AE 370: Veracity of Two-Body Dynamics for Modeling a Lunar Orbiter's Trajectory

Ameya Chandra, Dhruva Chowlur, Pranet Patil, Colin Reedy, Santiago Zapata Zuluaga
University of Illinois, Department of Aerospace Engineering

The effect of Earth's gravity on the trajectory of a lunar orbiter, when varying the size of the orbit, is analyzed. The Runge-Kutta-Fehlberg method was derived, discussed, and implemented to solve this problem.

I. Nomenclature

F	=	Force
G	=	Universal Gravitational Constant
m	=	Mass of Lunar Orbiter
M_E	=	Mass of Earth
M_M	=	Mass of the Moon
r	=	Radial Distance between Two Objects
$y(t)$	=	Exact Solution
y_{n+1}	=	Numerical Solution
ε	=	Desired Tolerance
$x(t)$	=	x-component of position
$y(t)$	=	y-component of position
$v_x(t)$	=	x-component of velocity
$v_y(t)$	=	y-component of velocity
h	=	step-size
t	=	time

II. Introduction

There is a rising interest in human exploration and colonization of the moon. A key component of such efforts would be developing effective infrastructure in lunar orbit, and part of that would involving computationally efficient modeling of trajectories of any such orbiters.

The simplest way to model this is by using a two-body system, consisting of the moon and the orbiter. This system has an analytical solution that would provide quick, predictable solutions that is resistant to chaos. However, a large enough orbit would be far more subject to the influence of Earth's gravity, becoming a three-body problem. The three-body problem would need to be solved numerically and is an inherently chaotic problem, forcing any system to constantly recalculate the system.

We seek to probe the boundary of the applicability of the two-body system. The first question this group seeks to answer is how large the orbit may be before any dynamical model must account for the influence of Earth's gravity. This will provide an estimate of the orbital radius at which the Earth's gravitational influence becomes significant enough to alter a single orbit beyond a typical satellite's ability to correct it. For planning a lunar mission, its crucial to understand at what altitudes a spacecraft can sit in a stable orbit for mission planning.

The second question is how the rate of change in orbit perturbations changes over time and subsequent orbits; that is, at differing initial orbit radii, how much will eccentricity change for each orbit, and does the rate of eccentricity change increase at higher altitudes? Answering this question will show how long an orbit can remain stable and give context to whether orbital altitude affects the rate of orbit perturbation.

To do this, we have implemented and discussed the Runge-Kutta-Fehlberg method (RK45) to model both the two-body and three-body system, and are using the use of an analytical two-body solution. We will then plot circular orbits with different altitudes as initial conditions, and use the change in eccentricity driven by the three-body system as a metric to determine how significantly a circular lunar orbit is altered by the earths gravitational influence.

III. Dynamical System of a Lunar Orbiter

The system we have elected to study is that of the a lunar orbiter.

A. Mathematical Description

The preliminary step in developing our mathematical formulation is to determine the bodies comprising our system. For a model of a lunar orbiter, the system will contain the Moon, Earth, and a lunar orbiter. All bodies will be described as a point mass, and so we are neglecting any possible moments on the spacecraft induced by variations in distributions of mass throughout the spacecraft.

We shall only consider the effects of gravity. This paper is primarily interested in analyzing the perturbing effects of Earth's gravity on a lunar orbit, so no secondary forces, such as spacecraft thrust, radiation pressure, or momentum transfers from debris located in the orbit will be considered. This reduces our problem to one described by a familiar equation, Newton's law of gravitation.

$$\vec{F} = \frac{Gm_1m_2}{r^2}\hat{r} \tag{1}$$

where G is the universal gravitational constant m_1 is the mass of object one, m_2 is the mass of object two, r is the distance between the center of mass of both objects, and F is the gravitational force.

1. State Vector

The state vector for the spacecraft consists of four elements: x and y components of position, and x and y components of velocity. These states are derived from the 2nd derivative and the derivative of acceleration, respectively, which is itself derived from the equation for gravitational force on a body divided by the spacecraft's mass.

$$\mathbf{y}(t) = \begin{bmatrix} x(t) \\ y(t) \\ v_x(t) \\ v_y(t) \end{bmatrix}$$

B. Simulation Parameters and Assumptions

1. Assumptions for the Model

When designing this model, there are certain compromises that must be made to simplify the system enough to be efficient and calculable. In our models, we assume the force the spacecraft exerts on the earth and the Moon is negligible and does not affect their position over time. Additionally, we model the earth and moon as perfectly circular and uniformly dense stationary bodies. Since we are focused on relatively few lunar orbits of the spacecraft, this assumption significantly reduces complexity while giving accurate results needed for the experiment.

For our simulation, we will use strictly circular orbits as the starting point for the satellite. Elliptical orbits become a significantly more complex problem since the orientation of the ellipse relative to the earth drastically changes how the orbit is influenced. Using a circular orbit ensures consistency and provides a good baseline for comparison since we can analyze the change in the eccentricity of the orbit due to the Earth's influence. A two-body system will have zero change in eccentricity, making the two models easy to compare and identify the point at which Earth's gravity becomes significant.

2. Fixed Parameters

We will use the gravitational constant G, the mass of the Earth M_E and Moon M_M, and the average distance between the earth and moon as fixed parameters for our model. We will also fix the position of the moon at the origin and fix the position of the Earth at the average distance between the Moon and the Earth.

