Numerical Methods and Tolerance Analysis for Orbit Propagation

a project presented to
The Faculty of the Department of Aerospace Engineering
San José State University

in partial fulfillment of the requirements for the degree
Master of Science in Aerospace Engineering

by

Angel Rocha

May 2018

approved by

Dr. Jeanine Hunter
Faculty Advisor

Numerical Methods and Tolerance Analysis for Orbit Propagation

Angel Rocha[1]
San Jose State University, San Jose, CA, 95110, US

Abstract

This report intends to recommend preliminary numerical integrator settings for low to medium fidelity orbit determination models based on orbital elements. Solutions from numerical simulations of propagated satellite orbit trajectory states are first used to quantify the relationship between computational cost of each algorithm and accuracy of each solution. This relationship is generally dependent on integrator characteristics and tolerance settings. Once the relationship between computation cost and accuracy is quantified, orbital element dependencies of each numerical integrator are explored. These dependencies are found to be primarily limited to some combination of eccentricity, semi-major axis, and orbital period or angular velocity. The combined analysis of integrators as well as orbital elements enables a refinement of algorithm recommendations for various types of orbit determination problems.

I. Nomenclature

a = *semi-major axis*
C = *individual state*
δ = *error tolerance*
e = *eccentricity*
ϵ = *error*
h = *step size*
i = *inclination*
k = *increment slope*
n = *step number*
ω = *argument of periapsis*
Ω = *right ascension*
q = *order of integrator*
p = *undetermined step*
ϕ = *incrementation function*
r = *radius*
$\dot{r}$ = *velocity*
$\ddot{r}$ = *acceleration*
a_l = *acceleration due to perturbation force*
s = *number of stages*
τ = *orbital period*
$*$ = *predicted state*
θ = *true anomaly*
μ = *gravitational parameter*
$\hat{\imath}, \hat{\jmath}, \hat{k}$ = *Cartesian unit vectors*
X, Y, Z = *Cartesian position vectors*
VX, VY, VZ = *Cartesian velocity vectors*

II. Introduction

Propagation in astrodynamics is concerned with the determination of trajectory states over time. These states may be propagated by use of numerical integrators which approximate a solution to a nonlinear ODE initial value problem of the general form

$$y'(x) = f(x, y(x)), \qquad y(x_0) = y_0 \tag{1}$$

In this case of orbit propagation, the specific ODE to consider is Cowell's formulation of the Kepler problem which captures two body dynamics in Eq. (2).

$$\ddot{\boldsymbol{r}} = -\mu \frac{\boldsymbol{r}}{r^3} \tag{2}$$

Due to the vectorized formulation of Keplerian dynamics, perturbing accelerations may easily be included to capture the influence of additional bodies, solar radiation, drag, etc. By introducing additional perturbation forces to the simplified two body formulation, the nonlinear ODE of interest may be defined with initial time, position, and velocity state conditions by Vallado [1] as

$$\text{ODE:} \quad \ddot{\boldsymbol{r}} = -\mu \frac{\boldsymbol{r}}{r^3} + \boldsymbol{a}_p(\boldsymbol{r}, \dot{\boldsymbol{r}}, t) \tag{3}$$

$$\text{IC:} \quad t_0, \boldsymbol{r}_0, \dot{\boldsymbol{r}}_0$$

This formulation suggests that the trajectory path then depends on the combined planetary and perturbation forces which act on the system over time. By solving the associated differential equations with numerical solvers, it is possible to propagate or predict trajectories over time. The accuracy of the solution then depends on a combination of factors including model fidelity, quality of numerical integrator, and orbital elements.

III. Background

A. Orbit Types

The orbits to be propagated for this analysis may be considered to belong to one of five broad categories. Four of these are Earth orbiting (EO) while the fifth is a general case for interplanetary missions which can be called Transfer orbits (TO). Each Earth orbit has a unique set of use cases and can be identified in terms of the orbital elements which define them. Three dimensional representations of each orbit are presented as results in Fig. 3 through Fig. 8 for reference. These five orbit categories include:

- **GEO: Geostationary Orbit**
  GEO is defined by a constant altitude of approximately 35,786km. This ensures that the orbit matches Earth's rotation frequency and appears stationary from a perspective on the surface of Earth.
- **HEO: Highly Eccentric Orbit**
  HEO does not specifically depend on orbit altitude and is instead defined by the Keplerian element eccentricity which describes the elliptical shape of an orbit. Orbits with eccentricities in the range $1 > e \geq 0.5$ are typically categorized as HEO.
- **LEO: Low Earth Orbit**
  LEO is defined by a range of orbital altitudes that place a satellite relatively close to the Earth's surface. LEO satellites typically orbit at altitudes between 200km and 2,000km. The minimal altitude gives these satellites very short orbital periods (τ) between 84 and 127 minutes.
- **MEO: Medium Earth Orbit**
  MEO, otherwise referred to as Intermediate Circular Orbit (ICO) encompasses the region of near circular orbital space between LEO and GEO. That is above 2000km and below 35,786km. Many HEO satellites including GOES-14 are placed into semi-synchronous orbits with altitudes of approximately 20,200km. This causes the satellite to orbit with a period of about 12 hours.
- **TO: Transfer Orbit (Mars Transfer Orbit, MTO & Venus Transfer Orbit, VTO)**

In order to simplify mathematical modeling and analyze the relationship between orbital elements and computational cost, TO is considered as a fifth category of orbits. This is accomplished by shifting the inertial frame of a propagated orbit from Earth-centered inertial (ECI) to Sun-centered inertial (SCI). Consequently, TOs are not strictly defined by Keplerian elements but are instead defined by the implementation of SCI as the reference frame of choice for propagation. Although EO and TO are defined relative to inertial reference frame, they tend to have dramatically different semi major axes and orbital periods. While EO have relatively small semi major axes and short orbital periods, TO have very large semi major axes and long orbital periods by comparison.

B. Orbital Elements

The set of orbital elements considered for analysis include the classical Keplerian elements as well as orbital period (τ). Orbital period is included as it provides a method to relate angular velocity (θ) of a satellite orbit to the performance results of various integrators. The Keplerian system is defined by six orbital elements. These elements may be categorized in one of three ways. They describe either the shape and size of the orbit, the orientation of the reference frame, or the position of the particle on the defined orbit. The combined Keplerian elements define an orbit as presented in Fig. 1.

- **Shape and Size of Orbit** (e, a)

  The two elements which define the shape and size of an orbit are the eccentricity (e) and semi-major axis (a). The eccentricity specifies the elliptical shape of the orbit in the range $0 < e < 1$. A value of 0 specifies a perfectly circular orbit while values of $e \geq 1$ relate to hyperbolic projectiles. Semi-major axis specifies the size of the orbit. It is a measure of one-half of the total major axis length from the center of the ellipse through the focus and to one end.

- **Orientation of Orbit** (Ω, i, ω)

  The three elements which define the orientation of an orbit are the right ascension (Ω), inclination (i), and argument of periapsis (ω). Right ascension is an angular measure of the orientation of the orbit from the Line of Nodes which is the linear intersection between the orbit plane and reference (equatorial) plane. Inclination is an angular measure of the tilt of an orbit relative to the reference plane used to define the Line of Nodes. Argument of periapsis, the final orbit orienting element, spins the obit it the plane defined by its right ascension and inclination. It is an angular measure of periapsis from the ascending node.

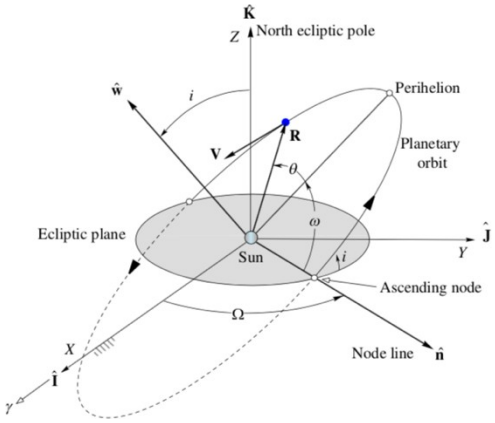

Fig. 1 Planetary Orbit in the heliocentric ecliptic frame [2]

- **Position Along Orbit** (θ)

  The element which defines the position of a particle along its orbit is true anomaly (θ). It is an angular measure between the direction of periapsis and the position of the body with respect to the primary focus of the ellipse.

C. Physical Model & Perturbations

The vector representation of Cowell's formulation of Kepler's problem with perturbing accelerations in Eq. (3) is ideal for modeling in Cartesian reference frames. The advantage of the vector representation is the ability to easily add any number of relevant perturbation forces and n-body interactions. Depending on the level of accuracy required for a simulation, the model fidelity may be scaled accordingly with the addition or omission of perturbing forces. In EO for example it is crucial to include J_2 since the magnitude of the perturbing force due to the oblateness of Earth is large. For TO systems in the SCI frame, however, perturbation due to oblateness is relatively negligible. For TO systems it would be more important to include perturbations due to solar gravity and solar radiation depending on the specific mission. For reference, estimated magnitudes of various perturbing accelerations are provided by Curtis [2]

$$\begin{aligned} a_{/\text{abcde}^{\text{f}}\text{g hijbdakagg}} &\approx 10^{\text{m}`} a_{\text{H}} \\ a_{/\text{jnkbc ocbpqdr}} &\approx a_{/\text{ghjbc ocbpqdr}} \approx 10^{\text{ms}} a_{\text{H}} \\ a_{/\text{ghjbc cbtqbdqhk}} &\approx 10^{\text{mu}} a_{\text{H}} \end{aligned} \tag{4}$$

From Eq. (4) Earth's oblateness (J_2) perturbation is identified as having the greatest magnitude. It is strong enough to produce phenomena including precession of the right ascension of the ascending node and apsidal nodal regression [1]. For this reason, it is critical to model J_2 dynamics for any satellite orbit propagation in EO. A vectorized mathematical model of J_2 is given by [2] as

$$\boldsymbol{a}_{p_{J_2}} = \frac{3J_2\mu R_e^2}{2r^4}\left[\frac{x}{r}\left(5\frac{z^2}{r^2}-1\right)\boldsymbol{i}+\frac{y}{r}\left(5\frac{z^2}{r^2}-1\right)\boldsymbol{j}+\frac{z}{r}\left(5\frac{z^2}{r^2}-3\right)\boldsymbol{k}\right] \qquad J_2 = 0.00108263 \tag{5}$$

Based on the summary of magnitudes of perturbing forces for EO satellites in Eq. (4), a second perturbing acceleration to consider is lunar gravity. The associated dynamics of both J_2 and lunar gravity perturbations are introduced to Eq. (4) which produces an updated formulation of the ODE in Eq. (3) as

$$\ddot{\boldsymbol{r}} = \underbrace{-\mu\frac{\boldsymbol{r}}{r^3}}_{\text{Two-body}} + \underbrace{\mu\frac{\boldsymbol{r}}{r^3}}_{\text{Third-body}} + \boldsymbol{a}_{p_{J_2}}(\boldsymbol{r}, \dot{\boldsymbol{r}}, t) \tag{6}$$

The results presented in this report for EO cases primarily depend on the dynamics of Eq. (6). Some cases of EO and all cases of TO are instead modeled with simplified two body dynamics of Eq. (2).

D. Numeric Integrator Characteristics

- **Explicit vs. Implicit**

  Traditional propagation algorithms estimate states by use of explicit numerical methods. The simple mathematics of explicit numerical methods is presented in Eq. (7). When using an explicit method, the IVP is solved independently at each step by calculating the subsequent state from an existing state. Examples of explicit integrators include the Runge-Kutta (RK) and Dormand-Prince (DP) variations used in this report.

$$y_{k+1} = y_k + hf(t_k, y_k) \tag{7}$$

  Alternatively, implicit methods solve an IVP though a prediction and correction of future states based on both existing and future states. Consequently, implicit methods rely on determination of the state x_{k+1} as a function of itself as presented in Eq. 8. Due to the nonlinearity of system dynamics, implicit methods cannot be solved analytically. Instead, they must use Newtonian iterations to determine a solution for x_{k+1}. The advantage of implicit numeric integration is improved system stability. Stability ensures that truncation errors decay as a propagation moves from one step to the next. This tends to produce superior convergence characteristics but does not necessarily improve solution accuracy. An example of implicit integrators is the Adams-Bashforth-Moulton (ABM) method used in this report.

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}) \tag{8}$$

- **Single-step vs. Multi-step**

  Single-step methods determine subsequent steps solely from calculated information of the most recent previous state. The RK and DP methods considered in this analysis are single step integrators.

  Multi-step methods determine subsequent steps from information of several previously calculated steps. The use of additional existing steps allows for multi-step methods to achieve higher orders of accuracy compared to single-step methods. This is done by determining and correcting the local truncation error at each step. Because multi-step methods continually update future states, it is possible to determine if a step size is small enough to satisfy tolerance conditions for y_{k+1} and simultaneously determine if a step size is large enough to avoid erroneous calculations. This feature optimizes calculations in a way that minimizes computational cost associated with solution determination.

  The ABM integrator considered in this analysis is an example of a multi-step method. Specifically, it uses y_{k-3}, y_{k-2}, y_{k-1}, *and* y_k to generate a solution for y_{k+1}. A caveat to this style of multi-step integration is that the four states y_{k-3}, y_{k-2}, y_{k-1}, *and* y_k must be determined in advance. Therefore, ABM must be initialized with an alternative single-step integrator such as RK or DP.

- **Fixed-step vs. Variable-step**

Classical numerical integrators including the RK4 method rely on fixed-step integration. With fixed-step integrators, the total propagation distance is divided into equal time or distance spaced steps. This simplifies mathematics while sacrificing computational cost and efficiency performance characteristics.

Unlike the constant step size formulation of the RK4 method, modern numerical integrators use prediction-correction mathematics to dynamically adjust step size. Variable-step size algorithms incorporate tolerance criteria to determine the accuracy at each step by considering two methods at each step. This allows for step-size adjustments to be made at each step. RKF45 for example compares a fourth order solution and a fifth order solution in order to calculate local error.

$$\begin{array}{ll} \text{Fixed-step} & h\text{: } constant \\ \text{Variable-step} & h\text{: } not\ constant \end{array} \tag{9}$$

- **RK, DP, ABM**

The Runge-Kutta (RK) family of numeric integrators originate from RKp methods where p represents order of the method. These are explicit single step methods meaning the states $y_{\text{q}\acute{\text{a}}\grave{\text{a}}}$ at time $t_{\text{q}} + h$, are obtained from the equation

$$y_{\text{k}\acute{\text{a}}\grave{\text{a}}} = y_{\text{k}} + h\,\phi(t_{\text{k}}, y_{\text{k}}, h) \tag{10}$$

where ϕ represents an incrementation function that averages multiple derivative evaluations over the time interval $[t_{\text{q}}\text{: } t_{\text{q}} + h]$. The average is obtained through evaluation of the derivative of the ODE of interest at some number of stages s within the specified time interval. For the fourth order RK4 method with $s = 4$ stages, Eq. (10) becomes

$$y_{\text{k}\acute{\text{a}}\grave{\text{a}}} = y_{\text{k}} + h\frac{(k_{\grave{\text{a}}} + 2k_{\cdot} + 2k_{\text{K}} + k_{\text{f}})}{6} \tag{11}$$

with ϕ being substituted for an expression of the weighted average of the four stages evaluated. In this formulation k_{g} represents the s^{de} increment based on the slope (derivative) at a particular subinterval of the time interval being considered. Fig. 2 provides a visual representation of the four slopes evaluated for an RK4 method. The process of evaluating the value of each k_{g} is to first evaluate $k_{\grave{\text{a}}}$, the slope at the originating point in time. This slope is then traced to the midpoint of the time interval where a new slope $k_{\cdot}$ is evaluated. From the originating point in time $k_{\cdot}$ is traced to the midpoint of the time interval where a third slope k_{K} is evaluated. This third slope is traced from the originating point in time to the final point in time where a final slope k_{f} is evaluated. The final slope is then traced from the originating point in time to the final point in time where a new solution is obtained by evaluating Eq. (11) with k_{g}. A more detailed derivation of the RK4 method is provided by [2], however the family of RK methods may be represented in similar ways.

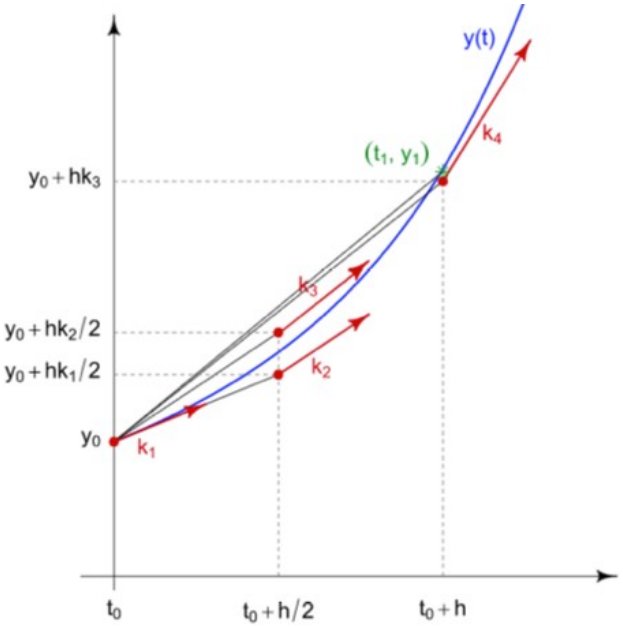

Fig. 2 RK4 Interpolation [1]

Unlike the constant step size nature of the RK4 method, modern adaptions of RK produce variable step-size capabilities by considering two methods at each step. Adaptive step sizes are estimated based on the local truncation error at each step by comparing a q^{de} order solution with a $(q-1)^{\text{de}}$ order solution. In the explicit Runge-Kutta-Fehlberg (RKF) family of algorithms for example, the RKF4(5) method tests a fourth order solution against a fifth order error estimate

[3]. If local error ϵ meets the tolerance δ setting then the higher fifth order solution is propagated forward, otherwise known as local interpolation.

The explicit DP family of integrators, on the other hand, interpolate with the lower order of two solutions once the tolerance criteria is satisfied. In the case of the DP5(4) [4], fourth order interpolation is applied although a fifth order solution is calculated. Since DP is a subclass of RK, they are also traditionally explicit single step methods. Newer mathematical formulations of DP methods, however are considered to have different stability characteristics compared to RKF due to lower order interpolation. Analysis of the stability characteristics of both styles of interpolation in RK and DP is presented in [4][5][6].

The only additional style of integrator used for this analysis is the ABM variation of MATLAB's ODE113 integrator [7]. This method differs from both the RKF and DP families in that it is a variable order method as opposed to a fixed order method. ABM relies on a combination of Adams-Bashforth (AB) for explicit prediction of future states and Adams-Moulton (AM) for implicit correction of predicted states. In practice, AB predicts $y^*_{\text{káà}}$ which is then evaluated as $f(t_{\text{káà}}, y^*_{\text{káà}})$ where the * notation specifies a predicted state. This function evaluation is then inserted into the AM method which corrects $y_{\text{káà}}$ and then evaluates $f(t_{\text{káà}}, y_{\text{káà}})$. The ABM method must satisfy the local truncation error formulation expressed in Eq. (12) which differs slightly from the local truncation error formulation of Eq. (15). The main difference being that Eq. (12) depends on predicted which may change at various steps which is not the case for explicit RK and explicit DP methods. Additionally, the variable order nature of ABM means that the order of the two methods considered for error estimation and interpolation are not necessarily consistent at each step as is the case in RKF and DP. The ABM method is a variable step, variable order integrator of orders 1 to 13.

$$\epsilon_{\text{káà}} \approx \frac{C_{/\text{áà}}}{C^*_{/\overline{\text{áà}}}\, C_{/\text{áà}}} \left(y_{\text{káà}} - y^*_{\text{káà}}\right) \tag{12}$$

In general, higher order integrators produce more accurate solutions for orbit determination. Other considerations for numerical methods include local and global error estimates, stiffness, etc. Analysis of local truncation error can be found by Verner [3] for select RK integrators. Butcher tableaus for each of the integrators used in the analysis are provided in appendix [A].

E. Tolerance Control

Numeric integrators use tolerance settings to control the accuracy of a solution. Integrators from the RKF family run on a single defined tolerance while others, including those from the DOPRI and ABM families distinguish between Absolute tolerance (AbsTol) and Relative tolerances (RelTol). Tolerance is Relative by default and therefore the RKF family of integrators used in this report only utilize RelTol settings.

RelTol specifies the allowable percent error at any step during simulation. Percent error is relative to the states being calculated at each step. Setting RelTol to 1.0e-2 (0.01) thus specifies a 1% error limit relative to each state value at each step. RelTol may be thought of as a control for the number of significant figures which must be accurate for a solution.

AbsTol controls allowable error specifically when the value of a solution approaches 0. It sets a threshold below which the accuracy of a solution may be ignored under the assumption that very small state values have insignificant errors. AbsTol settings therefor specify the decimal place in a solution beyond which variation may be ignored.

