Foundations of the Problem
The magnitude of the gravitational attraction between two objects can be calculated with the formula:

F = G * (M1*M2)/r^2

Where G is the gravitational constant 6.674e-8
M1 is mass of the first object
M2 is mass of the second object
And r Is the distance between the two objects

The direction of this force will be towards each other for both objects

We can use this equation to simulate planets orbiting the sun in our solar system. In order to get a correct orbit, we must configure the planets with correct initial values for their velocity. The magnitude of the velocity required to maintain orbit around the sun can be found with:

Sqrt(sunsMass*G/r)

And is in a direction perpendicular to the direction of the gravitation force

With these simple equations we write a program in MATLAB to simulate unlimited planets and their paths using Euler's method.

Text From the Program
In MATLAB I used arrays to hold the planets initial values, allowing any number of planets to be easily added and configured.

```
planetsStartPositions = [

    %147300000000 0
    0 147300000000 %earth
    772300000000 0 %jupiter
    772300000000 *1.5 0%asteroid 1
    147300000000*2.1 147300000000%asteroid 2
    772300000000 * 0.95 0%asteroid 3
];

planetsMass = [

    5.972e27 %earth
    1.898e30 %jupiter
    2.95e11 %asteroid 1
    2.95e10 %asteroid 2
    2.95e11 %asteroid 3
];
```

```
planetsVelocity = [
]
```

Of course, each planet needs to have an initial velocity that will result in an orbit, so this is not filled in but rather calculated using the formula mentioned earlier:

```
%get starting velocities and setup positions
for i = 1:length(planetsStartPositions)
    temp = [];
    planetsVelocity(i,1) = 0;
    direction = [sunsPositionX-planetsStartPositions(i,1), sunsPositionY-
planetsStartPositions(i,2)];
    radius = norm(direction);
    direction = direction/radius;
    perp = [-1*direction(2),direction(1)];
    velocity = sqrt((sunsMass*G)/radius);
    planetsVelocity(i,1) = velocity * perp(1);
    planetsVelocity(i,2) = velocity * perp(2);

    %setup start positions
    x(i,1) = planetsStartPositions(i,1);
    y(i,1) = planetsStartPositions(i,2);
end
```

I also fill in the first position of each planet in our x and y 2d arrays within this loop

Next, I setup the guts of the Euler Method using a time step of 20 days and a limit on how many iterations (10,000 so 20 days each would mean 200,000 days in this case)

```
t= 0;
step =20;

i = 1;
while t < 100000
```

Now, each iteration I must loop through every planet, and on every individual planet I must calculate the total force from every other planet. This requires a loop within a loop.

So, for each planet

```
 %for each planet
    for u = 1:length(planetsStartPositions)
```

First I Calculate the gravity created by the sun

```
        %get gravity of the sun
        position = [x(u,i),y(u,i)];
        mass = planetsMass(u);
        direction = [position(1) - sunsPositionX, position(2) - sunsPositionY];
        radius = norm(direction);
```

```
        direction = direction/radius;

        gravity = (G*mass*sunsMass)/(radius^2) * direction;
```

Then I calculate the gravity force from every other planet and keep track of the total sum

```
        %get gravity from other planets
        for r = 1:length(planetsStartPositions)
            %skip if its our own planet
            if r == u
                continue
            end

            massPlanet2 = planetsMass(r);
            positionPlanet2 = [x(r,i),y(r,i)];
            directionPlanet = [position(1) - positionPlanet2(1), position(2) - po
sitionPlanet2(2)];
             radiusPlanet = norm(directionPlanet);
        directionPlanet = directionPlanet/radiusPlanet;

         gravityPlanet = (G*mass*massPlanet2)/(radiusPlanet^2) * directionPlanet;
         gravity = gravity + gravityPlanet;%sum all forces
        end
```

Finally, using the total gravity force I can calculate the net acceleration of that planet

```
        acceleration = -(gravity/mass);
```

And use the Euler Method to update the planets velocity and position each iteration

