The Journal of the Astronautical Sciences, Vol. 58, No. 4, October-December 2011, pp. 615–642

Modified Chebyshev-Picard Iteration Methods for Solution of Boundary Value Problems

Xiaoli Bai[1] and John L. Junkins[2]

Abstract

Modified Chebyshev-Picard iteration methods are presented for solving boundary value problems. Chebyshev polynomials are used to approximate the state trajectory in Picard iterations, while the boundary conditions are maintained by constraining the coefficients of the Chebyshev polynomials. Using Picard iteration and Clenshaw–Curtis quadrature, the presented methods iteratively refine an orthogonal function approximation of the entire state trajectory, in contrast to step-wise, forward integration approaches, which render the methods well-suited for parallel computation because computation of force functions along each path iteration can be rigorously distributed over many parallel cores with negligible cross communication needed. The presented methods solve optimal control problems through Pontryagin's principle without requiring shooting methods or gradient information. The methods are demonstrated to be computationally efficient and strikingly accurate when compared with Battin's method for a classical Lambert's problem and with a Chebyshev pseudospectral method for an optimal trajectory design problem. The reported simulation results obtained on a serial machine suggest a strong basis for optimism of using the presented methods for solving more challenging boundary value problems, especially when highly parallel architectures are fully exploited.

Introduction

Solutions of boundary value problems (BVPs) provide the system trajectories that satisfy several constraints defined at different times. Although general BVPs can have constraint conditions on the states at any time in the interested time interval, we presently restrict discussion to two-point BVPs, for which some of the states are defined at an initial time and the others are defined at a final time.

Optimal control problems for continuous systems are considered. These give rise to a special type of BVPs addressed in this paper. Given a performance

[1]Postdoctoral Research Associate, Department of Aerospace Engineering, Texas A&M University, TAMU-3141, College Station, Texas 77843-3141.
[2]Regents Professor, Distinguished Professor of Aerospace Engineering, Holder of the Royce E. Wisenbaker '39 Chair in Engineering, Department of Aerospace Engineering, Texas A&M University, TAMU-3141, College Station, Texas 77843-3141.

function, a set of ordinary differential equations (ODEs) describing the dynamic system, and various system constraints, an optimal control design algorithm solves for the control input that drives the system to execute an optimal trajectory to minimize or maximize the performance function and satisfy the constraints. The two most popular sets of computational techniques for solving optimal control problems are direct shooting and indirect shooting methods [1]. Direct methods introduce a parametric representation of the control variables (and frequently the state variables as well), and then use nonlinear programming optimizers to solve the resulting nonlinear parameter optimization problems (a so-called nonlinear programming problem). With the increasing power of these optimizers, direct approaches can obtain sub-optimal solutions often more easily than indirect approaches. However, the accuracy and optimality of the returned control policy for the original continuous system are not guaranteed. Indirect approaches are based on calculus of variations without parameterization of state and control variables. Necessary conditions are derived from Pontryagin's principle, [2] giving rise to a set of co-state (adjoint) Lagrangian multiplier variables and transversality split boundary conditions. A simple shooting or multiple shooting method is usually used to solve the resulting two-point BVP, with the goal being to find the unknown initial states and co-states to satisfy terminal conditions and transversality conditions. When accurate solutions are obtained from the indirect approaches, they assure local optimality and accuracy. However, solving the resulting BVP is often challenging because the solutions can be very sensitive to the initial guess that is typically required to start the shooting methods.

The recent interest in using parallel computation architectures for solving optimal control problems can be categorized as either devising parallel linear algebra software and algorithms or developing new parallel strategies in global optimization [3]. Travassos [4] suggested using Miranker and Liniger's algorithms [5] for parallel integration. He compared different parallel algorithms for function minimization, and discussed using a parallel shooting method to solve for the unknown initial co-states. The level of parallelism of his approach is $n(2N-1)$, where n is the number of states and N is the number of subintervals used in the parallel shooting. Betts and Huffman broke the overall trajectory into phases to parallelize the trajectory optimization algorithms [6]. The sparse Jacobian matrix resulting from the multiple shooting method was solved using sparse finite differences by parallel computation. They implemented their algorithm on a BBN GP100 (Butterfly) parallel processor for four orbit transfer problems and the speedup they achieved was usually in the range of 1.9 to 9.9. However, the results also showed that their proposed parallel approach was not always faster than the serial direct approach, and the authors suggested this is because additional variables and constraints were introduced by the way they parallelized the problems.

Recently, Bai and Junkins have developed a parallel-structured modified Chebyshev-Picard iteration (MCPI) methodology for solving initial value problems (IVPs) [7–9]. Combining Chebyshev polynomial approximation and quadrature with Picard iteration, MCPI methods iteratively refine an approximation of the entire solution in contrast to traditional, step-wise, forward integration methods. MCPI methods have been reported to solve IVPs, especially orbit propagation problems, with great success in terms of both accuracy and efficiency on a serial machine. However, the most attractive feature of MCPI methods is that the methods are inherently parallel as computation of force functions along each path

iteration can be rigorously distributed over many parallel cores with negligible cross communication needed. Although a most straightforward way to benefit from MCPI methods in solving BVPs is to use MCPI formulations in solving IVPs to propagate the dynamic system (which is often required for both direct shooting and indirect shooting methods), interestingly and importantly, the basic methodology of MCPI methods is of a fundamental nature and can be used for solving BVPs directly. The result is a hybrid method that parameterizes the solution of the Pontryagin's necessary conditions and introduces a modified Picard iteration to determine an accurate solution of the necessary conditions.

Picard iteration is a successive solution approximation technique that is often used to prove the existence and uniqueness of the solutions to IVPs. A maybe less known truth is that Picard also used a similar approach to solve two-point BVPs described by second order ODEs [10]. Picard developed a theoretical condition for the length of the time intervals over which Picard iteration is guaranteed to converge. Including Picard himself, a number of authors have worked on improving these conditions [11–15]; generally these developments have sought to make Picard's original convergence conditions sharper and less conservative. However, these conditions, even after several improvements, still provide a very conservative estimate on the region of the convergence; the "best possible" interval where Picard iteration converges, for a general problem, is not yet known. Also, there are a number of counterexamples in which Picard iteration is guaranteed not to converge, even if the starting function is arbitrarily close to the true solution [16, 17].

Chebyshev polynomials are a complete set of orthogonal polynomials that are very important for function approximation. If the zeros of Chebyshev polynomials are used as the nodes for polynomial interpolations, the resulting approximating polynomial minimizes the Runge's phenomenon and provides the best approximation under the minimax norm [18]. There have been some research on using Chebyshev polynomials to solve BVPs [19–21], but typically not adopting Picard iteration as the basis for the solution process. Practical numerical methods for finding the Chebyshev coefficients using this approach are not available now.

The proposed MCPI approach builds on the historical formulations by a small group of researchers who studied solving BVPs using Picard iteration, Chebyshev polynomials, and Clenshaw–Curtis quadrature (Chebyshev-Picard methods) [22, 23]. These pioneers approximated the trajectory and the integrand by the same orthogonal basis functions (discrete Chebyshev polynomials) and integrated the basis functions term-by-term through Picard iteration to establish a recursive trajectory approximation technique. Because of the limited utility and rather simple example problems that have been studied for the previous Chebyshev-Picard methods, none of them has been considered a viable competitor to traditional direct and indirect shooting methods. With the available advanced and inexpensive parallel computation techniques such as Graphics Processing Units (GPUs), in solving BVPs, MCPI methods have made the following contributions.

- A unified vector-matrix form, together with an explicit formulation for a special kind of second order BVPs, have been developed and proven to be computationally efficient.

- The convergence characteristics of Chebyshev-Picard methods in solving BVPs has been studied and new results are reported.
- This is the first time that MCPI methods are compared with some state-of-the-art methods in solving two important BVPs in the celestial mechanics field. The significant computational efficiency and solution accuracy of Chebyshev-Picard methods are first reported.

We present the fundamental formulation of MCPI methods for solving BVPs first in the paper. Then we develop a formulation for a special type of second order BVPs, which is followed by a convergence analysis for a linear second order system. The classical Lambert's problem and an optimal control design problem are solved as examples to demonstrate the performance of MCPI methods, where the solution accuracy and computational time of MCPI methods are compared with Battin's method and a Chebyshev pseudospectral method respectively.

Fundamentals of the MCPI Approach

Consider a dynamic system described by a first order differential equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(t, \mathbf{x}) \tag{1}$$

A two-point BVP seeks the solution for $\mathbf{x}(t)$ that satisfies equation (1) and boundary conditions for $\mathbf{x}$, some of which are defined at the initial time ($t = t_0$) and others are defined at the final time ($t = t_f$). 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

$$t = \omega_1 + \omega_2\tau \qquad \omega_1 = \frac{t_f + t_0}{2} \qquad \omega_2 = \frac{t_f - t_0}{2} \tag{2}$$

Introducing this time transformation of equation (2) into equation (1), it is rewritten as

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\tau} = \mathbf{g}(\tau, \mathbf{x}) \equiv \omega_2\mathbf{f}(\omega_1 + \omega_2\tau, \mathbf{x}) \tag{3}$$

and the classical Picard iteration provides the solution of equation (3) as [24]

$$\mathbf{x}^i(\tau) = \mathbf{x}_0 + \int_{-1}^{\tau} \mathbf{g}(s, \mathbf{x}^{i-1}(s))\,\mathrm{d}s \quad i = 1, 2, \ldots \tag{4}$$

Now, we introduce Chebyshev polynomial approximations of both the unknown trajectory $\mathbf{x}^{i-1}$ and the integrand of equation (4) along the trajectory $\mathbf{x}^{i-1}$. The Chebyshev polynomial of degree k is denoted by T_k and the $(N + 1)$ discrete nodes that are used to represent the state trajectory are the Chebyshev-Gauss-Lobatto (CGL) nodes, which are computed through

$$\tau_j = \cos(j\pi/N) \quad j = 0, 1, 2, \ldots, N \tag{5}$$

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

$$\mathbf{g}(\tau, \mathbf{x}^{i-1}(\tau)) \approx \sum_{k=0}^{k=N}{}' \mathbf{F}_k^{i-1} T_k(\tau)$$

$$\equiv \tfrac{1}{2}\mathbf{F}_0^{i-1} T_0(\tau) + \mathbf{F}_1^{i-1} T_1(\tau) + \mathbf{F}_2^{i-1} T_2(\tau) + \cdots + \mathbf{F}_N^{i-1} T_N(\tau) \tag{6}$$

Using the discrete orthogonality property of Chebyshev polynomials, the coefficient vectors f_k can be calculated immediately through [18]

$$\mathbf{F}_k^{i-1} = \frac{2}{N}\left\{\sum_{j=0}^{N}{}'' \mathbf{g}(\tau_j, x^{i-1}(\tau_j)) T_k(\tau_j)\right\}$$

$$= \frac{2}{N}\left\{\frac{1}{2}\mathbf{g}(\tau_0, x^{i-1}(\tau_0)) T_k(\tau_0) + \mathbf{g}(\tau_1, x^{i-1}(\tau_1)) T_k(\tau_1) + \cdots \right.$$

$$\left. + \frac{1}{2}\mathbf{g}(\tau_0, x^{i-1}(\tau_0)) T_k(\tau_0)\right\} \tag{7}$$

