The Restricted Three Body Problem
Trans-Lunar Injection

Tyler Baines, YuanJea Hew, Shin Toyama
University of Arizona

To model the motion of spacecraft that travels from the Earth to the Moon, we need to take the dynamics into the context of the Circular Restricted Three Body Problem. Fitting the system into a rotating frame of reference, we investigate one of the methods to set spacecraft into the lunar trajectory, the Trans-Lunar Injection. Using numerical methods, we solve the equation of motion that is derived from the Newton’s Second Law in a rotating frame of reference in the Earth-Moon system. Through the numerical analysis, we successfully made plots for the motion of spacecraft and found out that the path can diverge and take different paths.

1. Introduction

After the World War II, human beings have been investigating the outer space; especially, the period of 1957-1969 is known well as the ‘Space Race’. In this period, Soviet Union and United States promoted the space exploration, and in competition they significantly developed the exploration of the Moon and sent hundreds of spacecraft to the Moon\(^1\). There were two main ways to get spacecraft injected to the Moon’s orbit. One was by direct ascent into the lunar trajectory. Another was by the trans-lunar injection\(^2\). The trans-lunar injection is a method to set a spacecraft on a trajectory with propulsion systems that will cause it to arrive at the Moon, and is only used in small systems like the Earth-Moon system. The trans-lunar injection was successfully performed by the Soviet Union’s Luna 4 on Apr. 2\(^{nd}\), 1963 for the first time, and after that it was used multiple times\(^3\). After the Moon landing in 1972, in fact, much less work has been done of the exploration of the Moon, but the trans-lunar injection is still a useful method to send spacecraft to the Moon.

When we consider the orbit of a spacecraft it is true that spacecraft’s orbits are affected by many bodies (Sun, Jupiter, the other planets, asteroids and so on), but the trans-lunar injection is approximately dominated by the gravitations from the Earth and Moon because it is in the small system. Since the mass of the spacecraft is negligible compared to the Earth and Moon, the trans-lunar trajectory can be modelled as a ‘circular restricted three-body problem’.

Heinrich Bruns and Henri Poincaré, in 1887, proved that general analytical solution of the Three Body Problem does not exist by any means, i.e. no algebraic expressions and integrals exist. Which leaves one to solve this problem using numerical methods. In order to do so we to set up our system in a simple as possible to make the numerical calculations as easy as possible. In our case, we examine the
Restricted Three Body Problem which is that one of the masses in our system is negligible reducing this problem in to a simple Two Body problem where the small mass is effected by the larger two. In doing so will allow us to calculate the trajectory of the “massless” particle which rely on some specific initial conditions.

Circular restricted three-body problem (CRTBP) is known as one of the cases in which the motion of the third body in a system is solved numerically. If the third body is negligible in comparison, and the other two bodies remain a same distance while they rotate, then the problem is called CRTBP. The binary system has the barycenter that is the center of mass of the Earth-Moon system. About the barycenter, the Earth and Moon rotate at the same rate of rotation (same $\omega$). Then, the motion of the spacecraft can be designed in a rotation frame of reference. If an inertial frame of reference is used, then we will have to account for the motions of all three bodies, but using a rotating frame of reference that is centered at the barycenter binary system, then the co-orbiting bodies will be fixed, while the third body is free to move about the system.

2. Assumptions

We investigate the motion of a spacecraft in the Earth-Moon system using a rotating frame of reference and examine the coplanar circular three body problem (xy-plane), thus $z$ will be set to zero. All bodies in our rotating frame are essentially point masses.

We are assuming that the gravity from the Earth and Moon is the only force acting after spacecraft get some velocity to escape from the Earth’s orbit. In fact, all other bodies in space will have an effect on the spacecraft, but the effect is small and ignorable. The Newton’s Law of gravitation explains this:

$$g = \frac{G M}{d^2}$$

g is gravitational acceleration on spacecraft by a body with a mass M, G is gravitational constant, and d is the distance between an object and the body. According to this equation, the farther the body M is, the smaller the gravitational acceleration by the body M will be; thus, the effect from other bodies (even from the planets in the solar-system) is ignorable. For the Sun that is far from the Earth’s orbit but super massive, the gravitational acceleration is about 0.00593m/s^2 when spacecraft is around the Earth: this value is much bigger than that from the other planets in the solar-system, but this is still ignorable. When spacecraft is orbiting around the Earth, the gravitational acceleration by the Earth is about 0.309m/s^2. When spacecraft is orbiting around the Moon, the gravitational acceleration is about 0.420m/s^2. In comparison the effect by the Sun is really small. Therefore, we can assume that the spacecraft we are considering is dominated by the gravity only from the Earth and Moon: we can use newton laws to describe the motion of spacecraft since gravity is the only force acting on.

3. Numerical Method for analysis
The Runge-Kutta method is formulated by 20th century German mathematicians, Carl Runge and Martin Kutta. The method is used in computational physics to perform numerical analysis with iterative methods. There are numerous kinds of Runge-Kutta techniques, however for the problem we are tackling, the Runge-Kutta 4th Order is selected to provide numerical solutions.

