, r_y, r_z) = V(\boldsymbol{q}) \tag{4.78}$$

The Hamiltonian is given by

$$H = \sum_{i=x,y,z} p_i \dot{q}_i - L \tag{4.79}$$

where $p_i = \frac{\partial L}{\partial \dot{q}_i}$ is the conical conjugate moment such that

$$p_x = \frac{\partial L}{\partial \dot{r}_x} = (\dot{r}_x - \omega r_y), \quad p_y = \frac{\partial L}{\partial \dot{r}_y} = (\dot{r}_y + \omega r_x), \quad p_z = \frac{\partial L}{\partial \dot{r}_z} = \dot{r}_z \tag{4.80}$$

Substituting Eq.(4.78), Eq.(4.78) and Eq. (4.80) into Eq.(4.79) , the Hamiltonian becomes

$$H = \frac{1}{2}v^2 - \frac{1}{2}\omega^2(r_x^2 + r_y^2) + V(r_x, r_y, r_z) \tag{4.81}$$

Since the Hamiltonian does not depend on time and all of the working forces are either conservative or do no work under virtual displacements. Consistent with constraints, we know [32] that the Hamiltonian is constant. This can be readily proven by formally differentiating Eq. (4.79) and substituting Hamilton's equations of motion with $H = H(q_1, q_2, \cdots, p_1, p_2, \cdots)$ as

$$\begin{aligned} \dot{q}_i &= \frac{\partial H}{\partial p_i} \\ \dot{p}_i &= -\frac{\partial H}{\partial q_i} \end{aligned} \tag{4.82}$$

H = Eq. (4.81) = *constant* is exactly the same as the Jacobi integral given in Eq.(4.75), where $H = -C$.

IV.G.3. MCPI vs Runge-Kutta 4-5 with EGM2008 Gravity Model in C++ Environment

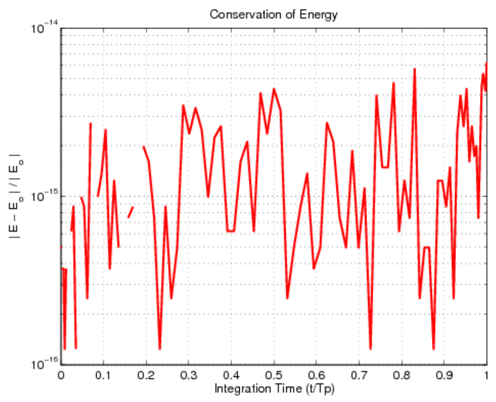
In this section we compare the performance of the MCPI algorithm with the classical Runge-Kutta (RK45) method. Computations for the final two examples were performed using a PC with the following settings and environment

- Intel Core i7 3610QM Processor 2.3GHz
- 1500GB 7200rpm Hard Drive
- Nvidia GTX 660M
- Windows 7 Home Premium 64-bit
- Microsoft Visual Studio 2008

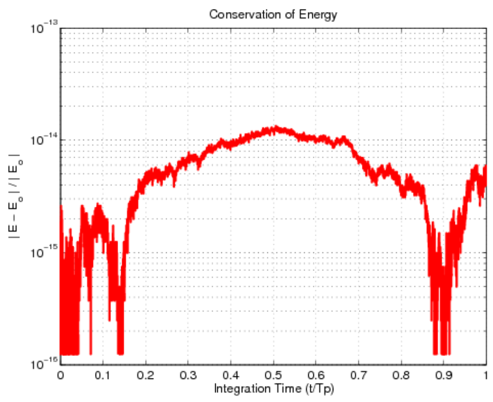
The following results represent time (computer execution speed) comparison between MCPI and RK45. The two propagators are used to integrate perturbed satellite motion using EGM2008 spherical harmonic 100x100. Both propagators are implemented in a C++ environment and a speed test is performed to investigate which code is faster. In order to ensure that the two propagators produce the required accuracy level in the solution, we investigate the exact energy integral (the Hamiltonian) of the two solutions in the rotating body frame. This rigorous motion constant accuracy check is carried out based on the degree to which the Hamiltonian is constant rather than simply comparing one slightly incorrect numerical solution to the other. The fact that this integral exists for the most elaborate gravity models does not appear to be widely appreciated, but it is quite a powerful "referee" when validating orbit propagator accuracy, and especially when judging the relative efficiency for a given accuracy. In this case, we have tuned both methods to maintain nearly the same level of accuracy for each relative "speed-up" test case. As is evident, we are maintaining over a 14 digit accuracy in both solutions. This is much more than needed in most geocentric orbits of interest, and requiring less precision will likely affect to some degree the relative efficiency.

For this example the number of nodes used in the MCPI calculation is $N = 100$, and for the RK45 method the step size is tuned until the same level of accuracy is achieved. In order to perform a fair comparison the total energy (kinetic + potential) of each system is calculated and the accuracy is plotted as a function of harmonic order. The results are for MCPI and RK45 are shown in Figures IV.18a and IV.18b respectively. It is evident that the accuracy of both methods falls below 1×10^{-14}.

The respective computation times for harmonic order 1 through to 100 are shown in Figure IV.17a. For this single orbit propagation example it is clear that the computation time required for the MCPI computation is over an order of magnitude faster than the RK45 method. The ratio of the computation times for RK45 and



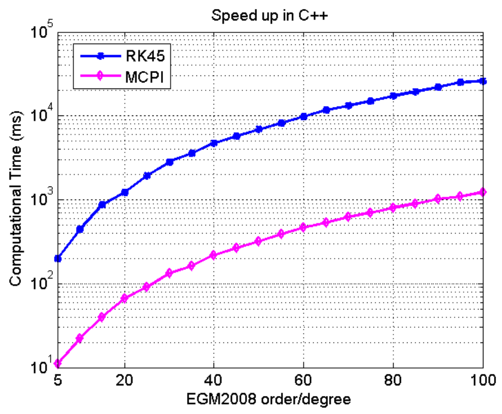
(a) MCPI Accuracy.



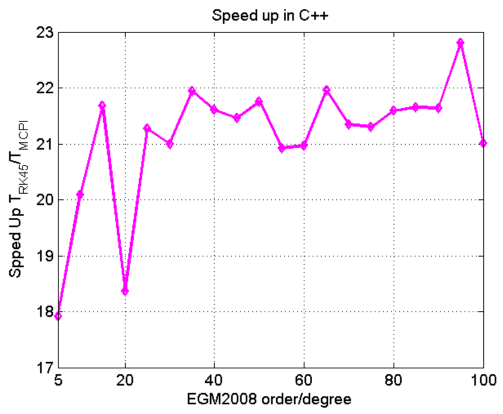
(b) RK45 Accuracy.

Figure IV.16: Accuracy Check by Hamiltonian for MCPI and RK45.

MCPI is shown in Figure IV.17b for increasing harmonic order.



(a) MCPI and RK45 Computation Times.



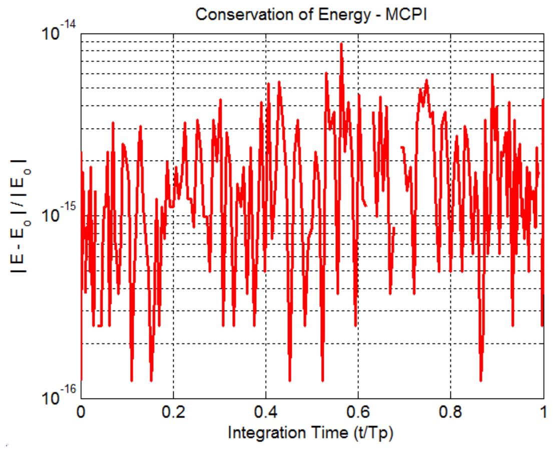
(b) Ratio of Computation Time.

Figure IV.17: Single Orbit Propagation: MCPI vs RK45 Computation Times.

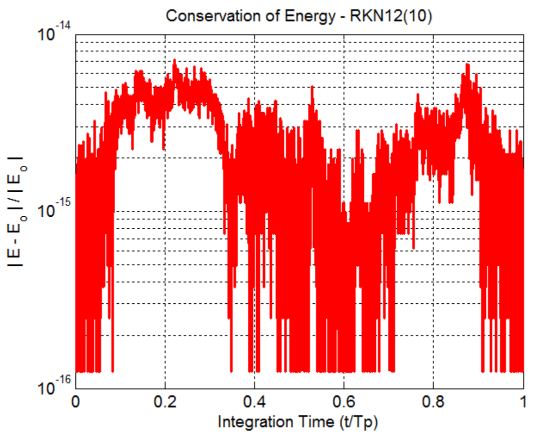
The MCPI method significantly out-performed the RK45 method. The computation time required to produce an orbital propagation solution with the same accuracy is considerably less for the MCPI method compared with the RK45 method. The main difference in these algorithms is that MCPI propagates the entire trajectory/orbit at each iterative step whereas RK45 only propagates one point/measurement during each iteration. Further more, MCPI is extremely well suited for parallelization and thus one can expect even greater speed ups once the algorithm is implemented in this form.

IV.G.4. MCPI vs Runge-Kutta 12-10 with EGM2008 Gravity Model in C++ Environment

In this section we compare the performance of the MCPI algorithm with the Runge-Kutta (RK12(10)) method. The results that follow represent time (speed) comparison between MCPI and RK12(10). The two propagators are used to integrate perturbed satellite motion (using EGM2008 spherical harmonic 50×50). Both propagators are implemented in C++ environment. A speed test is performed to investigate the faster code. Similarly, in order to ensure that the two propagators produce the accuracy level in the solution, we investigate the energy integral (the Hamiltonian) of the two solutions in the rotating body frame. As is evident, we are maintaining over 15 digit accuracy in both solutions, much more than needed in most geocentric orbits of interest, and requiring less precision will likely affect to some degree the relative efficiency. Example 1 is for a single orbit case and Example 2 is for a multi orbit case.



(a) MCPI Accuracy.



(b) RK12(10) Accuracy.

Figure IV.18: Accuracy Check by Hamiltonian for MCPI and RK12(10).

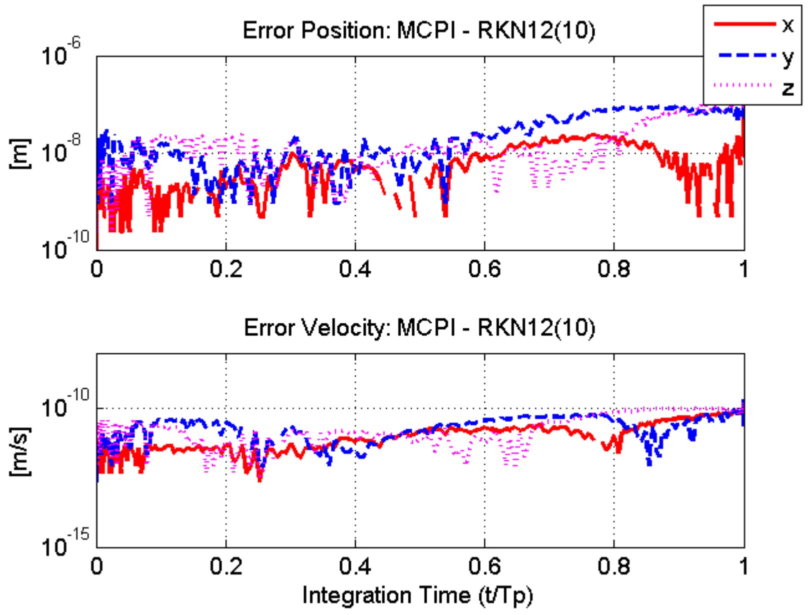


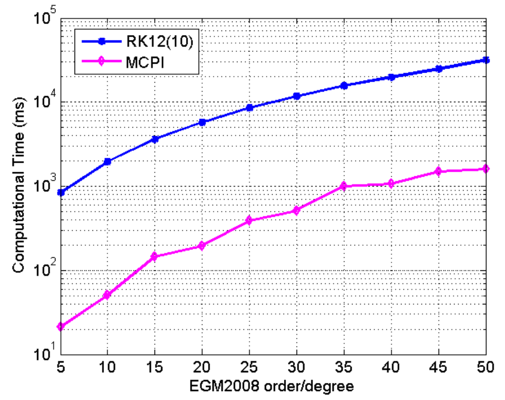
Figure IV.19: Error in Position and Velocity between MCPI - RK12(10).