When used in conjunction, RelTol specifies the accuracy of a solution as a number of significant digits for each state at each step except for when the absolute error falls below the threshold set by the AbsTol. At each step *i* the numeric integrator estimates local error ϵ for each *j* state to satisfy. If the error tolerance is not initially met, then the integrator must reduce the time step *i* until the calculated error of each state satisfies Eq. (15).

$$\text{RelTol:} \quad \frac{abs(X - Y)}{\min\left(abs(X), abs(Y)\right)} \tag{13}$$

$$\text{AbsTol:} \quad abs(X - Y) \tag{14}$$

$$|\epsilon(i,j)| \leq \max\left(RelTol * |y(i,j)|, AbsTol(i,j)\right) \tag{15}$$

The above formulation requires that state values with large magnitudes have their accuracy determined by the specified RelTol while the state values with small magnitudes have their accuracy determined by the specified AbsTol. Depending on the application, tolerances may either be increased (or loosened) to speed up simulation time at the cost

of accuracy or decreased (or tightened) to increase accuracy at the cost of simulation time. For orbit propagation applications we will see that very small tolerances are necessary in order to produce useful solutions [8].

F. Literature Review

A number of studies which compare various integration methods for the purpose of orbit propagation have been conducted in recent years. These studies commonly focus on optimizing a single integration method for a particular type of orbit and compare the results with a few other methods. Some popular integrators include variations of RK and Gauss-Jackson for orbit analysis. As there are many studies with data on computational cost and accuracy of integrators they will be used to compare results for a wider range of solvers.

A paper by Jones discusses Gauss-Legendre collocation for orbit propagation. Jones describes a variable-step implementation for propagation which is designed to be more effective for eccentric orbits. Variable-time steps are favorable over fixed-time orbits due to the nature of constant acceleration and deceleration along eccentric orbits. By using a fixed distance propagator, accuracy would be lost along the perigee where satellite speed is the fastest. Because orbits are inherently eccentric to some degree, variable step propagators are essential for high fidelity models. Jones continues by comparing the Gauss-Legendre collocation model to ordinary differential equation solvers. Furthermore, this method implements Gauss-Legendre in the form of an implicit RK scheme. The advantage gained by the RK scheme is the development of variable-step techniques which may autonomously determine step sizes based on tolerances. One of the models that Jones compares the Gauss-Legendre collocation results against is the DOPRI 8(7) and DORPI 5(4) methods which also implements step size control. An additional advantage by the implicit RK scheme used by Jones is parallelization. The majority of explicit methods cannot utilize multi core processing and suffer from long computation times on the force mode. The conclusions drawn from this report indicate that integration with Gauss-Legendre nodes with variable-step implementation outperformed DP 8(7) and 5(4) embedded RK, but not the Gauss-Jackson 8 integrator r= r3rin terms of computational cost for circular orbits. For Molniya orbits however, this method outperformed DP 5(4) and Gauss-Jackson 8 while matching DP 8(7).

Another paper by Berry and Healy specifically compares speed and accuracy of the variable-step Stormer-Cowell Integrator. Like the Gauss Legendre collocation method, this integrator utilizes autonomous step size control from local error approximations. Berry and Healy then compare results of the Stormer-Cowell method with two Gauss-Jackson methods and the Shampine Gordon method. There is a focus in this paper around multi-step integrators which are designed to be faster that single-step integrators. Additionally, double-integration methods have the advantage of computing second-order differential equations such as the Cowell second order formulation of two body equations of motion. This is in contrast to single integration methods which solve first-order differential equations and must be applied twice in order to compute the same variables.

More recently, Jones and Anderson have explored both symplectic and collocation methods for orbit propagation. The symplectic method is examined as it preserves the Hamiltonian and tends to reduce integration error as a result of truncation. This allows for large time steps during integration which reduces overall computational cost while maintaining accuracy. Node spacing for various collocation methods is also explored in this report as it relates to the varying distance between time steps. Gauss, Lobatto, and Chebyshev nodes are explored which all have variations in node density for propagation.

IV. Propagation Parameters

A. Satellites

Six satellites are propagated for this analysis for each of the EO cases while two satellites are propagated for TO cases. Initial conditions for each satellite are cartesian states retrieved from JPL Horizons [9] as ephemeris data. The four Earth orbiting satellites are propagated from ephemeris data for 2018-Jan-01 while MTO and VTO are propagated from 2011-Dec-01 and 2005-Dec-01, respectively. Table 1 presents a summary of initial orbital elements.

Initial Orbital Data							
Orbit	**Satellite**	$\tau\ (min)$	e (deg)	$a\ (km)$	$\Omega\ (deg)$	$i\ (deg)$	$\omega\ (deg)$
GEO	GOES-14	1436.1	0.00100297	42166	351.341	0.0337605	251.33
HEO	MMS-4	4053.0	0.910034	6778	308.958	19.0751	161.86
LEO	ISS	92.7	0.00123243	26562	131.92	51.6956	64.97
MEO	NAVSTAR-68	718.0	0.00529487	84187	32.1793	55.9997	20.22
MTO	MSL	750601 (~521 days)	0.223352	1.90e08	6.21437	0.396572	1.00094
VTO	VEX	410678 (~285 days)	0.171225	1.27e08	6.2748	0.406136	3.79567

Table 1 Initial Orbital Data

B. Variables

- **Initial States**

  Six sets of initial states which represent the satellites listed in Table 1 are considered for propagation. Further discussion on accuracy and computational cost of a solution for different orbit types, thus various initial orbital elements, are provided by Aristoff [10]. By association, initial states are explored to quantify this relationship.

- **Numerical Integrator**

  Six explicit numerical integrators are used to propagate the set of states presented for each satellite listed in Table 1. While implicit integrators are known to have favorable performance characteristics for many orbit propagation applications [10], explicit integrators are employed to reduce the complexity of the presented problem by reducing the number of variables under consideration.

  The RKF45 integrator is taken from [2] while the RKF 89 is an adaptation of the RKF45 formulation to fit the higher order terms. Integrators ODE45 and ODE113 are a part of MATLAB's ODE suite [7] while DOPRI54 and DOPRI87 are taken from [11] and [12]. Note that DOPRI54 and ODE45 are based on the same DP integrator mathematics. The programming for the two methods differs and provide different results.

Numerical Methods
RKF45
RK54 (ODE45)
RK54 (DOPRI54)
RK87 (DOPRI87)
RKF89
ABM (ODE113)

Table 2 Numerical Methods

- **Tolerance**

  The combined range of tolerances used for propagation in this report is $1e-4 \leq \delta \leq 1e-16$.

- **Model Fidelity**

  Two model fidelities are used to analyze the effect of additional perturbations on computational cost and tendency of a solution to converge. This only applies to the EO satellites as the applied perturbations have greater relative effects in the vicinity of Earth as opposed to solar system scale. The TO satellites are propagated with simple two body dynamics.

Physical Models	
Two-Body	Three-Body + J2
All satellites	EO satellites only

Table 4 Physical Models

- **Propagation Time & Propagation Distance**

  A single propagation time of $\Delta t = 25$ days (600 hours) is modeled across all orbits. Results for shorter propagation times are not presented as they do not accurately capture long term trends for variation of previously stated variables.

  Additionally, a single angular propagation distance of $\Delta\theta = 100$ orbits ($100*2\pi$) is modeled across all orbits. The wide range of orbital periods across the six satellites forces numerical integrators to propagate through vastly different final times. The purpose of considering constant angular distance is to isolate results for variation of Keplerian elements without a dependence on physical time.

Propagation Time & Distance
$\Delta t = 25 days$ (2,160,000s)
$\Delta\theta = 100 orbits$ ($100 * 2\pi$)

Table 5 Propagation Time & Distance

V. Results

Two graphical sets of data are presented for each propagation. The first is a set of computational cost as a function of tolerance. This includes total run time, number of steps for a solution, and number of erroneous steps for each solution. The second is a set of converge results for estimated states. For convergence results, two variables are measured as a function of computational cost. One is the actual state value of each solution while the other is the log scaled difference between the states of the current tolerance solution and the states of a reference solution. The resulting value give the decimal place accuracy with respect to the reference solution. This is further discussed in the analysis of "Accuracy of Integrator Solutions."

A. 3D Models

Three dimensional models of each satellite orbit are presented to illustrate the scale of each orbit. Important aspects to note are the size and shape of each ellipse which relate to semi-major axis and eccentricity measures.

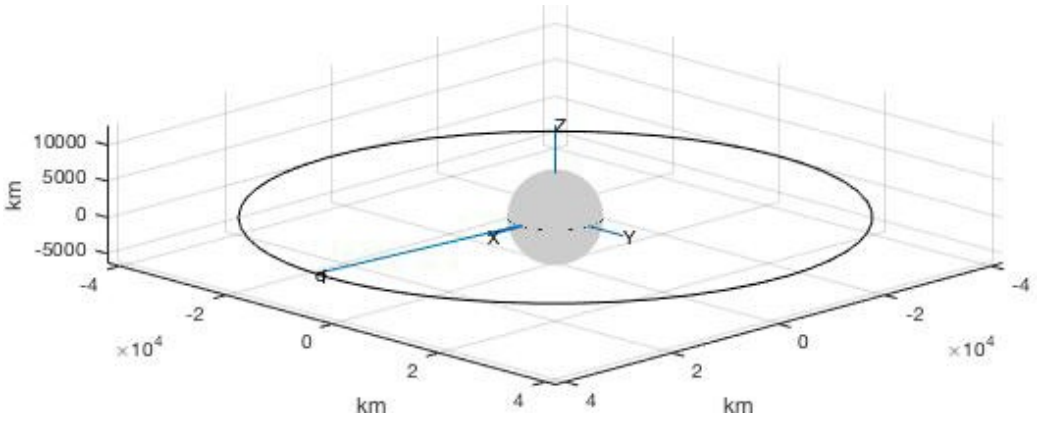

Fig. 3 GEO 3-D Orbit

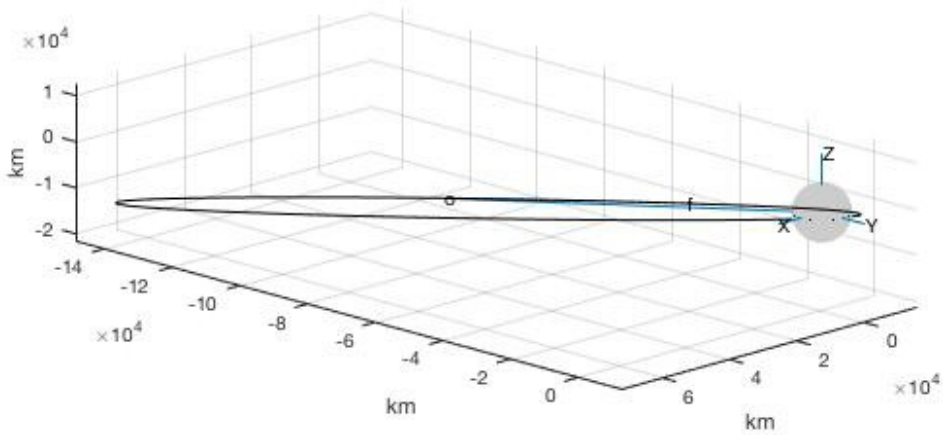

Fig. 4 HEO 3-D Orbit

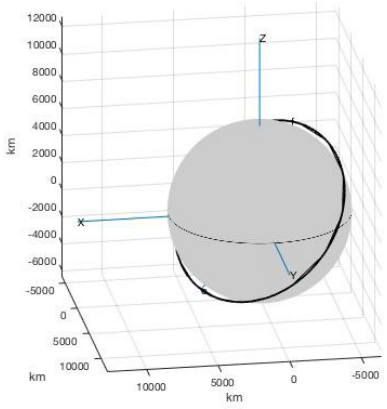

Fig. 5 LEO 3-D Orbit

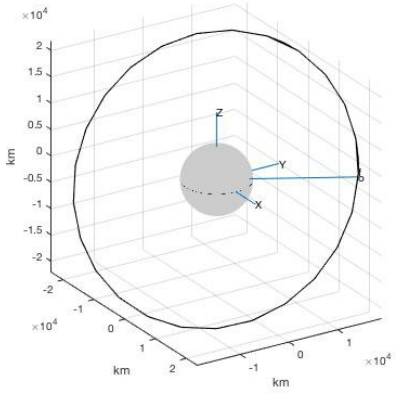

Fig. 6 MEO 3-D Orbit

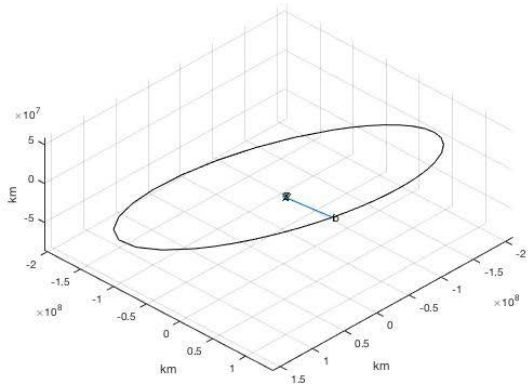

Fig. 7 MTO 3-D Orbit

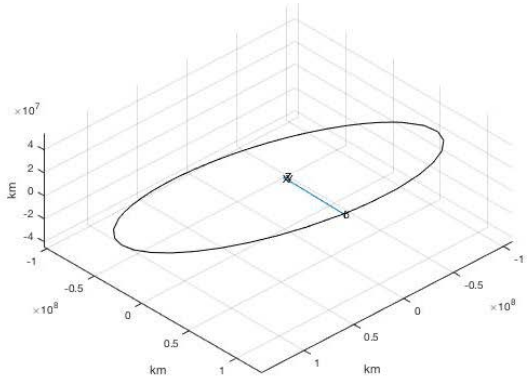

Fig. 8 VTO 3-D Orbit

B. Absolute & Relative Tolerance

To analyze the importance of AbsTol and RelTol in orbit determination, a single set of initial states are propagated with one tolerance held constant at $\delta = 1e-8$ and the other varied from $1e-4 \leq \delta \leq 1e-12$. This data set is propagated from the LEO ephemeris for 600 hours using ODE45 with a physical model which includes both lunar and J2 perturbations. The AbsTol and RelTol relations observed for ODE45 are representative of all other integrators considered in this report and may be applicable to orbit determination problems in general.

In both the computational and convergence results presented in Fig. 9 and Fig. 10 there is a crossover between data sets at 1e-8 where both the RelTol and AbsTol are $\delta = 1e-8$, thus producing equivalent results. In the case varying AbsTol there is an early plateau in run times due to the plateau in computational steps as seen in Fig. 9. The convergence results reveal an inability for ODE45 to converge when reducing AbsTol below $\delta = 1e-6$. Recall that AbsTol specifies decimal place accuracy and this plateau can be attributed to the large magnitude of orbital states. For the X position state which is measured in the thousands of kilometers, AbsTol response at $\delta = 1e-6$ reflects ten significant figures of control. The exact number of significant figures that each integrator is responsive to does vary primarily depending on the order of the integrator.

In the case varying RelTol there is a consistent increase in run times due to increased computational steps at each tolerance step. The convergence results reveal continued convergence through the minimum tested RelTol of 1e-12. Recall that RelTol specifies percent error accuracy and this trend reveals that solution accuracy may be manipulated at stringent RelTol settings. Based on VX results, a minimum RelTol setting of $\delta = 1e-10$ may be recommended due to the ability of the solution to continue converging at tighter tolerances. RelTol settings which are less stringent would not necessarily ensure accurate results.

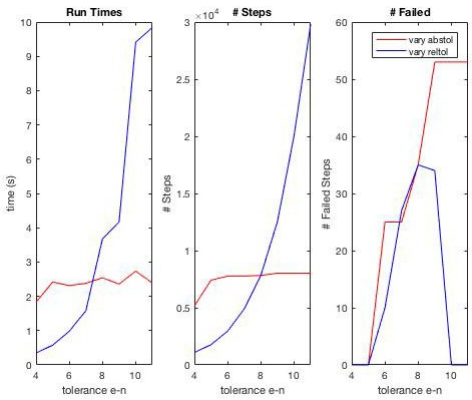

Fig. 42 Vary Tolerance: LEO, ODE45, 600hour, J2+Lunar (Computational Results)

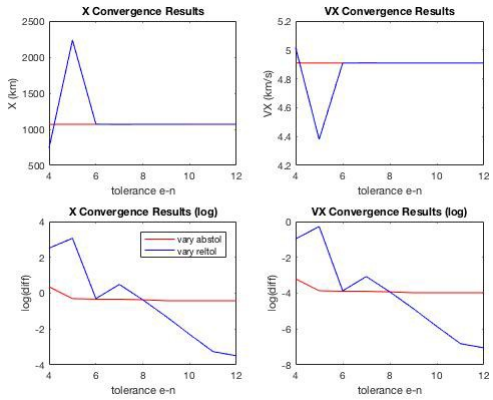

Fig. 43 Vary Tolerance: LEO, ODE45, 600hour, J2+Lunar (Convergence Results)

C. Accuracy of Integrator Solutions

This accuracy analysis begins with a discussion of the mathematical process used to quantify convergence and accuracy of integrator solutions throughout the report. This accuracy analysis is independent of computational cost associated with obtaining each solution. Table 6 presents final state data of a single set of LEO ephemeris propagations. These solutions are for a 24-hour propagation using the perturbation model of Eq. (6) which includes J2 and Lunar gravity with a RelTol setting of $\delta = 1e-10$ for all numerical integrators. The first row of data in Table 6 presents final ephemeris data from JPL Horizons while the following rows present final propagation states for each integrator. Table 7 then presents absolute final state error between the ephemeris states and the calculated states for each integrator by taking the difference between the states.

The X,VX, ΔX, and ΔVX data is then taken from Table 6 and Table 7 and graphed in Fig. 11 to show a graphical representation of convergence trends. The data in Table 6 and Table 7 only generates the data points at the tolerance of $\delta = 1e-10$. The rest of the data points are taken from solutions from each integrator at each of the tolerances in the range $1e-4 \leq \delta \leq 1e-12$. While the first row of graphs in Fig. 11 simply presents final states, the second row of graphs are calculated by taking the mathematical log of the absolute final state errors as show in Eq. (16). Plotting the log scaled difference presents a visual representation of the decimal place accuracy of each solution. For example, the "VX Convergence Results (log)" plot reveals one decimal place of accuracy for the solution of RKF45 integrator using a RelTol of $\delta = 1e-6$. The graph then reveals two decimal places of accuracy for the solution of RKF45 integrator using RelTol in the range $1e-8 \leq \delta \leq 1e-12$ since the resulting value settles at -2.

$$\log(diff)_{™} = \log_{àH}(\Delta X), \quad \log(diff)_{™} = \log_{àH}(\Delta VX) \tag{16}$$

State	X	Y	Z	VX	VY	VZ
EPH	**-3199.720073060143**	**-2728.382385352880**	**5314.021052617424**	**4.748436158196570**	**-6.018782640021895**	**-0.2282717591541034**
RKF45	-3207.436708283496	-2722.050539630536	5316.027065456157	4.740696716530949	-6.019181707087806	-0.2201508800557680
ODE45	-3207.804466052152	-2722.334836360222	5316.617135270371	4.741209424356703	-6.019856467125762	-0.2201565787853040
DOPRI54	-3207.804553745828	-2722.334726954116	5316.617140827567	4.741209348864262	-6.019856529796083	-0.2201564541305700
RKF89	-3195.490858013873	-2712.721129199226	5296.787748279240	4.723928954444030	-5.997203697572299	-0.2198964632547130
ODE87	-3207.804058366658	-2722.335344953416	5316.617109406212	4.741209775409592	-6.019856175909346	-0.2201571582897830
ODE113	-3207.804281190570	-2722.335066669972	5316.617123310268	4.741209584125698	-6.019856335383273	-0.2201568419190540

Table 6 Final States: LEO, 24hour, J2+Lunar, $\delta = 1e-10$

State	ΔX	ΔY	ΔZ	ΔVX	ΔVY	ΔVZ
RKF45	7.716635223352569	-6.331845722344042	-2.006012838733113	0.007739441665621	0.000399067065911	-0.008120879098335
ODE45	8.084392992009271	-6.047548992657994	-2.596082652946279	0.007226733839867	0.001073827103867	-0.008115180368799
DOPRI54	8.084480685685321	-6.047658398763815	-2.596088210143535	0.007226809332308	0.001073889774188	-0.008115305023534
RKF89	-4.229215046269928	-15.661256153653994	17.233304338185008	0.024507203752540	-0.021578942449596	-0.008375295899390
ODE87	8.083985306514478	-6.047040399464095	-2.596056788787791	0.007226382786978	0.001073535887451	-0.008114600864320
ODE113	8.084208130427214	-6.047318682907644	-2.596070692843568	0.007226574070873	0.001073695361378	-0.008114917235049