The fourth order Runge-Kutta method is used to solve Ordinary Differential Equations (ODE). The formula for the Runge-Kutta 4th Order is

\[
\begin{align*}
  k_1 &= dt \cdot g[t_i, f_i] \\
  k_2 &= dt \cdot g[t_i + \frac{dt}{2}, f_i + \frac{k_1}{2}] \\
  k_3 &= dt \cdot g[t_i + \frac{dt}{2}, f_i + \frac{k_2}{2}] \\
  k_4 &= dt \cdot g[t_i + dt, f_i + k_3] \\
  f_i &= f_i + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}
\end{align*}
\]

The limiting factor for this approach is that it only solves first order ODE's. Nevertheless, the superiority of the technique is because it completes 4 evaluations per step on the Right-Hand Side (RHS) of the ODE. From the set of equations above, \( k_1 \) is evaluated at the initial point while both \( k_2 \) and \( k_3 \) are half steps which evaluate on the midpoints and \( k_4 \) is assessed on the trial final point.

The Runge-Kutta 4th order method is much more accurate compared to its counterpart, Euler Method. This can be explained through the method’s first order precision, by only taking the derivative on the initial point in each step while providing only an estimation of the next function value.

4. General Formulation

We investigate the motion of a spacecraft in the Earth-Moon system using a rotating frame of reference and examine the coplanar circular three body problem (xy-plane), thus \( z \) will be set to zero. All bodies in our rotating frame are essentially point masses and on forces acting on the spacecraft are gravity. From here on we will regard our spacecraft as a point particle.

Since gravity is the only force acting on the we can use newton laws to do describe the motion of the particle. The equation for Newton’s Second Law in a rotating frame along the z-axis is given to be the following:

\[
\begin{align*}
  a_{rot} &= a_{linear} - 2\vec{w} \times \vec{v}_{rot} - \vec{w} \times (\vec{w} \times \vec{r}) \\
  \left(\frac{\partial \vec{v}}{\partial t}\right)_{rot} &= \left(\frac{\partial \vec{v}}{\partial t}\right)_{linear} - 2(\vec{w} \times \vec{v}_{rot}) - \vec{w} \times (\vec{w} \times \vec{r})
\end{align*}
\]

where \( a_{rot} \) and \( \left(\frac{\partial \vec{v}}{\partial t}\right)_{rot} \) is the acceleration in the rotating frame, \( a_{linear} \), and \( \left(\frac{\partial \vec{v}}{\partial t}\right)_{linear} \) is the gravitational acceleration in the linear (non-rotating) frame, Coriolis force is denoted as \( [- 2\vec{w} \times \vec{v}_{rot}] \), and the Centrifugal force is denoted as \( [- \vec{w} \times (\vec{w} \times \vec{r})] \).

The xy-plane is selected to be the perspective of observation, therefore, \( z = 0 \). Thus, eqn. (2) can be separated into their respective components, we find the following:
\[ \mathbf{\ddot{w}} \times \mathbf{\dot{v}}_{\text{rot}} = w(v_x y - v_y x) \quad (3) \]

\[ \mathbf{\ddot{w}} \times (\mathbf{\dot{w}} \times \mathbf{\hat{r}}) = -w^2(x\mathbf{\hat{x}} + y\mathbf{\hat{y}}) \quad (4) \]

\[ \left( \frac{d\mathbf{\ddot{w}}}{dt} \right)_{\text{rot}} = \left( \frac{d\mathbf{v}_x}{dt} \right)_{\text{rot}} + \left( \frac{d\mathbf{v}_y}{dt} \right)_{\text{rot}} \quad (5) \]

\[ \left( \frac{d\mathbf{\ddot{w}}}{dt} \right)_{\text{linear}} = -GM_1 \frac{[(x-x_1)x + (y-y_1)y]}{[(x-x_1)^2 + (y-y_1)^2]^2} - GM_2 \frac{[(x-x_2)x + (y-y_2)y]}{[(x-x_2)^2 + (y-y_2)^2]^2} \quad (6) \]

Thus, eqn. (3), (4), and (6) can be substituted into eqn. (2) to be rewritten in each component as:

\[ \left( \frac{d\mathbf{v}_x}{dt} \right)_{\text{rot}} = \frac{-GM_1(x-x_1)}{[(x-x_1)^2 + (y-y_1)^2]^2} \frac{x}{3} - \frac{-GM_1(x-x_2)}{[(x-x_2)^2 + (y-y_2)^2]^2} \frac{x}{3} + 2wv_y + w^2x \quad (7) \]

\[ \left( \frac{d\mathbf{v}_y}{dt} \right)_{\text{rot}} = \frac{-GM_1(y-y_1)}{[(x-x_1)^2 + (y-y_1)^2]^2} \frac{y}{3} - \frac{-GM_1(y-y_2)}{[(x-x_2)^2 + (y-y_2)^2]^2} \frac{y}{3} + 2wv_x + w^2y \quad (8) \]

where

\[ \omega = \frac{GM}{a^3} \quad \text{(by Kepler’s 3rd Law)} \quad (9) \]