IV.G.4.a. Example 1: Single Orbit Propagation

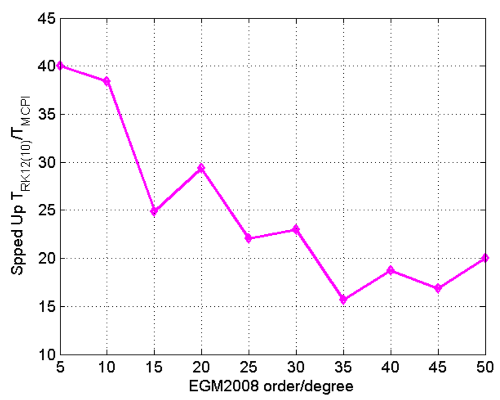
We first investigate the simulation speed up of the MCPI versus the RK12(10) for various spherical harmonic order (EGM2008), from 5×5 to 50×50. The energy check tolerance is chosen to be $1e^{-15}$, as seen in Figure IV.20 and Figure IV.19. This tolerance also maintains $1e^{-8}$ [m] error in position and $1e^{-11}$ [m/s] error in velocity. The simulation results are presented in Figure IV.20. Note that we increase the required polynomial order (number of nodes) for the MCPI as we increase the spherical harmonic order. This implies that, in order to maintain the same integration accuracy ($< 1e^{-15}$), we are required to include more nodes to fit the trajectory and capture the extra wrinkles/perturbations added to the classical two-body trajectory. Figure IV.21 shows that significant figures for the MCPI (EGM2008 50×50) energy check maximum error versus the MCPI order and integration segment. The simulation speedup curves show that, in the serial machine, the MCPI propagator (in C++ environment) is about $\sim 40\times$ faster than RK12(10) for 5×5 perturbed motion. While this speed factor decreases gradually to about $20\times$ for 50×50 gravity perturbed motion.

IV.G.4.b. Example 2: Multi Orbit Propagation

Here we propagate the same initial conditions for 15 orbits using MCPI and RK12(10) for 50×50 EGM2008 perturbed two-body problem. The energy check tolerance is chosen to be $1e^{-15}$, as seen in Figure IV.22. The simulation results are presented in Figure IV.23. The simulation speedup curves show that, in the serial machine, the MCPI propagator (in C++ environment) is about one order of

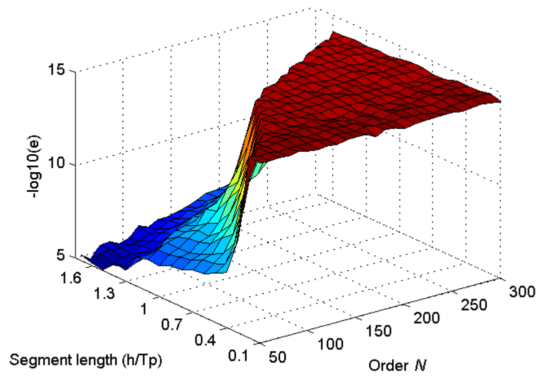


(a) MCPI and RK12(10) Computation Times.



(b) Ratio of Computation Time: RK12(10) to MCPI.

Figure IV.20: Single Orbit Propagation: MCPI vs RK12(10) Computation Times.



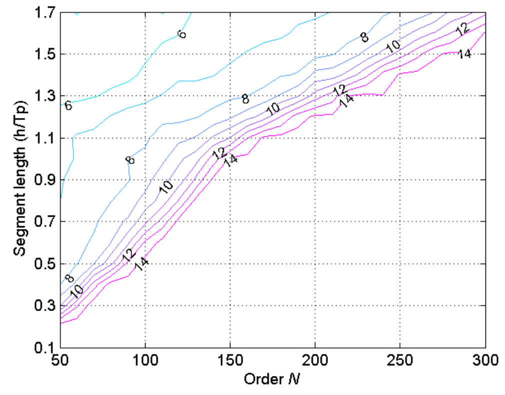
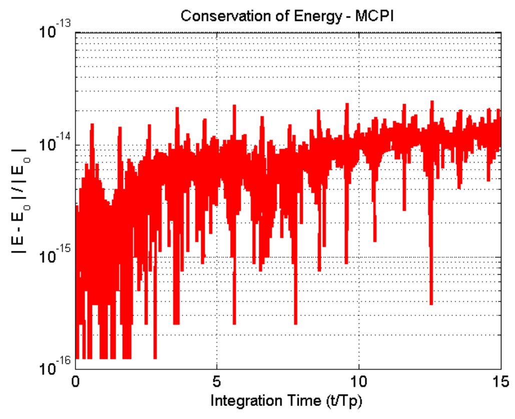
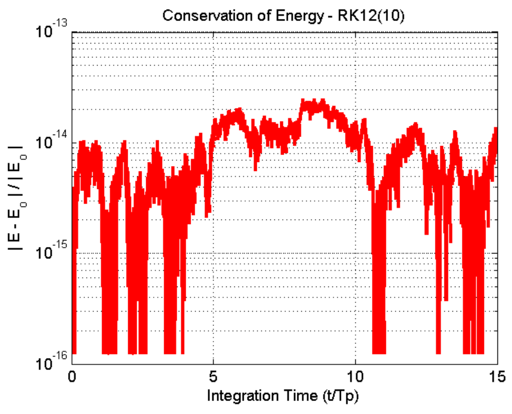


Figure IV.21: Significant Figures for MCPI at EGM2008 50×50.

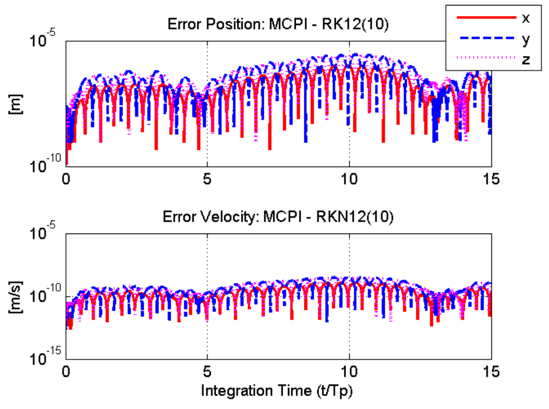
magnitude faster than RK12(10) for 50×50 gravity perturbed motion.



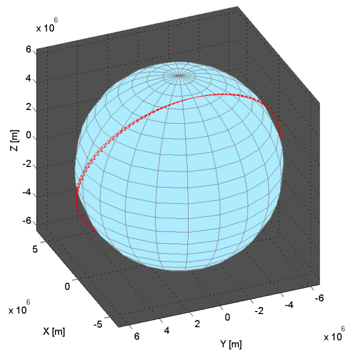
(a) MCPI Accuracy.



(b) RK12(10) Accuracy.



(c) Error in Position and Velocity.

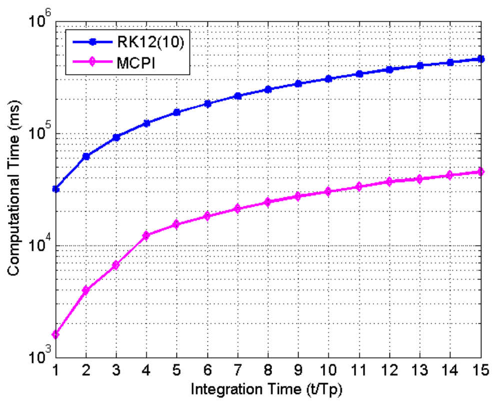


(d) Multi-Orbit in (ECEF).

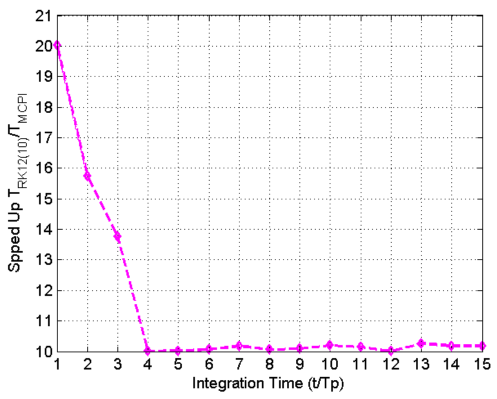
Figure IV.22: Accuracy Check by Hamiltonian for multi orbits.

IV.G.5. Trajectory Propagations Using FEM Versus Spherical Harmonic Gravity Model

In this section we compare the performance of the MCPI algorithm with the classical Runge-Kutta (RK45 or *ODE45*) method. Computations for the final two examples were performed using a serial computer with the following configuration settings and environment



(a) MCPI and RK12(10) Computation Times.



(b) Ratio of Computation Time: RK12(10) to MCPI.

Figure IV.23: Multi Orbit Propagation: MCPI vs RK12(10) Computation Times.

- Intel Core i7 3610QM Processor 2.3GHz
- 1500GB 7200rpm Hard Drive
- Nvidia GTX 660M
- Windows 7 Home Premium 64-bit
- MATLAB R2010b
- Microsoft Visual Studio 2008

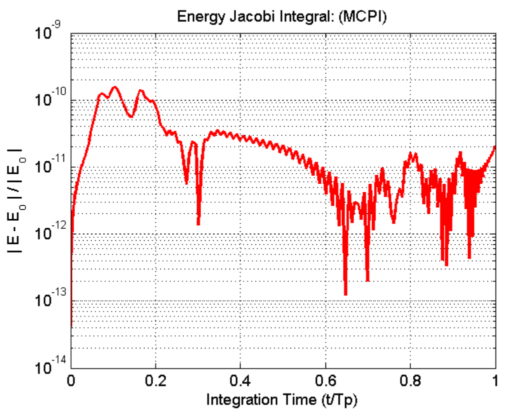
The gravity force EGM2008 200×200 is implemented in Matlab and C/C++ to calculate the acceleration using both 1) conventional spherical harmonics (SH) gravity fields, and 2) the interpolated finite element gravity model (FEM). Figure IV.24 shows the Hamiltonian check of three different different propagators using both spherical harmonic gravity field EGM2008 200×200 and the interpolated finite element model FEM. The subfigures IV.24a through IV.24f demonstrate:

- Test #1: Figure IV.24a shows the Hamiltonian check for the solution propagated by Modified Chebyshev Picard Iteration (MCPI) using the conventional spherical harmonics (SH) gravity fields, EGM2008 200×200.
- Test #2: Figure IV.24b shows the Hamiltonian check for the solution propagated by Modified Chebyshev Picard Iteration (MCPI) using the interpolated finite element gravity model (FEM), EGM2008 200×200.
- Test #3: Figure IV.24c shows the Hamiltonian check for the solution propagated by Adaptive Chebyshev Picard Iteration (ACPI) using the conventional spherical harmonics (SH) gravity fields, EGM2008 200×200.
- Test #4: Figure IV.24d shows the Hamiltonian check for the solution propagated by Adaptive Chebyshev Picard Iteration (ACPI) using the interpolated finite element gravity model (FEM), EGM2008 200×200.
- Test #5: Figure IV.24e shows the Hamiltonian check for the solution propagated by Runge-Kutta (RK45 or *ODE45*) method using the conventional spherical harmonics (SH) gravity fields, EGM2008 200×200.
- Test #6: Figure IV.24f shows the Hamiltonian check for the solution propagated by Runge-Kutta (RK45 or *ODE45*) method using the interpolated finite element gravity model (FEM), EGM2008 200×200.