Table 7 Final State Error: LEO, 24hour, J2+Lunar, $\delta = 1e-10$

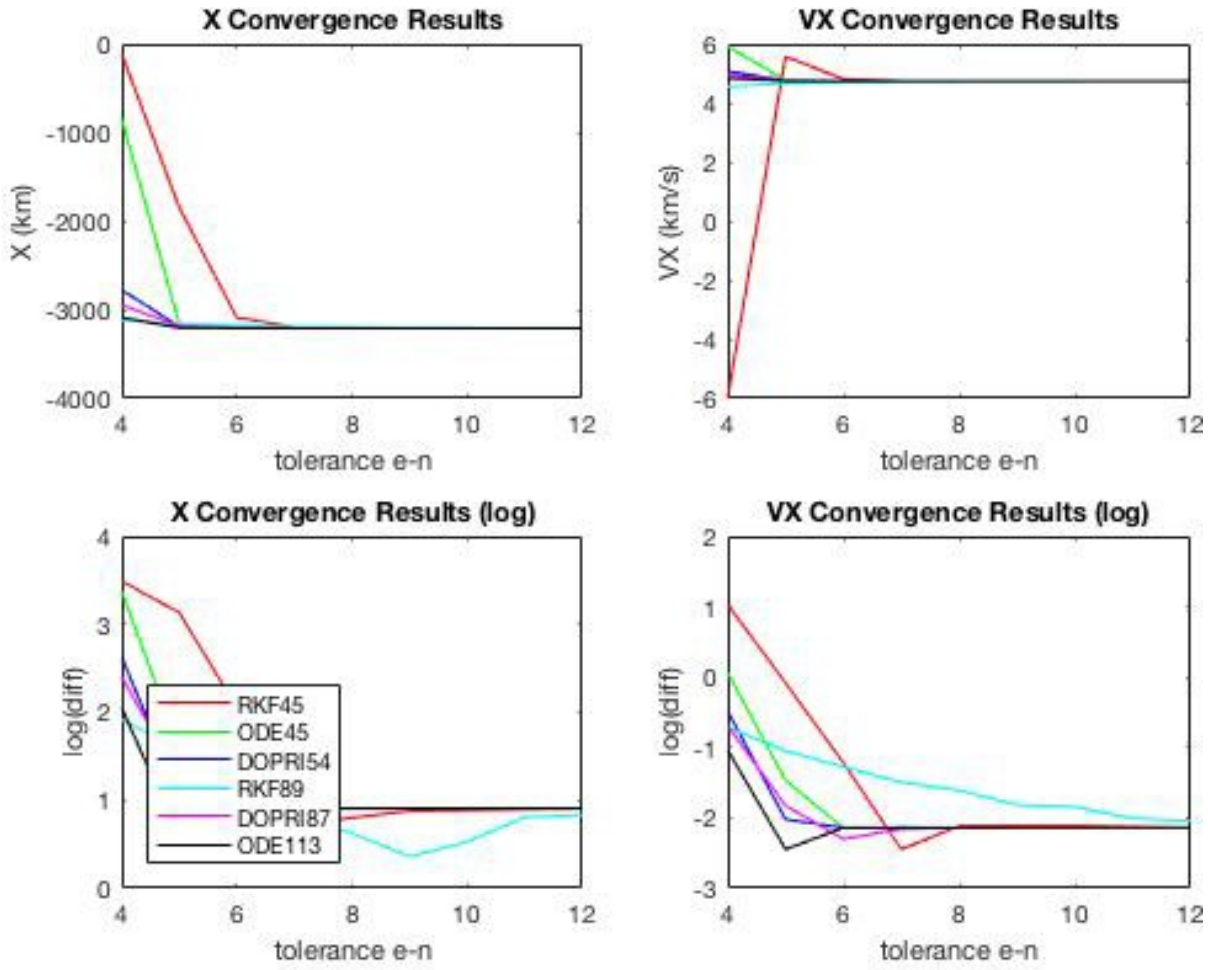

Fig. 11 Convergence Results: LEO, 24hour, J2+Lunar, $\delta = 1e-10$

Using the data of Table 7, the magnitudes of the vectorized displacement error (ΔV, ΔY, ΔZ states) and vectorized velocity error (ΔVX, ΔVY, ΔVZ) are calculated and presented in Table 8. This data set exposes RKF89 as an outlier in terms of solution accuracy for the LEO propagation as it has over twice the position error and three times the velocity error. To address this issue, an additional set of propagation error data for a RelTol setting of $\delta = 1e-14$ is presented in Table 9. This data set shows that RKF89 requires tighter RelTol settings in order to achieve the same level of accuracy as the rest of the integrators. Beyond $\delta = 1e-14$ there is no further minimization of absolute error for any of the integrators. For this reason, the remaining propagations for accuracy analysis are calculated using RelTol in the range $1e-8 \leq \delta \leq 1e-14$.

State	Δr	$\Delta \dot{r}$
RKF45	10.181493845748481	0.011225279023188
ODE45	10.424485791096648	0.010919475219049
DOPRI54	10.424618652986965	0.010919623986092
RKF89	23.667445629608348	0.033710523199942
ODE87	10.423868134205462	0.010918783571113
ODE113	10.424205839820633	0.010919160968800

Table 8 State Error Magnitude: LEO, 24hour, J2+Lunar, $\delta = 1e-10$

State	Δr	$\Delta \dot{r}$
RKF45	10.414561874631506	0.010931086423464
ODE45	10.424278419121871	0.010919242984191
DOPRI54	10.424278441465431	0.010919243009171
RKF89	9.841156350946937	0.012256465525769
ODE87	10.424278343879379	0.010919242899941
ODE113	10.424278377056831	0.010919242936969

Table 9 State Error Magnitude: LEO, 24hour, J2+Lunar, $\delta = 1e-14$

Of the four EO cases considered, GEO is the most accurate across all integrators when comparing a two-body propagation to ephemeris data. The associated magnitudes of displacement error and velocity error for 25-day, two-body propagations is presented in Fig. 12 and Fig. 13. Even with GEO being the most accurate, the absolute errors for two-body dynamics make the data unusable for practical purposes. The associated converged errors are approximately 430km for position and 0.031km/s for velocity. For comparison, the second most accurate orbit type is MEO with an absolute position error that settles above 740km. It should be noted that each of the integrators eventually converge to solutions with nearly identical absolute errors for both position and velocity states as shown in Fig. 12 and Fig. 13. This trend occurs for all four EO types when using two-body dynamics. The primary factor to consider for simple two-boy dynamics is tolerance. For such propagations it would be best to use easily implemented integrators such as RKF45 and ODE54 with tolerances of between $1e-10 \leq \delta \leq 1e-14$.

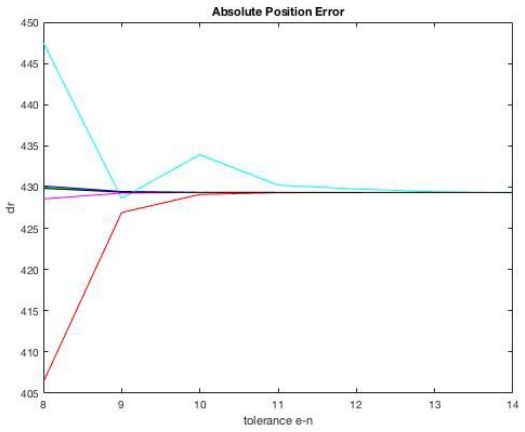

Fig. 12: Position Error: GEO, 25day, 2-body

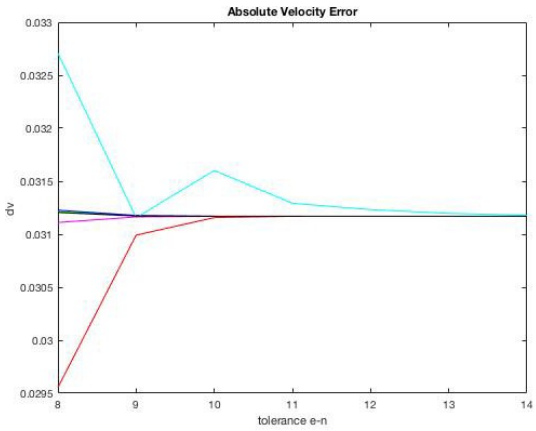

Fig. 13: Velocity Error: GEO, 25day, 2-body

The remaining propagations for EO accuracy analysis consider 1-day, 5-day, 10-day, and 15-day propagations using a force model which includes two-body dynamics (earth and satellite) with the addition of J_2. For these propagations we will consider only the absolute position error as a measure of accuracy for each solution. For the 1-day propagations we once again see RKF integrators providing less accurate solutions across all cases except the for a short range of tolerances in the HEO case of Fig. 15. Once all of the errors converge at a RelTol of \delta $= 1e-14$ we see nearly equivalent accuracies across all integrators. In Fig. 16 we also see slightly improved accuracy for RKF45 compared to other integrators at low RelTol when propagation LEO. This behavior is amplified for long propagation times in subsequent results. Across all data sets we see that ODE45, DOPRI54, DOPRI87, and ODE113 produce nearly identical position errors at all tolerances. These position errors are also converged at low tolerances unlike RKF integrator solutions which require tighter tolerances.

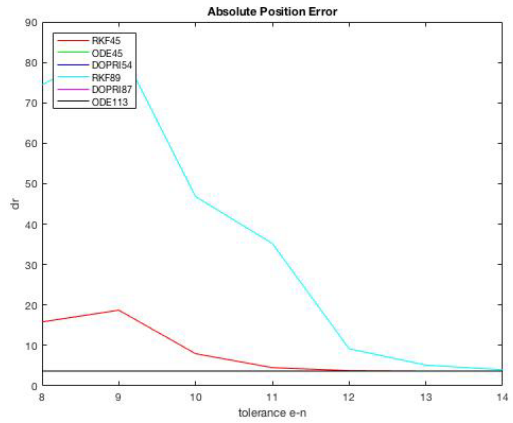

Fig. 14 Position Error: GEO, 1day, 2-body

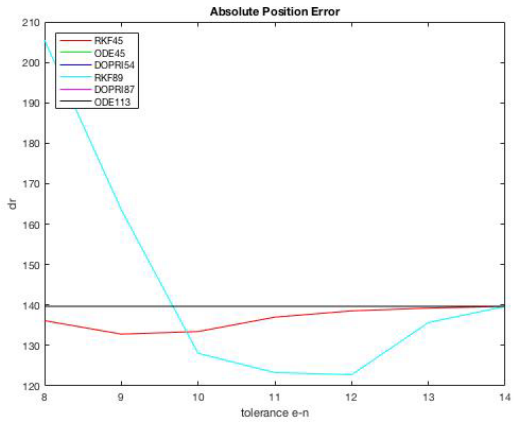

Fig. 15 Position Error: HEO, 1day, 2-body

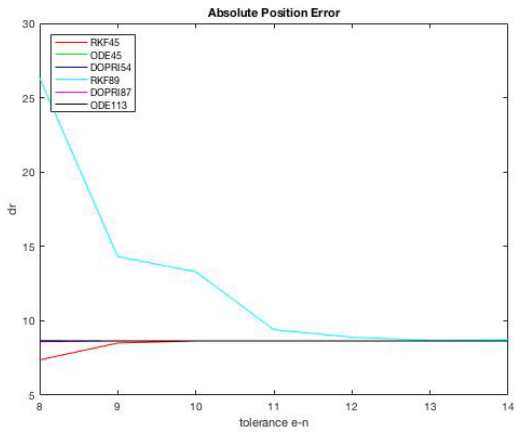

Fig. 16 Position Error: LEO, 1day, 2-body

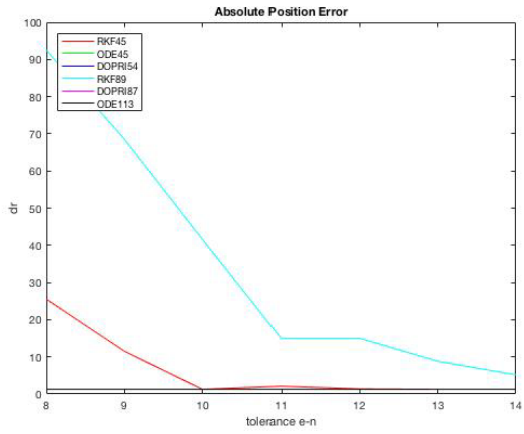

Fig. 17 Position Error: MEO, 1day, 2-body

The 5-day propagation results reiterate the deviation of RKF solutions compared to DP and ABM. In all EO cases we see that RKF89 performs poorly at equivalent tolerances. The exception to this is the LEO 5-day propagation where the difference in absolute position error for RKF89 is much smaller. Also in the LEO results we see that RKF45 has better performance than ABM, DP, and RKF89 at low RelTol of $\delta \leq 1e-10$. This is an amplification of what is seen in the 1-day propagation solutions. Once the recommended minimum RelTol of $\delta = 1e-10$ is implemented we see that all solutions except RKF89 have nearly identical absolute errors.

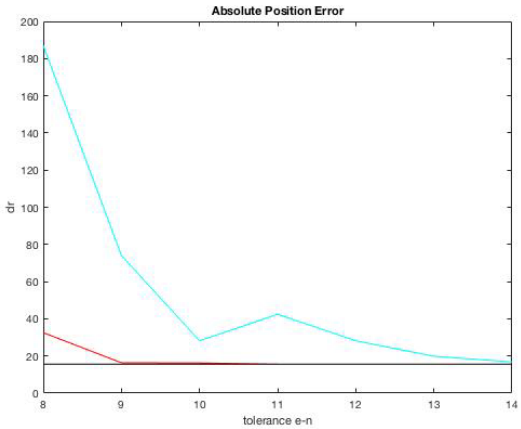

Fig. 18 Position Error: GEO, 5day, 2-body

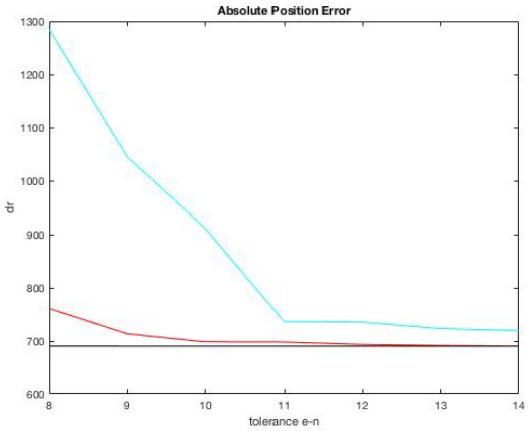

Fig. 19 Position Error: HEO, 5day, 2-body

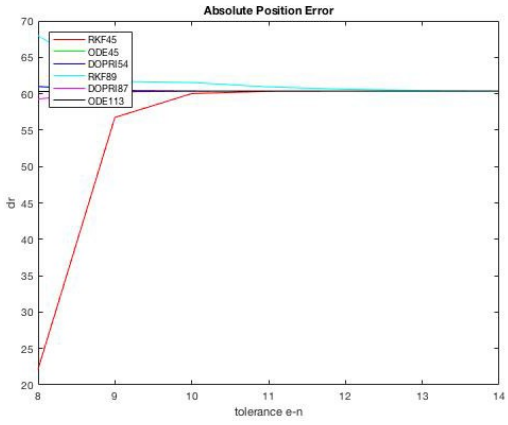

Fig. 20 Position Error: LEO, 5day, 2-body

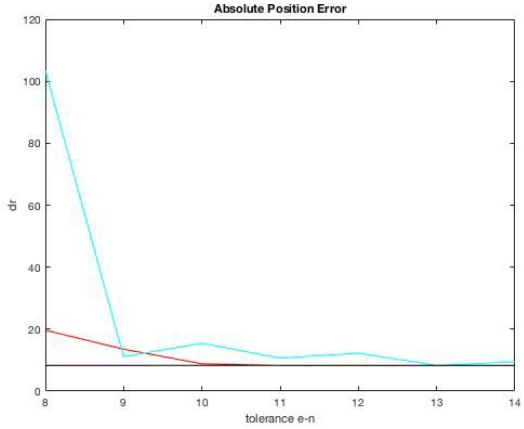

Fig. 21 Position Error: MEO, 5day, 2-body

The 10-day propagation results introduce an anomaly in the trends discussed for the 1-day and 5-day propagations. This is specifically for the HEO case where Fig. 23 shows both RKF45 and RKF89 producing significantly better results than DP and ABM. It is important to note that RKF89 with a RelTol of $\delta = 1e - 8$ does not fit this generalization as its absolute position error is approximately 50% greater than the average. If we consider the minimum recommended tolerance of $\delta = 1e - 10$ then both RKF integrators produce equivalent or superior accuracy for HEO results. From these results we may want to conclude that RKF handles highly eccentric orbits better than DP and ABM and moderate tolerance setting as propagation time increases, however the trend of error growth contradicts this conclusion. When analyzing Fig. 23 HEO 10-day propagation results we must compare the absolute error to those in Fig. 15 and Fig. 19 for 1-day and 5-day propagations. 1-day propagations present a converged absolute error of approximately 140km which is already significant. This grows to nearly 700km for the 5-day propagation. Because the 10-day propagation presents a smaller absolute position error we must conclude that reasonable results cannot be obtained using any integrator for HEO with a 10-day propagation using the current force model.

The large magnitudes of error presented in theis10-day propagation make it impossible to generate reasonable results with the 2-body force model with J_2 perturbations, however it provides insight into the various performance capabilities of each integrator across various EO types. We see that the RKF45 integrator provides better accuracy than others at loose tolerances for LEO propagations. We also see that in all other cases DP and ABM have superior convergence results and generally produce more accurate results.

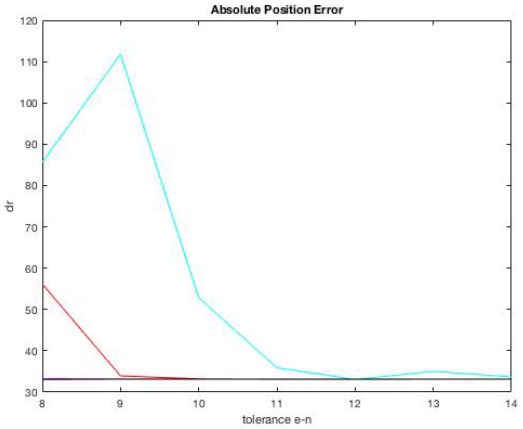

Fig. 22 Position Error: GEO, 10day, 2-body

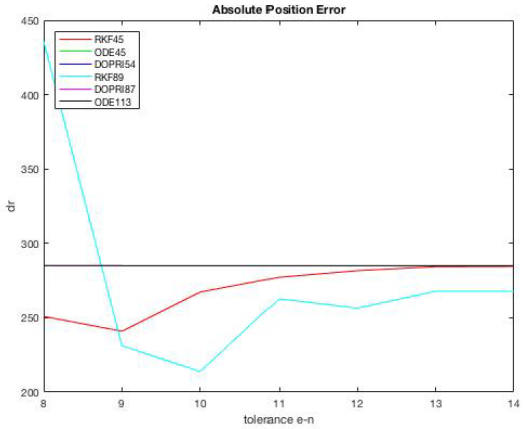

Fig. 23 Position Error: HEO, 10day, 2-body

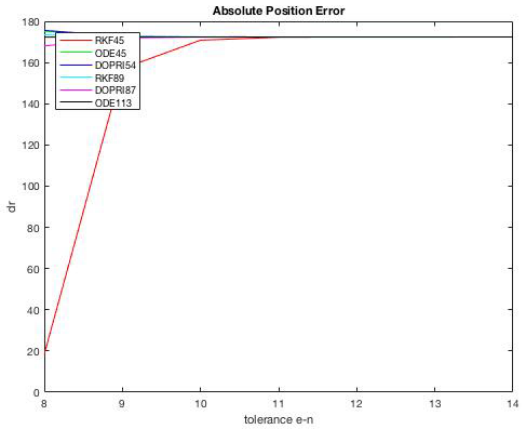

Fig. 24 Position Error: LEO, 10day, 2-body

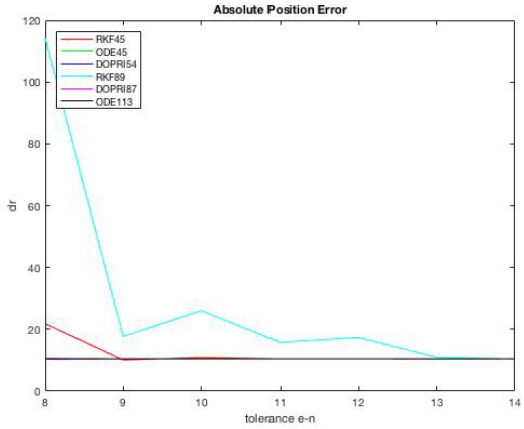

Fig. 25 Position Error: MEO, 10day, 2-body

D. Model Fidelity

Two physical models are considered for the propagation of the four EO satellites in order to analyze the relationship between model fidelity and computational cost at various tolerances. General trends for each of the four cases may be summarized by LEO results presented in Fig. 26 through Fig. 33. The missing data point at $\delta = 1e - 4$ for two body

propagation in Fig. 26,28,30,32 is caused by to one or more numerical integrators failing at such large tolerance settings for a large model. RKF45 is usually the integrator which fails at extremely loose tolerances.

