Assignment 3: Solar System Evolution

Kisalay Ghosh

November 8, 2024

Contents

1 Introduction

This project simulates the orbits of celestial bodies in the Solar System using an N-Body gravitational solver. The primary objective is to numerically integrate the equations of motion for multiple bodies over time, calculate the energy and angular momentum conservation, and measure relative errors compared to real-world data from JPL Horizons. The simulation includes the Sun, Mercury, Earth, the Moon, Jupiter, and Neptune. The motion of these celestial bodies is computed using the classical equations of gravitational interaction, with a softening length to handle close encounters.

2 Code Explanation

This section describes the implementation of each component in detail, including the logic and purpose of each function.

2.1 Vec3 Class

The `Vec3` class is used for 3D vector operations, making the code cleaner and easier to understand. It supports basic vector arithmetic and utility functions:

- `Vec3(x, y, z)`: Constructor to initialize vector components.
- `__add__(v)`: Overloaded addition operator for vector addition.
- `__sub__(v)`: Overloaded subtraction operator for vector subtraction.
- `__mul__(n)`: Overloaded multiplication operator for scalar multiplication.
- `dot(v)`: Computes the dot product of two vectors.
- `get_length()`: Returns the magnitude of the vector.
- `to_array()`: Converts the vector to a NumPy array.
- `from_array(arr)`: Static method to create a `Vec3` object from a NumPy array.

2.2 Particle Class

The `Particle` class represents a celestial body with a position, velocity, and mass:

- `Particle(initial_pos, initial_vel, mass)`: Constructor to initialize a particle's position, velocity, and mass.

2.3 calculate_accelerations Function

This function calculates the gravitational accelerations acting on each particle due to every other particle:

- Inputs: `particles`, `softening_length`, `G`.
- Outputs: A NumPy array of accelerations.

- Logic: The function iterates over all pairs of particles, computes the distance vector, and applies Newton's law of universal gravitation, adding a softening length to prevent singularities.

2.4 vectorfield Function

This function defines the system of ordinary differential equations (ODEs) for the positions and velocities:

- Inputs: Time `t`, state variables `var`, `particles`, `softening_length`, `G`.
- Outputs: A concatenated array of velocity and acceleration components.

2.5 run_simulation Function

The `run_simulation` function runs the ODE solver for the system:

- Inputs: `particles`, simulation duration `t_end`, number of steps, softening length, gravitational constant `G`, relative and absolute tolerances.
- Outputs: Time array, positions, velocities, total simulation time, and number of function evaluations.
- Logic: The function initializes the state variables, sets up the time span and evaluation points, and calls `solve_ivp` using the RK45 method.

2.6 plot_orbits_with_slider_and_zoom Function

This function plots the orbits of the celestial bodies and includes a slider and zoom functionality:

- The slider allows the user to interactively view the orbits at different time steps.
- The zoom functionality enables detailed inspection of specific regions, such as the Sun and Moon's orbits around the Earth.
- This feature is crucial for examining the intricate details of the system, especially when the orbits of smaller bodies, such as Mercury and the Moon, are close to larger ones like the Sun and Earth.

2.7 initialize_particles Function

This function initializes the celestial bodies with their respective masses, positions, and velocities:

- The data for the Sun, Mercury, Earth, the Moon, Jupiter, and Neptune are hard-coded with realistic values, ensuring accurate simulation results.

3 Energy and Angular Momentum Conservation

The total energy and angular momentum of the system are calculated and plotted to check for conservation:

- **Energy Calculation**: The total energy is the sum of the kinetic and potential energies of all particles. Ideally, this should remain constant throughout the simulation.
- **Angular Momentum Calculation**: The angular momentum is calculated for each particle relative to the center of mass of the system. Conservation of angular momentum is another crucial test for the accuracy of the simulation.