3. Initial Conditions Of Orbits

We will model different orbits in our system by inputting an altitude for the orbit into our code. This altitude will be added to the radius of the moon in the code to give an orbital radius of the spacecraft. Using the orbit radius, the orbital

speed can be found for the circular orbit using Kepler's 3rd law.

$$v = \sqrt{\frac{GM}{r}} \tag{2}$$

We will define a time duration for which we will model the satellite. For the two-body system, we only need to measure the time it takes to complete one orbit because there are no external forces on the satellite other than the moon, so the orbit will be perfectly circular. This is the equation that we use to determine the time it takes to complete one orbit for a circular orbit.

$$t_{orbit} = \sqrt{\frac{4\pi^2 r^3}{GM_{\text{moon}}}} \tag{3}$$

Our other inputs for the function include static time step, dynamic time step, and tolerance of the system. It is important to define these two because they allow us to better define the system and use as little computational power as possible, while also using information from our other inputs (altitude, total time duration). For a two-body system, the static time step does not need to be small, while the dynamic time step needs to start off small in order to maintain a smooth projected orbit. The tolerance was something we included in our inputs to experiment and determine this variable's effect on the system.

IV. Numerical Method

The method chosen to find the answers to our questions is the Runge-Kutta-Fehlberg method (RK45). This is a single-step explicit method that belongs to the Runge-Kutta family of numerical methods. This method is unique, as it utilizes both Runge-Kutta 4 (RK4) and Runge-Kutta 5 (RK5). To find accurate predictions, the method uses RK5; at the same time, the method uses RK4 to find errors in the predictions and adjust the time step to keep the method accurate and efficient.

A. Justification of Choice

RK45 was chosen for our system, as it balances accuracy, stability, and cost. The RK45 method offers great accuracy because it uses RK5 to numerically solve our ODE. The accuracy of RK5 is due to its global truncation error of $O(h^5)$. This means that if time step h is halved, the error will be reduced by a factor of $2^5 = 32$. For finding the answer to our guiding questions, we need a high level of accuracy in our models, which is offered by the RK5 method.

At the same time, RK45 offers improved computational efficiency (cost) due to its adaptive size steps. RK45 does this by computing RK5 and RK4. The accuracy of these two methods is different. The error of the answer provided by RK5 is obtained by comparing RK5 with RK4. If both answers are similar enough, then the time step is kept, but if the answers are different enough, the time step is made smaller. This ensures that longer time steps can be used without losing accuracy, which improves computational efficiency. To find the set of initial conditions which answer our two questions, we need to iterate through our simulations. To accomplish this in a short amount of time, we need high computational efficiency. The adaptable time step of RK45 guarantees computational efficiency.

In terms of stability, it is difficult to analyze the stability of RK45 alone, so analysis of the embedded methods is required. Fig. 1 shows the stability regions of RK2, RK3, RK4, and RK5. As seen, the RK4 and RK5 regions are larger than the lower-order ones, though they are limited to small step sizes. This makes sense, as explicit methods are less applicable to stiff problems, which this system likely is. However, the adaptive time-step of RK45 helps to ensure the the methods remains in the stable regions throughout the computation.

RK45 also is computationally cost-effective. RK5 uses all the same coefficients (k_n) as RK4 and only requires one more. It requires only six calculations to implement a fourth-order and fifth-order method, which makes it fairly efficient overall. Additionally, unlike other methods, RK45 does not need to solve differential equations over and over in itself everytime.

Thus, in general, RK45 balances accuracy and cost-effectiveness, though it may struggle to solve stiff systems effectively. However, the simplicity of integration and the familiarity that we have with the method play a major role in the selection of this method.

B. Mathematical Derivation, Error, and Stability Properties

Initially, the Runge-Kutta family of methods considers the initial value problem:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

An explicit Runge-Kutta method with s stages has the form:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

where each stage k_i is defined by:

$$k_i = f\left(t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j\right)$$

The coefficients a_{ij}, b_i, c_i are specific for each method and are often organized into a *Butcher tableau*. The RK45 method uses the following coefficients (Fehlberg's original version):

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
4th-order (b_i)	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
5th-order (b_i^*)	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

The values on the left are the c_i values; the values in the middle are the a_{ij} values; and, the values in the bottom are the b_i values. Therefore, the k_i stages are computed using the a_{ij} and c_i values,

$$\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f\left(t_n + \tfrac{1}{4}h,\ y_n + \tfrac{1}{4}hk_1\right) \\
k_3 &= f\left(t_n + \tfrac{3}{8}h,\ y_n + \tfrac{3}{32}hk_1 + \tfrac{9}{32}hk_2\right) \\
k_4 &= f\left(t_n + \tfrac{12}{13}h,\ y_n + \tfrac{1932}{2197}hk_1 - \tfrac{7200}{2197}hk_2 + \tfrac{7296}{2197}hk_3\right) \\
k_5 &= f\left(t_n + h,\ y_n + \tfrac{439}{216}hk_1 - 8hk_2 + \tfrac{3680}{513}hk_3 - \tfrac{845}{4104}hk_4\right) \\
k_6 &= f\left(t_n + \tfrac{1}{2}h,\ y_n - \tfrac{8}{27}hk_1 + 2hk_2 - \tfrac{3544}{2565}hk_3 + \tfrac{1859}{4104}hk_4 - \tfrac{11}{40}hk_5\right)
\end{aligned}$$

