KRITTIKA
IITB

DECEMBER 2024

KCAP -24

MODELLING GRAVITATIONAL SYSTEMS

PRESENTED BY

GURU JAHNAVI MADANA
ENGINEERING PHYSICS - 24B1809

MENTORED BY

VISHWAAJITH . N .K
MALAY KEDIA

TABLE OF CONTENTS

SUMMARY OF THE PROJECT

For 4 weeks in winter break - 2024

This report documents my week-by-week progress in developing simulations for celestial mechanics, as part of the KCAP project. The primary objective was to understand and model the dynamics of celestial bodies under gravitational influence, starting with basic Python programming and advancing to complex n-body simulations.

The journey began with learning Python basics, which laid the foundation for creating simulations. By Week 2, I moved on to implementing a two-body problem, focusing on visualizing the trajectories of two interacting masses. Over the next weeks, the project evolved into handling more intricate problems such as the three-body problem, chaotic systems, and eventually large-scale n-body simulations with up to 100 bodies.

Through this project, I not only deepened my understanding of celestial mechanics but also honed my coding skills by adopting modular and object-oriented programming approaches. Each simulation brought unique challenges, from computational efficiency to creating visually engaging animations, all of which contributed to a rewarding learning experience.

This report captures the code snippets, simulation outputs, and insights gained during this process, providing a comprehensive account of my efforts and results.

WEEK-01

BASICS OF PYTHON

OBJECTIVE - Learn Python fundamentals to prepare for celestial mechanics coding tasks.

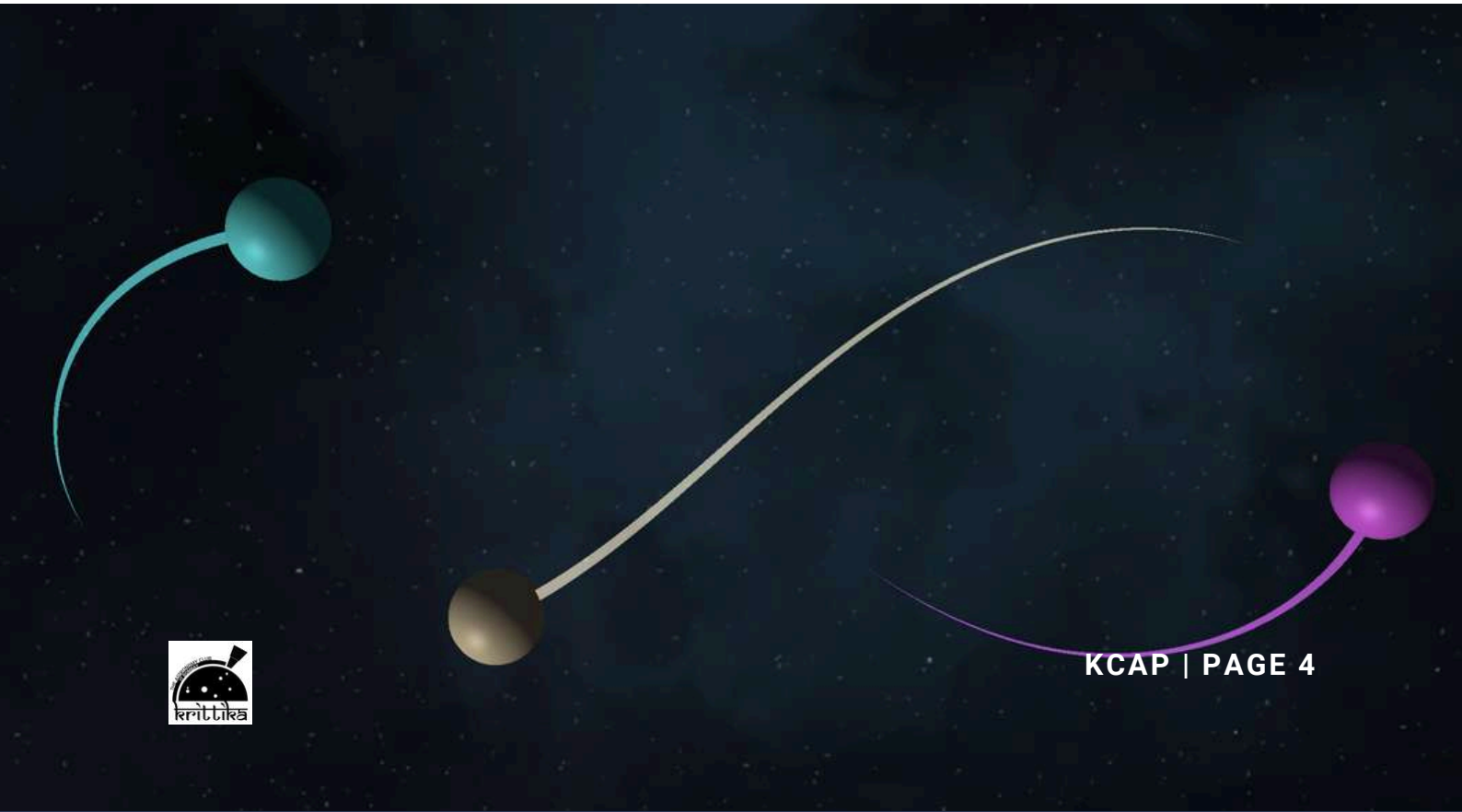
Activities:

- Practiced Python syntax, functions, and basic programming constructs.
- Developed confidence in array manipulations, loops, and conditionals to prepare for simulation tasks.
- Got basics in libraries such as NumPy and Mathplot

Outcome:

- Achieved proficiency in Python necessary for upcoming tasks.
- No simulations or animations created this week.

The google colab notebook- https://colab.research.google.com/drive/1N653Zu8839XtySHLbDaLcxWydsA30OjE



WEEK -02

INTRODUCTION TO CELESTIAL MECHANICS

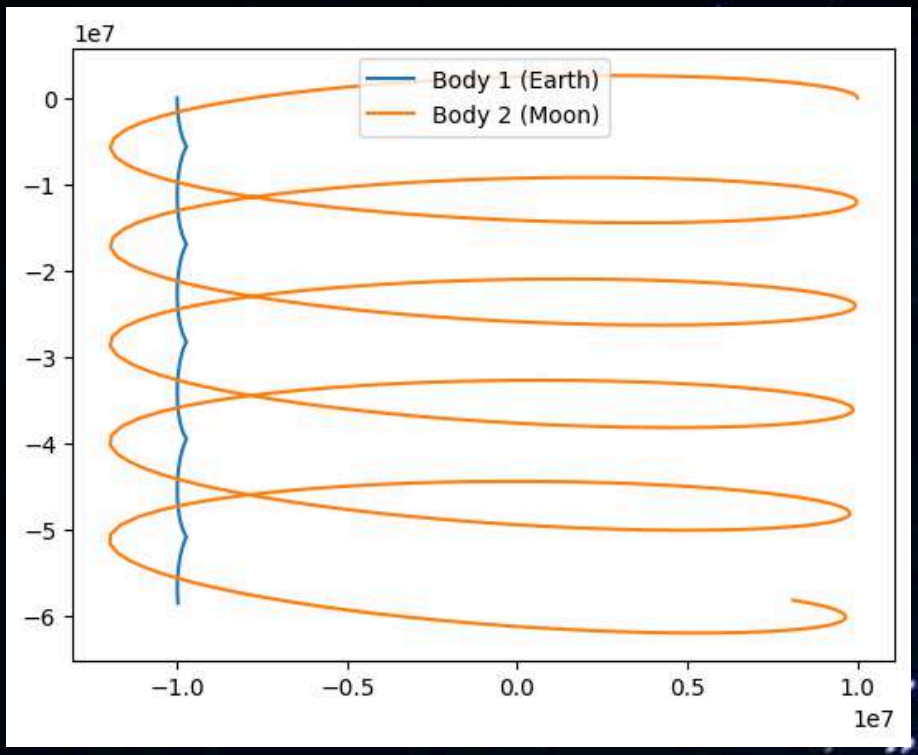
Objective: Understand celestial mechanics and develop a 2-body simulation.

Activities:

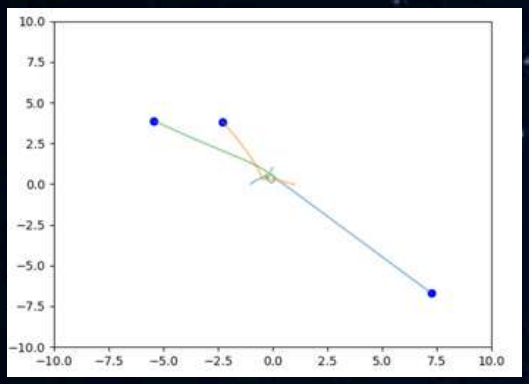
1. Two-body simulation:
   - Coded the simulation of two bodies under mutual gravitational forces.
   - Inputs: Masses and initial velocities of the bodies.
   - Output: Animated the trajectories of the two bodies.

```
# Define the two-body system
class TwoBodySystem:
    def __init__(self, m1, m2, r1, r2, v1, v2):
        self.m1 = m1  # Mass of body 1
        self.m2 = m2  # Mass of body 2
        self.r1 = np.array(r1, dtype=float)  # Position of body 1
        self.r2 = np.array(r2, dtype=float)  # Position of body 2
        self.v1 = np.array(v1, dtype=float)  # Velocity of body 1
        self.v2 = np.array(v2, dtype=float)  # Velocity of body 2

    def compute_accelerations(self):
        # Calculate the vector from body 1 to body 2
        r12 = self.r2 - self.r1
        distance = np.linalg.norm(r12)
        force_magnitude = gravitational_constant * self.m1 * self.m2 / distance**2
        force_direction = r12 / distance

        # Acceleration due to gravitational force
        a1 = force_magnitude / self.m1 * force_direction
        a2 = -force_magnitude / self.m2 * force_direction
        return a1, a2

    def update(self, dt):
        # Compute accelerations
        a1, a2 = self.compute_accelerations()

        # Update velocities
        self.v1 += a1 * dt
        self.v2 += a2 * dt

        # Update positions
        self.r1 += self.v1 * dt
        self.r2 += self.v2 * dt
```