A comparison of the computational results of both models in Fig. 26 and Fig. 27 show intuitive results that higher fidelity models require smaller steps to be taken in order to meet tolerance requirements. The smaller step sizes result in at least 50% more steps needed to propagate with the added J_2 and Lunar perturbations. Consequently, run times see anywhere from a 50%-100% increase across all integrators for the higher fidelity model. The only integrator which seems to experience a proportional increase in number of failed steps is the RKF45 integrator. All others appear to experience similar numbers of failed steps for each model fidelity.

A comparison of the convergence results for final position states in Figs. 28-33 reveal that convergence for all states does not occur until at least a tolerance of $\delta = 1e-10$. This applies to both physical models. When considering the log scale results relative to a solution with tolerance $\delta = 1e-13$, reasonable convergence does not occur until a tolerance of $\delta = 1e-12$. For position states this is when the log scale difference falls below 0 meaning accuracy at the decimal place is achieved. Because position states are measured in the thousands of kilometers, this represents four significant figures of accuracy compared to the lowest tolerance solution. For velocity states this is when the log scale difference falls below -2 meaning accuracy at the hundredth's place is achieved. Because velocity states do not exceed the ones place in km/s measurements this represents three significant figures of accuracy compared to the lowest tolerance solution. Log scale difference convergence results appear identical between both physical models.

As with the bulk of previously discusses solutions, the log scale results in Figs. 28-33 show that DP and ABM converge more quickly than RKF. At lower tolerances we see that DP models show the best convergence while at higher tolerances, ABM shows the best convergence. In all cases RKF have the worst convergence with agrees with the fact that RKF requires tighter tolerances to converge.

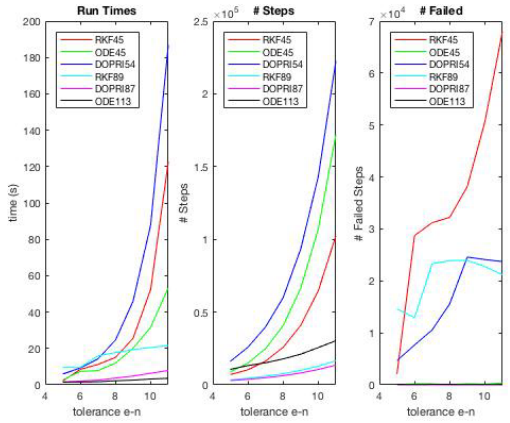

Fig. 26 Computation Results: LEO, 600hr, 2body

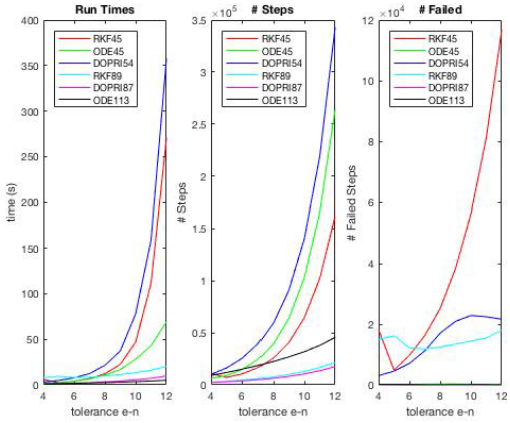

Fig. 27 Computation Results: LEO, 600hr, J2+Lunar

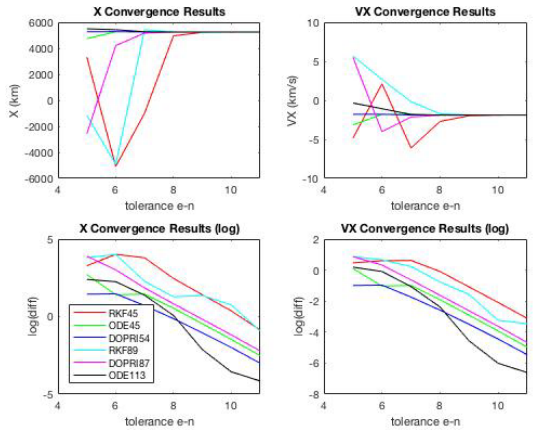

Fig. 28 X & VX Convergence Results: LEO, 600hr, 2body

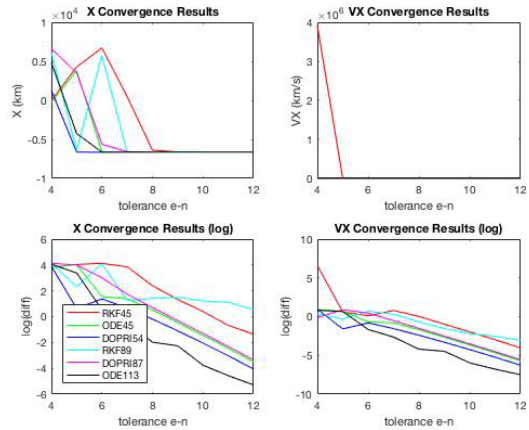

Fig. 29 X & VX Convergence Results: LEO, 600hr, J2+Lunar

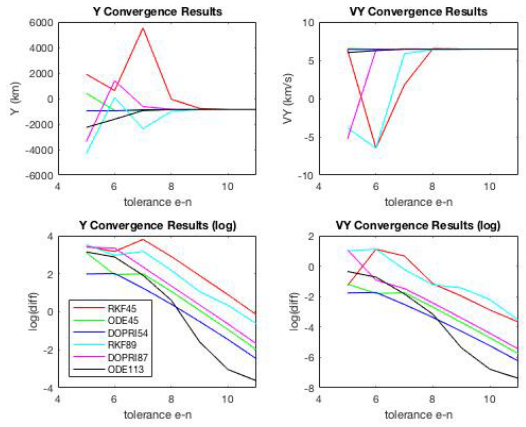

Fig. 30 Y & VY Convergence Results: LEO, 600hr, 2body

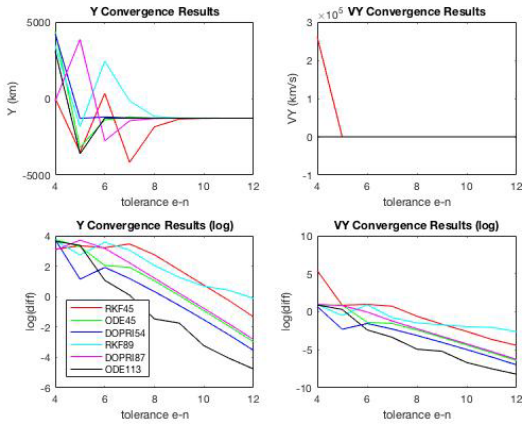

Fig. 31 Y & VY Convergence Results: LEO, 600hr, J2+Lunar

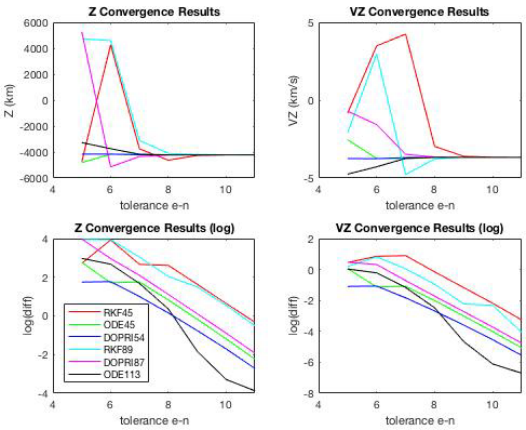

Fig. 32 Z & VZ Convergence Results: LEO, 600hr, 2body

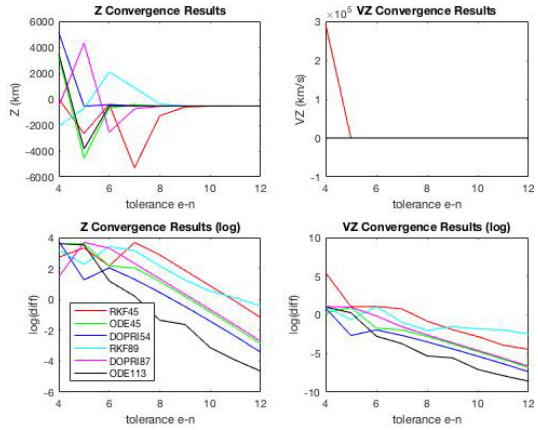

Fig. 33 Z & VZ Convergence Results: LEO, 600hr, J2+Lunar

E. Earth Orbit

The four EO propagations show a wide range of computational costs in terms of number of steps to solution and run times. For the 25-day propagation, run times vary from about a half of a second for the fastest integrator to around 350 seconds for the slowest integrator. In order of increasing run times, the EO cases are as follows: HEO, GEO, MEO, LEO with LEO being the outlier of the set. This is attributed to a strong dependency on orbital period τ. Table 10 lists run times and τ for the EO cases in increasing order of maximum run times. Maximum run times from DOPRI54 are considered to be representative of the trends across all other solution sets. The trend of lower order solutions taking longer to generate a solution in Figs. 34,36,38 and 40 is common across all propagation results when time is help constant. Consequently, computational time becomes more important as the number of orbits increases.

General conclusion to be drawn from the EO propagation data is that computational time is primarily a function of angular distance traveled ($\Delta\theta$). LEO, with the largest computation cost has the smallest orbital period by nearly a factor of 10. This means that for any given period of time it will complete 10 times the number of orbits compared to MEO and over 43 times the number of orbits compared to HEO. This trend is less important for Transfer Orbits.

EO Period & Computation Time		
Orbit	$\boldsymbol{\tau}$ **(min)**	$\boldsymbol{t_{comp}}$ **(sec)**
HEO	4053.0	~3.1
HEO	1436.1	~5.0
MEO	718.0	~12.5
LEO	92.7	~350

Table 10 Earth Orbit Period & Computation Time

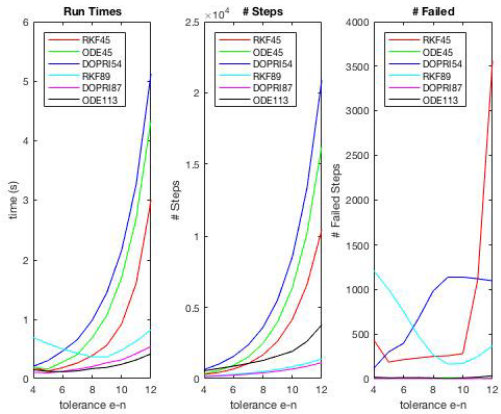

Fig. 34 Computation Results: GEO, 600hour, J2+Lunar

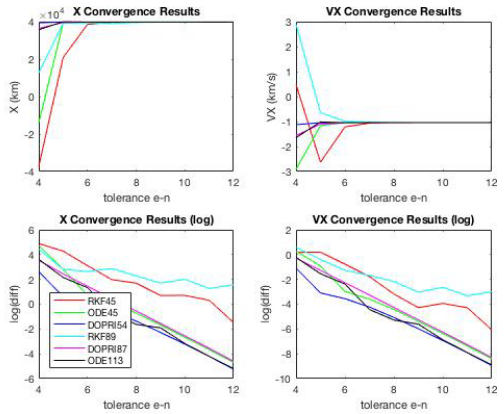

Fig. 35 X & VX Convergence Results: GEO, 600hour, J2+Lunar

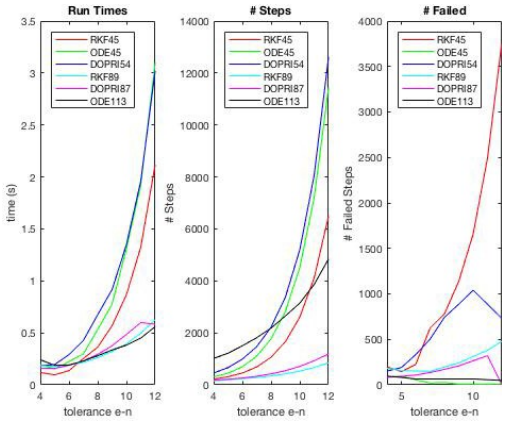

Fig. 36 Computation Results: HEO, 600hour, J2+Lunar

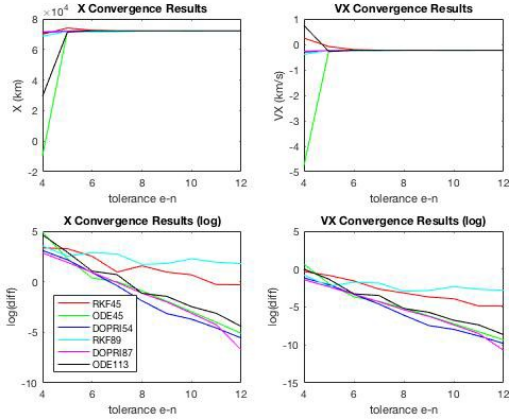

Fig. 37 X & VX Convergence Results: HEO, 600hour, J2+Lunar

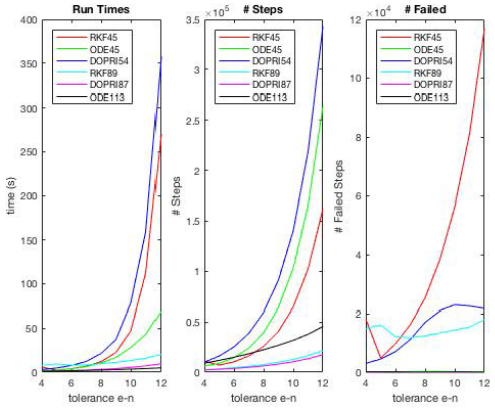

Fig. 38 Computation Results: LEO, 600hour, J2+Lunar

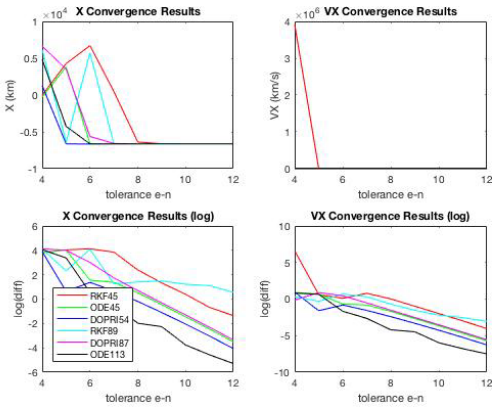

Fig. 39 X & VX Convergence Results: LEO, 600hour, J2+Lunar

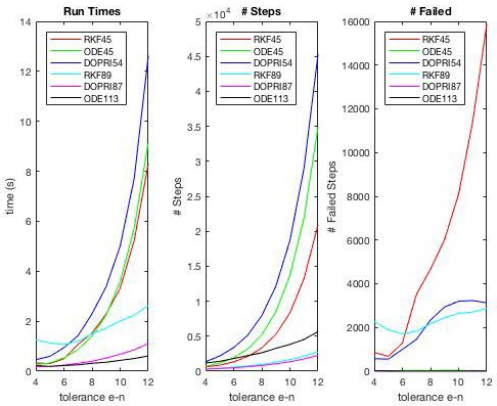

Fig. 40 Computation Results: MEO, 600hour, J2+Lunar

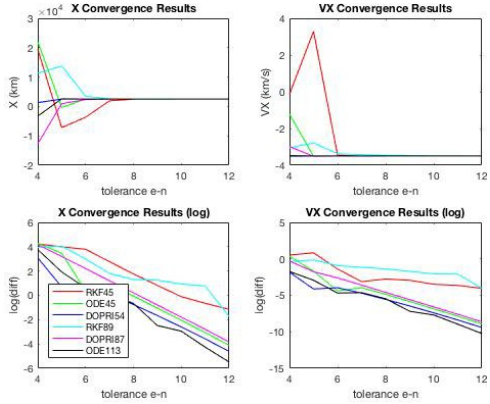

Fig. 41 X & VX Convergence Results: MEO, 600hour, J2+Lunar

F. Transfer Orbit

Unlike EO cases, the TO cases are plotted with log scale differences which have been calculated with respect to ephemeris data as opposed to a lower tolerance solution. Because results are compared to ephemeris data, log scale difference data shows flat lines as the integrators have reached the limit of achievable accuracy with the given force model. The following results magnify the inability of both RKF methods to converge as well as the DP and ABM methods. Fig. 29,30 show the lack of convergence for RKF through tolerances of $\delta = 1e - 12$. The flat line on the other hand is a combination of the other four integrators overlapping. A final point to note is the short run times for both TO cases. To explain this occurrence, a set of results for each satellite completing 500 orbits is propagated.

An observation of Run Time trends for MTO and VTO in Fig. 42 and Fig. 43 reveal a trend that contradicts the results of the EO cases above. As tolerance is tightened in EO cases, run times steadily increase. This expected trend is the result of tightened tolerances requiring smaller error margins for each step. When smaller error margins are allowed, smaller steps in time are taken to satisfy the tightened tolerance limits and thus more steps are taken. As more steps are taken there is a need for the numerical integrator to iterate through many more mathematical calculations in order to produce a solution. This ultimately slows down the run time as seen in EO results.

To properly compare EO and TO, consider the computational results presented for a 600-hour LEO propagation with two body dynamics in Fig. 11 to those presented for 600-hour MTO and VTO propagations with two body dynamics in Fig. 42 and Fig. 43. With equivalent model fidelities (simplified 2-body dynamics) and equivalent propagation times it may be expected to see comparable computational cost between EO and TO. Instead, we observe opposite trends for EO and TO. While LEO run times in Fig. 26 surpass 180 seconds for the slowest integrator (DOPRI54) and reach approximately 5 seconds for the fastest integrator (ODE113) with steadily increasing run times across all integrators, TO experience steadily decreasing run times. Additionally, the run time for all integrators (including DOPRI54) settle around one hundredth of a second (0.01s) for both MTO and VTO as seen in Fig. 42 and Fig. 43.

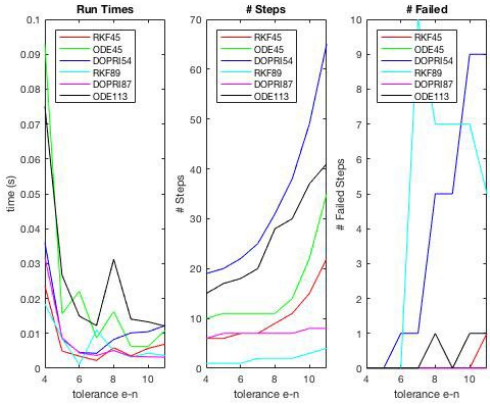

Fig. 42 Computation Results: MTO, 600hr, 2body

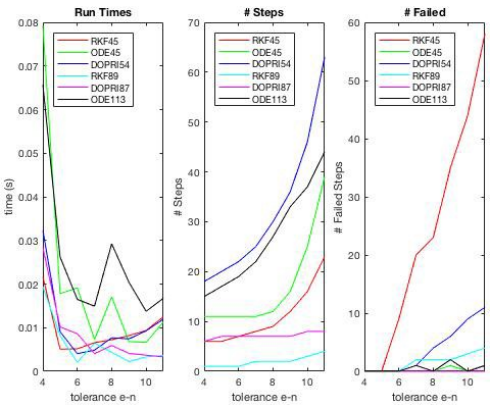

Fig. 43 Computation Results: VTO, 600hr, 2body

To identify the source of the discrepancy, dependencies and non-dependencies of integrators on orbital elements must be established. Orbital elements include $e, a, \Omega, i, \omega, \theta$, and τ as previously discussed. From EO results it has been established that the Keplerian elements which define orientation of an orbit (Ω, i, ω) and position along an orbit (θ) are not significant in the performance of orbit determination. This narrows potential dependencies to e, a and τ. These three orbital elements are responsible for defining the shape, size and speed of an orbit. When considering the eccentricities of all propagated EO and TO cases, eccentricity can also be eliminated as a source of the computational cost dependencies. This is because EO cases include eccentricities which range from $0.01 \leq e \leq 0.910$ while TO cases fall within the range $0.117 \leq e \leq 0.233$. As all EO propagation times are drastically greater than TO, yet the range of TO eccentricities falls within the range of EO eccentricities, we can ignore eccentricity for these results.