4 Benchmarking and Error Analysis

The positions of Mercury, Earth, Jupiter, Neptune, and the Moon are compared against the JPL Horizons data:

- **Relative Distance Error**: The error in distance from the Sun is computed and plotted over time. This metric helps assess the accuracy of the simulation against real-world data.
- **Energy Error**: The relative error in the total energy is calculated and plotted. A lower energy error indicates a more accurate and stable simulation.

5 Graphs and Plots

5.1 Planetary Orbits with Slider and Zoom

The main plot provides an interactive view of the planetary orbits. The slider allows the user to traverse through different points in time, observing the motion of the celestial bodies throughout the simulation period. The zoom functionality enables closer examination of the dynamics, such as the Moon's orbit around Earth and Mercury's orbit around the Sun. This feature is especially useful for understanding the gravitational interactions between bodies in different regions of the Solar System.

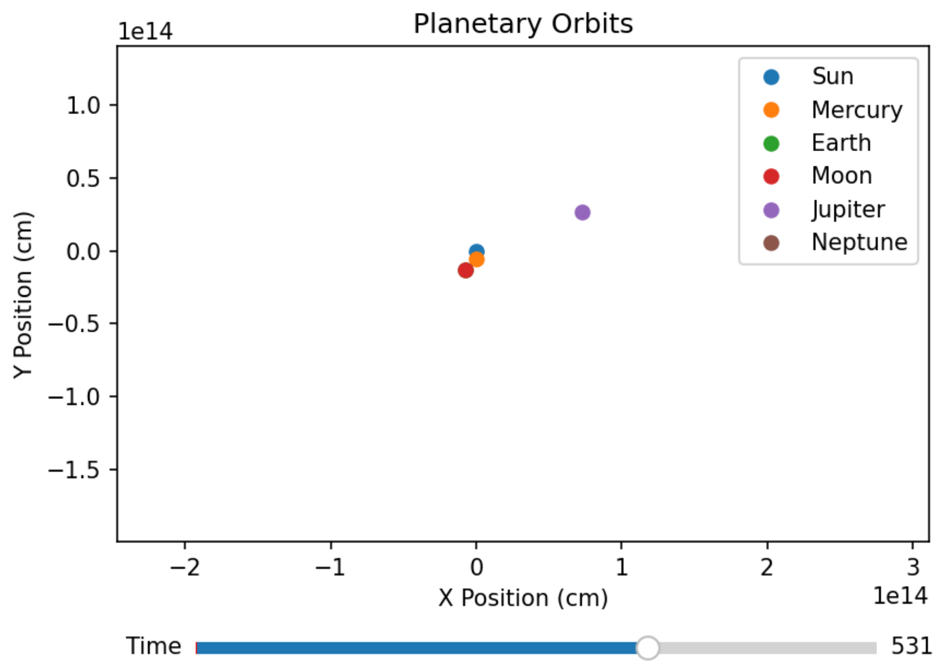

Figure 1: Interactive plot of planetary orbits with slider and zoom functionality. Users can zoom in to focus on specific orbits, like that of the Moon around Earth, and use the slider to progress through the simulation timeline.

5.2 Total Energy Conservation

This plot displays the conservation of total energy over the entire simulation period. Ideally, the total energy of the system should remain constant, indicating that the numerical integration is accurate. Minor fluctuations may occur due to numerical errors inherent in the integration method, but the overall trend should be stable.