Since we’re using a rotating reference frame, the center of mass of the system should be placed at the origin and both bodies fixed along the x-axis. This implements a shift of positions in our equations and result in the following:

\[ x_1 \rightarrow -x_1 \]

\[ y_1 = y_2 = 0 \]

\[ \dot{x} = \left( \frac{d\mathbf{v}_x}{dt} \right)_{\text{rot}} = \frac{-GM_1(x-x_1)}{[(x-x_1)^2 + (y)^2]^2} \frac{x}{3} - \frac{-GM_2(x-x_2)}{[(x-x_2)^2 + (y)^2]^2} \frac{x}{3} + 2wv_y + w^2x \quad (10) \]

\[ \dot{y} = \left( \frac{d\mathbf{v}_y}{dt} \right)_{\text{rot}} = \frac{-GM_1(y)}{[(x-x_1)^2 + (y)^2]^2} \frac{y}{3} - \frac{-GM_2(y)}{[(x-x_2)^2 + (y)^2]^2} \frac{y}{3} + 2wv_x + w^2y \quad (11) \]

\[ R_1 = \sqrt{(x-x_1)^2 + (y)^2} \quad (12) \]

\[ R_2 = \sqrt{(x-x_2)^2 + (y)^2} \quad (13) \]

and let \( R_1 \) and \( R_2 \) be distance to the particle (spacecraft) from \( M_1 \) and \( M_2 \) where \( M_1 > M_2 \).

Using the above equations may be quite difficult in terms of making them non-dimensional just by looking at the possible parameters in the system. Thus, we will take a different approach and relate variables based on the geometry of having both bodies orbiting a common center of mass but are fixed in a rotating frame along the x-axis referencing the derivation given by James[7] and the Circular Restricted Three Body Handout provide by the University of Colorado Boulder[8]. With the barycenter located at the origin and we aim to provide a sense of unity, which will yield the follow equations starting with the center of mass:
\[ X_{cm} = -M_1x_1 + M_2x_2 = 0 \quad (11) \]

where we can solve to acquire
\[ x_1 = \frac{M_2}{M_1} x_2 \quad (12) \]

The distance between the two primary object is \( a_{12} = x_1 + x_2 \) and \( x_2 = a_{12} - x_1 \). Then substitute \( x_2 \) into equation (12) to find:
\[ x_1 = \frac{M_2}{M_1} (a_{12} - x_1) \quad (13) \]

and with some algebra
\[ x_1 = \frac{M_2}{M_1 + M_2} a_{12} \quad \text{and} \quad x_2 = (1 - \frac{M_2}{M_1 + M_2}) a_{12} \quad (14) \]

We will define the parameter \( \mu = \frac{M_2}{M_1 + M_2} \) and since \( M_1 > M_2 \), \( 0 < \mu \leq \frac{1}{2} \), and \( \mu \) describes both location and masses of the two bodies. Now we would like to set various parameters equal to non-dimensional value of one. This will provide unity for our problem. Such as the distance between co-orbiting bodies will be the unit distance (du), the total mass of the system (mu), and defining the orbital frequency to be 2π so that the orbiting bodies complete 1 full or in time (rad/tu).
\[ a_{12} = 1 \quad (15) \]
\[ M_1 + M_2 = 1 \quad (16) \]
\[ \omega = 1 \quad (17) \]

Since \( \omega = \left( \frac{G(M_1 + M_2)}{a^3} \right)^{\frac{1}{2}} \) we can substitute in our non-dimensional parameters and find that \( G = 1 \). And we can solve \( \mu \) in terms of \( M_1 \) and \( M_2 \) and find
\[ M_1 = 1 - \mu \quad (18) \]
\[ M_2 = \mu \quad (19) \]

Now we can substitute these equations 17, 18, 19 back into equations 8 and 9 we find the follow
\[ \ddot{x} = 2\dot{y} + x - \frac{(1-\mu)(x+\mu)}{R_1^2} - \mu \frac{x-1+\mu}{R_2^2} \quad (20) \]
\[ \ddot{y} = 2\dot{x} + y - (1-\mu) \frac{y}{R_1^2} - \mu \frac{y}{R_2^2} \quad (21) \]

and
\[ R_1^2 = (x + \mu)^2 + y^2 \quad (22) \]
\[ R_2^2 = (x - 1 + \mu)^2 + y^2 \quad (23) \]

Eqn (20) and (21) will be the non-dimensional equations use to run our numerical iterations. As mentions before there is no analytical solution to compare our numerical calculations to. We will choose our initial conditions where we can integrate backwards or forward if we choose to. But for our purpose we will integrate forward.
5. Numerical Analysis