Potential orbital element dependencies now include semi-major axis (a) and orbital period (τ). These orbital elements are responsible for defining the shape and speed of an orbit. From the definition of orbital period in Eq. (17), we know that orbital period and semi major axis share a dependence on one another. We also know that both

$$\tau = 2\pi \sqrt{\frac{a^3}{GM}} \tag{17}$$

orbital period and semi major axis are dramatically different for EO and TO. For EO, semi-major axes of the satellites considered fall below10,000km which results in orbital periods that are measured in hours. For TO cases, however, semi-major axes of the satellites considered exceed 100,000,000km and have orbital periods in the hundreds of days. The difference in orbital periods results in a difference in average angular velocities (θ) from the definition of angular velocity given by Eq. (18). This definition produces a maximum EO average angular velocity of $\theta_{\text{bpo}} = 0.00155 \frac{\text{cbt}}{\text{Üqk}}$ for HEO and a minimum TO average angular velocity of $\theta_{\text{bpo}} = 0.0000153 \frac{\text{cbt}}{\text{Üqk}}$. The dependence of numerical integrator computational performance on semi-major axis and

$$\theta_{\text{bpo}} = \frac{2\pi}{\tau} \tag{18}$$

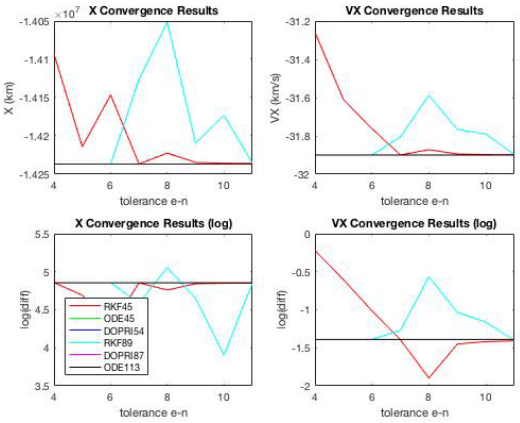

Fig. 44 X & VX Convergence Results: MTO, 600hr, 2body

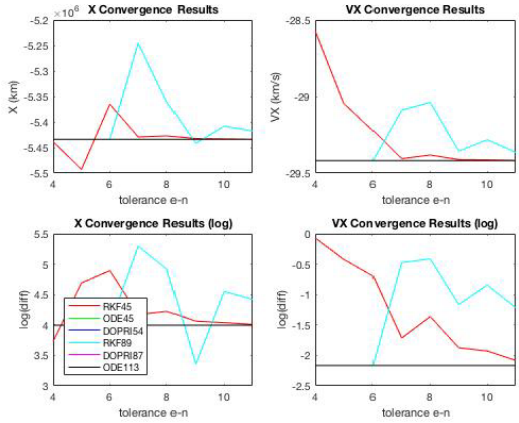

Fig. 45 X & VX Convergence Results: VTO, 600hr, 2body

G. 500 Orbit Propagation

With orbital period identified as the greatest contributing factor to computation cost, propagations with total number of orbits held constant are run to analyze the effect of varying additional orbital elements. By propagating each orbit for the same number of revolutions, the computational times are normalized within a smaller range of values as seen in Table 11. Satellite groups are listed in order of increasing eccentricity to identify the trend of increasing computational cost with increasing eccentricity.

ODE45 reveals a dependency only on eccentricity with the greatest run time being for the HEO case. ODE87 and ODE 113 have similar results when considering only the EO cases. Again, that is a dependence on eccentricity. When

including the data for TO cases, run times for these moderate eccentricity orbits are much greater. This suggests an additional dependency on semi-major axis unlike ODE45. When considering the EO cases for both RKF45 and RKF89 there is again, a dependence on eccentricity with the largest propagation times being for the HEO case. When propagating for TO cases, the run times drop to nearly zero. This suggests an inverse dependence on the size of the orbit being propagated. Based on the TO computational cost and convergence data provided in Fig. 29, 30 the low run times are attributed to an inability of RKF integrators accurately to handle large orbits at the tolerances used.

500 Orbit Propagation Computation Times *(s)*					
Orbit	**ODE45**	**ODE87**	**ODE113**	**RKF45**	**RKF89**
GEO	52.29	8.37	3.77	59.5	12.98
LEO	56.15	8.44	3.5	72.7	21.81
MEO	56.24	8.15	3.68	67.6	18.27
HEO	97.73	46.11	18.71	203.5	34.91
VTO	61.91	63.37	62.45	0.01	0.01
MTO	63.68	53.12	62.59	0.01	0.01
Dependency	$\boldsymbol{e}$ **only**	$\boldsymbol{e}, \boldsymbol{a}$	$\boldsymbol{e}, \boldsymbol{a}$	$\boldsymbol{e}$	$\boldsymbol{e}$

Table 11 500 Orbit Propagation: Computation Times & Keplerian Dependencies

VI. Conclusions

The computational experiments conducted in this study justify general recommendations for numerical integration settings specifically applicable to low to medium fidelity orbit propagation models. These recommendations stem from three primary conclusions which deal with tolerance settings, numerical integrators, and orbital elements.

The first conclusion to be made is the importance of tolerance selection. Across all solution sets we saw that convergence of orbit propagation states typically insufficient at RelTol settings above $\delta_{\text{caj}} = 1e - 10$. We also saw that various orbital elements, particularly orbital period (τ) and semi-major axis (a), require significantly tighter RelTol settings. This crucial consideration applies to all integrators and requires deliberate attention. Many integrators have built in default tolerances which will produce incorrect solutions if used. MATLAB's ODE45 has a default relative tolerance of $\delta_{\text{caj}} = 1e - 3$ for example. The one exception to this rule is the case of RKF45 which may be able to produce more accurate results at higher tolerances when determining LEO propagations with low to medium fidelity models. Best results are observed in this case at $\delta_{\text{caj}} = 1e - 8$.

The second conclusion drawn is that the fundamental mathematics of various integrators can produce widely different results for orbit propagations at similar tolerance settings. The factors considered for the experiments in this report include order of the integrator method and the method itself. Each set of results verify the trend that higher order solvers (ODE113, DOPRI87, RKF89) reduce computational time with their ability to take larger time steps compared to lower order solvers (RKF45, DOPRI54, ODE45). This becomes significant when determining high computational costs solutions such as long LEO propagations, especially as model fidelity increases. When dealing with very short, computationally inexpensive propagations the other hand, lower order solvers paired with tight tolerances are viable alternatives. This conclusion is in agreement with results by Uruxtan [13] which compares RKF67 with RKF78 and Ritschel [11] which compares RK, RKF, DP, and ESDIRK variants. Additionally, the TO results conclude that the RKF methods do not converge well for large scale orbits without very tight tolerances of at least $\delta_{\text{caj}} = 1e - 16$. The opposite extreme is that ODE113 (ABM) is the most stable integrator with the best convergence results for all cases. A third factor worth mentioning, although not considered in this analysis, is the relationship between step size and integrator order as they influence long term convergence stability [14].

The final conclusion to be drawn is that an exclusive set of orbital elements impact computational cost and thus the ability for computational solutions to converge with tightened tolerance. The three elements (τ, e, a) and their values for each of the six satellites considered for the above experiments are presented in Table 12.

Results show that the combined influence of the shape, size, and period of an orbit are responsible for the majority of the computational cost of a solution. The most extreme example the 600-hour LEO propagation with a force model that includes Lunar gravity and $J_{\cdot}$ perturbations. Due to its short period and rapidly changing states, it is the most difficult to determine a solution for. For these orbits a higher order solver that is stable enough to produce accurate solutions such as DP87 or ODE113 should be used. To a lesser degree, eccentricity also plays a hand in increasing computational cost due to the rapidly changing states near perigee. For more stable, and less computationally expensive low eccentricity orbits, lower order solvers may be considered. This only applies for Earth Orbiting satellites

since Transfer Orbit solutions are highly susceptible to the effects of local error compounding to produce large global errors. Furthermore, the fact that there is little change in angular position of Transfer Orbit trajectories, integrators tend to take very large steps which may lead to additional inaccuracies in solution determination.

Initial Orbit Elements				
Orbit	**Satellite**	***τ (min)***	***e (deg)***	***a (km)***
GEO	GOES-14	1436.1	0.00100297	42166
LEO	ISS	92.7	0.00123243	6778
MEO	NAVSTAR-68	718.0	0.00529487	26562
HEO	MMS-4	4053.0	0.910034	84187
VTO	VEX	410678 (~285 days)	0.171225	1.27e08
MTO	MSL	750601 (~521 days)	0.223352	1.90e08

Table 12 Summary of Relevant Initial Orbital Elements

The final recommendations for solver setups is to consider minimum relative tolerances of $\delta = 1e-10$ for EO cases while at least $\delta = 1e-12$ is preferable. As orbital period and semi major axis increase this may be scaled down as necessary. As tolerance is tightened these solvers may fail due to stiffness characteristics [15], at which point different solvers would need to be considered [16]. Additionally, for solutions with long propagation times one should consider higher order solvers which are capable of taking larger steps in time and thus fewer total steps. This tends to reduce global error as discussed by Aristoff [10]. A summary of preliminary recommendations for numerical integrator settings on low to medium fidelity models is listed in Table 13. These recommendations are minimum suggestions for simple models and may not apply to complex missions or high-fidelity models.

Summary of Conclusions	
Minimum δ_{caj} (EO)	1e-10
Minimum δ_{caj} (TO)	1e-14
Small $\Delta\theta$	Lower Order Integrators (ODE45 or RKF45)
Large $\Delta\theta$	Higher Order Integrators (DP87 or ODE113)
ECI frame	Lower or Higher Order Integrators
SCI frame	Higher Order Integrators only
LEO (large θ)	Higher Order Integrators
Large e	ODE45

Appendix A: Butcher Coefficients

0						
1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	−7200/2197	7296/2197			
1	439/216	−8	3680/513	−845/4104		
1/2	−8/27	2	−3544/2565	1859/4104	−11/40	
	16/135	0	6656/12825	28561/56430	−9/50	2/55
	25/216	0	1408/2565	2197/4104	−1/5	0

RKF 45

c_i	a_{ij}					$\hat{b}_i$	b_i
0						$\frac{19}{200}$	$\frac{431}{5000}$
$\frac{2}{9}$	$\frac{2}{9}$					0	0
$\frac{1}{3}$	$\frac{1}{12}$	$\frac{1}{4}$				$\frac{3}{5}$	$\frac{333}{500}$
$\frac{5}{9}$	$\frac{55}{324}$	$-\frac{25}{108}$	$\frac{50}{81}$			$-\frac{243}{400}$	$-\frac{7857}{10000}$
$\frac{2}{3}$	$\frac{83}{330}$	$-\frac{13}{22}$	$\frac{61}{66}$	$\frac{9}{110}$		$\frac{33}{40}$	$\frac{957}{1000}$
1	$-\frac{19}{28}$	$\frac{9}{4}$	$\frac{1}{7}$	$-\frac{27}{7}$	$\frac{22}{7}$	$\frac{7}{80}$	$\frac{193}{2000}$
1	$\frac{19}{200}$	0	$\frac{3}{5}$	$-\frac{243}{400}$	$\frac{33}{40}$ $\frac{7}{80}$	0	$-\frac{1}{50}$

DP 54

c_i	a_{ij}										$\hat{b}_i$	b_i
0											$\frac{14005451}{335480064}$	$\frac{13451932}{455176623}$
$\frac{1}{18}$	$\frac{1}{18}$										0	0
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$									0	0
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$								0	0
$\frac{5}{16}$	$\frac{5}{16}$	0	$-\frac{75}{64}$	$\frac{75}{64}$							0	0
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$						$-\frac{59238493}{1068277825}$	$-\frac{808719846}{976000145}$
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736538}{692538347}$	$-\frac{28693883}{1125000000}$	$\frac{23124283}{1800000000}$					$\frac{181606767}{758867731}$	$\frac{1757004468}{5645159321}$
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$-\frac{180193667}{1043307555}$				$\frac{561292985}{797845732}$	$\frac{656045339}{265891186}$
$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$-\frac{433636366}{683701615}$	$\frac{421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$			$-\frac{1041891430}{1371343529}$	$-\frac{3675574721}{1518517206}$
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$\frac{37695042795}{15268766246}$	$-\frac{309121744}{1061227803}$	$-\frac{12992083}{490766935}$	$\frac{6005943493}{2108947869}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$		$\frac{760417239}{1151165299}$	$\frac{465885868}{322736535}$
$\frac{1201146811}{1299019798}$	$\frac{1028468189}{846180014}$	0	0	$\frac{8478235783}{508512852}$	$\frac{1311729495}{1432422823}$	$\frac{10304129995}{1701304382}$	$\frac{48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$\frac{45442868181}{3398467696}$	$\frac{3065993473}{597172653}$	$\frac{118820643}{751138087}$	$\frac{53011238}{667516719}$
1	$\frac{185892177}{718116043}$	0	0	$\frac{3185094517}{667107341}$	$\frac{477755414}{1098053517}$	$\frac{703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$\frac{4093664535}{808688257}$	$\frac{3962137247}{1805957418}$ $\frac{65686358}{487910083}$	$-\frac{528747749}{2220607170}$	$\frac{2}{45}$
1	$\frac{403863854}{491063109}$	0	0	$\frac{5068492393}{434740067}$	$\frac{411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$\frac{13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$\frac{160528059}{685178525}$ $\frac{248638103}{1413531060}$ 0	$\frac{1}{4}$	0

DP 87

$$
\begin{aligned}
\beta_{86} &= 0.3254\,2131\,7015\,8914\,7114\,6774\,6964\,8853 \\
\beta_{86} &= 0.2847\,6660\,1385\,2790\,8888\,1824\,2057\,3687 \qquad \beta_{146} = -0.6557\,0189\,4497\,4164\,5138\,0068\,7998\,5251 \\
\beta_{87} &= 0.9783\,7801\,6759\,7915\,2435\,8683\,9727\,1099 \cdot 10^{-2} \quad \beta_{147} = -0.3908\,6144\,8804\,3986\,3435\,0255\,2024\,1310 \\
\beta_{90} &= 0.6084\,2071\,0626\,2205\,7051\,0941\,4520\,5182 \cdot 10^{-1} \quad \beta_{148} = 0.2679\,4646\,7128\,5002\,2936\,5844\,2327\,1209 \\
\beta_{95} &= -0.2118\,4565\,7440\,3700\,7526\,3252\,7525\,1206 \cdot 10^{-1} \quad \beta_{149} = -0.1038\,3022\,9913\,8249\,0865\,7695\,5850\,7427 \cdot 10^{+1} \\
\beta_{96} &= 0.1959\,6557\,2661\,7083\,1957\,4644\,9066\,2983 \qquad \beta_{1410} = 0.1667\,2327\,3242\,5867\,1664\,7273\,4616\,8501 \cdot 10^{+1} \\
\beta_{97} &= -0.4274\,2640\,3648\,1760\,3675\,1448\,3534\,2899 \cdot 10^{-2} \quad \beta_{1411} = 0.4955\,1925\,8553\,1597\,7067\,7329\,6707\,1441 \\
\beta_{98} &= 0.1743\,4365\,7368\,1491\,1945\,3234\,5255\,8189 \cdot 10^{-1} \quad \beta_{1412} = 0.1139\,4001\,1323\,9706\,3228\,5867\,3814\,1784 \cdot 10^{+1} \\
\beta_{99} &= 0.5105\,9783\,2959\,3191\,7345\,7857\,2411\,1182 \cdot 10^{-1} \quad \beta_{1413} = 0.5133\,6696\,4246\,5861\,3688\,1990\,9719\,1534 \cdot 10^{-1} \\
\beta_{103} &= 0.1102\,9325\,5978\,2892\,6539\,2831\,2764\,8228 \qquad \beta_{150} = 0.1046\,4847\,3406\,1481\,0391\,8730\,0240\,6755 \cdot 10^{-2} \\
\beta_{107} &= -0.1256\,5008\,5200\,7255\,6414\,1477\,6378\,2250 \cdot 10^{-2} \quad \beta_{158} = -0.6716\,3886\,8449\,9028\,2237\,7784\,4617\,8020 \cdot 10^{-3} \\
\beta_{108} &= 0.3679\,0043\,4775\,8146\,0136\,3840\,4356\,6339 \cdot 10^{-2} \quad \beta_{159} = 0.8182\,8762\,1894\,2502\,1265\,3300\,6524\,8999 \cdot 10^{-2} \\
\beta_{109} &= -0.5778\,0542\,7709\,7207\,3940\,8406\,2857\,1866 \cdot 10^{-1} \quad \beta_{1510} = -0.4264\,0342\,8644\,8334\,7277\,1421\,3808\,7561 \cdot 10^{-2} \\
\beta_{110} &= 0.1273\,2477\,0686\,6711\,4646\,6451\,8179\,9160 \qquad \beta_{1511} = 0.2800\,9029\,4741\,6893\,6545\,9763\,3115\,3703 \cdot 10^{-3} \\
\beta_{117} &= 0.1144\,8395\,0093\,9510\,5323\,6588\,7572\,1817 \qquad \beta_{1512} = -0.8783\,5333\,8762\,3867\,6639\,0578\,1314\,5633 \cdot 10^{-3} \\
\beta_{118} &= 0.2877\,3020\,7036\,9799\,2776\,2022\,0184\,9198 \qquad \beta_{1513} = 0.1025\,4505\,1108\,2535\,8084\,2177\,6966\,4009 \cdot 10^{-1} \\
\beta_{119} &= 0.5094\,5379\,4596\,1136\,3153\,7358\,8507\,9465 \qquad \beta_{160} = -0.1353\,6550\,7861\,7406\,7080\,4421\,6888\,9966 \cdot 10^{+1} \\
\beta_{1110} &= -0.1479\,9682\,2443\,7257\,5903\,2421\,4444\,9640 \qquad \beta_{165} = -0.1839\,6103\,1448\,4827\,0375\,0441\,9898\,8231 \\
\beta_{120} &= -0.3532\,6793\,8766\,1674\,0535\,8485\,4439\,4333 \cdot 10^{-2} \quad \beta_{166} = -0.6557\,0189\,4497\,4164\,5138\,0068\,7998\,5251 \\
\beta_{125} &= 0.8162\,9896\,0123\,1891\,9777\,8194\,2124\,7030 \cdot 10^{-1} \quad \beta_{167} = -0.3908\,6144\,8804\,3986\,3435\,0255\,2024\,1310 \\
\beta_{126} &= -0.3860\,7735\,6356\,9350\,6490\,5176\,9434\,3215 \qquad \beta_{168} = 0.2746\,6285\,5812\,9992\,5758\,9622\,0773\,2989 \\
\beta_{127} &= 0.3086\,2242\,9246\,0510\,6450\,4741\,6602\,5206 \cdot 10^{-1} \quad \beta_{169} = -0.1046\,4851\,7535\,7191\,5887\,0351\,8857\,2676 \cdot 10^{+1} \\
\beta_{128} &= -0.5807\,7254\,5283\,2060\,2815\,8293\,7473\,3518 \cdot 10^{-1} \quad \beta_{1610} = 0.1671\,4967\,6671\,2315\,5012\,0044\,8830\,6588 \cdot 10^{+1} \\
\beta_{129} &= 0.3359\,8659\,3288\,8497\,1493\,1434\,5136\,2322 \qquad \beta_{1611} = 0.4952\,3916\,8258\,4180\,8131\,1869\,9074\,0287 \\
\beta_{1210} &= 0.4106\,6890\,4019\,4995\,8613\,5496\,2278\,6417 \qquad \beta_{1612} = 0.1148\,1836\,4662\,7330\,1905\,2257\,9595\,4930 \cdot 10^{+1} \\
\beta_{1211} &= -0.1184\,0245\,9723\,5598\,5520\,6331\,5615\,4536 \cdot 10^{-1} \quad \beta_{1613} = 0.4108\,2191\,3138\,3305\,5603\,9813\,2752\,7525 \cdot 10^{-1} \\
\beta_{130} &= -0.1237\,5357\,9212\,4514\,3254\,9790\,9613\,5669 \cdot 10^{+1} \quad \beta_{1615} = 1 \\
\beta_{135} &= -0.2443\,0768\,5513\,5478\,5358\,7348\,6136\,6763 \cdot 10^{+2} \\
\beta_{136} &= 0.5477\,9568\,9327\,7865\,6050\,4365\,2899\,1173 \qquad c_0 = 0.3225\,6083\,5002\,1624\,9913\,6129\,0096\,0247 \cdot 10^{-1} \\
\beta_{137} &= -0.4441\,3863\,5334\,1324\,6374\,9598\,9656\,9346 \cdot 10^{+1} \quad c_8 = 0.2598\,3725\,2837\,1540\,3018\,8870\,2317\,1963 \\
\beta_{138} &= 0.1001\,3104\,8137\,1326\,6094\,7926\,1785\,1022 \cdot 10^{+2} \quad c_9 = 0.9284\,7805\,9965\,7702\,7788\,0637\,1430\,2190 \cdot 10^{-1} \\
\beta_{139} &= -0.1499\,5773\,1020\,5175\,8447\,1709\,8507\,3142 \cdot 10^{+2} \quad c_{10} = 0.1645\,2339\,5147\,6434\,2891\,6477\,3184\,2800 \\
\beta_{1310} &= 0.5894\,6948\,5232\,1701\,3620\,8245\,3965\,1427 \cdot 10^{+1} \quad c_{11} = 0.1766\,5951\,6378\,6007\,4367\,0842\,9839\,7547 \\
\beta_{1311} &= 0.1738\,0377\,5034\,2898\,4877\,6168\,5744\,0542 \cdot 10^{+1} \quad c_{12} = 0.2392\,0102\,3203\,5275\,9374\,1089\,3332\,0941 \\
\beta_{1312} &= 0.2751\,2330\,6931\,6673\,0263\,7586\,2286\,0276 \cdot 10^{+2} \quad c_{13} = 0.3948\,4274\,6042\,0285\,3746\,7521\,1882\,9335 \cdot 10^{-2} \\
\beta_{140} &= -0.3526\,0859\,3883\,3452\,2700\,5029\,5887\,5588 \qquad c_{14} = 0.3072\,6495\,4758\,6064\,0406\,3683\,0552\,2124 \cdot 10^{-1} \\
\beta_{145} &= -0.1839\,6103\,1448\,4827\,0375\,0441\,9898\,8231
\end{aligned}
$$