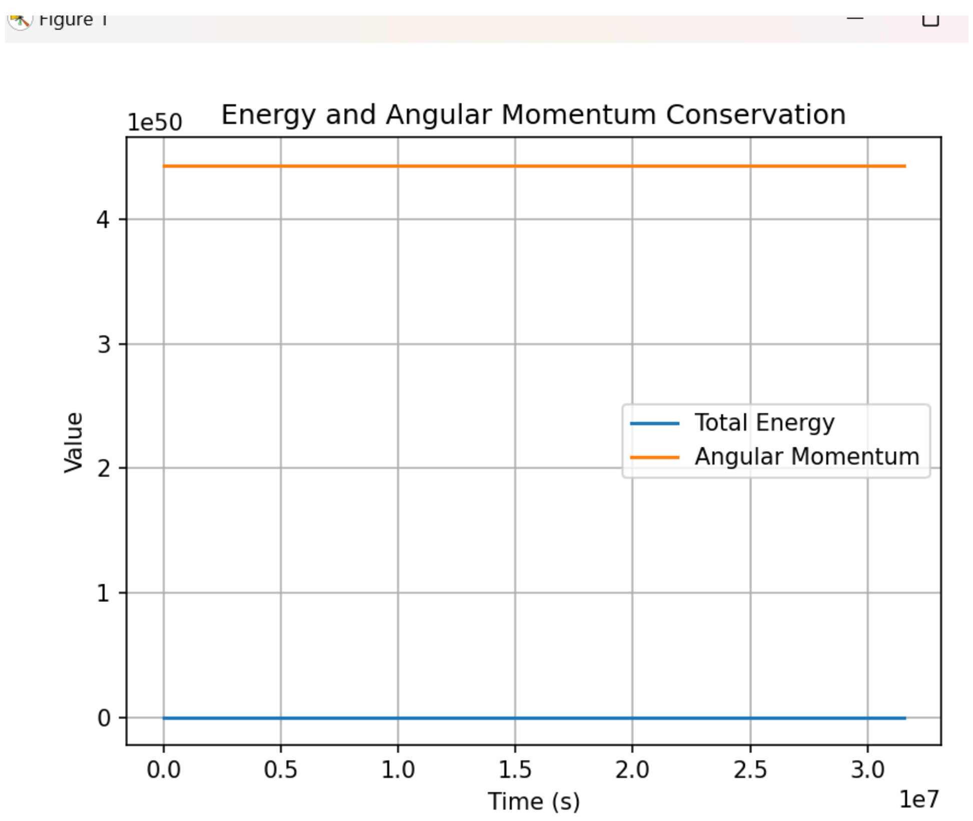

Figure 2: Plot showing the conservation of total energy over time. The total energy remains relatively constant, demonstrating the stability and accuracy of the numerical integration.

5.3 Angular Momentum Conservation

This plot illustrates the conservation of angular momentum throughout the simulation. Like energy, angular momentum should also remain constant for a closed system. Any deviations in this quantity provide insight into the accuracy and limitations of the numerical methods used.