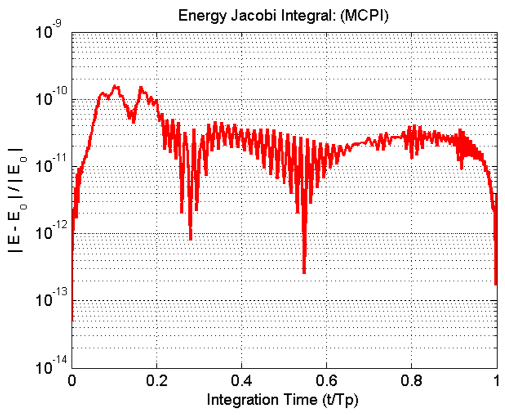
The propagated orbits are displayed in Figure IV.25, which shows perturbed propagations for 20 orbits using EGM2008 200×200; presented in (a) Earth-Centered Earth-Fixed (ECEF) coordinates, as shown in IV.25a, and (b) Earth Rotating coordinates, as shown in IV.25b.

The computation time CPU cost is shown in Figure IV.26. All tests are performed in MATLAB R2010b. The pre-computed J_5 perturbed two body problem is used to warm-start the iterations. For the spherical harmonic computation, it is obvious that both MCPI and ACPI are about an order of magnitude faster than the ODE45 in serial processor. However, this is not the case when the finite element model FEM is used, as the ODE45 computation speed is better than both MCPI and ACPI speed. Computation routines in MATLAB are stored in the optimized cache memory, which allows fast processing if the same routine is called in the runtime. In other words, the MATLAB environment is designed to make ODE45 smarter for sequential operations. This speed optimization does not exits if the same code is run in dry C/C++ or Fortran. Moreover, all methods show that using the interpolated finite element gravity model (FEM) in the force function improves the overall speed up of the integrator; e.g. for the ODE45, the FEM speed up reaches four order of magnitude compared with SH 200×200. It is evident that there is a slight speed up of ACPI versus MCPI for both cases: with SH and with FEM. This speed can be further enhanced by adapting the choice of the polynomial order in the L.H.S and R.H.S in Eq. 4.33.

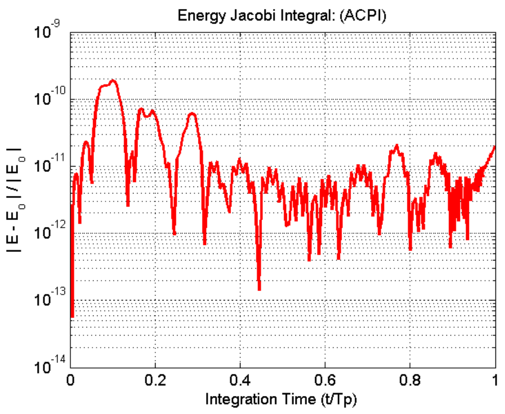
To avoid the MATLAB cache memory optimization, both C++ FEM and SH of the EGM2008 200×200 is ported to both MCPI and RK12(10) propagators in serial processor. The computation time CPU cost is shown in Figure IV.27. The pre-computed J_5 perturbed two body problem is used to warm-start the iterations. For the spherical harmonic computation, it is obvious that MCPI is over an order of magnitude faster than the RK12(10) in serial processor. The interpolated finite element gravity model (FEM) in the force function improves the overall speed up of the both integrators. This speed can be further enhanced by optimizing the FEM C++ code.



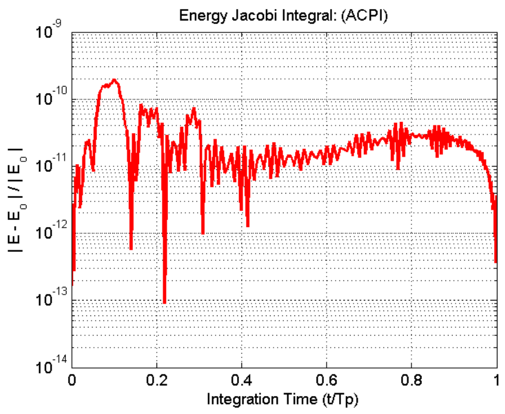
(a) MCPI with SH.



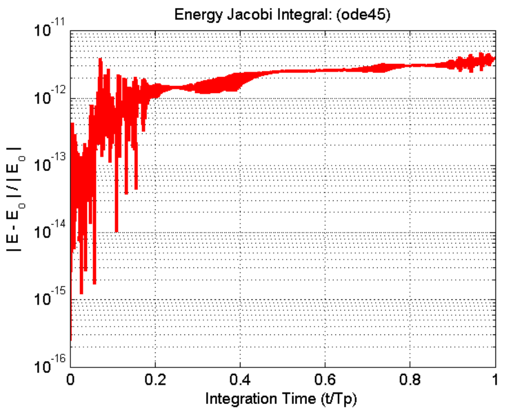
(b) MCPI with FEM.



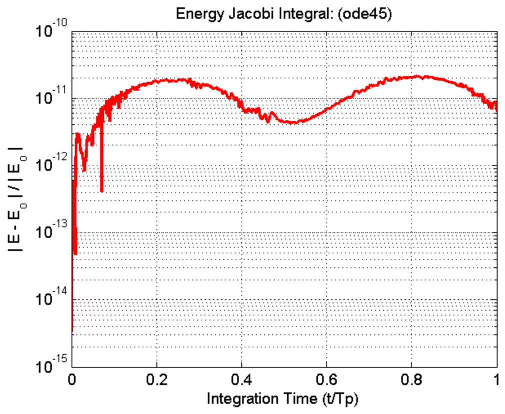
(c) ACPI with SH.



(d) ACPI with FEM.

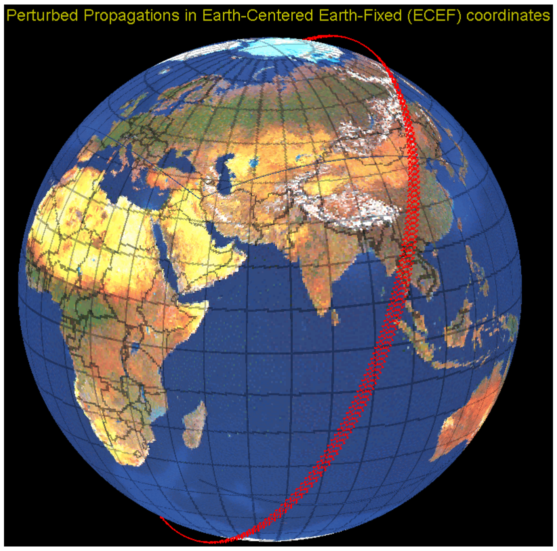


(e) ODE45 with SH.

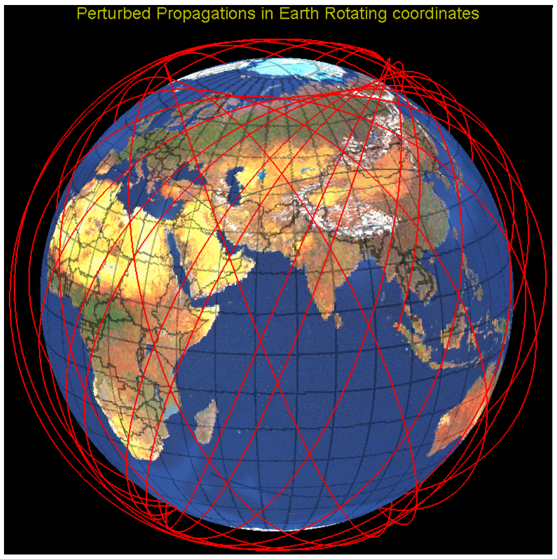


(f) ODE45 with FEM.

Figure IV.24: Hamiltonian (or Energy Jacobi) Energy Check.


Perturbed Propagations in Earth-Centered Earth-Fixed (ECEF) coordinates

(a) Earth-Centered Earth-Fixed (ECEF).



(b) Earth Rotating Coordinates.

Figure IV.25: Perturbed Propagations for 20 Orbits using EGM2008 200 × 200; Presented in (a) Earth-Centered Earth-Fixed (ECEF) Coordinates, (b) Earth Rotating Coordinates.

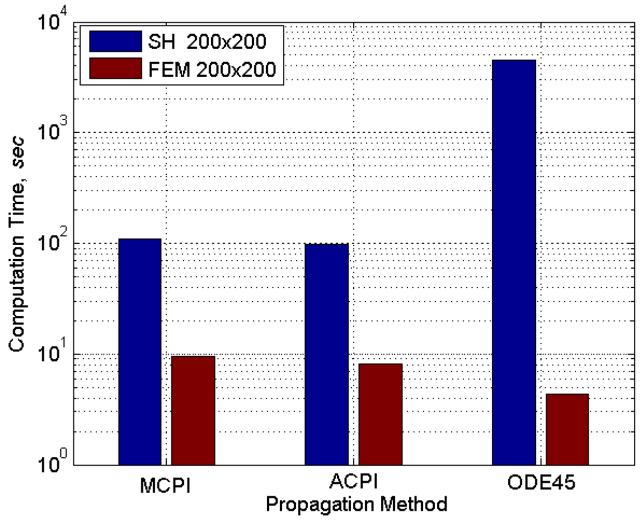


Figure IV.26: Computation Time Cost for Test #1 through Test #6.

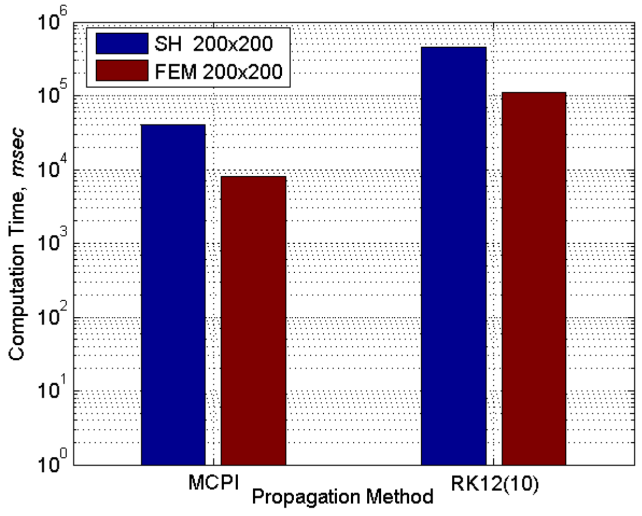


Figure IV.27: Computation Time cost for MCPI and RK12(10).

