PHYS 325 Assignment 7:
The Solar System

Author: A. J. Ogle
Date: May 8th, 2015
PHYS 325 Computational Physics

1 Problem 4.1: Stability of Earth Orbits

In this problem, the effect of varying values for the time step was explored.

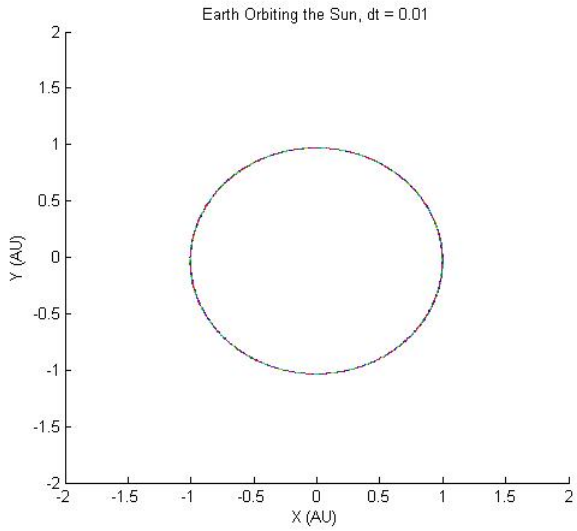

Figure 1: Orbit of the Earth about the sun using the Euler-Cromer method and $dt = 0.01$ and total time $ft = 10\pi$.

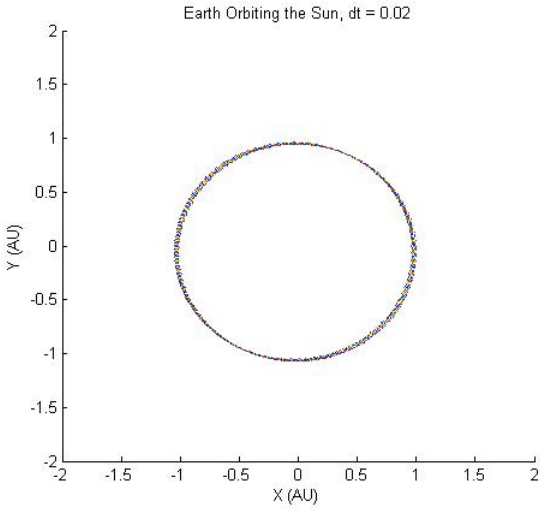

Figure 2: Orbit of the Earth about the sun using the Euler-Cromer method and $dt = 0.02$ and total time $ft = 10\pi$.

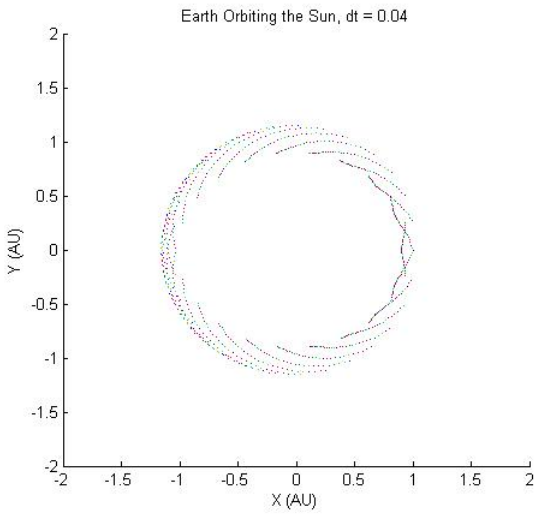

Figure 3: Orbit of the Earth about the sun using the Euler-Cromer method and $dt = 0.04$ and total time $ft = 10\pi$.

