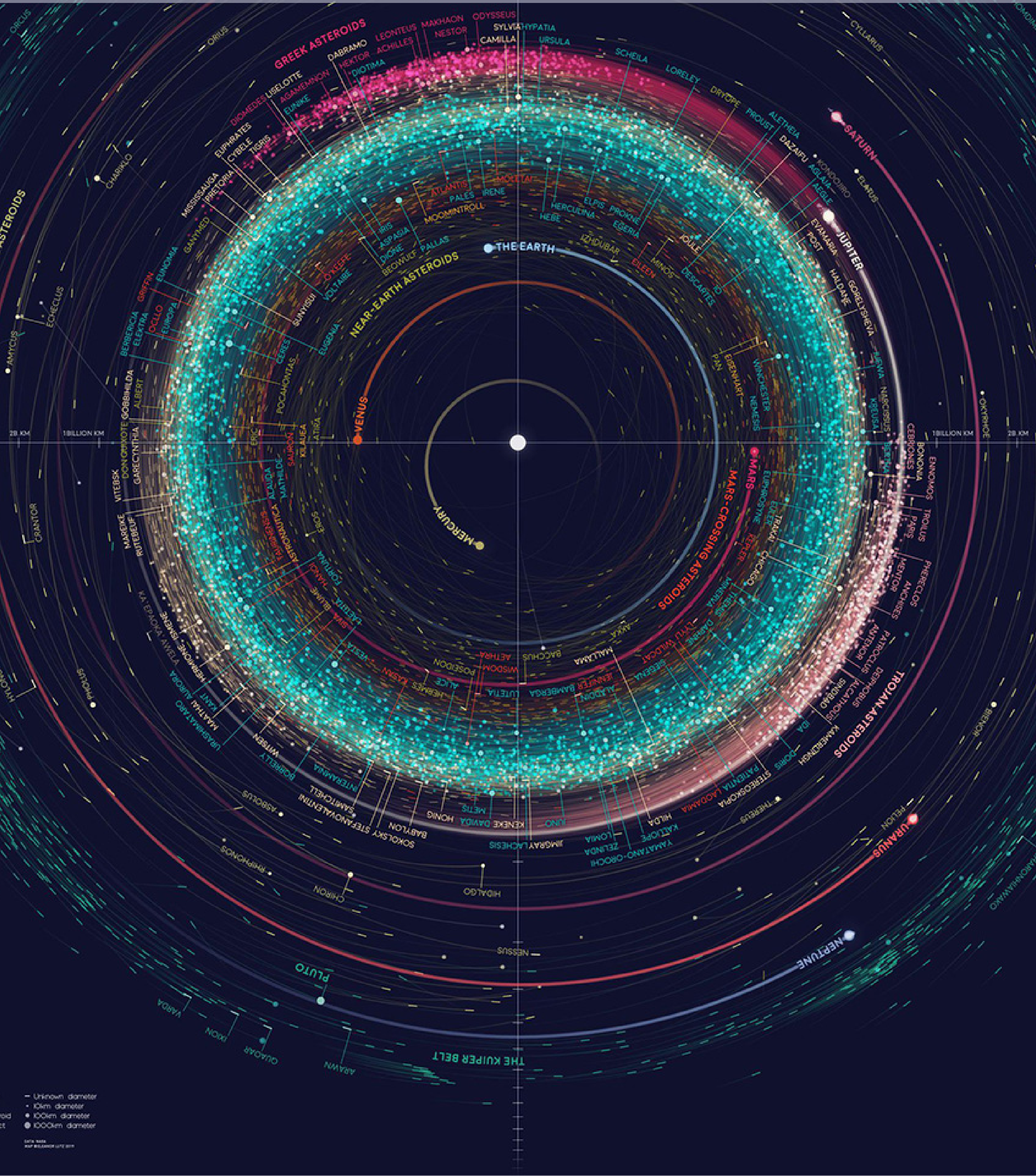
Solar System Dynamics

Suryansh Patidar



Solar System Dynamics

Krittika Summer Project 4.0

in

Astronomy

Author: Suryansh Patidar
Supervisor: Adarsh Reddy Madur
Second supervisor: Dhananjay Raman

Contents

1. System Involving 2 Bodies

1.1 Two Body Problem

The two-body problem is a fundamental problem in classical mechanics and celestial mechanics. It deals with the motion of two-point masses (or bodies) that interact solely through gravitational forces. The problem can be simplified by considering the masses of the bodies to be concentrated at their centers, which allows us to treat them as point masses.

In the two-body problem, the two masses are subject to the gravitational attraction between them. This means that each mass exerts a force on the other, causing them to move in specific paths. The challenge is to determine the positions and velocities of the bodies at any given time, given their initial positions and velocities and the law of gravitation.

The two-body problem was first solved by Johannes Kepler, who formulated three laws of planetary motion based on the observations of Tycho Brahe. Kepler's laws describe the motion of planets around the Sun and provide fundamental insights into orbital mechanics:

Kepler's First Law (Law of Orbits): Each planet moves in an elliptical orbit, with the Sun at one of the two foci.

Kepler's Second Law (Law of Areas): A line segment joining a planet and the Sun sweeps out equal areas during equal intervals of time. This implies that a planet moves faster when closer to the Sun (perihelion) and slower when farther away (aphelion).

Kepler's Third Law (Law of Harmonies): The square of the orbital period of a planet is directly proportional to the cube of the semi-major axis of its orbit.

For more complex cases, where the masses of the two bodies are comparable or if other forces are involved (e.g., perturbations from other celestial bodies), the problem becomes more challenging and requires more sophisticated mathematical and numerical methods for solutions.

The two-body problem is not only relevant to celestial mechanics but also finds applications in various areas, such as satellite motion, space missions, and interplanetary travel, where the gravitational interaction between two masses plays a crucial role in determining their trajectories.

1.2 Plots

1.2.1 Elliptical Orbit

An elliptical orbit is a type of closed orbit followed by a celestial object (e.g., a planet or satellite) around a central body (e.g., a star). In an elliptical orbit, the path traced by the object is an ellipse, which is a shape similar to a flattened circle. The central body, such as a star, is located at one of the two foci of the ellipse. The other focus remains empty.

In an elliptical orbit, the distance between the celestial object and the central body varies as the object moves along its path. At one point, known as the perihelion (for objects orbiting the Sun), the object is closest to the central body. At another point, known as the aphelion, it is farthest from the central body.

Elliptical orbits follow Kepler's laws of planetary motion, which state that planets move in elliptical orbits with the Sun at one of the foci.
Starting with the normal Euler integration method, I wrote the whole code for plotting the orbit in 'Class' as suggested by mentors, so that I will be able to use that further when it will be required.

1.2.2 Hyperbolic Orbit

A hyperbolic orbit is an open orbit followed by a celestial object when it moves under the influence of a gravitational field, but its speed exceeds the escape velocity of the central body. In a hyperbolic orbit, the path traced by the object is a hyperbola, which is a curve that extends infinitely away from the central body.

