Physics 215 - Lab 6: Planetary Motion

Arman Rasheed

November 2nd, 2020

1 Introduction

In this exercise we will examine the motion of different planets which are in orbit around the sun.

1.1 Part 1: The Two Body Problem

The motion of the planets can be modeled with Newton's Law of Gravitation. Looking at a simple case of the Earth traveling around the Sun, the equation for the forces attracting the two planetary masses are:

$$F_g = \frac{GM_S M_E}{r^2} \tag{1}$$

where G is the gravitational constant, M_S is the mass of the Sun, and M_E is the mass of the Earth. When attempting to put this equation of motion in terms of a differential equation which can be numerically solved, it is much simpler to decide that the motion of the Sun in comparison to that of the Earth is negligible. This leaves us with the following differential equations for the x and y components of motion:

$$\frac{d^2x}{dt^2} = \frac{F_{gx}}{M_E} \tag{2}$$

$$\frac{d^2y}{dt^2} = \frac{F_{gy}}{M_E} \tag{3}$$

Furthermore, there can be useful geometric identities in these kinds of motion, one of which is especially useful for this lab:

$$r = \sqrt{x^2 + y^2} \tag{4}$$

It is also useful to understand some properties of ellipses since many, if not all, planets do not orbit around the sun in a perfect circle, but rather have a kind of ovular shape to their motion. Most orbits are characterized by the semi-major axis a, which is half the distance of the longest radius of the ellipse, and the eccentricity e, which is a numerical representation of how ovular the ellipse is from a circle. There is also a semi-minor axis which corresponds to the smallest radius of the curve. These variables can be represented as follows:

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{5}$$

$$b = a\sqrt{1 - e^2} \tag{6}$$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{7}$$

Combining the above information allows us to set up a system of first order differential equations which can easily be numerically solved for:

$$\frac{dvx}{dt} = -\frac{GM_S x}{r^3} \tag{8}$$

$$\frac{dvy}{dt} = -\frac{GM_S y}{r^3} \tag{9}$$

$$\frac{dx}{dt} = v_x \tag{10}$$

$$\frac{dy}{dt} = v_y \tag{11}$$

1.2 Part 2: Kepler's Second Law

This next part of the lab asks us to prove Kepler's second law which states that for a set of triangles made from discretizing an ellipse with some $\Delta\theta$, the areas of each of the triangles are going to be equal. The area of these triangles is given by the equation:

$$dA = -\frac{1}{2} r * r d\theta \tag{12}$$

We were also asked to model the energy: kinetic, potential, and total for the path of motion of the planet as well as show the conservation of angular momentum through this system, which could all be calculated with the points of the simulated motion graph.

1.3 Part 3: Triple Body Problem

The lab can finally be made more intricate through adding the influence of another planetary body as follows:

$$r_1(t) = \sqrt{x_1(t)^2 + y_1(t)^2} \qquad r_2(t) = \sqrt{x_2(t)^2 + y_2(t)^2}$$
$$r_{12}(t) = \sqrt{(x_1(t) - x_2(t))^2 + (y_1(t) - y_2(t))^2}$$

$$v_{1x}(t + \Delta t) = v_{1x}(t) - \frac{4\pi^2 x_1(t)}{r_1(t)^3}\Delta t - \frac{4\pi^2\left(\frac{m_2}{M}\right)(x_1(t) - x_2(t))}{r_{12}(t)^3}\Delta t$$
$$v_{2x}(t + \Delta t) = v_{2x}(t) - \frac{4\pi^2 x_2(t)}{r_2(t)^3}\Delta t - \frac{4\pi^2\left(\frac{m_1}{M}\right)(x_2(t) - x_1(t))}{r_{12}(t)^3}\Delta t$$
(Similarly for v_y components)

Where $r1(t)$ is the radial distance from the sun for Mars, $r2(t)$ is for Jupiter, and $r12(t)$ is the distance between the two planetary masses. The velocities and different components of position are similarly labeled 1 and 2 for Mars and Jupiter respectively.