In order to test how accurate our numerical method treated out system, the step sized for the integration needs to be optimized. One way to go about this is to loop through several step sizes and compare the numerical result to the analytical solution. In our case, not analytical solution can be compared to, thus we must resort to method. Observing trajectories of several step sizes allowed us to visually see changes in the path of the particle in motion. By randomly picking initial x and y values at some time \( t > 0 \) and integrate backwards in time, we can see how the particle is affect. Another is if initial conditions are known, then we can iterate forward in time and observes how the path change over different step sizes \( \Delta t \). We defined our step as

\[
\Delta t = \frac{b-a}{N}
\]

where is \( N \) the number slices and we chose out interval to be \([x_1, x_2]\) which results in the following

\[
\Delta t = \frac{x_2-x_1}{N} = \frac{(1-\mu)-(\mu)}{N} = \frac{1}{N}
\]

(Refer Initial Conditions section to view how we choose values to test.)

Figure 1 shows the different time steps trajectories from the Earth to Moon where we observe a Trans-Lunar Injection occur under the following conditions

\[
\mu = 0.01215453583
\]

\[
\theta_E = 226.81^\circ
\]

\[
R_{orb} = R_{Earth} \text{ (starting at the surface)}
\]

\[
\Delta V = 3.14
\]

\[
x = (t_o, x_o, y_o, v_x, v_y) = (0, -0.023498, -0.012038, 7.90601, -7.40949)
\]

These conditions were chosen in hopes to produce similar findings to No, Tae Soo, et al. (2012)\(^6\) and compare the two. We find that our trajectory maps a similar path that of No, Tae Soo, et al. while our \( \Delta V \) are different values. This is because we didn’t take in account of a controlled acceleration which would have been added term our EOM to allow the particle to orbit the Moon’s point source.

From these initial conditions we can see that the optimal step time should be \( 10^{-6} \leq \Delta t \leq 10^{-5} \). For our purpose we chose \( \Delta t = 10^{-6} \) to maximize convergence of particle path in all test trials looking for Trans-Lunar Injection conditions. The integration ran until \( x \leq 1 - \mu \) and \( y = 0 \) because this indicated the position of the moon in our dimensionless system which tells us that out spacecraft has reached the moon.
Figure 1: This plot shows the trajectory of the given x at various time steps. You can see as N increase our values begin to converge to their true values. You can see that values $N > 10000$ nearly overlap one another which suggest the optimal time step. The dot and circle correspond to the Earth and its ‘radius’ while the black dot is the moon and its ‘radius’.

6. Initial Conditions

Before running numerical integrations, we want to determine a set of conditions not randomly selection to be at any point in space. Initially we start at some point near Earth, either at the surface or in orbit. No, Tae Soo, et al. (2012)\cite{6} examine a circular parked orbit of a particle a distance $r_{orb}$ away from the Earth undergoing circular motion. They provide a series of equations for initial conditions for $x$ and $y$ components as well as their initial velocities with respect to proper components. These equations can be derived examining circular motion about a large body which is then shift to make the Earth-Moon system in the rotating frame. The equations are as followed

$$x_o = r_{orb} \cos(\theta_E) - \mu$$  \hspace{1cm} (23)

$$y_o = r_{orb}\sin(\theta_E)$$  \hspace{1cm} (24)

$$v_x = -(V_{orb} + \Delta V) \sin(\theta_E) + y_o$$  \hspace{1cm} (25)

$$v_y = (V_{orb} + \Delta V) \cos(\theta_E) - x_o$$  \hspace{1cm} (26)

where $V_{orb}$ is the particle’s speed in the orbit about the Earth and $r_{orb} = R_{Earth} + H$ and $R_{Earth}$ and $H$ are non-dimensional quantities of the Earth’s radius and some Height above the surface. These values are determined by the following relations.
\[ R_{\text{Earth}} = \frac{\text{True Radius of Earth}}{\text{Distance to Moon}} = \frac{6371 \text{ km}}{384400 \text{ km}} = 1.657388137 \times 10^{-2} \]
\[ H = \frac{\text{Distance about surface} = D}{\text{Distance to moon}}; \text{where } D \geq 0 \]

7. Results

Setting conditions \( \Delta V = 0 \) and \( H > 0 \) and observing one orbital period, \( P_{\text{orb}} \), we find the circular orbits about the Earth where

\[ P_{\text{orb}} = 2\pi \sqrt{\frac{a_{\text{orb}}}{1 - \mu}} \quad (27). \]

These trajectories suggest circular orbits are obtained that complete one cycle until at large distances, where we noticed gaps began to appear. These gaps arise due to the effect of the gravitation force due to the Moon becoming more influential on the particle. The Moon is essentially applying a Tidal Brake to the particle causing it slow down thus resulting in an increase in its orbit which relates back to an increase orbital period which is why these gaps occur (Moon’s gravitational pull). So the expected period is smaller than the actual period in the system. If we plot the calculated expected period and fit the data to some power law we find that fits data is similar.
to Kepler’s 3\textsuperscript{rd} law, which related period to distance where \( P \propto a^{1.5} \) where our fit contains about 0.633\% error.

\[
P_{\text{exp}} = 6.1444r^{1.4905}_{\text{orb}}
\]

\textbf{Testing various conditions to obtain a Trans-Lunar Injection}

Testing several initial conditions, we can identify a variety of trajectories. These initial conditions are very important because even slight changes can cause the outcomes to show diverging behaviour.