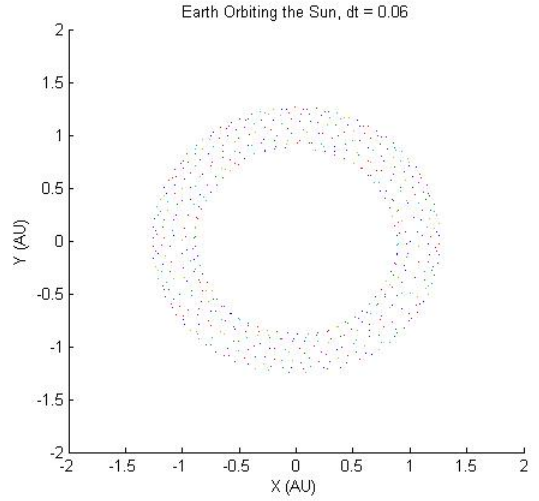

Figure 4: Orbit of the Earth about the sun using the Euler-Cromer method and $dt = 0.06$ and total time $ft = 10\pi$.

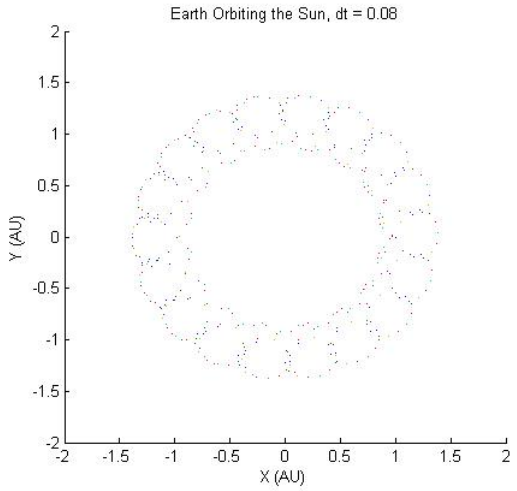

Figure 5: Orbit of the Earth about the sun using the Euler-Cromer method and $dt = 0.08$ and total time $ft = 10\pi$.

Overall, it can be seen that as dt becomes increasingly greater than 0.01, the results begin to look less and less like a stable orbit. For $dt = 0.08$, the orbit appears to be an orbit within an orbit as the earth travels around the sun. Thus it can be seen that the scale of dt must be considered for the individual problem, as a dt that is relatively "too large" will result in an accumulation of errors as the simulation runs. This results in a simulation not representative of the actual orbit of the planet.

2 Problem 4.3: Kepler's 2nd Law

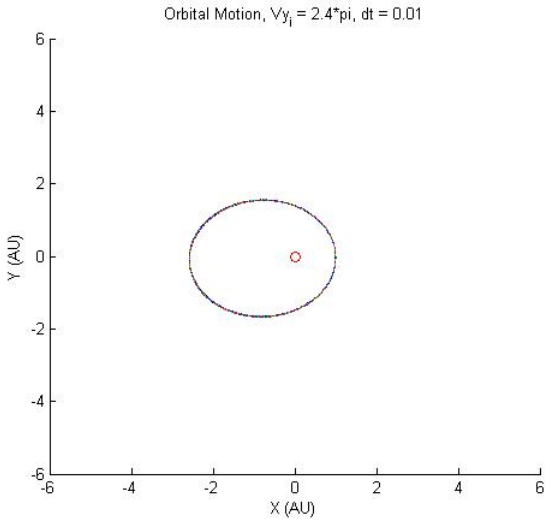

Figure 6: Elliptical orbits are explored in this model. Here, the initial velocity is 2.4π with $dt = 0.01$. The area of the triangles created by each time step about the star (indicated by the dot) were found to be the same with a standard deviation of $Std(A) = 2.4507e - 11$.

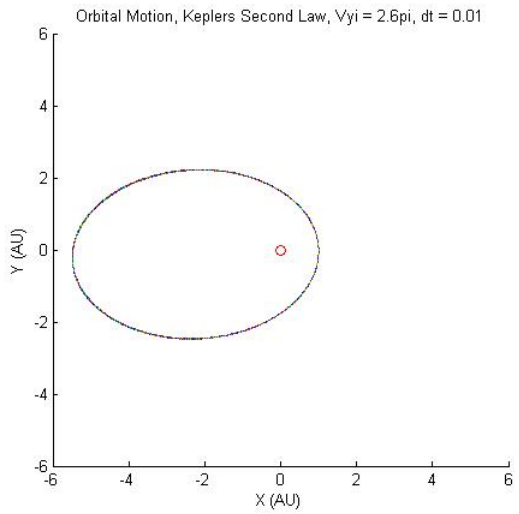

Figure 7: Elliptical orbits are explored in this model. Here, the initial velocity is 2.6π with $dt = 0.01$. The area of the triangles created by each time step about the star (indicated by the dot) were found to be the same with a standard deviation of $Std(A) = 3.0324e - 17$.