$$
\begin{aligned}
\alpha_1 &= 0.4436\,8940\,3764\,9818\,3109\,5994\,0428\,1370 \\
\alpha_2 &= 0.6655\,3410\,5647\,4727\,4664\,3991\,0642\,2055 \\
\alpha_3 &= 0.9983\,0115\,8471\,2091\,1996\,5986\,5963\,3083 \\
\alpha_4 &= 0.3155\,0000\,0000\,0000\,0000\,0000\,0000\,0000 \\
\alpha_5 &= 0.5054\,4100\,9481\,6906\,8626\,5161\,2673\,7384 \\
\alpha_6 &= 0.1714\,2857\,1428\,5714\,2857\,1428\,5714\,2857 \\
\alpha_7 &= 0.8285\,7142\,8571\,4285\,7142\,8571\,4285\,7143 \\
\alpha_8 &= 0.6654\,3966\,1210\,1156\,2534\,9537\,6925\,5586 \\
\alpha_9 &= 0.2487\,8317\,9680\,6265\,2069\,7222\,7456\,0771 \\
\alpha_{10} &= 0.1090\,0000\,0000\,0000\,0000\,0000\,0000\,0000 \\
\alpha_{11} &= 0.8910\,0000\,0000\,0000\,0000\,0000\,0000\,0000 \\
\alpha_{12} &= 0.3995\,0000\,0000\,0000\,0000\,0000\,0000\,0000 \\
\alpha_{13} &= 0.6005\,0000\,0000\,0000\,0000\,0000\,0000\,0000 \\
\alpha_{14} &= 1 \\
\alpha_{15} &= 0 \\
\alpha_{16} &= 1
\end{aligned}
$$

$$
\begin{aligned}
\beta_{10} &= 0.4436\,8940\,3764\,9818\,3109\,5994\,0428\,1370 \\
\beta_{20} &= 0.1663\,8352\,6411\,8681\,8666\,0997\,7660\,5514 \\
\beta_{21} &= 0.4991\,5057\,9235\,6045\,5998\,2993\,2981\,6541 \\
\beta_{30} &= 0.2495\,7528\,9617\,8022\,7999\,1496\,6490\,8271 \\
\beta_{38} &= 0.7487\,2586\,8853\,4068\,3997\,4489\,9472\,4812 \\
\beta_{40} &= 0.2066\,1891\,1634\,0060\,2426\,5567\,1039\,3185 \\
\beta_{48} &= 0.1770\,7880\,3779\,8634\,7040\,3803\,9728\,8319 \\
\beta_{43} &= -0.6819\,7715\,4138\,6949\,4669\,3770\,7681\,5048 \cdot 10^{-1} \\
\beta_{50} &= 0.1092\,7823\,1526\,6640\,8227\,9038\,9092\,6157 \\
\beta_{53} &= 0.4021\,5962\,6423\,6799\,5421\,9905\,6369\,0087 \cdot 10^{-2} \\
\beta_{58} &= 0.3921\,4118\,1690\,7898\,0444\,3923\,3017\,4325 \\
\beta_{60} &= 0.9889\,9281\,4091\,6466\,5304\,8447\,6543\,4355 \cdot 10^{-1} \\
\beta_{63} &= 0.3513\,8370\,2279\,6396\,6951\,2044\,8735\,6703 \cdot 10^{-2} \\
\beta_{64} &= 0.1247\,6099\,9831\,6001\,6621\,5206\,2587\,2489 \\
\beta_{65} &= -0.5574\,5546\,8349\,8979\,9643\,7429\,0146\,6348 \cdot 10^{-1} \\
\beta_{70} &= -0.3680\,6865\,2862\,4220\,3724\,1531\,0108\,0691 \\
\beta_{74} &= -0.2227\,3697\,4694\,7600\,7645\,0240\,2094\,4166 \cdot 10^{+1} \\
\beta_{75} &= 0.1374\,2908\,2567\,0291\,0729\,5656\,9124\,5744 \cdot 10^{+1} \\
\beta_{76} &= 0.2049\,7390\,0271\,1160\,3002\,1593\,5409\,2206 \cdot 10^{+1} \\
\beta_{80} &= 0.4546\,7962\,6413\,4715\,0077\,3519\,5060\,3349 \cdot 10^{-1}
\end{aligned}
$$

RKF 89

Appendix B: Ephemeris Data

```
 4.308029391424928E+03  1.228804400513577E+03 -5.097561378230870E+03 -3.876789759410662E+00  6.372826698311722E+00 -1.738322943468619E+00
-3.199720073060143E+03 -2.728382385352880E+03  5.314021052617424E+03  4.748436158196570E+00 -6.018782640021895E+00 -2.282717591541034E-01
 2.036895116419154E+03  4.148333396950018E+03 -4.968452632335481E+03 -5.230966165425668E+00  5.159924490034640E+00  2.166745490447705E+00
-8.728232216690237E+02 -5.356883655310816E+03  4.062226348139089E+03  5.387472454423563E+00 -3.833626732033810E+00 -3.885480293857576E+00
-1.764891573220144E+02  6.205110694916026E+03 -2.735749217191903E+03 -5.250115934369798E+00  2.124362335270759E+00  5.167150511717062E+00
 1.128267032630239E+03 -6.601725793918355E+03  1.079533707421177E+03  4.900445373001888E+00 -1.340527393139956E-01 -5.896099166686429E+00
-1.914837952445431E+03  6.470441093911177E+03  6.646389559278248E+02 -4.406583010745754E+00 -1.924757556788652E+00  5.977895443447658E+00
 2.600655079801802E+03 -5.804586605739816E+03 -2.368191742809247E+03  3.799819589977785E+00  3.899381397450958E+00 -5.392619010836091E+00
-3.157031956711906E+03  4.650376671367556E+03  3.785336167471402E+03 -3.135265683411856E+00 -5.580235389966049E+00  4.231582678708974E+00
 3.642564603144835E+03 -3.109424487306966E+03 -4.810031930612537E+03  2.387331934576316E+00  6.801239617932749E+00 -2.587114785298229E+00
-4.029389352452426E+03  1.313097876833400E+03  5.285921143868931E+03 -1.566226606966379E+00 -7.480234437718042E+00  6.660284853246635E-01
 4.330978831969692E+03  5.433943264200270E+02 -5.196614342630389E+03  6.444342375472271E-01  7.515217743440275E+00  5.977895443447658E+00
-4.490238672185535E+03 -2.330776428076773E+03  4.510096630981359E+03  3.985561302033808E-01 -6.964048502769697E+00 -3.190198546014057E+00
 4.483579943372205E+03  3.834036753133360E+03 -3.350584832530470E+03 -1.531129667954725E+00  5.872725211419854E+00  4.681418244417912E+00
-4.226450882265652E+03 -4.996629040832176E+03  1.782408373049207E+03  2.753076221073407E+00 -4.364773454886085E+00 -5.672436126657855E+00
 3.706589029222082E+03  5.677835641879478E+03 -5.223141793389379E+01 -3.956089092808994E+00  2.628959985213941E+00  6.024165757948373E+00
-2.856516483564052E+03 -5.910308333182886E+03 -1.724619508574395E+03  6.066548185905654E+00 -7.928613213648402E-01 -5.694962657484495E+00
 1.730396276556428E+03  5.677209201366152E+03  3.270324500458696E+03 -5.956925598853123E+00 -9.274497285239940E-01  4.747975391421907E+00
-3.822011741153508E+02 -5.106229711585856E+03 -4.457453421565165E+03  6.488411401352309E+00  2.388059590187383E+00 -3.292117336559213E+00
-1.111623071371455E+03  4.263850214425825E+03  5.145206280089033E+03 -6.631458827621091E+00 -3.552157519417201E+00  1.509958645845772E+00
 2.664199360904041E+03 -3.282240351084468E+03 -5.311291458489531E+03  6.267002088593805E+00  4.376292682596802E+00  4.426032699995737E-01
-4.137865860682597E+03  2.212222421084800E+03  4.888326797011404E+03 -5.415676585127468E+00 -4.894675028823272E+00 -2.359543777140542E+00
 5.374057997532877E+03 -1.186847565915642E+03 -3.971065423599650E+03  4.080695976495812E+00  5.097947330585654E+00  4.006945932716858E+00
-6.262221346821281E+03  1.923289039070273E+02  2.593515017469547E+03 -2.341642472358671E+00 -5.077331884741346E+00 -5.250651495601354E+00
 6.678388592696607E+03  6.737945453165955E+02 -9.765090736592348E+02  3.659867489952943E-01  4.866091632857900E+00  5.916239133440384E+00
-6.574056682377489E+03 -1.484467210413474E+03 -7.920106633764153E+02  1.728392271448749E+00 -4.516347950589427E+00 -5.948005798696118E+00
```

LEO Ephemeris

```
 4.207271318711324E+04 -3.043301564196011E+03  1.959564540450188E+00  2.245585730258557E-01  3.065165867973303E+00  1.805428524179769E-03
 4.211927259895070E+04 -2.318529951638129E+03 -1.132393518800868E+00  1.717252812489627E-01  3.068549770705613E+00  1.791671887550077E-03
 4.215335952716568E+04 -1.593155743708334E+03 -4.084810769714537E+00  1.188458971694311E-01  3.071021672900843E+00  1.734861365621663E-03
 4.217499863605884E+04 -8.677088741628230E+02 -6.269750279760288E+00  6.596232187528822E-02  3.072577787571593E+00  1.648770677946229E-03
 4.218407849493427E+04 -1.422650525852266E+02 -8.040874787465498E+00  1.307523226732939E-02  3.073228150149543E+00  1.529987253612576E-03
 4.218067848567109E+04  5.823576402999624E+02 -9.271645696146159E+00 -3.975085425584283E-02  3.072970100421487E+00  1.404446337153119E-03
 4.216489727511265E+04  1.306324529211338E+03 -9.955446315465132E+00 -9.252876755569861E-02  3.071801628859798E+00  1.306324022996694E-03
 4.213669834330771E+04  2.028863126123141E+03 -1.063681694200769E+01 -1.452045309916920E-01  3.069733339767688E+00  1.216024690407891E-03
 4.209623714747690E+04  2.751459732376084E+03 -1.139278472007631E+01 -1.978761606710072E-01  3.066747778534666E+00  1.199002891148444E-03
 4.204326389850879E+04  3.472856446078324E+03 -1.263179254671056E+01 -2.504675651094427E-01  3.062870924656863E+00  1.194158711655243E-03
 4.197794533187628E+04  4.193498049235276E+03 -1.428806240518188E+01 -3.029988478591700E-01  3.058088280138865E+00  1.234894737710917E-03
 4.190026939341454E+04  4.913219681607753E+03 -1.629849353504763E+01 -3.554596760379907E-01  3.052400477194666E+00  1.309006378188835E-03
 4.181019344306212E+04  5.631902647466284E+03 -1.862123787160116E+01 -4.078443881192039E-01  3.045809152200777E+00  1.394177743198444E-03
 4.170774802857226E+04  6.349005031655473E+03 -2.131528451247946E+01 -4.601101631127577E-01  3.038321432158777E+00  1.471287436154170E-03
 4.159301840566221E+04  7.064373629618010E+03 -2.535346359188541E+01 -5.122473033149056E-01  3.029933661104203E+00  1.535612359145449E-03
 4.146596277558835E+04  7.778029045715635E+03 -2.801953428338941E+01 -5.642567536134656E-01  3.020645718866378E+00  1.605758631475930E-03
 4.132672465696019E+04  8.489382227022679E+03 -3.037764708542669E+01 -6.160931405669174E-01  3.010462442103101E+00  1.638060985498580E-03
 4.117518190696019E+04  9.198131009334826E+03 -3.245564469364815E+01 -6.677381282856335E-01  2.999401651008726E+00  1.609823708538974E-03
 4.101155881853382E+04  9.904037408212667E+03 -3.384923276503324E+01 -7.191742565586451E-01  2.987454559796675E+00  1.526335984634917E-03
 4.083585417056803E+04  1.060689342259991E+04 -3.500749222016329E+01 -7.703870036886097E-01  2.974628855993383E+00  1.380704173150964E-03
 4.064825595177521E+04  1.130627852598333E+04 -3.554647388952322E+01 -8.213447115810126E-01  2.960929763529371E+00  1.215085434606264E-03
 4.044850887588888E+04  1.200233319893255E+04 -3.601393357758735E+01 -8.720619853305589E-01  2.946369409335050E+00  1.080373230662870E-03
 4.023691929623953E+04  1.269471678283234E+04 -3.590343710981351E+01 -9.225108252389941E-01  2.930942253665164E+00  1.055132414349378E-03
 4.001340842970183E+04  1.338368276902571E+04 -3.606958966153730E+01 -9.727070643125885E-01  2.914641097469401E+00  1.089355265557577E-03
 3.977800969141015E+04  1.406889676059752E+04 -3.457589825178917E+01 -1.022632825970009E+00  2.897476711020478E+00  1.046657719635822E-03
 3.953065943741363E+04  1.475039321713394E+04 -3.498222616680826E+01 -1.072291353758565E+00  2.879448534632058E+00  1.185489009539945E-03
```

GEO Ephemeris

```
2.040128884969721E+04  1.623981855256800E+04  4.269754878646445E+03 -1.782376704004259E+00  1.401521629151137E+00  3.165933430292275E+00
1.996261277489026E+04  1.655873680290374E+04  5.043162018318596E+03 -1.888435322035183E+00  1.315981700846523E+00  3.141214883281617E+00
1.949841071718300E+04  1.685675374825021E+04  5.810375911774780E+03 -1.992189040367365E+00  1.228831626834502E+00  3.112406000774639E+00
1.900916681050630E+04  1.713351023642267E+04  6.570325536480119E+03 -2.093490203086505E+00  1.140169423901217E+00  3.079559459589021E+00
1.849559510420892E+04  1.738865672713816E+04  7.321680446707894E+03 -2.192176993142109E+00  1.050139655843423E+00  3.042735083332058E+00
1.795843147562981E+04  1.762189040556015E+04  8.063276550440554E+03 -2.288096634849812E+00  9.588739171026283E-01  3.001988319951357E+00
1.739844798367016E+04  1.783301001083641E+04  8.793989585520369E+03 -2.381115785738745E+00  8.665186896366943E-01  2.957366989874409E+00
1.681634960450037E+04  1.802178097730307E+04  9.512795811136562E+03 -2.471130732981901E+00  7.732099756435346E-01  2.908948165918461E+00
1.621288001792704E+04  1.818805299479973E+04  1.021883333622883E+04 -2.558026982817275E+00  6.790795726094413E-01  2.856773320751814E+00
1.558879019329698E+04  1.833171493629367E+04  1.091133320711294E+04 -2.641724686535292E+00  5.842330067179049E-01  2.800897467092808E+00
1.494477052684938E+04  1.845256069485601E+04  1.158952913503857E+04 -2.722122644040519E+00  4.887906755348781E-01  2.741396061501225E+00
1.428149851149784E+04  1.855047799745067E+04  1.225274323684530E+04 -2.799137322107117E+00  3.928527880048343E-01  2.678325874957987E+00
1.359978938967161E+04  1.862532776091655E+04  1.290017245216836E+04 -2.872676825098534E+00  2.965347372252209E-01  2.611773523710231E+00
1.290037859259209E+04  1.867699755930967E+04  1.353113452038492E+04 -2.942652535617566E+00  1.999311800737836E-01  2.541814939441683E+00
1.218419275778091E+04  1.870539548556354E+04  1.414480598653455E+04 -3.008956784210870E+00  1.031623544336664E-01  2.468548553459367E+00
1.145205097503920E+04  1.871048050576336E+04  1.474040199562001E+04 -3.071507154646561E+00  6.341719557890933E-03  2.392073641544728E+00
1.070493721889174E+04  1.869224537595412E+04  1.531711146786130E+04 -3.130206384395009E+00 -9.040795647093615E-02  2.312495210262735E+00
9.943805140234592E+03  1.865070598397352E+04  1.587413561094786E+04 -3.184975432473220E+00 -1.869702284705368E-01  2.229932598823162E+00
9.169727078089913E+03  1.858591474772638E+04  1.641068639117574E+04 -3.235734104793243E+00 -2.832107388426315E-01  2.144510861795339E+00
8.383725517685956E+03  1.849812580217543E+04  1.692598652114684E+04 -3.282399606984718E+00 -3.790084105667340E-01  2.056341761090132E+00
7.586859361387896E+03  1.838742152363843E+04  1.741929160840637E+04 -3.324918377246460E+00 -4.742252128166786E-01  1.965564142178956E+00
6.780174034504481E+03  1.825409898383862E+04  1.788987273180565E+04 -3.363232395842103E+00 -5.687339096500429E-01  1.872303989362466E+00
5.964675119159324E+03  1.809841989354317E+04  1.833710631216293E+04 -3.397300273663465E+00 -6.624118716206899E-01  1.776683953858326E+00
5.141380718771181E+03  1.792064859503309E+04  1.876040994878011E+04 -3.427084119315542E+00 -7.551349452337444E-01  1.678827757218488E+00
4.311185489118364E+03  1.772109382942193E+04  1.915927684881175E+04 -3.452563120770427E+00 -8.467942094693227E-01  1.578834565237770E+00
3.475019404104248E+03  1.749995530597227E+04  1.953329725122838E+04 -3.473715071255469E+00 -9.372701672601800E-01  1.476835073253016E+00
```

MEO Ephemeris

5.736366022845366E+04	-5.448781186552533E+04	3.577781202704391E+03	-9.068305796155037E-01	2.112449806850424E+00	2.154521545293401E-01
2.612150645587634E+04	-1.313850620048007E+05	-2.148306482628055E+04	7.050740008665590E-01	-8.035618902109056E-01	1.477348150284217E-02
6.896365654382645E+04	-1.388609259531748E+05	-1.161044051895507E+04	2.427584408947046E-01	5.500304299455903E-01	1.837098747984810E-01
3.450420926614475E+04	-1.488574362420839E+04	5.950720714194456E+03	-2.500618094636844E+00	3.155787572257679E+00	1.614033663456661E-02
3.700221228301618E+04	-1.413698702549801E+05	-2.055454624925135E+04	6.402937984182504E-01	-5.052987084943021E-01	6.063700648975699E-02
7.227493639507903E+04	-1.281183476376338E+05	-8.399806328898032E+03	1.162699819263699E-01	7.903343750832549E-01	1.994728388645208E-01
-1.512683612584620E+04	-3.454030722577406E+04	-1.130834408537800E+04	4.012653329648400E-01	-3.852488068105104E+00	-7.153266456818677E-01
4.676932887409420E+04	-1.471260929857623E+05	-1.905031363213538E+04	5.685338617083687E-01	-2.428388240417577E-01	9.665792716356408E-02
7.314246764830519E+04	-1.131476060503970E+05	-4.946363248657245E+03	-4.840808034089647E-02	1.064001762686982E+00	2.129037880613259E-01
-3.799322932840724E+03	-7.903973180510917E+04	-1.770159182850831E+04	8.141990309957347E-01	-2.093177824587258E+00	-2.310658263881195E-01
5.542266272788151E+04	-1.490272648714071E+05	-1.708033455993056E+04	4.854698437042826E-01	-2.100149558220347E-03	1.256160424613417E-01
7.070640833573986E+04	-9.365089680092708E+04	-1.408006879206009E+03	-2.801308223693115E-01	1.389822127850832E+00	2.203964680254883E-01
9.601128982521155E+03	-1.064196868318082E+05	-1.989870823037283E+04	8.017315411944476E-01	-1.397139807151427E+00	-8.625135604777678E-02
6.274389882720088E+04	-1.473032507821230E+05	-1.471328174828224E+04	3.916670945568547E-01	2.235747367896822E-01	1.480742272007791E-01
6.378925226139960E+04	-6.874987158579545E+04	2.022469602002571E+03	-6.446933162085724E-01	1.817505361610909E+00	2.145560926798086E-01
2.226168773417521E+04	-1.247239941741877E+05	-2.031107794578054E+04	7.453712779193490E-01	-9.613693358663337E-01	-9.522358704533388E-03
6.856187986733265E+04	-1.419520137310570E+05	-1.203372965624663E+04	2.891654993410867E-01	4.453739534575745E-01	1.661344705033552E-01
4.882626236265407E+04	-3.547493028465596E+04	5.003611341953681E+03	-1.442248292568789E+00	2.514795589139231E+00	1.553674685035059E-01
3.376690424578599E+04	-1.369706254514189E+05	-1.969545945115564E+04	6.798670163341719E-01	-6.324494980363885E-01	4.064594622674343E-02
7.268886830565677E+04	-1.329397396036127E+05	-9.105050353730567E+03	1.714987376475091E-01	6.761523762013204E-01	1.809694124695999E-01
-6.968887663012642E+03	4.400011444742745E+03	-8.276795864169537E+02	-6.839278558874583E+00	-5.968650733283848E+00	-2.902881116313941E+00
4.410340337104807E+04	-1.445700627057249E+05	-1.844800746187631E+04	6.085716664696780E-01	-3.556111381801396E-01	7.752567626599285E-02
7.465050716115248E+04	-1.200155733096704E+05	-5.978066056178676E+03	2.526124827737670E-02	9.297321220811799E-01	1.935761265125865E-01
-9.134978003496717E+03	-6.128474466796824E+04	-1.452944079104580E+04	7.865958749625235E-01	-2.655950269091409E+00	-3.420093027356714E-01
5.332171000273689E+04	-1.481645532401726E+05	-1.675222690439900E+04	5.280643922135257E-01	-1.077210299392201E-01	1.065341083532638E-01
7.372733556221463E+04	-1.029111424137851E+05	-2.736853126673946E+03	-1.715253524129725E-01	1.221587289590352E+00	2.025289354661174E-01

