JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 42, No. 5, May 2019

Nonlinear Differential Equation Solvers via Adaptive Picard–Chebyshev Iteration: Applications in Astrodynamics

Robyn Woollands* and John L. Junkins†
Texas A&M University, College Station, Texas 77843-3141

DOI: 10.2514/1.G003318

An adaptive self-tuning Picard–Chebyshev numerical integration method is presented for solving initial and boundary value problems by considering high-fidelity perturbed two-body dynamics. The current adaptation technique is self-tuning and adjusts the size of the time interval segments and the number of nodes per segment automatically to achieve near-maximum efficiency. The technique also uses recent insights on local force approximations and adaptive force models that take advantage of the fixed-point nature of the Picard iteration. In addition to developing the adaptive method, an integral quasi-linearization "error feedback" term is introduced that accelerates convergence to a machine precision solution by about a of two. The integral quasi linearization can be implemented for both first- and second-order systems of ordinary differential equations. A discussion is presented regarding the subtle but significant distinction between integral quasi linearization for first-order systems, second-order systems that can be rearranged and integrated in first-order form, and second-order systems that are integrated using a kinematically consistent Picard–Chebyshev iteration in cascade form. The enhanced performance of the current algorithm is demonstrated by solving an important problem in astrodynamics: the perturbed two-body problem for near-Earth orbits. The adaptive algorithm has proven to be more efficient than an eighth-order Gauss–Jackson and a 12th/10th-order Runge–Kutta while maintaining machine precision over several weeks of propagation.

I. Introduction

THE perturbed two-body problem is a fundamental problem in celestial mechanics that can only be solved through the use of numerical integration methods. Historically, the numerical integrators that have been considered for solving this problem can be categorized as either single-step methods or multistep methods. For single-step methods, the state at a specific time $\boldsymbol{x}(t_k)$ is used to compute the state at some future time $\boldsymbol{x}(t_k + h)$ through a linear combination of weighted evaluations of the ordinary differential equation at intermediate times $t_k \leq t \leq t_k + h$. The method is considered "single step" because only information from the current state and forward is used to compute the state at a future time $\boldsymbol{x}(t_k + h)$. Multistep methods differ in that the state at some future time is estimated using the current state value as well as state values at several previous times. Multistep methods are also known as predictor–corrector methods because the value of the state at some future time is extrapolated from previous states in a forward estimation step (predictor step), and this is then refined in a backward estimation step (corrector step) [1]. The predictor–corrector procedure is iterated until the error falls below a certain threshold; at which point, the algorithm proceeds to take the next time step.

The family of explicit Runge–Kutta methods is an example of single-step methods. This family is generally further classified by considering two parameters. The first is the number of "stages," which refers to the number of function evaluations required at each time step; and the second is the order p, which refers to the local truncation error at each step h and matches a local Taylor series expansion with an error of $\mathcal{O}(hp + 1)$ [2]. Butcher developed a classification of Runge–Kutta methods, known as the Butcher tableau, for which he arranged the methods based on the number of stages and their order [3]. Some simple explicit methods use fixed time steps in which the forward prediction time interval is always the same; however, more advanced methods make use of lower- and higher-order algorithms so that the difference in the state predicted by the two algorithms can be used to adaptively adjust the time step. Dormand–Prince RK5(4) and Dormand–Prince RK8(7) are examples of adaptive step-size methods in which the step-size control leads to essentially uniform errors for the duration over which integration is performed [4]. The RK5(4) combines fifth- and fourth-order explicit Runge–Kutta methods and requires seven stages [4], whereas the RK8(7) combines eighth- and seventh-order explicit Runge–Kutta methods and requires 13 stages [5]. Adaptive step-size control is very useful when the local complexity of the ordinary differential equation varies and smaller or larger steps are required to meet the desired accuracy, as well as maximize efficiency. In both cases, the Dormand–Prince methods require an initial guess for the step size in order to start the adaptive algorithm, and several algorithms exist to aid in the selection of the starting step size for general ordinary differential equations [6,7].

Multistep, or predictor–corrector, methods use the current state and a set of back points (previous state values) to predict the state at some future time. The predicted state is then added to the set of back points and a separate corrector algorithm is used to refine the approximation of the future state. The prediction and correction phases of the algorithm are performed by linearly combining forward, backward, and central differences of the ordinary differential equation evaluated at the set of back points until the refined estimate at some future state meets an error criterion. The Gauss–Jackson is an example of a predictor–corrector method, and it has historically been the state of the practice for solution of the perturbed two-body problem in celestial mechanics [8]. In general, predictor–corrector methods use a fixed step size; however, Berry presented a second-order Stormer–Cowell method with variable step size and internal error control [9]. A severe limitation of the predictor–corrector methods is that the initial set of back points must be computed using a "startup procedure" and this can be computationally expensive, especially if a restart is frequently required during the orbit propagation. For highly elliptic orbits, such as a Molniya orbit, the algorithm must either use restarts or a fixed time interval everywhere that is small enough to produce a solution that meets the user-specified tolerance at perigee, where the dynamics are the most nonlinear. An alternative may be to reformulate the differential equations using the Sundman transformation such that the independent variable step is a true/eccentric anomaly instead

Received 18 March 2018; revision received 13 November 2018; accepted for publication 18 November 2018; published online 31 December 2018.

*Adjunct Assistant Professor, Department of Aerospace Engineering, TAMU 3141.

†Distinguished Professor, Department of Aerospace Engineering, TAMU 3141.

of time. Typically, an iterative startup procedure is implemented whereby the back points are determined through a semianalytic approximation or the use of a lower-order numerical method [10].

Historical studies regarding the accuracy and efficiency of various numerical integrators have lead to the conventional wisdom that problems with near-constant step size around the orbit (near-circular orbits) are best solved with multistep methods, such as the Gauss–Jackson; whereas the solution of eccentric orbits benefits more from the adaptive single-step methods, such as the Runge–Kutta [8,11,12]. More recent studies revealed that the newer implicit Runge–Kutta (IRK) algorithms and collocation methods competed with or outperformed the traditional methods in serial mode [13], and parallelization of these methods may result in orders of magnitude speedup with regard to computational cost. IRK methods require the solution of a set of coupled nonlinear differential equations and need an initial guess of the intermediate state values in order for the algorithm to start. Collocation is a numerical method that makes use of orthogonal basis functions to approximate state values subject to known boundary conditions under the constraint that the derivative of the approximation exactly satisfies the ordinary differential equation at a set of prior determined sample points. It can be shown that collocation methods fall under the umbrella of IRK methods, and thus share some positive attributes with regard to stability [1,14,15].

In this paper, we present the adaptive Picard–Chebyshev (APC) numerical integration method, which also falls into the "newer" class of methods, like the IRK and collocation methods mentioned previously; but, it differs, in that the equations are written in integral form and the matrix to be inverted in the least-squares equations is diagonal, thus leading to a trivial and inexpensive computation. We discuss the traditional modified Chebyshev–Picard iteration (MCPI) formulation briefly to establish notation and set the stage for the developments that follow. We then incorporate a recently developed error correction feedback term (integral quasi linearization) that accelerates terminal convergence of the Picard iteration. We follow this with a discussion on the development of an adaptive method for tuning the size of time segments and the number of nodes per segment automatically to achieve near-maximum efficiency while maintaining prescribed precision. We demonstrate the performance of the APC by solving several benchmark problems in astrodynamics, and we compare the results of our adaptive algorithms with those obtained using an eighth-order Gauss–Jackson (GJ8) integrator and a 12th-order Runge–Kutta [RK12(10)] integrator. All three algorithms are coded in C to allow for an efficiency comparison that considers both the number of function evaluations and the computation time to achieve a solution for a specified tolerance. The accuracy of each method is validated by computing the relative error in an energy integral (the Hamiltonian).

Our APC approach embodies enhancements that are not part of the traditional MCPI literature, and it is thus recommended that future users of these algorithms use the following acronyms to identify a particular Picard–Chebyshev algorithm:

1) APC-I represents the adaptive Picard–Chebyshev algorithm for integrating first-order systems.

2) APC-II represents the adaptive Picard–Chebyshev algorithm for integrating second-order systems in cascade form.

3) APC-QI represents the adaptive Picard–Chebyshev algorithm with integral error feedback quasi linearization for first-order systems.

4) APC-QII represents the adaptive Picard–Chebyshev algorithm with integral error feedback quasi linearization for second-order systems in cascade form.

5) APC-QIIG (or simply APC) is the adaptive Picard–Chebyshev algorithm with integral error quasi linearization and local gravity models for propagating near-Earth orbits.

II. Modified Chebyshev–Picard Iteration

The modified Chebyshev–Picard iteration differs from well-known step-by-step explicit single or multistep integrators such as the Gauss–Jackson, the Runge–Kutta–Nystrom, in that it is an iterative path approximation method. Relatively long state trajectory subarcs (segments) can be approximated continuously using the Picard iteration to update these state-space trajectory subarcs on each iteration. The MCPI technique combines the Picard iteration with the orthogonal Chebyshev polynomials. Emile Picard [16] and others observed that any first-order differential equation

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

with an initial condition of $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ could be rearranged as the corresponding integral equation:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau))\,\mathrm{d}\tau \tag{2}$$

A sequence of approximate solutions $\boldsymbol{x}^i(t)$, $(i = 1, 2, 3, \ldots, \infty)$ of the true solution $\boldsymbol{x}(t)$ that satisfies this integral equation may be obtained through the Picard iteration using the following set of approximate paths:

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}^{i-1}(\tau))\,\mathrm{d}\tau, \quad i = 1, 2, \ldots \tag{3}$$

Picard [16] proved that for smooth, differentiable, single valued nonlinear functions $\boldsymbol{f}(t, \boldsymbol{x}(t))$, there was a time interval $|t_f - t_0| < \delta$ and a starting trajectory $\boldsymbol{x}^0(t)$ satisfying $\|\boldsymbol{x}^0(t) - \boldsymbol{x}(t)\|_\infty < \Delta$ that, for suitable finite bounds (δ, Δ), the Picard sequence of trajectories represented a contraction operator that converged to the unique solution of the initial value problem (IVP). These convergence bounds are typically quite large in astrodynamics, in which δ can be up to three low-Earth-orbit periods and Δ is sufficiently large enough to allow $\boldsymbol{x}^0(t)$ to be a (very poor) "cold start," with the given initial conditions held constant over the orbit [17]. For this cold start case, the classical unaccelerated algorithm will take over 20 path iterations to achieve a high-precision orbit. The rate of convergence is geometric and, although this is generally slower in the end game than quadratic convergence, the path approximation nature of the MCPI exhibits several unique characteristics that, if leveraged (as we have done herein, and is shown in Refs. [18,19]), lead to significantly improved efficiency. These attributes, especially the large convergence domain of the Picard iteration, set the method apart from other numerical integration approaches.

A finite set of orthogonal Chebyshev polynomials is used as the basis function set to approximate the integrand along the available approximate trajectory $\boldsymbol{x}^{i-1}$. That is,

$$\boldsymbol{f}(\tau, \boldsymbol{x}^{i-1}(\tau)) = \sum_{k=0}^{N-1} \boldsymbol{a}_k T_k(\tau) \tag{4}$$

where $\boldsymbol{a}_k$ are the coefficients of the fit of $\boldsymbol{f}(\tau, \boldsymbol{x}^{i-1}(\tau))$, and $T_k(\tau)$ are the Chebyshev polynomials with the forward and reverse time transformations of $\tau = -1 + 2(t - t_0)/(t_f - t_0)$ and $t = t_0 + (\tau + 1)(t_f - t_0)/2$. A key feature of the MCPI is the use of Chebyshev–Gauss–Lobatto samples, which are a nonuniform cosine distribution of samples over the transformed time domain $\{-1 \le \tau \le 1\}$:

$$\tau_j = -\cos\left(\frac{j\pi}{N}\right), \qquad j = 0, 1, 2, \ldots, N \tag{5}$$

The cosine set of nodes has high nodal density near the ± 1 domain boundaries that compensates and approximately minimizes the Runge phenomena. Note that the cosine nodes are the extrema of the $T_k(\tau)$, which are also the zeros of

$$U_k(\tau) = \frac{\mathrm{d}}{\mathrm{d}\tau} T_k(\tau)$$

plus the end points of the interval. Alternatively, one may use the zeros of $T_k(\tau)$ as nodes. As shown in Ref. [20], the cosine nodes of Eq. (5) are slightly preferred to the zeroes of $T_k(\tau)$ when accurate boundary conditions are required at the ends of the interval. The coefficients $\boldsymbol{a}_k$ that linearly combine the Chebyshev basis functions are approximated by the method of least squares. Linear least squares generally requires a matrix inversion; however, a consistent choice of basis functions, weights, and node locations ensures orthogonality and leads to a trivial inversion; the coefficients reduce to simple ratios of discrete inner products. Refer to Appendix A for the MCPI vector–matrix derivations of the IVP first- and second-order systems and the two-point boundary value problems (TPBVPs).

For sufficiently large N, the Chebyshev polynomials approach a complete set; thus, near-machine precision approximation of the integrand can be achieved. Because the integration of a Chebyshev series can be performed term-by-term analytically, the integral can also be written as a Chebyshev series with near-machine precision. The free constants of integration can be chosen to enforce any combination of n initial and final boundary conditions. The state vector being represented by a Chebyshev series provides an inbuilt interpolation function to compute the state at any instant and to precisely solve various times of interest (perigee times, conjunction times, etc.). In a piecewise continuous fashion, converged solutions along successive time segments are patched together (head to tail), and so an efficient and accurately converged piecewise approximation of the state trajectory is obtained over arbitrary large time intervals. As with any method for numerically propagating long-term motion, the issue of secular accumulation of arithmetic errors will arise. Long-term orbit propagations (up to several weeks) with even highly eccentric orbits such as the Molniya orbit indicate the stability of the resulting algorithm is competitive with the state of the art. As an example, with 15-digit arithmetic, Molniya orbits are computed with over 12-digit precision for seven weeks of propagation, with a high-fidelity force model and while showing significant speedup as compared to the step-by-step explicit and multistep predictor–corrector algorithms.

