The Three-Body Problem

Dylan Oelmann, Andrew MacDiarmid

May 16, 2019

Contents

1 Introduction **3**

2 Deriving the Equations for the Newtonian Three-Body Problem **7**

3 Coding the Three-Body Problem **10**

4 Case Studies **14**

4.1 The Earth-Moon-Sun System 14

4.2 The Lemniscate 15

4.3 Burrau's Chaotic Solution 18

5 Conclusion **19**

6 Appendix **20**

6.1 The Equations 20

6.2 The Code 21

Abstract

The three-body problem studies the motion of three mutually attracting gravitational bodies, given their positions and initial velocities. The chaotic nature of the model due to its high sensitivity to initial conditions renders it impossible to use in the prediction of real world phenomena. Some of the most influential mathematicians studied the problem, including Newton, Euler, Lagrange, Jacobi, and Poincaré. This paper uses the Newtonian law of gravity to model the motion of three bodies. Using python, simulations were run using three different sets of initial conditions: The conditions for the Earth-Sun-Moon system, the Lemniscate solution, and Burrau's solution. Case studies were done on each to analyze the results they produced, and we discuss the solver error of the models. The solutions all differ from their true, analytic solutions due to this error. The three-body problem can be expanded to the n-body problem, which has various applications in the real world, such as providing models of the orbits of the outer planets, the planets in the solar system, and even all the stars in the milky way galaxy. These models cannot be used to make predictions however, since the highly chaotic dependence on unmodelled variables can cause great variation in comparison to the model. The nonexistent general analytic solution to the problem could be used to make accurate predictions of phenomena in our solar system and universe, which would further develop the human understanding of the world.

1 Introduction

The three-body problem is a mathematical problem on the motion of three, mutually attracting bodies given their initial positions and velocities. It models the motion of celestial bodies, and has been one of the most significant mathematical problems to date. Beginning with Kepler, the three-body problem has been studied by some of the most influential mathematicians in history, including Newton, Euler, Lagrange, Jacobi, and Poincaré. Despite this, a general analytic solution to the problem has yet to be found.

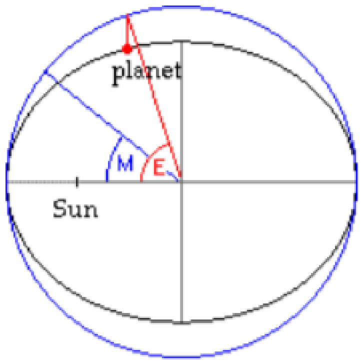
One of the first people to model the motion of celestial objects was Johannes Kepler. In his publishings titled *Astronomia Nova* and *Epitome Astronomiae Copernicanae*, he introduced his three laws of planetary orbit, which state that planets follow elliptical paths around the sun, with higher velocities at their perigees and smaller velocities at their apogees. He also introduced Kepler's equation:

$$E(t) - e * sinE(t) = M(t)$$

This introduced the eccentricity anomaly, E, the mean anomaly, M, and the eccentricity, e, of the orbit. The eccentricity anomaly of a body orbiting in elliptical motion is a parameterization of polar angle. It defines the position of a body moving along an elliptical orbit. The mean anomaly represents the angular distance from the perihelion if the orbit were circular (which made it a parameter of time), and was defined by:

$$M = \frac{2\pi t}{T}$$

Where t represents time since periapsis, and T represents period. The eccentricity is a value between 0 and 1 which represented the "flatness" of the ellipse, where 1 represents a perfect circle and 0 represents a line. The following diagram demonstrates what each value represents in an elliptical orbit:



Kepler had found a model which successfully made predictions of the orbit of the planets in the solar system; however why these planets followed orbital paths around the sun were still a mystery to him. The answer to this question would not come until Isaac Newton produced his groundbreaking work on the three laws of motion and law of gravitation.

Newton solved the problem of why planets orbit the sun when he published the *Philosophiæ Naturalis Principia Mathematica* in 1687, which contained his theory of gravity. This formalized Kepler's laws, and united them with "Galileo Galilei's theory of falling bodies" (thestargarden.co.uk 2019). After Newton had solved the case of two bodies that exert a gravitational force on each other, he turned his attention to the case of three bodies, in an attempt to include the moon in the model. The attempt to model lunar motion with the effects of gravity by both the Sun and Earth is the first case of the three-body problem, which was mentioned in his Principia; however, lunar motion proved to be quite difficult for Newton. In fact, the majority of the progress that was made on Lunar theory was after Newton's lifetime. Newton had claimed that his "head never ached but with his studies on the moon". He provided a model that was inconsistent with the dynamics of the real-world moon, due to a condition he imposed on it: He only accounted for the radial component of disturbing forces, ignoring the tangential component. This was later fixed, however many people were skeptical of his results. Despite struggling, and potentially failing at modelling lunar motion, his work on the motion of three bodies was still extremely useful. Once Uranus was discovered, a model of its movement given by Newton's laws dictated that an additional eighth planet was required to give it the motion that it had. This led to the discovery of Neptune, and Newton's law of gravity predicted its motion with high accuracy. In modern day, Newton's equations work particularly well in computer simulations of the three-body problem.

The first exact stable solution of the three-body problem was found by Leonhard Euler in 1767, which was a motion on a collinear ellipse. He was also the first person to propose and study the restricted three-body problem in 1772. In the context of the Earth-Sun-Moon system, the restricted three-body problem models the motion of the moon where it is considered massless, and the orbit of the Earth around the Sun is circular and not elliptical. In other words, it sets one mass so that the gravitational effects produced by it are negligible relative to the other masses, and assumes circular orbit. At the same time, Lagrange was also working on the three-body problem: he had discovered the equilateral triangle solution. This also helped him develop Lagrangian Mechanics, which were important to not only the three-body problem, but to general dynamical systems. Later on, Euler also studied the perturbations in planetary motion by using variation of parameters.

Carl Gustav Jacob Jacobi took a particular interest in Euler's restricted three-body problem in 1836, and did extensive work on it. He introduced the Jacobi integral, or Jacobi constant, which relates position to velocity at any point (gereshes.com 2018). It is a constant of motion, and is the only known conserved quantity in the restricted three-body problem. It served two purposes. The first being "to construct surfaces of zero velocity which limit the regions of space in which the small body under given initial conditions, can move" (Ovendey, Roy 1960), which was used to make long-term predictions. The second was "to derive a criterion for the re-identification of a comet whose orbit has suffered severe perturbations by a planet" (Ovendey, Roy 1960), which provided short-term predictions.