HEO Ephemeris

5.384966428588518E+07	1.268302287235357E+08	5.453607146699951E+07	-3.015968140474427E+01	1.272290165907758E+01	4.446791073318775E+00
5.123623353882399E+07	1.279095588929556E+08	5.491194578901459E+07	-3.033537173113337E+01	1.226199926550702E+01	4.253891013308101E+00
4.860791443416640E+07	1.289491416468753E+08	5.527112527370161E+07	-3.050406996299098E+01	1.180251645346221E+01	4.060347126136358E+00
4.596536366273962E+07	1.299490373528460E+08	5.561355727769589E+07	-3.066467903250459E+01	1.134318066629364E+01	3.866230846018227E+00
4.330930819095571E+07	1.309092308144029E+08	5.593919583766550E+07	-3.081660511875918E+01	1.088342298197128E+01	3.671624273110140E+00
4.064051449350177E+07	1.318296738094964E+08	5.624800236180461E+07	-3.095950095134766E+01	1.042300994453610E+01	3.476615358135437E+00
3.795977269296946E+07	1.327103063211564E+08	5.653994600392208E+07	-3.109315401877623E+01	9.961884407211361E+00	3.281295114176741E+00
3.526788751861329E+07	1.335510680805483E+08	5.681500385547214E+07	-3.121743239320493E+01	9.500088745339017E+00	3.085756065108877E+00
3.256567270202240E+07	1.343519052464523E+08	5.707316103059050E+07	-3.133225614669565E+01	9.037724479009068E+00	2.890091269770395E+00
2.985394724182938E+07	1.351127744471383E+08	5.731441068322179E+07	-3.143758126277413E+01	8.574929593296305E+00	2.694393690043925E+00
2.713353274977240E+07	1.358336452999639E+08	5.753875397876532E+07	-3.153339003002658E+01	8.111865038440598E+00	2.498755753003871E+00
2.440525145620481E+07	1.365145020055469E+08	5.774620003339599E+07	-3.161968504285253E+01	7.648706326800337E+00	2.303269030913576E+00
2.166992463758596E+07	1.371553443524007E+08	5.793676582907186E+07	-3.169648525529480E+01	7.185638038650064E+00	2.108023993178006E+00
1.892837135940802E+07	1.377561883316090E+08	5.811047613245959E+07	-3.176382155872687E+01	6.722850167064713E+00	1.913111020281741E+00
1.618140738421891E+07	1.383170664841524E+08	5.826736338666561E+07	-3.182174214117406E+01	6.260535393686230E+00	1.718615365803194E+00
1.342984397865602E+07	1.388380285029576E+08	5.840746740284287E+07	-3.187029979528572E+01	5.798887455624240E+00	1.524623436869387E+00
1.067448739304221E+07	1.393191390088597E+08	5.853083543844837E+07	-3.190955865277977E+01	5.338099619743105E+00	1.331222316594012E+00
7.916137965325995E+06	1.397604819645278E+08	5.863752196021936E+07	-3.193959131512925E+01	4.878363697210354E+00	1.138491786669094E+00
5.155589420393094E+06	1.401621560129980E+08	5.872758848094145E+07	-3.196047815368544E+01	4.419869250065306E+00	9.465133338176670E-01
2.393628225265332E+06	1.405242765052505E+08	5.880110338865907E+07	-3.197230671031736E+01	3.962802964982470E+00	7.553655181597795E-01
-3.689670072779814E+05	1.408469747794200E+08	5.885814176930434E+07	-3.197517117149319E+01	3.507348153524227E+00	5.651248122588244E-01
-3.131426071994661E+06	1.411303978510379E+08	5.889878522360810E+07	-3.196917189416668E+01	3.053684345688747E+00	3.758655395457420E-01
-5.893030939711476E+06	1.413747202008105E+08	5.892313061944572E+07	-3.195492865787046E+01	2.602135485797176E+00	1.877650651268254E-01
-8.652987590753678E+06	1.415801077295197E+08	5.893126323287374E+07	-3.193152527644088E+01	2.152575728970004E+00	6.828225895506648E-04
-1.141055283974718E+07	1.417467516458817E+08	5.892328291587008E+07	-3.189959191816154E+01	1.705319947251890E+00	-1.852083624813690E-01
-1.416499489053359E+07	1.418748580298086E+08	5.889295412451290E+07	-3.185924954558264E+01	1.260529743054753E+00	-3.698432154589614E-01

MTO Ephemeris

5.639810487255618E+07	1.222099771239883E+08	5.275721656886709E+07	-2.684299205764671E+01	7.678778274069781E+00	3.205351378428584E+00
5.406970726002696E+07	1.228533543446511E+08	5.302549783441531E+07	-2.705379431079633E+01	7.213395406277923E+00	3.004516056970143E+00
5.172343681936929E+07	1.234563114252949E+08	5.327633717364078E+07	-2.725669537416563E+01	6.743135516395827E+00	2.801606670965544E+00
4.935998126813414E+07	1.240184292836412E+08	5.350955650909426E+07	-2.745157750045522E+01	6.268052067444511E+00	2.596649178691908E+00
4.698003875647748E+07	1.245392935002198E+08	5.372497983848787E+07	-2.763831714991220E+01	5.788199520131315E+00	2.389666352645683E+00
4.458431818133900E+07	1.250184943784074E+08	5.392243326108490E+07	-2.781678871618985E+01	5.303632312354554E+00	2.180683152950470E+00
4.217353950567936E+07	1.254556270469959E+08	5.410174501363350E+07	-2.798686243629995E+01	4.814407893321462E+00	1.969724567011332E+00
3.974843405677543E+07	1.258502915647622E+08	5.426274551932418E+07	-2.814840457729240E+01	4.320583288335347E+00	1.756815892819379E+00
3.730974482963074E+07	1.262020930785600E+08	5.440526744798545E+07	-2.830127843829991E+01	3.822218825587557E+00	1.541984091375573E+00
3.485822678317241E+07	1.265106419736467E+08	5.452914578530622E+07	-2.844534421049967E+01	3.319376105307115E+00	1.325255738958200E+00
3.239464713302338E+07	1.267755540703024E+08	5.463421790730903E+07	-2.858045833577488E+01	2.812118017859313E+00	1.106660223094544E+00
2.991978564747888E+07	1.269964508193513E+08	5.472032367543867E+07	-2.870647361931729E+01	2.300513240228202E+00	8.862252679831671E-01
2.743443492298961E+07	1.271729595332684E+08	5.478730552589276E+07	-2.882323917018515E+01	1.784629319835740E+00	6.639817200161285E-01
2.493940070803127E+07	1.273047136228288E+08	5.483500857549673E+07	-2.893060239141409E+01	1.264538824622565E+00	4.399622835217393E-01
2.243550215368580E+07	1.273913528660450E+08	5.486328073899908E+07	-2.902840553366288E+01	7.403173846510001E-01	2.141988046460128E-01
1.992357212297454E+07	1.274325236846892E+08	5.487197284106313E+07	-2.911648921065736E+01	2.120426508006119E-01	-1.327420960565357E-02
1.740445747327391E+07	1.274278794468768E+08	5.486093875698676E+07	-2.919468936051738E+01	-3.202027189744879E-01	-2.424203287235873E-01
1.487901933549229E+07	1.273770807941093E+08	5.483003554132928E+07	-2.926283819592002E+01	-8.563321972415010E-01	-4.732015833335839E-01
1.234813340520212E+07	1.272797959697501E+08	5.477912358266823E+07	-2.932076689536005E+01	-1.396255581849617E+00	-7.055784925230666E-01
9.812690213036824E+06	1.271357011916398E+08	5.470806675487681E+07	-2.936830109176513E+01	-1.939879654473630E+00	-9.395109550456475E-01
7.273595412442015E+06	1.269444810330625E+08	5.461673258424544E+07	-2.940526520328839E+01	-2.487104535577983E+00	-1.174954764794409E+00
4.731770046355580E+06	1.267058288241755E+08	5.450499242627481E+07	-2.943147834552409E+01	-3.037825952400737E+00	-1.411863953513820E+00
2.188150819981151E+06	1.264194470820937E+08	5.437272164484794E+07	-2.944675853322527E+01	-3.591936176151948E+00	-1.650192261194593E+00
-3.563096293788769E+05	1.260850479559829E+08	5.421979981147813E+07	-2.945092009747767E+01	-4.149320084184895E+00	-1.889889057981603E+00
-2.900642480117455E+06	1.257023537155261E+08	5.404611090736855E+07	-2.944377324957837E+01	-4.709858461401081E+00	-2.130902201789429E+00
-5.443862467484711E+06	1.252710972157100E+08	5.385154353570618E+07	-2.942512723009175E+01	-5.273425035067571E+00	-2.373177738851362E+00

VTO Ephemeris

Appendix C: MATLAB Main Code

```
%This program solves n-body dynamics with optional J2 perturbation
%dynamics depend on rates equation applied in integrator declaration

clear all; close all; clc;
%...Constants
global muE muL muS R
muE    = 3.986004415e5;    %Earth
muL    = 4902.799;         %Moon
muS    = 1.32712428e11;  %Sun
R      = 6378;             %Earth Radius
%R      = 695700;           %Sun Radius
G      = 6.67259e-20;      %Gravitational Constant
hours  = 3600;             %conversion variable between seconds & hours
days   = hours*24;         %conversion variable between seconds & days

%...Retrieve Ephemeris Data
eph_LEO = importdata('/Users/angelrocha/Desktop/ephemeris_LEO.txt');
eph_GEO = importdata('/Users/angelrocha/Desktop/ephemeris_GEO.txt');
eph_MEO = importdata('/Users/angelrocha/Desktop/ephemeris_MEO.txt');
eph_HEO = importdata('/Users/angelrocha/Desktop/ephemeris_HEO.txt');
eph_MTO = importdata('/Users/angelrocha/Desktop/ephemeris_MTO.txt');
eph_VTO = importdata('/Users/angelrocha/Desktop/ephemeris_VTO.txt');

%------------------------------------------------------------------------
%...Input Data: propagation time & initial states beginning 01/01/2018
%***User input changes occur here
n    = 7;                  %number of tolerances to test
span = 5;                  %days to propagate
t0 = 0; tf = span*days; %initial and final times
f0   = eph_HEO(1,:)';      %retrieve initial ephemeris as initial states
ff = eph_HEO(span+1,:); %retrieve final ephemeris as final states
%------------------------------------------------------------------------

%...Initialize variables to capture propagation data
atol = zeros(n,1);     %list of absolute tolerances
rtol = zeros(n,1);     %list of relative tolerances
y    = zeros(n,6,6); %array of final states for each propagation
x    = zeros(n,1);     %extra variable used to define relative tolerance
time = zeros(n,6);     %list of computational times for each propagation
ns   = zeros(n,6);     %list of numbers of steps (correct calculations)
nf   = zeros(n,6);     %list of numbers of failed steps

%...Loop through Orbit Propagations
for i = 1:n
   x(i) = (7+i)
   z = x(i)+1;
   rtol(i) = 1*10^-(x(i));
   rtol9(i) = 1*10^-(x(i)+2);
   atol(i) = 1*10^-(z);
   dum = 1e-10;
   opts = odeset('Reltol', rtol(i), 'AbsTol', atol(i));

   tic
   [t1,f1, stats1] = rkf45(@nrates, [t0 tf], f0, rtol(i));
```

```
    time(i,1) = toc; tic
    q=1;
    [t2,f2, stats2] = ode45(@nrates, [t0 tf], f0, opts);
    time(i,2) = toc; tic
    q=2;
    [t3,f3, stats3] = DOPRI54(@nrates, [t0 tf], f0, atol(i), rtol(i));
    time(i,3) = toc; tic
    q=3;
    [t4,f4, stats4] = rkf89(@nrates, [t0 tf], f0, rtol9(i));
    time(i,4) = toc; tic
    q=4;
    [t5,f5, stats5] = ode87(@nrates, [t0 tf], f0, opts);
    time(i,5) = toc; tic
    q=5;
    [t6,f6, stats6] = ode113(@nrates, [t0 tf], f0, opts);
    time(i,6) = toc;
    q=6;

    y(i,:,1) = f1(end,:);   ns(i,1) = stats1(1);   nf(i,1) = stats1(2);
    y(i,:,2) = f2(end,:);   ns(i,2) = stats2(1);   nf(i,2) = stats2(2);
    y(i,:,3) = f3(end,:);   ns(i,3) = stats3(1);   nf(i,3) = stats3(2);
    y(i,:,4) = f4(end,:);   ns(i,4) = stats4(1);   nf(i,4) = stats4(2);
    y(i,:,5) = f5(end,:);   ns(i,5) = stats5(1);   nf(i,5) = stats5(2);
    y(i,:,6) = f6(end,:); ns(i,6) = stats6(1); nf(i,6) = stats6(2);
end

%...Calculate log scale difference between propagated states and
%...ephemeris states
diff = zeros(n,6,6);
StateError = ff-y;
diff = log10(abs(StateError)); %use when comparing to ephemeris solution
%diff = log10(abs(y-y(end,:,:)));%use when comparing to highest order solution

%...Calculate magnitude of vectorized error between propagated solution and
%...ephemeris states
Rerror = zeros(6,n);
VError = zeros(6,n);
for c = 1:n
   for r = 1:6 %dont forget to update
      Rerror(r,c) = norm(StateError(c,1:3,r));
      VError(r,c) = norm(StateError(c,4:6,r));
   end
end
%rnewnew
figure
plot(x,Rerror(1,:),'-r'); hold on;
plot(x,Rerror(2,:),'-g');
plot(x,Rerror(3,:),'-b');
plot(x,Rerror(4,:),'-c');
plot(x,Rerror(5,:),'-m');
plot(x,Rerror(6,:),'-k'); hold off;
title('Absolute Position Error')
xlabel('tolerance e-n')
ylabel('dr')
legend('RKF45','ODE45','DOPRI54','RKF89','DOPRI87','ODE113',...
       'Location','northwest')
```

```
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution
figure
plot(x,VError(1,:),'-r'); hold on;
plot(x,VError(2,:),'-g');
plot(x,VError(3,:),'-b');
plot(x,VError(4,:),'-c');
plot(x,VError(5,:),'-m');
plot(x,VError(6,:),'-k'); hold off;
title('Absolute Velocity Error')
xlabel('tolerance e-n')
ylabel('dv')
legend('RKF45','ODE45','DOPRI54','RKF89','DOPRI87','ODE113',...
       'Location','northwest')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%...Calculate initial & final orbital elements from state vectors
%...simplification: only earth gravitational parameter considered
coe0 = zeros(1,7);
[coe0] = coe_from_sv(f0(1:3),f0(4:6),muS);
period = 2*pi*coe0(7).^1.5/sqrt(muS)/60;

%---------------------------------------------------------------------
%% Print Results
%...Orbital Element Results
fprintf('Initial Orbital Elements\n')
fprintf('Angular momentum (km^2/s) = %g\n', coe0(1))
fprintf('Eccentricity              = %g\n', coe0(2))
fprintf('Right ascension (deg)     = %g\n', coe0(3))
fprintf('Inclination (deg)         = %g\n', coe0(4))
fprintf('Argument of perigee (deg) = %g\n', coe0(5))
fprintf('True anomaly (deg)        = %g\n', coe0(6))
fprintf('Semimajor axis (km):      = %g\n', coe0(7))
fprintf('Orbital Period (min):     = %g\n', period)

%---------------------------------------------------------------------
%% Plot Results
%...2D X & VX Resuts
figure
subplot(2,2,1)
plot(x,y(:,1,1),'-r'); hold on;
plot(x,y(:,1,2),'-g');
plot(x,y(:,1,3),'-b');
plot(x,y(:,1,4),'-c');
plot(x,y(:,1,5),'-m');
plot(x,y(:,1,6),'-k'); hold off;
title('X Convergence Results')
xlabel('tolerance e-n')
ylabel('X (km)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution
```

```
subplot(2,2,2)
plot(x,y(:,4,1),'-r'); hold on;
plot(x,y(:,4,2),'-g');
plot(x,y(:,4,3),'-b');
plot(x,y(:,4,4),'-c');
plot(x,y(:,4,5),'-m');
plot(x,y(:,4,6),'-k'); hold off;
title('VX Convergence Results')
xlabel('tolerance e-n')
ylabel('VX (km/s)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

subplot(2,2,3)
plot(x,diff(:,1,1),'-r'); hold on;
plot(x,diff(:,1,2),'-g');
plot(x,diff(:,1,3),'-b');
plot(x,diff(:,1,4),'-c');
plot(x,diff(:,1,5),'-m');
plot(x,diff(:,1,6),'-k'); hold off;
legend('RKF45','ODE45','DOPRI54','RKF89','DOPRI87','ODE113',...
      'Location','southwest')
title('X Convergence Results (log)')
xlabel('tolerance e-n')
ylabel('log(diff)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

subplot(2,2,4)
plot(x,diff(:,4,1),'-r'); hold on;
plot(x,diff(:,4,2),'-g');
plot(x,diff(:,4,3),'-b');
plot(x,diff(:,4,4),'-c');
plot(x,diff(:,4,5),'-m');
plot(x,diff(:,4,6),'-k'); hold off;
title('VX Convergence Results (log)')
xlabel('tolerance e-n')
ylabel('log(diff)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

%...2D Y & VY Results
figure
subplot(2,2,1)
plot(x,y(:,2,1),'-r'); hold on;
plot(x,y(:,2,2),'-g');
plot(x,y(:,2,3),'-b');
plot(x,y(:,2,4),'-c');
plot(x,y(:,2,5),'-m');
plot(x,y(:,2,6),'-k'); hold off;
title('Y Convergence Results')
xlabel('tolerance e-n')
ylabel('Y (km)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution
```

```
subplot(2,2,2)
plot(x,y(:,5,1),'-r'); hold on;
plot(x,y(:,5,2),'-g');
plot(x,y(:,5,3),'-b');
plot(x,y(:,5,4),'-c');
plot(x,y(:,5,5),'-m');
plot(x,y(:,5,6),'-k'); hold off;
title('VY Convergence Results')
xlabel('tolerance e-n')
ylabel('VY (km/s)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

subplot(2,2,3)
plot(x,diff(:,2,1),'-r'); hold on;
plot(x,diff(:,2,2),'-g');
plot(x,diff(:,2,3),'-b');
plot(x,diff(:,2,4),'-c');
plot(x,diff(:,2,5),'-m');
plot(x,diff(:,2,6),'-k'); hold off;
legend('RKF45','ODE45','DOPRI54','RKF89','DOPRI87','ODE113',...
      'Location','southwest')
title('Y Convergence Results (log)')
xlabel('tolerance e-n')
ylabel('log(diff)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

subplot(2,2,4)
plot(x,diff(:,5,1),'-r'); hold on;
plot(x,diff(:,5,2),'-g');
plot(x,diff(:,5,3),'-b');
plot(x,diff(:,5,4),'-c');
plot(x,diff(:,5,5),'-m');
plot(x,diff(:,5,6),'-k'); hold off;
title('VY Convergence Results (log)')
xlabel('tolerance e-n')
ylabel('log(diff)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

%...2D Z & VZ Results
figure
subplot(2,2,1)
plot(x,y(:,3,1),'-r'); hold on;
plot(x,y(:,3,2),'-g');
plot(x,y(:,3,3),'-b');
plot(x,y(:,3,4),'-c');
plot(x,y(:,3,5),'-m');
plot(x,y(:,3,6),'-k'); hold off;
title('Z Convergence Results')
xlabel('tolerance e-n')
ylabel('Z (km)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution
```

```
subplot(2,2,2)
plot(x,y(:,6,1),'-r'); hold on;
plot(x,y(:,6,2),'-g');
plot(x,y(:,6,3),'-b');
plot(x,y(:,6,4),'-c');
plot(x,y(:,6,5),'-m');
plot(x,y(:,6,6),'-k'); hold off;
title('VZ Convergence Results')
xlabel('tolerance e-n')
ylabel('VZ (km/s)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

subplot(2,2,3)
plot(x,diff(:,3,1),'-r'); hold on;
plot(x,diff(:,3,2),'-g');
plot(x,diff(:,3,3),'-b');
plot(x,diff(:,3,4),'-c');
plot(x,diff(:,3,5),'-m');
plot(x,diff(:,3,6),'-k'); hold off;
legend('RKF45','ODE45','DOPRI54','RKF89','DOPRI87','ODE113',...
      'Location','southwest')
title('Z Convergence Results (log)')
xlabel('tolerance e-n')
ylabel('log(diff)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

subplot(2,2,4)
plot(x,diff(:,6,1),'-r'); hold on;
plot(x,diff(:,6,2),'-g');
plot(x,diff(:,6,3),'-b');
plot(x,diff(:,6,4),'-c');
plot(x,diff(:,6,5),'-m');
plot(x,diff(:,6,6),'-k'); hold off;
title('VZ Convergence Results (log)')
xlabel('tolerance e-n')
ylabel('log(diff)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]) %use when comparing to highest order solution

%...Plot Run Times, Total Steps, Number of Errors
figure
subplot(1,3,1)
plot(x,time(:,1),'-r'); hold on;
plot(x,time(:,2),'-g');
plot(x,time(:,3),'-b');
plot(x,time(:,4),'-c');
plot(x,time(:,5),'-m');
plot(x,time(:,6),'-k'); hold off;
legend('RKF45','ODE45','DOPRI54','RKF89','DOPRI87','ODE113',...
      'Location','northwest')
title('Run Times')
xlabel('tolerance e-n')
ylabel('time (s)')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
```

```
%xlim([x(1),x(end-1)]); %use when comparing to highest order solution

subplot(1,3,2)
plot(x,ns(:,1),'-r'); hold on;
plot(x,ns(:,2),'-g');
plot(x,ns(:,3),'-b');
plot(x,ns(:,4),'-c');
plot(x,ns(:,5),'-m');
plot(x,ns(:,6),'-k'); hold off;
legend('RKF45','ODE45','DOPRI54','RKF89','DOPRI87','ODE113',...
      'Location','northwest')
title('# Steps')
xlabel('tolerance e-n')
ylabel('# Steps')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]); %use when comparing to highest order solution

subplot(1,3,3)
plot(x,nf(:,1),'-r'); hold on;
plot(x,nf(:,2),'-g');
plot(x,nf(:,3),'-b');
plot(x,nf(:,4),'-c');
plot(x,nf(:,5),'-m');
plot(x,nf(:,6),'-k'); hold off;
legend('RKF45','ODE45','DOPRI54','RKF89','DOPRI87','ODE113',...
      'Location','northwest')
title('# Failed')
xlabel('tolerance e-n')
ylabel('# Failed Steps')
xlim([x(1), x(end)]); %use when comparing to ephemeris solution
%xlim([x(1),x(end-1)]); %use when comparing to highest order solution


%% Functions
function dfdt = Erates(t,f)
global muE
% f  -
% t  -
% mu -
% J2 - oblateness coefficient
% R - mean radius of Earth (km)

rx  = f(1);     % X component of r (ECI frame) (km)
ry  = f(2);     % Y component of r (ECI frame) (km)
rz  = f(3);     % Z component of r (ECI frame) (km)
vx  = f(4);     % X component of v (ECI frame) (km/s)
vy  = f(5);     % Y component of v (ECI frame) (km/s)
vz  = f(6);     % Z component of v (ECI frame) (km/s)
r = norm([rx ry rz]); % magnitude of position vector (km)

%...Linearized Acceleration
ax = -muE*rx/r^3; % x component of a (ECI frame) (km/s^2)
ay = -muE*ry/r^3; % y component of a (ECI frame) (km/s^2)
az = -muE*rz/r^3; % z component of a (ECI frame) (km/s^2)

%...Output Vector
```

```
dfdt = [vx vy vz ax ay az]';
end %rates
function dfdt = Srates(t,f)
global muS
% f  -
% t  -
% mu -
% J2 - oblateness coefficient
% R - mean radius of Earth (km)

rx  = f(1);     % X component of r (ECI frame) (km)
ry  = f(2);     % Y component of r (ECI frame) (km)
rz  = f(3);     % Z component of r (ECI frame) (km)
vx  = f(4);     % X component of v (ECI frame) (km/s)
vy  = f(5);     % Y component of v (ECI frame) (km/s)
vz  = f(6);     % Z component of v (ECI frame) (km/s)
r = norm([rx ry rz]); % magnitude of position vector (km)

%...Linearized Acceleration
ax = -muS*rx/r^3; % x component of a (ECI frame) (km/s^2)
ay = -muS*ry/r^3; % y component of a (ECI frame) (km/s^2)
az = -muS*rz/r^3; % z component of a (ECI frame) (km/s^2)

%...Output Vector
dfdt = [vx vy vz ax ay az]';
end %rates
function dfdt = nrates(t,f)
global muE muL muS R
% This function evaluates acceleration of each member of 3 body system at
% time t from their positions and velocities at that time.
% t           - time (s)
% f           - column vector of position and velocity componenets
% R12           - cube of distance between m1 and m2 (km^3)
% R13           - cube of distance between m1 and m3 (km^3)
% R23           - cube of distance between m2 and m3 (km^3)
% AX1,AY1,AY3 - acceleration components of mass 1 (km/s^2)
% dydt          - column vector of velocity and acceleration componenets at
%                 time t

%...Initial Conditions (Particle radius & velocity components)
prx = f(1);
pry = f(2);
prz = f(3);
pvx = f(4);
pvy = f(5);
pvz = f(6);

%...Caclulate Vector & Scalar Radii (km)
JD_0 = 2458119.500000000;               % (2018-Jan-01 00:00:00.00)
JD = JD_0 + t/86400;                    % Julian Date
r_moon = lunar_position(JD);            % ECI frame lunar position
lrx = r_moon(1);
lry = r_moon(2);
lrz = r_moon(3);
% [lam eps r_sun] = solar_position(JD); % Ecliptic frame solar position
% srx = r_sun(1);
```

```
% sry = r_sun(2);
% srz = r_sun(3);
RE = norm([prx pry prz]);                % Scalar distance Earth/Particle
RL = norm([lrx-prx lry-pry lrz-prz]);    % Scalar distance Lunar/Particle
%RS = norm([srx-prx sry-pry srz-prz]);    % Scalar distance Solar/Particle

%...J2 Purturbation (from Curtix 12.30)
J2 = 0.00108263;
fac = (3/2) * (J2*muE*R^2) / (RE^5);
J2x = fac*(prx)*(5*prz^2/RE^2-1);
J2y = fac*(pry)*(5*prz^2/RE^2-1);
J2z = fac*(prz)*(5*prz^2/RE^2-3);

%...Lunar Purturbation
lax = muL*(lrx-prx)/RL^3;
lay = muL*(lry-pry)/RL^3;
laz = muL*(lrz-prz)/RL^3;

%...Final Particle Acceleration States
pax = -muE*prx/RE^3 + J2x;% - lax;
pay = -muE*pry/RE^3 + J2y;% - lay;
paz = -muE*prz/RE^3 + J2z;% - laz;

%...Particle Output Vector
dfdt = [pvx pvy pvz pax pay paz]';
end %nrates
function r_moon = lunar_position(jd)
%... Calculate geocentric equatorial position vector of moon given JD
RE = 6376; %Earth radius (km);

%...Time in centuries since J2000
T = (jd-2451545)/36525;

%...Ecliptic longitude (deg):
e_long = 218.32 + 481267.881*T ...
       + 6.29*sind(135.0 + 477198.87*T) - 1.27*sind(259.3 - 413335.36*T)...
       + 0.66*sind(235.7 + 890534.22*T) + 0.21*sind(269.9 + 954397.74*T)...
       - 0.19*sind(357.5 + 35999.05*T) - 0.11*sind(186.5 + 966404.03*T);
e_long = mod(e_long,360);


%...Ecliptic latitude (deg):
e_lat = 5.13*sind( 93.3 + 483202.02*T) + 0.28*sind(228.2 + 960400.89*T)...
     - 0.28*sind(318.3 + 6003.15*T) - 0.17*sind(217.6 - 407332.21*T);
e_lat = mod(e_lat,360);

%...Horizontal parallax (deg):
h_par = 0.9508 ...
     + 0.0518*cosd(135.0 + 477198.87*T) + 0.0095*cosd(259.3 - 413335.36*T)...
     + 0.0078*cosd(235.7 + 890534.22*T) + 0.0028*cosd(269.9 + 954397.74*T);
h_par = mod(h_par,360);


%...Angle between earth's orbit and its equator (deg):
obliquity = 23.439291 - 0.0130042*T;
```

```
%...Direction cosines of the moon's geocentric equatorial position vector:
l = cosd(e_lat) * cosd(e_long);
m = cosd(obliquity)*cosd(e_lat)*sind(e_long) - sind(obliquity)*sind(e_lat);
n = sind(obliquity)*cosd(e_lat)*sind(e_long) + cosd(obliquity)*sind(e_lat);

%...Earth-moon distance (km):
dist = RE/sind(h_par);
%...Moon's geocentric equatorial position vector (km):
r_moon = dist*[l m n];
end %lunar_position()
function [lamda eps r_S] = solar_position(jd)
% This function calculates the geocentric equatorial position vector
% of the sun, given the Julian date.
%
% User M-functions required: None
% ----------------------------------------------------------------------
%...Astronomical unit (km):
AU = 149597870.691;
%...Julian days since J2000:
n = jd - 2451545;
%...Julian centuries since J2000:
cy = n/36525;
%...Mean anomaly (deg{:
M = 357.528 + 0.9856003*n;
M = mod(M,360);
%...Mean longitude (deg):
L = 280.460 + 0.98564736*n;
L = mod(L,360);
%...Apparent ecliptic longitude (deg):
lamda = L + 1.915*sind(M) + 0.020*sind(2*M);
lamda = mod(lamda,360);
%...Obliquity of the ecliptic (deg):
eps = 23.439 - 0.0000004*n;
%...Unit vector from earth to sun:
u = [cosd(lamda); sind(lamda)*cosd(eps); sind(lamda)*sind(eps)];
%...Distance from earth to sun (km):
rS = (1.00014 - 0.01671*cosd(M) - 0.000140*cosd(2*M))*AU;
%...Geocentric position vector (km):
r_S = rS*u;
end %solar_position
```