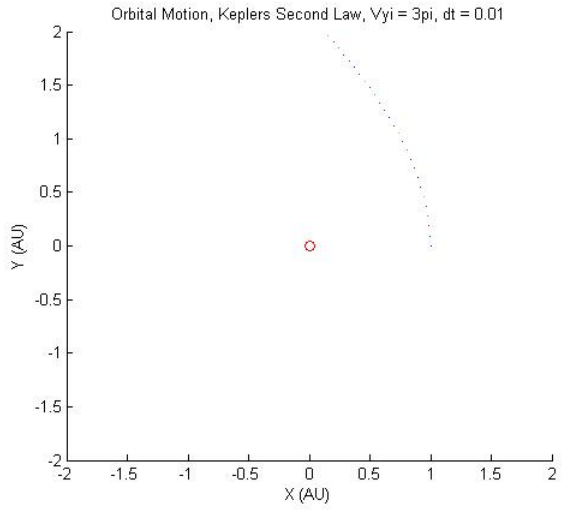

Figure 8: Elliptical orbits are explored in this model. Here, the initial velocity is 3.0π with $dt = 0.01$. The area of the triangles created by each time step about the star (indicated by the dot) were found to be the same with a standard deviation of $Std(A) = 7.9967e - 14$.

Kepler's 2nd Law states that the area swept over by an orbiting body about a point will be the same for any points about the orbit over the same time interval. As can be seen from **??**, **??**, and **??**, Kepler's 2nd law holds. The areas of the triangles created by each successive time step are all the same, with a standard deviation between them of essentially zero. Even for the case presented by **??**, in which the planet was actually slung out of its orbit into the deep reaches of space, Kepler's 2nd Law was still found to hold.

3 Problem 4.8: Kepler's 3rd Law

Kepler's 3rd Law predicts a universal constant relating the orbits of the satellites about a star and the semi-major axes of their orbits. In a more mathematical formalism, $\frac{T^2}{a^3} = k$. Optimally, the constant k would be equivalent to 1, showing that T^2 and a^3 are perfectly proportional to each other.

Planet	Radius (Earth Units)	Mass (Earth Units)	$\frac{T^2}{a^3}$	Percent Diff. From Expected
Venus	0.72	0.82	1.2137	21.37
Earth	1.00	1	0.9962	0.38

From **??**, it can be seen that the values produced from the model give a proportionality between T^2 and a^3 that is close to 1. Some issues with calculating these values for other planets has to do with finding a stable orbit for the different planets. If a stable orbit cannot be found, then the proportionality constant cannot be derived. For This problem, the proportionality for Jupiter and Saturn was difficult to calculate because a stable orbit could not yet be found.

Comparing to values generated from similar models (Giordano06), the values for Earth agree with only a -0.18 percent difference. For the values of Venus, there was a 21.61 percent difference between those listed in Giordano and those from the model shown in **??**.