The Jacobi integral can be defined as

$$C_J = n^2(x^2 + y^2) + 2(\frac{\mu_1}{r_1} + \frac{\mu_2}{r_2}) - (x'^2 + y'^2 + z'^2)$$

In 1878, George Hill demonstrated the useful application of the Jacobi integral, "describing the regions of possible motion for the body of negligible mass (Worthington 2012).

The final major historical event to be discussed on the three-body problem is the work done by Poincaré. In 1884, a man by the name of Mittag-Leffler wanted to arrange an international prize competition in mathematics, to honour King Oscar II of Sweden's 60th birthday. The competition was arranged, and three judges were chosen: Leffler, Karl Weierstrass, and Charles Hermite. The prize was 2,500 Swedish Kronor (an instructors salary at the time was 7,000 Kronor per year), and a gold medal. The challenge was to solve an unsolved mathematical problem of the judges' choosing, which happened to be the following:

For a system of arbitrarily many mass points that attract each other according to Newton's laws, assuming no two points ever collide, find a series expansion of the coordinates of each point in known fractions of time conveying uniformly for any period of time.

This was the n-body problem. The hope was to find a fundamental concept that could provide a stable solution to the system, in order to model the dynamics of the solar system. A model of the solar system would help answer many of the questions they had, such as: could a planet be ejected from the solar system due to chaos? A variety of mathematicians worked on the problem, including Poincaré. Poincaré had a close connection to Leffler, and worked extensively on the problem, finally submitting his memoir titled "Sur les problèmes des trois corps et les équations de la dynamique", which also contained the first demonstration of his recurrence theorem. Poincaré's paper was in a class of its own, as he tackled the problem much differently than was expected. He had not solved the problem, however his work on it was considered revolutionary enough to be considered prize-worthy, making him set to win the competition. This was conditional, however: the paper was "written in his typical, rather intuitive style of omission of many details" (Institut Mittag-Leffler 2019), and he was given the chance to attach an appendix to clarify his work. This was done, and Poincaré won the competition; however immediately after printing of his paper started, he immediately contacted Leffler and told him to halt printing: he had found an error. He had originally assumed a stability in the three-body problem, however his revision led him to discover that the system could be much more chaotic than he anticipated. His error led him to discover that the highly "sensitive dependence on initial conditions renders it meaningless to make predictions of the fate of an orbit". His correction lay the foundation of "chaos theory", which changed many core ideas of the mathematical study of dynamics, which saved his reputation from his error entirely. Poincaré had shown that a solution to the three-body problem was near impossible because of the highly chaotic nature of the system. To this day, a general solution to the three-body problem that can make accurate predictions of the real world has yet to be found; however mathematicians still enjoy finding stable solutions of the problem, regardless if they reflect the real world.

In this paper, the Newtonian gravitational equation will be used to model the motion of three bodies. Using computer simulation, the equations of the model will be encoded to analyze various initial conditions of the three-body problem in and provide three case studies. The case studies will show the chaotic nature of the system, as well as stable solutions. The masses will also be considered points which never collide, similar to what was seen in the Mittag-Leffler competition.

2 Deriving the Equations for the Newtonian Three-Body Problem

Using Newton's law of gravitation, the gravitational force exerted on mass 1 by masses 2 and 3 reads as follows:

$$\Sigma F = m_1\vec{a}_1 = -\frac{Gm_1m_2}{(r_{21})^2}\hat{r}_{21} - \frac{Gm_1m_3}{(r_{31})^2}\hat{r}_{31}$$

Here, m_1, m_2, and m_3 are masses 1, 2 and 3 respectively, G is the gravitational constant $6.67 * 10^{-11}$, r_{21} and r_{31} are the distances between masses 1 and 2, and masses 1 and 3 respectively, and $\hat{r}_{21}$ and $\hat{r}_{31}$ are the unit vectors of mass 1 pointed to masses 2 and 3 respectively. Some of the examples that will be discussed in the case study set the gravitational constant to 1 in order to simplify the notation. Now, we will expand the equation in order to simplify it. Denote:

$\vec{a}_1 = \vec{r}_1''$
$r_{21} = ||\vec{r}_2 - \vec{r}_1||$
$r_{31} = ||\vec{r}_3 - \vec{r}_1||$
$\hat{r}_{21} = \frac{(\vec{r}_2-\vec{r}_1)}{||\vec{r}_2-\vec{r}_1||}$
$\hat{r}_{31} = \frac{(\vec{r}_3-\vec{r}_1)}{||\vec{r}_3-\vec{r}_1||}$

Substituting these terms into the above equation results in the expanded form:

$$m_1\vec{r}_1'' = -\frac{Gm_1m_2}{||\vec{r}_2-\vec{r}_1||^2}\frac{(\vec{r}_2-\vec{r}_1)}{||\vec{r}_2-\vec{r}_1||} - \frac{Gm_1m_3}{||\vec{r}_3-\vec{r}_1||^2}\frac{(\vec{r}_3-\vec{r}_1)}{||\vec{r}_3-\vec{r}_1||}$$

which can be simplified to the following by grouping the magnitudes together and cancelling m_1:

$$\vec{r}_1'' = -\frac{Gm_2(\vec{r}_2-\vec{r}_1)}{||\vec{r}_2-\vec{r}_1||^3} - \frac{Gm_3(\vec{r}_3-\vec{r}_1)}{||\vec{r}_3-\vec{r}_1||^3}$$

Furthermore, $\vec{r}_1$, $\vec{r}_2$, and $\vec{r}_3$ are vectors that encapsulate the position of mass 1 relative to masses 2 and 3. These can be expanded to represent the direction vectors that capture the relative positions of the masses in space:

$$\vec{r}_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, \quad \vec{r}_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}, \quad \vec{r}_3 = \begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix}$$

Substituting these into the equation results in the following:

$$\vec{r}_1'' = -\frac{Gm_2(\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix})}{||\vec{r}_2 - \vec{r}_1||^3} - \frac{Gm_3(\begin{bmatrix} x_3 \\ y_3 \\ z_3 \end{bmatrix} - \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix})}{||\vec{r}_3 - \vec{r}_1||^3}$$

This can be broken up into three equations for each row in the column vector, which represents the $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ components of the acceleration of mass 1. Additionally, the magnitudes will remain the same, and not be broken up into their components.

Now this second order differential equation (three equations in actuality) will be reduced to a first order differential equation. Furthermore, it will be rewritten in a more compact form that is easier to understand.