The original fusion of the orthogonal approximation theory and the Picard iteration was introduced by Clenshaw [21], and Clenshaw and Curtis [22], and Clenshaw and Norton in 1963 [23]. Nine years later, Feagin published a dissertation [24] extending Clenshaw and Norton's work [23] to establish the first vector–matrix version of the Picard iteration using orthogonal basis functions [25]. In 1980, Shaver wrote a related dissertation, giving new insights on parallel computation using the Picard iteration and the Chebyshev approximation [26]. In 1997, Fukushima [27] also addressed parallelization of the Picard iteration; however, his particular software and hardware implementations did not give the anticipated theoretical speedup.

More than a decade later, Bai and Junkins revisited this approach and developed improved algorithms for solving IVPs and TPBVPs [17,28]. They established new convergence insights and developed alternate vector–matrix formulations for solving initial and boundary value problems. These were published in Bai's Ph.D. dissertation [29]. Bani Younes and Junkins followed this work to establish methods that efficiently accommodated high-order gravity perturbations, including local gravity models, to represent realistic perturbed gravity acting on near-Earth satellites more efficiently than globally convergent gravity models [30]. Macomber et al. developed further enhancements, taking advantage of the "fixed-point" convergence nature of the MCPI by using local force models and radially adaptive gravity approximations [18,19]. They also made use of improved warm and hot starts to reduce Picard iterations without accuracy loss. We mentioned a number of additional developments that have been introduced to accommodate second-order differential equations and two-point boundary value problem solutions, as well as to further enhance vector–matrix forms of the algorithms [19,29,31,32]. These enhancements result in further efficiency increases: all of which informed the developments in this paper.

III. Integral Error Feedback Quasi Linearization

We now present an approach for acceleration of the Picard iteration during the terminal quasi-linear iterations by including an error feedback term that significantly enhances convergence. This formulation is closely related to that proposed by Wang et al. [33] for integrating equations in first-order form. However, our adaptation methods and local force model techniques result in a self-starting, self-tuning, and computationally efficient approach optimized for astrodynamics applications. By demonstration, we show the utility of these algorithms for use with high-fidelity gravitational field models and show the robustness, efficiency, and stability with regard to extreme variations in orbit eccentricity and force model fidelity. More important, in this paper, we extend the ideas to develop an algorithm that is compatible with the second-order Picard–Chebyshev cascade formulation. The cascade formulation is similar to the second-order formulation commonly used in IRK methods [34]. We discuss the subtle but significant distinction between the integral quasi-linearization formulation for systems that are naturally first order, systems that are naturally second order but can be numerically integrated in first-order form, and systems that are naturally second order and are integrated using the kinematically consistent Picard–Chebyshev cascade formulation. The latter approach turns out to be a significant improvement as compared to rewriting the n second-order equations as $2n$ first-order equations and applying the first-order algorithm to the second-order system.

A. First-Order Picard–Chebyshev Algorithm with Integral Error Feedback

Consider a system of first-order differential equations given by

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad t \in [t_0, t_f] \tag{6}$$

with an initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$. This can be rewritten in integral form as

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau))\,\mathrm{d}\tau \tag{7}$$

The equation error is computed as

$$\boldsymbol{e}(\boldsymbol{x}(t)) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau))\,\mathrm{d}\tau - \boldsymbol{x}(t) \tag{8}$$

where $\boldsymbol{e}(\boldsymbol{x}(t))$ should be zero, to within the specified convergence tolerance, at the final iteration. Said another way, if $\boldsymbol{x}(t)$ is the exact solution, then $\boldsymbol{e}(\boldsymbol{x}(t))$ is clearly zero for all t. Recall the Picard iteration update, computed as

$$\tilde{\boldsymbol{x}}^i(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}^{i-1}(\tau))\,\mathrm{d}\tau \quad i = 1, 2, 3, \ldots \tag{9}$$

Let us consider small variations $\Delta\boldsymbol{x}(t)$ about the best current estimated trajectory $\tilde{\boldsymbol{x}}^i(t)$ such that the true trajectory satisfies

$$\boldsymbol{x}(t) = \tilde{\boldsymbol{x}}^i(t) + \Delta\boldsymbol{x}(t) \tag{10}$$

Therefore, from Eq. (8), the equation error to the first order in $\Delta\boldsymbol{x}(t)$ is as follows:

$$\boldsymbol{e}(\boldsymbol{x}(t)) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(\tau, \boldsymbol{x}(\tau))\,\mathrm{d}\tau - \boldsymbol{x}(t) \approx \boldsymbol{x}(t_0) + \int_{t_0}^{t} \Big[\boldsymbol{f}(\tau, \tilde{\boldsymbol{x}}^i(\tau)) + \boldsymbol{J}(\tau, \tilde{\boldsymbol{x}}^i(\tau))\Delta\boldsymbol{x}(\tau)\Big]\,\mathrm{d}\tau - \tilde{\boldsymbol{x}}^i(t) - \Delta\boldsymbol{x}(t) \tag{11}$$

Requiring the correction $\Delta\boldsymbol{x}(t)$ in the rightmost form of Eq. (11) to make $\boldsymbol{e}(\boldsymbol{x}(t))$ vanish to the first order gives

$$\Delta \boldsymbol{x}(t) = \int_{t_0}^{t} \Big[\boldsymbol{J}(\tau, \tilde{\boldsymbol{x}}^i(\tau))\Big]\Delta \boldsymbol{x}(\tau)\,\mathrm{d}\tau + \mathcal{O}(\Delta \boldsymbol{x}(t))^2 \tag{12}$$

Making use of Eq. (11), and estimating the best available approximation of the solution error as $\Delta \boldsymbol{x}(t) = \tilde{\boldsymbol{x}}^i(t) - \boldsymbol{x}^{i-1}(t) + \mathcal{O}(\Delta \boldsymbol{x}(t))^2$, gives rise to the following integral feedback update form that corrects the Picard iteration update of Eq. (9) such that the equation error is zero to the first order:

$$\boldsymbol{x}^i(t) = \tilde{\boldsymbol{x}}^i(t) + \int_{t_0}^{t} \Big\{\Big[\boldsymbol{J}(\tau, \tilde{\boldsymbol{x}}^i(\tau))\Big]\Big[\tilde{\boldsymbol{x}}^i(t) - \boldsymbol{x}^{i-1}(\tau)\Big]\Big\}\,\mathrm{d}\tau \tag{13}$$

where $\boldsymbol{x}^i(t)$ is the corrected trajectory for the next iteration, $\tilde{\boldsymbol{x}}^i(t)$ is the uncorrected Picard iterative trajectory at the current iteration, and

$$\Big[\boldsymbol{J}(\tau, \tilde{\boldsymbol{x}}^i(\tau))\Big] \equiv \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \tau)}{\partial \boldsymbol{x}}\right|_{\tilde{\boldsymbol{x}}^i(\tau)}$$

is the local force Jacobian. We note that an alternate development linearizes about $\boldsymbol{x}^{i-1}(t)$ instead of $\tilde{\boldsymbol{x}}^i(t)$, and this alternate approach leads to the same feedback correction to within $\mathcal{O}(\Delta \boldsymbol{x}(t))^2$. More important, because $\Delta \boldsymbol{x}(t) \approx \tilde{\boldsymbol{x}}^i(t) - \boldsymbol{x}^{i-1}(t)$ is a small quantity, the integral term is $\mathcal{O}(\Delta \boldsymbol{x}(t))$ and we can accept approximations that neglect $\mathcal{O}(\Delta \boldsymbol{x}(t))^2$ in computing the integrand of Eq. (13). We also mention that Eq. (13) bears an important connection to the historical works of Bellman and Kalaba [35]. The integral error correction term is an integral form of Bellman and Kalaba's famous quasi-linearization algorithm [35]. Figure 1 shows a significant reduction in the number of iterations required for integrating first-order differential equations with and without the quasi-linearization technique.

B. Second-Order Picard–Chebyshev Algorithm with Integral Error Feedback

We now augment the aforementioned developments for the case of a second-order system. We do not simply recast n second-order equations as $2n$ first-order state-space form equations and apply the results of the previous section because a simple yet subtle issue arises. Consider a second-order system of differential equations given by

$$\dot{\boldsymbol{x}} = \boldsymbol{v}, \qquad \boldsymbol{x}_0 = \boldsymbol{x}(t_0) \tag{14}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{v}), \qquad \boldsymbol{v}_0 = \boldsymbol{v}(t_0) \tag{15}$$

The classical Picard iteration cascade formulation for the velocity is given by

$$\boldsymbol{v}^i(t) = \boldsymbol{v}_0 + \int_{t_0}^{t} \boldsymbol{f}\Big(s, \boldsymbol{x}^{i-1}(s), \boldsymbol{v}^{i-1}(s)\Big)\,\mathrm{d}s \tag{16}$$

and we form the updated position trajectory as

$$\boldsymbol{x}^i(t) = \boldsymbol{x}_0 + \int_{t_0}^{t} \boldsymbol{v}^i(s)\,\mathrm{d}s \tag{17}$$

Observe that only the velocity update of Eq. (16) is of the Picard form [the (i)th velocity history is obtained from an acceleration path integral along the $(i-1)$th state trajectory], whereas the (i)th position trajectory of Eq. (17) is obtained by simply analytically integrating the (i)th velocity approximation. Remarkably, this kinematically con*sistent* cascade structure of the computations requires about half as many iterations as integrating a naturally second-order system in first-order form [36]. Also note that the cascade formulation is called kinematically consistent because analytical integration of the velocity solution $\boldsymbol{v}^i(t)$ gives the position $\boldsymbol{x}^i(t)$ (i.e., the current position approximation is the exact integral of the current velocity approximation rather than the integral of the previous velocity approximation). More details may be found in Ref. [32].

The departure motion for this second-order system is written in cascade form:

$$\Delta \dot{\boldsymbol{x}} = \Delta \boldsymbol{v} \tag{18}$$

$$\Delta \dot{\boldsymbol{v}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\Delta \boldsymbol{x} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}}\Delta \boldsymbol{v} \tag{19}$$

Any approximate solution to the system $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}})$ is given by

$$\boldsymbol{x} = \tilde{\boldsymbol{x}} + \Delta \boldsymbol{x} \tag{20}$$

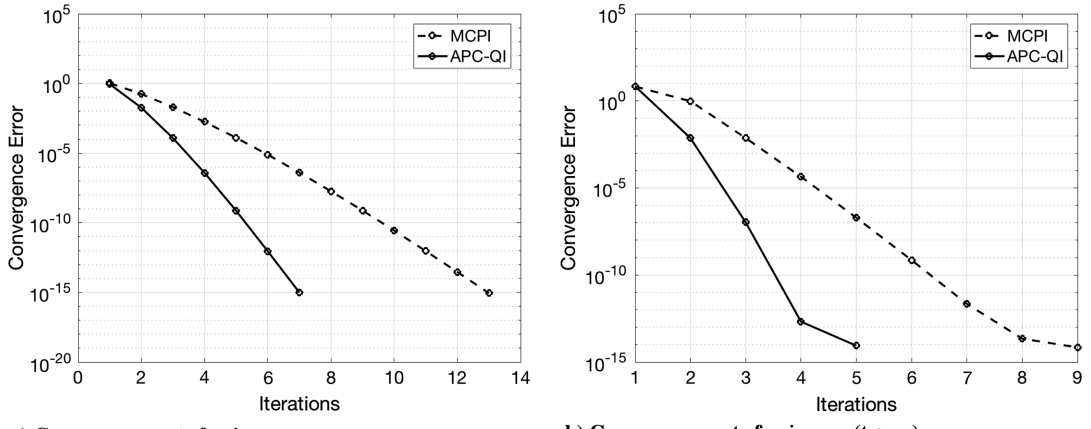
$$\boldsymbol{v} = \tilde{\boldsymbol{v}} + \Delta \boldsymbol{v} \tag{21}$$

The $\Delta \boldsymbol{x}(t)$ departure motion in cascade integral form is

$$\Delta \boldsymbol{x}(t) = \Delta \boldsymbol{x}_0 + \int_{t_0}^{t} \Delta \boldsymbol{v}(s)\,\mathrm{d}s \tag{22}$$

$$\Delta \boldsymbol{v}(t) = \Delta \boldsymbol{v}_0 + \int_{t_0}^{t} \left[\frac{\partial \boldsymbol{f}(s, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}})}{\partial \boldsymbol{x}}\Delta \boldsymbol{x}(s) + \frac{\partial \boldsymbol{f}(s, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{v}})}{\partial \boldsymbol{v}}\Delta \boldsymbol{v}(s)\right]\mathrm{d}s \tag{23}$$

Let $\tilde{\boldsymbol{x}}(t) = \boldsymbol{x}^i(t)$ and $\tilde{\boldsymbol{v}}(t) = \boldsymbol{v}^i(t)$. The velocity correction at each iteration is



a) Convergence rate for $\dot{x} = -\epsilon x$ **b) Convergence rate for $\dot{x} = \cos(t + \epsilon x)$**

Fig. 1 Accelerated vs regular Picard–Chebyshev convergence rate comparison for two first-order systems with known analytical solutions.

$$\boldsymbol{v}^i(t) = \tilde{\boldsymbol{v}}^i + \int_{t_0}^{t}\left[\frac{\partial \boldsymbol{f}(s,\boldsymbol{x}^{i-1}(s),\boldsymbol{v}^{i-1}(s))}{\partial \boldsymbol{x}}\left(\tilde{\boldsymbol{x}}^i(s)-\boldsymbol{x}^{i-1}(s)\right) + \frac{\partial \boldsymbol{f}(s,\boldsymbol{x}^{i-1}(s),\boldsymbol{v}^{i-1}(s))}{\partial \boldsymbol{v}}\left(\tilde{\boldsymbol{v}}^i(s)-\boldsymbol{v}^{i-1}(s)\right)\right]\mathrm{d}s \tag{24}$$

Now, the position at the current iteration may be computed as

$$\boldsymbol{x}^i(t) = \boldsymbol{x}_0 + \int_{t_0}^{t}\boldsymbol{v}^i(s)\,\mathrm{d}s \tag{25}$$

We reemphasize that, in this second-order cascade approach, all Picardlike approximations are at the velocity level, and so only half of the approximation coefficients are obtained from integrating a Chebyshev fit of acceleration along the prior approximation path: the remaining (position) half of the state trajectory approximation.