In equations (6) and (7), Σ' denotes that the first term is halved and Σ'' represents that both the first and last terms are halved. Notice each coefficient of $\mathbf{F}_k^{i-1}$ is obtained through the summation of $(N + 1)$ independent terms, each of which is an inner product of the force function $\mathbf{g}(\tau, x(\tau))$ and the Chebyshev polynomials $\mathbf{T}_k(\tau)$ evaluated at the CGL points of equation (5). Furthermore, all the coefficient vectors are independent of each other, and can therefore be computed in parallel processors. Most importantly, for problems where calculating the force vector function $\mathbf{g}(\tau, x(\tau))$ is time consuming, significant time performance improvement can be achieved by simultaneously computing $\mathbf{g}(\tau_j, x(\tau_j))$ at each τ_j by using $(N + 1)$ parallel processors. Picard iteration in equation (4) provides the recursion to calculate $x^i(\tau)$, the solution at the i^{th} step, as a Chebyshev polynomial approximation over the entire time interval as

$$\mathbf{x}^i(\tau) = \mathbf{x}_0 + \sum_{r=0}^{N}{}' \mathbf{F}_r^{i-1} \int_{-1}^{\tau} T_r(s)\, ds \equiv \sum_{k=0}^{N}{}' \beta_k^i T_k(\tau) \tag{8}$$

Imposing the two-point boundary conditions

$$\mathbf{x}_0 = \mathbf{x}(-1) = \sum_{k=0}^{k=N}{}' \beta_k T_k(-1) \tag{9}$$

$$\mathbf{x}_f = \mathbf{x}(1) = \sum_{k=0}^{k=N}{}' \beta_k T_k(1) \tag{10}$$

and recollecting the term-by-term analytical integration of equation (8), the coefficient vectors for this new trajectory are expressed below in equation (11–14). Detailed derivation of these formulations can be found in Bai's dissertation [9].

$$\beta_r^i = \frac{1}{2r}(\mathbf{F}_{r-1} - \mathbf{F}_{r+1}) \qquad r = 2, 3, \ldots, N-1 \tag{11}$$

$$\beta_N^i = \frac{\mathbf{F}_{N-1}}{2N} \tag{12}$$

$$\beta_0^i = \mathbf{x}_0 + \mathbf{x}_f - 2(\beta_2 + \beta_4 + \beta_6 + \cdots) \tag{13}$$

$$\beta_1^i = \frac{(\mathbf{x}_f - \mathbf{x}_0)}{2} - (\beta_3 + \beta_5 + \beta_7 + \cdots) \tag{14}$$

Notice the only difference of the MCPI formulation between solving IVPs and solving BVPs is how the constraint conditions are embedded to constrain the Chebyshev coefficients. For the states that only have initial-time or final-time constraints, equation (9) or (10) leads to the corresponding equations to constrain the zero coefficient. The updated coefficient vectors define the new trajectory approximation for use in integrand (equation (6)) for the next step $(i + 1)$—see Fig. 1 for the algorithm overview. Thus the solutions are iteratively improved until some accuracy requirements are satisfied. To account for the nonlinearity issues, the stopping criterion we choose is to require both the maximum difference (among all the N+1 CGL nodes) between solutions $x^i(\tau)$ and $x^{i-1}(\tau)$ and the maximum difference between solutions $x^i(\tau)$ and $x^{i+1}(\tau)$ are less than some tolerance.

Instead of term by term scalar process to solve for the state values at the $(N + 1)$ CGL nodes, the $(N + 1)$ Chebyshev coefficients, and the updated $(N + 1)$ Chebyshev coefficients, a computationally more efficient vector-matrix approach can be used to implement MCPI methods. With the definition of three diagonal matrices

$$W = \text{diag}([\tfrac{1}{2}, 1, 1, \ldots, 1]) \tag{15}$$

$$R = \text{diag}\left(\left[1, \frac{1}{2}, \ldots, \frac{1}{2r}, \ldots, \frac{1}{2(N-1)}, \frac{1}{2N}\right]\right) \qquad r = 1, 2, \ldots, N \tag{16}$$

$$V = \text{diag}\left(\left[\frac{1}{N}, \frac{2}{N}, \frac{2}{N}, \ldots, \frac{2}{N}, \frac{1}{N}\right]\right) \tag{17}$$

and

$$T = \begin{bmatrix} T_0(\tau_0) & T_1(\tau_0) & \ldots & T_N(\tau_0) \\ T_0(\tau_1) & T_1(\tau_1) & \ldots & T_N(\tau_1) \\ \vdots & \vdots & \vdots & \vdots \\ T_0(\tau_N) & T_1(\tau_N) & \ldots & T_N(\tau_N) \end{bmatrix} \tag{18}$$

the state value vector $\vec{\mathbf{X}} = [x(\tau_0), x(\tau_1), x(\tau_2), \ldots, x(\tau_N)]^T$ is computed from its Chebyshev coefficient vector $\vec{\beta} = [\beta_0, \beta_1, \beta_2, \ldots, \beta_N]^T$ through equation (19) (the derivation can be found in Bai's dissertation [9].)

$$\vec{\mathbf{X}} = \text{TW}\vec{\boldsymbol{\beta}} \equiv C_x\vec{\boldsymbol{\beta}} \tag{19}$$

and the coefficient vector is computed from the state value vector using

$$\frac{dx}{dt} = f(t, x(t)), \quad t_0 \le t \le t_f, \quad \{x(t_0) = x_0, t_0, t_f\} \text{ specified}$$

$$\Downarrow$$

Variable Change

$$t \equiv \omega_1 + \omega_2 \tau, \quad \omega_1 = \left(\frac{t_f + t_0}{2}\right), \quad \omega_2 = \left(\frac{t_f - t_0}{2}\right), \quad -1 \le \tau \le 1$$

$$\Downarrow$$

$$\frac{dx}{d\tau} = g(\tau, x) \equiv \omega_2 f(\omega_1 + \omega_2 \tau, x)$$

$$\Downarrow$$

Starting Trajectory Estimate

$$i = 1, \; x^0(\tau) \text{ specified}$$

$$\Downarrow$$

Picard Iteration:

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

$$\Downarrow$$

Discrete Orthogonal Integrand Approximation along the $(i-1)^{th}$ Trajectory:

$$g(\tau, x^{i-1}(\tau)) \cong \sum_{k=0}^{N}{}' F_k^{i-1} T_k(\tau), \text{ with } \begin{cases} F_k^{i-1} = \dfrac{2}{N} \sum_{j=0}^{N}{}'' g(\tau_j, x^{i-1}(\tau_j)) T_k(\tau_j), \quad \tau_j = \cos(j\pi / N) \\ \sum_{j=0}^{N}{}'(\bullet)_j \equiv \dfrac{1}{2}(\bullet)_0 + (\bullet)_1 + \cdots + (\bullet)_{N-1} + (\bullet)_N \\ \sum_{j=0}^{N}{}''(\bullet)_j \equiv \dfrac{1}{2}(\bullet)_0 + (\bullet)_1 + \cdots + (\bullet)_{N-1} + \dfrac{1}{2}(\bullet)_N \end{cases}$$

Enables the $(i-1)^{th}$ Picard Path Integrals to be Analytically Approximated:

$$x^i(\tau) = x_0 + \int_{-1}^{\tau} g(s, x^{i-1}(s))ds \cong x_0 + \sum_{r=0}^{N}{}' F_r^{i-1} \int_{-1}^{\tau} T_r(s)ds \equiv \sum_{k=0}^{N}{}' \beta_k^i T_k(\tau)$$

$$\Downarrow$$

Trajectory Approximation Update:

$$i = i+1 \quad \Leftarrow \quad x^i(\tau) = \sum_{k=0}^{N}{}' \beta_k^i T_k(\tau), \quad \text{where} \begin{cases} \beta_r^i = \dfrac{1}{2r}(F_{r-1}^{i-1} - F_{r+1}^{i-1}), r = 2, 3, \ldots, N-1 \\ \beta_0^i = x_0 + x_f - 2(\beta_2 + \beta_4 + \beta_6 + \cdots) \\ \beta_1^i = \dfrac{x_0 + x_f}{2} - (\beta_3 + \beta_5 + \beta_7 + \cdots) \\ \beta_N^i = \dfrac{F_{N-1}^{i-1}}{2N} \end{cases}$$

FIG. 1. MCPI Iteration for Solution of Boundary Value Problems.

$$\vec{\boldsymbol{\beta}} = \mathrm{TV}\vec{\mathbf{X}} \tag{20}$$

With the integrand evaluated at the $(N + 1)$ CGL nodes defined as $\vec{\mathbf{g}} = [g(\tau_0), g(\tau_1), g(\tau_2), \ldots, g(\tau_N)]^{\mathrm{T}}$, a vector-matrix form usually exists that transforms $\vec{\mathbf{g}}$ to the updated Chebyshev coefficients $\vec{\boldsymbol{\beta}}$ directly and also guarantees the approximate Chebyshev polynomials satisfy the constraint conditions. For IVPs, a unified formula defined in references [7–9]. exists to update the Chebyshev

coefficients. For general BVPs especially those resulting from optimal control problems, different forms exist according to where the boundary conditions are defined. For the state that only has initial condition constraint $x(t_0) = x_0$, together with the definition $\Theta_{x0} = [2x_0, 0, 0, \ldots, 0]^T \in R^{N+1}$, the updated Chebyshev coefficients are computed from

$$\vec{\boldsymbol{\beta}} \equiv \mathrm{RS}_i \mathrm{TV}\vec{\mathbf{g}} + \Theta_{x0} \tag{21}$$

For the states that the constraints are defined at the final time $x(t_f) = x_f$, together with the definition $\Theta_{xf} = [2x_f, 0, 0, \ldots, 0]^T \in R^{N+1}$, the updated Chebyshev coefficients are computed from

$$\vec{\boldsymbol{\beta}} \equiv \mathrm{RS}_f \mathrm{TV}\vec{\mathbf{g}} + \Theta_{xf} \tag{22}$$

For the state that has two-point constraints $x(t_0) = x_0$ and $x(t_f) = x_f$, together with the definition $\Theta_{xif} = [x_0 + x_f, (x_f - x_0)/2, 0, 0, \ldots, 0]^T \in R^{N+1}$, the updated Chebyshev coefficients are computed from

$$\vec{\boldsymbol{\beta}} \equiv \mathrm{RS}_{if} \mathrm{TV}\vec{\mathbf{g}} + \Theta_{xif} \tag{23}$$

The explicit forms of S_i, S_f, and S_{if} are given in Appendix A. These matrices are derived by using the properties of Chebyshev polynomials and imposing the boundary conditions. The derivation of S_i can be found in Bai's dissertation, in which S_f and S_{if} are derived in a similar way. Notice that all the matrices C_x, T, V, R, S_i, S_f, and S_{if} are constant (once N is selected), and so all computations of inner product can be efficiently precomputed.