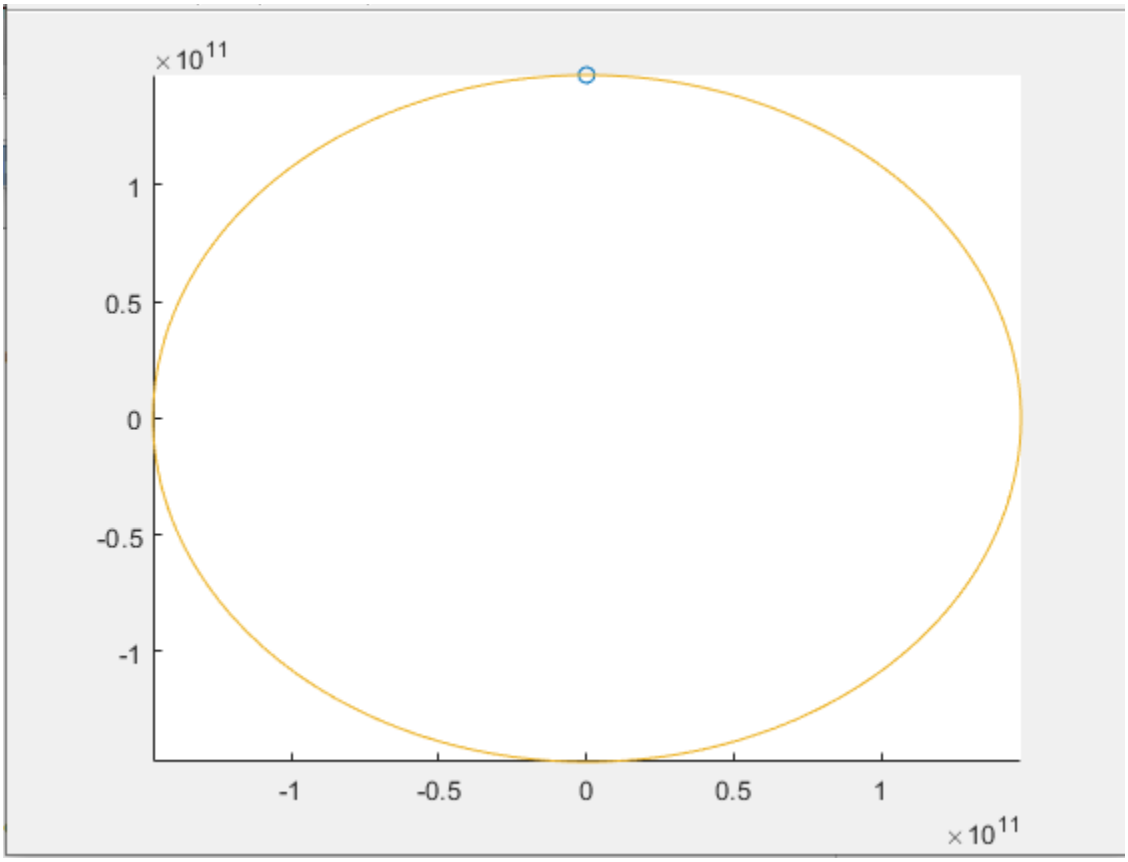
```
         planetsVelocity(u,1) =  planetsVelocity(u,1) + (acceleration(1) * step);
         planetsVelocity(u,2) = planetsVelocity(u,2) + (acceleration(2) * step);

         x(u,i + 1) =  x(u,i) + (planetsVelocity(u,1) * step);
           y(u,i + 1) =  y(u,i) + (planetsVelocity(u,2) * step);
    end
```

Once every position is calculated and stored in our x and y 2d arrays, I plot the results using the Comet function so we can watch the planets do their orbits

```
end
hold on
axis equal;
   for u = 1:length(planetsStartPositions)
        comet(x(u,:),y(u,:));
   end
```