For the orbit problem, when using complicated force models, the distinct elements of the full Jacobian (symmetric for conservative systems) are expensive to rigorously derive and compute. Typically, we do not need more than four accurate digits in computing $\boldsymbol{J}$ because the integral correction term in Eq. (24) is itself recursively smaller, contributing to the rightmost few digits of $\boldsymbol{v}^i(t)$, and is ultimately vanishingly small in the terminal convergence region when corrections of $\mathcal{O}(\Delta\boldsymbol{v}(t))$ approach the desired tolerance. In the case of near-Earth orbits, we have found that the algebraically simple two-body approximation of $\boldsymbol{J}$ (i.e., the two-body gravity gradient), evaluated using the local state along the trajectory from the previous iteration, provides more than sufficient accuracy to enjoy the speedup implicit in Eq. (24) with greatly reduced computation. We note that, even with the high-fidelity gravity models in low Earth orbit (LEO), the two-body approximation of $\boldsymbol{J}$ agrees everywhere with the actual Jacobian with errors in the fourth digit. With reference to Wang et al. [33] and based on our own computations, some of which are reported in the following, we find that the integral feedback significantly accelerates terminal convergence of the Picard iteration. In this paper, we also use this accelerated algorithm in conjunction with several other recent enhancements of Picard–Chebyshev methods. These enhancements when taken together with methods for adapting the segment size and degree of approximation lead to an efficient, accurate, and adaptive method for solving the fundamental differential equations in astrodynamics with a physically meaningful high-degree and -order gravity field.

The convergence curves shown in Figs. 2c and 2d are for one of three orbit segments. The terminal times of the segment break points correspond to three equal $(2\pi/3)$ increments of true anomaly. The symbols along the convergence curves represent iterations in which the equations of motion are integrated using different fidelity force models. The square represents the Picard iterations of the two-body plus zonal gravity perturbations (J2 to J6) being included in the force model. The triangle represents the iterations each using the full fidelity force model (specified by the user), and the circles (which are denoted as "approx" in the legend) represent the iterations using a force model that includes two-body plus zonals plus the truncated Taylor series correction of the difference between the full and zonal force models on the previous iteration. The approx model is simply a low-degree Taylor series correction (in this case, just the first term) at each approximation node. More details on the variable-fidelity force model may be found in Ref. [18], and they are discussed later in the Picard Enhancement section (Sec. V). For now, it is sufficient to note that the full force model computational cost is over one order of magnitude more expensive than the J2 to J6 and approximate local force models. We note that the use of the less expensive force models is done in a way that does not degrade the accuracy of the final converged orbit; however, these local models substantially reduce the computational cost of Picard iterations.

In all cases, it is clear that the APC outperforms the MCPI by reducing by about one-third the number of iterations to reach a near-machine precision solution. However, the most expensive part of the computational procedure is the full force evaluations, and so the advantages of the accelerated algorithm over the traditional algorithm are partially mitigated because only one additional full force evaluation is required at each node. Note that, for the LEO case, the MCPI requires four full force model evaluations at each node; whereas the APC requires three full force model evaluations at each node for achieving the same solution accuracy. For the medium-Earth-orbit (MEO) cases in which the gravity gradients are much weaker during most of each orbit, the MCPI requires three full force evaluations, whereas the APC requires only two. For the best efficiency when solving a system of second-order differential equations, one should use the cascade algorithm with the error feedback correction term in conjunction with local force approximations.

Figures 2a and 2b show the results for two orbit test cases, with each computed with two different fidelity gravity models. The first is a perturbed near-circular LEO orbit with a semimajor axis of 8000 km. Figures 2a and 2b are for the 40 and 120 deg and order spherical harmonic gravity fields, respectively. The second test case is a MEO orbit with the semimajor axes of 20,000 km, and both orbits have zero nominal (unperturbed) inclination. As before, Figs. 2a and 2b are for a 40 and 120 deg and order spherical harmonic gravity field, respectively.

IV. Adaptation of Segment Length and Polynomial Degree for Picard–Chebyshev Methods

In this section, we explain the ideas underlying our method for tuning the segment size and the number of nodes required per segment to result in an efficient solution satisfying a user-specified precision tolerance. We mentioned earlier that the Picard iteration can converge over large time intervals for orbit dynamics (up to several orbits); however, computational efficiency typically requires that the time intervals be kept to a fraction of an orbit. We have developed two different methods for segmentation. The first method is used for propagating elliptic, gravitationally perturbed orbits near Earth (or some other gravitating primary body in the solar system), and the second method is used for numerically integrating any (smooth) generic nonlinear ordinary differential equation (not presented in this paper). The reason for adhering to a specialized segmentation scheme when propagating perturbed near-Earth orbits is that we can incorporate approximate solutions and other insights to improve the starting iterative and segmentation for this well-studied problem. Our adaptation approach recognizes that there are three main issues that must be considered: 1) the length of the time segment over which the integration is to occur; 2) the number of nodes ($M \geq N$) used to obtain the Nth-order Chebyshev series; and 3) the nonlinearity of the force model over the given time segment, which is most likely unknown. Note that the following discussion on segmentation pertains only to the Picard–Chebyshev IVP because only one segment is permitted for the Picard–Chebyshev TPBVP. However, the same procedure for determining the number of nodes for this one segment is used for the TPBVP. More details on the Picard–Chebyshev TPBVP were given in Refs. [17,31,32].

The nonlinearity experienced on a particular orbit can vary from very bland (for near-GEO motion) to extremely nonlinear if the orbit penetrates near perigee relatively deep into the busier part of the gravity model and high atmospheric drag region. Therefore, the perigee radius of particular orbits is a critical factor with regard to how much adaption in N and the segment length can be anticipated, especially if a high-precision solution is sought [19]. We make use of constant angle (true anomaly) segments that are qualitatively motivated by the Sundman transformation [37]. The Sundman transformation has been adopted in many integrators to use nearly uniform steps in the true anomaly (which correspond to highly variable steps in time) to integrate perturbed orbital motion. Macomber [19] confirmed that uniform segments of the true anomaly generally lead to a near-optimal segmentation scheme for the Picard iteration. With reference to Fig. 3, our adaptation scheme uses an odd number of segments per orbit, with the one-, three-, and five-segment cases shown. We note, qualitatively, that the pattern in Fig. 3 results in the highest node density at perigee (where the gravity field and other

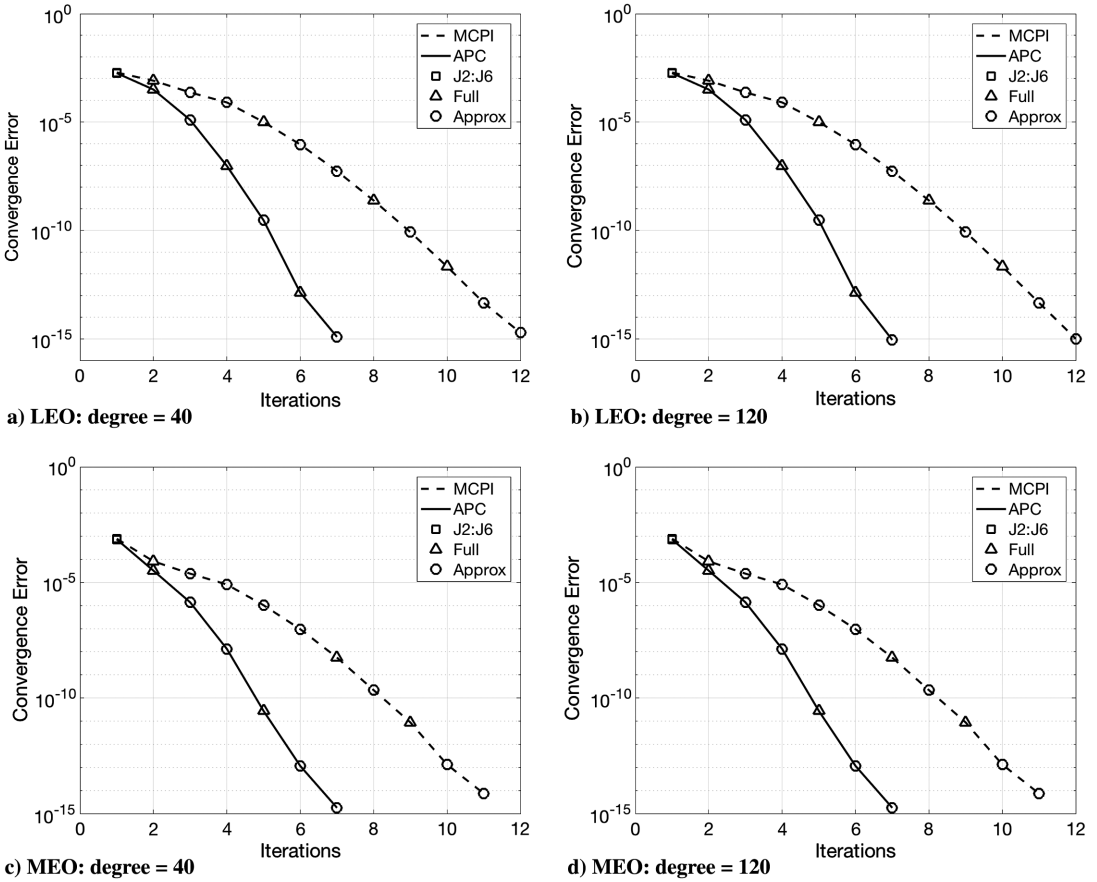


Fig. 2 Accelerated vs regular Picard–Chebyshev convergence rate comparison for LEO and MEO.

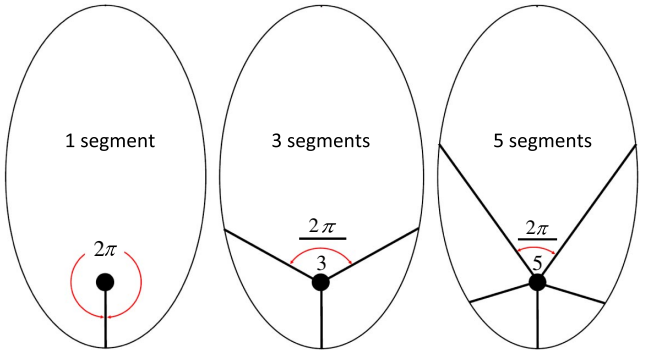


Fig. 3 Equal true anomaly segments.

perturbations are typically most nonlinear) and the most widely spaced nodes centered on apogee (where the gravity is weakest and has the least space/time variation). For multiorbit propagations, the same segmentation scheme is used; however, to prevent, over many orbits, an undesirable secular rotational displacement from the nominal segmentation pattern as shown in Fig. 3, it is necessary to reosculate each perigee passage instead of simply using the final states at the end of an orbit as the initial states for the beginning of the next orbit.

The total number of function evaluations required to achieve convergence is $(N+1)$ times the number m of Picard iterations where, for the sake of this discussion, we assume that $N = M$. For a given time interval, increasing N will increase the precision of the fit and increase the rate of convergence. However, at some point, the rate of convergence will become constant and further increasing N will no longer increase the precision of the fit but decrease the computational efficiency. For a given N, decreasing the time interval will increase the rate of convergence and increase the precision of the fit. However, using very short segments will result in more segments per orbit, which could also negatively impact computational efficiency. After extensive simulations and studies, we converged to the following automated and adaptive segmentation procedure.

The user-specified initial conditions (Cartesian) are converted to classical orbit elements, and these are then used to compute the Cartesian position and velocity at perigee. The perigee state is propagated forward in time for one-third of an orbit (true anomaly) using the analytical two-body solution. The two-body positions and velocities along this arc are then used to compute the corresponding accelerations, taking into account the fully perturbed model. The accelerations along this arc are fit with Chebyshev polynomials, starting with degree $N = 10$. Following the rule of thumb of Clenshaw and Curtis [22], if the last three coefficients of the integrand fit have magnitudes smaller than the $0.01 *$ tolerance, then the fit is acceptable. If fewer than the last three coefficients of the Chebyshev fit are smaller than the $0.01 *$ tolerance, then we continue doubling N until this requirement is achieved or some maximum N is reached (our default is $N_{\max} = 40$). If still unsatisfied, then we increase the number of segments by two to the next odd number of segments and repeat the process starting with $(N = 10)$. We apply this test to all three components of acceleration independently and require all three to pass this test. Finally, for the overfit case when more than three negligible coefficients are present on all three components of acceleration, we truncate the series such that only the last three coefficients are below $0.01 *$ tolerance. The reason for doubling N is that it allows us to take advantage of the fact that all the nodes for a certain value (say, $N = 20$) are contained within the $N = 40$ set (Fig. 4), thus allowing us to try several different fitting options, but have no more than $N_{\max}$ function evaluations. Once the number of nodes per segment and number of segments per orbit have been determined, we begin the propagation, holding these fixed for the entire propagation to the final time. Note that, if the initial or final conditions do not lie on a precomputed segments break (very likely), then the first or last segment is shortened and the number of nodes in that segment is scaled accordingly. We have found this approach to be

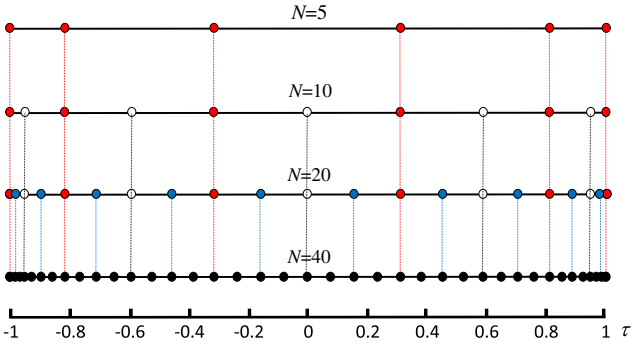


Fig. 4 Cosine nodes repeated when N is doubled.

very efficient while producing solutions that satisfy the user-specified precision tolerance.