First, we will re-parameterize x_1 as a_{0x}, y_1 as a_{0y}, and z_1 as a_{0z}, which are the x, y, and z coordinates of mass 1. Mass 2 will be denoted by b, such that $x_2 = b_{0x}$, $y_2 = b_{0y}$, and $z_2 = b_{0z}$. Mass 3 will be denoted by c, such that $x_3 = c_{0x}$, $y_3 = c_{0y}$, and $z_3 = c_{0z}$. Furthermore, we will say that

$$a'_{0x} = a_{1x} \tag{1}$$
$$a'_{0y} = a_{1y} \tag{2}$$
$$a'_{0z} = a_{1z} \tag{3}$$

$$b'_{0x} = b_{1x} \tag{4}$$
$$b'_{0y} = b_{1y} \tag{5}$$
$$b'_{0z} = b_{1z} \tag{6}$$

$$c'_{0x} = c_{1x} \tag{7}$$
$$c'_{0y} = c_{1y} \tag{8}$$
$$c'_{0z} = c_{1z} \tag{9}$$

Thus the quantities with sub-index 0 are positions and those with sub-index 1 are velocities. This rewrites the equation as a system of first-order differential equations. In order to simplify this, we will denote a_{0x}, a_{0y}, and a_{0z} as $a_{0\alpha}$, where α = x, y, z. The same applies to b and c. Thus, the derivatives of the velocities are accelerations, and Newton's equation reads:

$$a'_{1\alpha} = -\frac{Gm_2(b_{0\alpha} - a_{0\alpha})}{||\vec{r}_2 - \vec{r}_1||^3} - \frac{Gm_3(c_{0\alpha} - a_{0\alpha})}{||\vec{r}_3 - \vec{r}_1||^3} \quad \textit{where } \alpha = x, y, z \qquad (10, 11, 12)$$

which represents three equations, where α = x, y, and z. This represents the final three equations required to model mass 1. The differential equations for masses 2 and 3 are derived in a similar way, which provides the final equations for the model:

$$b'_{1\alpha} = -\frac{Gm_1(a_{0\alpha} - b_{0\alpha})}{||\vec{r}_1 - \vec{r}_2||^3} - \frac{Gm_3(c_{0\alpha} - b_{0\alpha})}{||\vec{r}_3 - \vec{r}_2||^3} \quad \textit{where } \alpha = x, y, z \qquad (13, 14, 15)$$

$$c'_{1\alpha} = -\frac{Gm_1(a_{0\alpha} - c_{0\alpha})}{||\vec{r}_1 - \vec{r}_3||^3} - \frac{Gm_2(b_{0\alpha} - c_{0\alpha})}{||\vec{r}_2 - \vec{r}_3||^3} \quad \textit{where } \alpha = x, y, z \qquad (16, 17, 18)$$

Thus we have all of the equations necessary to model the three-body problem. A table containing all 18 equations can be found in the Appendix.

Furthermore, we can generalize the Newtonian gravitational equations to a summation:

$$\vec{r}_i'' = G \sum_{i \neq j}^{3} \frac{m_j(\vec{r}_j - \vec{r}_i)}{||\vec{r}_j - \vec{r}_i||^3}, \qquad \textit{where } \ i = 1, 2, 3$$

This allows us to extend it to the n-body problem, with n masses:

$$\vec{r}_i'' = G \sum_{i \neq j}^{n} \frac{m_j(\vec{r}_j - \vec{r}_i)}{||\vec{r}_j - \vec{r}_i||^3}, \qquad \textit{where } \ i = 1, 2, ..., n$$

Which represents the force experienced by a mass given all other masses. Similarly to the three-body problem, this would have n-1 summations for each mass in the system, which results in a very high number of degrees of freedom. An additional six equations would be added to the system for every mass added above three.

These equations will be encoded, and used to model different masses and initial conditions. Additionally, three case studies will be done. In honour of the problem's history, a model of the Earth-Sun-Moon system will be made, as well as a very stable lemniscate orbit, and a highly chaotic system called the Burrau's solution.

3 Coding the Three-Body Problem

Now, an explanation of the code that was used for the model will be provided. This section will go over the code box by box, and the complete code can be found in the Appendix. The coding language that was used was python.

The first box of code simply imports all the necessary packages for the code. The second box is used to define the initial conditions:

```
#Constants
#Earth-Moon-Sun Constants
#G = 6.67*10**(-11)
#m1 = 1.989*10**30 #sun
#m2 = 5.972*10**24 #earth
#m3 = 7.348*10**22 #moon
#Lemniscate Constants
#G = 1
#m1 = 1
#m2 = 1
#m3 = 1
#Burrau's Problem Constants
G = 1
m1 = 3
m2 = 4
m3 = 5
```

Here, G denotes the gravitational constant, and m1, m2, and m3 denote masses 1, 2, and 3 respectively (in kg). Pound symbols are placed in front of the initial conditions that are not being run. In this case, the code will run using the Burrau system constants. The third box is given by

```
def der_state(t, state): """compute the derivative of the given state"""
#Vectors and Magnitudes
r1 = np.array([state[0], state[2], state[4]]) #x1, y1, z1
r2 = np.array([state[6], state[8], state[10]]) #x2, y2, z2
r3 = np.array([state[12], state[14], state[16]]) #x3, y3, z3

r21 = r2 - r1
r31 = r3 - r1
r12 = r1 - r2
r32 = r3 - r2
r13 = r1 - r3
```

```
r23 = r2 - r3

R21 = np.linalg.norm(r21)
R31 = np.linalg.norm(r31)
R12 = np.linalg.norm(r12)
R32 = np.linalg.norm(r32)
R13 = np.linalg.norm(r13)
R23 = np.linalg.norm(r23)

#System of Differential Equations
der = np.zeros_like(state)
der[0] = state[1] #a0x' = a1x
der[2] = state[3] #a0y' = a1y
der[4] = state[5] #a0z' = a1z
der[6] = state[7] #b0x' = b1x
der[8] = state[9] #b0y' = b1y
der[10] = state[11] #b0z' = b1z
der[12] = state[13] #c0x' = c1x
der[14] = state[15] #c0y' = c1y
der[16] = state[17] #c0z' = c1z

#The Actual Equations
#Sun
der[1] =
G*((m2*(state[6]-state[0])/R21**3)+(m3*(state[12]-state[0])/R31**3))
der[3] =
G*((m2*(state[8]-state[2])/R21**3)+(m3*(state[14]-state[2])/R31**3))
der[5] =
G*((m2*(state[10]-state[4])/R21**3)+(m3*(state[16]-state[4])/R31**3))

#Earth
der[7] =
G*((m1*(state[0]-state[6])/R12**3)+(m3*(state[12]-state[6])/R32**3))
der[9] =
G*((m1*(state[2]-state[8])/R12**3)+(m3*(state[14]-state[8])/R32**3))
der[11] =
G*((m1*(state[4]-state[10])/R12**3)+(m3*(state[16]-state[10])/R32**3))

#Moon
der[13] =
G*((m1*(state[0]-state[12])/R13**3)+(m2*(state[6]-state[12])/R23**3))
der[15] =
G*((m1*(state[2]-state[14])/R13**3)+(m2*(state[8]-state[14])/R23**3))
der[17] =
G*((m1*(state[4]-state[16])/R13**3)+(m2*(state[10]-state[16])/R23**3))

return der
```