2 Numerical Approach

2.1 Problem 1

The numerical solution for this first order initial value problem is derived from the Taylor Expansion of the equation:

$$v(t+\delta t) = v(t) + \frac{dv}{dt}\delta t + \frac{1}{2}\frac{d^2v}{dt^2}\delta t^2 + ... \tag{13}$$

With a very small value for δt we can remove the need to factor in higher order terms from the Taylor Expansion and obtain the following approximation:

$$v(t+\delta t) \approx v(t) + \frac{dv}{dt}\delta t \tag{14}$$

Substituting equation 8 into this equation results in the Euler numerical method solution with an added step for the differential equation for position which makes the result more accurate.

$$v_x(t+\delta t) \approx v_x(t) - \frac{GM_s x(t)}{r^3}dt = v_x(t) - \frac{4\pi^2 x(t)}{r^3}dt \tag{15}$$

$$x(t+\delta t) \approx x(t) + v_x(t+\delta t)\delta t \tag{16}$$

Repeating this for the other equations results in:

$$v_y(t+\delta t) \approx v_y(t) - \frac{4\pi^2 y(t)}{r^3}dt \tag{17}$$

$$y(t+\delta t) \approx y(t) + v_y(t+\delta t)\delta t \tag{18}$$

2.2 Problem 2

Instead of having to manually calculate each segment's $\Delta\theta$ I simply used the points I already calculated and added the areas of a series of almost approximate triangles which spanned the whole ellipse. This process can roughly be modeled as follows:

$$\Delta A \approx \frac{1}{2}r(t) * r(t)(\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}) \tag{19}$$

2.3 Problem 3

The equations from the image in section 1.3 are already prepared numerically by the Euler-Cromer method previously discussed above for problem 1. All I needed to do was to translate them into code for the computer to process.

3 Pseudo-code

- Declare variables and arrays, and initialize initial values:

  Problem 1 and 2:

```
const int n = 1000;  //number of iterations
float dt = .01;  //time step size for calculations
float pi = 3.14159 //constant
float a0 = .5;  //initial displacement
float vhigh = 0.0;
float vlow = 1000.0;
float dist = 0.0; //necessary initial temp values for calculations
float t[n], x[n], y[n], r[n], vx[n], vy[n],
v[n], E[n], k[n], p[n], m[n];
float A[N];  //arrays for plotting and calculations
t[0] = 0; x[0] = 1; y[0] = 0;
r[0] = sqrt(x[0]*x[0] + y[0]*y[0]);
vx[0] = 0; vy[0] = 2 * pi;
v[0] = sqrt(vx[0]*vx[0] + vy[0]*vy[0]);
k[0] = .5 * mass * v[0]*v[0];
p[0] = (-4 * pi * pi * mass) / (r[0]*r[0]);
E[0] = k[0] + p[0];
m[0] = mass * r[0] * v[0];  //initial values
```

  Problem 3:

```
const int n = 15000; float dt = .001; //iterations & step
float pi = TMath::Pi(); float mars = 6.6 * pow(10, 23);
float jupe = 1.9 * pow(10, 30); float sun = 2 * pow(10, 30); //constants
float t[n], x[n], y[n], r[n], vx[n], vy[n],
x2[n], y2[n], r2[n], vx2[n], vy2[n], r12[n]; //arrays used
t[0] = 0; x[0] = 1.66136; y[0] = 0;
r[0] = sqrt(x[0]*x[0] + y[0]*y[0]);
vx[0] = 0; vy[0] = 4.6425; x2[0] = 5.4496;
y2[0] = 0; r2[0] = sqrt(x2[0]*x2[0] + y2[0]*y2[0]);
vx2[0] = 0; vy2[0] = 2.626;
r12[0] = sqrt((x[0] - x2[0])*(x[0] - x2[0])
+ (y[0] - y2[0])*(y[0] - y2[0])); //initial values
```

- Iterative calculations:

  Problem 1:

```
//Euler-Cromer method uses previously calculated values
//to approximate a numerical solution. Examples:
vx[i + 1] = vx[i] - (((4 * pow(pi, 2)) * x[i]) / (pow(r[i], 3))) * dt;
x[i + 1] = x[i] + vx[i + 1] * dt;
```

```
vy[i + 1] = vy[i] - (((4 * pow(pi, 2)) * y[i]) / (pow(r[i], 3))) * dt;
y[i + 1] = y[i] + vy[i + 1] * dt;
//Next had to determine where the apogee,
//perigee, and co-vertices were to determine the major and minor axis.
if(v[i+1] is the highest value in array) {
    apogee = i + 1;
} if(v[i+1] is the lowest value in array) {
    perigee = i + 1;
} if(vy[i+1] is zero) {
    if(y is positive) {
        save the location of the covertex here
    } else {
        here's the other covertex
    }
}
```