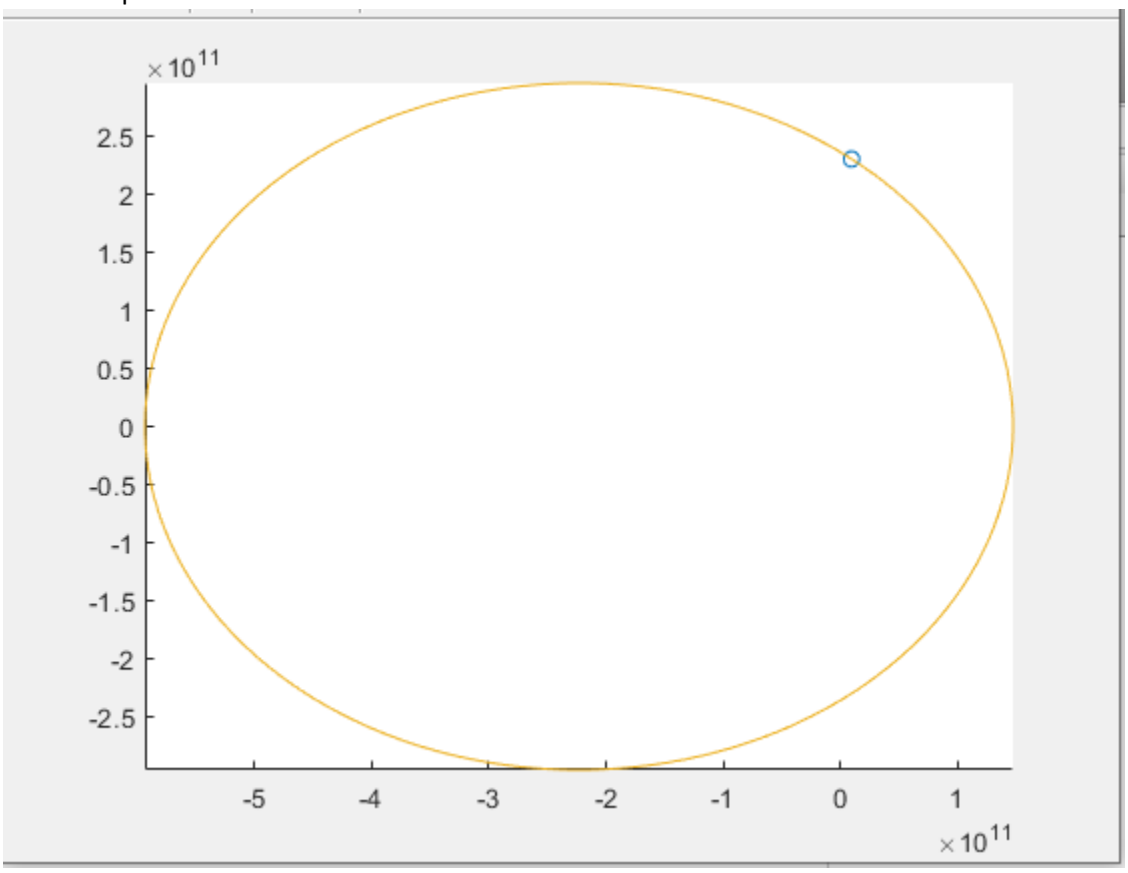
Outputs

The first result I looked at was the earth doing a normal orbit around the sun. You can see its orbit is a perfect circle as its initial velocity was calculated to be the exact velocity required for a circular orbit using the formula mentioned earlier: Sqrt(sunsMass*G/r)



If we change the initial velocity to be something else, the earth takes on an elliptical orbit. In this Image the earth went from about a 3e7 calculated orbit to a specifically set 3.8e7 orbit, and you can see its orbi