4 Problem 4.13: Binary Stars

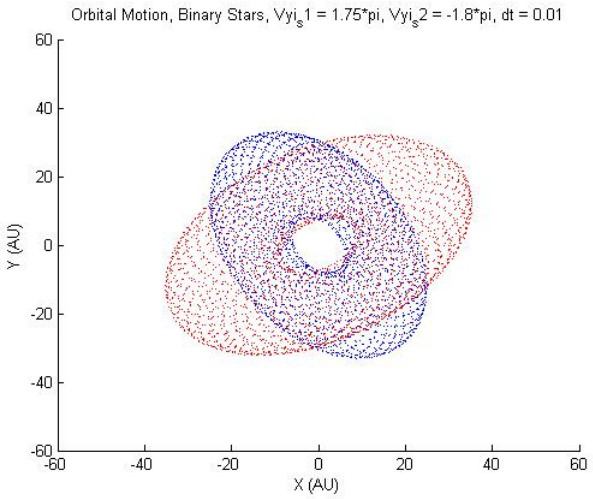

Figure 9: The orbits of two stars rotating about each other for a time interval of $ft = 100\pi$. The mass of the stars is the same, $m = 331.5ke$, where mass is measured in units of 1 earth mass ($5.972e24kg$ is the mass of the earth). The distance is also measured in units relative to the Earth's distance from the sun, astronomical units, with $1AU = 149,597,871km$.

A stable orbit for two stars orbiting each other was found. The stars have the same mass, with $m = 331.5ke$, measured with 1 unit of mass equaling the mass of the Earth. For a planet orbiting the binary star system, it must be sufficiently far away from the stars such that their mass appears to be a single point source, or that of a single star. Otherwise, the oscillation of the binary star system produces a gravitational field which is constantly changing, making it difficult for planets to orbit nearby. A solution for a planet orbiting a binary star system was attempted, but failed due to the sensitivity required to find a stable orbit. This sensitivity is demonstrated in **??**. For values of $p_{mf} \approx 10$, the trajectory of the planet was found to begin curling about the binary star system, but a stable solution was not found.

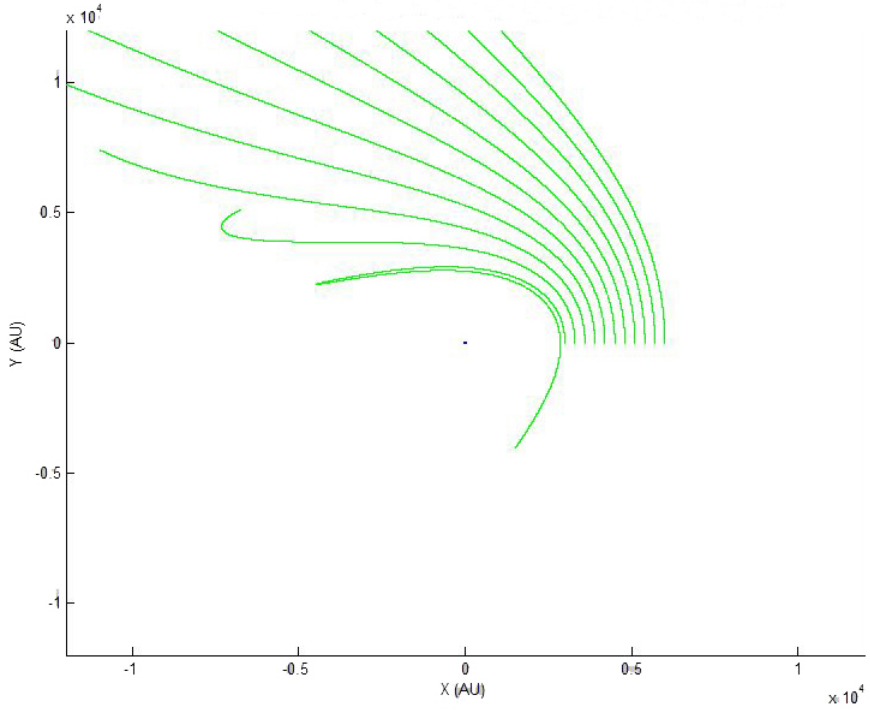

Figure 10: This shows the attempts at finding a stable orbit for the planet about the binary star system. A multiplicative factor $300 * p_{mf}$ was used to control the distance of the planet from the star system, with p_{mf} varying from 20 to 10 in increments of 1. The planet has a mass of $m_p = 1e$, so its effect on the binary stars is very small.