Second Order MCPI Approach

For a system that is described by a second order differential equation, an explicit formula is presented here when the two-point boundary conditions of the position are specified. Assume the dynamic system is described as

$$\frac{\mathrm{d}^2\mathbf{x}(t)}{\mathrm{d}t^2} = \mathbf{f}(t, \mathbf{x}(t)) \quad t \in [t_0, t_f] \tag{24}$$

and the boundary conditions are $\mathbf{x}(t = t_0) = \mathbf{x}_0$ and $\mathbf{x}(t = t_f) = \mathbf{x}_f$. Consistent with discussion in the previous section, using the definition in equation (2), equation (24) is first transformed to a new form as

$$\frac{\mathrm{d}^2\mathbf{x}(\tau)}{\mathrm{d}\tau^2} = \omega_2^2 \mathbf{f}(\omega_1 + \omega_2\tau, \mathbf{x}(\tau)) \equiv \mathbf{g}(\tau, \mathbf{x}(\tau)) \tag{25}$$

Assume the trajectory at the i^{th} iteration is represented by $x^i(\tau)$ and the integrand is approximated by an N^{th} order Chebyshev polynomial

$$\mathbf{g}(\tau, \mathbf{x}^i(\tau)) \approx \sum_{k=0}^{k=N}{}' \mathbf{F}_k T_k(\tau) \tag{26}$$

The solution at the $(i + 1)^{\text{th}}$ step is assumed as $\mathbf{x}^{i+1}(\tau) = \sum_{k=0}^{k=N}{}' \boldsymbol{\beta}_k T_k(\tau)$, and the Picard method provides the iteration form in equation (27) to solve equation (25)

$$\mathbf{x}^{i+1}(\tau)=\mathbf{x}_0 + (\tau+1)\dot{\mathbf{x}}_0 + \int_{-1}^{\tau}\int_{-1}^{s} \mathbf{g}(q, \mathbf{x}^i(q))\, dq\, ds \tag{27}$$

Using the integration properties of Chebyshev polynomials, equation (27) leads to

$$\tfrac{1}{2}\beta_0 T_0(\tau) + \beta_1 T_1(\tau) + \beta_2 T_2(\tau) + \cdots + \beta_N T_N(\tau)$$

$$= x(-1) + (\tau+1)\dot{\mathbf{x}}_0 + \int_{-1}^{\tau}\int_{-1}^{s} (\tfrac{1}{2}\mathbf{F}_0 T_0(q) + \mathbf{F}_1 T_1(q) + \cdots + \mathbf{F}_N T_N(q))\, dq\, ds$$

$$= \zeta_0(\mathbf{F}_0, \mathbf{F}_2, \mathbf{x}_0, \mathbf{x}_f) T_0 + \zeta_1(\mathbf{F}_1, \mathbf{F}_3, \mathbf{x}_0, \mathbf{x}_f) T_1 \tag{28}$$

$$+ \sum_{k=2}^{k=N-2} \zeta_k(\mathbf{F}_k, \mathbf{F}_{k-2}, \mathbf{F}_{k+2}) T_k + \zeta_{N-1}(\mathbf{F}_{N-1}, \mathbf{F}_{N-3}) T_{N-1} + \zeta_N(\mathbf{F}_{N-2}, \mathbf{F}_N) T_N$$

where $\zeta(\cdot)$ denotes functions that are dependent on indicated subsets of the coefficients $\mathbf{f}_k$. Equating the first to N^{th} order coefficients of the Chebyshev polynomials on the left side and on the right side, also including the boundary constraints in equations (9) and (10) lead to the coefficient update formula as

$$\vec{\boldsymbol{\beta}} = \begin{bmatrix} x_f + x_0 - 2(\beta_2 + \beta_4 + \beta_6 + \cdots) \\ \dfrac{x_f - x_0}{2} - (\beta_3 + \beta_5 + \beta_7 + \cdots) \\ \vdots \\ \dfrac{1}{4\times k}\left(\dfrac{F_{k-2} - F_k}{k-1} + \dfrac{F_{k+2} - F_k}{k+1}\right) \\ \vdots \end{bmatrix} \equiv S_B \vec{\mathbf{F}} + \Theta_{xif} \tag{29}$$

The explicit form of S_B is shown in Appendix B, $\vec{\mathbf{F}}$ is the coefficient vector defined as $\vec{\mathbf{F}} = [F_0, F_1, F_2, \ldots, F_N]^{\mathrm{T}} \in R^{N+1}$, and Θ_{xif} is defined by the boundary condition as $\Theta_{xif} = [x_0 + x_f, (x_f - x_0)/2, 0, 0, \ldots, 0]^{\mathrm{T}} \in R^{N+1}$. Equations (19), (29), and (20) lead to the state update equation as

$$\vec{\mathbf{X}}^{k+1} = C_x(S_B\vec{\mathbf{F}} + \Theta_{xif}) = \omega_2^2 C_x S_B \mathrm{TV}\vec{\mathbf{f}} + C_x\Theta_{xif} \equiv \omega_2^2 C_x C_\alpha^B \vec{\mathbf{f}} + C_x\Theta_{xif} \tag{30}$$

where we have defined $C_\alpha^B = S_B\mathrm{TV}$ and $\vec{\mathbf{f}} = [f(\tau_0), f(\tau_1), f(\tau_2), \ldots, f(\tau_N)]^{\mathrm{T}}$. Notice although the second order formula in equation (30) only solves for the position, the velocity solution can be computed conveniently by taking the derivative of the position, which has been obtained in a form of Chebyshev polynomials.

Convergence Analysis of MCPI Methods

One challenge for Chebyshev-Picard methods is when their limited convergence domain does not cover the time interval of interest. This problem has been addressed since the beginning of this avenue of research, but it has not been satisfactorily solved up to today. In their pioneering paper, Clenshaw and Norton correctly concluded that the convergence for the proposed Chebyshev-Picard

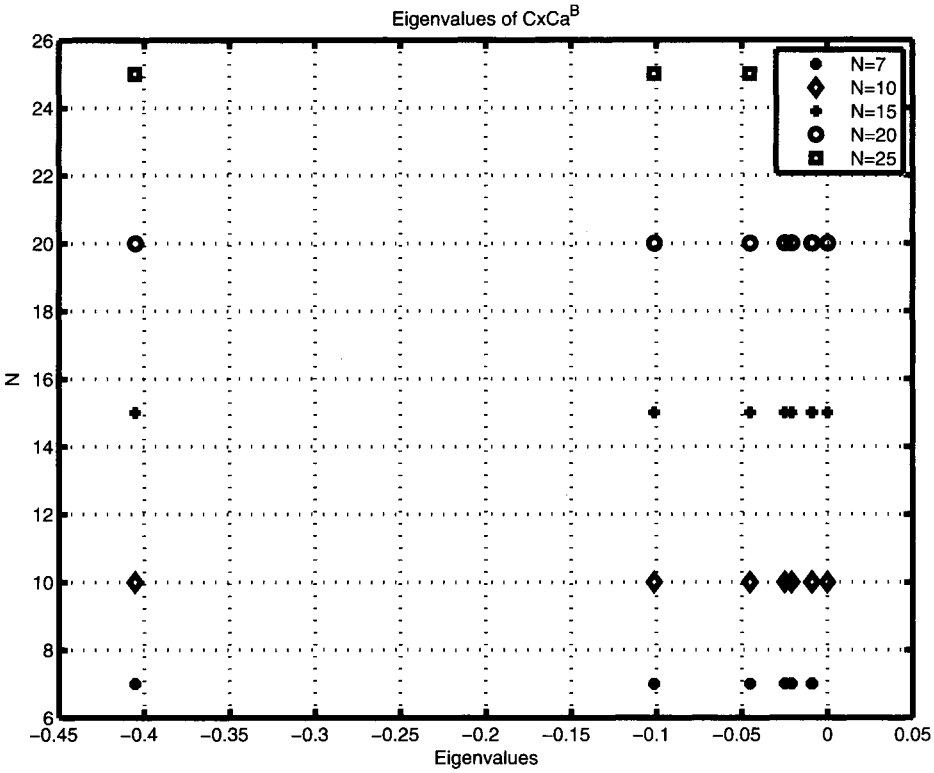

FIG. 2. Eigenvalue of $C_x C_\alpha^B$.

method, which is quite good in many problems, is not generally guaranteed. They suggested additional research to design more powerful methods for solving two-point BVPs [22]. Wright compared a Picard method, a linearization method, and some Taylor-series-based techniques for solving the problems associated with the Chebyshev collocation methods [21]. Establishing the rigorous convergence domain of MCPI methods applicable for general BVPs is not possible by any known approach. To obtain some essential insight we use the explicit formulation in equation (30) to conduct a convergence analysis for the simple linear second order cases. Consider a BVP described by

$$\frac{\mathrm{d}^2 x(t)}{\mathrm{d}t^2} = cx(t) \qquad t \in [t_0, t_f] \quad x(t_0) = x_0 \quad x(t_f) = x_f \tag{31}$$

Equation (30) leads to the iteration formula as

$$\vec{\mathbf{X}}^{k+1} = c\omega_2^2 C_x C_\alpha^B \vec{\mathbf{X}}^k + C_x \Theta_{xif} \qquad \omega_2 = \frac{t_f - t_0}{2} \tag{32}$$

It is known from linear system theory that this sequence is convergent to a fixed point only if all the eigenvalues of the matrix $[c\omega_2^2 C_x C_\alpha^B]$ are within a unit circle (i.e., equation (32) must be a contraction mapping, to converge to a fixed point). The eigenvalues of $C_x C_\alpha^B$ for various N are shown in Fig. 2, revealing the maximum magnitude of eigenvalues of $C_x C_\alpha^B$ is almost invariant about 0.4053 with respect to the Chebyshev order. Thus, the time interval length factor $\omega_2 = (b - a)/2$ and the

linear coefficient c will dictate the convergence of the method. The necessary condition for the convergence is

$$c(t_f - t_0)^2 < \frac{4}{0.4053} \approx 9.87 \tag{33}$$

Although 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 that convergence does not degrade for large N, or put another way, N can be adjusted to achieve high solution precision, without affecting the rate of Picard iteration convergence. We mention that the convergence condition of MCPI methods for the same dynamic system that only has initial time constraints is $c(t_f - t_0)^2 < 4/0.003 \approx 1333$, thus the convergence domain for solving BVPs is about two orders of magnitude smaller than the one for solving IVPs. We emphasize that for solving BVPs, the current MCPI methods are not guaranteed to converge for general time interval length while a new patching methodology is under development.

Numerical Examples

Simulations were done in a conventional PC. The settings of the computer and the development environment used are: Intel(R) Pentium(R) D CPU 3.4 GHz, 3.4 GHz, 2.0 GB of RAM; Windows XP Operating System; MATLAB R2009b. The computational time shown below was the average CPU time from ten running cases. We use second order MCPI in equation (30) to solve the Lambert's problem and use the first order formulas in equations (19) and (21–23) to solve the optimal control problem.