Unlike elliptical orbits, which are closed and bound, hyperbolic orbits are unbound and open. The object never returns to the central body, and its path does not form a closed loop. Instead, it continues moving away from the central body, reaching greater distances.

Hyperbolic orbits are often associated with comets or other celestial objects that come from distant regions of the solar system or beyond. When they approach a star, they can be accelerated to speeds exceeding the escape velocity, resulting in a hyperbolic trajectory.

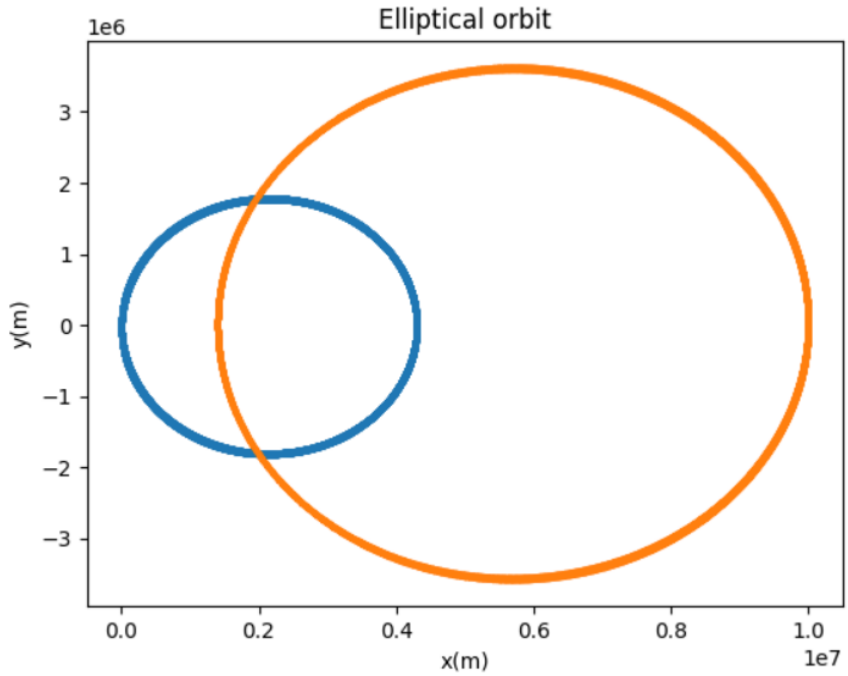
In summary, elliptical orbits are closed and bound, forming elliptical paths around a central body, while hyperbolic orbits are open and unbound, following hyperbolic paths away from a central body. Both types of orbits play crucial roles in celestial mechanics and space exploration.

Using the same code as for the elliptical orbit, I just changed the parameter values for the two particle.
The format of code that I used for plotting the trajectory(both for Elliptical and Hyperbolic orbit) has been given below:

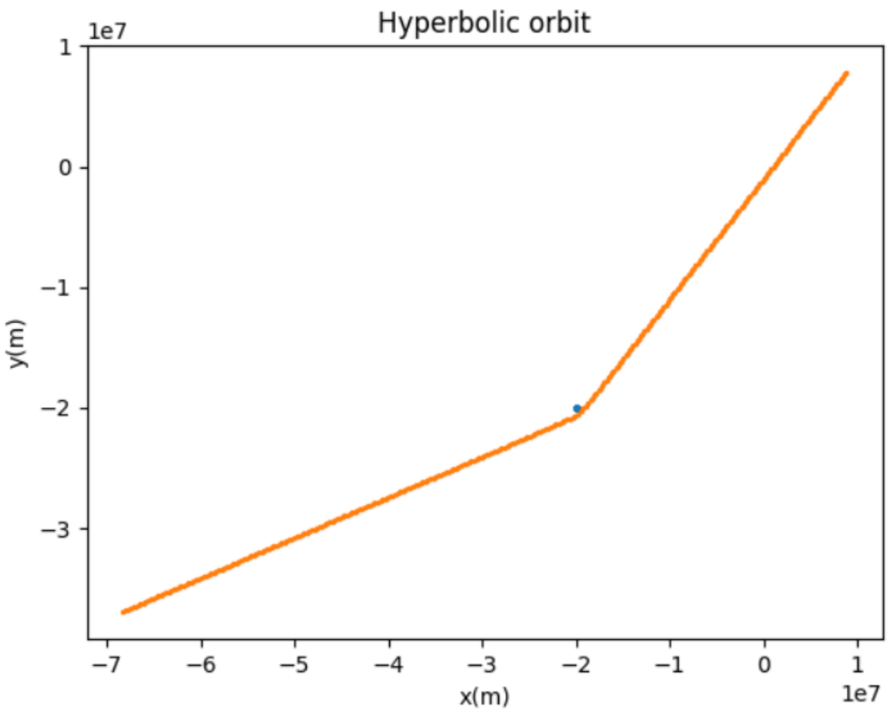
```


class Particle:
    def __init__(self, mass, x,y,vx,vy):
        # For defining the mass, co-ordinates and velocity components for the particles.
     def update(self,time_step,f_x,f_y):
        # For updating the vel and x,y-coordinates according to the acceleration, that comes from the type of force for which we want to plot
class GravitationalSimulation:
      def __init__(self, particle1, particle2):
        # Initializing particle1 and particle2, so that I can use its parameters for further code
     def calculate_force(self):
        # For storing the gravitational forces that i calculated(component-wise) in a numpy array
     def simulate(self, time_step, num_steps):
        # For simulating/ plotting the trajectory(scatter plot) by updating the positions according to the x and y forces calculated in above function
# Parameter values that i used for Elliptical Orbit was
    "new_particle_1 = Particle(2e30,0,0,0,1e6)
     new_particle_2 = Particle(1e30,1e7,0,0,-2e6)"
# And the parameter values that i used for Hyperbolic Orbit was
    "particle_1=Particle(2e30,-2e7,-2e7,0,0)
     particle_2=Particle(6e18,0.9e7,0.8e7,-2e7,-2e7)"
```

The plots that i got was the ones given below,



(a) *Elliptical Orbit*



(b) *Hyperbolic orbit*

2. Three Body System

2.1 Introduction

The three-body problem is a classical mechanics problem in physics that deals with the motion of three objects (usually celestial bodies) under the influence of their mutual gravitational attraction, without any other external forces. The problem is to find the positions and velocities of the three bodies at any given time, considering their initial positions and velocities.

The three-body problem is notoriously difficult to solve analytically, and in most cases, there is no general closed-form solution. This is because the gravitational interactions between the bodies result in complex and chaotic behaviors. While solutions can be found in special cases, such as when one body is much less massive than the other two (reducing it to a two-body problem), finding precise solutions for arbitrary three-body systems is a major challenge.

2.2 Lagrange Points

Lagrange points are special points in a three-body system where the gravitational forces of two large bodies and the centrifugal force of a smaller body combine to create an equilibrium point. At these points, the gravitational attraction of the two large bodies exactly balances the centrifugal force of the smaller body, allowing it to remain in a stable relative position.