To get a particle from a point near the Earth’s location to the moon we must consider three important conditions, the ejection velocity (a kick to get the particle out it orbit/position), the angle of ejection (where the particle will be launched from), and the orbital height to be released from. Here we adopted the ejection angle 226.81\(^\circ\) and a \( \Delta V = 3.14 \) at a \( r_{\text{orb}} = R_{\text{Earth}} \), with \( H = 0 \). We find a direct Lunar Injection where our
trajectories match results from No, Tae Soo, et al. (2012). To test for divergence in our particles path we off set the initial conditions by ±0.01% and ±5%. Figure 4 and 5 show the divergence test at the same unit time scale $t = 0.55391200$.

Both plots show some similar traits but are very different. Small changes in the kick velocity do show that the particle paths diverge. Increasing or decreasing the velocity by 1% will cause the particle to fall back toward earth. While a decrease in 5% causes to part to mimic the 1% increase and an increase in 5% eventually throws the particle out of the system missing its target.

Any change in the ejection velocity leads to different trajectories, which supports that the initial conditions are crucial in this problem based off observations. And this also tells us that if the particle has too much velocity to start with, it will become practically unaffected by the large bodies and become unbound, escaping the system where as time unit increases the trajectory will become that of a spiral feature (See Figure 6).
figure 6). Since we observed a Direct Earth-Moon Transfer it would be appropriate to examine how the positions and velocity change over time (see figures 8 and 9).

It is clear that over time the particle have a greater change in $x$ direction than in the $y$ direction. This is because the object is launch near the Earth point mass with an velocity in the $x$-direction greater than the $y$-direction in magnitude. The particle is essentially hooked around and escapes the Earth. As the particle travels away from the Earth, the Earth’s gravity is still exerting a force on the particle which causes a decrease in the velocity over time. A unit times UT $\approx 0.0$ and $0.3$ the $y$-velocity goes to zero, and tell us the minimum and maximum of the positions of our particle in the $y$-direction. But at UT = 0.3 the Moon begins to gravitational attract the particle, pulling it towards it direction. As the particles moves further away from the Earth we expect the force exerted on the particle to decrease and the Moon to become more dominate. The sharp increases in velocity at the end of the plot are due to that our object are just point masses and don’t account for realistic situation. The sharp turn implies that the particle gets very close to the point mass, get a large kick and sent away. But if we were to plot where the radius of the moon would actually be in our dimensionsless coordinates, the trajectory intersects with the surface boundary of the Moon, (see figure 10).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{position_vs_time.png}
\caption{Position vs Time}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{velocity_vs_time.png}
\caption{Velocity vs Time}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{moon_radius.png}
\caption{Moon dimensionless radius enclosing the point mass.}
\end{figure}

8. Conclusion
We have examined the three-body problem or in our case the restricted three-body and plotted a course of our spacecraft (the particle) from the Earth to the Moon, i.e. the Trans-Lunar Injection, in a rotating coordinate system governed by equations 18 and 19. We tested small deviations in the kick velocity to determine how the trajectory is effected, and show that the particle path can diverge and take different paths. The particle will remain bound under Earth’s gravitational force if not given enough of a kick to escape from. To much cause the particle to escape and become unbound from the system. We compare our particle path to that of another group, for similar structure, to verify numerical calculations and equations acquired sufficient results. Future investigations can incorporate an extra acceleration that could put the particle into orbit about the moon and remained bound or adding an extra body to the system such as the Sun to observe the extra effects if any. Although there may be not be an analytical solution to the three body problem, but using numerical methods will allow to test the three-body problem to find Earth to Moon trajectories and compare one’s findings to another for verification.

Other Trajectory to the Moon example(s):

These figures display changes in \( \theta_j \) and \( \Delta V \) to show various trajectories from Earth to Moon. The title of the trajectory plots will show the ejected angle and velocity the particle is ejected at, and the time it takes to complete each trajectory is given in the position or velocity vs time plots. Different Trajectories will result in different flight time to target.