The simulation / trajectory image-



2. Tried to generalize the code to multi-body simulation, but it was not so perfect and didn't work for the earth, moon and sun systems, it was able to work only with the masses = [1,1,1] and G=1, which resulted into a chaotic trajectory in the beginning and later dispersed. The possible errors were the time-scale and the solving of the equation. Followed them up in next weeks.



3. Also went through the theory provided by the mentors - celestial mechanic from the krittika resources-

the code for the two body simulation-
https://colab.research.google.com/github/GJ-007-sage/Astronomy/blob/main/2_body_simulation.ipynb#scrollTo=pHf4DKY-8q2e

WEEK-03

EXTENSION TO THREE-BODY AND REDUCED THREE-BODY PROBLEMS

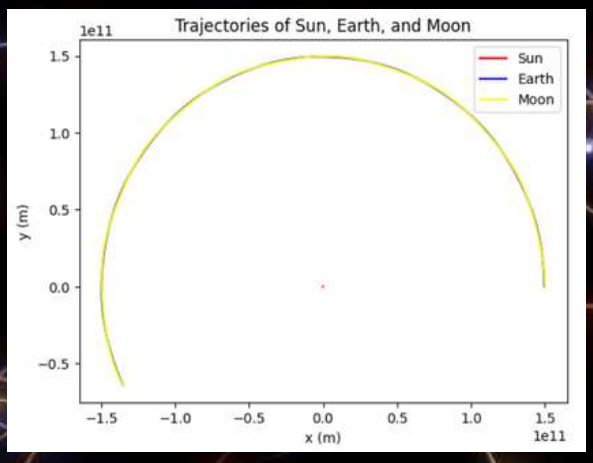
OBJECTIVE - Simulate 3-body motion and analyze chaotic dynamics.

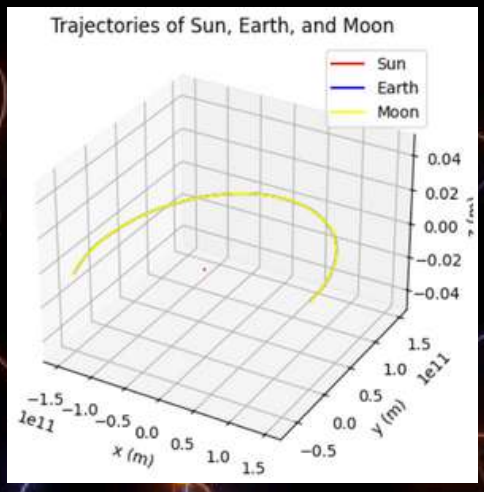
Activities:

- Generalization to 3-body motion:
- Simulated motion in 3D for a three-body system.
- Sanity check performed with Sun-Earth-Moon system.

```
class ThreeBodySystem:
    def __init__(self, bodies):
        self.bodies = bodies

    def compute_forces(self):
        for body in self.bodies:
            body.force = np.zeros(2, dtype=float)
        for i, body1 in enumerate(self.bodies):
            for j, body2 in enumerate(self.bodies):
                if i != j:
                    r = body2.position - body1.position
                    distance = np.linalg.norm(r)
                    if distance != 0:
                        force_magnitude = G * body1.mass * body2.mass / distance**2
                        body1.force += force_magnitude * r / distance

    def update(self, dt):
        self.compute_forces()
        for body in self.bodies:
            body.update_velocity(dt)
            body.update_position(dt)
```

- I have defined a class for the body also , the results for the Earth-moon-sun simulation is below-