and the two approximations (4^{th} and 5^{th} order) are:

$$y_{n+1}^{[4]} = y_n + h\left(\tfrac{25}{216}k_1 + 0 \cdot k_2 + \tfrac{1408}{2565}k_3 + \tfrac{2197}{4104}k_4 - \tfrac{1}{5}k_5 + 0 \cdot k_6\right)$$

$$y_{n+1}^{[5]} = y_n + h\left(\tfrac{16}{135}k_1 + 0 \cdot k_2 + \tfrac{6656}{12825}k_3 + \tfrac{28561}{56430}k_4 - \tfrac{9}{50}k_5 + \tfrac{2}{55}k_6\right)$$

The local truncation error can be estimated by:

$$\text{Error} = \|y_{n+1}^{[5]} - y_{n+1}^{[4]}\|$$

To maintain a desired tolerance ε of error, which is a user input that was found through trial and error. The step size h is adjusted using:

$$h_{\text{new}} = h\dot{(}\frac{\varepsilon}{\text{Error}})^{1/5}$$

Where h is a user input and was adjusted through trial and error for better method performance. The exponent $1/5$ comes from the order of the method (local error is $O(h^5)$, so the error behaves like h^5).

1. Derivation of Local Trunctation Error

An important concept underlying RK45 is the use of both the RK4 and RK5 method to create an adaptive time-step. The method as a whole allows for a level of control over the error, so it's difficult to directly quantify the error of the method. However, the solution is specifically advanced by the R45 method, and so the truncation error of that method can be derived.

Starting from the initial-value problem:

$$y'(t) = f(t, y(t)), \quad y(t_n) = y_n. \tag{4}$$

The exact solution at time $t_{n+1} = t_n + h$ can be written as a Taylor series about t_n:

$$y(t_n + h) = y(t_n) + h\, y'(t_n) + \frac{h^2}{2}\, y''(t_n) + \frac{h^3}{6}\, y'''(t_n) + \frac{h^4}{24}\, y^{(4)}(t_n) + O(h^5). \tag{5}$$

Applying the chain rule and above equalities allows us to express the first and second derivatives of $y'(t)$:

$$\begin{aligned} y'(t_n) &= f\big(t_n, y(t_n)\big) \\ y''(t_n) &= f_t\big(t_n, y_n\big) + f_y\big(t_n, y_n\big)\, f\big(t_n, y_n\big) \\ y'''(t_n) &= f_{tt}(t_n, y_n) + 2\, f_{ty}(t_n, y_n)\, f(t_n, y_n) + f_{yy}(t_n, y_n)\, f(t_n, y_n)^2 \\ y''''(t_n) &= f_{ttt}(t_n, y_n) + 3\, f_{tty}(t_n, y_n)\, f(t_n, y_n) + 3\, f_{tyy}(t_n, y_n)\, f(t_n, y_n)^2 + f_{yyy}(t_n, y_n)\, f(t_n, y_n)^3 \end{aligned} \tag{6}$$

where $f_t = \frac{\partial f}{\partial t}$ and $f_y = \frac{\partial f}{\partial y}$ and so forth.

The next step similarly would be to Taylor expand the RK4 formula:

$$y_{n+1} = y_n + \frac{h}{6}\Big(k_1 + 2k_2 + 2k_3 + k_4\Big) \tag{7}$$

with the coefficients k_n defined as

$$\begin{aligned} k_1 &= f\big(t_n, y_n\big) \\ k_2 &= f\Big(t_n + \frac{h}{2},\ y_n + \frac{h}{2} k_1\Big) \\ k_3 &= f\Big(t_n + \frac{h}{2},\ y_n + \frac{h}{2} k_2\Big) \\ k_4 &= f\big(t_n + h,\ y_n + h\, k_3\big) \end{aligned} \tag{8}$$

Each k_n needs to be expanded around the point (t_n, y_n):

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\Big(t_n + \frac{h}{2},\ y_n + \frac{h}{2} f(t_n, y_n)\Big) \\ &\approx f(t_n, y_n) + \frac{h}{2}\, f_t(t_n, y_n) + \frac{h}{2}\, f_y(t_n, y_n)\, f(t_n, y_n) \\ &\quad + \frac{h^2}{8}\Big[f_{tt}(t_n, y_n) + 2\, f_{ty}(t_n, y_n)\, f(t_n, y_n) + f_{yy}(t_n, y_n)\, f(t_n, y_n)^2\Big] + O(h^3) \\ k_3 &\approx f(t_n, y_n) + \frac{h}{2}\Big(f_t + f_y f\Big) + \frac{h^2}{8}\Big[f_{tt} + 2\, f_{ty}\, f + f_{yy}\, f^2\Big] + O(h^3) \\ k_4 &= f\Big(t_n + h,\ y_n + h\, f(t_n, y_n) + O(h^2)\Big) \\ &\approx f(t_n, y_n) + h\, f_t(t_n, y_n) + h\, f_y(t_n, y_n)\, f(t_n, y_n) \\ &\quad + \frac{h^2}{2}\Big[f_{tt}(t_n, y_n) + 2\, f_{ty}(t_n, y_n)\, f(t_n, y_n) + f_{yy}(t_n, y_n)\, f(t_n, y_n)^2\Big] + O(h^3) \end{aligned} \tag{9}$$

It should be noted that k_3 was evaluated at $(t_n + h,\ y_n + h\, k_3)$.