5 Conclusions

Overall, although the Runge-Kutta method has a longer computational time, it is more accurate in representing the motion of the SHO and conserves energy over any time interval. For a faster computation, the Euler-Cromer method can be used, but energy is only conserved over specific time intervals for oscillatory motion. For most applications with projectile motion, the Euler-Cromer method is more than enough. For oscillatory motion, it suffices. In general, computational accuracy must be considered prior to computational efficiency. If the computational approach does not produce a simulation which reasonably suits the physical situation, then its efficiency is irrelevant.

References

[1] Giordano, Nicholas J., and Hisao Nakanishi. "3. Oscillatory Motion and Chaos" Computational Physics. Upper Saddle River, NJ: Pearson/Prentice Hall, 2006. N. pag. Print.

6 MATLAB code

```
%%problem 4.1

clear all;

x(1) = 1;
y(1) = 0;
v_x(1) = 0;
v_y(1) = 2.4*pi;
dt = 0.008;
ft = 10*pi;
t = [0:dt:ft+dt];

for i = 1:length(t)+dt
    r = sqrt(x(i)^2 + y(i)^2);
    v_x(i+1) = v_x(i) - (((4*pi^2)*x(i))/r^3)*dt;
    v_y(i+1) = v_y(i) - (((4*pi^2)*y(i))/r^3)*dt;
    x(i+1) = x(i) + v_x(i+1)*dt;
    y(i+1) = y(i) + v_y(i+1)*dt;
    scatter(x(i), y(i), 0.5);
    hold on;
end

n = 6;
m = n;

title('Earth Orbiting the Sun, Vy_i = 2.4*pi, dt = 0.008');
axis([-n,m,-n,m]);
xlabel('X (AU)');
ylabel('Y (AU)');


%%problem 4.2

clear all;

x(1) = 1;
y(1) = 0;
v_x(1) = 0;
v_y(1) = 3*pi;
dt = 0.01;
ft = 10*pi;
t = [0:dt:ft+dt];
r(1) = sqrt(x(1)^2 + y(1)^2);
A(1) = 0;

for i = 1:length(t)
```

```
    r = sqrt(x(i)^2 + y(i)^2);
    v_x(i+1) = v_x(i) - (((4*pi^2)*x(i))/r^3)*dt;
    v_y(i+1) = v_y(i) - (((4*pi^2)*y(i))/r^3)*dt;
    x(i+1) = x(i) + v_x(i+1)*dt;
    y(i+1) = y(i) + v_y(i+1)*dt;

    %finding the area of the triangles created by each dt using Heron's
    %formula
    r(i) = sqrt(x(i)^2 + y(i)^2);
    r(i+1) = sqrt(x(i+1)^2 + y(i+1)^2);
    a = r(i);
    b = r(i+1);
    c = sqrt((x(i+1)-x(i))^2 + (y(i+1)-y(i))^2);
    p = (r(i)+r(i+1)+c)/2;
    A(i) = sqrt(p*(p-a)*(p-b)*(p-c));

    scatter(x(i), y(i), 0.5);
    hold on;
end

ave_A = mean(A);

for i=1:length(A)
   total_diff = (ave_A-A(i))^2;
   variance_A = total_diff/length(A);
   std_dev_A = sqrt(variance_A);
end

n = 2;
m = n;

x = 0;
y = 0;
plot(x,y,'ro');
title('Orbital Motion, Keplers Second Law, Vyi = 3pi, dt = 0.01');
axis([-n,m,-n,m]);
xlabel('X (AU)');
ylabel('Y (AU)');
disp(std_dev_A);


%% problem 4.3

clear all;

x(1) = 1;
y(1) = 0;
v_x(1) = 0;
v_y(1) = 3*pi;
dt = 0.01;