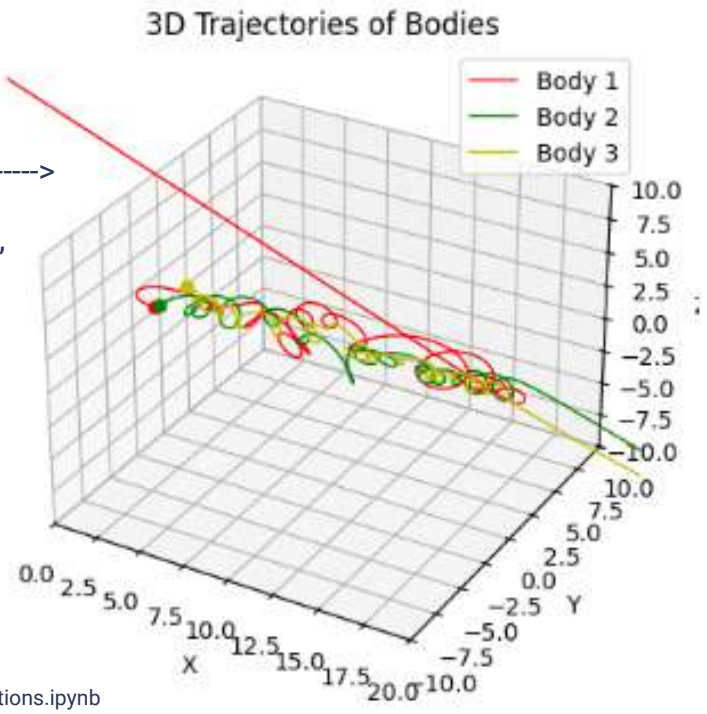


Chaotic behavior study:

- Experimented with random values to observe chaotic motion

```
body1 = Body(10, [-1, 1, -1.1], [-3, 0
body2 = Body(20, [0, 0, 0], [0, 0, 0])
body3 = Body(30, [1, 1, 1.2], [3, 0, 0
```

the above values and G = 1 , gives ---------->

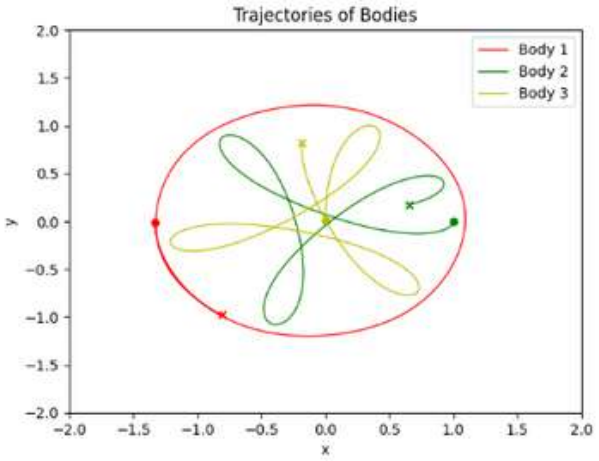


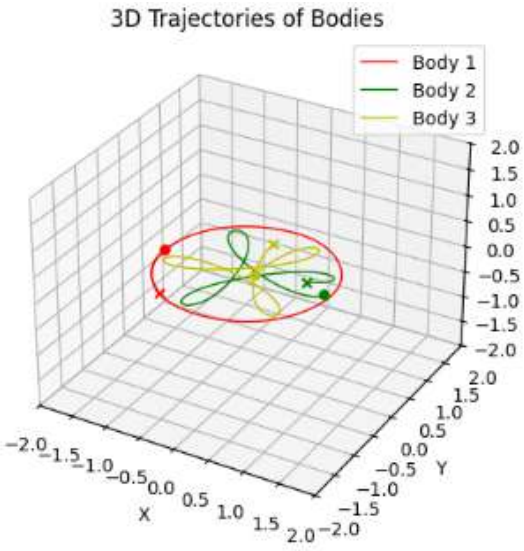
The system is highly sensitive to initial conditions, meaning small changes can lead to drastically different outcomes. This unpredictability is a hallmark of chaos theory, making long-term predictions extremely difficult. These systems don't settle into simple orbits and instead exhibit complex, seemingly random motion over time.

the code can be referred in-
https://github.com/GJ-007-sage/Astronomy/blob/main/3_body_simulations.ipynb

```
m1 = 1.0124     x1 = -1.32962     v1 = -0.88963
m2 = 0.9968     x2 = 1            v2 = -0.28501
m3 =1           x3= 0             v3 = -(m1*v1 + m2*v2)
```

taking the values of y co-ordinates of position =0 and the given velocities are y components, the remaining velocity components are taken to be 0, G = 1