CHAPTER V

SUMMARY AND CONCLUDING REMARKS

We have summarized classical and recent developments from approximation theory, with emphasis on representing given complicated functions by orthogonal polynomials in one, two, and higher dimensions. We have shown that arranging the regression matrix to be Kronecker factorable allows array algebra identities to generate the multidimensional orthogonal least square operators directly from the corresponding one dimensional operators. We showed that a four order of magnitude speedup in the computation time to obtain state-of-the-are gravitational acceleration is possible. As a consequence, a new generation of very efficient algorithms results for orbit integration, regardless of the method used to propagate the orbit. A number of nonlinear test functions of one and two variables were introduced and used to show that machine precision approximation results are routinely obtained. It was further shown that integration of these orthogonal approximations led to especially attractive accuracy increases, whereby the oscillatory zero mean least square approximation error amplitudes are smoothed and reduced by one order of magnitude through trajectory integration. This has immediate implications and explains in part the impressive results obtained in the Chebyshev-Picard recently presented by [22], in solving the boundary value problems of celestial mechanics. Based on recent contributions from satellite geodesy, we are now in a position to confidently compute gravity globally with 9 or more significant figures at all orbit altitudes. The challenge is that the classical spherical harmonic expansion and analogous global models require $> 10^5$ terms in a series to compute > 9 digit converged local acceleration with a single global expansion. Therefore, it is not attractive to utilize high order global models to compute local gravity for the purpose of efficient and accurate trajectory computation. We showed in this dissertation that it is feasible to develop adaptive finite element approximation methods that inherently answer a heretofore unanswered question: For FEM gravitational approximation methods, how can we determine approximations that automatically adapt as a function of radial distance, so that (for example) the number of terms in the approximation is automatically minimized to maintain a prescribed accuracy? The results obtained are illustrated with a number of tests cases that show several orders of magnitude reduction in computation time in comparison to the correspondingly accurate spherical harmonic series. The implications are substantial for efficient and accurate propagation of space object catalogs, efficient Monte Carlo studies, and analogous operations where orbits must be iteratively computed and propagated over long time intervals.

Both the Modified Chebyshev Picard Iteration (MCPI) and the Adaptive Chebyshev Picard Iteration (ACPI) methods are derived for both the first and second order cases. Flow diagrams of the iteration routines are included and we also present a vector-matrix representation of the algorithms. The algorithms are executed for a number of first and second order example problems. The results are discussed and compared with other differential equations solvers such as Runge-Kutta. The comparison leads to the anticipated result that MCPI is a far superior method with regard to accuracy (one order of magnitude) and computation time (two orders of magnitude). In addition, the parallel nature of the algorithm allows for further enhancements as refinements with regard to use of parallel computing generally and Graphics Processing Units in particular. The applications studied in this dissertation show extremely promising results and illuminate a clear pathway along which significant future develops can lead to great advancements that are not limited to the differential equations of astrodynamics. These exciting new derivations leave the door wide open for the realization of new developments and applications in dynamical systems and control systems.

REFERENCES

[1] Kessler, D. J. and Cour-Palais, B. G., "Collision Frequency of Artificial Satellites: The Creation of a Debris Belt," *J. Geophys. Res.*, Vol. 83, No. A6, 1978, pp. 2637–2646.

[2] Kessler, D. J., Stansbery, E. G., Zhang, J., Matney, M. J., Eichler, P., Reynolds, R. C., and Anz-Meador, P. D., "A Computer-Based Orbital Debris Environment Model for Spacecraft Design and Observation in Low Earth Orbit," Tech. Rep. NASA-TM-104825, NAS 1.15:104825, S-822, NASA-Johnson Space Center, Nov. 1996.

[3] Anselmo, L., Cordelli, A., Jehn, R., Pardini, C., and Rossi, A., "New Results of the Upgraded SDM Space Debris Modeling Software," *Space Debris*, edited by J. Bendisch, Vol. 100, 1999, pp. 187–203.

[4] Kessler, D. J. and Anz-Meador, P. D., "Critical Number of Spacecraft in Low Earth Orbit: Using Satellite Fragmentation Data to Evaluate the Stability of the Orbital Debris Environment," *Space Debris*, edited by H. Sawaya-Lacoste, Vol. 473, 2001, pp. 265–272.

[5] Johnson, N. L., Krisko, P. H., Liou, J. C., and Anz-Meador, P. D., "NASA's New Breakup Model of Evolve 4.0," *Advances in Space Research*, Vol. 28, No. 9, 2001, pp. 1377 – 1384.

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

[7] Liou, J. C., Matney, M. J., Anz-Meador, P. D., Kessler, D., Jansen, M., and Theall, J. R., “The New NASA Orbital Debris Engineering Model ORDEM2000,” *NASA STI/Recon Technical Report N*, Vol. 2, May 2002, pp. 51086.

[8] Liou, J. C., Hall, D. T., Krisko, P. H., and Opiela, J. N., “LEGEND - a Three-dimensional LEO-to-GEO Debris Evolutionary Model,” *Advances in Space Research*, Vol. 34, Jan. 2004, pp. 981–986.

[9] Oswald, M., Wegener, P., Stabroth, S., Wiedemann, C., Rosebrock, J., Martin, C., Klinkrad, H., and Vörsmann, P., “The Master 2005 Model,” *4th European Conference on Space Debris*, edited by D. Danesy, Vol. 587 of *ESA Special Publication*, Aug. 2005, p. 235.

[10] Liou, J. C., “Collision Activities in the Future Orbital Debris Environment,” *Advances in Space Research*, Vol. 38, No. 9, 2006, pp. 2102 – 2106.

[11] Ailor, W., “Collision Avoidance and Improving Space Surveillance,” *Astropolitics*, Vol. 2, No. 2, 2004, pp. 107–120.

[12] Gaposchkin, E., von Braun, C., and Sharma, J., “Space-based Space Surveillance with the Space-Based Visible,” *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 1, 2000, pp. 148–152.

[13] Naka, F., Canavan, G., Clinton, R., Judd, O., and Pensa, A., "Space Surveillance, Asteroids and Comets, and Space Debris. Volume 1: Space Surveillance," Tech. Rep. SAB-TR-96-04, United States Air Force Scientific Advisory Board - DTIC Document, Pentagon, Washington, DC, June 1997.

[14] Wright, D., "Space Debris," *Physics Today*, Vol. 60, 2007, pp. 35–40.

[15] Sharma, J., Stokes, G. H., von Braun, C., Zollinger, G., and Wiseman, A. J., "Toward Operational Space-based Space Surveillance," *Lincoln Laboratory Journal*, Vol. 13, No. 2, 2002, pp. 309–334.

[16] Sergei, N., de Vries, W. H., Phillion, D., and H. K. Springer, L., "High-Fidelity Kessler Syndrome Simulations," private correspondence with J Junkins, 2010.

[17] Bai, X., *Modified Chebyshev-Picard Iteration Methods for Solution of Initial Value and Boundary Value Problems.*, Ph.D. dissertation, Texas A&M Univ, College Station, TX, 2010.

[18] Feagin, T., *The Numerical Solution of Two Point Boundary Value Problems Using Chebyshev Series*, Ph.D. dissertation, The Universtiy of Texas at Austin, Austin, TX, 1973.

[19] Feagin, T. and Nacozy, P., "Matrix Formulation of the Picard Method for Parallel Computation," *Celestial Mechanics and Dynamical Astronomy*, Vol. 29, No. 2, Feb. 1983, pp. 107–115.

[20] Shaver, J., *Formulation and Evaluation of Parallel Algorithms for the Orbit Determination Problem*, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 1980.

[21] Clenshaw, C. W. and Norton, H. J., "The Solution of Nonlinear Ordinary Differential Equations in Chebyshev Series," *The Computer Journal*, Vol. 6, No. 1, 1963, pp. 88–92.

[22] Bai, X. and Junkins, J. L., "Modified Chebyshev-Picard Iteration Methods for Solution of Initial Value Problems," *Kyle T. Alfriend Astrodynamics Symposium*, No. AAS 10-322, Monterey, CA., May 2010.

[23] Arora, N. and Russell, R., "Fast, Efficient, and Adaptive Interpolation of the Geopotential," *AAS/AIAA Astrodynamics Specialist Conference*, Vol. AAS 11-501, Girdwood, AK, 2011.

[24] Junkins, J., Miller, G., and Jancaitis, J., "Weighting Function Approach to Modeling of Irregular Surface," *Journal of Geophysical Research*, Vol. 78, No. 11, 1973, pp. 1794–1803.

[25] Junkins, J., "Investigation of Finite-Element Representations of the Geopotential," *AIAA Journal*, Vol. 14, No. 6, 1976, pp. 803–808.

[26] Junkins, J. and Jancaitis, J., "Modeling in n-Dimensions Using a Weighting Function Approach," *Journal of Geophysical Research*, Vol. 79, No. 23, 1974, pp. 3361–3366.

[27] Singla, P. and Junkins, J. L., *Multi-resolution Methods for Modeling and Control of Dynamical Systems*, CRC Press, 2009.

[28] Mason, J. and Handscomb, D., *Chebyshev Polynomials*, Chapman and Hall/CRC, 2003.

[29] Chebyshev, P. L., "Thorie De Mcanismes Connus Sous Le Nom De Paralllogrammes," *Mèmoires des Savants ètrangers prèsentès à l'Acadèmie de Saint-Pètersbourg*, Vol. 7, 1857, pp. 539–586.

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

[31] Snay, R. A., "Applicability of Array Algebra," *Rev. Geophys.*, Vol. 16, No. 3, 1978, pp. 459–464.

[32] Schaub, H. and Junkins, J. L., *Analytical Mechanics of Space Systems*, Reston, VA, AIAA Education Series, 2nd ed., 2009.

[33] Battin, R., *An Introduction to the Mathematics and Methods of Astrodynamics*, Reston, VA: American Institute of Aeronautics and Astronautics, Inc, revised ed., 1999.

[34] GGM03, "Gravity Recovery and Climate Experiment (GRACE)," *http://www.csr.utexas.edu/grace/gravity/*, Aug 16, 2011.

[35] Russell, R. P., "GRACE Spherical Harmonic Coefficients," Private Communication with J Junkins, 2012.

[36] EGM2008, "Earth Gravitational Model 2008 (EGM2008)," *http://Earth-info.nga.mil/GandG/wgs84/gravitymod/egm2008/*, May 06, 2013.

[37] Pines, S., "Uniform Representation of the Gravitational Potential and its Derivatives," *AIAA Journal*, Vol. 11, 1973, pp. 15081511.

[38] Casotto, S. and Fantino, E., "Evaluation of Methods for Spherical Harmonic Synthesis of the Gravitational Potential and its Gradients," *Advances in Space Research*, Vol. 40, 2007, pp. 69–75.

[39] Lundberg, J. B. and Schutz, B. E., "Recursion Formulas of Legendre Functions for Use with Nonsingular Geopotential Models," *Journal of Guidance Control Dynamics*, Vol. 11, Feb. 1988, pp. 31–38.

[40] Tapley, B., Ries, J., Bettadpur, S., Chambers, D., Cheng, M., Condi, F., and Poole, S., "The GGM03 Mean Earth Gravity Model from GRACE," *AGU Fall Meeting Abstracts*, Dec. 2007, pp. A3.

[41] Fahroo, F. and Ross, I. M., "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, Jan.-Feb. 2002, pp. 160–166.

[42] Gong, Q., Fahroo, F., and Ross, I. M., "Spectral Algorithm for Pseudospectral Methods in Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 3, 2008, pp. 460–471.

[43] Fahroo, F. and Ross, I. M., "A Spectral Patching Method for Direct Trajectory Optimization," *Journal of the Astronautical Sciences*, Vol. 48, No. 2/3, 2000, pp. 269–286.

[44] Junkins, J. L. and Engels, R. C., "Local Representation of the Geopotential by Weighted Orthonormal Polynomials," *Journal of Guidance, Control, and Dynamics*, Vol. 3, No. 1, 1980, pp. 55–61.

[45] Beylkin, G. and Cramer, R., "Toward Multiresolution Estimation and Efficient Representation of Gravitational Fields," *Celestial Mechanics and Dynamical Astronomy*, Vol. 84, Sept. 2002, pp. 87–104.

[46] Lekien, F. and Marsden, J., "Tricubic Interpolation in Three Dimensions," *Journal of Numerical Methods and Engineering*, Vol. 63, 2005, pp. 455–471.