```

```
ft = 10*pi;
t = [0:dt:ft+dt];
r(1) = sqrt(x(1)^2 + y(1)^2);
A(1) = 0;

for i = 1:length(t)
    r = sqrt(x(i)^2 + y(i)^2);
    v_x(i+1) = v_x(i) - (((4*pi^2)*x(i))/r^3)*dt;
    v_y(i+1) = v_y(i) - (((4*pi^2)*y(i))/r^3)*dt;
    x(i+1) = x(i) + v_x(i+1)*dt;
    y(i+1) = y(i) + v_y(i+1)*dt;

    %finding the area of the triangles created by each dt using Heron's
    %formula
    r(i) = sqrt(x(i)^2 + y(i)^2);
    r(i+1) = sqrt(x(i+1)^2 + y(i+1)^2);
    a = r(i);
    b = r(i+1);
    c = sqrt((x(i+1)-x(i))^2 + (y(i+1)-y(i))^2);
    p = (r(i)+r(i+1)+c)/2;
    A(i) = sqrt(p*(p-a)*(p-b)*(p-c));

    scatter(x(i), y(i), 0.5);
    hold on;
end

ave_A = mean(A);

for i=1:length(A)
   total_diff = (ave_A-A(i))^2;
   variance_A = total_diff/length(A);
   std_dev_A = sqrt(variance_A);
end

n = 2;
m = n;

x = 0;
y = 0;
plot(x,y,'ro');
title('Orbital Motion, Keplers Second Law, Vyi = 3pi, dt = 0.01');
axis([-n,m,-n,m]);
xlabel('X (AU)');
ylabel('Y (AU)');
disp(std_dev_A);


%%problem 4.8

clear all;
```

```
%planet stuff
%k ->mass ratio: (planet mass)/(earth mass)
k = 1;

clear fig;

B = 2;
x(1) = 1;
y(1) = 0;
v_x(1) = 0;
v_y(1) = 2.4*pi;
dt = 0.01;
ft = 100*pi;
t = [0:dt:ft+dt];
r(1) = sqrt(x(1)^2 + y(1)^2);
A(1) = 0;
theta(1) = 0;
less_05 = false;
more_05 = false;
count = false;
cycles = 0;

%details about the planets
%the mass of the sun given in terms of 1 = 1*10^23
Ms = 1.989*10^7;
Mp = 4.9*10^1;

for i = 1:length(t)
    r(i) = sqrt(x(i)^2 + y(i)^2);
    v_x(i+1) = v_x(i) - (((k*4*pi^2)*x(i))/((r(i)^B)*r(i)))*dt;
    v_y(i+1) = v_y(i) - (((k*4*pi^2)*y(i))/((r(i)^B)*r(i)))*dt;
    x(i+1) = x(i) + v_x(i+1)*dt;
    y(i+1) = y(i) + v_y(i+1)*dt;

    r_t = r(i);
    theta(i) = acos(x(i)/r(i));


    if(theta(i) < 0.5)
       less_05 = true;
       more_05 = false;
    end
    if(theta(i) > 0.5)
        more_05 = true;
        less_05 = false;
        count = false;
    end

    if(~count && less_05)
```

```
       cycles = cycles + 1;
       count = true;
    end
    %finding the area of the triangles created by each dt using Heron's
    %formula
    r(i) = sqrt(x(i)^2 + y(i)^2);
    r(i+1) = sqrt(x(i+1)^2 + y(i+1)^2);
    a = r(i);
    b = r(i+1);
    c = sqrt((x(i+1)-x(i))^2 + (y(i+1)-y(i))^2);
    p = (r(i)+r(i+1)+c)/2;
    A(i) = sqrt(p*(p-a)*(p-b)*(p-c));

    scatter(x(i), y(i), 0.5);
    hold on;
end

ave_A = mean(A);

for i=1:length(A)
   total_diff = (ave_A-A(i))^2;
   variance_A = total_diff/length(A);
   std_dev_A = sqrt(variance_A);
end

%Calculate the T^2/a^3 ratio;
T = ft/cycles;
a = (1/2)*(min(r)+max(r));
K_3 = T^2/a^3;