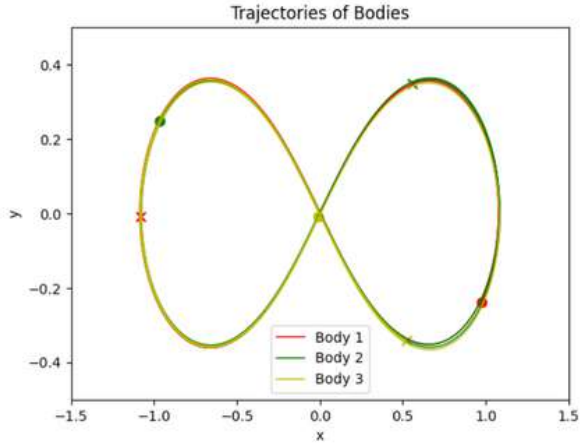
the values are taken from the website -
https://www.sciencedirect.com/science/article/pii/S1384107622000598

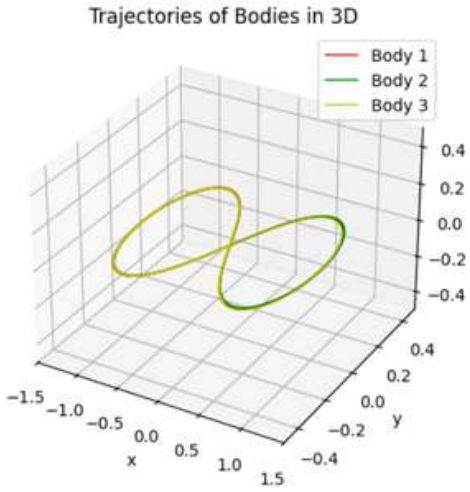
Famous "8" Orbit:

- Simulated the classic "8" trajectory.

In the previous code of 3-body simulation, by substituting the below vales and G = 1 , gave me the following result

```
body1 = Body(1.0, [0.97000436, -0.24308753], [0.93240737/2, 0.86473/2])
body2 = Body(1.0, [-0.97000436, 0.24308753], [0.93240737/2, 0.86473/2])
body3 = Body(1.0, [0.0, 0.0], [-0.93240737, -0.86473146])
```





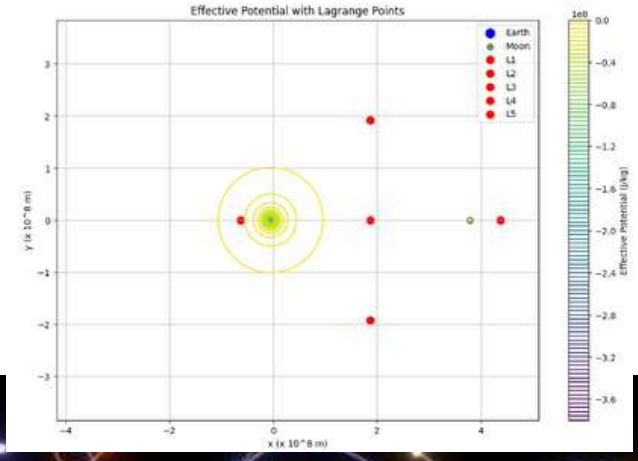
the code can be referred in- https://github.com/GJ-007-sage/Astronomy/blob/main/3_body_simulation.ipynb

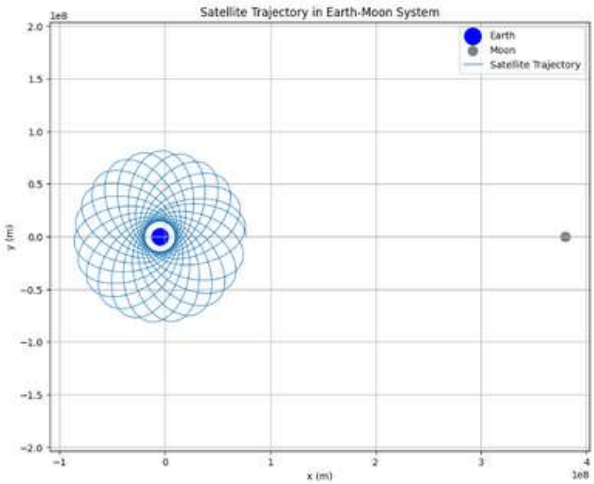
Reduced 3-body problem:

- Simulated motion under the reduced 3-body framework.
- Plotted the contour of the reduced gravitational field.

tried this for various reduced three body systems like - Earth-Moon-Satellite, Sun-Jupiter-asteroid, binary stars-exoplanet