Example 1: Solving a Classical Lambert's Problem

Lambert's problem is a classical two-point BVP of celestial mechanics that was first stated and solved by Johann Heinrich Lambert in 1761. Given two positions at an initial time and a prescribed final time, Lambert's problem is to solve for a terminal initial velocity, using which the generated orbit connects the two known positions within the prescribed time of flight. For general perturbed motion, the generalization of Lambert's problem remains a family of nonlinear problems that must be frequently solved today for orbit transfer. Battin developed an elegant, now classical approach to solve the unperturbed Lambert's problem [25]. However, Battin's method only works for the special case of inverse-square gravity field. Numerical iteration techniques such as shooting methods are the most frequently used current approach to solve the more general perturbed orbit transfer problems.

We use the orbital propagation problem that was studied in references [8, 9] as an example, which has its orbital elements as $a = 6644.754$ km, $e = 0.01$, $i = 68°$, $\Omega = 92°$, $\omega = -160°$, $T_p = 5.3905 \times 10^3$ sec, with dynamic equations

$$\ddot{x} = -\frac{\mu}{r^3}x \qquad \ddot{y} = -\frac{\mu}{r^3}y \qquad \ddot{z} = -\frac{\mu}{r^3}z \qquad r = \sqrt{x^2 + y^2 + z^2} \tag{34}$$

We varied the time of flight from 27 s to 1900 s, corresponding to about 5% to 35% of the orbit, beyond which MCPI methods diverged. The initial position and final position from the F and G solutions [26] were the input variables to both Battin's

method and MCPI methods. The MCPI method approximated the position by using Chebyshev polynomials of order twenty and started the iteration by assuming zero velocity solutions and constant position solutions along all the three dimensions (those constants are the initial positions), thus a very poor starting guess was provided for the MCPI method so that the timing results are very conservative. We have verified that the obtained initial velocity errors for both methods are less than 10^{-8} km/s and their computational time is compared in Fig. 3, where we can see the MCPI method is faster than the Battin's reference solution as long as the time interval is less than ⅓ of the orbit period. Although Battin's method is generally more efficient than MCPI methods for larger time intervals and applicable to arbitrary time of flight, it is important to keep in mind that the same MCPI algorithm routinely solves the Lambert problem with arbitrary perturbations, whereas Battin's approach can only solve the classical inverse square gravity field version of Lambert's problem.

Example 2: An Earth to Apophis Optimal Trajectory Design Problem

A two-dimensional Earth to Apophis interplanetary trajectory is studied. We investigate a hybrid propulsion approach where a constant low thrust propulsion is used during the flight and terminal velocity impulses are available at both the launch time and the rendezvous time. Because of the recent successful applications of the pseudospectral methods [27–29], we also solved the same problem using a pseudospectral method in order to compare its performance with that of MCPI methods. To be consistent, a Chebyshev pseudospectral method [27] was chosen, such that both the MCPI method and the Chebyshev pseudospectral method use 101 CGL nodes as the collocation points. We emphasize the difference between these two methods is fundamental. The pseudospectral method transforms the original problem to a nonlinear programming problem, thus it belongs to the category of the direct approach, whereas the proposed MCPI method transforms the original problem to a two-point BVP by using Pontryagin's principle, thus it is a Chebyshev-Picard variation of the indirect approach. However, in contrast to most existing indirect methods, gradient information is not required for MCPI methods.

The spacecraft is assumed to be a point mass with a variable mass m. As shown in Fig. 4, the position of the spacecraft in the solar-centric polar coordinates is (r, θ), where r is the distance of the spacecraft to the Sun and θ is the true anomaly phase angle with respect to some inertial axis. Moreover, u is the velocity along the radial direction and v is the velocity along the local horizontal direction. The angle between the thrust direction and the local horizontal direction is the control variable represented by β. The dynamic equations for the spacecraft are

$$\dot{r} = u \tag{35}$$

$$\dot{\theta} = v/r \tag{36}$$

$$\dot{u} = v^2/r - \mu/r^2 + T/m \sin\beta \tag{37}$$

$$\dot{v} = -uv/r + T/m \cos\beta \tag{38}$$

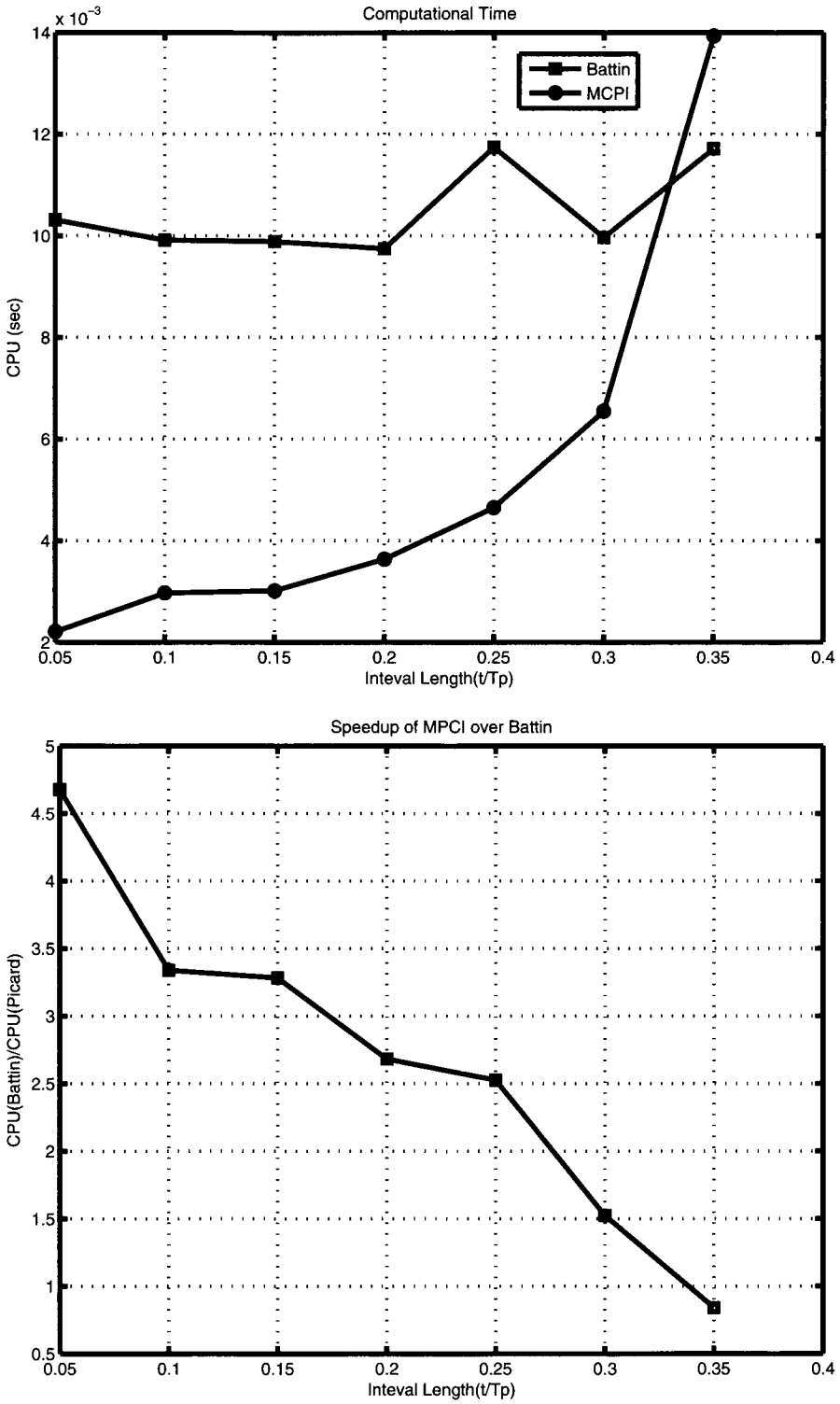

FIG. 3. Computational Time Comparison for Lambert's Problem for Short Arc Orbit Transfer.

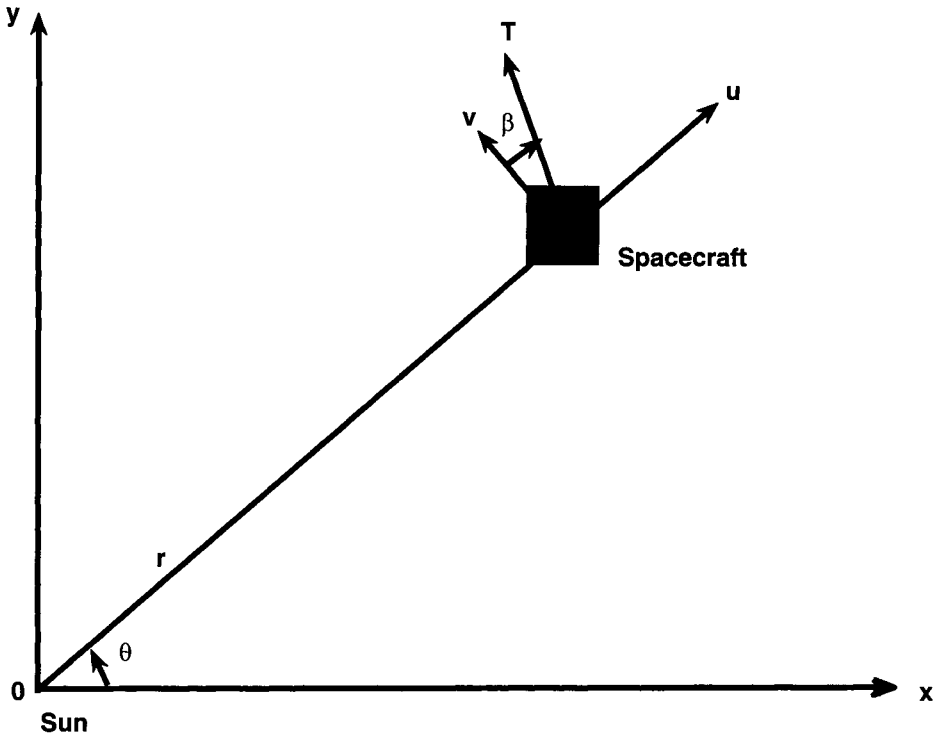

FIG. 4. Frame, State and Control Definition.

where T is a constant thrust. The mass flow rate α is constant and the resulting mass equation is

$$m = m_0 - \alpha(t - t_0) \qquad t_0 \leq t \leq t_f \tag{39}$$

where m_0 is the initial mass, t_0 is the initial time, and t_f is the final time. The thrust magnitude is related to the mass flow rate through

$$T = c\alpha \tag{40}$$

where we choose the specific impulse constant $c = 1.872$ as suggested by Oberle and Taubert [30]. All the distance variables are nondimensionalized by 1.496×10^{11} meters, which is the Earth's distance to the Sun. All the time-related values are nondimensionalized by 5.023×10^6 sec. Subsequently, μ in equation (37) becomes one after the nondimensionalization. The initial mass is nondimensionalized as $m_0 = 1$. We choose the mass flow rate to be $\alpha = 0.007487$, resulting the nondimensionalized thrust as 0.014, which corresponds to $0.05N$ for a 600-kg spacecraft. The position and velocity of the Earth on the launch date and the position and velocity of Apophis on the rendezvous date are the ephemeris data obtained through Jet Propulsion Laboratory online horizon system.[3] The goal of the optimal control is to minimize the total impulse energy at the terminal times, defined as

[3]http://ssd.jpl.nasa.gov/?horizons/.

$$J = \tfrac{1}{2}V_1^2 + \tfrac{1}{2}V_2^2 \tag{41}$$

For the polar reference frame defined in Fig. 4, we have

$$V_1^2 = (u_{s/c}(t_0) - u_E(t_0))^2 + (v_{s/c}(t_0) - v_E(t_0))^2 \tag{42}$$

$$V_2^2 = (u_{s/c}(t_f) - u_A(t_f))^2 + (v_{s/c}(t_f) - v_A(t_f))^2 \tag{43}$$

where u_E and v_E are the velocities of the Earth, and u_A and v_A are the velocities of Apophis. The control variable is the thrust steering angle β and the optimization constraints are the dynamic equations from equation (35) to equation (39).