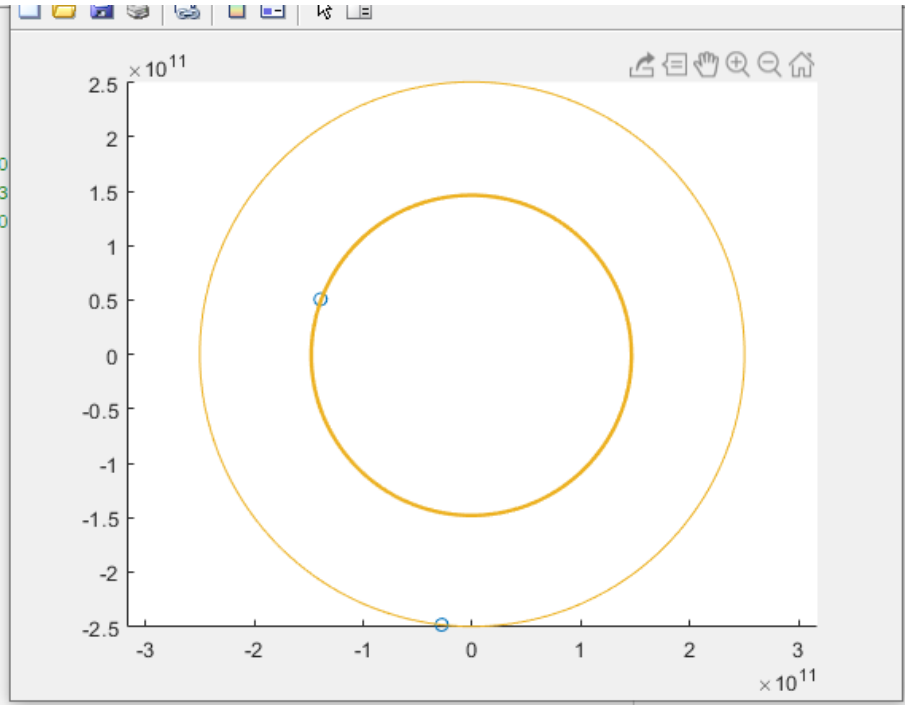
is now elliptical.



Next, I added a smaller inner planet and a large outer planet and varied the mass to see how they would affect each other.

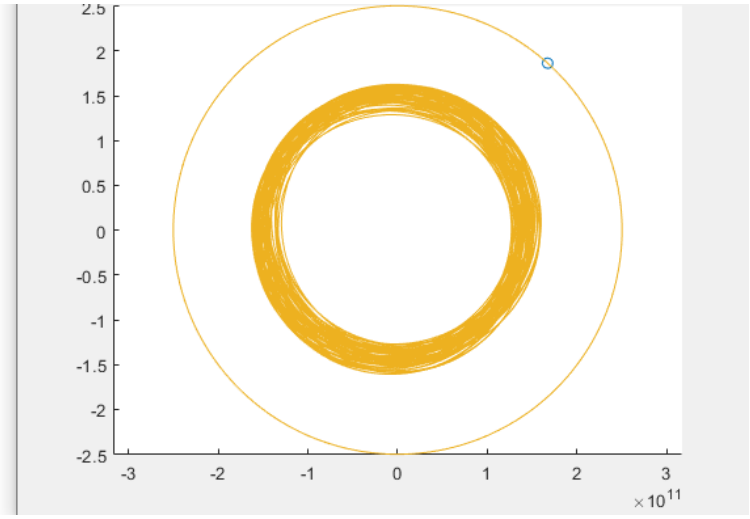
Here, the planets have a relatively close mass and almost no effect is seen from each planet.

```
planetsStartPositions = [

    %147300000000 0
    0 147300000000
    147300000000*1.7 0
   % 772300000000 * 1.05 0
    %147300000000*2.1 1473
   % 772300000000 * 0.95 0
];

planetsMass = [

    5.972e27
    1.898e30
   % 2.95e11 %asteroid 1
   % 2.95e10 %asteroid 2
   % 2.95e11 %asteroid 3
];

x = [];
y = [];

sunsMass = 1.989e33;
G=6.674e-8;
```