[47] Colombi, A., Hirani, A. N., and Villac, B. F., "Adaptive Gravitational Force Representation for Fast Trajectory Propagation Near Small Bodies," *Journal of Guidance Control and Dynamics*, Vol. 31, 2008, pp. 1041–1051.

[48] Hujsak, R. S., "Gravity Acceleration Approximation Functions," *Advances in the Astronautical Sciences*, Vol. 93, 1996, pp. 335–349.

[49] Oltrogge, D., "AstroHD: Astrodynamics Modeling With a Distinctly Digital Flavor," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Honolulu, HI, August 18-21 2008.

[50] Van de Craats, J., "On the Region of Convergence of Picard's Iteration," *ZAMM-Journal of Applied Mathematics and Mechanics*, Vol. 52, 1971, pp. 487–491.

[51] Agarwal, R. P., "Nonlinear Two–Point Boundary Value Problems," *Indian Journal of Pure and Applied Mathematics*, Vol. 4, 1973, pp. 757–769.

[52] Coles, W. J. and Sherman, T. L., "Convergence of Successive Approximations for Nonlinear Two-Point Boundary Value Problems," *SIAM Journal on Applied Mathematics*, Vol. 15, No. 2, Mar. 1967, pp. 426–433.

[53] Bailey, P. B., "On the Interval of Convergence of Picard's Iteration," *ZAMM - Journal of Applied Mathematics and Mechanics*, Vol. 48, No. 2, 1968, pp. 127–128.

[54] Bailey, P., Shampine, L. F., and Waltman, P., "Existence and Uniqueness of Solutions of the Second Order Boundary Value Problem," *Bulletin of the American Mathematical Society*, Vol. 72, No. 1, 1966, pp. 96–98.

[55] Van de Craats, J., "On the Region of Convergence of Picard's Iteration," *ZAMM - J of Applied Math and Mech / Zeitschrift für Angewandte Mathematik und Mechanik*, Vol. 62, 1972, pp. 487–491.

[56] Coddington, E. A. and Levinson, N., *Theory of Ordinary Differential Equations*, New York, McGraw-Hill, 1955.

[57] Lindelöf, E., "Sur l'Application de la Methode des Approximations Successives aux Equations Dif-ferentielles Ordinaires du Premier Ordre," *Comptes rendus*

hebdomadaires des seances de l Academie des sciences, Vol. 114, 1894, pp. 454–457.

[58] Parker, G. E. and Sochacki, J. S., “Implementing the Picard Iteration,” *Neural, Parallel & Scientific Computations*, Vol. 4, No. 1, 1996, pp. 97–112.

[59] Vlassenbroeck, J. and Dooren, R. V., “A Chebyshev Technique for Solving Nonlinear Optimal Control Problems,” *IEEE Transactions on Automatic Control*, Vol. 33, No. 4, Apr. 1988, pp. 333–340.

[60] Fukushima, T., “Vector Integration of Dynamical Motions by the Picard-Chebyshev Method,” *The Astronomical Journal*, Vol. 113, No. 6, Jun. 1997, pp. 2325–2328.

[61] Fukushima, T., “Picard Iteration Method, Chebyshev Polynomial Approximation, and Global Numerical Integration of Dynamical Motions,” *The Astronomical Journal*, Vol. 113, No. 5, May. 1997, pp. 1909–1914.

[62] Bai, X. and Junkins, J. L., “Solving Initial Value Problems by the Picard-Chebyshev Method with NVIDIA GPUS,” *20th Spaceflight Mechanics Meeting*, No. AAS 10-197, San Diego, CA, 2010.

[63] Butcher, J. C., “Implicit Runge-Kutta Processes,” *Math. Comp.*, Vol. 18, 1964, pp. 50–64.

[64] Burrage, K., *Parallel and Sequential Methods for Ordinary Differential Equations*, Monographs on Numerical Analysis, Clarendon Press, 1995.

[65] Butcher, J., *Numerical Methods for Ordinary Differential Equations*, Wiley, 2008.

[66] Hairer, E., Nørsett, S., and Wanner, G., *Solving Ordinary Differential Equations I: Nonstiff Problems*, Solving Ordinary Differential Equations, Springer, 1993.

[67] Hairer, E., Nrsett, S., and Wanner, G., *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Lecture Notes in Economic and Mathematical Systems, Springer, 1993.

[68] Iserles, A., *A First Course in the Numerical Analysis of Differential Equations*, Cambridge texts in Applied Mathematics, Cambridge University Press, 1996.

[69] Crassidis, John L.; Junkins, J. L., *Optimal Estimation of Dynamic Systems*, New York, NY: CRC Press - Taylor and Francis., 2011.

[70] Lewis, F. L. and Syrmos, V. L., *Optimal Control*, New York, NY: Wiley-Interscience, 1995.

[71] Russell, R. and Arora, N., "Global Point Mascon Models for Simple, Accurate, and Parallel Geopotential Computation," *AAS/AIAA Space Flight Mechanics Meeting*, No. AAS 11-158, 2011.

[72] Leopardi, P., "A Partition of the Unit Sphere into Regions of Equal Area and Small Diameter," *Electronic Transactions on Numerical Analysis*, Vol. 25, 2006, pp. 309–327.

[73] Bai, X. and Junkins, J. L., "Modified Chebyshev-Picard Iteration Methods for Orbit Propagation," *Journal of the Astronautical Sciences*, Vol. 58, Oct.-Dec. 2011, pp. 583–613.

[74] Scraton, R. E., "The Solution of Linear Differential Equations in Chebyshev Series," *The Computer Journal*, Vol. 8, No. 1, 1965, pp. 57–61.

[75] Wright, K., "Chebyshev Collocation Methods for Ordinary Differential Equations," *The Computer Journal*, Vol. 6, No. 4, 1964, pp. 358–365.

[76] Norton, H. J., "The Iterative Solution of Non-linear Ordinary Differential Equations in Chebyshev Series," *The Computer Journal*, Vol. 7, No. 2, 1964, pp. 76–85.

[77] Schaub, H. and Junkins, J. L., *Analytical Mechanics of Space Systems, 2nd ed*, AIAA Education Series, Reston, VA, 2011.

APPENDIX A

CHEBYSHEV POLYNOMIALS

Chebyshev polynomials are a set of orthogonal polynomials developed by the Russian mathematician Pafnuty Lvovich Chebyshev in 1857 [29, 30]. There are two kinds of Chebyshev polynomials. The k^{th} Chebyshev polynomials of the first kind usually are denoted by T_k and the k^{th} Chebyshev polynomials of the second kind usually are denoted by U_k. In this dissertation, we refer to Chebyshev polynomials of the first kind as Chebyshev polynomials. The Chebyshev polynomials are computed through the recurrence relation

$$T_0(x) = 1 \;\; T_1(x) = x \;\; T_{k+1}(\tau) = 2xT_k(x) - T_{k-1}(x), \tag{A.1}$$

or the Chebyshev polynomial of degree k is defined by the identity:

$T_k(x) = \cos(k\cos^{-1}(x)) : x\epsilon[-1, 1]$.

The continuous orthogonality conditions for Chebyshev polynomials are

$$\int_{-1}^{1} w(x)T_n(x)T_m(x)dx = \begin{cases} 0 : n \neq m \\ \pi : n = m = 0 \\ \pi/2 : n = m \neq 0 \end{cases} \quad \text{and } w(x) = (1 - x^2)^{-\frac{1}{2}}. \tag{A.2}$$

The discrete orthogonality conditions for the Chebyshev polynomials using the CGL nodes are

$$\sum_{k=0}^{M} w_k T_n(x_k)T_m(x_k) = \begin{cases} 0 : n \neq m \\ M : n = m = 0 \\ M/2 : n = m \neq 0 \end{cases} \quad \text{and } w_0 = w_M = \frac{1}{2},\; w_k = 1; k = 1, 2, ..., M-1. \tag{A.3}$$

The $(N+1)$ CGL (or "cosine") nodes for the N^{th} order Chebyshev polynomials are calculated from

$$x_k = \cos\left(\frac{k\pi}{M}\right);\ k = 0, 1, 2, ..., M. \tag{A.4}$$

Indefinite integration of the Chebyshev polynomials has the property $\int T_k(x)dx = \frac{1}{2}\left(\frac{T_{k+1}}{k+1} - \frac{T_{k-1}}{k-1}\right)$.

The first derivative of the Chebyshev polynomials satisfies

$$\frac{dT_k(x)}{dx} = kU_{k-1}(x) = k(1-x^2)^{-1}[-xT_k(x) + T_{k-1}(x)]. \tag{A.5}$$

Thus integrals and the derivatives are expressed as recursions contiguous degree Chebyshev polynomials. The first six Chebyshev polynomials are shown in Figure A.1.

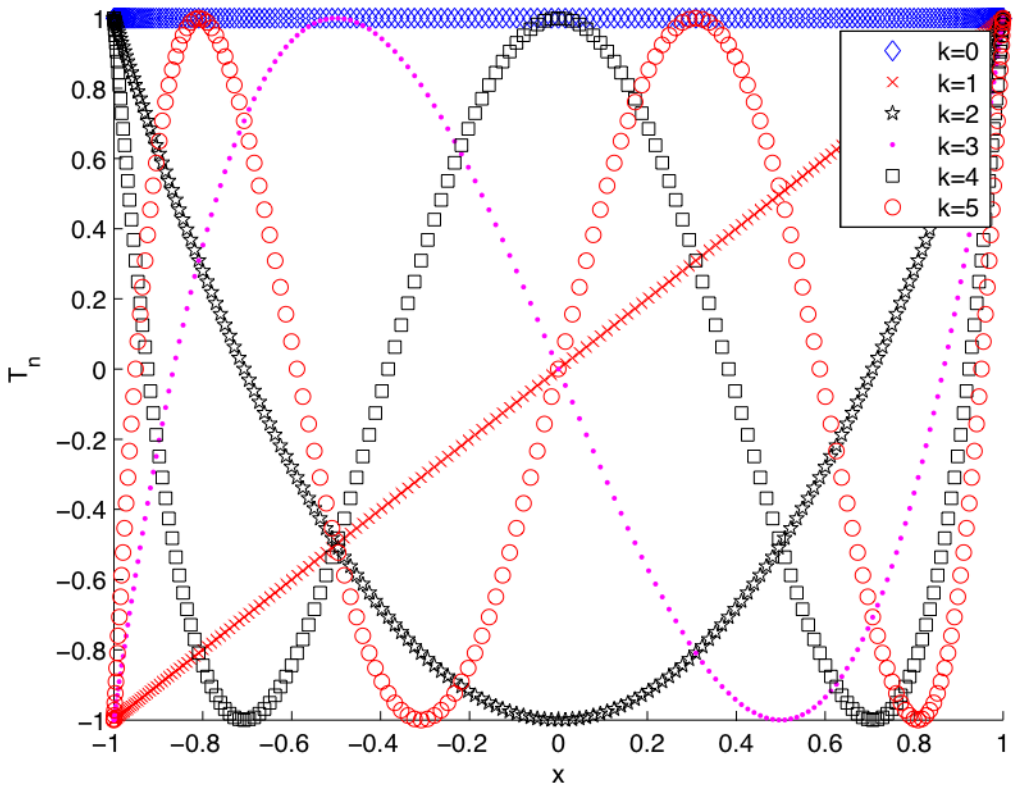


Figure A.1: Chebyshev Polynomials of the First Kind.