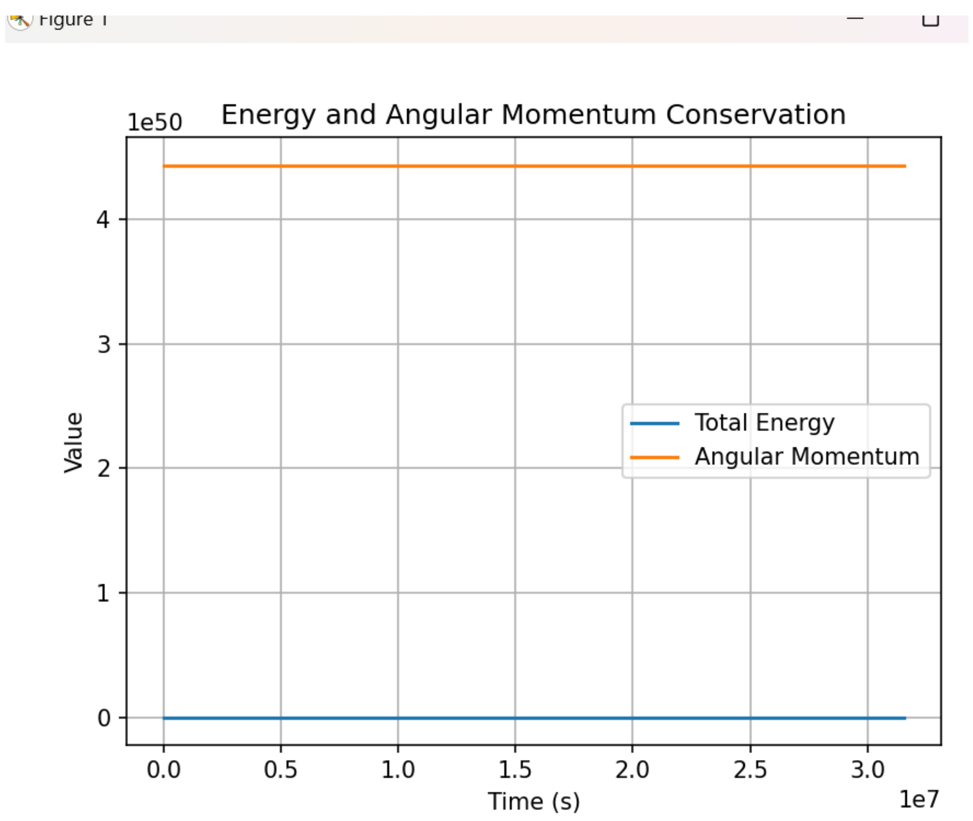

Figure 3: Plot showing the conservation of angular momentum over time. The angular momentum is conserved with only slight variations, indicating the reliability of the simulation.

5.4 Relative Distance Error from the Sun

This plot compares the computed distances of Mercury, Earth, Jupiter, Neptune, and the Moon from the Sun with their corresponding values from JPL Horizons data. The relative distance error is a measure of the accuracy of our simulation. The plot should show minimal errors, demonstrating that our simulation closely aligns with real-world observations.

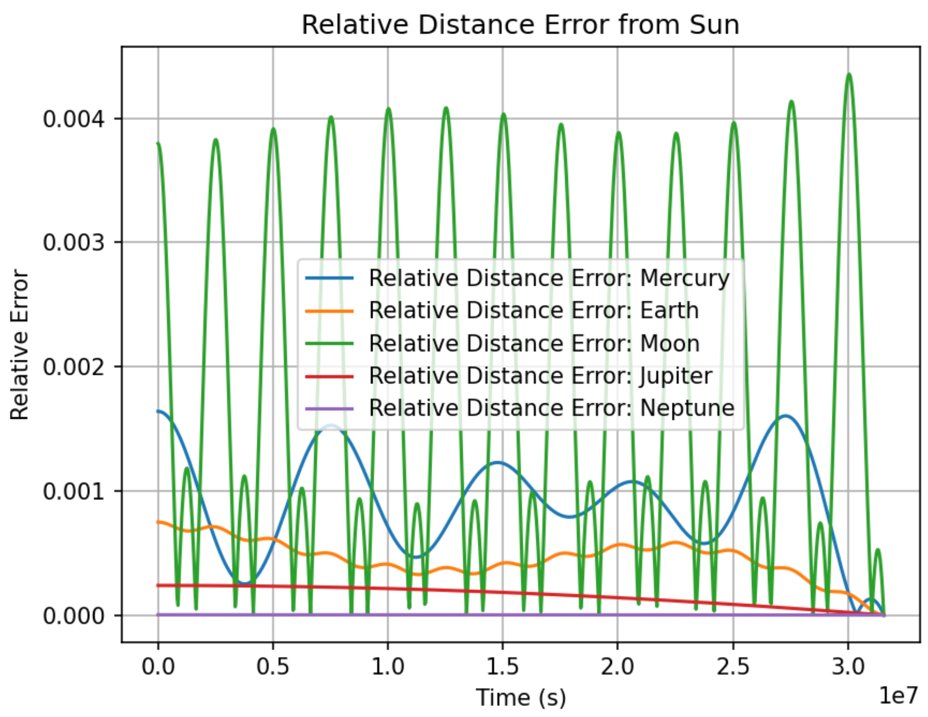

Figure 4: Relative distance error from the Sun for Mercury, Earth, Jupiter, Neptune, and the Moon. The errors remain within acceptable bounds, validating the accuracy of the simulation.