Here, “state” is used to denote the 18 degrees of freedom of the system, and “der” is used to denote the derivative of the state. First, $\vec{r}_1$, $\vec{r}_2$, and $\vec{r}_3$ are used to define the xyz-coordinates of the masses 1, 2, and 3 respectively. In the case of $\vec{r}_1$, it is given states 0, 2, and 4, which denote the x, y, and z positions of mass 1. This allows the $\vec{r}_1$ vector to continuously update based on the new position of mass 1. $\vec{r}_{21}$, $\vec{r}_{31}$, ..., $\vec{r}_{23}$ are the differences of the $\vec{r}_2$ and $\vec{r}_1$ vectors, $\vec{r}_3$ and $\vec{r}_1$, and so on. This is done out of convenience for the next step, where R21, R31, ..., R23 denote the respective magnitudes of the difference vectors $\vec{r}_{21}$, $\vec{r}_{31}$, ..., $\vec{r}_{23}$. Writing the magnitudes like this allows the equations of the system to be written much neater.

Furthermore, the terms are now redefined such that they represent a first order system of equations. der[0] = state[1] corresponds to $a'_{0x} = a_{1x}$, as it was defined when deriving the equations. Once all of the velocities are defined in terms of $a_{1\alpha}$, the Newtonian equations can be rewritten as first order equations, which can be seen in the code. Here, there are 3 equations for each mass, which correspond to the alphas = x, y, and z. Thus, all the equations for the three-body problem are encoded.

Box four denotes the various runtimes of each simulation.

```
#tf for Earth-Sun-Moon System #tf = 6000000 #simulation for tf seconds
#Lemniscate #tf = 30 #simulation for tf seconds
#Burrau's Problem Initial Conditions tf = 20 #simulation for tf seconds
n = 1000 #number of evaluation points dt = tf/n T = np.linspace(0.0, tf,
n+1)
```

For convenience, all three runtimes are included, considering that they cannot use the same ones. Given that the actual values of the Earth, Sun, and Moon are given, if the solver ran in real time it would take a full year to complete one rotation around the sun. Thus this system is given a much higher tf, which is compressed down to 30 seconds in the animation. Additionally The lemniscate ran better given a tf of 30, and the Burrau under a tf of 20. Once again, pound symbols are placed in front of the code not to be run. In order to remain consistent, Burrau's system is used in this example. All three systems use the same number of n = 1000 evaluation points.

The fifth box defines the initial conditions

```
#Initial State
#([xa, vxa, ya, vya, za, vza, xb, vxb, yb, vyb, zb, vzb, xc, vxc, yc, vyc, zc,
vzc])

#Earth-Sun-Moon Initial Conditions
```

```
#state0 = [0, 0, 0, 0, 0, 0, 149.6*10**9, 0, 0, 30000, 0, 0, 1.49984*10**11, 0, 0,
33000, 0, 0]

#Lemniscate Initial Conditions
#state0 = [0.97000436, 0.4662036850, -0.24308753, 0.4323657300, 0, 0, 0,
-0.93240737, 0, -0.86473146, 0, 0, -0.97000436, 0.4662036850, 0.24308753,
0.4323657300, 0, 0]

#Burrau’s Problem Initial Conditions
state0 = [1.0, 0.0, 3.0, 0.0, 0.0, 0.0, -2.0, 0.0, -1.0, 0.0, 0.0, 0.0, 1.0, 0.0, -1.0,
0.0, 0.0, 0.0]
```

Here, state0 defines the array that corresponds to the initial values of all 18 states, starting from state[0] to state[17].

Box 6 solves the differential equation using solve_IVP.

```
#Solve_IVP
sol = integrate.solve_ivp(der_state, (0, tf), state0, t_eval=T, method=”RK45”)
aX = sol.y[0]
ay = sol.y[2]
az = sol.y[4]
bx = sol.y[6]
by = sol.y[8]
bz = sol.y[10]
cx = sol.y[12]
cy = sol.y[14]
cz = sol.y[16]
```

The solutions to each state are redefined such that they reflect the equations derived. aX corresponds to the solution of m1, where α = x, and so on. This is also where the solver method is defined. In this case, Runge-Kutta was used (defined as ”RK45”), since the Burrau solution was computationally heavy for the computer being used. This is a less powerful solver than LSODA (which stands for ” Livermore Solver for Ordinary Differential Equations”). This is used for the lemniscate and the Earth-Sun-Moon system, and provides much more accurate results.

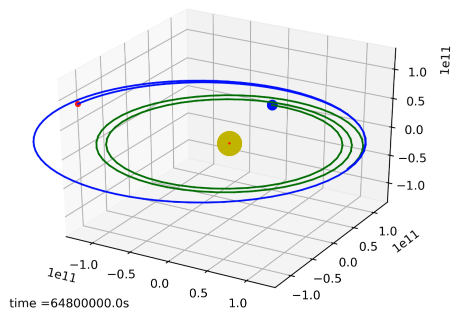
Finally, the animation box is defined, which plots and animates a graph of the movement of the three bodies. In the case study, these graphs will be analyzed and discussed, and will include simulations of each system.