1.
This trajectory presents a long flight time starting from near Earth and ending at the Moon that shows a slingshot like trajectory.
9. References


10. Code (2 of them: C code and Python script)

C code:

```c
#include <stdio.h>
#include <math.h>

/*Program to calculate spacecraft position velocity and time using RK4 Method in a rotating frame of reference*/
/*Function that determines right handsie of equations*/

double RHS(int ieq, double t, double x, double y, double u_x, double u_y, double mu)
{
```

Python script:

```python
import math

def RHS(ieq, t, x, y, u_x, u_y, mu):
    # Implement the RK4 method here
    pass
```
//Define new variables of equations to simplified equations to type
double A, B, C, c, R_1, R_2;
A = (x + mu);
B = (x - 1 + mu);
c = 2.0;
C = (1 - mu);
R_1 = pow(((A*A) + (y*y)),1.5); //denom
R_2 = pow(((B*B) + (y*y)),1.5); //
//xeq:2y'+x-
((1-
mu)(x+mu)/R_1^3)
-(mu(x-
1+mu)/(R_2^3) :y' is vy
//yeq:-2x'+y-
(y(1-
mu)/R_1^3)
-(y*mu/R_2^3) : x' is vx
if(ieq == 1)
{
    return u_x; //Determine
}
if(ieq == 2)
{
    return (c*u_y) + x
}-
((A*C)/R_1)
-
((B*mu)/R_2) ;//Determine velocity x
if(ieq == 3)
{
    return u_y; //Determine position y
}
if(ieq == 4)
{
    return (-1.0*c*u_x) + (y*(1-
(C/R_1)
-
(mu/R_2)))); //Determine velocity of y
}

/*Define functions to be used to calculate several initial conditions to be test*/
double func(int ieq, double r, double theta, double mu, double dv)
{
    double v_e;
v_e = sqrt((1-
mu)/r);
double V = (v_e+dv);
    if(ieq == 1) //calculate x value
    {
        return r*cos(theta)-mu;
    }
    if(ieq == 3) //calculate x velocity for rotating frame
    {
        return V*cos(theta); //
    }
}

int main(void)
{
    /*System Setup: Earth-Moon*/
    //System Properties
    double m1 = 5.972e24; //mass of Earth kg
    double m2 = 7.348e22; //mass of Moon kg
    double M = m1+m2; //Total Mass of system kg
    double E_to_M = 384400; //distance from earth-moon in kilometers
    double r_e = R_Earth + H; //radius of spacecraft park orbit
    double r_m = 0.00451873; //radius of moon
    double theta_e = 226.81*(M_PI/180); //Phase angle earth degress 0 - 360 then convert to radians
    double theta_e = 226.81*(M_PI/180); //Phase angle earth degress 0 - 360 then convert to radians

//variables to be computed
double x, y, vx, vy; //initial coordinates to be calculated for spacecraft
x = func(1, r_e, theta_e, mu_i, dv);
y = func(2, r_e, theta_e, mu_i, dv);
vy = func(4, r_e, theta_e, mu_i, dv) - x;

//uncomment the below section to view x0 y0 to view initial starting points
/*printf("n

\nEarth
|tR|t|tx|ty|tvx|ty
",R_Earth,x,y,vx,vy);*/

/*Define initial conditions*/
double t_i = 0.0;
double x_i = x;
double y_i = y;
double u_xi = vx;
double u_yi = vy;