Substituting the expansions back in the the RK4 formula, grouping by order of terms, and simplifying leaves us with:

$$
\begin{aligned}
y_{n+1} &= y_n + h\, f(t_n, y_n) \\
&+ \frac{h^2}{2}\Big[f_t(t_n, y_n) + f_y(t_n, y_n)\, f(t_n, y_n)\Big] \\
&+ \frac{h^3}{6}\Big[f_{tt}(t_n, y_n) + 2\, f_{ty}(t_n, y_n)\, f(t_n, y_n) + f_{yy}(t_n, y_n)\, f(t_n, y_n)^2\Big] \\
&+ \frac{h^4}{24}\Big[f_{ttt}(t_n, y_n) + 3\, f_{tty}(t_n, y_n)\, f(t_n, y_n) + 3\, f_{tyy}(t_n, y_n)\, f(t_n, y_n)^2 + f_{yyy}(t_n, y_n)\, f(t_n, y_n)^3\Big] \\
&+ O(h^5).
\end{aligned} \tag{10}
$$

This matches the exact solution up to the $O(h^4)$ term.

From here, the definition of local truncation error (τ_{n+1}) is useful:

$$
\tau_{n+1} = \frac{1}{h}\Big(y(t_n + h) - y_{n+1}\Big) \tag{11}
$$

By substituting in both Taylor series expansions and canceling like terms, we are first left with terms on the order of $O(h^5)$. This is the local truncation error of RK4, and so the global truncation error will be $O(h^4)$.

The derivation of the truncation error follows the exact same procedure, just with the extra higher order term. Since this is a well studied method, the derivation won't be repeated. We will note that for RK5, the local truncation error is $O(6)$, and the global error is $O(5)$.

As mentioned above, RK45 is an explicit method which are typically not the most effective for stiff problems. Fig. 1 demonstrates the individual stability of the embedded methods, which is a relatively small range. And as mentioned above, by starting with a step-size within the stability region and using the adaptive step-size, the method should tend to remain in the region of stability.

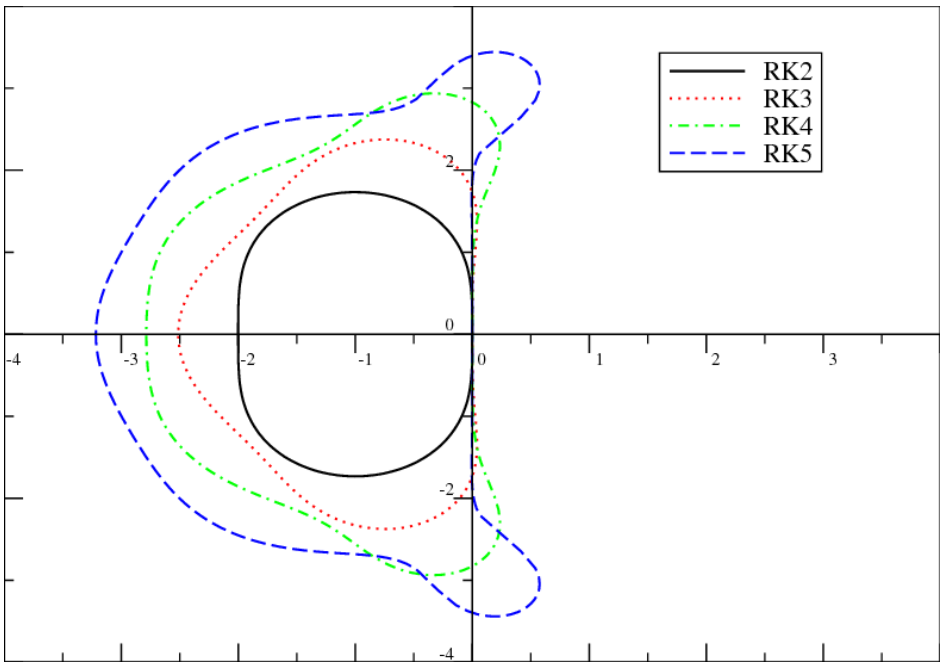

Fig. 1 Stability Plots of the Runge-Kutta Family

C. Algorithmic Summary

This method finds the solution for each time step by finding a weighted average of the slopes for the ODE. This is depicted in the first equation:

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

Where y_n is the "actual" value of the ODE. For the first step, this "actual" value is the boundary condition y_0, and for subsequent steps, it is the previously derived step. Then, the second term is the sum of the different slopes at the point being evaluated. These slopes can be obtained through

$$k_i = f\left(t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j\right)$$

The coefficients a_{ij}, b_i, c_i are the same for each method, and are given by the *Butcher tableau*. The following is a great analogy for what this method does. Imagine you are driving a car in a foggy day. The only information you have is your current position on the road, and the curvature of the road. Based on the direction the road seems to be leaving, you know somewhat in which direction to steer the vehicle. y_n is your current location, while $\sum_{i=1}^{s} b_i k_i$ is the curvature of the road.