There are five Lagrange points in a three-body system, denoted as L1 to L5. The first three, L1, L2, and L3, form a line connecting the two large bodies, with L1 and L2 being located on opposite sides of the smaller body. L3 lies directly opposite the smaller body on the line connecting the two larger bodies.

L4 and L5, known as the Trojan points, are located 60 degrees ahead of and behind the smaller body in its orbit, forming equilateral triangles with the two larger bodies. These points are often stable and have been observed in celestial systems, like the Jupiter-Trojan asteroids.

Lagrange points have significant practical importance in space missions, as they offer relatively stable positions for spacecraft and satellites. For example, some space observatories are placed at Lagrange points to maintain a stable position relative to the Earth and the Sun, allowing for continuous observations of distant regions of space.

In summary, the three-body problem involves the motion of three bodies under their mutual gravitational attraction, and Lagrange points are special points in such systems where the gravitational forces create an equilibrium that can be used in space mission planning.

2.3 Reduced 3 body

Simulating for the Reduced 3 body problem by plotting the Contour of Potential energy that shows the Lagrange points.
I was not getting correct plot even after writing the code correctly.
Then after changing the parameter values and making masses more comparable, I got a decent plot but not totally accurate..
For the code, i only had to add these functions in above code for trajectories of 2-Body system.

```
     def potential_energy(self,x,y):
        # For calculating the potential energy in a reduced 3 body problem

    def generate_contour_plot(self,x0,y0,x1,y1,n):

        # It will generate the contour according to the number of x and y values

        # Parameter values that i used for Contour plotting was
            "new_particle_1 = Particle(2e30,0,0,0,1e6)
             new_particle_2 = Particle(1e30,1e7,0,0,-2e6)"

        # I chose this as to make the masses comparable so that i need not had to zoom in the figure to my area of interest that i might have to do in the case of sun-earth where masses are not comparable
```

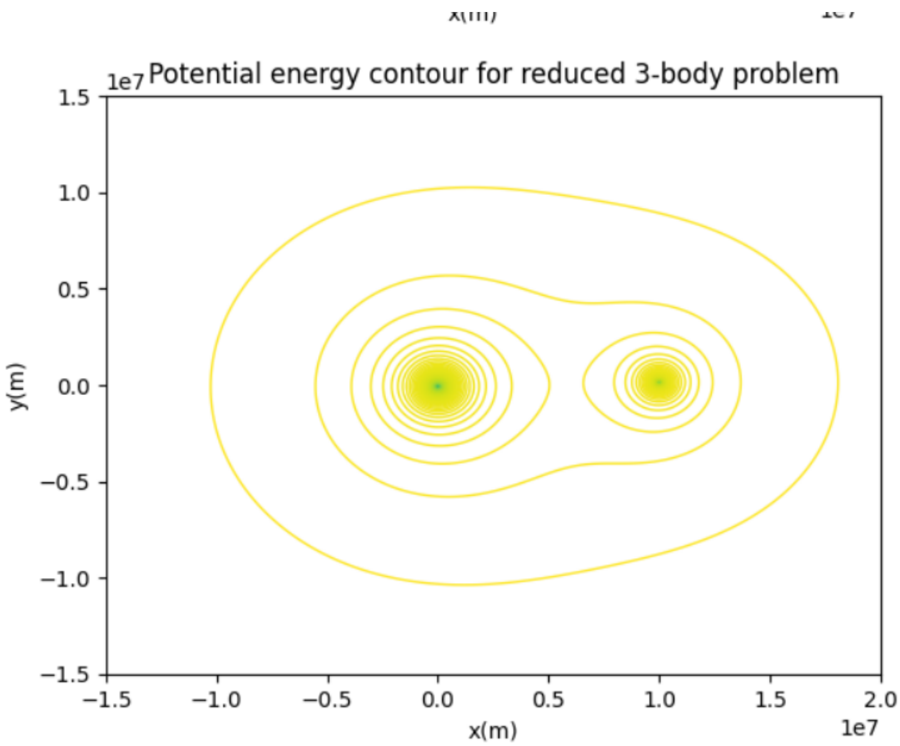


Figure 2.1: *Contour plot of potential energy for the reduced 3 body*

3. N-body System

The N-body problem is a classical problem in physics and astronomy that deals with predicting the motion of N celestial objects (typically planets, stars, or other massive bodies) under the influence of gravitational forces. In simple terms, it involves determining how a collection of N bodies will move over time due to their mutual gravitational interactions.

The fundamental principle behind the N-body problem is Newton's law of universal gravitation, which states that every particle in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them.

Despite its apparent simplicity, solving the N-body problem analytically is extremely challenging, especially as the number of bodies (N) increases. In the case of just two bodies, as demonstrated by Isaac Newton, the problem can be solved exactly. However, for three or more bodies, the problem becomes highly complex, and there is no known general analytical solution for an arbitrary number of bodies.

For this reason, numerical methods and computer simulations are commonly used to approximate the behavior of N-body systems. Various techniques, such as the N-body simulation algorithms (e.g., direct summation, Barnes-Hut algorithm) and symplectic integrators, are employed to predict the positions and velocities of the bodies at discrete time steps.

The N-body problem has significant implications in astrophysics, cosmology, and space mission planning. It is used to study celestial phenomena, such as planetary motion, star clusters, galaxy dynamics, and even the evolution of the entire universe. Moreover, it plays a crucial role in designing spacecraft trajectories and understanding the stability and long-term behavior of planetary systems.

The N-body problem is a fundamental challenge in physics and astronomy, involving the prediction of the motion of multiple celestial bodies under the influence of gravitational forces. While no general analytical solution exists, numerical methods and simulations are commonly employed to study and approximate the behavior of N-body systems in various scientific applications.

```
    class Particle:
        def __init__(self, mass, x,y,vx,vy,label):
            # For defining the mass, co-ordinates and velocity components for the particles.
    def update(particles,dt):
        # I wrote two 'for' loops, and calculated the gravitational force for every particle due to the other particles and updating its position and velocities according to the force.
    def simulate(particles, dt, steps):
        # I maintained two list in which i appended the 'x' and 'y' positions for every particle.
    # Parameter i used was for a 7 body system, which was:
      "particles=[ Particle(2e30,0,0,0,0,"p1"),
    Particle(5.97e24,1.496e11,0,0,2.978e4,"p2"),
    Particle(6e10,7.48e10,1.2956e11,-2.579e4,1.489e4,"p3"),
    Particle(4e6,7.48e10,-1.2956e11,2.579e4,1.489e4,"p4"),
    Particle(2e6,1.481e11,0,0,2.948e4,"p5"),
    Particle(2e6,1.511e11,0,0,3.008e4,"p6"),
    Particle(2e6,-1.496e11,0,0,-2.978e4,"p7")]; for 400 steps and dt=10000"
    # Second Parameter i for a 3 body system was:
    "particles=[ Particle(2e30,0,0,0,-1.5e6,"particle 1"),
      Particle(1e30,1.5e7,0,0,3e6,"particle 2"),
      Particle(6e24,1e7,0,0,0.5e6,"particle 3")
      ]; for 3200 steps and dt=0.1."
```