APPENDIX B

KRONECKER FACTORIZATION AND LEAST SQUARE APPROXIMATION

Proof of the important property regarding Kronecker factorization in Least squares that if a matrix Φ of rank n with $\Phi \in R^{m\times n}$; $m \geq n$ can be *Kronecker factorized* as

$$\Phi = \Phi_x \otimes \Phi_y, \tag{B.1}$$

then the classical normal equations

$$a = \left\{ \left(\Phi^T \Phi\right) \Phi^T \right\} \boldsymbol{f} \tag{B.2}$$

can, amazingly, be rewritten as

$$a = \left\{ \left(\Phi_x^T \Phi_x\right)^{-1} \Phi_x^T \right\} \otimes \left\{ \left(\Phi_y^T \Phi_y\right)^{-1} \Phi_y^T \right\} \boldsymbol{f}. \tag{B.3}$$

That is, the large "least square operator" $\left\{ \left(\Phi^T \Phi\right)^{-1} \Phi^T \right\}$ is rewritten as simply the Kronecker product of two small matrices:

$$\left\{ \left(\Phi^T \Phi\right)^{-1} \Phi^T \right\} = \left\{ \left(\Phi_x^T \Phi_x\right)^{-1} \Phi_x^T \right\} \otimes \left\{ \left(\Phi_y^T \Phi_y\right)^{-1} \Phi_y^T \right\}. \tag{B.4}$$

The matrices $\left(\Phi_x^T \Phi_x\right), \left(\Phi_y^T \Phi_y\right)$ must obviously be non-singular. To prove this identity, we need the following three properties of Kronecker matrix operations for square and nonsingular matrices A and B

$$\left(A \otimes B\right)^T = A^T \otimes B^T, \tag{B.5}$$

$$\left(A_1 \otimes A_2\right)\left(B_1 \otimes B_2\right) = \left(A_1 B_1\right) \otimes \left(A_2 B_2\right), \tag{B.6}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \tag{B.7}$$

The property of Eq. (B.4) is proven as follows: Using the assumed factorization of Eq. (B.1), $\left\{\left(\Phi^T\Phi\right)^{-1}\Phi^T\right\}$ is written as

$$\left\{\left(\Phi^T\Phi\right)^{-1}\Phi^T\right\} = \Big((\Phi_x \otimes \Phi_y)^T (\Phi_x \otimes \Phi_y)\Big)^{-1} (\Phi_x \otimes \Phi_y)^T . \tag{B.8}$$

Then using Eqs. (B.5), (B.8) re-arranges to

$$\left\{\left(\Phi^T\Phi\right)^{-1}\Phi^T\right\} = \left(\left(\Phi_x^T \otimes \Phi_y^T\right)(\Phi_x \otimes \Phi_y)\right)^{-1}\left(\Phi_x^T \otimes \Phi_y^T\right), \tag{B.9}$$

and using Eq. (B.6), Eq. (B.9) becomes condition

$$\left\{\left(\Phi^T\Phi\right)^{-1}\Phi^T\right\} = \left(\left(\Phi_x^T \otimes \Phi_x\right)\left(\Phi_y^T \otimes \Phi_y\right)\right)^{-1}\left(\Phi_x^T \otimes \Phi_y^T\right). \tag{B.10}$$

Using Eq. (B.7), Eq. (B.10) is

$$\left\{\left(\Phi^T\Phi\right)^{-1}\Phi^T\right\} = \Big(\left(\Phi_x^T \otimes \Phi_x\right)^{-1}\left(\Phi_y^T \otimes \Phi_y\right)^{-1}\Big)\left(\Phi_x^T \otimes \Phi_y^T\right), \tag{B.11}$$

and finally, using the property of Eq. (B.6), Eq.(B.11) becomes Eq. (B.4), Q.E.D.

This property extends to high dimensioned Kronecker factorizations, i.e., if

$$\Phi = \Phi_x \otimes \Phi_y \otimes \Phi_z, \tag{B.12}$$

then the large least square operator is written as the Kronecker product of three small least square operators as

$$\left\{\left(\Phi^T\Phi\right)^{-1}\Phi^T\right\} = \left\{\left(\Phi_x^T\Phi_x\right)^{-1}\Phi_x^T\right\} \otimes \left\{\left(\Phi_y^T\Phi_y\right)^{-1}\Phi_y^T\right\} \otimes \left\{\left(\Phi_z^T\Phi_z\right)^{-1}\Phi_z^T\right\}. \tag{B.13}$$

These results are easily extended to include the weighted least square case, as well. For the special case that the basis functions in 1, 2, and 3 dimensions satisfy orthogonality conditions such that the off-diagonal elements of $\left(\Phi_x^T\Phi_x\right)$, $\left(\Phi_y^T\Phi_y\right)$, $\left(\Phi_z^T\Phi_z\right)$

vanish, then likewise the larger matrix $\left(\Phi^T \Phi\right)$ is diagonal and we see that Eq. (B.12) and (B.13) also provide very convenient means for generalizing one dimensional orthogonal approximation operators to higher dimensions. Care must always be taken to understand and properly choose the multidimensional nodal sample patterns and weight matrices, to ensure orthogonality of the basis functions with respect to both the weight matrices and nodal locations.

APPENDIX C

GRACE GEOPOTENTIAL MODEL APPROXIMATION

The reference gravity potential is simply the two point mass term and rhe J_2 perturbation, everything else is approximated. Figure C.1 shows the perturbed (GRACE 156×156) Global FEM Gravity Potential Approximation at the Earth's surface.

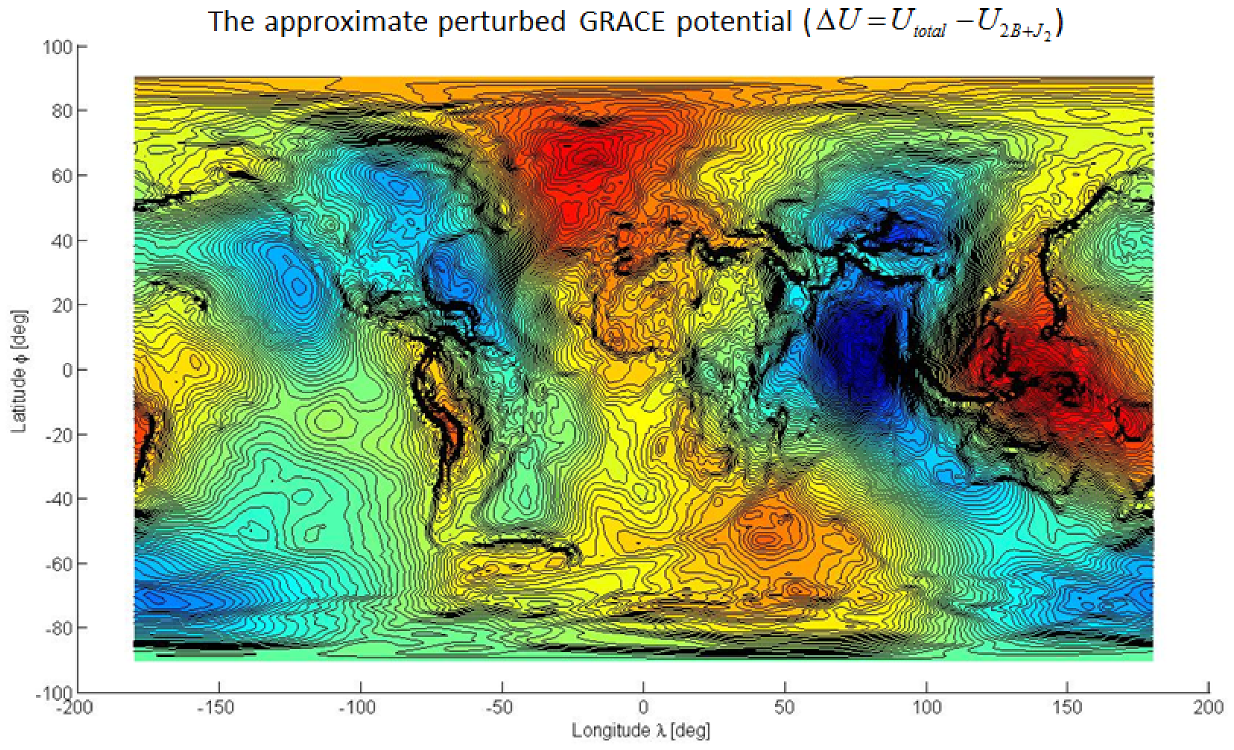

The approximate perturbed GRACE potential ($\Delta U = U_{total} - U_{2B+J_2}$)
Latitude ϕ [deg]
Longitude λ [deg]
100
80
60
40
20
0
-20
-40
-60
-80
-100
-200
-150
-100
-50
0
50
100
150
200

Figure C.1: Approximation of the Perturbed GRACE Geopotential.

APPENDIX D

NUMERICAL EXAMPLES FOR ILLUSTRATIVE TEST FUNCTIONS OF THREE VARIABLES

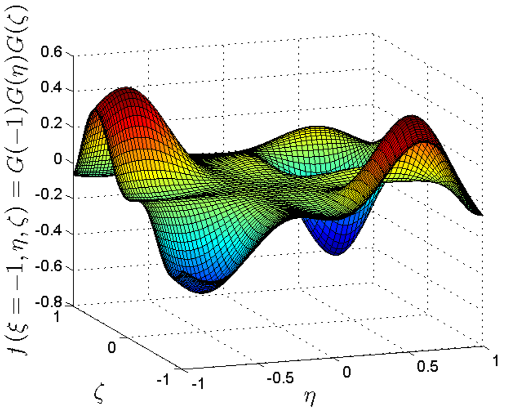
To construct some three dimensional test cases that relate closely to the one and two dimensional examples, we define Test Function 3: $f(\xi, \eta, \zeta) \equiv G(\xi)G(\eta)G(\zeta)$ where

$$G(x) = \frac{x}{2} + \left(\frac{\left[\left(\frac{1}{10} + x\right) sin(5x - 1)\right]}{\left[1 + x^2 sin^2\left(x - \frac{1}{2}\right)\right]} \right). \tag{D.1}$$

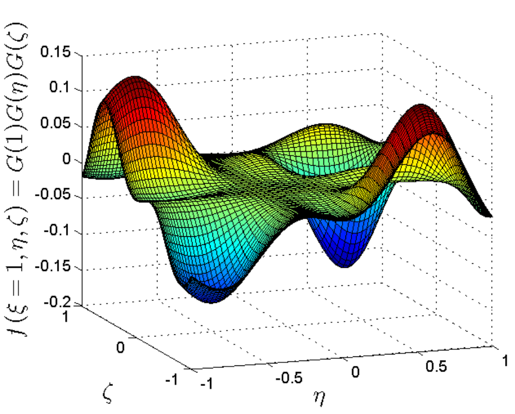
Figure II.9 is an illustration of the cosine nodal distribution in one, two and three dimensional spaces; the generalization to a hypercube is straightforward. We now consider several cases analogous to the one and two dimensional cases. Similar experiments as in the Test Function 1 and Test Function 2 case are performed to generate measurements with either $M_x = M_y = M_z = M = 25$ or $M_x = M_y = M_z = M = N$, with N swept. Because of having three variables, note that we present the 4-D plots at different locations of the third variable to make the visualization much easier. The true Test Function 3 Boundary Surfaces are shown in Figure D.1 and the true Test Function 3 Center Slices Surfaces are shown in Figures D.2. Figures D.3 through D.6 are several approximations. The Runge Phenomena as (note large boundary errors for the power series approximation versus the more uniform Chebyshev approximation, which is concentrated in the center) can be expected to generalize for higher dimensioned cases. We approximate Test Function 3 with a residual error approaching a machine zero. All computations are performed using MATLAB® with 16 digit floating point arithmetic.