Following this analogy, the time step can be interpreted as the speed of the vehicle. the time step is adjusted based on the error of RK4 and RK5. If the vehicle is moving too fast, then you may not be able to react to sudden changes in the curvature of the road. That is if the error is too big, then the method's prediction will be inaccurate. To correct this, the method adjusts the current time step until the error is within tolerance. That is, the speed of the vehicle is adjusted, so that the car is able to stay on the road.

The method repeats this process for multiple points along each time step until a solution is reached.

D. Checking Implementation

In order to check proper implementation of the Runge-Kutta methods, we must study the error convergence, showing that the method scales at the expected convergence rate. We must also evaluate which simulation parameters the method needs so we can accurately study the problem.

1. Convergence

To check the accuracy of our implementation, we will work with the two-body model. The two-body model has an analytical solution, and so we can check the accuracy by comparing the numerical two-body solution with the analytical two-body solution, which provides precise error measurements of the method. We can define error for this case as the relative error between the analytical and numerical solution:

$$\text{Error} = \frac{\left\| y_{\text{numerical}} - y_{\text{analytical}} \right\|}{\left\| y_{\text{analytical}} \right\|}$$

By plotting this error value against a range of time-steps, we can visualize how order of converges of the method. However, since RK45 relies on a variable time step, it is difficult to obtain the order of converges for this whole method. However, we argue that if we can demonstrate that if the two embedded models (RK4 and RK5) separately have the correct error convergence rate, then it is fair to claim that our RK45 implementation will therefore have an appropriate convergence rate.

For each of the methods, RK4, RK5, we can plot the error against varying timesteps, showing that the error converges at an expected rate. On logarithmic graph, where both axes are scaled logarithmically, the error of the RK4 and RK5 approximations should resemble a linear graph with a slope of 4 and 5, corresponding to the respective orders of the methods.

As can be seen in Fig. 2, both methods have approximately correct slopes, and therefore can be considered to be properly implemented. We attribute variations from the expected values to rounding error.

If we were to plot the error of the RK45 function against its initial timestep, we would see a flat line, as changing the initial timestep has almost no impact on how the RK45 function iterates. It uses an adaptive timestep, as explained above. Thus, the initial does not effect the error. However, the RK45 function is derived from both the RK4 and RK5

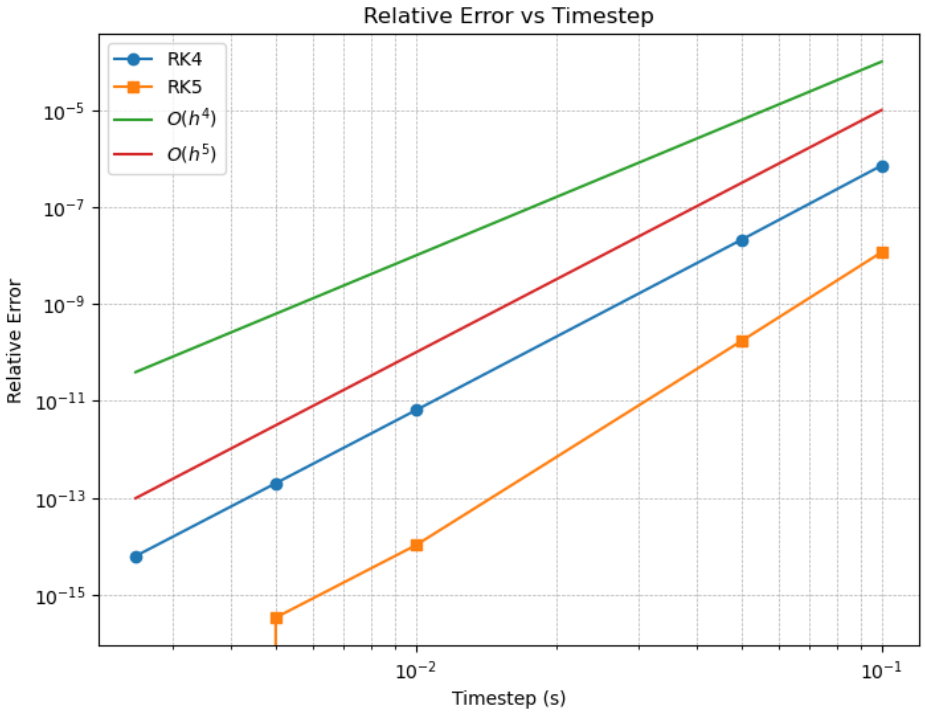

Fig. 2 Graph of Relative Error Across Varying Timesteps

functions. This proves that the RK4, RK5, and RK45 methods are correctly implemented, as the error converges at the rate predicted by the order of the method.

2. Initial Method Parameters

The initialization of RK45 requires two main parameters: the initial timestep, and an error tolerance which imposes a control upon the method's error. Our initial value of the timestep is set to $1 \cdot 10^{-8} seconds$, and this was chosen primarily as the adaptive time step devalues the importance of choosing the initial value on computational time, but starting with a small time step ensures that the method starts and begins in stable regions, which is useful for stiff problems. The tolerance was chosen through iteration; through trial and error, we selected a value of $1 \cdot 10^{-8}$ as that seemed to best balance accuracy and computational time.