The plots that I got was:

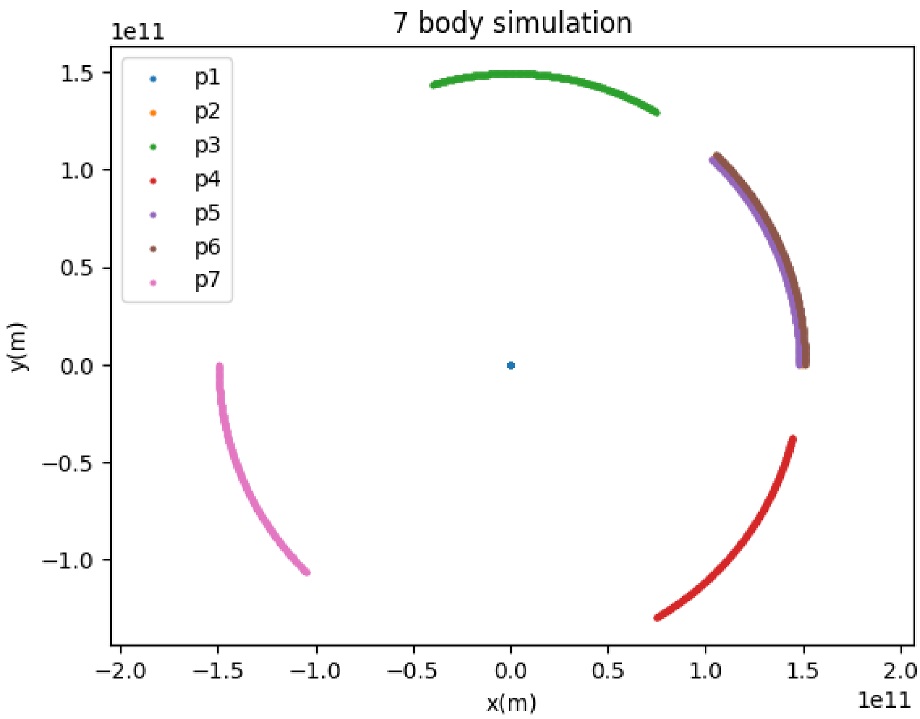


Figure 3.1: Example of a 7 Body system

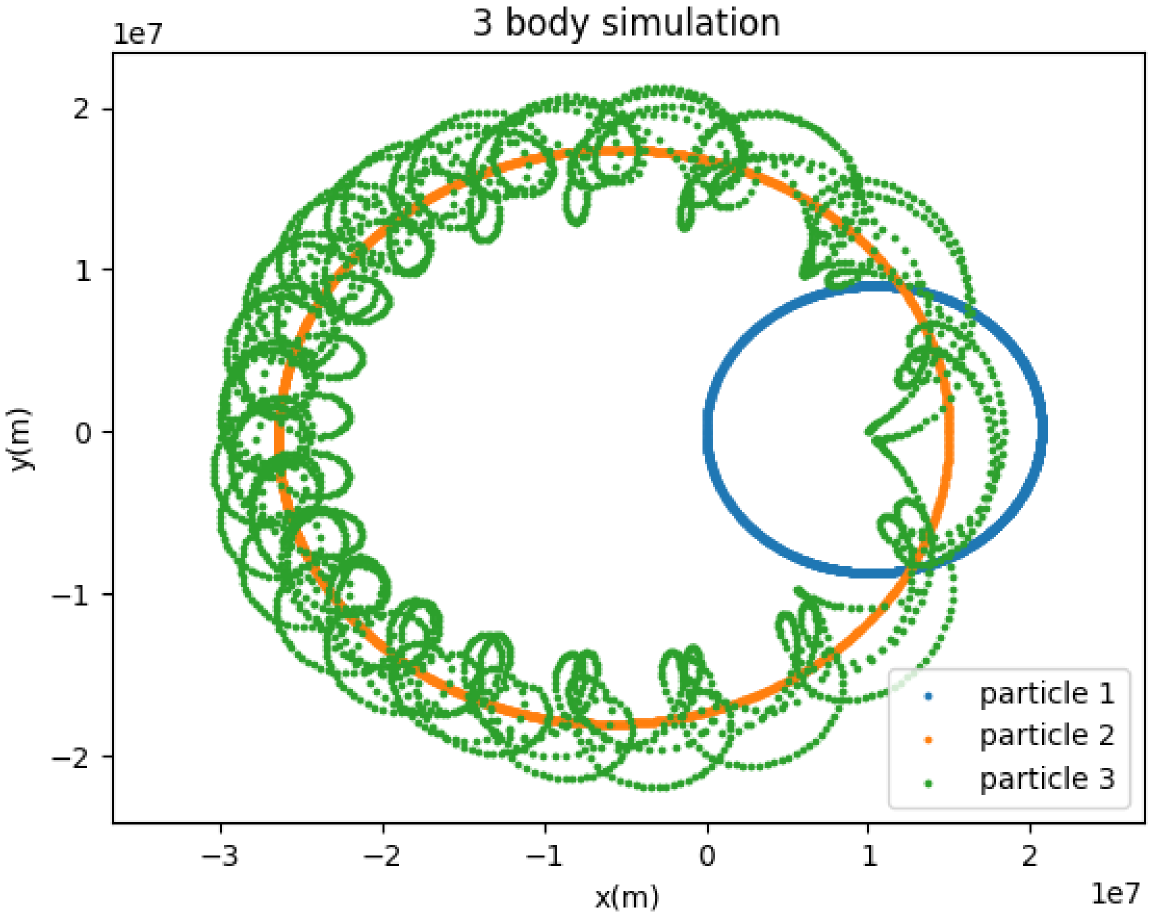


Figure 3.2: Example of a 3 Body system

4. Restricted 3 Body

Horshoe and tadpole orbits are two interesting types of orbits that occur in celestial mechanics, particularly in the context of the three-body problem, where three celestial bodies (typically a planet, a moon, and a larger body like a star) influence each other's motion through gravitational interactions. Both types of orbits involve the interaction of two smaller bodies around a larger central body.

4.1 Horseshoe Orbits

A horseshoe orbit is a type of three-body orbit in which a smaller celestial body, such as a moon, oscillates back and forth around the Lagrange points L3 and L4 of a larger body's orbit. The Lagrange points are stable points of equilibrium where the gravitational forces from the two larger bodies (planet and star) balance out the centripetal force of the smaller body, allowing it to remain in a relatively stable position. In a horseshoe orbit, the smaller body initially orbits the larger body in the same direction, but gravitational perturbations from the other larger body cause it to gradually slow down and move away from its original orbit. As a result, the smaller body eventually reaches the Lagrange point L3 or L4. However, due to conservation of angular momentum, it cannot simply stay at the Lagrange point and continues to move along a path that appears to trace the shape of a horseshoe relative to the larger body.