"RK4 Terms"/
double N = 1000000;
double step = 1/N; //step size for RK4
double K1_x, K2_x, K3_x, K4_x; // RK4 x
double K1_y, K2_y, K3_y, K4_y; // RK4 y
double K1_ux,K2_ux, K3_ux, K4_ux; // RK4 u_x
double K1_uy,K2_uy, K3_uy, K4_uy; // RK4 u_y
do //start of RK4
{  //RHS(int ieq, double t, double x, double y, double u_x, double u_y, double mu)
  /*Calculate all K1 values: full step*/
  K1_x = step*RHS(1, t_i, x_i, y_i, u_xi, u_yi, mu_i);
  K1_y = step*RHS(3, t_i, x_i, y_i, u_xi, u_yi, mu_i);
  K1_ux = step*RHS(2, t_i, x_i, y_i, u_xi, u_yi, mu_i);
  K1_uy = step*RHS(4, t_i, x_i, y_i, u_xi, u_yi, mu_i);
  /*Calculate all K2 values: half step*/
  K2_x = step*RHS(1, t_i + step/2, x_i + K1_x/2, y_i + K1_y/2, u_xi + K1_ux/2, u_yi + K1_uy/2, mu_i);
  K2_y = step*RHS(3, t_i + step/2, x_i + K1_x/2, y_i + K1_y/2, u_xi + K1_ux/2, u_yi + K1_uy/2, mu_i);
  K2_ux = step*RHS(2, t_i + step/2, x_i + K1_x/2, y_i + K1_y/2, u_xi + K1_ux/2, u_yi + K1_uy/2, mu_i);
  K2_uy = step*RHS(4, t_i + step/2, x_i + K1_x/2, y_i + K1_y/2, u_xi + K1_ux/2, u_yi + K1_uy/2, mu_i);
  /*Calculate all K3 values: half steps*/
  K3_x = step*RHS(1, t_i + step/2, x_i + K2_x/2, y_i + K2_y/2, u_xi + K2_ux/2, u_yi + K2_uy/2, mu_i);
  K3_y = step*RHS(3, t_i + step/2, x_i + K2_x/2, y_i + K2_y/2, u_xi + K2_ux/2, u_yi + K2_uy/2, mu_i);
  K3_ux = step*RHS(2, t_i + step/2, x_i + K2_x/2, y_i + K2_y/2, u_xi + K2_ux/2, u_yi + K2_uy/2, mu_i);
  K3_uy = step*RHS(4, t_i + step/2, x_i + K2_x/2, y_i + K2_y/2, u_xi + K2_ux/2, u_yi + K2_uy/2, mu_i);
  /*Calculate all K4 values: full step*/
  K4_x = step*RHS(1, t_i + step, x_i + K3_x, y_i + K3_y, u_xi + K3_ux, u_yi + K3_uy, mu_i);
  K4_y = step*RHS(3, t_i + step, x_i + K3_x, y_i + K3_y, u_xi + K3_ux, u_yi + K3_uy, mu_i);
  K4_ux = step*RHS(2, t_i + step, x_i + K3_x, y_i + K3_y, u_xi + K3_ux, u_yi + K3_uy, mu_i);
  K4_uy = step*RHS(4, t_i + step, x_i + K3_x, y_i + K3_y, u_xi + K3_ux, u_yi + K3_uy, mu_i);
  /*point update values*/
  t_i += step;
  x_i += ((K1_x + K4_x)/6) + ((K2_x + K3_x)/3);
  u_xi += ((K1_ux + K4_ux)/6) + ((K2_ux + K3_ux)/3);
  y_i += ((K1_y + K4_y)/6) + ((K2_y + K3_y)/3);
  u_yi += ((K1_uy + K4_uy)/6) + ((K2_uy + K3_uy)/3);
  printf("%e , %e , %e , %e, %e, %e", t_i, x_i, y_i, u_xi, u_yi); // print outputs to be save to file and be plotted.
} //end of RK4
while(x_i <= 1 - mu_i); // comment out above command, and uncomment this one to look at various flight times
return 0;
} //end of main
Python script:

```python
# This script runs our three body problem using python to give better visions representations, originally had C code and convert to python

###import packages###
import time
start_time = time.time()
import numpy as np
import math
import matplotlib.pyplot as plt
%matplotlib inline

# Function for initial conditions/

def func(ieq, r, theta, mu, dv):
    v_e = np.sqrt((1.0-mu)/r)
    V = v_e + dv
    if ieq == 1: # calculate initial x
        return r*np.cos(theta)-mu
    if ieq == 2: # calculate initial y
        return r*np.sin(theta)
    if ieq == 3: # calculates initial vx
        return -1.0*V*np.sin(theta) # - 2eq
    if ieq == 4: # calculates initial vy
        return V*np.cos(theta) # + 1eq

# Function for RK4/

def f(RK4, t, x, y, u_x, u_y, mu):
    # Make new parameters to simplify typing inputs inputs
    A = (x+mu)
    B = (x-1.0+mu)
    c=2.0
    R1 = ((A**2)+(y**2))**(3.0/2.0)
    R2 = ((B**2)+(y**2))**(3.0/2.0)
    # Defining equations in order: x, vx, y, vy
```
if ieq == 1;
    return u_x
if ieq == 2;
    return x*{u+}^2-{(A}*C)/(R1)-{(B}*mu/R2)
if ieq == 3;
    return u_y
if ieq == 4;
    return -(1.0*{u}*u_x)+(y*(1-(C/R1)-(mu/R2))

"Define System Parameters"

m1 = 5.9726E24 #mass of earth
m2 = 3.3486E22 #mass of moon
R = m1+m2 # total mass of system
R to M = 3.84401E5 #distance to moon
R_Earth = 6371/F_to_M #radius of Earth
R_Moon = 1737/F_to_M #radius of moon
H = 0/F_to_M #choose height above for dimensional form
r_orb = R_Earth #parameter to determine initial x,y, vx, vy
#P_orb = (2*F_to_M)*np.sqrt((r_orb**3)/(1-R_Earth)) #calculated expected orbital period, uncomment and set Dv = 0 and adjust H
theta_orb = 226.81*(np.pi/180) # covert degrees to radians
Dv = 3.14 #set ejection velocity

"Calculate Initial values"
x = func(1, r_orb, theta_orb, mu_i, Dv)
y = func(2, r_orb, theta_orb, mu_i, Dv)
vx = func(3, r_orb, theta_orb, mu_i, Dv, y) + y
vy = func(4, r_orb, theta_orb, mu_i, Dv) - x

#Print "(\text{X}|t|\text{Y}|t|\text{Vx}|t|\text{VY}|t)|\text{E}|t|\text{E}|t|\text{E}|t|\text{E}|t|\text{E}|t|\text{E}|t|\text{E}|t)" *(R_Earth, x, y, vx, vy) uncomment to print