n = 4;
m = n;

x = 0;
y = 0;
plot(x,y,'ro');
title('Orbital Motion, Keplers Third Law, Vyi = 2.4pi, dt = 0.008, Venus');
axis([-n,m,-n,m]);
xlabel('X (AU)');
ylabel('Y (AU)');
%to add text into the graph
txt1 = -n+0.5;
txt2 = -m+0.5;
T_constant = num2str(K_3);
T_graph = strcat('(T^2)/(a^3) = ',T_constant);
%text(txt1,txt2,T_graph);
disp(std_dev_A);
disp(cycles);
disp(K_3);
```

```
%%problem 4.13

%Current issue: [5-4-15] Adding the planet now, the given conditions
%for the binary star system produce a stable orbit for the time interval of
%t = 100*pi. Took the plotting functions out of the for loop, added to the
%end of the code, has made the code run about 1000x faster now.

clear all;

%planet stuff
%k ->mass ratio: (planet mass)/(earth mass)
%for our stars here, we'll use the mass of the sun in terms of # of earths
Ms1 = 331500;
Ms2 = 331500;
Mp1 = 1;
m_f = 30;
pm_f = 13;
%m_f = 30-46 produces a stable orbit for t = 2*pi
%note that 'stable' orbits are really only relative to some time scale

%controls the size of the dots in the scatter plots
dot_size = 1;
planet_dot_size = 1;

B = 2;

%star 1
x_s1(1) = 1*m_f;
y_s1(1) = 0*m_f;
v_xs1(1) = 0*m_f;
v_ys1(1) = 1.75*pi*m_f;

%star 2
x_s2(1) = -1/sqrt(2)*m_f;
y_s2(1) = 1/sqrt(2)*m_f;
v_xs2(1) = 0*m_f;
v_ys2(1) = -1.8*pi*m_f;

l = 2.5629;

%planet 1
x_p1(1) = 300*pm_f;
y_p1(1) = 0*pm_f;
v_xp1(1) = -2*pi*pm_f;
v_yp1(1) = l*pi*pm_f;

dt = 0.0001;
ft = 100*pi;
t = [0:dt:ft+dt];
```

```
%Begin all necessary radii
r_s1s2(1) = sqrt(((x_s1(1) - x_s2(1))^2) + ((y_s1(1) - y_s2(1))^2));
r_s1p1(1) = sqrt(((x_s1(1) - x_p1(1))^2) + ((y_s1(1) - y_p1(1))^2));
r_s2p1(1) = sqrt(((x_s1(1) - x_s2(1))^2) + ((y_s1(1) - y_s2(1))^2));

%{
A(1) = 0;

theta1(1) = 0;
theta2(1) = 0;

%boolean for the first star
less_05_s1 = false;
more_05_s1 = false;
count_s1 = false;
cycles_s1 = 0;

%boolean for the second star
less_05_s2 = false;
more_05_s2 = false;
count_s2 = false;
cycles_s2 = 0;
%}

for i = 1:length(t)

    %find all the necessary radii
    r_s1s2(i) = sqrt(((x_s1(i) - x_s2(i))^2) + ((y_s1(i) - y_s2(i))^2));
    r_s1p1(i) = sqrt(((x_s1(i) - x_p1(i))^2) + ((y_s1(i) - y_p1(i))^2));
    r_s2p1(i) = sqrt(((x_s1(i) - x_p1(i))^2) + ((y_s1(i) - y_p1(i))^2));

    %Calculate stuff for the first star
    v_xs1(i+1) = v_xs1(i) - (((Ms1*4*pi^2)*x_s1(i))/((r_s1s2(i)^B)*r_s1s2(i)))*dt - (
    v_ys1(i+1) = v_ys1(i) - (((Ms1*4*pi^2)*y_s1(i))/((r_s1s2(i)^B)*r_s1s2(i)))*dt - (
    x_s1(i+1) = x_s1(i) + v_xs1(i+1)*dt;
    y_s1(i+1) = y_s1(i) + v_ys1(i+1)*dt;

    %calculate stuff for the second star