In Figs. 5 and 6, we show two examples in which the absolute values of the Chebyshev coefficients for the x, y, and z components of the acceleration fit are monitored with regard to the user-specified tolerance. The orbit considered has a semimajor axis of 8000 km, an eccentricity of zero, and an inclination of 45 deg. All other elements are zero. In Figs. 5a and 6a, we show that a Chebyshev series with 16 coefficients (and three segments) is required for generating an engineering precision solution (Hamiltonian conserved to seven digits). In Figs. 5b and 6b, we show that a Chebyshev series with 40 coefficients (and seven segments) is required for generating a machine precision solution (Hamiltonian conserved to 15 digits). For the plots shown, acceleration has been nondimensionalized, and so the exponents correspond to which significant figure of acceleration is affected by the coefficients plotted. Boyd [20] suggested that, for fitting nondimensional acceleration, any coefficients with magnitudes smaller than 1×10^{-13} would not individually degrade the accuracy of the fit if our tolerance was 1×10^{-12}; however, a sum of several such small terms may indeed, occasionally, violate our precision tolerance, which is why we opt to use the conservative Clenshaw and Curtis small coefficient criteria of $0.01 *$ tolerance [22].

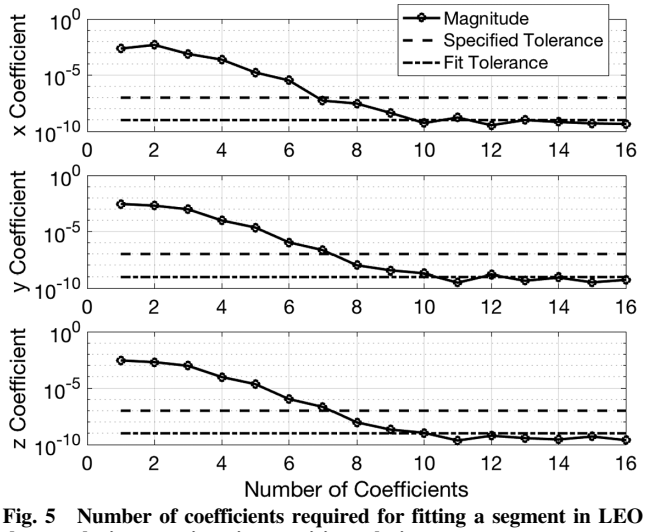


Fig. 5 Number of coefficients required for fitting a segment in LEO that results in an engineering precision solution.

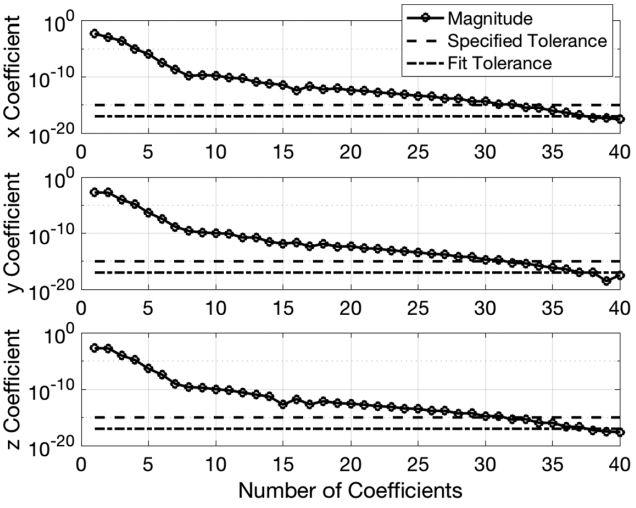


Fig. 6 Number of coefficients required for fitting a segment in LEO that results in a machine precision solution.

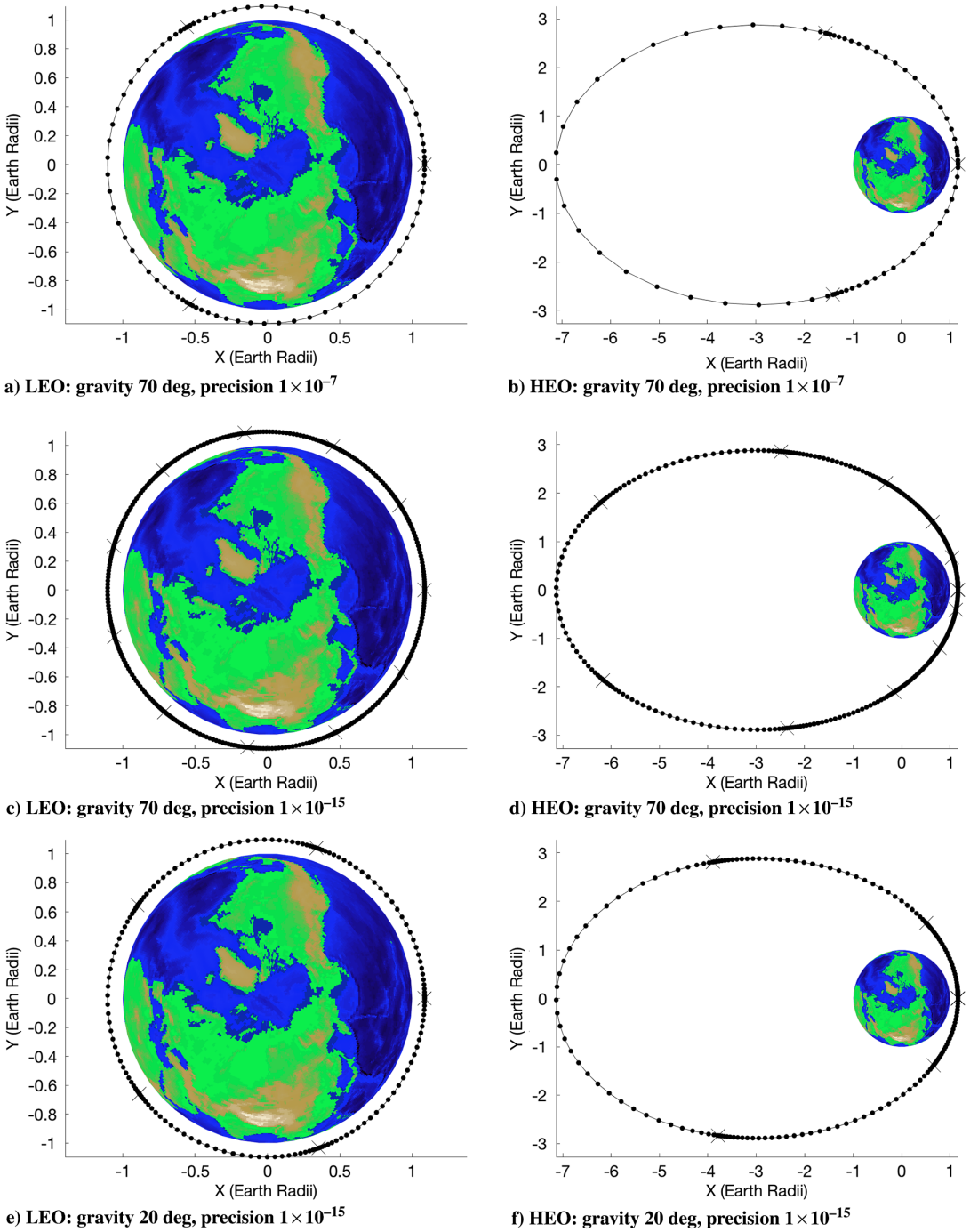
We present the required nodes/segmentation for three example orbits (see Fig. 7). The figures on the left (Figs. 7a, 7c, and 7e) show a LEO with a semimajor axis of $a = 7000$ km and an eccentricity of $e = 0.01$ (other elements are zero), and the figures on the right (Figs. 7b, 7d, and 7f) show a highly eccentric orbit (HEO) with a semimajor axis of $a = 26,554$ km and an eccentricity of $e = 0.72$ (other elements are zero). In Figs. 7a and 7b, the requirement is engineering precision (1×10^{-7}), whereas in Figs. 7c through 7f, the requirement is machine precision (1×10^{-15}). However, in Figs. 7a though 7d, the degree and order of the spherical harmonic gravity field are 70, whereas in Figs. 7e and 7f, the degree and order are 20. As is evident in Figs. 7a through 7d, the number of segments per orbit and the number of nodes per segment are greater for both the LEO and HEO cases when higher precision is required. Note that, in the cases shown in Figs. 7e and 7f, less nodes are required for achieving a machine precision solution than in the cases shown in Figs. 7c and 7d. This is because the degree of the gravity field for Figs. 7e and 7f is 20 as opposed to 70, and thus less nodes are required to precisely approximate the lower-fidelity model. Note that, in all cases, the segments follow the true anomaly pattern starting at perigee (far right) and, for each example, every segment spans the same true anomaly angle. This, by design, results in the nodal density being higher at perigee where the variations in the gravity field are stronger and lower at apogee where the variations in the gravity field and weaker.

V. Picard–Chebyshev Enhancements: Perturbed Two-Body Problem

A. Warm and Hot Start Orbit Approximations

The Picard iteration method and its enhancements all begin with a warm start, for all near-earth orbit calculations. We can use the classical two-body orbit as a warm start or, for a warmer start, we can use one of the several analytical perturbation theories that account for the first few zonal harmonics and the cannonball model of atmospheric drag. These orbit approximations typically provide three- or four-digit accuracy everywhere for an interval of several orbits and offer a significant advantage that is not available for traditional step-by step methods. The utility of this truth is illustrated in the applications discussed in the following.

For multiple orbits, the successive quasi-periodic orbits' perturbations are correlated to a significant degree (this is especially true for the typical 90 min period orbits) because the macroscopic gravity field along successive passes is highly correlated. As a result, if the time nodes are the same relative to the previous perigee as they were on the previous orbit, the final converged positions' displacements from the nodal positions of the respective warm or hot starts have been found to be correlated as well. Adding the converged nodal state displacements of the previous orbit to the warm start nodal states to provide a hot start has been found to typically reduce the number of Picard iterations on the subsequent orbit by one iteration. This is not true for long-period orbits, such as the Molniya, for the obvious reasons that the gravity perturbations on successive orbits are weakly correlated for long orbit periods. These hot starts are used for the multirevolution LEO performance of our algorithm.



a) LEO: gravity 70 deg, precision 1×10^{-7}

b) HEO: gravity 70 deg, precision 1×10^{-7}

c) LEO: gravity 70 deg, precision 1×10^{-15}

d) HEO: gravity 70 deg, precision 1×10^{-15}

e) LEO: gravity 20 deg, precision 1×10^{-15}

f) HEO: gravity 20 deg, precision 1×10^{-15}

Fig. 7 Number of nodes/segments required for propagating orbits around the Earth with different accuracy and precision requirements. Crosses denote segment breaks, and dots denote nodes.

The advantage gained using this device is small as compared to using the quasi-linearization acceleration and the local force approximation ideas in the following. We note that the use of warm start methods has also been implemented for the IRK integration method [38].