Introducing co-state variables λ_r, λ_θ, λ_u, λ_v, we define the Hamiltonian as

$$H = \lambda_r u + \frac{\lambda_\theta v}{r} + \lambda_u\left(\frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T}{m\sin\beta}\right) + \lambda_v\left(-\frac{uv}{r} + \frac{T}{m\cos\beta}\right) \tag{44}$$

The co-state equations obtained through Pontryagin's principle are

$$\dot{\lambda}_r = -\lambda_u\left(-\frac{v^2}{r^2} + \frac{2}{r^3}\right) - \frac{\lambda_v uv}{r^2} + \frac{\lambda_\theta v}{r^2} \tag{45}$$

$$\dot{\lambda}_\theta = 0 \tag{46}$$

$$\dot{\lambda}_u = -\lambda_r + \frac{\lambda_v v}{r} \tag{47}$$

$$\dot{\lambda}_v = -\frac{2\lambda_u v}{r} + \frac{\lambda_v u}{r} - \frac{\lambda_\theta}{r} \tag{48}$$

and the optimal thrust direction is

$$\beta = \text{atan } 2(-\lambda_u, -\lambda_v) \tag{49}$$

where atan 2( ) is the four-quadrant inverse tangent function.

With the transversality conditions, we have the following two-point boundary conditions for two states and two co-states

$$r(t_0) = r_E(t_0) \tag{50}$$

$$\theta(t_0) = \theta_E(t_0) \tag{51}$$

$$r(t_f) = r_A(t_f) \tag{52}$$

$$\theta(t_f) = \theta_A(t_f) \tag{53}$$

$$\lambda_u(t_0) = -(u(t_0) - u_E(t_0)) \tag{54}$$

$$\lambda_v(t_0) = -(v(t_0) - v_E(t_0)) \tag{55}$$

$$\lambda_u(t_f) = u(t_f) - u_A(t_f) \tag{56}$$

$$\lambda_v(t_f) = v(t_f) - v_A(t_f) \tag{57}$$

which are maintained by constraining their Chebyshev coefficients through equations (58) to (65) during MCPI iterations. And these two states and co-states are updated through formula in equation (23)

$$\alpha_0^r = r(t_0) + r(t_f) - 2(\alpha_2^r + \alpha_4^r + \alpha_6^r + \cdots) \tag{58}$$

$$\alpha_1^r = \frac{r(t_f) - r(t_0)}{2} - (\alpha_3^r + \alpha_5^r + \alpha_7^r + \cdots) \tag{59}$$

$$\alpha_0^\theta = \theta(t_0) + \theta(t_f) - 2(\alpha_2^\theta + \alpha_4^\theta + \alpha_6^\theta + \cdots) \tag{60}$$

$$\alpha_1^\theta = \frac{\theta(t_f) - \theta(t_0)}{2} - (\alpha_3^\theta + \alpha_5^\theta + \alpha_7^\theta + \cdots) \tag{61}$$

$$\alpha_0^{\lambda_u} = \lambda_u(t_0) + \lambda_u(t_f) - 2(\alpha_2^{\lambda_u} + \alpha_4^{\lambda_u} + \alpha_6^{\lambda_u} + \cdots) \tag{62}$$

$$\alpha_1^{\lambda_u} = \frac{\lambda_u(t_f) - \lambda_u(t_0)}{2} - (\alpha_3^{\lambda_u} + \alpha_5^{\lambda_u} + \alpha_7^{\lambda_u} + \cdots) \tag{63}$$

$$\alpha_0^{\lambda_v} = \lambda_v(t_0) + \lambda_v(t_f) - 2(\alpha_2^{\lambda_v} + \alpha_4^{\lambda_v} + \alpha_6^{\lambda_v} + \cdots) \tag{64}$$

$$\alpha_1^{\lambda_v} = \frac{\lambda_v(t_f) - \lambda_v(t_0)}{2} - (\alpha_3^{\lambda_v} + \alpha_5^{\lambda_v} + \alpha_7^{\lambda_v} + \cdots) \tag{65}$$

In equations (58) to (65), the superscript notation indicates the states or co-states being approximated and the subscript notation represents the order of the coefficient. Additionally, equations (35), (36), (47), and (48) lead to the following initial condition constraint for the other two states and two co-states, which are updated using the formula in equation (21)

$$u(t_0) = \frac{2}{t_f - t_0} \sum_{k=1}^{k=N} \alpha_k^r k^2 (-1)^{(k+1)} \tag{66}$$

$$v(t_0) = \frac{2}{t_f - t_0} r(t_0) \sum_{k=1}^{k=N} \alpha_k^\theta k^2 \tag{67}$$

$$\lambda_r(t_0) = -\frac{2}{t_f - t_0} r(t_0) \sum_{k=1}^{k=N} \alpha_k^{\lambda_u} k^2 + \frac{\lambda_v(t_0) v(t_0)}{r(t_0)} \tag{68}$$

$$\lambda_\theta(t_0) = r(t_0) \left(-\frac{2\lambda_u(t_0) v(t_0)}{r(t_0)} + \frac{\lambda_v(t_0) u(t_0)}{r(t_0)} - \frac{2}{t_f - t_0} \sum_{k=1}^{k=N} \alpha_k^{\lambda_v} k^2 (-1)^{(k+1)} \right) \tag{69}$$

where we have used the following property of the derivative of Chebyshev polynomials $y(\tau) = \sum_{k=0}^{k=N\prime} \alpha_k T_k(\tau)$ as it is evaluated at the left boundary

$$\dot{y}(-1) = \sum_{k=1}^{k=N} \alpha_k k^2 (-1)^{(k+1)} \tag{70}$$

In the simulation, we assume that the spacecraft leaves the sphere of influence of the Earth on April 18, 2012 and rendezvous with Apophis on August 6, 2012: a time of flight of 110 days. The homotopy method presented in reference [31] was first used to find a high precision reference solution, to use as an accuracy check in lieu of the fact this problem does not have an analytical solution. Because the boundary condition violations through the homotopy approach are of the magnitude of 10^{-14}, we use this highly accurate solution to compute the solution errors for the other two methods. The homotopy method took about six CPU seconds to find the solution. For both Chebyshev pseudospectral method and the MCPI method, the starting solutions of the radius and phase angle were chosen as simply the straight lines connecting the specified initial conditions and final conditions; the starting solution estimate for the radial velocity is chosen as a zero vector; the starting estimate of the tangential velocity is computed from the radius information by assuming a circular orbit. For the MCPI method, all the co-state starting solutions were chosen as zero vectors. We found that the initial guess for the Chebyshev pseudospectral method requires user insight or some preliminary trial and error process. To circumvent this issue, the initial guess of the steering angle for the Chebyshev pseudospectral method was chosen as the true solution from the homotopy method. We note that this information would not generally be available for the pseudospectral method, so some other suitable starting guess would be required. This "help" for the pseudospectral method makes our convergence efficiency claims very conservative. The Chebyshev pseudospectral method took about 4.5 s to converge from these starting estimates, where "fmincon" in MATLAB was used as the nonlinear programming optimizer. The termination tolerance on the function value, constraint violation, and states for "fmincon" were set as 10^{-10} ("fmincon" did not converge when setting these parameters stricter than this). The MCPI method used 192 iterations (a relatively large number of iterations were required, because the time interval was near the maximum for Picard iteration convergence over a single solution interval, the maximum eigenvalues of the corresponding linear operator were just barely within the unit circle) but found the solution in only 0.15 s. Thus the speedup of the MCPI method over the Chebyshev pseudospectral method is about 30 for this application. The optimal transfer orbit is shown in Fig. 5(a) and the resulting optimal thrust angles are shown in Fig. 5(b). Figures 5–10 compare the relative errors of the radius, phase angle, radial and tangential velocity, and thrust angle by using the two approaches. More important than the speedup factor of 30, we can see that the state solutions from the MCPI method have six to seven orders of magnitude better accuracy than the Chebyshev pseudospectral method and the control angle obtained from the MCPI method has about eight digits better accuracy than the Chebyshev pseudospectral method. We have verified that the Chebyshev pseudospectral method took about one second to generate solutions with comparable accuracy as MCPI methods if the true state values were used as the initial guess, which in general is