    v_xs2(i+1) = v_xs2(i) - (((Ms2*4*pi^2)*x_s2(i))/((r_s1s2(i)^B)*r_s1s2(i)))*dt - (
    v_ys2(i+1) = v_ys2(i) - (((Ms2*4*pi^2)*y_s2(i))/((r_s1s2(i)^B)*r_s1s2(i)))*dt - (
    x_s2(i+1) = x_s2(i) + v_xs2(i+1)*dt;
    y_s2(i+1) = y_s2(i) + v_ys2(i+1)*dt;

    %calculate stuff for the first planet
    v_xp1(i+1) = v_xp1(i) - (((Ms1*4*pi^2)*x_p1(i))/((r_s1p1(i)^B)*r_s1p1(i)))*dt - (
    v_yp1(i+1) = v_yp1(i) - (((Ms1*4*pi^2)*y_p1(i))/((r_s1p1(i)^B)*r_s1p1(i)))*dt - (
    x_p1(i+1) = x_p1(i) + v_xp1(i+1)*dt;
    y_p1(i+1) = y_p1(i) + v_yp1(i+1)*dt;
```

```
%{
theta1(i) = acos(x_s1(i)/r_s1s2(i));
theta2(i) = acos(x_s2(i)/r_s1s2(i));

%counting orbits for the first star
if(theta1(i) < 0.5)
   less_05_s1 = true;
   more_05_s1 = false;
end
if(theta1(i) > 0.5)
    more_05_s1 = true;
    less_05_s1 = false;
    count_s1 = false;
end

if(~count_s1 && less_05_s1)
   cycles_s1 = cycles_s1 + 1;
   count_s1 = true;
end

  %counting orbits for the second star
if(theta2(i) < 0.5)
   less_05_s2 = true;
   more_05_s2 = false;
end
if(theta1(i) > 0.5)
    more_05_s2 = true;
    less_05_s2 = false;
    count_s2 = false;
end

if(~count_s2 && less_05_s2)
   cycles_s2 = cycles_s2 + 1;
   count_s2 = true;
end
%}

%finding the area of the triangles created by each dt using Heron's
%formula
%{
r_s1s2(i) = sqrt(x(i)^2 + y(i)^2);
r(i+1) = sqrt(x(i+1)^2 + y(i+1)^2);
a = r(i);
b = r(i+1);
c = sqrt((x(i+1)-x(i))^2 + (y(i+1)-y(i))^2);
p = (r(i)+r(i+1)+c)/2;
A(i) = sqrt(p*(p-a)*(p-b)*(p-c));
%}
```

```
end

%{
% Calculate the variance of the areas created by the distance traveled by
% the orbit over some time interval

ave_A = mean(A);

for i=1:length(A)
   total_diff = (ave_A-A(i))^2;
   variance_A = total_diff/length(A);
   std_dev_A = sqrt(variance_A);
end

%Calculate the T^2/a^3 ratio;

%first star
T_s1 = ft/cycles_s1;
a_s1 = (1/2)*(min(r_s1s2)+max(r_s1s2));
K_3_s1 = T_s1^2/a_s1^3;


second star
T = ft/cycles_s2;
a = (1/2)*(min(r_s1s2)+max(r_s1s2));
K_3 = T^2/a^3;

%}

%plot the things
n = 40*m_f*pm_f;
m = n;

scatter(x_s1, y_s1, dot_size,'r');
hold on;
scatter(x_s2, y_s2, dot_size,'b');
hold on;
scatter(x_p1, y_p1, planet_dot_size,'g');
hold on;

% x = 0;
% y = 0;
% plot(x,y,'ro');
title('Orbital Motion, Binary Stars, Vyi_s1 = 1.75*pi, Vyi_s2 = -1.8*pi, dt = 0.01');
axis([-60,60,-60,60]);
xlabel('X (AU)');
ylabel('Y (AU)');

%to add text into the graph
%{
```

```
txt1 = -n+0.5;
txt2 = -m+0.5;
T_constant = num2str(K_3);
T_graph = strcat('(T^2)/(a^3)',T_constant);
text(txt1,txt2,T_graph);
%}

%send results to the console
%disp(std_dev_A);
%disp(cycles);
%disp(K_3);
```