B. Gravitational Force Approximation

The magnitude of gravitational acceleration and its spatial frequency decays rapidly as r increases. Note that the full series spherical harmonic gravity potential has the form shown in the following, with ∞ as the theoretical upper limit, replaced by N:

$$V = V(r,\phi,\lambda) = \underbrace{\underbrace{\frac{\mu}{r}}_{V_{2\text{BODY}}(r)} + \frac{\mu}{r}\sum_{n=2}^{N}\left(\frac{r_\oplus}{r}\right)^n J_n P_n(\sin\phi)}_{V_{N_{\text{ZONAL}}}(r,\phi)} + \underbrace{\frac{\mu}{r}\sum_{n=2}^{N}\left(\frac{r_\oplus}{r}\right)^n \sum_{m=1}^{n} P_{n,m}(\sin\phi)\left(C_{n,m}\cos m\lambda + S_{n,m}\sin m\lambda\right)}_{V_{N_{\text{FULL}}}(r,\phi,\lambda)} \tag{26}$$

This globally valid and convergent infinite series is characterized by rapid early convergence to about four-digit accuracy (with seven or fewer terms at $r = 1.04r_{\oplus}$), followed by slow convergence thereafter, requiring about 40,000 terms to reach eight-digit convergence. As a consequence, the beauty of a global series model is offset by computational expense. The harmonic functions have a spatial period of about $(2\pi r_{\oplus})/200 \approxeq 200$ km; therefore, on a scale of less than ± 1 km, the gravity variations near a point of interest (i.e., a nodal point associated with Picard iteration) are nearly constant. This heuristic observation is not new [39–42], but it is important in the current discussion because, during the Picard iteration, the sequences of nodal positions in the Earth's gravity field are very close together (successive corrections, after a warm start, are typically less than 100 m immediately, and they quickly shrink to meters and smaller). The gravitational acceleration is computed from the potential by taking the gradient $\rightarrow \nabla V(r, \phi, \lambda)$ at the typical point in the spherical coordinate system (r, ϕ, λ). The gradient in spherical coordinates gives radial, east, and north components that can be projected into any desired frame, of course. The approach of Macomber et al. [18,19] is to determine the correction to an approximate global potential by a Taylor series expansion about each of the $N + 1$ nodes in the gravity field. Thus, in the vicinity of the kth node, we write

$$\boldsymbol{G}(r, \phi, \lambda) \approxeq \boldsymbol{G}_{6_{\text{ZONAL}}}(r, \phi, \lambda) + \Delta \boldsymbol{G}_{6_{\text{ZONAL}}}(r, \phi, \lambda)$$

where $\Delta \boldsymbol{G}_{6_{\text{ZONAL}}}(r, \phi, \lambda)$ is locally approximated by the Taylor series:

$$\boldsymbol{G}(r, \phi, \lambda) = \boldsymbol{G}_{6_{\text{ZONAL}}}(r, \phi) + \underbrace{\Delta \boldsymbol{G}_{6_{\text{ZONAL}}}(r_k, \phi_k, \lambda_k)}_{\text{local 6 ZONAL offset}} + \left[\frac{\partial \Delta \boldsymbol{G}_{6_{\text{ZONAL}}}}{\partial(r, \phi, \lambda)}\bigg|_k\right] \begin{Bmatrix} r - r_k \\ \phi - \phi_k \\ \lambda - \lambda_k \end{Bmatrix} + \ldots \tag{27}$$

Remarkably, following a warm start of the Picard iteration, within less than ± 0.5 km of a local evaluation, we find that we need only the once-computed local offset term

$$\Delta \boldsymbol{G}_{6_{\text{ZONAL}}}(r_k, \phi_k, \lambda_k) = \Delta \boldsymbol{G}_{200_{\text{FULL}}}(r_k, \phi_k, \lambda_k) - \boldsymbol{G}_{6_{\text{ZONAL}}}(r_k, \phi_k)$$

to achieve engineering precision gravity computation with 10^{-7} or smaller relative errors, whereas within less than ± 1 m, high-fidelity accuracy of 10^{-15} is achieved. Thus, remarkably, the fixed point convergence nature of the Picard iteration permits us to replace expensive spherical harmonic gradients of Eq. (26) with Eq. (27), ignoring the rightmost gradient correction, without accuracy loss, while reducing the computational cost of computing high-fidelity gravity by over an order of magnitude. In Figs. 8–10, the three orthogonal ± 500 m traverses of the gravity field are referenced to a point 320 km above Greenwich and are representative of gravity approximation errors in the vicinity of a local node in an atmosphere-skimming orbit. Note that the largest errors are smaller than 10^{-7} within ± 500 m of the local expansion nodes and smaller than 10^{-14} within ± 10 m, dropping to very near-machine precision for displacements smaller than a fraction of a meter. Thus, the Taylor series is ideally suited to the local gravity needed in asymptotic convergence of the Picard iteration with only intermittent use of the full-order gravity to reosculate in the vicinity of the current nodal

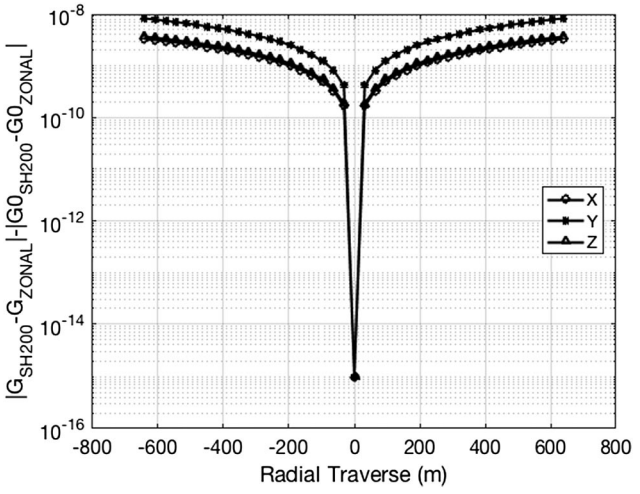


Fig. 8 Local gravitational radial variations in the vicinity of a typical node.

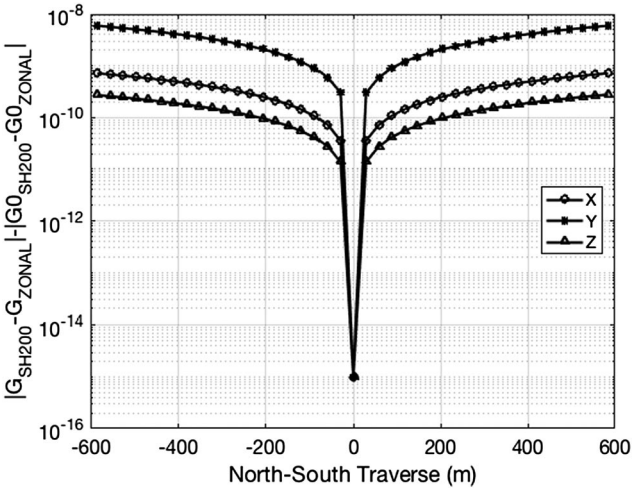


Fig. 9 Local gravitational north–south variations in the vicinity of a typical node.

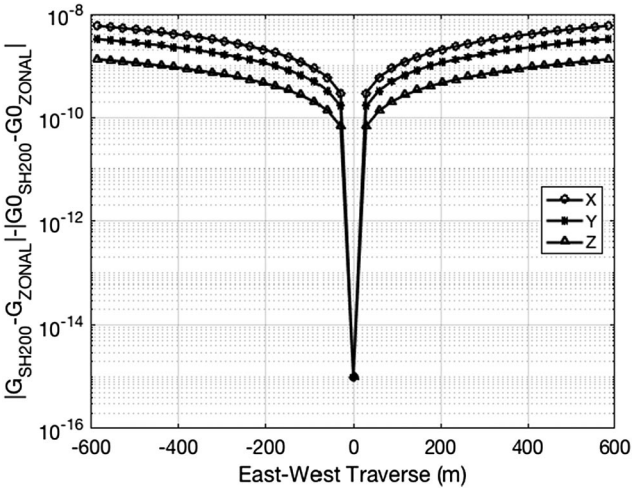


Fig. 10 Local gravitational east–west variations in the vicinity of a typical node.

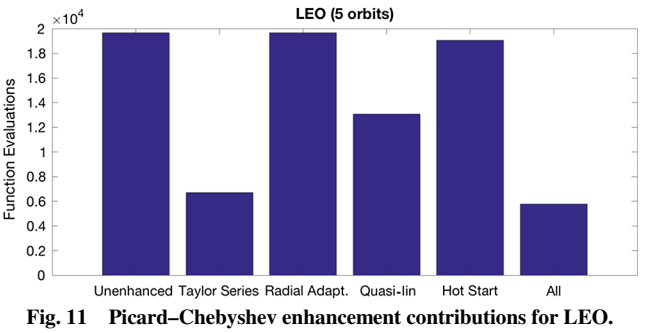


Fig. 11 Picard–Chebyshev enhancement contributions for LEO.

approximations. Notice that "on the fly" construction and use of these local Taylor approximations are "tailor made" for the Picard iteration, and the internal convergence metrics can readily dictate when the full gravity model is needed to reosculate the local approximations. The use of the local force model is the single most important computational acceleration of the Picard iteration vis-à-vis reducing computational cost, as well as the fact that the local approximations can be "established on the fly" by simply differencing the full-order model of gravitational acceleration from the zonal harmonic reference gravitational acceleration at the $N+1$ nodes of the Chebyshev representation.

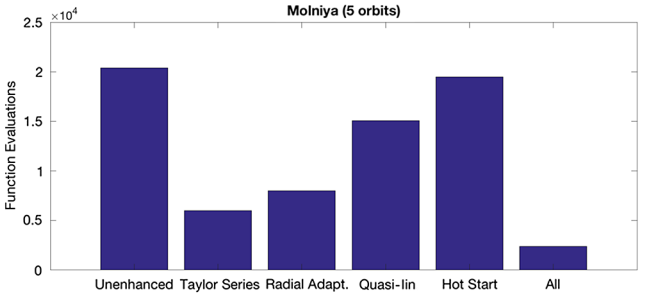


Fig. 12 Picard–Chebyshev enhancement contributions for Molniya.

Finally, we mention that the global gravity calculation from the spherical harmonic series itself can be readily adapted because the maximum contribution, on a sphere of given radius r, of the terms in the spherical harmonic series drops off sharply because $\propto (1/r^n)$ as the index n is swept over the range. The maximum contribution to the gravitational acceleration components for n on the sphere of radius r obtained from the gradient of V from Eq. (26) reveals a logarithmic waterfall. The saving is variable with r and desired precision

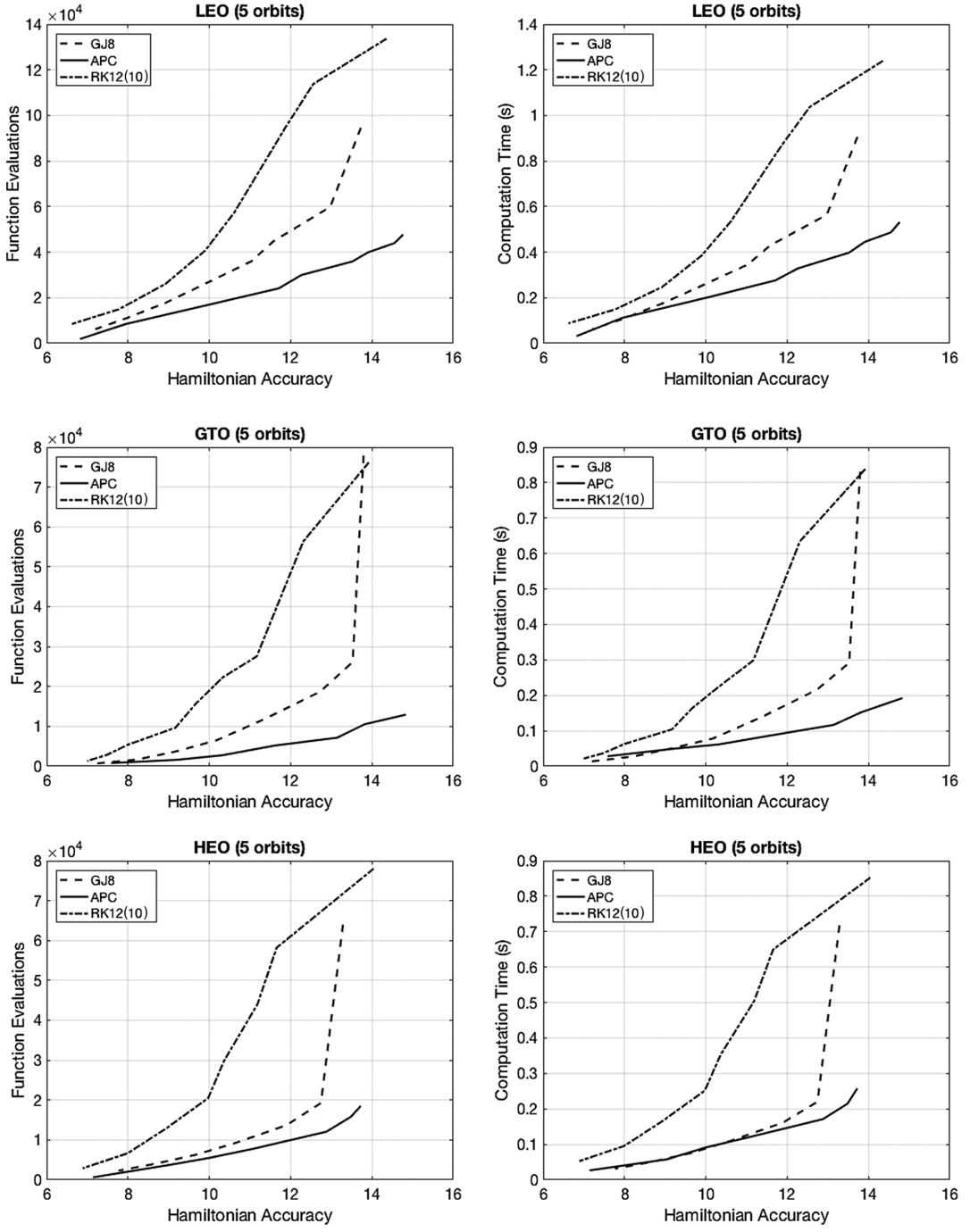


Fig. 13 Integrator accuracy and efficiency comparison for propagating five orbit periods of LEO, GTO, and HEO considering a 70 deg and order spherical harmonic gravity field.

(especially so for highly eccentric orbits, of course), and this is the whole point. We do not need to spend orders of magnitude more computational effort than necessary near the apogee of a Molniya orbit computing 40,000 terms in a series when only 10 or fewer are necessary to meet the desired precision tolerance. Frequently, analysts (and their resulting software) do not worry about this issue and simply call the spherical harmonic gravity routine with the upper limit fixed, irrespective of whether or not most terms in the series can be known a priori to contribute negligibly to local acceleration. We use radial adaptation to determine the upper limit of the spherical harmonic summation, which is consistent with maintaining precision in the gravitational acceleration [18,19,43].