Problem two just added on some other equations into the loop previously used, as well as calculated the area of the segments in another loop:

```
//Energy calculations in the loop already used:
k[i + 1] = .5 * mass * v[i + 1]*v[i + 1];
p[i + 1] = (-4 * pi * pi * mass) / (r[i + 1]*r[i + 1]);
E[i + 1] = k[i + 1] + p[i + 1];
m[i + 1] = mass * r[i + 1] * v[i + 1];
//New loop for Area:
for(int j = 0; j < n - 100; j++) { //had to have proper
//distance between pts.
calculated distance with formula here
calculated area for the proper index in the area array,
while simultaneously creating a numbered array for plotting
}
```

Problem 3 simply added onto the Euler approximation in problem one:

```
\\Here are the correct changes for velocity
vx[i + 1] = vx[i] - ((((4 * pow(pi, 2)) * x[i]) / (pow(r[i], 3))) * dt)
- (((4 * pow(pi, 2) * (jupe / sun))/(pow(r12[i], 3))) * dt);
vy[i + 1] = vy[i] - ((((4 * pow(pi, 2)) * y[i]) / (pow(r[i], 3))) * dt)
- (((4 * pow(pi, 2) * (jupe / sun))/(pow(r12[i], 3))) * dt);
vx2[i + 1] = vx2[i] - ((((4 * pow(pi, 2)) * x2[i]) / (pow(r2[i], 3))) * dt)
- (((4 * pow(pi, 2) * (mars / sun))/(pow(r12[i], 3))) * dt);
vy2[i + 1] = vy2[i] - ((((4 * pow(pi, 2)) * y2[i]) / (pow(r2[i], 3))) * dt)
- (((4 * pow(pi, 2) * (mars / sun))/(pow(r12[i], 3))) * dt);
```

- Visualization:
  Plotted and visualized necessary graphs with primarily the TGraph function.

4 Results

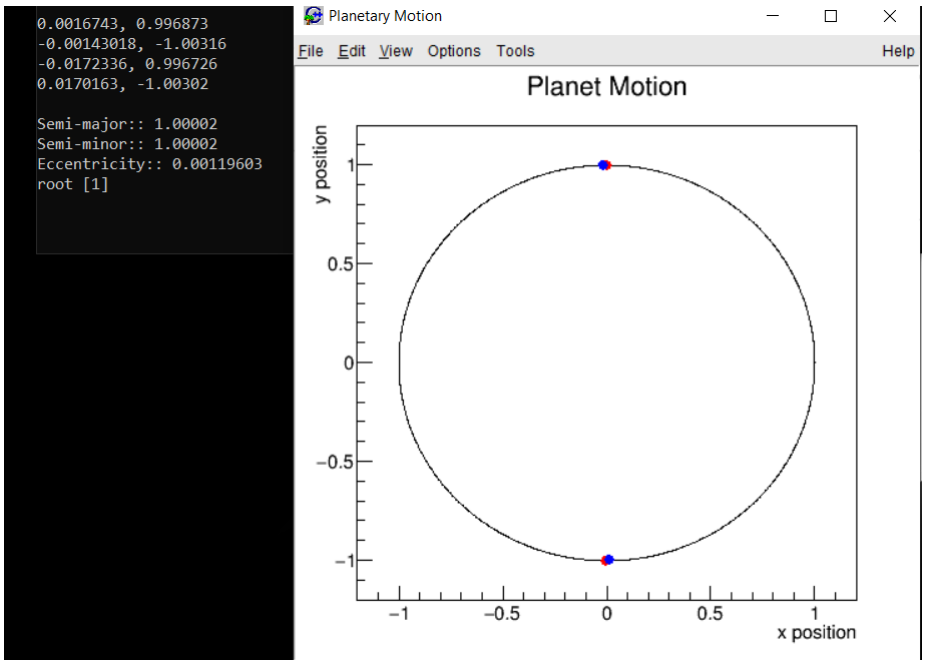

Figure 1: Earth's trajectory around sun with the pertinent data on the left side.