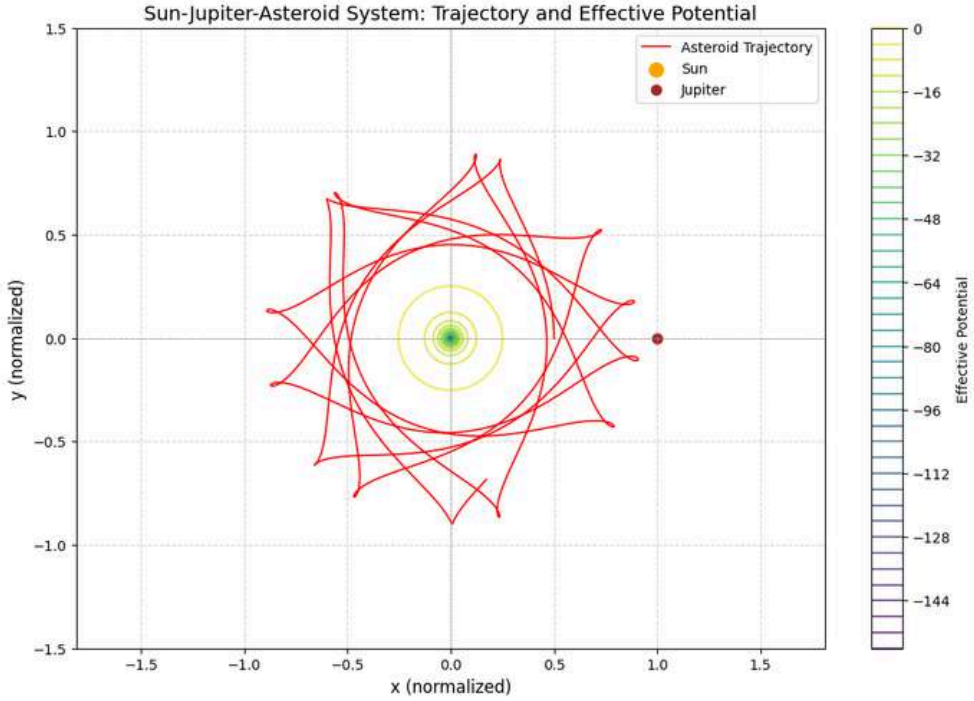
```
def earth_moon_satellite(t, state):
    x, y, vx, vy = state

    # Distances to Earth and Moon
    r1 = np.sqrt((x + mu * d_earth_moon)**2 + y**2)
    r2 = np.sqrt((x - (1 - mu) * d_earth_moon)**2 + y**2)

    # Accelerations
    ax = 2 * omega * vy + omega**2 * x - G * M_earth * (x + mu * d_earth_moon) / r1**3 \
         - G * M_moon * (x - (1 - mu) * d_earth_moon) / r2**3
    ay = -2 * omega * vx + omega**2 * y - G * M_earth * y / r1**3 - G * M_moon * y / r2**3
```





the plot includes the Lagrange points. The Lagrange points align well with the expected saddle points (L1, L2, L3) and stable minima (L4, L5) of the potential field.

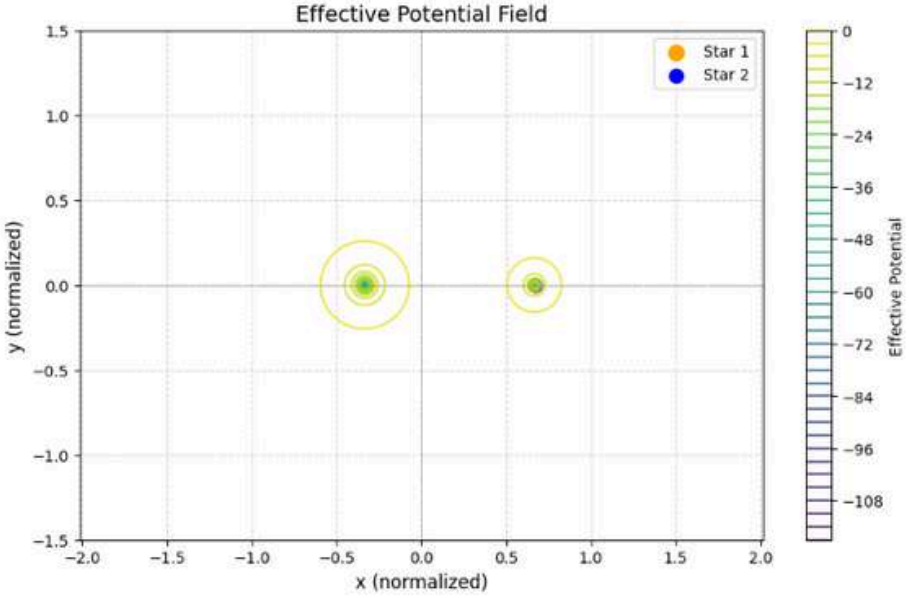
The Jupiter - Sun - Asteroid system was done as follows;