C. Enhancement Contributions: Illustrated by Computational

Figures 11 and 12 display the number of equivalent function evaluations required for propagating a low Earth orbit and a Molniya orbit for five Keplerian orbit periods considering a 70 deg and order spherical harmonic gravity field. The equivalent function evaluations are the number of full force evaluations plus the number of operations for one approximate force calculation divided by the number of operations for one full force calculation. In each plot, the leftmost bar (unenhanced) represents the number of function evaluations when none of the enhancements (discussed in the previous sections) are included. The second bar (Taylor series) represents the algorithm when only the local Taylor series gravity approximation is included. In both the LEO and the Molniya cases, this provides the greatest decrease in the number of function equivalent evaluations. The third bar (radial adapt.) represents the algorithm when only radial adaptation is included. Radial adaptation is not significant for the LEO case because the orbit is near circular; however, for the Molniya case, radial adaptive is very significant because the orbit is highly eccentric. The fourth bar (quasi-lin.) represents the integral quasi linearization, and for both test cases this results in a significant reduction in the number of function evaluations. The fifth bar (hot start) represents the inclusion of the hot start in the algorithm. In both cases, this results in a small reduction in the number of function evaluations. Finally, the rightmost bar (all) represents the algorithm when all the enhancements are included. In both cases, the rightmost bar is significantly lower than the leftmost bar (algorithm with no enhancements). All of these enhancements (except radial adaptation) are unique to our algorithm due to the path approximation nature, i.e., nodes along the trajectory converge to fixed points in space. Similar techniques may be suitable for implementation in other implicit numerical integration methods.

VI. Integrator Performance Comparison

We demonstrate the performance of our algorithm by comparing it to the performance of an eighth-order Gauss–Jackson integrator and a 10th-/12th-order Runge–Kutta integrator. The results for three orbit test cases are shown in Fig. 13. The top panel shows the number of function evaluations as a function of the Hamiltonian accuracy (left) and the computation as a function of the Hamiltonian accuracy (right) for the low Earth orbit ($a = 7000$ km, $e = 0.01$, $i = 45$ deg, $\Omega = 0$ deg, $w = 0$ deg, and $M = 0$ deg). (The left and right columns show the number of function evaluations as a function of the Hamiltonian accuracy and the computation time, respectively.) The APC clearly outperforms the GJ8 and the RK12(10) over a range of accuracies, and it is notably more efficient for high accuracies. Similarly, the second and third panels show the results for the geostationary transfer orbit ($a = 25,200$ km, $e = 0.68$, $i = 0$ deg, $\Omega = 0$ deg, $w = 0$ deg, and $M = 0$ deg) and the Molniya orbit ($a = 26,554$ km, $e = 0.72$, $i = 63$ deg, $\Omega = 0$ deg, $w = 0$ deg, and $M = 0$ deg), respectively. The trends are similar, showing that the APC is more efficient than the other integrators, especially when high accuracy is required. In all cases, the test cases are propagated for five orbit periods considering a 70 deg and order spherical harmonic gravity model (Earth Gravity Model 2008). In addition, the long-term stability of the APC is

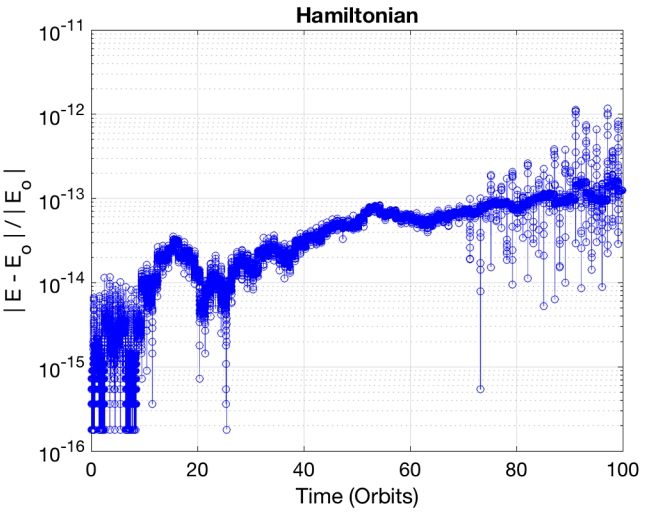


Fig. 14 APC achieving 12-digit accuracy in the Hamiltonian for seven weeks of orbit propagation for a Molniya orbit with a 70 deg and order spherical harmonic gravity field.

shown in Fig. 14, in which the Hamiltonian preserves 12 digits for a Molniya orbit propagated over seven weeks (100 orbits). We mention that GJ8 and RK12(10) had comparable stability but with increased function evaluation cost. The examples shown in this paper are for IVPs, but similar results are achievable for TPBVPs. In Refs. [31,32], Woollands et al. showed a similar comparison for solving TPBVPs in which the Picard–Chebyshev method did not use integral quasi linearization and still outperformed competing algorithms.

VII. Conclusions

New formulations, algorithms, and computational studies based on these developments for solving initial value problems using an adaptive segmentation modified Chebyshev–Picard integration are presented. The approach includes an integral error feedback term based on quasi linearization to accelerate terminal convergence. The approach also incorporates local gravity approximations that exploit the fixed-point nature of convergence. The formulation is specialized for first-order systems of differential equations and second-order differential equations in cascade form. The kinematic consistency enforced in the latter algorithm reduces the Picard iterations by about a factor of two. The computational studies compare the new approach with two familiar integrators and study the relative efficiency for a given precision tolerance. The Picard algorithms are well suited for parallelization, although the present implementation and a comparison with competing algorithms are serial mode implementations. The performance of the adaptive Picard–Chebyshev algorithm is evaluated by solving benchmark problems in astrodynamics focusing on the perturbed two-body problem, and the results were compared with those obtained using a 12th-order Runge–Kutta integrator and an eighth-order Gauss–Jackson. The adaptive Picard–Chebyshev algorithm is more efficient than the other methods, as measured by the number of function evaluations and the computation time. Stability was demonstrated with respect to error buildup over long-term (seven week) integration of the Molniya orbit, maintaining 12-digit accuracy while using 15-digit arithmetic and a high-fidelity force model. It is mentioned that the Gauss–Jackson and 12th-order Runge–Kutta algorithms have comparable stability but with increased function evaluation cost. This accelerated and adaptive Picard iteration method represents a significant contribution to the methodologies available for solving one of the most important differential equations in astrodynamics.

Appendix A: Picard–Chebyshev Vector–Matrix Derivations

A.1. First-Order Initial Value Problem

Emile Picard [16] and others observed that any first-order differential equation

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}\big(t, \boldsymbol{x}(t)\big), \quad t \in [t_0, t_f], \quad \boldsymbol{x} \in R^{n\times 1}, \quad \boldsymbol{f} \in R^{n\times 1} \tag{A1}$$

with an initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ could be rearranged as the corresponding integral equation:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}\big(s, \boldsymbol{x}(s)\big)\,\mathrm{d}s \tag{A2}$$

A sequence of approximate solutions $\boldsymbol{x}^i(t)$, $(i = 1, 2, 3, \ldots, \infty)$, of the true solution $\boldsymbol{x}(t)$ that satisfies this integral equation may be obtained through the Picard iteration using the following set of approximate paths:

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}\big(s, \boldsymbol{x}^{i-1}(s)\big)\,\mathrm{d}s, \qquad i = 1, 2, \ldots \tag{A3}$$

Picard [16] proved that, for smooth, differentiable single-valued nonlinear functions $\boldsymbol{f}(t, \boldsymbol{x}(t))$, there was a time interval $|t_f - t_0| < \delta$ and a starting trajectory $\boldsymbol{x}^0(t)$ satisfying $\|\boldsymbol{x}^0(t) - \boldsymbol{x}(t)\|_\infty < \Delta$ and that, for suitable finite bounds (δ, Δ), the Picard sequence of trajectories represented a contraction operator that converged to the unique solution of the initial value problem.

A finite set of orthogonal Chebyshev polynomials is used as the basis function set to approximate the integrand along the available approximate trajectory $\boldsymbol{x}^{i-1}$. That is,

$$\boldsymbol{f}\big(s, \boldsymbol{x}^{i-1}(s)\big) = \sum_{k=0}^{N-1} \boldsymbol{a}_k^{i-1} T_k(\tau) \tag{A4}$$

where $\boldsymbol{a}_k$ are the least squares coefficients of the fit of $\boldsymbol{f}(s, \boldsymbol{x}^{i-1}(s))$, and $T_k(\tau)$ are the Chebyshev polynomials with the forward and reverse time transformations $\tau = -1 + 2(t - t_0)/(t_f - t_0)$ and $t = t_0 + (\tau + 1)(t_f - t_0)/2$. Computation of the least-squares coefficients is shown in Eq. (A5), where the weight matrix

$$W = \mathrm{diag}\left\{\tfrac{1}{2} \quad 1 \quad \cdots \quad 1 \quad \tfrac{1}{2}\right\}$$

$$\boldsymbol{a} = (T^T W T)^{-1} T^T W \boldsymbol{f} \tag{A5}$$

The explicit solution for the coefficients of Eq. (A5) is given by the independent/uncoupled ratios of inner products as

$$\boldsymbol{a}_k = \frac{\langle T_k(\tau), \boldsymbol{f}(\tau)\rangle}{\langle T_k(\tau), T_k(\tau)\rangle} \equiv \frac{\sum_{j=0}^{M} W_{jj} T_k(\tau_j) \boldsymbol{f}(\tau_j)}{\sum_{j=0}^{M} W_{jj} T_k^2(\tau_j)}$$
$$\equiv \frac{1}{c_k} \sum_{j=0}^{M} W_{jj} T_k(\tau_j) \boldsymbol{f}(\tau_j), \quad \text{for } k = 0, 1, 2, \ldots, N-1 \tag{A6}$$

Note that careful selection of the basis functions (Chebyshev polynomials), node locations (Cosine nodes), and weight matrix W leads to a diagonal $T^T W T$ matrix, and thus a trivial matrix inverse as represented by $1/c_k$ on the far right side of Eq. (A6). More details can be found in Ref. [32].

Using Eq. (A4), we can write Eq. (A3) as Eq. (A7). Note that p in Eq. (A7) is the scale factor that transforms the independent variable from the time domain into the τ domain; that is,

$$\frac{\mathrm{d}\boldsymbol{x}(\tau)}{\mathrm{d}\tau} = \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = \frac{t_f - t_0}{2}\boldsymbol{f}\big(t, \boldsymbol{x}(t)\big) = p\boldsymbol{f}\big(t, \boldsymbol{x}(t)\big)$$

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + p\int_{-1}^{\tau} \sum_{k=0}^{N-1} \boldsymbol{a}_k^{i-1} T_k(s)\,\mathrm{d}s, \quad i = 1, 2, \ldots \tag{A7}$$

Expanding the summation notation and bringing the least-squares coefficients $\boldsymbol{a}$ out of the integral lead to

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + p\boldsymbol{a}_0^{i-1}\int_{-1}^{\tau} T_0(s)\,\mathrm{d}s + p\boldsymbol{a}_1^{i-1}\int_{-1}^{\tau} T_1(s)\,\mathrm{d}s + , \ldots, + p\boldsymbol{a}_{N-2}^{i-1}\int_{-1}^{\tau} T_{N-2}(s)\,\mathrm{d}s + p\boldsymbol{a}_{N-1}^{i-1}\int_{-1}^{\tau} T_{N-1}(s)\,\mathrm{d}s \tag{A8}$$

The following relationship exists for Chebyshev polynomials and their integrals:

$$\int T_0(s)\,\mathrm{d}s = T_1(s), \quad \int T_1(s)\,\mathrm{d}s = \frac{1}{4}\big(T_2(s) + T_0(s)\big),$$
$$\int T_k(s)\,\mathrm{d}s = \frac{1}{2}\left(\frac{T_{k+1}(s)}{k+1} - \frac{T_{k-1}(s)}{k-1}\right) \tag{A9}$$

Substituting the Chebyshev integral relationships from Eq. (A9) into Eq. (A8) leads to Eq. (A10). Note that Chebyshev polynomials exist on the range $[-1 \ldots 1]$; thus, the lower and upper limits of integration are -1 and τ, respectively:

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + p\left[\boldsymbol{a}_0^{i-1} T_1(s) + \frac{1}{4}\boldsymbol{a}_1^{i-1}\big(T_2(s) + T_0(s)\big) + \frac{1}{2}\boldsymbol{a}_2^{i-1}\left(\frac{T_3(s)}{3} - T_1(s)\right) + \ldots + \frac{1}{2}\boldsymbol{a}_{N-2}^{i-1}\left(\frac{T_{N-1}(s)}{N-1} - \frac{T_{N-3}(s)}{N-3}\right) + \frac{1}{2}\boldsymbol{a}_{N-1}^{i-1}\left(\frac{T_N(s)}{N} - \frac{T_{N-2}(s)}{N-2}\right)\right]\Bigg|_{-1}^{\tau} \tag{A10}$$

Writing Eq. (A10) in matrix form leads to

$$\overset{1\times n}{\boldsymbol{x}^i(t)} = \overset{1\times n}{\boldsymbol{x}(-1)} + \overset{1\times(N+1)}{\begin{bmatrix} T_0(s) & \ldots & T_N(s)\end{bmatrix}\Big|_{-1}^{\tau}} p \overset{(N+1)\times n}{\begin{Bmatrix} \frac{1}{4}\boldsymbol{a}_0^{i-1} \\ \frac{1}{2}\left(\boldsymbol{a}_0^{i-1} - \boldsymbol{a}_2^{i-1}\right) \\ \frac{1}{4}\left(\boldsymbol{a}_1^{i-1} - \boldsymbol{a}_3^{i-1}\right) \\ \vdots \\ \frac{1}{2k}\left(\boldsymbol{a}_{k-1}^{i-1} - \boldsymbol{a}_{k+1}^{i-1}\right) \\ \vdots \\ \frac{1}{2(N-1)}\boldsymbol{a}_{N-2}^{i-1} \\ \frac{1}{2N}\boldsymbol{a}_{N-1}^{i-1} \end{Bmatrix}} \tag{A11}$$