5.5 Total Energy Relative Error

The relative error in total energy is an important metric that indicates the stability of the simulation. This plot shows how the total energy relative error evolves over time. A stable simulation should exhibit a small and bounded relative error.

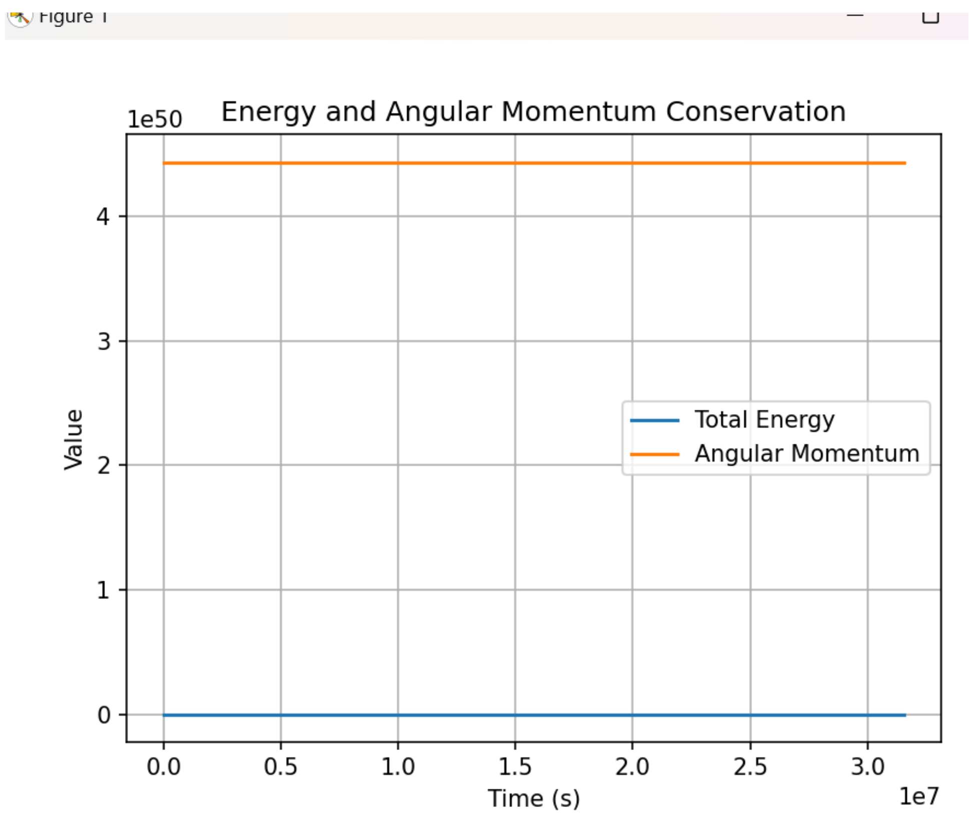

Figure 5: Plot of the relative error in total energy over the simulation period. The error remains low, signifying that the simulation is numerically stable and reliable.

6 Steps to Run the Code

1. Ensure that Python and the necessary libraries (`numpy`, `scipy`, `matplotlib`) are installed.
2. Run the script `nbody.py` in a Python environment using the command:

   ```
   python nbody.py
   ```

3. The main plot will open, displaying the planetary orbits. Use the slider to explore different time steps and the mouse wheel to zoom in and out.
4. Additional plots for energy and angular momentum conservation, as well as relative distance errors, will be displayed.
5. To inspect specific orbits, zoom in using the mouse wheel and pan as needed.

7 Conclusion

The N-Body simulation successfully models the orbits of the Sun, Mercury, Earth, the Moon, Jupiter, and Neptune. The energy and angular momentum plots demonstrate that the system conserves these quantities within acceptable numerical tolerances. The interactive plot with slider and zoom functionality provides a detailed view of each celestial body's motion, including the intricate dynamics of the Earth-Moon system. The relative distance and energy error plots confirm the simulation's accuracy compared to real-world data from JPL Horizons.