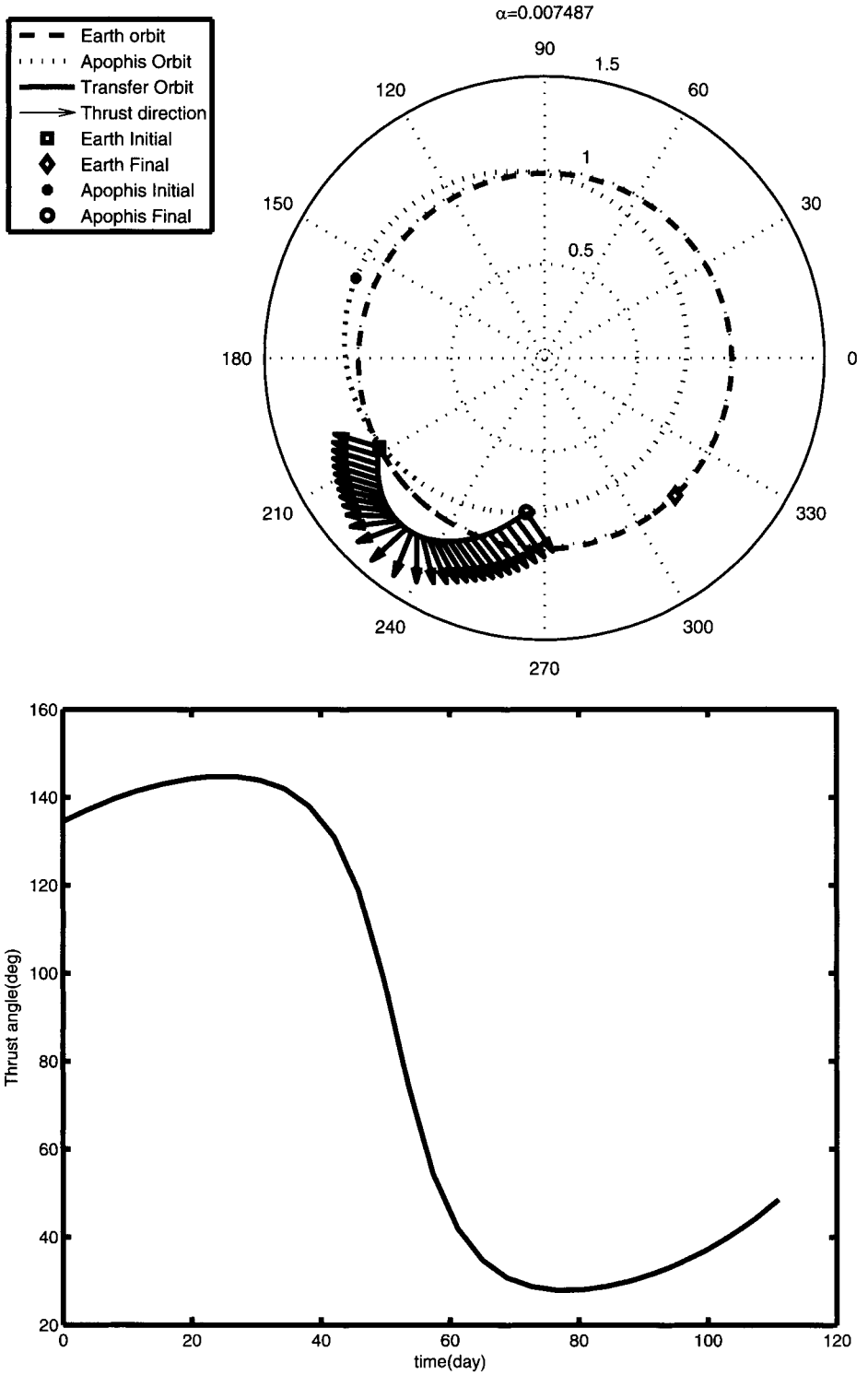

FIG. 5. Transfer Orbit and Control.

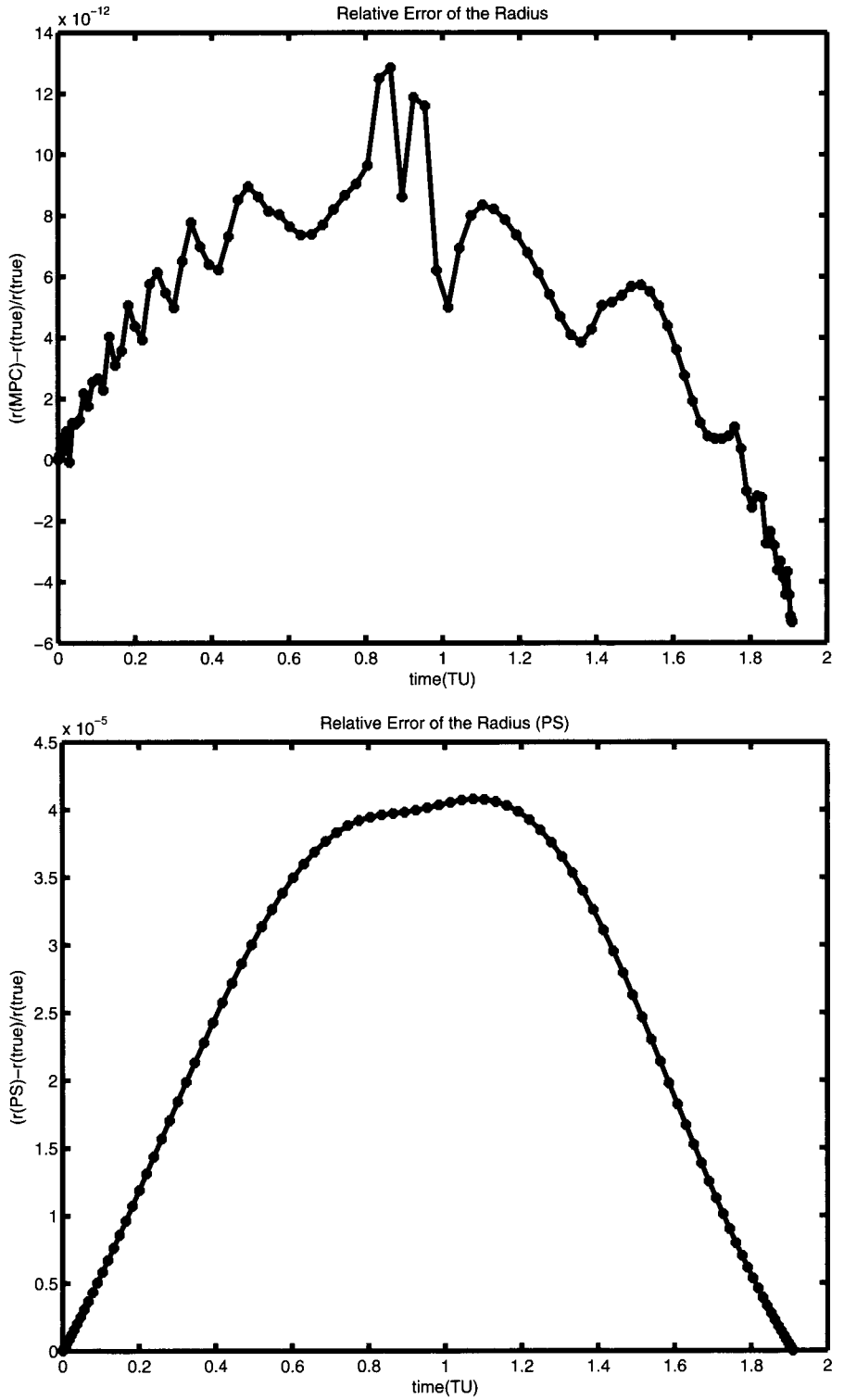

FIG. 6. Solutions Errors of r.

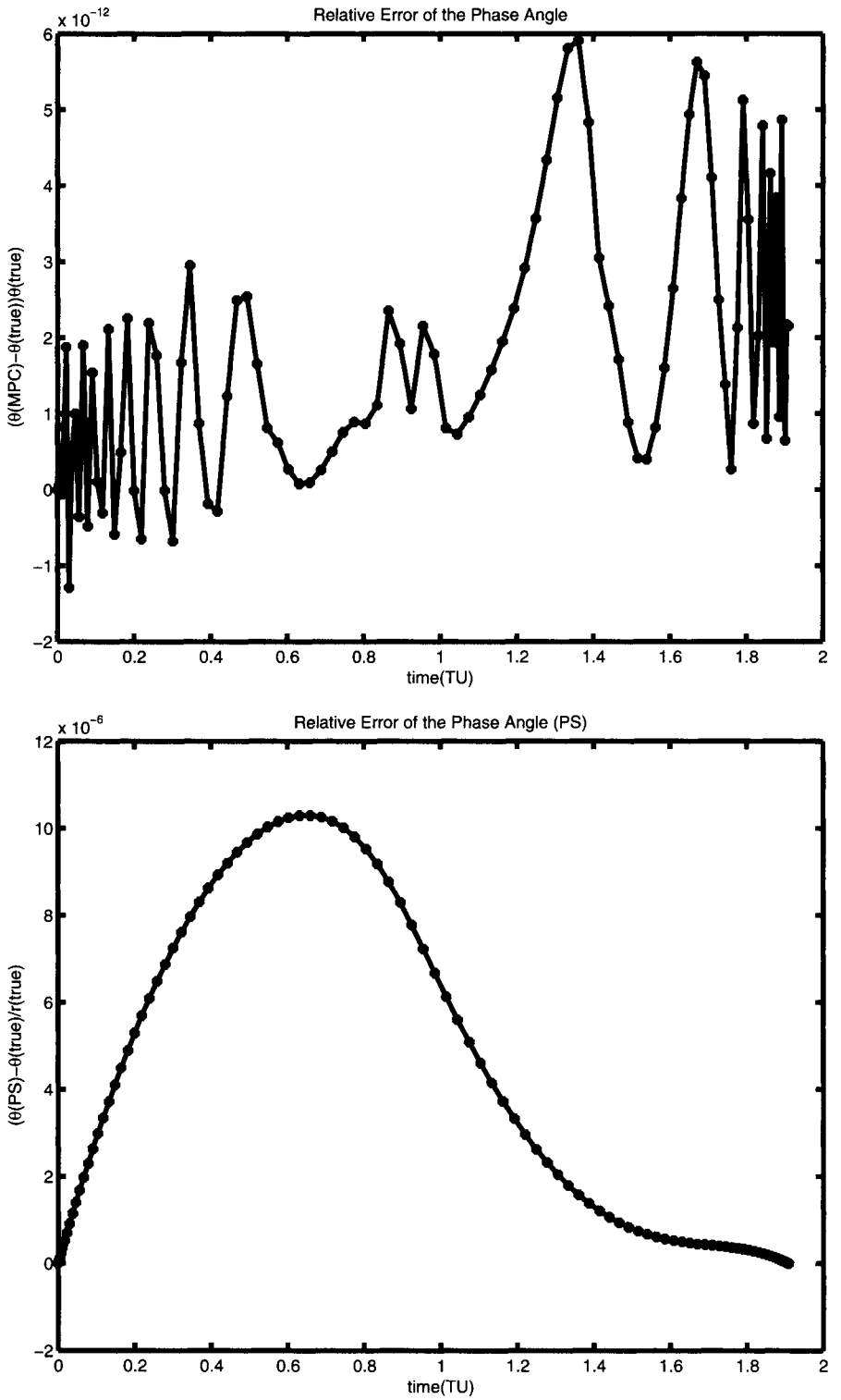

FIG. 7. Solutions Errors of θ.