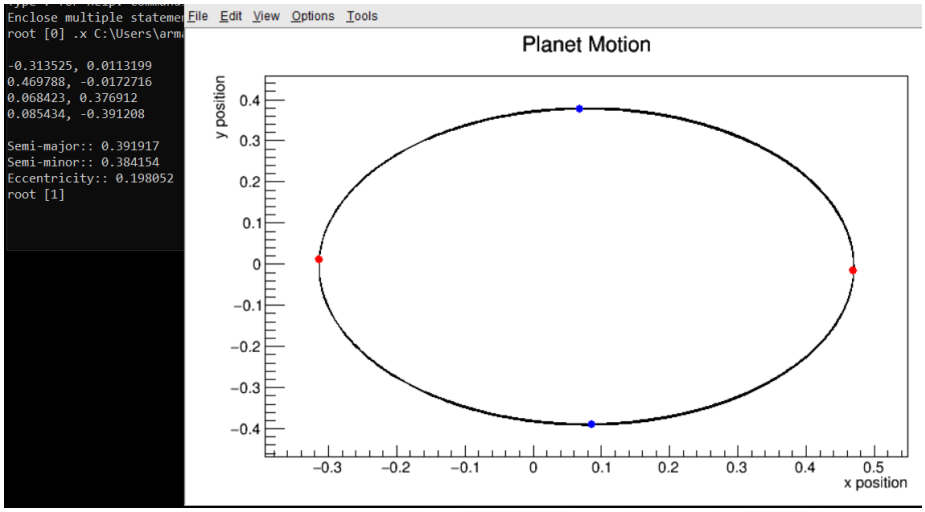

Figure 2: Trajectory for Mercury. Markers are the vertices that my program approximated.

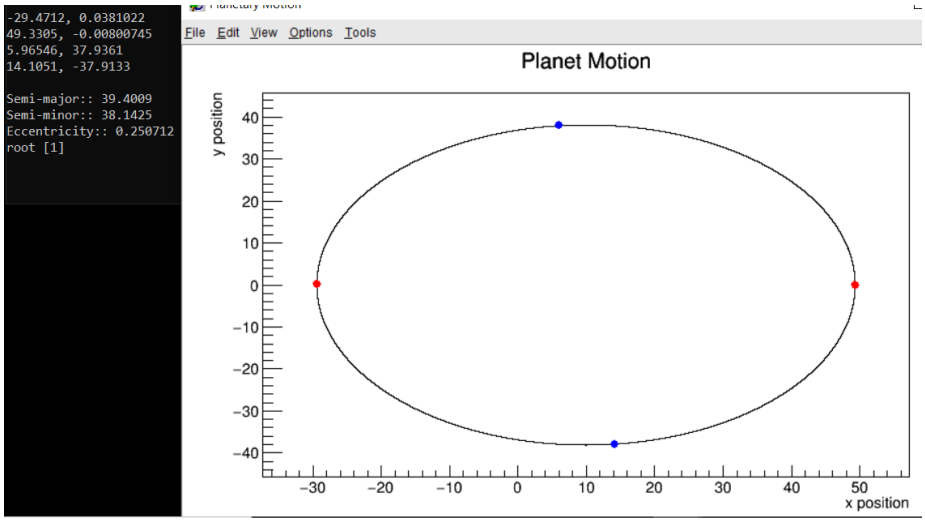

Figure 3: Trajectory and vertices for Pluto.

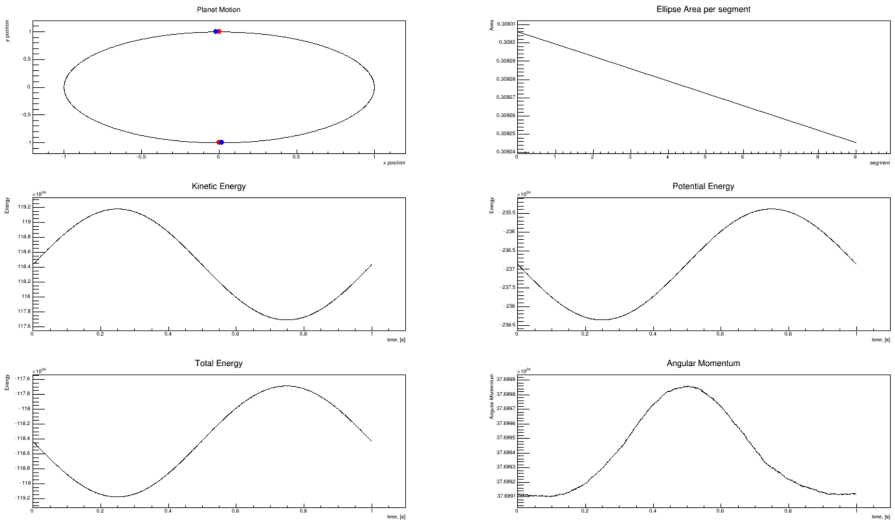

Figure 4: Energy and Area calculations for a circular trajectory (Earth)