V. Results

Raw results that plot the trajectory of your state variables as a function of time are rarely the most direct way to address the specific questions we posed. In order to properly convey that the numerical method works, we will be displaying a variety of convergence error graphs, trajectory plots, and eccentricity plots.

1. Error Definition

Our two goals with this study are to determine an orbit radius at which the mass and gravitational pull of the Earth start to have a meaningful effect on the orbit of the satellite and to determine how the orbit perturbations affect subsequent orbits. Thus, we must choose a metric to quantify how the orbit changes due to the extra gravitational pull of the Earth. To do this, we compare the RK45 approximation of a three-body-problem and the analytical solution of the two-body-problem. Given that the timesteps will not necessarily match between the two models, directly comparing position at each timestep will be difficult. Instead, we can look at another parameter of the orbit to compare the models, such as eccentricity. Eccentricity measures how elliptical the orbit is. Our two-body solution assumes a circular orbit, with an eccentricity of zero. Thus, any perturbation will result in an elongation of the orbit, increasing the eccentricity. This definition of error allows for a simple, normalized, and physically meaningful quantifiable value of error.

Eccentricity was quantified via the following values: r_p, which is the distance from the moon to the closest part of the satellite orbit, and r_a, which is the distance from moon to the furthest part of the satelite orbit. It is all calculated according the following equation:

$$e = \frac{r_a - r_p}{r_a + r_p} \tag{12}$$

A. Distance

We determined that as the altitude above the moon's surface of the satellite increases, the Earth's gravity has a much more significant impact on the satellite's trajectory. In order to model this, we calculated the position data using RK45 of five different altitudes: 100 km, 500 km, 1000 km, 2500 km, and 5000 km. The position data was modeled for 24 hours. After obtaining those values, we found the eccentricity of the first orbit for each altitude, then plotted eccentricity as a function of altitude.

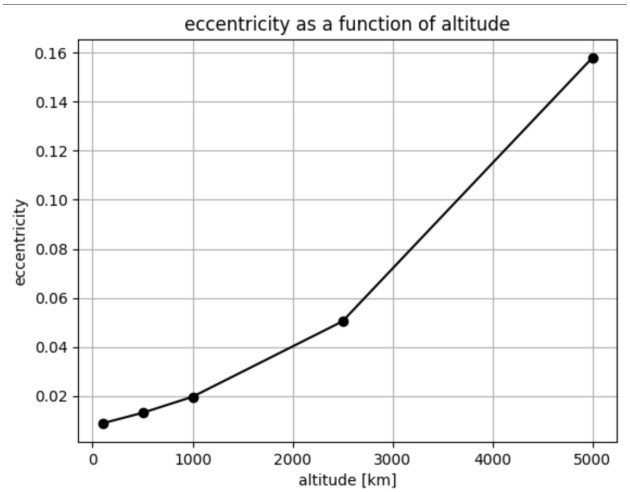

Fig. 3 Eccentricity: Multiple Orbits

As observed from the data, the eccentricity strongly increases as the altitude above the moon of the satellite increases. As the altitude increases, the satellite's orbit deviates from a circular orbit, transitioning from an orbit to a trajectory. The following trajectory plot describes the orbit of a satellite at a 5000 km altitude over the course of 15 days. We can see from the plot that the orbit's eccentricity increases dramatically. For the sum total, we conclude that the perturbation begin having a significant effect at an altitude of about 1000 kilometers, though it's effects become non-negligent at at around 500 kilometers.

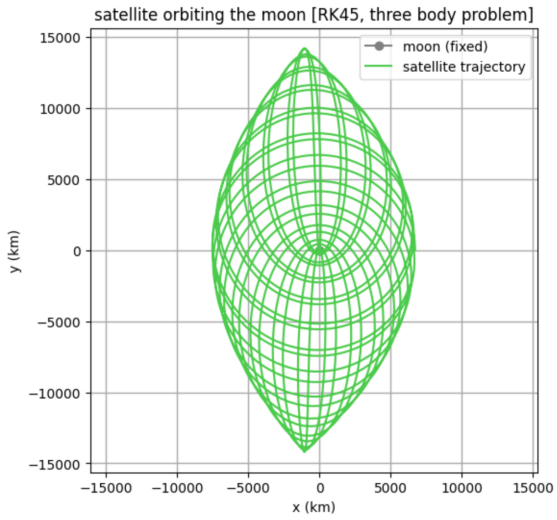

Fig. 4 Trajectory Plot [5000 km, 15 days]

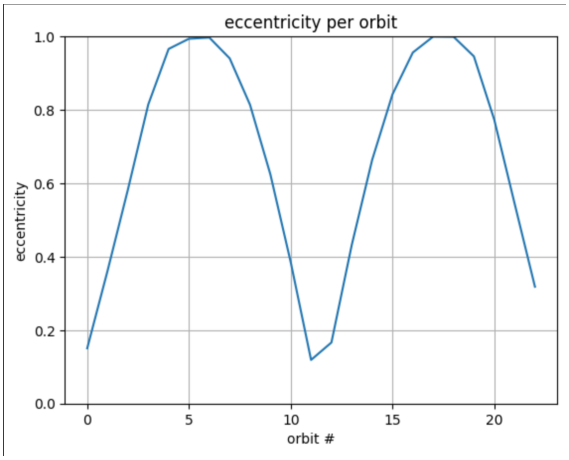

Fig. 5 Eccentricity Plot [5000 km, 15 days]