The matrix of least-squares coefficients can be further partitioned into a sparsely populated matrix $[S]$ and matrix $[G^{i-1}]$ containing the least-squares coefficients of the approximated forcing function. Matrix $[G^{i-1}]$ can be further partitioned into the least-squares operator matrix $[A]$ and the forcing function evaluated at the discrete sample points $[F^{i-1}]$.

$$\overset{(N+1)\times n}{\begin{Bmatrix} \frac{1}{4}\boldsymbol{a}_0^{i-1} \\ \frac{1}{2}\left(\boldsymbol{a}_0^{i-1}-\boldsymbol{a}_2^{i-1}\right) \\ \frac{1}{4}\left(\boldsymbol{a}_1^{i-1}-\boldsymbol{a}_3^{i-1}\right) \\ \vdots \\ \frac{1}{2k}\left(\boldsymbol{a}_{k-1}^{i-1}-\boldsymbol{a}_{k+1}^{i-1}\right) \\ \vdots \\ \frac{1}{2(N-1)}\boldsymbol{a}_{N-2}^{i-1} \\ \frac{1}{2N}\boldsymbol{a}_{N-1}^{i-1} \end{Bmatrix}} = \underbrace{\overset{(N+1)\times N}{\begin{bmatrix} \frac{1}{4} & 0 & & \cdots & & & & 0 \\ 1 & 0 & -\frac{1}{2} & 0 & & & & \\ 0 & \frac{1}{4} & 0 & -\frac{1}{4} & 0 & & & \\ & 0 & \ddots & \ddots & \ddots & 0 & & \\ \vdots & \vdots & \vdots & \frac{1}{2k} & 0 & -\frac{1}{2k} & \vdots & \vdots \\ & & & & \ddots & \ddots & \ddots & 0 \\ & & & & & \frac{1}{2(N-2)} & 0 & \frac{-1}{2(N-1)} \\ & & & & & 0 & \frac{1}{2(N-1)} & 0 \\ 0 & & & & & \cdots & 0 & \frac{1}{2N} \end{bmatrix}}}_{[S]} \overset{N\times n}{[G^{i-1}]} = [S][A]\left[F^{i-1}\right] \tag{A12}$$

In Eq. (A12),

$$[G^{i-1}] = \overbrace{(T^TWT)^{-1}T^TW}^{N\times(M+1)}\ \overset{(M+1)\times n}{[F^{i-1}]} = A[F^{i-1}]$$

where

$$[W] = \overset{(M+1)\times(M+1)}{\begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}}$$

and

$$[F^{i-1}] = \overset{(M+1)\times n}{\begin{bmatrix} \boldsymbol{f}_1\left(s_0,\boldsymbol{x}^{i-1}(s_0)\right) & \cdots & \boldsymbol{f}_n\left(s_0,\boldsymbol{x}^{i-1}(s_0)\right) \\ \boldsymbol{f}_1\left(s_1,\boldsymbol{x}^{i-1}(s_1)\right) & \cdots & \boldsymbol{f}_n\left(s_1,\boldsymbol{x}^{i-1}(s_1)\right) \\ \vdots & \ddots & \vdots \\ \boldsymbol{f}_1\left(s_M,\boldsymbol{x}^{i-1}(s_M)\right) & \cdots & \boldsymbol{f}_1\left(s_M,\boldsymbol{x}^{i-1}(s_M)\right) \end{bmatrix}}$$

Referring back to Eq. (A11) and evaluating $[T]$ between limits leads to the following matrix equation:

$$\boldsymbol{x}^i(t) = \boldsymbol{x}(-1) + \Big[\begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \end{bmatrix}\Big]p[S][A]\left[F^{i-1}\right] \tag{A13}$$

The left side of Eq. (A11), which is the solution trajectory $\boldsymbol{x}^i(t)$, can also be written as a Chebyshev series as follows:

$$\boldsymbol{x}^i(t) = \sum_{k=0}^{N}\boldsymbol{\beta}_k^{i-1}T_k(s) \tag{A14}$$

Writing both the left and right sides of Eq. (A11) in matrix form leads to the following:

$$\boldsymbol{x}^i(t) = \begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix}\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \boldsymbol{x}(-1) + \Big[\begin{bmatrix} T_0(\tau) & \cdots & T_N(\tau) \end{bmatrix} - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \end{bmatrix}\Big]p[S][A]\left[F^{i-1}\right] \tag{A15}$$

As expected, grouping the constant terms clearly reveals that $\boldsymbol{C}$ only contributes to the $\boldsymbol{\beta}_0^i$ coefficient:

$$\boldsymbol{C} = \boldsymbol{x}(-1) - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \end{bmatrix}p[S][A]\left[F^{i-1}\right] \tag{A16}$$

Equating the coefficients of $[T]$ on each side of Eq. (A15) results in an expression for the $\boldsymbol{\beta}$ coefficients in terms of the least-squares coefficients:

$$\begin{bmatrix} \boldsymbol{\beta}_0^i \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \underbrace{\begin{bmatrix} x_1(-1) & \cdots & x_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} - \begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}p[S][A]\left[F^{i-1}\right]}_{\boldsymbol{C}} + p[S][A]\left[F^{i-1}\right] \tag{A17}$$

Equation (A17) can be more compactly written as

$$\begin{bmatrix} \overset{(N+1)\times n}{\boldsymbol{\beta}_0^i} \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-1}^i \\ \boldsymbol{\beta}_N^i \end{bmatrix} = \overset{(N+1)\times n}{\boldsymbol{X}_0} + \overbrace{p\underbrace{[[I]-[L]]}_{(N+1)\times(N+1)}\underbrace{[S]}_{(N+1)\times N}}^{\overset{(N+1)\times N}{[P_1]}} \overset{N\times(M+1)}{[A]} \overset{(M+1)\times n}{[F^{i-1}]} \tag{A18}$$

where

$$\boldsymbol{X}_0 = \begin{bmatrix} \overset{(N+1)\times n}{x_1(-1)} & \cdots & x_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

$$[L] = \begin{bmatrix} \overset{(N+1)\times(N+1)}{T_0(-1)} & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

and $[P_1] = p[[I]-[L]][S]$.

In summary, the vector–matrix form of the first-order Picard–Chebyshev integration is shown in Eq. (A19), where $[P_1]$ is the first-order integration operator and $[A]$ is the least-squares operator:

$$\boldsymbol{\beta}^i = \boldsymbol{X}_0 + [P_1][A]\left[F^{i-1}\right], \qquad \boldsymbol{x}^i(t) = T(\tau)\boldsymbol{\beta}^i \tag{A19}$$

A.2. Second-Order Initial Value Problem

Consider the following second-order ordinary differential equation:

$$\frac{\mathrm{d}^2\boldsymbol{x}(t)}{\mathrm{d}t^2} = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{v}(t)\big), \quad t \in [t_0, t_f], \quad \boldsymbol{x} \in R^{n\times 1}, \quad \boldsymbol{v} \in R^{n\times 1}, \quad \boldsymbol{f} \in R^{n\times 1} \tag{A20}$$

with initial conditions $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ and $\boldsymbol{v}(t_0) = \boldsymbol{v}_0$. This can be rearranged to obtain the corresponding integral equation for velocity:

$$\boldsymbol{v}^i(\tau) = \boldsymbol{v}(-1) + \int_{-1}^{s} \boldsymbol{f}\big(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q)\big)\,\mathrm{d}q \tag{A21}$$

Note that the position may be computed by directly integrating the velocity in Eq. (A21):

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}(-1) + \int_{-1}^{\tau}\left\{\boldsymbol{v}(-1) + \int_{-1}^{s} \boldsymbol{f}\Big(q, \boldsymbol{x}^{i-1}(q), \boldsymbol{v}^{i-1}(q)\Big)\,\mathrm{d}q\right\}\mathrm{d}s \tag{A22}$$

Similar to the first-order case, the velocity can be written in terms of a Chebyshev series, where the $\boldsymbol{\beta}$ are the velocity coefficients and the $\boldsymbol{a}$ are the least-squares coefficients. Note that the upper limit on the summation for the least-squares coefficients is $N-2$ as compared with $N-1$ for first-order systems:

$$\boldsymbol{v}^i(\tau) = \sum_{k=0}^{N-1}\boldsymbol{\beta}_k^i T_k(s) = \boldsymbol{v}(-1) + p\int_{-1}^{s}\sum_{k=0}^{N-2}\boldsymbol{a}_k^{i-1}T_k(q)\,\mathrm{d}q \tag{A23}$$

The position can also be written in terms of a Chebyshev series, where the α are the position coefficients and the $\boldsymbol{a}$ are the least-squares coefficients:

$$\begin{aligned}\boldsymbol{x}^i(\tau) &= \sum_{k=0}^{N}\alpha_k^i T_k(\tau) \\ &= \boldsymbol{x}(-1) + p\int_{-1}^{\tau}\left\{\boldsymbol{v}(-1) + p\int_{-1}^{s}\sum_{k=0}^{N-2}\boldsymbol{a}_k^{i-1}T_k(q)\,\mathrm{d}q\right\}\mathrm{d}s\end{aligned} \tag{A24}$$

However, it is more convenient to compute the position coefficients α directly from the velocity coefficients $\boldsymbol{\beta}$ by applying the integration operator twice:

$$\begin{aligned}\boldsymbol{x}^i(\tau) &= \sum_{k=0}^{N}\alpha_k^i T_k(\tau) \\ &= \boldsymbol{x}(-1) + p\int_{-1}^{\tau}\sum_{k=0}^{N-1}\boldsymbol{\beta}_k^i T_k(s)\,\mathrm{d}s\end{aligned} \tag{A25}$$

In vector–matrix form, the velocity is computed as follows:

$$\begin{bmatrix} \overset{N\times n}{\boldsymbol{\beta}_0^i} \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-2}^i \\ \boldsymbol{\beta}_{N-1}^i \end{bmatrix} = \overset{N\times n}{\boldsymbol{V}_0} + \overbrace{p\underbrace{[[I]-[L]]}_{N\times N}\underbrace{[S]}_{N\times(N-1)}}^{\overset{N\times(N-1)}{[P_1]}} \overset{(N-1)\times(M+1)}{[A]} \overset{(M+1)\times n}{[F^{i-1}]} \quad \text{and}$$

$$\boldsymbol{v}^i(t) = T(\tau)\boldsymbol{\beta}^i \tag{A26}$$

where

$$\boldsymbol{V}_0 = \begin{bmatrix} \overset{N\times n}{v_1(-1)} & \cdots & v_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

$$[L] = \begin{bmatrix} \overset{N\times N}{T_0(-1)} & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

In vector–matrix form, the position is computed as follows:

$$\begin{bmatrix} \overset{(N+1)\times n}{\boldsymbol{\alpha}_0^i} \\ \alpha_1^i \\ \vdots \\ \boldsymbol{\alpha}_{N-2}^i \\ \alpha_{N-1}^i \end{bmatrix} = \overset{(N+1)\times n}{\boldsymbol{X}_0} + \overbrace{p\underbrace{[[I]-[L]]}_{(N+1)\times(N+1)}\underbrace{[S]}_{(N+1)\times N}}^{\overset{(N+1)\times N}{[P_2]}} \begin{bmatrix} \overset{N\times n}{\boldsymbol{\beta}_0^i} \\ \boldsymbol{\beta}_1^i \\ \vdots \\ \boldsymbol{\beta}_{N-2}^i \\ \boldsymbol{\beta}_{N-1}^i \end{bmatrix} \quad \text{and}$$

$$\boldsymbol{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i \tag{A27}$$

where

$$\boldsymbol{X}_0 = \overset{(N+1)\times n}{\begin{bmatrix} x_1(-1) & \cdots & x_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}$$

$$[L] = \overset{(N+1)\times(N+1)}{\begin{bmatrix} T_0(-1) & \cdots & T_N(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}$$

and $[P_2] = p[[I] - [L]][S]$.