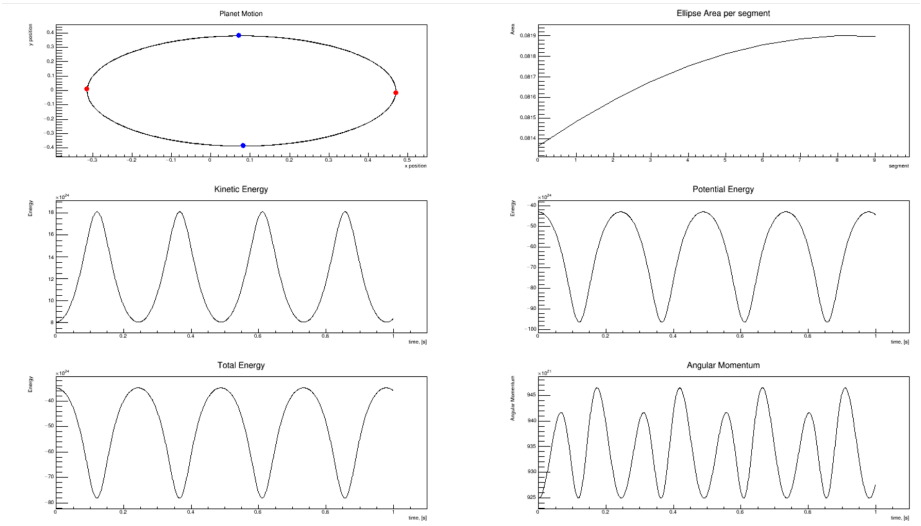

Figure 5: Energy and Area calculations for an elliptical trajectory (Mercury)