Figures D.3 through D.7show the approximation error for the Chebyshev and power series polynomial approximation of Test Function 3 for ($M = N = 5$). The power series experienced large Runge errors near the boundary and the least square solutions “died” altogether due to ill-conditioning around $N \sim 13$. Note for low degree approximation that the power series works fairly well in the center of the interval, but encounters large errors near the boundary. The maximum errors are shown in Figure D.7a and D.7b that result from least square approximation when $M = 25$ measurement nodes are used, for the case of the Chebyshev and power series polynomial approximation of Test Function 3. The Chebyshev approximations converged to 6 digit accuracy around $N = 22$, and ~ 15 digit accuracy is obtained (essentially a machine zero approximation error) around $N \cong 50$.

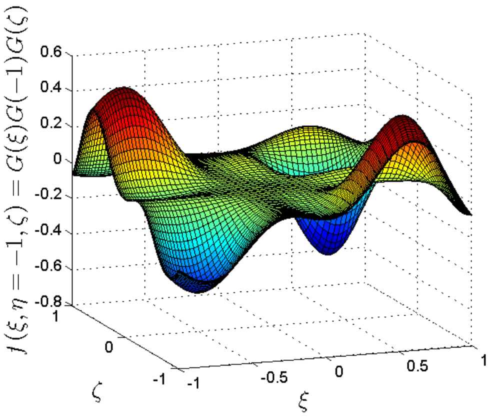
The uniform convergence of the Chebyshev approximation again approaches machine precision by $N \cong 50$, with the maximum error decreasing about one order of magnitude every time the degree N is increased by $\Delta N \approx 3$. On the other hand, the slope is much less for $N < 13$ for the power series case (due to the Runge Phenomena), and the power series cannot be computed accurately above $N \sim 13$ due to poor conditioning of the normal Eqs (2.9), which must be inverted numerically for the case of non-orthogonal basis functions.



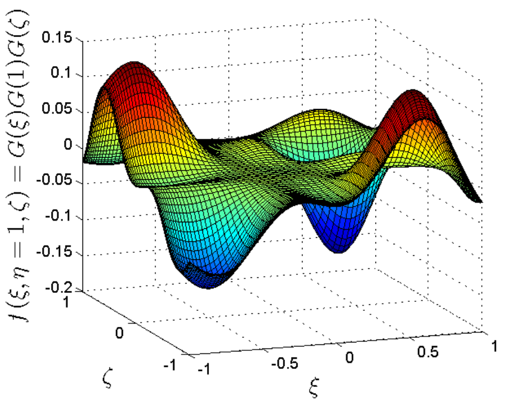
(a) Test Function 3 (@ $\xi = -1$).



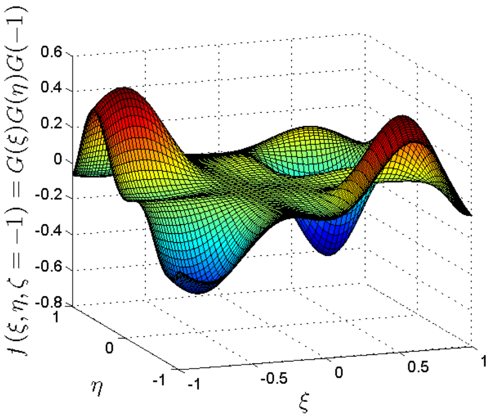
(b) Test Function 3 (@ $\xi = 1$).



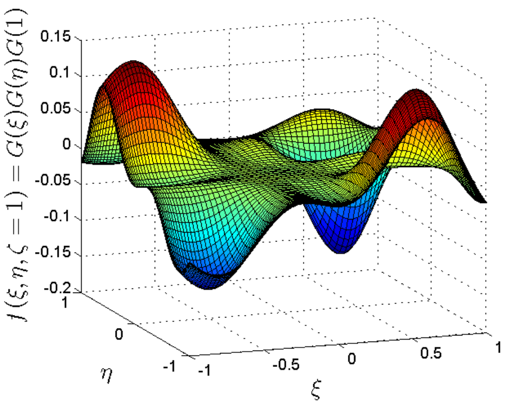
(c) Test Function 3 (@ $\eta = -1$).



(d) Test Function 3 (@ $\eta = 1$).

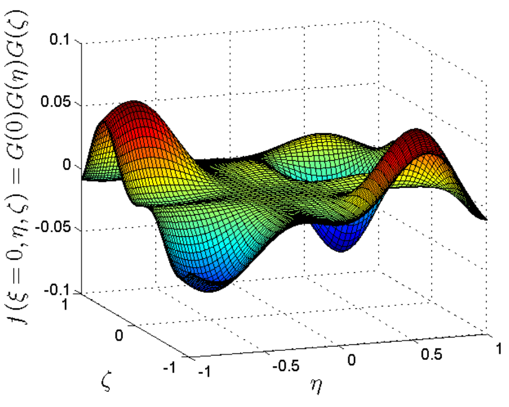


(e) Test Function 3 (@ $\zeta = -1$).

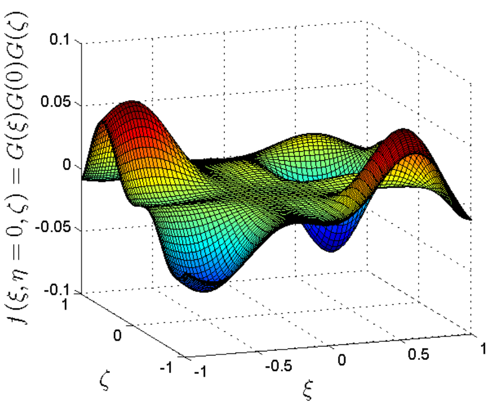


(f) Test Function 3 (@ $\zeta = 1$).

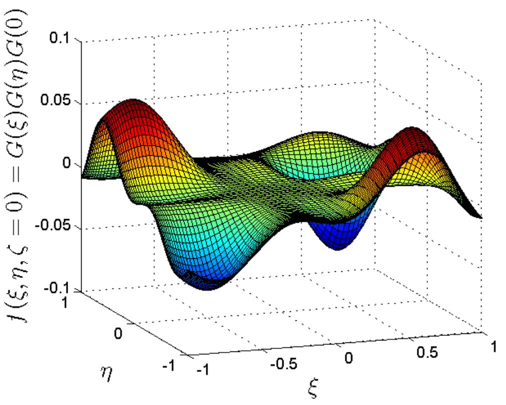
Figure D.1: Test Function 3 Boundary Surfaces.



(a) Test Function 3 (@ $\xi = 0$).

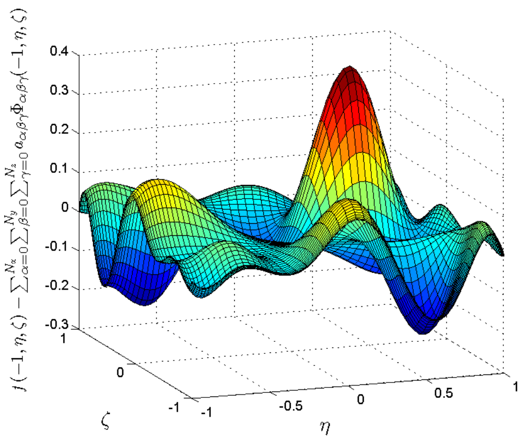


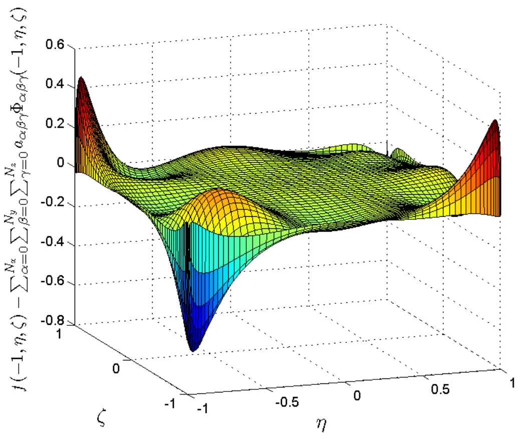
(b) Test Function 3 (@ $\eta = 0$).

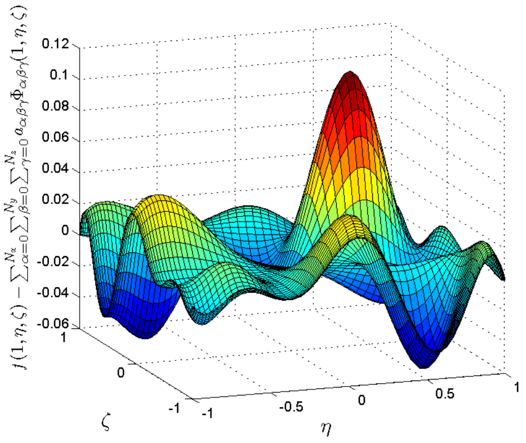


(c) Test Function 3 (@ $\zeta = 0$).

Figure D.2: Test Function 3 Center Slices Surfaces.


(a) Chebyshev Approximation (@ $\xi = -1$).


(b) Power Series Approximation (@ $\xi = -1$).


(c) Chebyshev Approximation (@ $\xi = 1$).

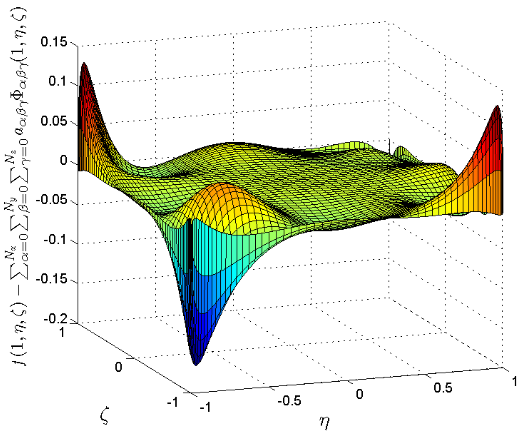
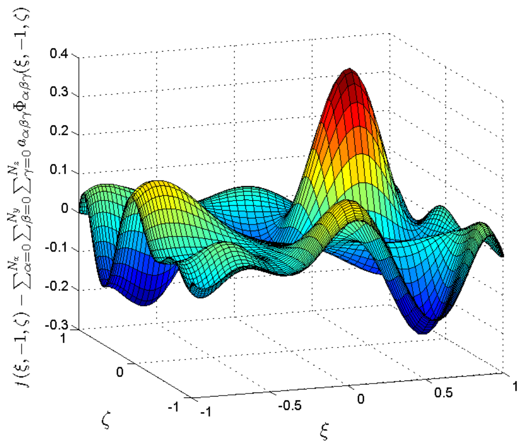
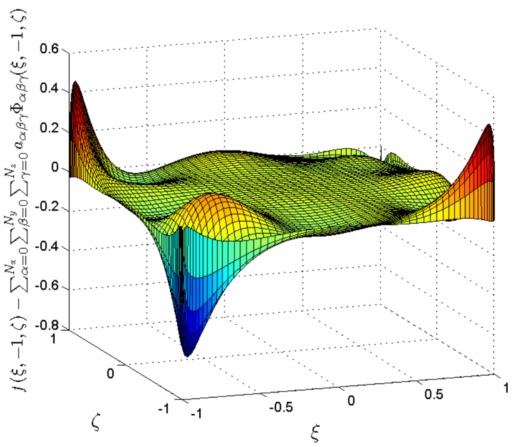
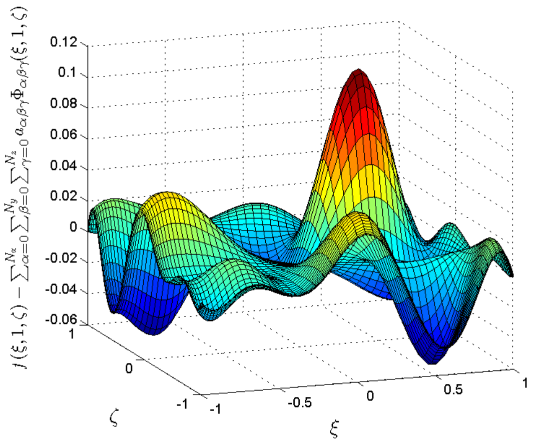

(d) Power Series Approximation (@ $\xi = 1$)

Figure D.3: Approximation of Test Function 3 Boundary Surfaces (@ $\xi = \pm 1$).