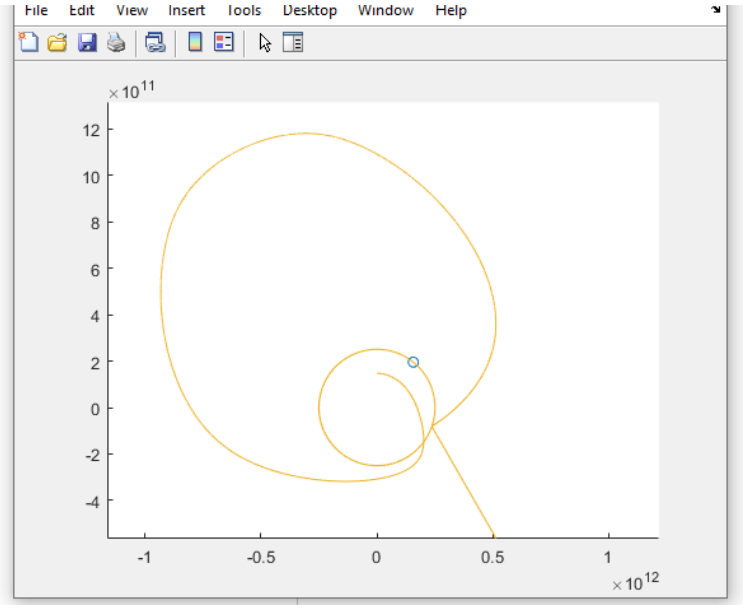
Yet, when I increased the mass by a factor of 10, you can see the smaller middle planet's orbit begins to be affected

```
%a  little resonance in the middle one
planetsStartPositions = [

    %147300000000 0
    0 147300000000
    147300000000*1.7 0
   % 772300000000 * 1.05 0%asteroid 3
    %147300000000*2.1 147300000000%asteroid 2
   % 772300000000 * 0.95 0%asteroid 3
];

planetsMass = [

    5.972e27
    1.898e31
   % 2.95e11 %asteroid 1
   % 2.95e10 %asteroid 2
   % 2.95e11 %asteroid 3
];

x = [];
```