Appendix D: MATLAB Variation of Parameters Code

```
% This program solves 2body + J2 perturbation dynamics while varying
% orbital elements and tolerances

% Each of the following numbers represents an orbital element state. Select
% vary = 4 to vary inclination. Only elements 3,4,5 should be varied.

% (1) h - angular momentum
% (2) e - eccentricity
% (3) RA - Right Ascension
% (4) incl - inclination
% (5) w -
% (6) TA   - True Anomaly
% (7) a    - semi-major axis

clear all; close all; clc;
%...Constants
global muE muL muS R
muE    = 3.986004415e5;    %Earth
muL    = 4902.799;         %Moon
muS    = 1.32712428e11;  %Sun
R      = 6378;             %Earth Radius
G      = 6.67259e-20;      %Gravitational Constant
hours  = 3600;             %conversion variable between seconds & hours
days   = hours*24;         %conversion variable between seconds & days

%...Retrieve Ephemeris Data
eph_LEO = importdata('/Users/angelrocha/Desktop/ephemeris_LEO.txt');
eph_GEO = importdata('/Users/angelrocha/Desktop/ephemeris_GEO.txt');
eph_MEO = importdata('/Users/angelrocha/Desktop/ephemeris_MEO.txt');
eph_HEO = importdata('/Users/angelrocha/Desktop/ephemeris_HEO.txt');

%-------------------------------------------------------------------------
%...Input Data: propagation time & initial states beginning 01/01/2018
%***User input changes occur here
vary = 4;                  %select orbital element to vary
n    = 6;                  %number of times to vary orbital element
ntol = 9;                  %number of tolerances to test
span = 25;                 %days to propagate
t0 = 0; tf = span*days; %initial and final times
f0   = eph_MEO(1,:)';    %retrieve initial ephemeris as initial states
ff  = eph_MEO(span+1,:); %retrieve final ephemeris as final states
r0   = f0(1:3);            %initial position vector from ephemeris
v0   = f0(4:6);            %initial velocity vector from ephemeris
%-------------------------------------------------------------------------

%...Initialize variables to capture propagation data
atol = zeros(n,1);      %list of absolute tolerances
rtol = zeros(n,1);      %list of relative tolerances
y    = zeros(n,6,6); %array of final states for each propagation
x    = zeros(n,1);      %extra variable used to define relative tolerance
time = zeros(ntol,n);      %list of computational times for each propagation
ns   = zeros(ntol,n);      %list of numbers of steps (correct calculations)
nf   = zeros(ntol,n);      %list of numbers of failed steps
```

```
%...Retrieve State Vectors from Varied Orbital Element
state = sv_vary_coe(r0,v0,muE,vary,n);
variations = linspace(0,90,n+1);
variations = variations(1:end-1);
%...Loop through propagation for varied states and tolerances
for i = 1:ntol %loop through tolerances
   for j = 1:n %loop through varied states
      x(i) = (3+i);
      z = x(i)+1;
      rtol(i) = 1*10^-(x(i)); %set relative tolerance
      atol(i)  = 1*10^-(z);    %set absolute tolerance (1e-1 tighter)
      opts = odeset('Reltol', 1e-4, 'AbsTol', 1e-5, 'stats', 'on');
      tic
      [t, f, stats] = ode87(@nrates,[t0 tf], state(j,:)');
      %...Store data
      time(i,j) = toc;      y(j,:,i) = f(end,:);
      ns(i,j) = stats(1); nf(i,j) = stats(2);
   end
end

%% Text Results
switch vary
   case 1
      fprintf('Cannot vary angular momentum, select vary = 3,4, or 5.\n')
   case 2
      fprintf('Cannot vary eccentricity, select vary = 3,4, or 5.\n')
   case 3
      fprintf('%g variations of Right Ascension\n',n)
      fprintf('Values of Right Ascension (deg) tested: %g\n', variations)
   case 4
      fprintf('%g variations of Inclination\n',n)
      fprintf('Values of Inclination (deg) tested: %g\n', variations)
   case 5
      fprintf('%g variations of Argument of Periapsis\n',n)
      fprintf('Values of Argument of Periapsis (deg) tested: %g\n', variations)
   case 6
      fprintf('Cannot vary True Anomaly, select vary = 3,4, or 5.\n')
   case 7
      fprintf('Cannot vary semi-major axis, select vary = 3,4, or 5.\n')
end


%...Plot Run Times, Total Steps, Number of Errors
figure
subplot(1,3,1)
plot(x,time(:,1),'-r'); hold on;
plot(x,time(:,2),'-g');
plot(x,time(:,3),'-b');
plot(x,time(:,4),'-c');
plot(x,time(:,5),'-m');
plot(x,time(:,6),'-k'); hold off;
title('Run Times')
xlabel('tolerance e-n')
ylabel('time (s)')
%xlim([x(1), x(end)]); %use when comparing to ephemeris solution
```

```
xlim([x(1),x(end-1)]); %use when comparing to highest order solution

subplot(1,3,2)
plot(x,ns(:,1),'-r'); hold on;
plot(x,ns(:,2),'-g');
plot(x,ns(:,3),'-b');
plot(x,ns(:,4),'-c');
plot(x,ns(:,5),'-m');
plot(x,ns(:,6),'-k'); hold off;
title('# Steps')
xlabel('tolerance e-n')
ylabel('# Steps')
%xlim([x(1), x(end)]); %use when comparing to ephemeris solution
xlim([x(1),x(end-1)]); %use when comparing to highest order solution

subplot(1,3,3)
plot(x,nf(:,1),'-r'); hold on;
plot(x,nf(:,2),'-g');
plot(x,nf(:,3),'-b');
plot(x,nf(:,4),'-c');
plot(x,nf(:,5),'-m');
plot(x,nf(:,6),'-k'); hold off;
title('# Failed')
xlabel('tolerance e-n')
ylabel('# Failed Steps')
%xlim([x(1), x(end)]); %use when comparing to ephemeris solution
xlim([x(1),x(end-1)]); %use when comparing to highest order solution

function dfdt = nrates(t,f)
global muE muL muS R
% This function evaluates acceleration of each member of 3 body system at
% time t from their positions and velocities at that time.
% t            - time (s)
% f            - column vector of position and velocity componenets
% R12          - cube of distance between m1 and m2 (km^3)
% R13          - cube of distance between m1 and m3 (km^3)
% R23          - cube of distance between m2 and m3 (km^3)
% AX1,AY1,AY3 - acceleration components of mass 1 (km/s^2)
% dydt         - column vector of velocity and acceleration componenets at
%                time t

%...Initial Conditions (Particle radius & velocity components)
prx = f(1);
pry = f(2);
prz = f(3);
pvx = f(4);
pvy = f(5);
pvz = f(6);

%...Caclulate Vector & Scalar Radii (km)
JD_0 = 2458119.500000000;              % (2018-Jan-01 00:00:00.00)
JD = JD_0 + t/86400;                   % Julian Date
r_moon = lunar_position(JD);           % ECI frame lunar position
lrx = r_moon(1);
lry = r_moon(2);
lrz = r_moon(3);
```

```
% [lam eps r_sun] = solar_position(JD); % Ecliptic frame solar position
% srx = r_sun(1);
% sry = r_sun(2);
% srz = r_sun(3);
RE = norm([prx pry prz]);                % Scalar distance Earth/Particle
RL = norm([lrx-prx lry-pry lrz-prz]);    % Scalar distance Lunar/Particle
%RS = norm([srx-prx sry-pry srz-prz]);    % Scalar distance Solar/Particle

%...J2 Purturbation (from Curtix 12.30)
J2 = 0.00108263;
fac = (3/2) * (J2*muE*R^2) / (RE^5);
J2x = fac*(prx)*(5*prz^2/RE^2-1);
J2y = fac*(pry)*(5*prz^2/RE^2-1);
J2z = fac*(prz)*(5*prz^2/RE^2-3);

%...Lunar Purturbation
lax = muL*(lrx-prx)/RL^3;
lay = muL*(lry-pry)/RL^3;
laz = muL*(lrz-prz)/RL^3;

%...Final Particle Acceleration States
pax = -muE*prx/RE^3 + J2x;% + lax;
pay = -muE*pry/RE^3 + J2y;% + lay;
paz = -muE*prz/RE^3 + J2z;% + laz;

%...Particle Output Vector
dfdt = [pvx pvy pvz pax pay paz]';
end %nrates
```

References

[1] Vallado, D. A., *Fundamentals of Astrodyamics and Applications,* 2nd ed., Space Technology Library, California, 2007, Chap. 8.

[2] Curtis, H. D., *Orbital Mechanics for Engineering Students*, 3nd ed., Elsevier Ltd., Oxford, UK, 2014, Chaps. 1,2,4,12.

[3] Verner, H. K., "Explicit Runge-Kutta Methods with Estimates of the Local Truncation Error" *SIAM Journal on Numerical Analysis*, Vol. 15, 1978, pp. 772-790.

[4] Dormand, J. R., and Prince, P. J., "A Family of Embedded Runge-Kutta Formulae," *Journal of Computational and Applied Mathematics*, Vol. 6, 1980, pp. 19-26.

[5] Prince, P. J., and Dormand, J. R., "High Order Embedded Runge-Kutta Formulae," *Journal of Computational and Applied Mathematics,* Vol. 7, 1981, pp. 67-75.

[6] Calvo, M., Montijano, J. I., and Randez, L., "A Fifth-Order Interpolant for the Dormand and Prince Runge-Kutta Method," *Journal of Computational and Applied Mathematics*, Vol. 29, 1990.

[7] Shampine, L. F., and Reichelt, M. W., "The MATLAB ODE Suite," *SIAM Journal on Scientific Computing*, Vol. 18, 1997.

[8] Higham, D. J., "The Tolerance Proportionality of Adaptive ODE Solvers," *Journal of Computational and Applied Mathematics*, Vol. 45, 1993, pp. 227-236.

[9] "HORIZONS Web-Interface Solar System Dynamics", *Jet Propulsion Laboratory* [online database], https://ssd.jpl.nasa.gov/horizons.cgi [retrieved 20 March 2018].

[10] Aristoff, J. M., and Poore, A. B., "Implicit Runge-Kutta Methods for Orbit Propagation," *AIAA/AAS Astrodynamics Specialist Conference*, 2012. (Analysis on local and global error, also different orbit types have different costs)

[11] Ritschel, T., "Numerical Methods for Solution of Differential Equations," Technical University of Denmark.

[12] Govorukhin, V., "ode87 Integrator," *MathWorks File Exchange*, https://www.mathworks.com/matlabcentral/fileexchange/3616-ode87-integrator [retrieved 5 January 2018].

[13] Urutxua, H., Bombardelli, C., Pelaez, J., and Huhn, A., "High Fidelity Models for Orbit Propagation: DROMO vs. Stormer-Cowell", *European Space Surveillance Conference*, 2011. (RK F 67 vs RKF 78, physical model conclusions)

[14] Berry, M. M., and Healy, L. M., "Implementation of Gauss-Jackson Integration for Orbit Propagation," *The Journal of the Astronautical Sciences*, Vol. 52, 2004, pp. 356.

[15] Skeel, R. D., "Thirteen Ways to Estimate Global Error," *Numerische Mathematik*, Vol. 48, 1986, pp1-20.

[16] "Types of Solvers," *MathwWorks*, https://www.mathworks.com/help/simulink/ug/types-of-solvers.html [retrieved 8 November 2017].

[17] Fehlberg, E., "Classical Fifth-, Sixth-, Seventh-, and Eighth-Order Runge-Kutta Formulas with Stepsize Control," *National Aeronautics and Space Administration*, [Retrieved 8 November 2017].