(a) Chebyshev Approximation (@ $\eta = -1$).


(b) Power Series Approximation (@ $\eta = -1$).


(c) Chebyshev Approximation (@ $\eta = 1$).

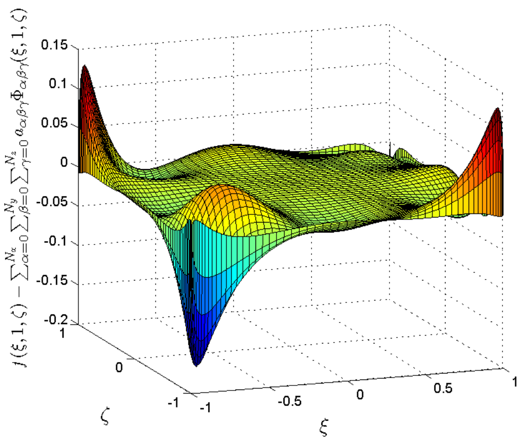
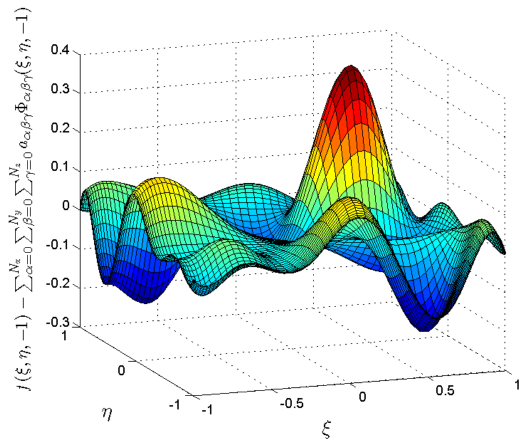
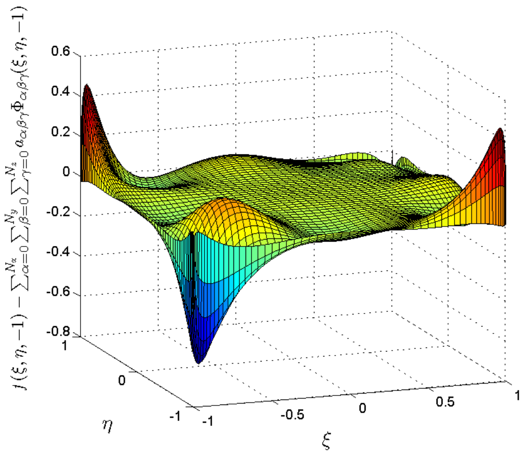
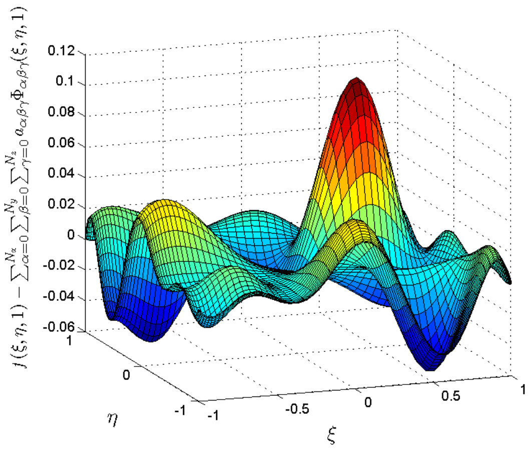

(d) Power Series Approximation (@ $\eta = 1$).

Figure D.4: Approximation of Test Function 3 Boundary Surfaces (@ $\eta = \pm 1$).


(a) Chebyshev Approximation (@ $\zeta = -1$).


(b) Power Series Approximation (@ $\zeta = -1$).


(c) Chebyshev Approximation (@ $\zeta = 1$).

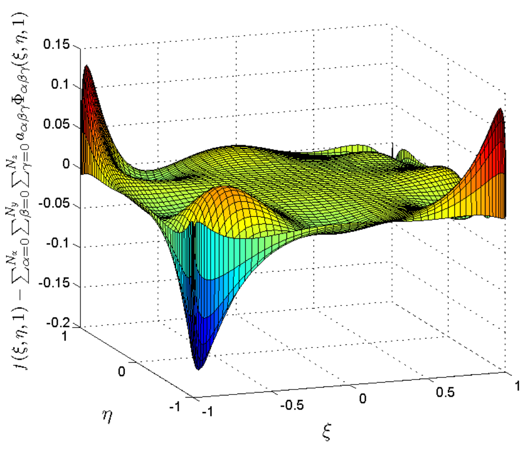
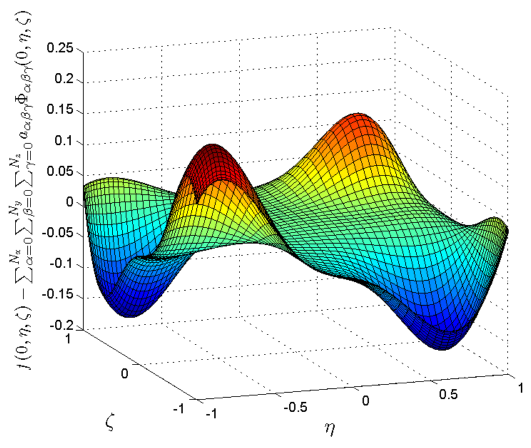
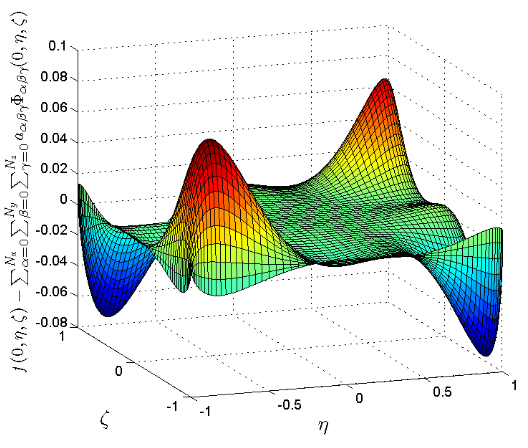

(d) Power Series Approximation (@ $\zeta = 1$).

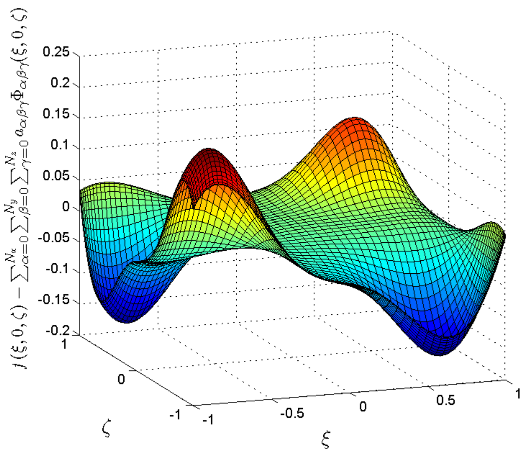
Figure D.5: Approximation of Test Function 3 Boundary Surfaces (@ $\zeta = \pm 1$).



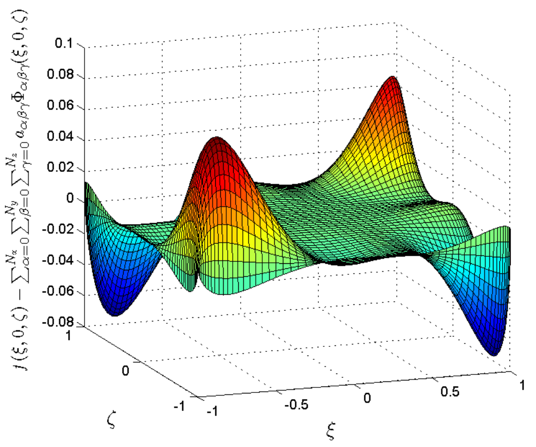
(a) Chebyshev Approximation (@ $\xi = 0$).



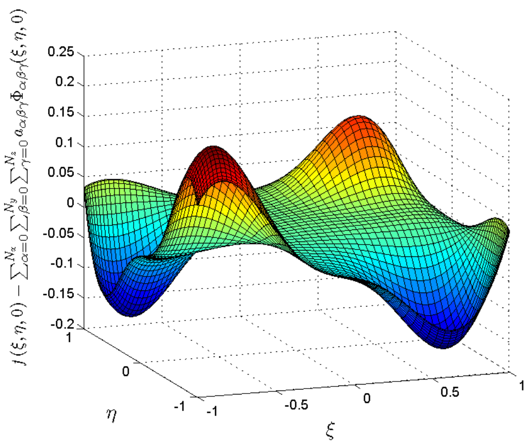
(b) Power Series Approximation (@ $\xi = 0$).



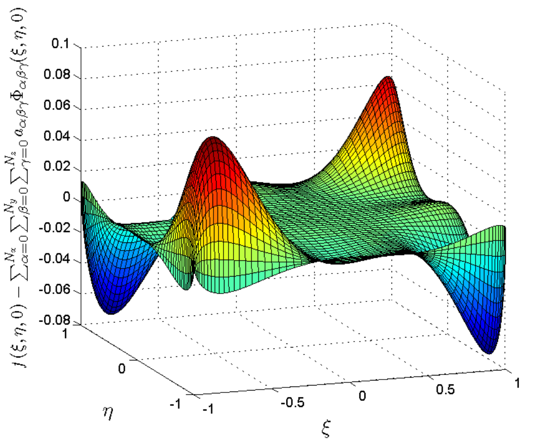
(c) Chebyshev Approximation (@ $\eta = 0$).



(d) Power Series Approximation (@ $\eta = 0$).

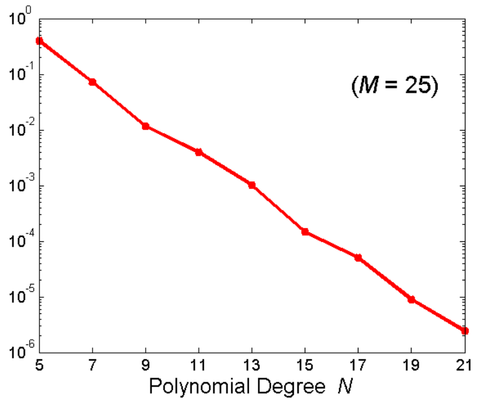


(e) Chebyshev Approximation (@ $\zeta = 0$).

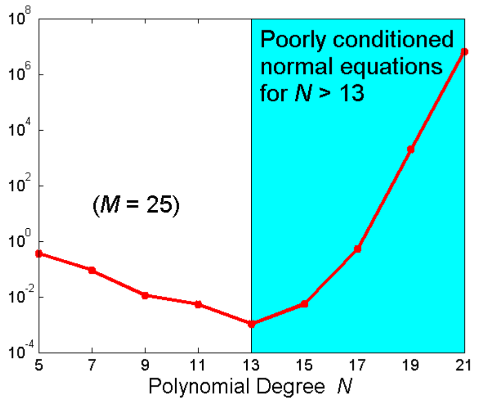


(f) Power Series Approximation (@ $\zeta = 0$).

Figure D.6: Test Function 3 Center Slices Surfaces (@ $\{\xi,\eta,\zeta\} = 0$).



(a) Chebyshev Maximum Approximation Error.



(b) Power Series Maximum Approximation Error.

Figure D.7: Approximation Error of Test Function 3.