The smaller body then continues this oscillatory motion, crossing in front of and behind the larger body, forming a "horseshoe" shape in the reference frame of the larger body.

4.2 Tadpole Orbit

A tadpole orbit is another type of three-body orbit in which a smaller celestial body orbits a larger body in a peculiar pattern, which can resemble the shape of a tadpole. In a tadpole orbit, the smaller body orbits around one of the Lagrange points L4 or L5 of the larger body's orbit.

At these Lagrange points, the gravitational forces from both the larger body and the other smaller body create a stable equilibrium position, allowing the smaller body to remain relatively stationary with respect to the larger body. However, the presence of the larger body's gravitational field causes the smaller body to slowly drift in a tadpole-like pattern around the Lagrange point, following an elongated looping path.

Both horseshoe and tadpole orbits are fascinating examples of complex orbital dynamics arising from gravitational interactions in the three-body problem. These types of orbits are important in celestial mechanics and have been observed in various astronomical systems involving multiple celestial bodies.

```python
class Particle:
    def __init__(self, mass, pos,vel):
        # For defining the mass, co-ordinates and velocity components for the particles in a list.
    def update(self,dt,f):
        # The velocity and position will get updated according to the force and for every timestep
class Grav_Simulation:
    def __init__(self,n,mass,pos,v):
        # Here 'n' is the no. of particles for which we are simulating. So basically I'm also writing the code for N-body system but in 'class' format now
    def force(self,i):
        # Here we are maintaining a 2d numpy array which stores force in both x and y directions for every particle
    def final_update(self,dt):
        # This is the final update function which takes only one aregument and we will take the force for the above update function from the force function that we defined
    def simulating(self,dt,num_steps):
        # Here we are simulating for 'dt' timestep and no. of steps as num_steps. But the issue here is we are simulating horshoe and tadpole orbits, which involve the simulation in frame of C.O.M of the other 2 massive bodies. So for that we are maintaining an array which stores the x and y coordinates in COM frame
```

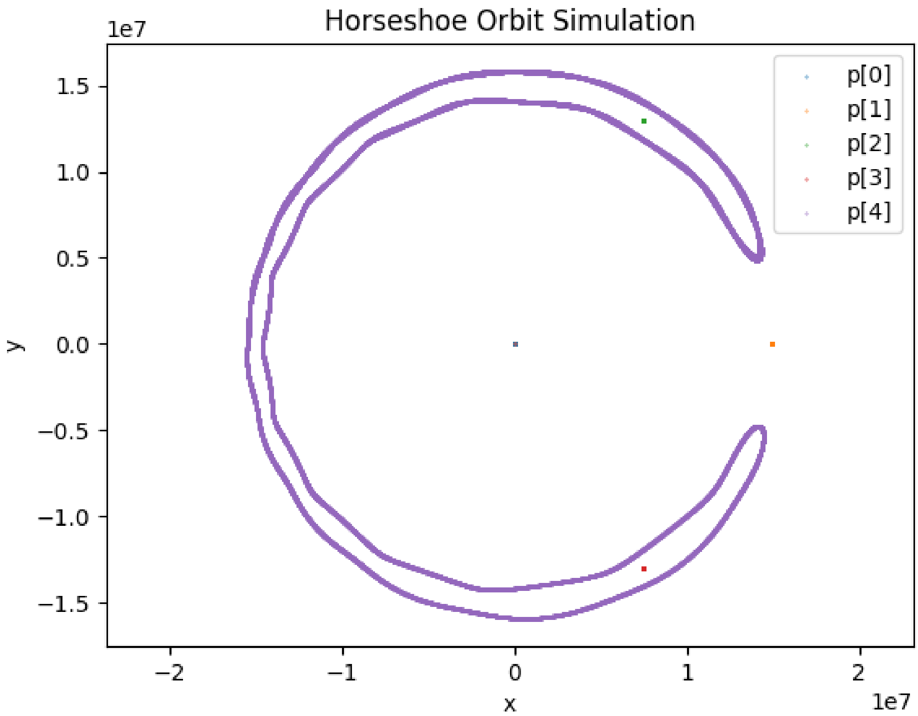


Figure 4.1: Horshoe Orbit

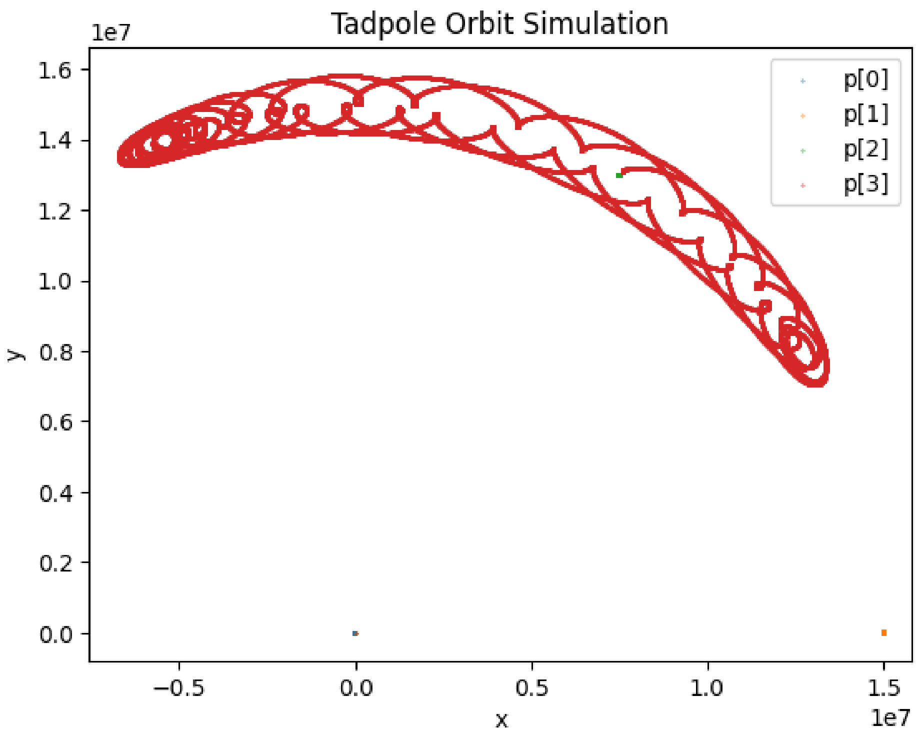


Figure 4.2: Tadpole Orbit

5. Analemma

An analemma is a fascinating astronomical phenomenon that results from plotting the position of the Sun in the sky at the same time of day throughout the year. When observed at the same local solar time each day (e.g., noon), the Sun's apparent position traces out a figure-eight-shaped curve known as the analemma.