"Define list to append values of from RK4"
t_list = [0.0]
x_list = [x]
y_list = [y]
vx_list = [vx]
yy_list = [vy]

"Define initial conditions for RK4 numerical calculation"
t_i = 0.0
x_i = x
y_i = y
u_xi = vx
u_yi = vy
N = 100000 #intervals
step = 1/N #step size

"Run RK4: Two conditions just need to uncomment one and comment out the other when examining Trajectory"
while(t_i <= 13.2):
    while(x_i <= (-mu_i)):
        """Define initial conditions"
        """
            """/Calculate all K1 values: first step/""
            K1_x = step*RHS(1, t_i, x_i, y_i, u_xi, u_yi, mu_i)
            K1_y = step*RHS(2, t_i, x_i, y_i, u_xi, u_yi, mu_i)
            K1_u_x = step*RHS(3, t_i, x_i, y_i, u_xi, u_yi, mu_i)
            K1_u_y = step*RHS(4, t_i, x_i, y_i, u_xi, u_yi, mu_i)
        """/Calculate all K2 values: half steps/"
        K2_x = step*RHS(1, t_i + step/2, x_i + K1_x/2, y_i + K1_y/2, u_xi + K1_u_x/2, u_yi + K1_u_y/2, mu_i)
        K2_y = step*RHS(2, t_i + step/2, x_i + K1_x/2, y_i + K1_y/2, u_xi + K1_u_x/2, u_yi + K1_u_y/2, mu_i)
        K2_u_x = step*RHS(3, t_i + step/2, x_i + K1_x/2, y_i + K1_y/2, u_xi + K1_u_x/2, u_yi + K1_u_y/2, mu_i)
        K2_u_y = step*RHS(4, t_i + step/2, x_i + K1_x/2, y_i + K1_y/2, u_xi + K1_u_x/2, u_yi + K1_u_y/2, mu_i)
        """/Calculate all K3 values: full steps/"
        K3_x = step*RHS(1, t_i + step, x_i + K2_x, y_i + K2_y, u_xi + K2_u_x, u_yi + K2_u_y, mu_i)
        K3_y = step*RHS(2, t_i + step, x_i + K2_x, y_i + K2_y, u_xi + K2_u_x, u_yi + K2_u_y, mu_i)
        K3_u_x = step*RHS(3, t_i + step, x_i + K2_x, y_i + K2_y, u_xi + K2_u_x, u_yi + K2_u_y, mu_i)
        K3_u_y = step*RHS(4, t_i + step, x_i + K2_x, y_i + K2_y, u_xi + K2_u_x, u_yi + K2_u_y, mu_i)
        """/Update conditions/"
        t_i += step
        x_i += ((K1_x + K4_x)/6) + ((K2_x + K3_x)/3)
        u_xi += ((K1_u_x + K4_u_x)/6) + ((K2_u_x + K3_u_x)/3)

Baines, Hew, and Toyama 18
y_i += ((K1_y + K4_y) / 6) + ((K2_y + K3_y) / 3)
y_yi += ((K1_yi + K4_yi) / 6) + ((K2_yi + K3_yi) / 3)

# Plot both vx and vy velocity vs time
f = plt.figure(figsize=(10, 10))
plt.title("Velocity vs Time", fontsize=16)
plt.xlabel("Unit Time")
plt.plot(Time, vx_val, 'k--', label = "V_x(t)")
plt.plot(Time, vy_val, 'c--', label = "V_y(t)")
plt.legend()
plt.grid(linestyle='--')
plt.savefig("vvt.png") # uncomment to save fig, velocity vs time

**Plot for Earth and moon circle**
phi = np.linspace(0, 2*np.pi, 100) # generate values for plot
r_Ex = R_Earth*np.cos(phi)-mu_i
r_Ey = R_Earth*np.sin(phi)
r_Mx = R_Moon*np.cos(phi)+1-mu_i
r_My = R_Moon*np.sin(phi)

***Plotting Features***
f = plt.figure(figsize=(10,10))
plt.title("Direct Transfer: $\Delta V = 3.14, \$\theta(\text{deg})=226.81\$\circ$")
plt.xlabel("(nondimensional)")
plt.ylabel("(nondimensional)")
plt.axis([-0.2,1.1,-0.1,0.1])
plt.axis([0.9,1.1,-0.025,0.025])
plt.axis([-0.3,1,1,-0.5,2.0])
plt.plot((0,0),'r') # center of mass of earth-moon system
plt.plot(x_val, y_val, 'k--') # plot x y phase space
plt.plot(r_Ex, r_Ey, 'b--', 'mu_i,0, b.') # plot radius of earth
plt.plot(r_Mx, r_My, 'k--', 'mu_i, 0, k,') # plot radius of moon
plt.grid(linestyle = '--')
plt.savefig("226_81deg.png") # save fig uncomment rerun to save

plt.savefig("vvt.png") # uncomment to save fig, velocity vs time
plt.show()

print ("My program took", time.time() - start_time, "to run") # print time that it takes to run code