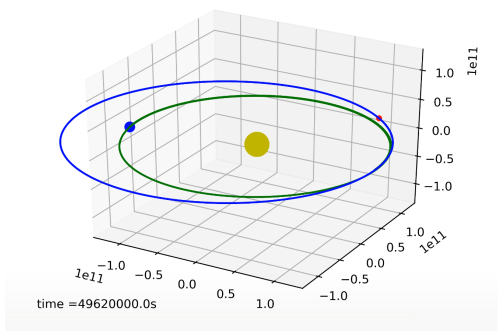
4 Case Studies

Now that the system of equations for the three-body problem has been encoded, simulations of various initial conditions can be run. This section will discuss three different systems: The Earth-Sun-Moon system, the Lemniscate orbit, and the Burrau's solution.

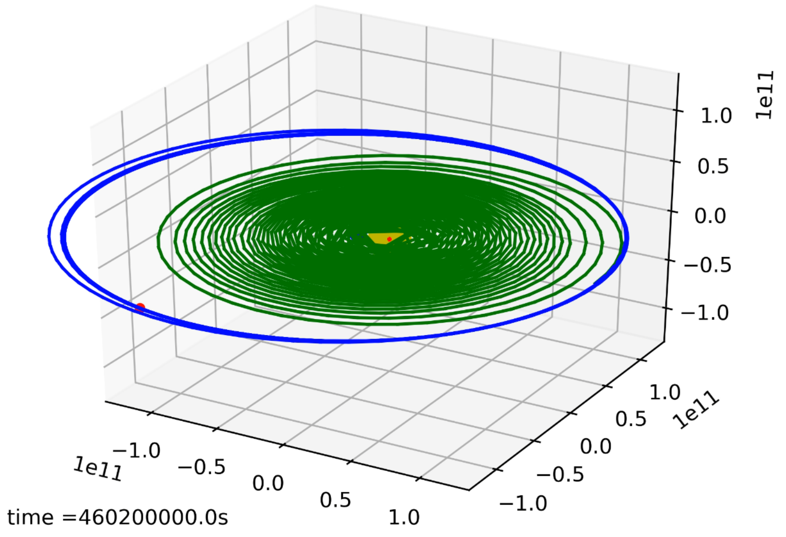
4.1 The Earth-Moon-Sun System

The first case that will be analyzed is the the Earth-Sun-Moon System, which was the first case of the three-body problem ever discussed historically. In this case, the masses of the Earth, Sun, and Moon are taken to be their actual masses, which are $5.972 * 10^{24}$ kg, $1.989 * 10^{30}$ kg, and $7.348 * 10^{22}$ kg respectively. Furthermore the gravitational constant is taken to be its actual value as well. The Sun is placed at the origin of the 3D space, the Earth is $1.50 * 10^{11}$ m from the sun in the positive x-axis (which represents its actual distance from the sun), and the moon will be placed an additional $3.844 * 10^7$ m along the x-axis to represent its distance from the Earth, placing it at $1.5000384 * 10^{11}$ m from the sun. Furthermore, the velocity of the sun is initially set to zero, the earth is given a velocity of $30,000$ $\frac{m}{s}$ in the z-direction, (making the velocity zero in the x and y directions), and the moon is given a velocity of $33,000$ $\frac{m}{s}$ in the z-direction so that it can orbit the Earth. This means that the orbit of the Earth and Moon around the Sun are in the x-z plane, where the y-axis is not considered. The true velocity of the moon is $31,000$ $\frac{m}{s}$, however using $33,000$ $\frac{m}{s}$ produced cleaner results. Thus, state[6] is set to $149.6 * 10^9$, state[9] is set to 30000, state[12] is set to $1.49984 * 10^{11}$, and state[15] is set to 33000. The rest of the states are set to zero.





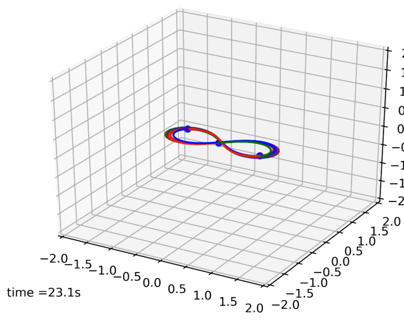
Here, the graph on the left represents the system when it was solved using RK45, and the graph on the right represents it when LSODA was used. The Earth is denoted by the blue circle with green path, the Moon is defined as the red dot with blue path, and the Sun is defined at the yellow circle in the center of the orbits. One can see that using RK45, the Earth slowly spirals into the sun, which thankfully is inaccurate. This is due to the error in the solver, thus a stronger solver is required. LSODA provides consistent orbits, which only slightly deviate from their initial orbits. Despite being a better solver, LSODA fails to produce results that match the real world system, where the moon orbits the Earth. The moon drifts off into its own orbit around the sun as time increases. This is most likely due to error in the solver. As the RK45 solution of this case is run over a longer period of time, it produces the following graph:

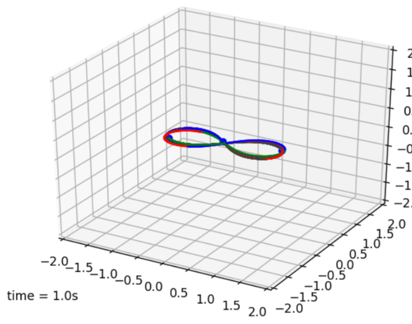


From this, one can observe how drastically the error in a weaker solver can compile to produce highly inaccurate results. The Moon's orbit maintains its integrity around the Sun for the most part, however the Earth spirals into the sun, which progressively gets worse as the solver is run. It can be seen that the outer orbit of the Earth is round, however the inner orbit is polygonal, and looks like a quadrilateral.
To view a video simulation of this case study, click here to view the simulation using RK45, and here for the one using LSODA.

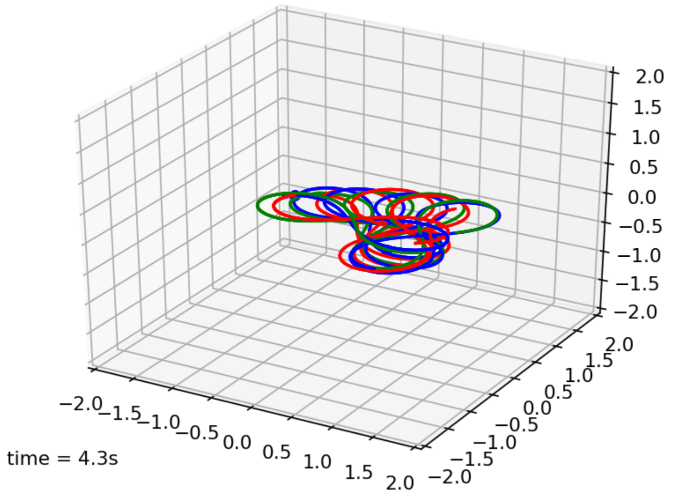
4.2 The Lemniscate

The lemniscate orbit is a relatively simple orbit in which the three masses follow each other to produce a figure-8 pattern. The model is relatively simple as the three masses and the gravitational constant are all set to 1. The complication in this model is finding initial positions and velocities such that they are given the very stable lemniscate orbit. This requires precise initial positions and velocities, of up to ten decimal places. When RK45 and LSODA were used on these initial conditions, they produced quite different deviations as a result of solver error. In the case of RK45, the lemniscate shrunk as the masses began to move towards each other. In the case of LSODA however, the masses follow a relatively stable orbit around each other, but after a certain amount of time, a slight rotation of the system can be observed, as if it were rotating around some central axis. These are both very different errors considering the only difference was the solver used.