The analemma is primarily a result of the combination of two distinct motions of the Earth: its axial tilt and its elliptical orbit around the Sun. The axial tilt causes the Sun's declination (angle above or below the celestial equator) to vary throughout the year, while the elliptical orbit results in a varying speed of the Earth along its orbital path.

The analemma is an essential tool for understanding and predicting the Sun's position in the sky at various times of the year, which is particularly important for activities such as navigation, astronomy, and agriculture. It is often depicted on sundials and globes to aid in timekeeping and celestial observations.

The shape of the analemma can vary depending on the observer's latitude on Earth. For observers at the equator, the analemma is a straight line, while at higher latitudes, it takes on a more pronounced figure-eight shape.

The two lobes of the analemma correspond to the two solstices (summer and winter) when the Sun reaches its highest and lowest points in the sky. The center point of the figure-eight corresponds to the two equinoxes (spring and autumn) when the Sun is directly above the equator.

The analemma is a beautiful visual representation of the Earth's position relative to the Sun throughout the year, demonstrating the intricacies of our planet's orbital motion and axial tilt. It serves as a reminder of the cyclic and ever-changing nature of our solar system and its impact on our daily lives.

Now solving for Elliptical Orbit(for ex. Sun-Earth system) by polar co-ordinates equation for an ellipse:

$$r = \frac{a(1-e^2)}{1+e\cos\theta}$$

Also by Kepler's second law:

$$\frac{1}{2}r^2\omega = \frac{\pi a^2\sqrt{1-e^2}}{T}$$

and then using above equation of ellipse in polar co-ordinates , plus writing $\omega = \frac{d\theta}{dt}$. We get,

$$\int_{\theta_0}^{\theta} \frac{d\theta}{(1+e\cos\theta^2)} = \frac{2\pi t}{T\sqrt{(1-e^2)^3}}$$

Solving for earth like planet, which traverses an orbit whose eccentricity is very small, we can use binomial approximation. We get an equation;

$$c(\theta - \theta_0 - 2e(\sin\theta - \sin\theta_0)) = t$$

where $c \approx 58.1$ days and $e = 0.0167$ and also taking $\theta_0 = 75^\circ$ as the angular position of spring equinox and t as the measured time from spring equinox day (21st March).

```
def equation_to_solve(theta,t,theta_0,c,e):
    # Wrote a function which will return the above equation

# Then i maintained an array which will store the solutions of above equation for different values of t. I used an in-built function "fsolve" in scipy.optimize

# As we know that shape of Analemma is independent of latitude, so i took my observer at North pole to get values of azimuth and altitude for each day.

# Now as we know the ecliptic latitude, and by solving for the co-ordinates by spherical trigonometry. We can get now the desired Alt and Az.

# Plotting Alt vs Az, we get the analemma
```

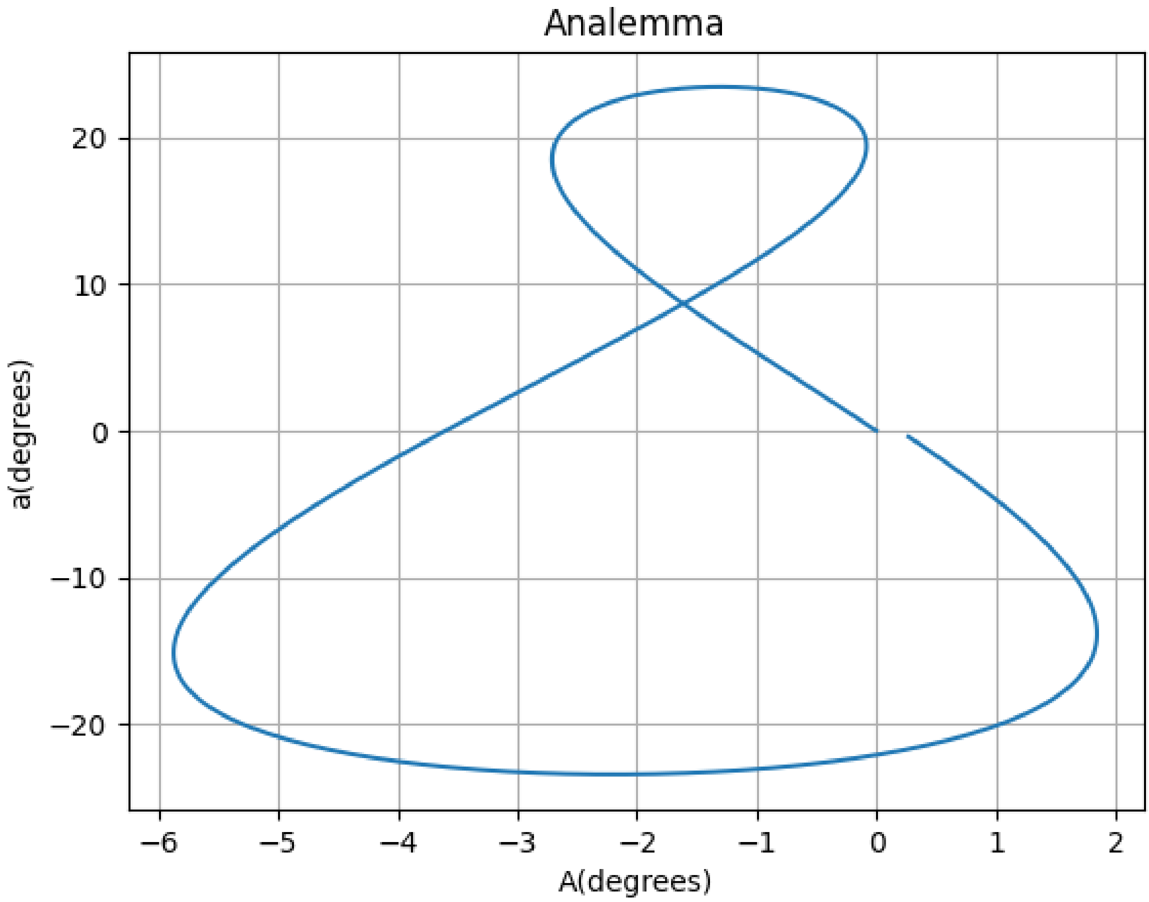


Figure 5.1: *Analemma*

6. Shadow of Black Hole

The "shadow" of a black hole refers to a dark region in space directly surrounding the black hole where light cannot escape. It is a critical feature associated with the strong gravitational pull of a black hole.

When a black hole is situated in a region with a background of bright light sources, such as stars or gas clouds, the extreme gravitational field of the black hole causes the light from those background sources to be deflected and bent. As a result, some of the light rays are bent inward toward the black hole, while others are bent outward, creating a distorted and magnified image of the background light sources.

The region where light rays are bent inward and never escape is the "shadow" of the black hole. This shadow appears as a dark, circular region against the backdrop of bright light sources. The size and shape of the shadow depend on the mass and spin of the black hole, as well as the observer's vantage point.