B. Time Variation of Eccentricity

The second question that was posed for this model was centered around the change of eccentricity over time, given different initial orbit altitudes. By determining the initial conditions that lead to faster increases in eccentricity, long-term orbital stability vs altitude can be shown. To plot this, we calculated the eccentricity of each orbit of the satellite for a total time of 5 days. Then the eccentricity was graphed vs orbit number, with the slope representing the change in eccentricity per orbit as shown in Fig.7. This was done for 3 different initial orbit altitudes of 100km, 250km, and 500km as shown in Fig.6.

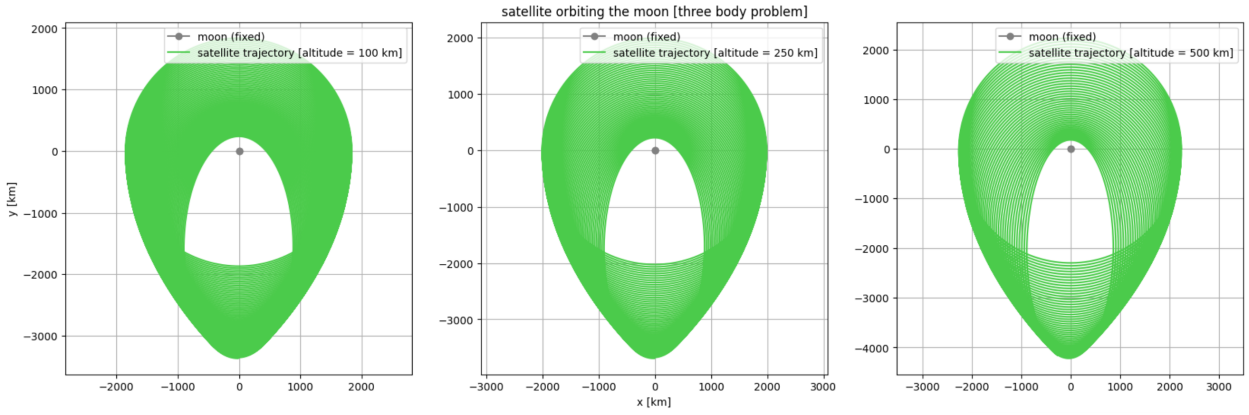

Fig. 6 Visualized Orbits for Which Eccentricity Was Found

From Fig.6, we can see that the eccentricity of the final orbit, which is the centermost, is most elliptical for the 500km initial orbit, and least elliptical for the 100km initial orbit. This suggests that the higher the lunar orbit is, the faster it becomes elliptical, and the less stable it is for the same amount of time.

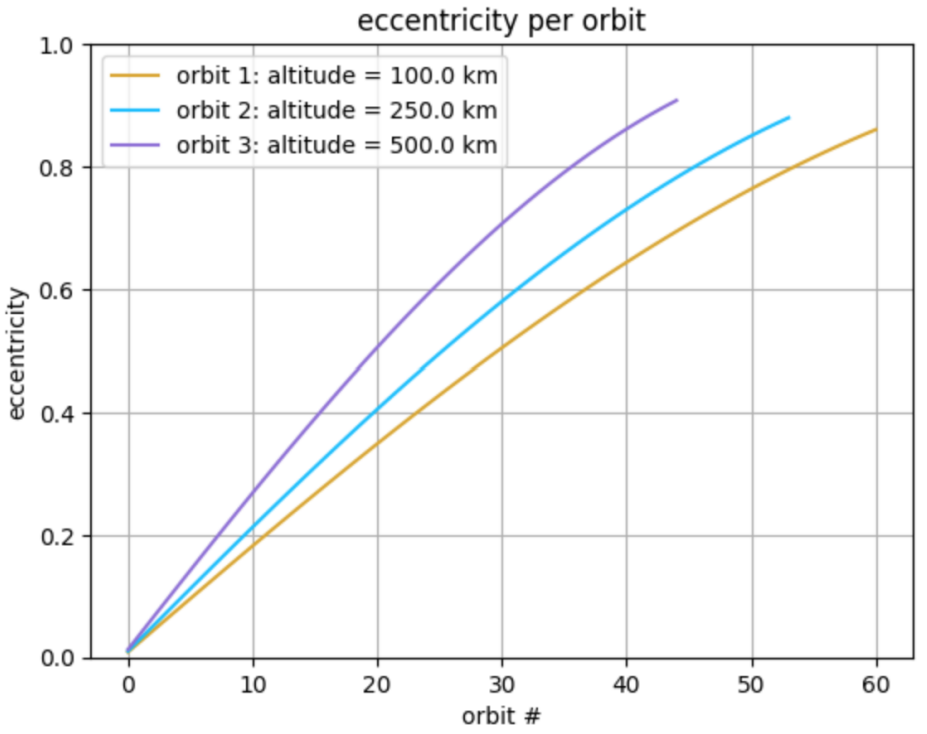

Fig. 7 Eccentricity of 3 Different Altitude Orbits vs Orbit Number

This trend is further illustrated when the eccentricity vs orbit number is graphed for the three different initial orbits. As shown in Fig.7, the 500km orbit has the highest slope on the graph and makes the closest approach to an eccentricity of 1, showing that it is the least stable. Note that the lines end at different points on the x-axis because the lines are plotted over the same time duration of 5 days. This further illustrates that a higher-altitude orbit has a higher change in eccentricity for both the same amount of orbits(as illustrated by the slope) and the same amount of time(as illustrated by the final plotted y-value) as a lower-altitude orbit.