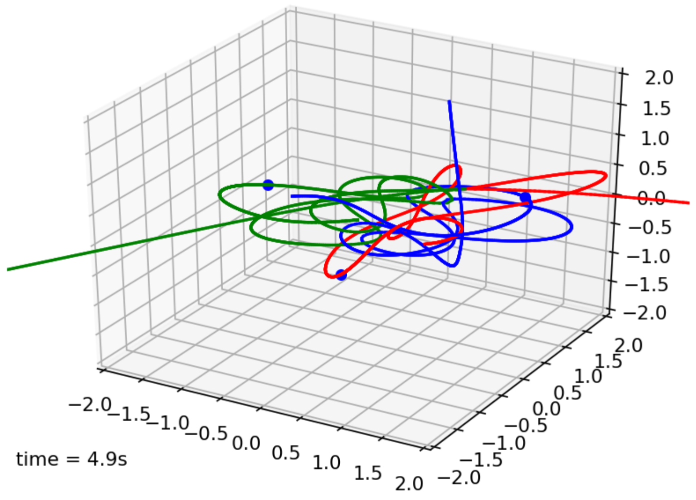




The left graph demonstrates the RK45 solution, whereas the right one demonstrates the LSODA solution. A system like this would be very rare in nature due to its extremely high sensitivity to initial conditions. Considering the chaotic state of the universe, the alteration of any one of these masses would send the system into a more chaotic state. To test this, the only the x-component of the initial velocity of mass 1 was altered, from 0.4662036850 to 0.5662036850. This resulted in the following motion:



The masses still follow each other, however the stable lemniscate orbit was immediately lost into a chaotic state. As a result, this motion would be extremely rare in nature because perturbations much greater than the one just tested are present in the universe. In order to further explore the sensitivity of the orbit, every velocity component of every mass was increased by 0.1. This resulted in a highly chaotic yet elegant solution, in which the masses drift apart from each other given time.



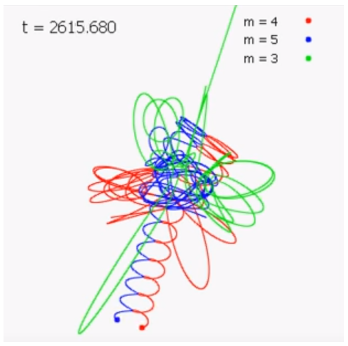
This reflects the idea presented by Poincaré on how the sensitive dependence on initial conditions renders long-term predictions of the real world impossible.

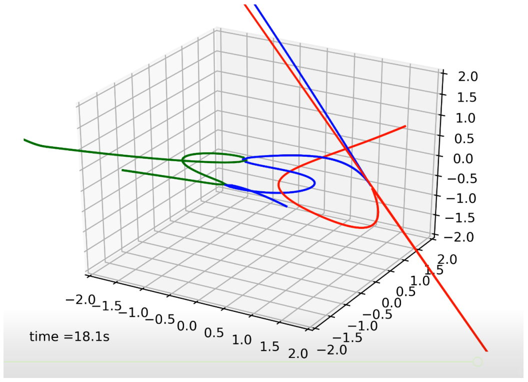
Any given orbit in nature could be greatly affected by the chaos present in the universe, which will increase the inaccuracy of the predictions as time goes on.

To view the RK45 solution, click here, for the LSODA solution click here, for the first chaotic solution click here, and for the second chaotic solution, here.

4.3 Burrau's Chaotic Solution

A perfect example of a chaotic solution to the three-body problem is Burrau's solution, in which the masses are all placed at a relative distance from each other and given no initial velocities. As a result, the motion is caused by the gravitational force between the three masses. In this case, a stable solution is produced despite the chaos of the motion, which eventually causes the masses to drift apart. The computer used for this simulation was unable to run the system using the LSODA solver, meaning only RK45 was used. In comparison to the theoretical results, the obtained results were extremely inaccurate:





The left image depicts the theoretical results, whereas the right demonstrates the results obtained using RK45. The theoretical solution has the three masses orbit each other for several minutes before they drift apart from each other; however the RK45 solution only ran for a few seconds before flying away from each other. A much stronger solver could produce results that more closely resemble the theoretical results.

A simulation of the theoretical result can be seen here, and the obtained results can be seen here

5 Conclusion

The three-body problem is a mathematical problem and an important physical model which studies the motion of three, mutually attracting bodies. Their motion is highly dependent on the initial positions and velocities they are given. Starting with Kepler and Newton, some of the most influential mathematicians have worked on the problem; however a general analytical solution to the problem has yet to have been found due to its highly chaotic nature: all specific cases that can be modelled are done so numerically. An analytical solution of the problem could be used to model the movements of bodies in space and our solar system. Unfortunately, the model's high sensitivity to the conditions imposed on it makes it impossible to use the model for long-term predictions of the solar system or mutual attraction of multiple masses. Today, the three-body problem and n-body problem are used to model the motion of pluto, of comets, and of rocket trajectories. A more interesting application of the n-body problem is being worked on by computational astronomers, who are attempting to model the motion of all stars in our galaxy and tracing their trajectories over a period of several billion years (Hayes 2015). This effectively makes it an n-body problem with approximately 10^{11} bodies. To model such a system requires much more research and technological development. Successfully modelling this system could provide information on a variety of things in our solar system that we not may have thought existed, and could lead to an understanding of space that would otherwise be impossible; however it all starts with understanding the solutions of the three-body problem.