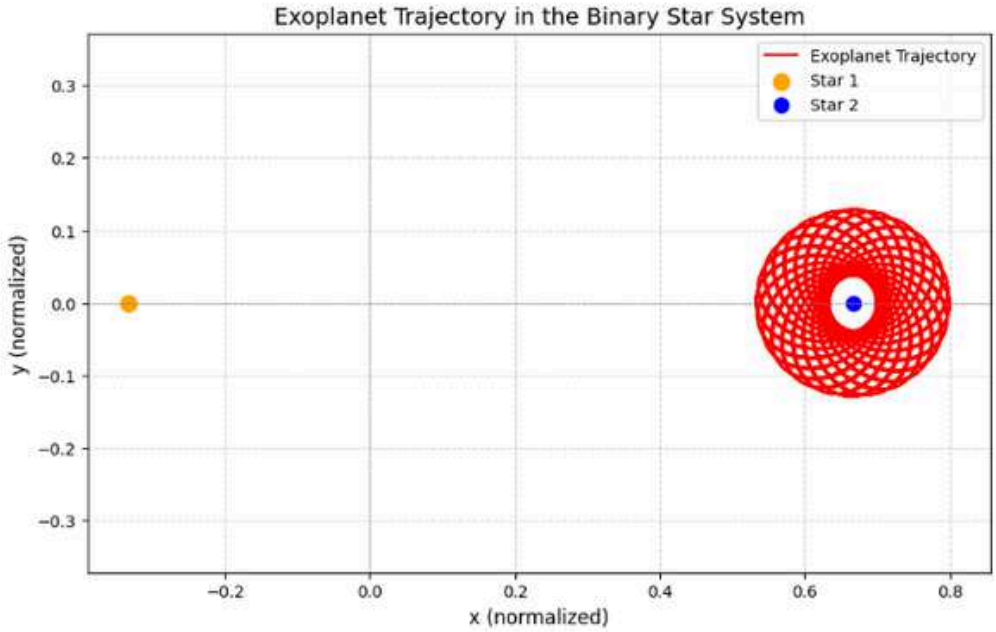


The asteroid exhibits a quasi-periodic or chaotic motion due to the gravitational resonance between the Sun and Jupiter. Such motion is commonly observed in systems involving a small body (asteroid) under the influence of two massive primary bodies (Sun and Jupiter).

The background contours (yellow- gradient) represent the effective potential field, showing equipotential lines. The asteroid's trajectory overlaps the effective potential landscape, particularly around Jupiter's influence region, which explains the orbital perturbations.

The Binary star - exoplanet system was done as follows;





The red curve represents a stable, periodic orbit around the center of mass of the binary stars. The exoplanet orbits primarily around the heavier star (in blue), indicating that it is closer to the more massive component of the system.
Star 2 (blue)- The more massive star, which dominates the gravitational influence on the exoplanet. The slightly elliptical trajectory suggests a stable resonance between the gravitational pull of the two stars

I have implemented the code in SI units and normalized gravitational constant to make the complex calculation much simpler. It is helpful to work with dimensionless units, focusing on the dynamics of the system relative to one another, instead of using the full, physical values. (Normalized G = 1)

- Outcome:
  - Successfully simulated 3-body motion, famous "8" orbit, and reduced 3-body problems.
  - Gained insights into chaotic dynamics and closed curves in the reduced field.

The code for reduced 3-body simulations-
https://github.com/GJ-007-sage/Astronomy/blob/main/reduced_3body_systems.ipynb

WEEK -04

N-BODY SIMULATIONS

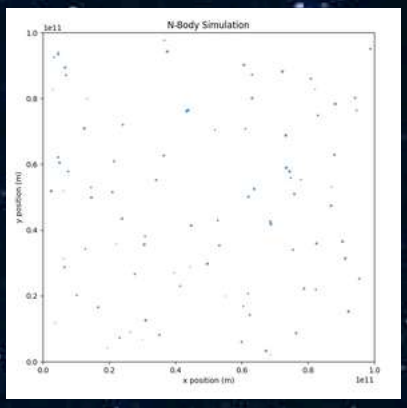
- Objective: Simulate n-body systems (e.g., n=100).

Activities:

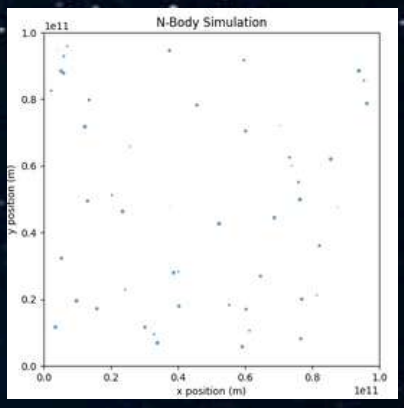
- Developed and tested an n-body simulation framework for large numbers (e.g., 100 bodies).
- Optimized code for performance to handle computational load.
- Animated the trajectories of 100 bodies under gravitational forces.