From this information, we can conclude that the lower the altitude of the lunar orbit, the longer a satellite can remain without significant deviation from its original elliptical orbit. However, based on our model, that time, even for an orbit as low as 100km is relatively short, as it quickly becomes eccentric. This means that satellites in lunar orbit need a capable propulsion system for orbit maintenance, as the influence of Earth's gravity can significantly deform a low lunar orbit in the order of hours. It also shows that for staging a lunar landing, the lower the lunar orbit the spacecraft enters, the less correction is needed to maintain the orbit before landing, which can lead to more efficient use of propellant.

VI. Project Roles

This project was divided in as close to even parts as possible. Responsibilities were distributed taking expertise and area of interest into consideration. Those with a deeper understanding of orbital mechanics worked on the derivation of equations of motion, and pieces of code; whereas, those interested in learning more in depth about numerical methods, researched, derived, and implemented the method. Though it should be mentioned that there was a great deal of mutual support for all the roles.

We chose our dynamical system together, as well as the corresponding questions that we wanted to answer.

Santiago worked on the most math intensive part of the project. He worked researching the strengths and weaknesses of the method, its derivation, and special characteristics. Furthermore, he dove deep into the understanding of why the method works the way it does, and what each of the "stages" of the method do. Ultimately, he wrote the numerical method section.

Pranet operated as the general leader of the group. He developed the initial idea for the chosen dynamical system and the questions, conducted the derivation of the error of RK45, and wrote the justification for the method, including the discussion on stability. He heavily contributed to all components, including debugging, defining error definition,

and contributed writing to every section in this report. For the group, he helped schedule meetings, delegate work, structured report, and set-up the GitHub repository for the code.

Colin and Dhruva worked together on the most code intensive part of the project to help distribute the load. They used python to model the 2-body and 3-body systems using RK45, RK4, and RK5 to plot the trajectory of the spacecraft over time, and determine orbital eccentricity. It included extensive coding and troubleshooting, as well as analysis of the RK45 method. Dhruva also worked to write the code to calculate the individual orbits, and differentiate them from the plethora of position data. Colin worked to graph eccentricity plots coherently while finding appropriate initial conditions to address the questions posed for this project. He also was the primary author of the Time variation of Eccentricity portion of the results and contributed along with Pranet to the introduction.

Ameya worked on the implementation of the method and creating error convergence plots. He wrote Python code to plot the how the error of the RK4, RK5, and RK45 converges, in order to prove that the numerical methods were implemented correctly. He also helped troubleshoot the implementation of the RK4 and RK5 methods when they did not converge accurately. Additionally, he wrote out the Checking Implementation portion of the report, including the Convergence section and the error definition in the Results section which defines the system parameters.

VII. Conclusion

Through the implementation of the Runge-Kutta-Fehlberg (RK45) method, this study has provided insight into the boundary at which the Earth's gravitational influence on a lunar orbiter becomes non-negligible. By simulating a range of circular orbits at varying altitudes around the Moon and introducing the perturbing force of Earth's gravity, we have demonstrated how the classical two-body assumption breaks down beyond a certain orbital radius.

Our results show that at lower altitudes, the two-body approximation remains accurate and stable for a a few hours, but is inevitably warped by Earth's gravity. However, as the orbital radius increases, the effects of Earth's gravity lead to noticeable deviations in the orbiter's trajectory, primarily seen through changes in orbital eccentricity. These perturbations indicate that a three-body model is essential for high-altitude and/or lunar orbits longer than a few hours.

This analysis has important implications for mission planning and satellite stability in cislunar space. It helps define the safe operational altitudes for satellites that can rely on simpler models, as well as the regimes where adaptive, numerically-integrated three-body simulations are essential for maintaining orbital integrity over time. Furthermore, this study suggests that any lunar orbiter should be equipped with a propulsion system that is capable of repairing the orbit when it decays.

An important detail that was overlooked when selecting a numerical method was the stiffness of the system. The presence of more than one body in the system makes the system stiff, as the influence by the gravity of the moon is way greater than the influence of the gravity of the Earth. As it was noted earlier in the study, RK45 is not well suited for stiff systems. This issue may have been the reason of difficulties encountered during the implementation check process. In the future, a different method better suited for stiff systems should be used. Such a method like Implicit Runge-Kutta Method can handle stiff systems while retaining accuracy.

Appendix

Here we link to our Github repository containing our code: `https://github.com/pranetp/AE370-Proj1-Earth-Moon-and-Fire`

Acknowledgments

ChatGPT was used by some members to generate and debug code. It was also used to supplement our understanding of the method, but conceptual understanding was best found through conversations with our more experience peers. It was used for some derivations and to convert mathematical notation into Latex, but the vast majority of writing was done by ourselves.