6 Appendix

6.1 The Equations

1. $a'_{0x} = a_{1x}$
2. $a'_{0y} = a_{1y}$
3. $a'_{0z} = a_{1z}$
4. $b'_{0x} = b_{1x}$
5. $b'_{0y} = b_{1y}$
6. $b'_{0z} = b_{1z}$
7. $c'_{0x} = c_{1x}$
8. $c'_{0y} = c_{1y}$
9. $c'_{0z} = c_{1z}$

$$a'_{1\alpha} = -\frac{Gm_2(b_{0\alpha} - a_{0\alpha})}{||\vec{r}_2 - \vec{r}_1||^3} - \frac{Gm_3(c_{0\alpha} - a_{0\alpha})}{||\vec{r}_3 - \vec{r}_1||^3}$$

$where:$

10. $\alpha' = x$
11. $\alpha' = y$
12. $\alpha' = z$

$$b'_{1\alpha} = -\frac{Gm_1(a_{0\alpha} - b_{0\alpha})}{||\vec{r}_1 - \vec{r}_2||^3} - \frac{Gm_3(c_{0\alpha} - b_{0\alpha})}{||\vec{r}_3 - \vec{r}_2||^3}$$

$where:$

13. $\alpha' = x$
14. $\alpha' = y$
15. $\alpha' = z$

$$c'_{1\alpha} = -\frac{Gm_1(a_{0\alpha} - c_{0\alpha})}{||\vec{r}_1 - \vec{r}_3||^3} - \frac{Gm_2(b_{0\alpha} - c_{0\alpha})}{||\vec{r}_2 - \vec{r}_3||^3}$$

$where:$

16. $\alpha' = x$
17. $\alpha' = y$
18. $\alpha' = z$

6.2 The Code

Each box of code is separated by a line so that it is easily legible

---

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import matplotlib.animation as animation
from mpl_toolkits.mplot3d import Axes3D
```

```
#Constants
#Earth-Moon-Sun Constants
G = 6.67*10**(-11)
m1 = 1.989*10**30 #sun
m2 = 5.972*10**24 #earth
m3 = 7.348*10**22 #moon
#Lemniscate Constants
#G = 1
#m1 = 1
#m2 = 1
#m3 = 1
#Burrau's Problem Constants
#G = 1
#m1 = 3
#m2 = 4
#m3 = 5
```

```
def der_state(t, state):
"""compute the derivative of the given state"""
#Vectors and Magnitudes
r1 = np.array([state[0], state[2], state[4]]) #x1, y1, z1
r2 = np.array([state[6], state[8], state[10]]) #x2, y2, z2
r3 = np.array([state[12], state[14], state[16]]) #x3, y3, z3
r21 = r2 - r1
r31 = r3 - r1
r12 = r1 - r2
r32 = r3 - r2
r13 = r1 - r3
r23 = r2 - r3
R21 = np.linalg.norm(r21)
R31 = np.linalg.norm(r31)
R12 = np.linalg.norm(r12)
R32 = np.linalg.norm(r32)
R13 = np.linalg.norm(r13)
R23 = np.linalg.norm(r23)
```

```
#System of Differential Equations
der = np.zeros_like(state)
der[0] = state[1] #a0x’ = a1x
der[2] = state[3] #a0y’ = a1y
der[4] = state[5] #a0z’ = a1z
der[6] = state[7] #b0x’ = b1x
der[8] = state[9] #b0y’ = b1y
der[10] = state[11] #b0z’ = b1y
der[12] = state[13] #c0x’ = c1x
der[14] = state[15] #c0y’ = c1y
der[16] = state[17] #c0z’ = c1z
#The Actual Equations
#Mass 1
der[1] =
G*((m2*(state[6]-state[0])/R21**3)+(m3*(state[12]-state[0])/R31**3))
der[3] =
G*((m2*(state[8]-state[2])/R21**3)+(m3*(state[14]-state[2])/R31**3))
der[5] =
G*((m2*(state[10]-state[4])/R21**3)+(m3*(state[16]-state[4])/R31**3))
#Mass 2
der[7] =
G*((m1*(state[0]-state[6])/R12**3)+(m3*(state[12]-state[6])/R32**3))
der[9] =
G*((m1*(state[2]-state[8])/R12**3)+(m3*(state[14]-state[8])/R32**3))
der[11] =
G*((m1*(state[4]-state[10])/R12**3)+(m3*(state[16]-state[10])/R32**3))
#Mass 3
der[13] =
G*((m1*(state[0]-state[12])/R13**3)+(m2*(state[6]-state[12])/R23**3))
der[15] =
G*((m1*(state[2]-state[14])/R13**3)+(m2*(state[8]-state[14])/R23**3))
der[17] =
G*((m1*(state[4]-state[16])/R13**3)+(m2*(state[10]-state[16])/R23**3))
return der
```

```
#tf for Earth-Sun-Moon System
tf = 60000000 simulation for tf seconds
#Lemniscate
#tf = 30 #simulation for tf seconds
#Burrau’s Problem Initial Conditions
#tf = 20 #simulation for tf seconds
n = 1000 #number of evaluation points
dt = tf/n
T = np.linspace(0.0, tf, n+1)
```

```
#Initial State
```

```
#([xa, vxa, ya, vya, za, vza, xb, vxb, yb, vyb, zb, vzb, xc, vxc, yc, vyc, zc,
vzc])
#Earth-Sun-Moon Initial Conditions
state0 = [0, 0, 0, 0, 0, 0, 149.6*10**9, 0, 0, 30000, 0, 0, 1.49984*10**11, 0, 0,
33000, 0, 0]
#Lemniscate Initial Conditions
#state0 = [0.97000436, 0.4662036850, -0.24308753, 0.4323657300, 0, 0, 0,
-0.93240737, 0, -0.86473146, 0, 0, -0.97000436, 0.4662036850, 0.24308753,
0.4323657300, 0, 0]
#Burrau's Problem Initial Conditions
#state0 = [1.0, 0.0, 3.0, 0.0, 0.0, 0.0, -2.0, 0.0, -1.0, 0.0, 0.0, 0.0, 1.0, 0.0, -1.0,
0.0, 0.0, 0.0]
```