In summary, the vector–matrix form of the second-order Picard–Chebyshev integration is shown in Eq. (A28), where $[P_1]$ is the first-order integration operator, $[P_2]$ is the second-order integration operator, and $[A]$ is the least-squares operator:

$$\boldsymbol{\beta}^i = \boldsymbol{V}_0 + [P_1][A]\Big[F^{i-1}\Big], \qquad \boldsymbol{v}^i(t) = T(\tau)\boldsymbol{\beta}^i \tag{A28}$$

$$\alpha^i = \boldsymbol{X}_0 + [P_2]\,\boldsymbol{\beta}^i, \qquad \boldsymbol{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i \tag{A29}$$

A.3. Two-Point Boundary Value Problem

Consider the following second-order ordinary differential equation:

$$\frac{\mathrm{d}^2\boldsymbol{x}(t)}{\mathrm{d}t^2} = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{v}(t)\big), \qquad t \in [t_0, t_f], \qquad \boldsymbol{x} \in R^{n\times 1}, \quad \boldsymbol{v} \in R^{n\times 1}, \qquad \boldsymbol{f} \in R^{n\times 1} \tag{A30}$$

with boundary conditions $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ and $\boldsymbol{x}(t_f) = \boldsymbol{x}_f$. The velocity can be written in terms of a Chebyshev series and, similar to the second-order initial value problem, the $\boldsymbol{\beta}$ coefficients can be computed in terms of the least-squares $\boldsymbol{a}$ coefficients. The initial velocity is unknown, and the resulting pseudovelocity is correct to within the constant of integration:

$$\boldsymbol{v}^i_{\text{pseudo}}(\tau) = \sum_{k=0}^{N-1}\boldsymbol{\beta}^i_{k\,\text{pseudo}}T_k(s) = p\int_{-1}^{s}\sum_{k=0}^{N-2}\boldsymbol{a}_k^{i-1}T_k(q)\,\mathrm{d}q \tag{A31}$$

The position can also be written in terms of a Chebyshev series; however, it is important to note that the unknown integration constant at the velocity level is contained within the α_0 and α_1 coefficients:

$$\begin{aligned}\boldsymbol{x}^i_{\text{pseudo}}(\tau) &= \sum_{k=0}^{N}\alpha^i_{k\,\text{pseudo}}T_k(\tau) \\ &= p\int_{-1}^{\tau}\left\{\boldsymbol{v}(-1) + p\int_{-1}^{s}\sum_{k=0}^{N-2}\boldsymbol{a}_k^{i-1}T_k(q)\,\mathrm{d}q\right\}\mathrm{d}s\end{aligned} \tag{A32}$$

The vector–matrix form of the pseudovelocity and pseudoposition are computed as follows:

$$\boldsymbol{\beta}^i_{\text{pseudo}} = [P_1][A]\Big[F^{i-1}\Big], \qquad \boldsymbol{v}^i_{\text{pseudo}}(t) = T(\tau)\boldsymbol{\beta}^i_{\text{pseudo}} \tag{A33}$$

$$\alpha^i_{\text{pseudo}} = [P_2]\boldsymbol{\beta}^i_{\text{pseudo}}, \qquad \boldsymbol{x}^i_{\text{pseudo}}(t) = T(\tau)\boldsymbol{\alpha}^i_{\text{pseudo}} \tag{A34}$$

The unknown initial velocity is contained linearly, and it can be solved as shown in the following steps:

$$\begin{aligned}\boldsymbol{x}^i(\tau) &= \sum_{k=0}^{N}\alpha^i_k T_k(\tau) \\ &= \boldsymbol{x}(-1) + p\int_{-1}^{\tau}\left\{\boldsymbol{v}(-1) + p\int_{-1}^{s}\sum_{k=0}^{N-2}\boldsymbol{a}_k^{i-1}T_k(q)\,\mathrm{d}q\right\}\mathrm{d}s \\ &= \boldsymbol{x}(-1) + \left(\frac{t_f - t_0}{2}\right)\boldsymbol{v}_0(\tau+1) + p\int_{-1}^{\tau}\sum_{k=0}^{N-1}\boldsymbol{\beta}^i_{k\,\text{pseudo}}T_k(s)\,\mathrm{d}s\end{aligned} \tag{A35}$$

Equation (A35) may be written in matrix form as follows:

$$\boldsymbol{x}^i(\tau) = \boldsymbol{x}(-1) + \left(\frac{t_f - t_0}{2}\right)\boldsymbol{v}_0(\tau+1) + \Big[T(\tau)\Big]\begin{bmatrix}\boldsymbol{\alpha}^i_{0\,\text{pseudo}} \\ \vdots \\ \boldsymbol{\alpha}^i_{N\,\text{pseudo}}\end{bmatrix} \tag{A36}$$

and, at the final time, where $\tau = 1$ and we have the known final position $\boldsymbol{x}(t_f)$, Eq. (A36) becomes

$$\boldsymbol{x}(1) = \boldsymbol{x}(-1) + (t_f - t_0)\boldsymbol{v}_0 + \Big[T(1)\Big]\begin{bmatrix}\boldsymbol{\alpha}^i_{0\,\text{pseudo}} \\ \vdots \\ \boldsymbol{\alpha}^i_{N\,\text{pseudo}}\end{bmatrix} \tag{A37}$$

Equation (A37) can now be rearranged for the initial velocity:

$$\boldsymbol{v}_0 = \boldsymbol{v}(-1) = \frac{\boldsymbol{x}(1) - \boldsymbol{x}(-1)}{t_f - t_0} - \frac{1}{t_f - t_0}\Big[T(1)\Big]\begin{bmatrix}\boldsymbol{\alpha}^i_{0\,\text{pseudo}} \\ \vdots \\ \boldsymbol{\alpha}^i_{N\,\text{pseudo}}\end{bmatrix} \tag{A38}$$

The velocity and position may now be computed using the pseudovelocity coefficients and the recently computed initial velocity:

$$\boldsymbol{\beta}^i = \boldsymbol{V}_0 + \boldsymbol{\beta}^i_{\text{pseudo}}, \qquad \boldsymbol{v}^i(t) = T(\tau)\boldsymbol{\beta}^i \tag{A39}$$

$$\alpha^i = \boldsymbol{X}_0 + [P_2]\boldsymbol{\beta}^i, \qquad \boldsymbol{x}^i(t) = T(\tau)\boldsymbol{\alpha}^i \tag{A40}$$

where

$$\boldsymbol{V}_0 = \overset{N\times n}{\begin{bmatrix} v_1(-1) & \cdots & v_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}$$

$$\boldsymbol{X}_0 = \overset{(N+1)\times n}{\begin{bmatrix} x_1(-1) & \cdots & x_n(-1) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}}$$

and $[P_2]$ is the same as for the second-order initial value problem derivation in the previous section.

Acknowledgments

We thank our sponsors, the U.S. Air Force Office of Scientific Research (Stacie Williams) and the U.S. Air Force Research Laboratory (Alok Das), for their support and collaborations under various contracts and grants. We acknowledge the following individuals who conducted research on the Picard–Chebyshev iteration algorithms and related developments during their time as graduate students at Texas A&M University: Xiaoli Bai, Ahmad Bani

Younes, Donghoon Kim, Brent Macomber, Tarek Elgohary, Julie Read, Austin Probe, Chris Shelton, Nathan Budd, and Ahmed Atallah. Finally, we also acknowledge Satya N. Atluri for sharing his ideas about incorporating an error correction term into the modified Chebyshev–Picard iteration formulation. This work was presented as paper AAS 17-827 at the AIAA/AAS Astrodynamics Specialist Conference.

References

[1] Iserles, A., *A First Course in Numerical Analysis of Differential Equations*, 2nd ed., Cambridge Univ. Press, New York, 2009.

[2] Butcher, J. C., *Numerical Methods for Ordinary Differential Equations*, Wiley, New York, 2008.

[3] Butcher, J. C., "Implicit Runge–Kutta Processes," *Mathematics of Computation*, Vol. 18, No. 18, 1964, pp. 50–64.

[4] Dormand, J., and Prince, P., "A Family of Embedded Runge–Kutta Formulae," *Journal of Computational and Applied Mathematics*, Vol. 6, No. 1, 1980, pp. 19–26. doi:10.1016/0771-050X(80)90013-3

[5] Dormand, J., and Prince, P., "A Reconsideration of Some Embedded Runge–Kutta Formulae," *Journal of Computational and Applied Mathematics*, Vol. 15, No. 2, 1986, pp. 203–211. doi:10.1016/0377-0427(86)90027-0

[6] Watts, H., "Starting Step Size for an ODE Solver," *Journal of Computational and Applied Mathematics*, Vol. 9, No. 2, 1983, pp. 177–191. doi:10.1016/0377-0427(83)90040-7

[7] Gladwell, L., Shampine, L., and Brankin, R., "Automatic Selection of the Initial Step Size for an ODE Solver," *Journal of Computational and Applied Mathematics*, Vol. 18, No. 2, May 1987, pp. 175–192.

[8] Montenbrook, O., and Gill, E., *Satellite Orbits*, Springer, New York, 2000.

[9] Berry, M., "A Variable-Step Double-Integration Multi-Step Integrator," Ph.D. Dissertation, Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, 2004.

[10] Berry, M., and Healy, L., "Implementation of Gauss–Jackson Integration for Orbit Propagation," *Journal of Astronautical Sciences*, Vol. 52, No. 3, 2004, pp. 331–357.

[11] Fox, K., "Numerical Integration of the Equations of Motion of Celestial Mechanics," *Celestial Mechanics*, Vol. 33, No. 2, 1984, pp. 127–142. doi:10.1007/BF01234151

[12] Montenbrook, O., "Numerical Integration Methods for Orbital Motion," *Celestial Mechanics and Dynamical Astronomy*, Vol. 53, 1992, pp. 59–69.

[13] Aristoff, J., Horwood, J., and Poore, B., "Orbit and Uncertainty Propagation: a Comparison of Gauss-Legendre, Dormand–Prince, and Chebyshev–Picard-Based Approaches," *Celestial Mechanics and Dynamical Astronomy*, Vol. 118, No. 1, 2013, pp. 13–28.

[14] Wright, K., "Some Relationships Between Implicit Runge–Kutta, Collocation, and Lanczos τ Methods, and Their Stability Properties," *BIT Numerical Mathematics*, Vol. 10, No. 2, 1970, pp. 217–227. doi:10.1007/BF01936868

[15] Jones, B., and Anderson, R., "A Survey of Symplectic and Collocation Methods for Orbit Propagation," *Advanced in the Astronautical Sciences*, Vol. 143, 2012.

[16] Picard, E., Sur l'application des methods d'approximation successives b latude de certaines bquatioiis differentielles ordinaries, *Journal de Mathématiques*, Vol. 9, 1893, pp. 217–271.

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

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

[19] Macomber, B., "Enhancements of Chebyshev–Picard Iteration Efficiency for Generally Perturbed Orbits and Constrained Dynamics Systems," Ph.D. Dissertation, Texas A&M Univ., College Station, TX, 2015.

[20] Boyd, J., *Chebyshev and Fourier Spectral Methods*, Dover, New York, 2001.

[21] Clenshaw, C. W., "The Numerical Solution of Linear Differential Equations in Chebyshev Series," *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 53, 1957, pp. 134–149.

[22] Clenshaw, C. W., and Curtis, A. R., "A Method for Numerical Integration on an Automatic Computer," *Numerische Mathematik*, Vol. 2, No. 1, Dec. 1960, pp. 197–205.

[23] Clenshaw, C. W., and Norton, H. J., "The Solution of Nonlinear Ordinary Differential Equations in Chebyshev Series," *Computer Journal*, Vol. 6, No. 1, 1963, pp. 88–92. doi:10.1093/comjnl/6.1.88

[24] Feagin, T. W., "The Numerical Solution of Two-Point Boundary Value Problems Using Chebyshev Polynomial Series," Ph.D. Dissertation, Univ. of Texas at Austin, Austin, TX, 1972.

[25] Feagin, T., and Nacozy, P., "Matrix Formulation of the Picard Method for Parallel Computation," *Celestial Mechanics and Dynamical Astronomy*, Vol. 29, No. 2, 1983, pp. 107–115. doi:10.1007/BF01232802

[26] Shaver, J., "Formulation and Evaluation of Parallel Algorithms for the Orbit Determination Problem," Ph.D. Dissertation, Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology, Cambridge, MA, 1980.

[27] Fukushima, T., "Vector Integration of Dynamical Motions by the Picard–Chebyshev Method," *Astronomical Journal*, Vol. 113, 1997, pp. 2325–2328. doi:10.1086/118443

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

[29] Bai, X., "Modified Chebyshev–Picard Iteration for Solution of Initial Value and Boundary Value Problems," Ph.D. Dissertation, Texas A&M Univ., College Station, TX, 2010.

[30] Bani Younes, A., "Orthogonal Polynomial Approximation in Higher Dimensions: Applications in Astrodynamics," Ph.D. Dissertation, Texas A&M Univ., College Station, TX, 2013.

[31] Woollands, R., Bani Younes, A., and Junkins, J., "New Solutions for the Perturbed Lambert Problem Using Regularization and Picard Iteration," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 9, 2015, pp. 1548–1562. doi:10.2514/1.G001028

[32] Woollands, R., "Regularization and Computational Methods for Precise Solution of Perturbed Orbit Transfer Problems," Ph.D. Dissertation, Texas A&M Univ., College Station, TX, 2016.

[33] Wang, X., Yue, X., Dai, H., and Atluri, S., "Feedback-Accelerated Iteration for Orbit Propagation and Lambert's Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 10, 2017, pp. 2442–2451.

[34] Hairer, E., Norsett, S., and Wagner, G., *Solving Ordinary Differential Equations I*, Springer, New York, 1993.

[35] Bellman, R. E., and Kalaba, R. E., "Quasilinearization and Nonlinear Boundary-Value Problems," RAND Corp., Santa Monica, CA, 1965.

[36] Swenson, T., Woollands, R., Junkins, J., and Lo, M., "Application of Modified Chebyshev–Picard Iteration to Differential Correction for Improved Robustness and Computation Time," *Journal of Astronautical Sciences*, No. 3, 2017, pp. 267–284.

[37] Sundman, K. F., "Memoire sur le Probleme Retreint des Trois Corps," *Acta Mathematica*, Vol. 30, 1912, pp. 105–179.

[38] Aristoff, J., and Poore, A., "Implicit Runge–Kutta Method for Orbit Propagation," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA Paper 2012-4880, 2012.

[39] Junkins, J., "Investigation of Finite-Element Representations of the Geopotential," *AIAA Journal*, Vol. 14, No. 6, 1976, pp. 803–808. doi:10.2514/3.61420

[40] Engles, R., and Junkins, J., "Local Representation of the Geopotential by Weighted Orthogonal Polynomials," *Journal of Guidance and Control*, Vol. 3, No. 1, 1980, pp. 55–61. doi:10.2514/3.55947

[41] Arora, N., Vittaldev, V., and Russell, R., "Parallel Computation of Trajectories Using Graphics Processing Units and Interpolated Gravity Models," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 8, 2015, pp. 1345–1355.

[42] Bradley, B., Jones, B., Beylkin, G., Sandberg, K., and Axelrad, P., "Bandlimited Implicit Runge–Kutta Integration for Astrodynamics," *Celestial Mechanics and Dynamical Astronomy*, Vol. 119, No. 2, 2014, pp. 143–168. doi:10.1007/s10569-014-9551-x

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