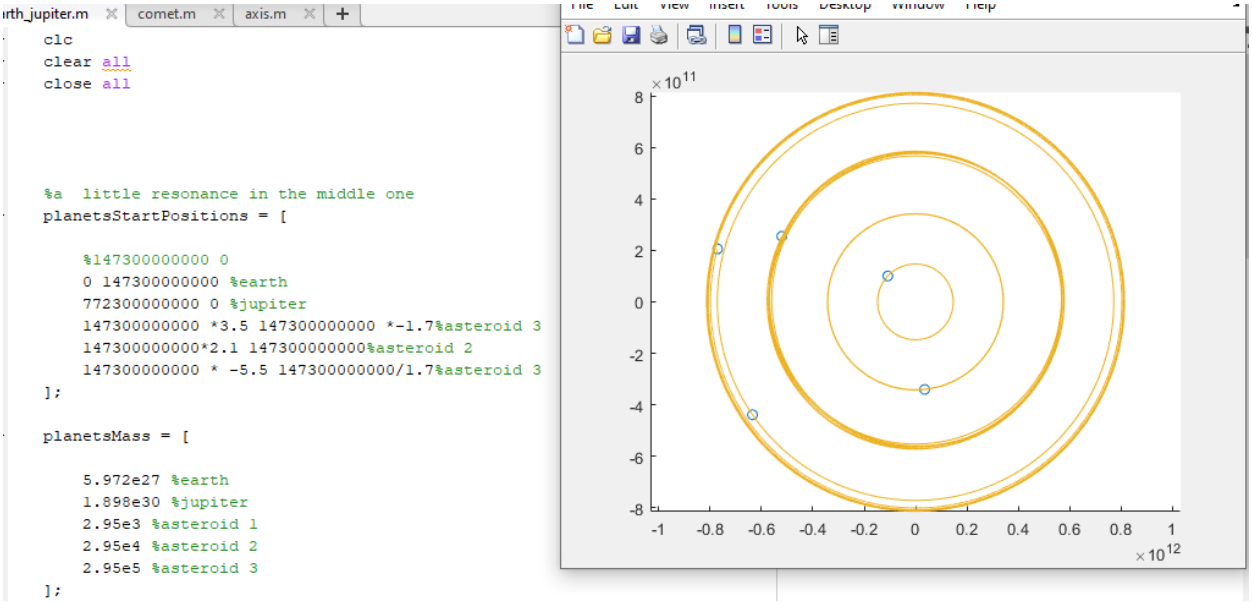


I increased the mass by another factor of 100 and the smaller inner planet was thrown straight out of the solar system!

```
close all



%a  little resonance in the middle one
planetsStartPositions = [

    %147300000000 0
    0 147300000000
    147300000000*1.7 0
   % 772300000000 * 1.05 0%asteroid 3
    %147300000000*2.1 147300000000%asteroid 2
   % 772300000000 * 0.95 0%asteroid 3
];

planetsMass = [

    5.972e27
    1.898e33
   % 2.95e11 %asteroid 1
   % 2.95e10 %asteroid 2
   % 2.95e11 %asteroid 3
];

x = [];
```

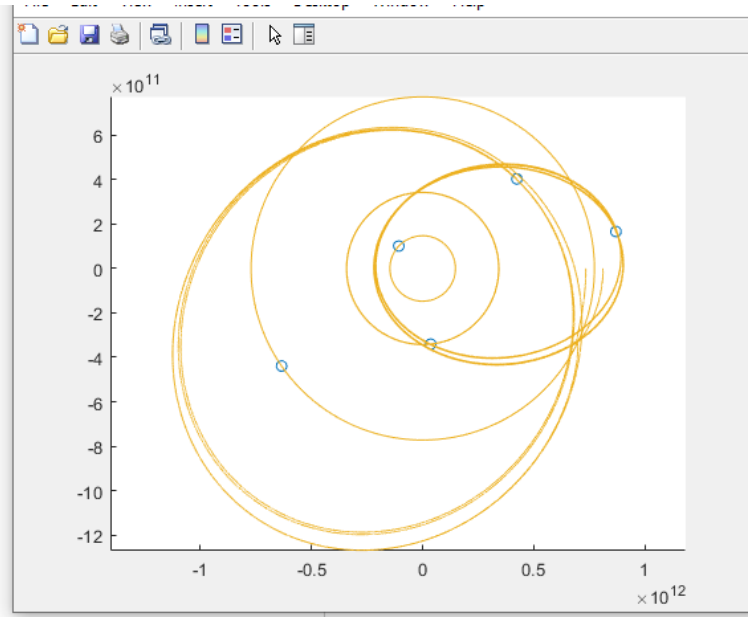


Finally, I setup the planets to simulate the Earth, Jupiter, and 3 asteroids. You can see the asteroids orbits are slightly affected, most likely by the pull of Jupiter as its mass is much larger than earths.



When I moved the asteroids a little closer to Jupiter the effect is much more interesting, and the asteroid's paths become irregular orbits. Notice that the two asteroids with irregular orbits are pulled of course not once, but two times as they pass close to Jupiter.

```
close all



%a  little resonance in the middle one
planetsStartPositions = [

    %147300000000 0
    0 147300000000 %earth
    772300000000 0 %jupiter
    772300000000 * 1.05 0%asteroid 3
    147300000000*2.1 147300000000%asteroid 2
    772300000000 * 0.95 0%asteroid 3
];

planetsMass = [

    5.972e27 %earth
    1.898e30 %jupiter
    2.95e11 %asteroid 1
    2.95e10 %asteroid 2
    2.95e11 %asteroid 3
];

x = [];
```



Then, for fun, I made the asteroids have a mass 10 times bigger than Jupiter. The orbits were surprisingly more calm then expected and only one asteroid was launched from the solar system.

```
%a  little resonance in the middle one
planetsStartPositions = [

    %147300000000 0
    0 147300000000 %earth
    772300000000 0 %jupiter
    147300000000 *3.5 147300000000 *-1.7%asteroid 3
    147300000000*2.1 147300000000%asteroid 2
    147300000000 * -5.5 147300000000/1.7%asteroid 3
];

planetsMass = [

    5.972e27 %earth
    1.898e30 %jupiter
    2.95e31 %asteroid 1
    2.95e31 %asteroid 2
    2.95e31 %asteroid 3
];

x = [];
```