```
#Solve_IVP
sol = integrate.solve_ivp(der_state, (0, tf), state0, t_eval=T, method="RK45")
aX = sol.y[0]
ay = sol.y[2]
az = sol.y[4]
bx = sol.y[6]
by = sol.y[8]
bz = sol.y[10]
cx = sol.y[12]
cy = sol.y[14]
cz = sol.y[16]
%matplotlib notebook
fig=plt.figure()
#Earth-Sun-Moon System
ax = fig.add_subplot(111, projection='3d', autoscale_on=False,
xlim=(-13*10**10,13*10**10), ylim=(-13*10**10,13*10**10),
zlim=(-13*10**10,13*10**10))
#Lemniscate and Burrau
#ax = fig.add_subplot(111, projection='3d', autoscale_on=False, xlim=(-2,2),
ylim=(-2,2), zlim=(-2,2))
ax.grid()
#Earth-Mass-Sun System
mass1, = ax.plot([], [], [], 'yo', markersize=20)
mass2, = ax.plot([], [], [], 'bo', markersize=8)
mass3, = ax.plot([], [], [], 'ro', markersize=4)
#Lemniscate and Burrau #mass1, = ax.plot([], [], [], 'bo', markersize=5)
#mass2, = ax.plot([], [], [], 'bo', markersize=5)
#mass3, = ax.plot([], [], [], 'bo', markersize=5)
time_template = 'time ='
time_text = ax.text(0.05, 0.9, 1, '', transform=ax.transAxes)
def animate(i):
mass1.set_data(aX[i],ay[i])
mass1.set_3d_properties(az[i])
```

```
domain = [aX[i-1],aX[i]]
ranges = [ay[i-1],ay[i]]
space = [az[i-1],az[i]]
plt.plot(domain,ranges,space,'r')
mass2.set_data(bx[i],by[i])
mass2.set_3d_properties(bz[i])
domain = [bx[i-1],bx[i]]
ranges = [by[i-1],by[i]]
space = [bz[i-1],bz[i]]
plt.plot(domain,ranges,space,'g')
mass3.set_data(cx[i],cy[i])
mass3.set_3d_properties(cz[i])
domain = [cx[i-1],cx[i]]
ranges = [cy[i-1],cy[i]]
space = [cz[i-1],cz[i]]
plt.plot(domain,ranges,space,'b')
time_text.set_text(time_template + ':4.1f'.format(i*dt) + 's')
return mass1, mass2, mass3, time_text
ani = animation.FuncAnimation(fig, animate, frames=np.arange(1, len(T)),
interval=1, blit=True)
#This bit saves the animation to whatever folder this file is in
#Writer = animation.writers['ffmpeg']
#writer = Writer(fps=24, metadata=dict(artist='Me'), bitrate=1800)
#ani.save('ESM.mp4', writer=writer, dpi=400)
```

Works Cited

Cook, Alan. “Success and Failure in Newton’s Lunar Theory.” OUP Academic, Oxford University Press, 1 Dec. 2000, academic.oup.com/astrogeo/article/41/6/6.21/225623.

Dyck, Joel A. “Periodic Solutions to the n-Body Problem.” Combinatorial Math, Sept. 2015, www.combinatorialmath.ca/Students/DyckThesis.pdf.

“Eccentric Anomaly.” Wikipedia, Wikimedia Foundation, 10 May 2019, en.wikipedia.org/wiki/Eccentric_anomaly.

Frank, Juhan. “PHYS 7221 - The Three-Body Problem.” Phys.lsd.edu, 11 Oct. 2006, www.phys.lsu.edu/faculty/gonzalez/Teaching/Phys7221/ThreeBodyProblem.pdf.

Hayes, Brian. “The 100-Billion-Body Problem.” American Scientist, 28 Apr. 2018, www.americanscientist.org/article/the-100-billion-body-problem.

“Jacobi and His Constant - The 3-Body Problem.” Gereshes, 6 Mar. 2019, gereshes.com/2018/11/26/jacobi-and-his-constant-the-3-body-problem/.

“Kepler’s Equation.” Wikipedia, Wikimedia Foundation, 18 Apr. 2019, en.wikipedia.org/wiki/Kepler%27s_equation.

Klus, Helen. “Newton’s Theory of Gravity.” The Star Garden, 6 Aug. 2017, www.thestargarden.co.uk/Newtons-theory-of-gravity.html.

Krizek, Michael. “Numerical Experience with the Three-Body Problem.” Journal of Computational and Applied Mathematics, North-Holland, 5 Apr. 2000, www.sciencedirect.com/science/article/pii/0377042795000674.

M, Zack. “What Are the Relative Positions of the Sun, Earth, and Moon at the Times of Lunar and Solar Eclipses, as Well as New, First-Quarter, Full, and Last-Quarter Phases of the Moon? — Socratic.” Socratic.org, 17 Feb. 2016, socratic.org/questions/what-are-the-relative-positions-of-the-sun-earth-and-moon-at-the-times-of-lunar-.

“Mean Anomaly.” Wikipedia, Wikimedia Foundation, 17 Jan. 2019, en.wikipedia.org/wiki/Mean_anomaly.

Musielak, Dora E. “Euler: Genius Blind Astronomer Mathematician.” Arxiv, arxiv.org/ftp/arxiv/papers/1406/1406.7397.pdf.

Ovenden, M W, and A E Roy. “On The Use of The Jacobi Integral of The Restricted Three-Body Problem.” 1961MNRAS.123....1O Page 1, 14 Oct. 1960, adsabs.harvard.edu/full/1961MNRAS.123....1O.

Ozaki, Hiroshi. “Determination of Motion From Orbit in The Three-Body Problem.” Research Gate, 2011, www.researchgate.net/profile/Hiroshi_Ozaki/publication/236966171_Determination_of_motion_from_orbit_in_the_three-body_problem/links/54ca3ac40cf2c70ce5219e22/Determination-of-motion-from-orbit-in-the-three-body-problem.pdf.

Ragsteadt, Michael. “FROM ORDER TO CHAOS: THE PRIZE COMPETITION IN HONOUR OF KING OSCAR II.” FROM ORDER TO CHAOS: THE PRIZE COMPETITION IN HONOUR OF KING OSCAR II — IML, 2016, www.mittag-leffler.se/library/henri-poincare.

Revolvy, LLC. “‘Jacobi Integral’ on Revolvy.com.” Revolvy, 2016, www.revolvy.com/page/Jacobi-integral.

Worthington, Joachim. “A Study of the Planar Circular Restricted Three Body Problem and the Vanishing Twist.” Math.usyd, Oct. 2012, www.maths.usyd.edu.au/u/joachimw/thesis.pdf.

“‘Pythagorean Three Body Problem’ - Need Some Points from an Accurate Solution for Comparison.”
Space Exploration Stack Exchange, 2016, space.stackexchange.com/questions/15364/pythagorean-three-body-problem-need-some-points-from-an-accurate-solution-fo.