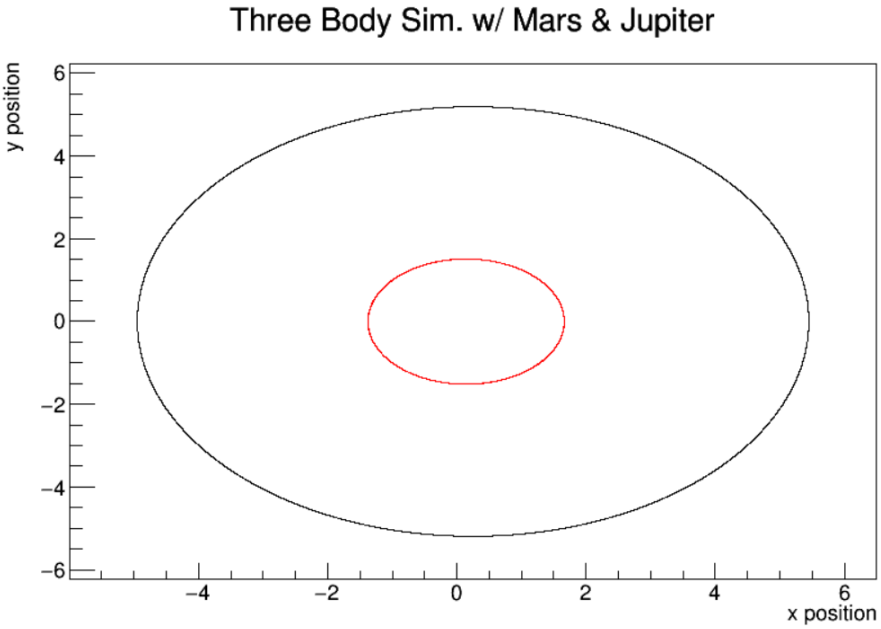

Figure 6: Normal Triple body simulation for Mars and Jupiter

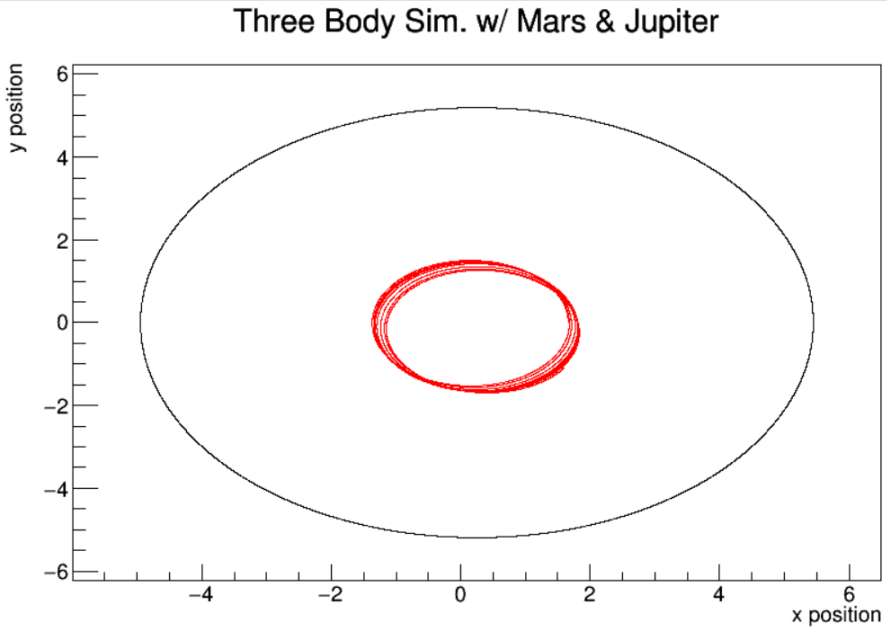

Figure 7: Triple body simulation for Mars and Jupiter with 1000x more mass for Jupiter

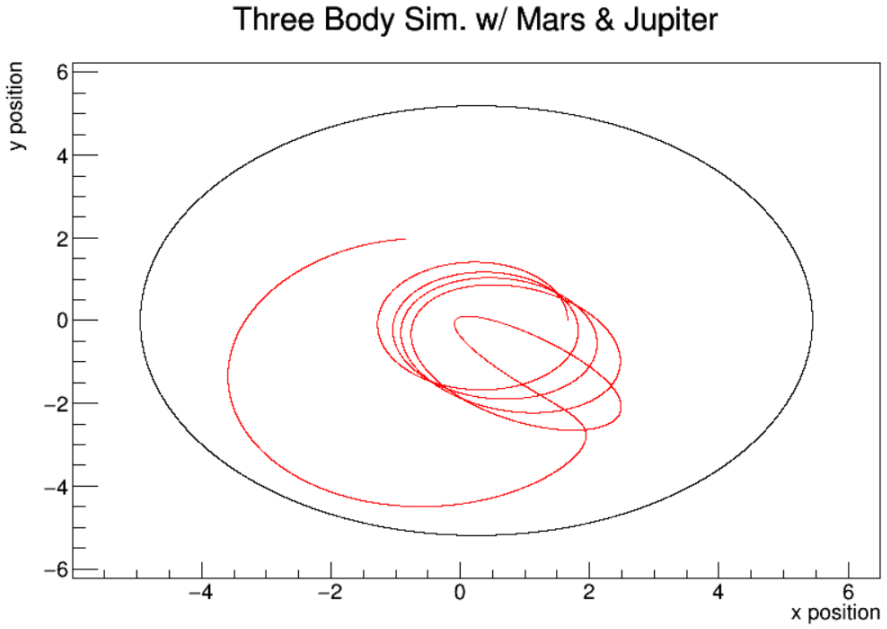

Figure 8: Triple body simulation for Mars and Jupiter with 10000x more mass for Jupiter

5 Discussion

Problem 1: The approximations my program calculated compared very closely with the data given in lecture. Limitations however could be found in the manner in which I arrived at the values for the co-vertices since I am estimating where $vy = 0$ in a margin with a non negligible size. The vertex not appearing in that margin can make my code simply use the index zero for a vertex which can be wrong in most scenarios. Furthermore, because the Earth has such a circular trajectory, my method of determining vertices can confuse the marker location for these kinds of paths of motion.

Problem 2: Unlike the example given in lecture, there seems to be a reasonable amount of variance in the upper and lower bounds for the total energy of the trajectory I chose. I don't know whether that's because of an error I made in calculating the kinetic and potential energies, or just due to the specific way in which my plot bounds the graph. Furthermore, there is minuscule variance in the vertical bounds of the angular momentum graph, which suggests that the program accurately simulated the conservation of momentum. Lastly, and again, the bounds of the graph for the areas I calculated differ by negligible values, meaning that as a whole, the areas between the elliptical segments are approximately equal, and that Kepler's second law hasn't been violated by my simulation.

Problem 3: It was interesting to notice how the path of Mars became increasingly erratic as the mass of Jupiter increased exponentially. Of course, the path of Jupiter didn't change whatsoever due to the magnitude of force from Mars, which makes complete sense given the vast difference in the planets' masses.