The first direct observation of the shadow of a black hole was made possible by the Event Horizon Telescope (EHT) project. In April 2019, the EHT collaboration released the first-ever image of the shadow of the supermassive black hole at the center of the galaxy M87. This groundbreaking observation provided strong evidence for the existence of black holes and confirmed some predictions of Einstein's theory of general relativity.

```
In[1] : def def_eq(r,th):
            # This returns a differential equation that we got by solving the equation ↪
                for energy of photon.

        # I solved the above differential equation using 'odeint' , and got a particular ↪
            solution or trajectory for the photon


In[2] : class Photon:
            def ___init__(self,pos,v):
                # It will define the velocity and position in a list for the photon

            def update :
                # It will update the pos and vel of photons for given acceleration. And ↪
```

```
                this acceleration, we know by differentiating the potential(not the ↩
                effective, but other than due to centrifugal force)
        class Traj:
            def __init__(self,n,pos,v):
                # Here I am just declaring the pos and vel of every photon and 'n' is the↩
                    no. of photon trajectories we want to plot

            def simulation(self,dt,n_steps):
                # Finally simulating for 'n_steps' no. of steps and with timestep 'dt'.
```

The two plots that I got, first for the special case in 'In[1]' and other for different photons in 'In[2]'.

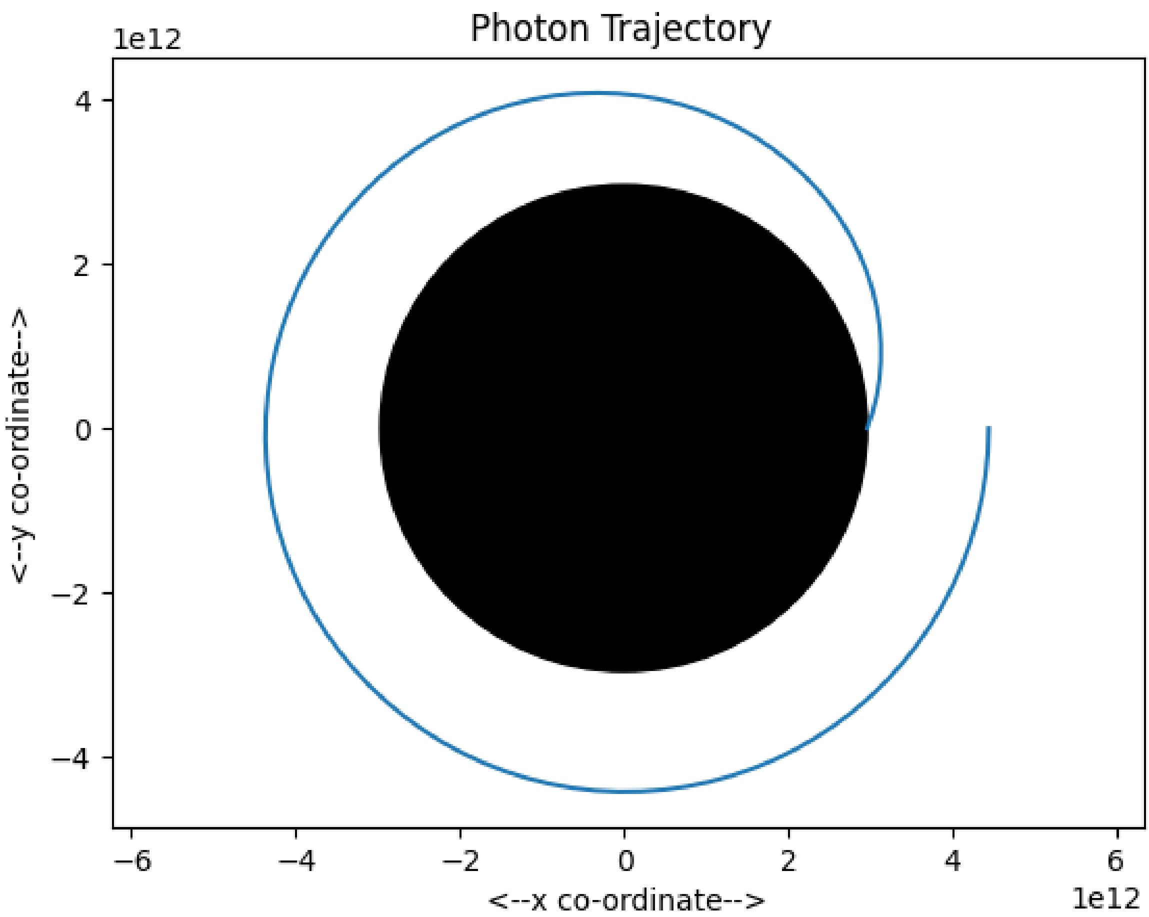


Figure 6.1: A solution of photon trajectory

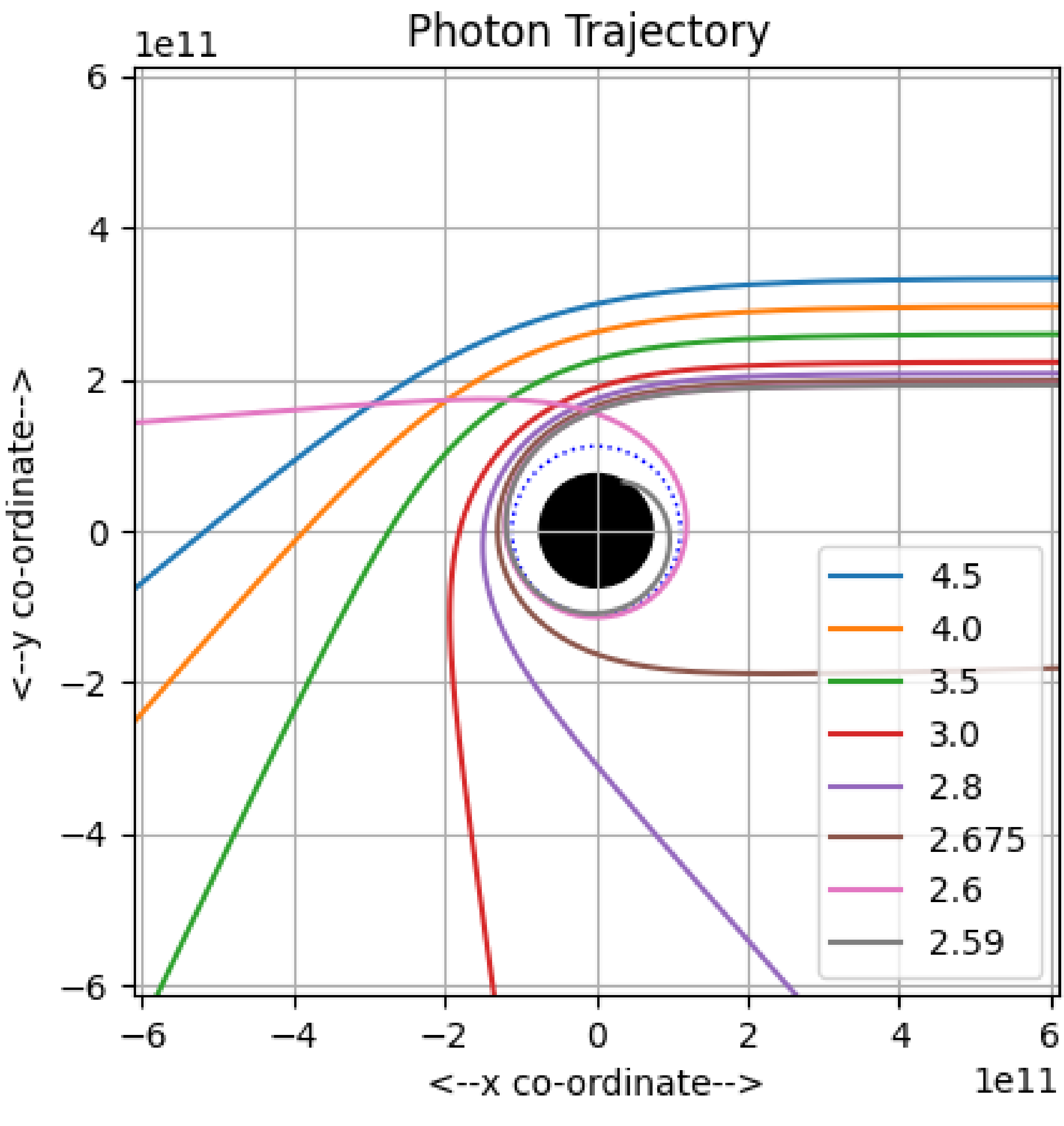


Figure 6.2: Photon trajectories for different values of impact parameter

7. Kirkwood Gaps

So there are some gaps in the asteroid belt of our Solar System. These gaps are called the Kirkwood Gaps. These gaps are caused primarily by the interaction of asteroids by Jupiter(as this is the most massive planet in the Solar System).

These interactions lead to orbital resonances of asteroid orbit with Jupiter's orbit. This happens when the period of the asteroid's orbit and Jupiter's orbit are related in a simple fractional ratio.

These gravitational perturbations can accumulate over time, causing the asteroids to experience changes in their orbits. Over millions of years, these perturbations can become significant and, in some cases, can lead to the ejection of asteroids from the resonance region. As a result, the regions corresponding to the orbital resonances become sparsely populated, creating the Kirkwood gaps.

The below figure gives a good feel of what is actually the Kirkwood gaps;

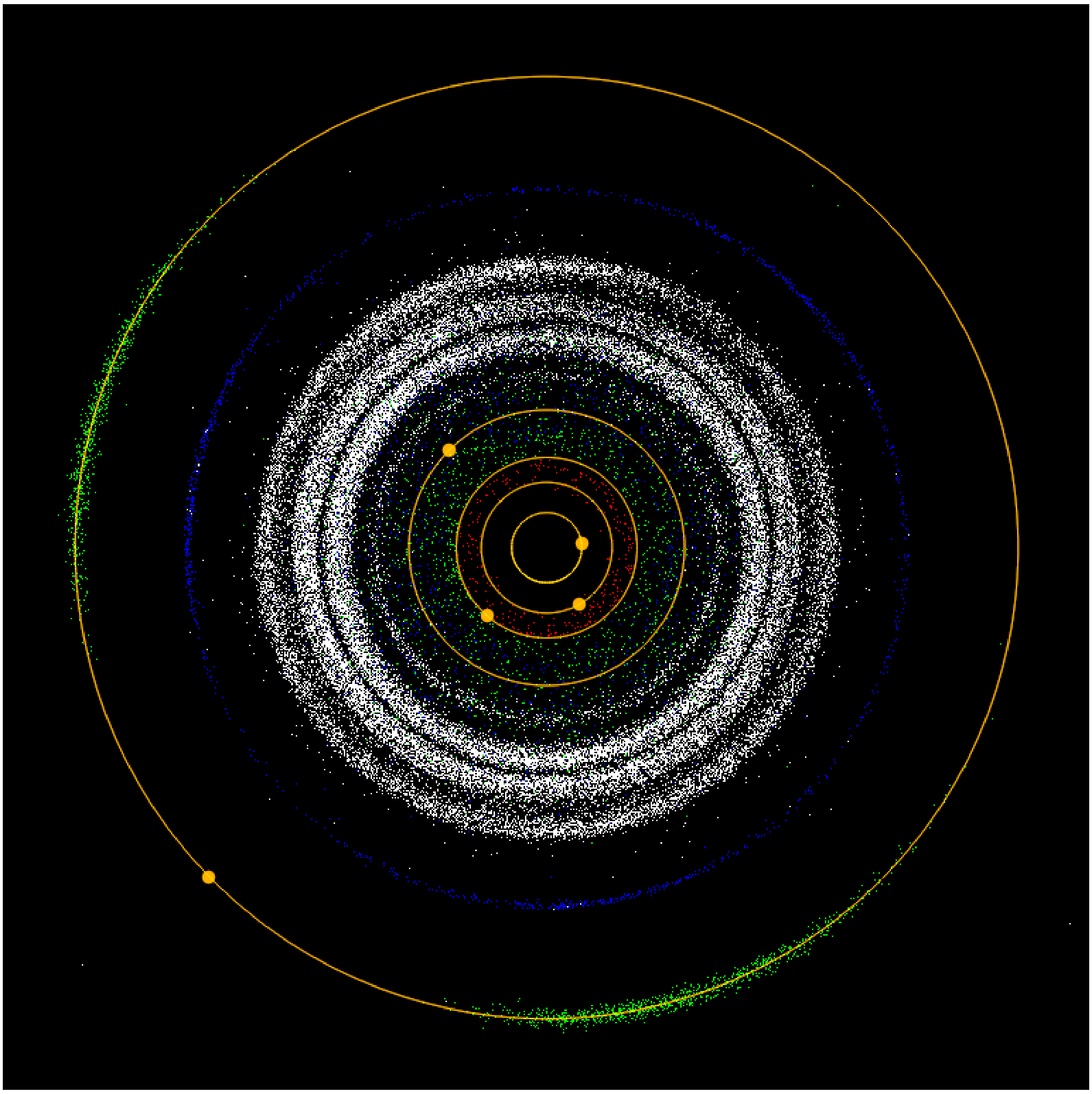


Figure 7.1: *Kirkwood gaps. Main-belt asteroids are white. Inside the main belt, there are the Atens (red), Apollos (green), and Amors (blue). Outside the main belt are the Hildas (blue) and the Trojans (green). The Kirkwood gaps are visible in the main belt.*

8. References

1. An introduction to mechanics by Kleppner.
2. The Circular Restricted Three-Body Problem by Richard Frnka.
3. Analemma
4. Kirkwood Gap
5. Shadow of a Black Hole