Outcome:

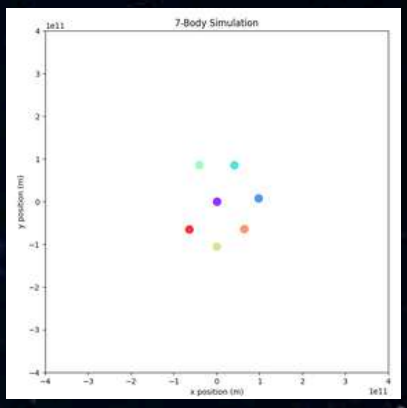
- Successfully simulated and visualized n-body systems.
- Gained understanding of large-scale celestial dynamics.



N = 100



N = 50



N = 7

refer the code-
https://github.com/GJ-007-sage/Astronomy/blob/main/n_body_simulation.ipynb

It looks better if the code is opened in the google colab and run to look at the animations made

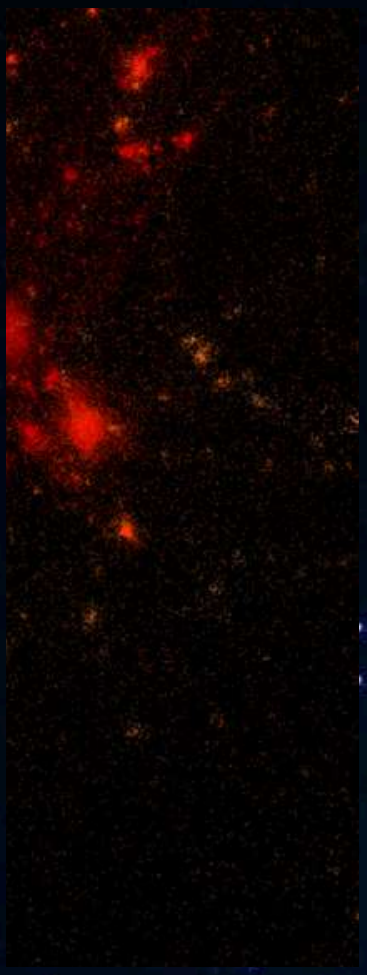
SOURCES & REFERENCES

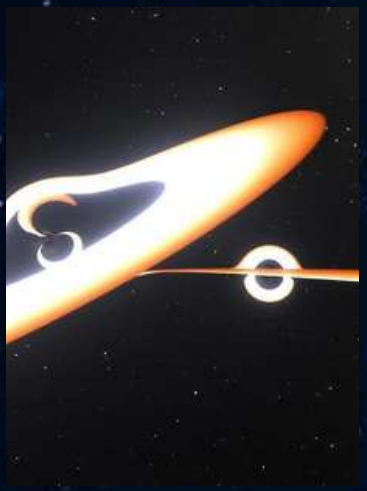
- Google
- ChatGPT
- Mentors
- https://en.wikipedia.org/wiki/Three-body_problem
- https://www.sciencedirect.com/science/article/pii/S1384107622000598
- https://blbadger.github.io/3-body-problem.html
- https://www.youtube.com/watch?v=FXkH9-4EN_8
- https://www.youtube.com/watch?v=otRtUiCcCh4
- https://www.youtube.com/watch?v=UC40kDpAI8M
- https://towardsdatascience.com/use-python-to-create-two-body-orbits-a68aed78099c
- https://krittikaiitb.github.io/Learners'_Space/Basic_modules/Celestial%20Mechanics%20and%20Gravity.pdf
- https://en.wikipedia.org/wiki/N-body_simulation

Code part -

- https://colab.research.google.com/drive/1N653Zu8839XtySHLbDaLcxWydsA30OjE
- GitHub - https://github.com/GJ-007-sage/Astronomy

(has all the code i have used , for simulations run in colab)





CONCLUSION

Throughout this project, I have gained invaluable insights into celestial mechanics and simulation programming. From mastering Python basics to diving deep into the complexities of multi-body simulations, the journey has been both intellectually stimulating and rewarding. One of the most amusing aspects was observing the chaotic yet mesmerizing behavior of celestial bodies, especially in the 3-body and n-body simulations. It was fascinating to see how small changes in initial conditions could lead to drastically different results, showcasing the inherent unpredictability of such systems.

The ability to visualize the trajectories of celestial bodies, especially when experimenting with various orbital patterns, was a highly engaging experience. The process of generalizing code to handle increasingly complex systems, and then organizing it into modular classes for better scalability, felt both empowering and creative.

Overall, this project has not only deepened my understanding of celestial mechanics but also honed my skills in Python programming, problem-solving, and simulation techniques. The knowledge gained and the exciting discoveries made along the way will serve as a solid foundation for future projects in both coding and scientific analysis.