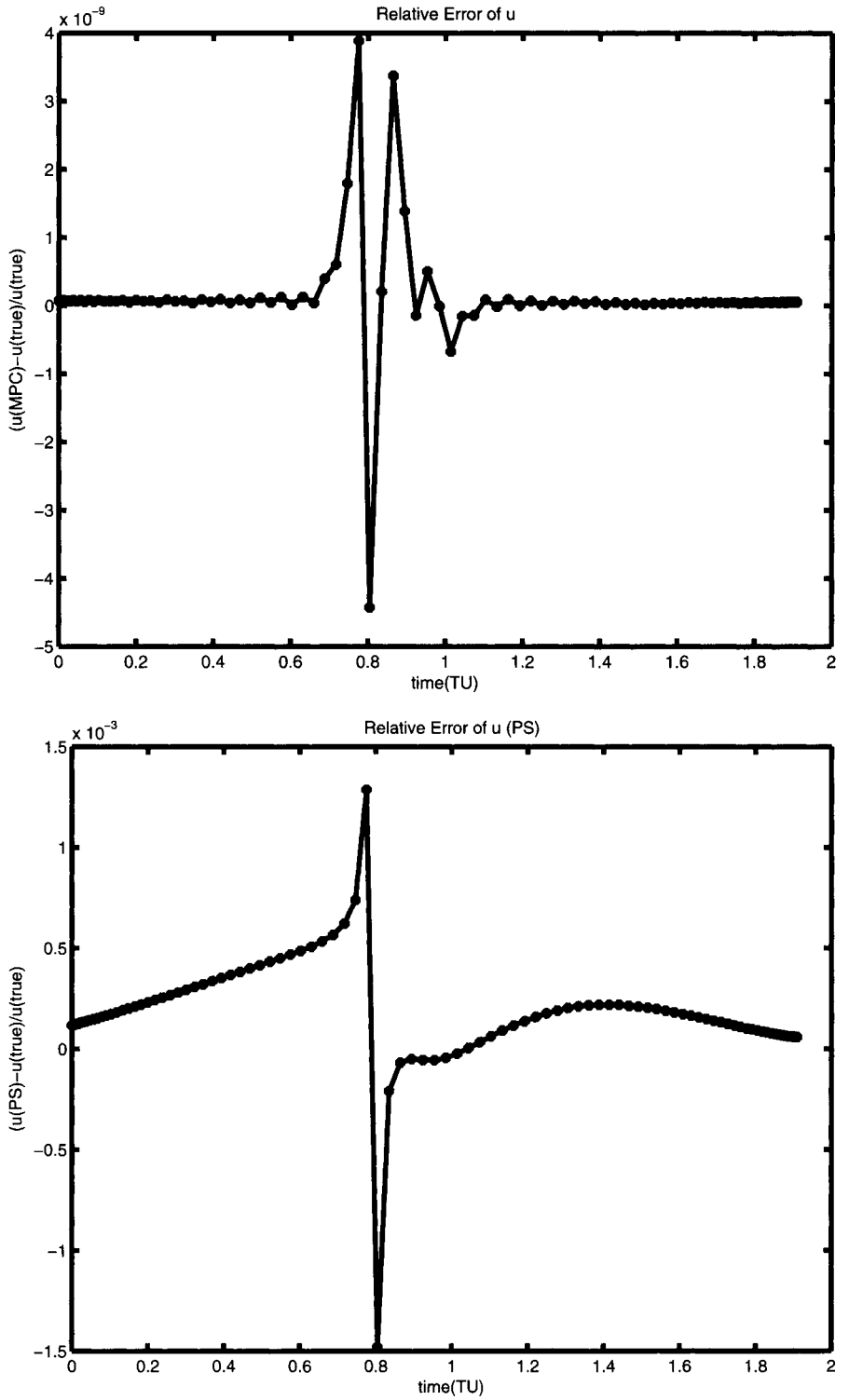

FIG. 8. Solutions Errors of u.

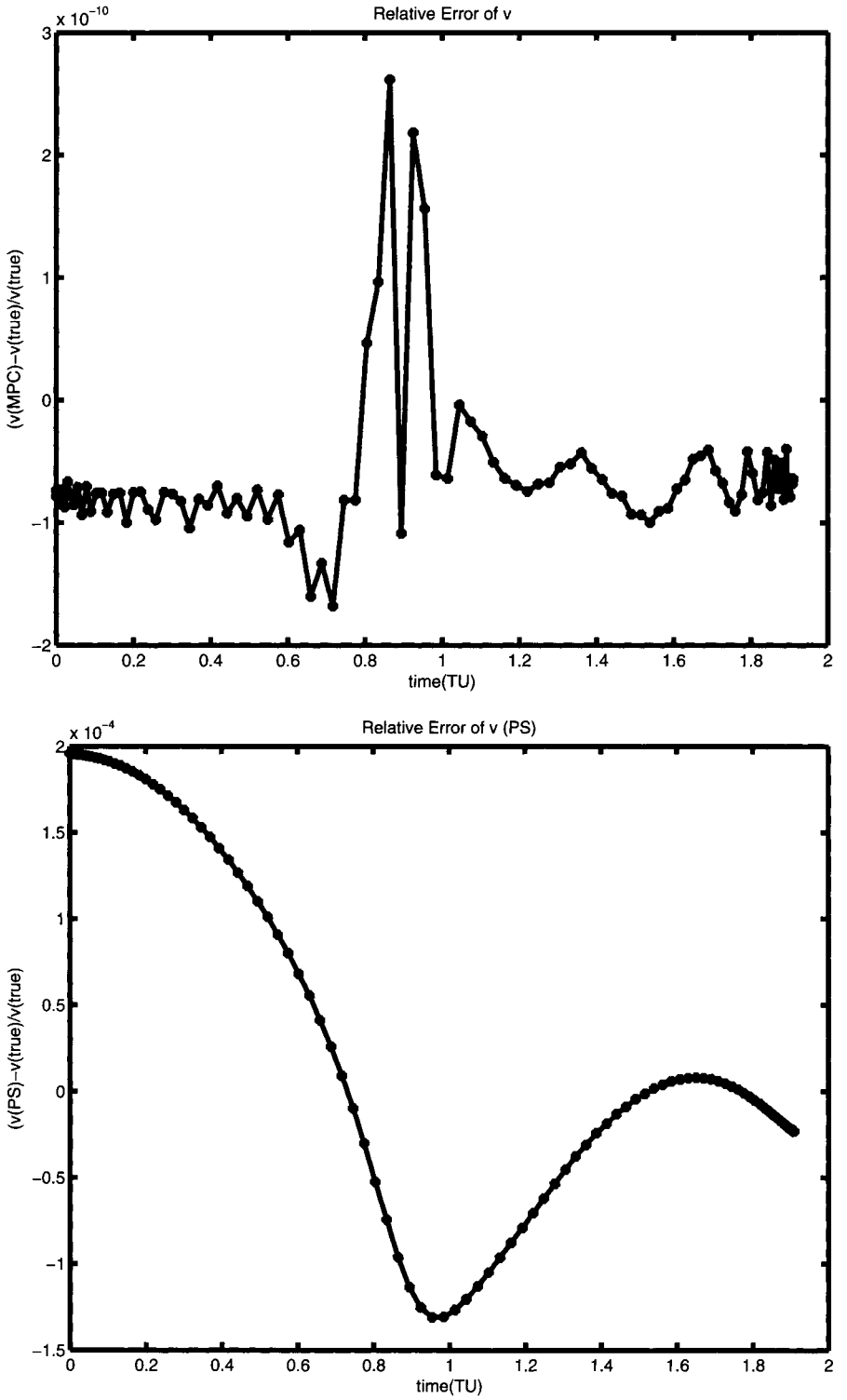

FIG. 9. Solutions Errors of v.

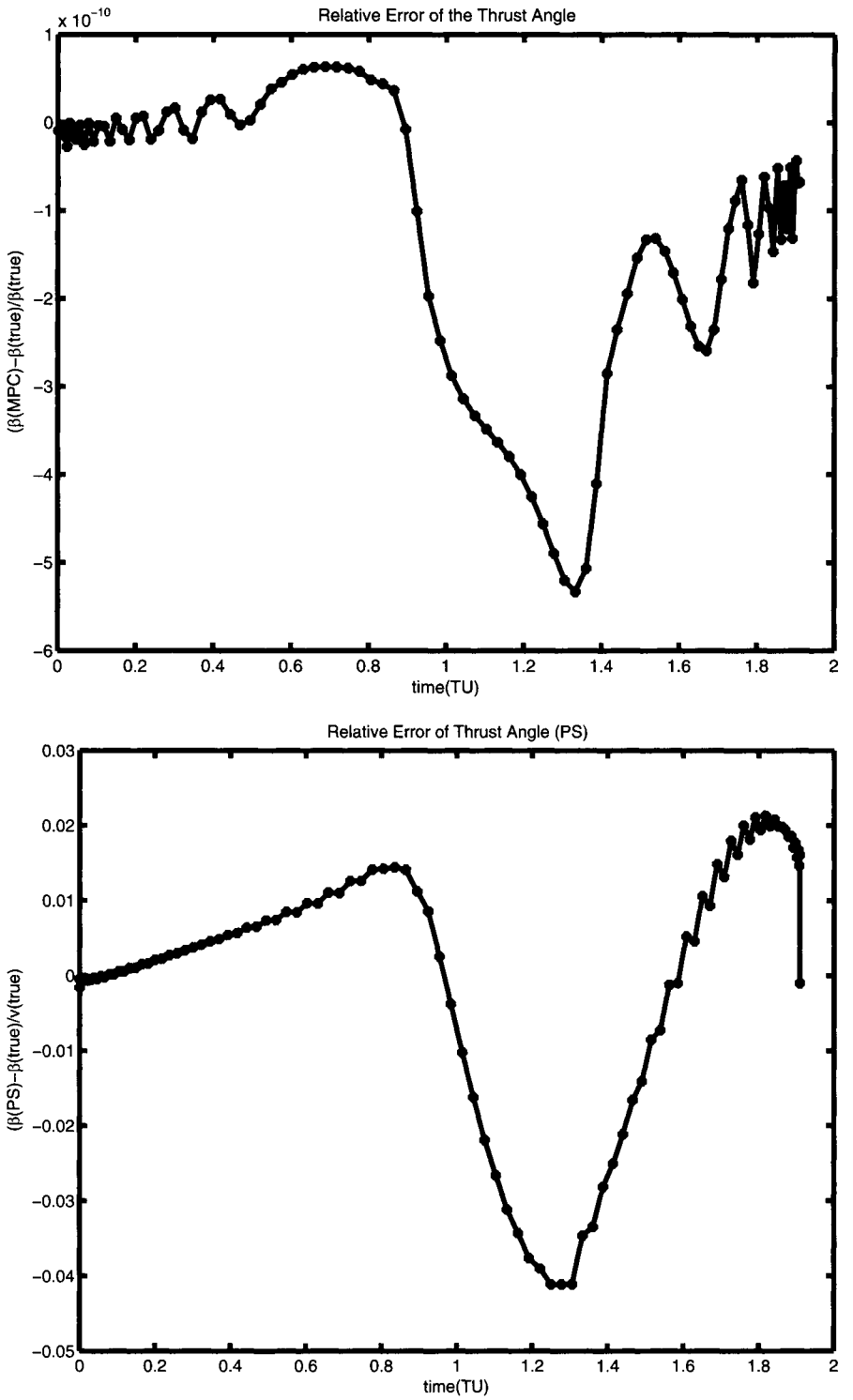

FIG. 10. Solutions Errors of β.

not available. Additionally, we should consider the possibility that using other numerical optimizers, the performance of the Chebyshev pseudospectral method may obtain better results than those shown here. Although a conclusion about the performance comparison of MCPI methods and Chebyshev pseudospectral methods is difficult to draw because Chebyshev pseudospectral methods can resort to various numerical optimizers to solve the nonlinear programming problems, the accuracy and non-sensitive nature to the starting estimate of the proposed MCPI methods shown here are extraordinary. We mention the drawback, at the current state of the methodology, the convergence of the MCPI method for long time of flight is not guaranteed although several approaches have been investigated in Bai's dissertation, where the longest convergent time of flight was found to be 150 days for this problem [9]. Thus the Chebyshev pseudospectral method can solve more general optimal control problems than the MCPI method, at least for the current version of the MCPI method.

