**Aim:**

To simulate and visualize the motion of celestial bodies in a two-body and three-body system using Newton's law of gravity and generate animations to illustrate the dynamic behavior of these systems.

**Objectives:**

1. To implement the equations of motion for a two-body and three-body problem using Python.
2. To solve the ordinary differential equations (ODEs) governing the motion of the bodies using numerical methods.
3. To create and visualize animations that depict the motion of celestial bodies over time.
4. To analyze the stability and behavior of the systems under different initial conditions.

**Summary:**

This project focuses on simulating the motion of celestial bodies in two distinct scenarios: a two-body problem (e.g., Earth and Sun) and a three-body problem, which is more complex and chaotic. By applying Newton's law of gravitation, the motion equations for these systems are derived and solved using numerical integration methods. The project culminates in animations that demonstrate the orbital paths of the bodies over time, providing insight into their dynamic interactions.

**Tools and Libraries Used:**

- **Python**: Programming language for implementation.
- **NumPy**: For numerical computations.
- **Matplotlib**: For plotting and creating visualizations.
- **SciPy**: For solving differential equations.
- **Pillow**: For saving animations as GIFs.
- **Jupyter Lab**: Development environment.

**Procedure:**

1. **Import Libraries**: The necessary Python libraries are imported, including NumPy for numerical operations, Matplotlib for plotting, and SciPy for solving differential equations.
2. **Define Constants and Initial Conditions**: The masses of the bodies and their initial positions and velocities are set up. For the two-body problem, the Sun and Earth are modeled, with initial conditions chosen to simulate Earth's orbit around the Sun. For the three-body problem, initial conditions are chosen based on research papers for specific configurations.
3. **Formulate the Equations of Motion**: Newton's law of gravity is applied to derive the equations of motion for the bodies. The second-order differential equations are converted into a system of first-order ODEs to facilitate numerical solving.

4. **Solve the ODEs**: The odeint function from SciPy is used to solve the two-body problem, while the solve_ivp function with the DOP853 solver is used for the three-body problem. These functions compute the positions and velocities of the bodies over time.
5. **Plot the Results**: The positions of the bodies are plotted to visualize their orbits over time. The time is converted to years to provide a real-world context.
6. **Create Animations**: Using Matplotlib's animation functionality, the motion of the bodies is animated, showing their orbits dynamically. The animations are saved as GIFs using the Pillow library.

**Highlights:**

- **Dimensional Analysis**: The problem is converted to dimensionless form to make computations more efficient and to generalize the solution.
- **Numerical Integration**: The ODEs are solved using SciPy's odeint and solve_ivp functions, ensuring accurate simulations for complex systems like the three-body problem.
- **Stable Configurations**: The project explores specific initial conditions that lead to stable orbits in the three-body problem, a rare and intriguing phenomenon.

**Conclusion:**

The project successfully simulates the two-body and three-body problems, providing visual insights into the gravitational interactions between celestial bodies. The animations created illustrate the complexity and beauty of orbital dynamics, highlighting the differences between the relatively simple two-body system and the chaotic nature of the three-body system. The approach and methods used can be extended to explore more complex systems and configurations in celestial mechanics.