Conclusion

In solving boundary value problems with the special cases of optimal control problems, MCPI methods generate solutions in a way significantly different from the usual shooting methods: there are no local Taylor series approximations, gradient computations, or matrix inversions. MCPI solutions also are more accurate than the direct methods because the rigorous Pontryagin necessary conditions are satisfied to a high degree of approximation. However, the current MCPI methods cannot guarantee convergence for arbitrary time interval and also only solve problems with equality constraints. However, the simple structure, computational efficiency, and solution accuracy of MCPI methods demonstrated in solving the classical Lambert's problem and an optimal trajectory design problem are extraordinary. Furthermore, because the level of parallelism for MCPI methods can at least be the Chebyshev polynomials order, large-scale parallel environment can be conveniently utilized, which opens the door to extremely attractive new parallel computing means for solving boundary value problems, especially for the cases where computation of the force functions are time-consuming.

We are currently extending MCPI formulations for solving more general BVPs. Through the concept of a novel shooting method, multiple segments can be patched together for solving long interval problems. We have obtained promising results for the Lambert type of problems where closed-form solutions for the state transition matrix are available. More creative approach is under development since solving general nonlinear BVPs by the patching concept requires gradient information computation, through which the efficiency advantages of MCPI formulation for single interval could not be maintained for arbitrary multiple intervals.

References

[1] BETTS, J.T. "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–20.

[2] LEWIS, F.L. and SYRMOS, V.L. *Optimal Control*. Wiley-Interscience, New York, NY, 1995.

[3] MIGDALASA, A., TORALDO, G., and KUMAR, V. "Nonlinear Optimization and Parallel Computing," *Parallel Computing*, Vol. 29, Apr. 2003, pp. 375–391.

[4] TRAVASSOS, R. and KAUFMAN, H. "Parallel Algorithms for Solving Nonlinear Two-Point Boundary-Value Problems which Arise in Optimal Control," *Journal of Optimization Theory and Applications*, Vol. 30, No. 1, 1980, pp. 53–71.

[5] MIRANKER, W.L. and LINIGER, W. "Parallel Methods for the Numerical Integration of Ordinary Differential Equations," *Mathematics of Computation*, Vol. 21, Jul. 1969, pp. 303–320.

[6] BETTS, J.T. and HUFFMAN, W. "Trajectory Optimization on a Parallel Processor," *Journal of Guidance, Control, and Dynamics,* Vol. 14, Mar.–Apr. 1991, pp. 431–439.

[7] BAI, X. and JUNKINS, J.L. "Solving Initial Value Problems by the Picard-Chebyshev Method with NVIDIA GPUS," presented at 20th Spaceflight Mechanics Meeting, San Diego, CA, Feb., 2010.

[8] BAI, X. and JUNKINS, J.L. "Modified Chebyshev-Picard Iteration Methods for Solution of Initial Value Problems," presented at Kyle T. Alfriend Astrodynamics Symposium, Monterey, CA, May, 2010.

[9] BAI, X. *Modified Chebyshev-Picard Iteration Methods for Solution of Initial Value and Boundary Value Problems*. Ph.D. Dissertation, Texas A&M University, College Station, TX, 2010.

[10] PICARD, E. "Sur l'Application des Methodes d'approximations successives à l'Etude de certaines Equatioins differentielles ordinaires," *Journal de Mathematiques*, Vol. 9, 1893, pp. 217–271.

[11] PICARD, E. *Traité d'analyse*, Vol. 1, ch. 4.7. Gauthier-Villars, Paris, France, Third Ed., 1922.

[12] CRAATS, J.V. "On the Region of Convergence of Picard's Iteration," *ZAMM-Journal of Applied Mathematics and Mechanics*, Vol. 52, Dec. 1971, pp. 487–491.

[13] LETTENMEYER, F. "Ober die von einem Punkt ausgehenden Integralkurven einer Differentialgleichung 2. Ordnung," *Deutsche Math*, Vol. 7, 1944, pp. 56–74.

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

[15] 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, Mar. 1967, pp. 426–433.

[16] 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.

[17] 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.

[18] FOX, L. and PARKER, I.B. *Chebyshev Polynomials in Numerical Analysis*. London, UK: Oxford University Press, 1972.

[19] URABE, M. "An Existence Theorem for Multi-Point Boundary Value Problems," *Funkcialaj Ekvacioj*, Vol. 9, 1966, pp. 43–60.

[20] 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.

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

[22] 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.

[23] 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.

[24] INCE, E.L. *Ordinary Differential Equations*. Dover Publications, Inc, New York, NY, 1956.

[25] BATTIN, R. *An Introduction to the Mathematics and Methods of Astrodynamics*. American Institute of Aeronautics and Astronautics, Inc, Reston, VA, Revised Ed., 1999.

[26] SCHAUB, H. and JUNKINS, J.L. *Analytical Mechanics of Space Systems*. American Institute of Aeronautics and Astronautics, Inc, Reston, VA, First Ed., 2003.

[27] FAHROO, F. and ROSS, I.M. "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, Vol. 25, Jan.–Feb. 2002, pp. 160–166.

[28] 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.

[29] 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.

[30] OBERLE, H.J. and TAUBERT, K. "Existence and Multiple Solutions of the Minimum-Fuel Orbit Transfer Problem," *Journal of Optimization Theory and Applications*, Vol. 95, Nov. 1997, pp. 243–262.

[31] BAI, X., TURNER, J.D., and JUNKINS, J.L. "Optimal Thrust Design of a Mission to Apophis Based on a Homotopy Method," presented at the AAS/AIAA Spaceflight Mechanics Meeting, Savannah, GA, Feb. 2009.

Appendix A: Coefficient Update Matrices

For the states that only have initial condition constraints

$$S_i = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{2}{3} & \frac{1}{4} & -\frac{2}{15} & \cdots & (-1)^{N+1}\frac{1}{N-1} \\ 1 & 0 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{71}$$

where the $r^{\text{th}}(r = 2, 3, \ldots, N-1)$ column of the first row has a form as

$$S_i[1, r] = (-1)^{r+1}\left(\frac{1}{r-1} - \frac{1}{r+1}\right) \tag{72}$$

For the states that only have final condition constraints

$$S_f = \begin{bmatrix} -1 & -\frac{1}{2} & \frac{2}{3} & \frac{1}{4} & \frac{2}{15} & \cdots & -\frac{1}{N-1} \\ 1 & 0 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{73}$$

where the $r^{\text{th}}(r = 3, 4, \ldots, N-1)$ column of the first row has a form as

$$S_f[1, r] = \frac{1}{r-2} - \frac{1}{r} \tag{74}$$

For the states that have initial and final condition constraints, S_{if} has different formulas depending on whether the order N is an even number or an odd number.

When N is an even number

$$S_{if} = \begin{bmatrix} 0 & -\frac{1}{2} & 0 & \frac{1}{2}-\frac{1}{4} & 0 & \cdots & 0 \\ 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3}-\frac{1}{5} & \cdots & \frac{1}{N-1} \\ 0 & 1 & 0 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{75}$$

where the $r^{th}(r = 4, 6, \ldots, N)$ (r is an even number) column of the first row has a form as

$$S_{if}[1, r] = \frac{1}{r-2} - \frac{1}{r} \tag{76}$$

and the other columns of the first row are zeros. The $r^{th}(r = 5, 7, \ldots, N-1)$ (r is an odd number) column of the second row has a form as

$$S_{if}[2, r] = \frac{1}{r-2} - \frac{1}{r} \tag{77}$$

Except $S_{if}[2, N+1] = 1/(N-1)$, all the other columns are zeros.

When N is an odd number

$$S_{if} = \begin{bmatrix} 0 & -\frac{1}{2} & 0 & \frac{1}{2}-\frac{1}{4} & 0 & \cdots & \frac{1}{N-1} \\ 0 & 0 & -\frac{1}{3} & 0 & \frac{1}{3}-\frac{1}{5} & \cdots & 0 \\ 0 & 1 & 0 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{78}$$

where the $r^{th}(r = 4, 6, \ldots, N-1)$ (r is an even number) column of the first row has a form as

$$S_{if}[1, r] = \frac{1}{r-2} - \frac{1}{r} \tag{79}$$

and the other columns of the first row are zeros except $S_{if}[1, N+1] = 1/(N-1)$. The $r^{th}(r = 5, 7, \ldots, N)$ (r is an odd number) column of the second row has a form as

$$S_{if}[2, r] = \frac{1}{r-2} - \frac{1}{r} \tag{80}$$

and all the other columns are zeros.

Appendix B: Second Order Update Matrices S_B

$$S_B(k+1, k-1) = \frac{1/4}{k/(k-1)} \qquad k = 2, 3, \ldots, N-2 \tag{81}$$

$$S_B(k+1, k+1) = -\frac{1/2}{(k^2-1)} \qquad k = 2, 3, \ldots, N-2 \tag{82}$$

$$S_B(k+1, k+3) = \frac{1/4}{k/(k+1)} \qquad k = 2, 3, \ldots, N-2 \tag{83}$$

$$S_B(N, N) = -\frac{1/2}{((N-1)^2 - 1)} \tag{84}$$

$$S_B(N, N-2) = \frac{1/4}{(N-1)/(N-2)} \tag{85}$$

$$S_B(N+1, N+1) = -\frac{1/2}{(N^2 - 1)} \tag{86}$$

$$S_B(N+1, N-1) = \frac{1/4}{N/(N-1)} \tag{87}$$

$$S_B(1, k+1) = -\frac{3(1 + (-1)^k)}{(k^2 - 4)/(k^2 - 1)} \qquad k = 4, 5, \ldots, N-2 \tag{88}$$

$$S_B(2, k+1) = -\frac{3/2(1 - (-1)^k)}{(k^2 - 4)/(k^2 - 1)} \qquad k = 4, 5, \ldots, N-2 \tag{89}$$

$$S_B(1, 1) = -1/4 \tag{90}$$

$$S_B(1, 2) = 0 \tag{91}$$

$$S_B(1, 3) = 7/24 \tag{92}$$

$$S_B(1, 4) = 0 \tag{93}$$

$$S_B(2, 1) = 0 \tag{94}$$

$$S_B(2, 2) = -1/24 \tag{95}$$

$$S_B(2, 3) = -0 \tag{96}$$

$$S_B(2, 4) = 1/20 \tag{97}$$

$$S_B(1, k+1) = \frac{(1 + (-1)^k)/4(k-5)}{(k^2 - 1)/(k-2)} \qquad k = N-1 \tag{98}$$

$$S_B(2, k+1) = \frac{(1 - (-1)^k)/8(k-5)}{(k^2 - 1)/(k-2)} \qquad k = N-1 \tag{99}$$

$$S_B(1, k+1) = \frac{(1 + (-1)^k)/4(k-5)}{(k^2 - 1)/(k-2)} \qquad k = N \tag{100}$$

$$S_B(2, k+1) = \frac{(1 - (-1)^k)/8(k-5)}{(k^2 - 1)/(k-2)} \qquad k = N \tag{101}$$

where $S(i, j)$ represents the element in the i^{th} row and j^{th} column.