

Project Report

Introduction

The objective of this project was to simulate the orbital motion of Earth and Jupiter around the Sun, incorporating the gravitational interactions between all three bodies. This simulation aims to provide an understanding of how these interactions influence the trajectories of the planets. By leveraging numerical methods to solve the governing equations of motion, we can visualize and analyze the resulting orbits. We will also compare the differences in the orbits of Earth and Jupiter when their mutual gravitational influences are included versus when they are ignored.

Understanding the orbital dynamics of planetary bodies is fundamental to astrophysics and has far-reaching implications. The motion of planets around the Sun, governed by gravitational forces, forms the basis for understanding more complex celestial workings. Simulating these orbits provides valuable insights into:

1. **Celestial Mechanics:** The principles governing the motion of planets and other astronomical objects, which are crucial for predicting future positions and understanding the past configurations of the solar system.
2. **Space Exploration:** Accurate models of planetary motion are essential for planning space missions, including satellite launches, interplanetary travel, and spacecraft navigation.
3. **Astrophysical Research:** Simulations help in studying the stability of planetary systems, the effects of gravitational perturbations, and the potential for resonances and chaotic behavior in multi-body systems.
4. **Educational Purposes:** Visualizing the orbits and understanding the underlying physics enhances the comprehension of fundamental concepts in physics and astronomy for students and enthusiasts.

By incorporating both Newton's laws of motion and Kepler's laws of planetary motion, this simulation bridges classical mechanics with observational astronomy, providing a holistic view of planetary dynamics. The comparison between two-body and three-body simulations highlights the complexity and beauty of gravitational interactions in our solar system.

Physical Concepts

The simulation of Earth and Jupiter's orbits around the Sun relies heavily on fundamental physical concepts. Newton's Laws provide the framework for calculating the forces, accelerations, and subsequent motion of the planets. Kepler's Laws describe the general behavior and characteristics of the planetary orbits, which emerge naturally from the numerical integration of Newton's laws.

By combining these laws, we create a realistic model of planetary motion that captures the essential dynamics of gravitational interactions in a multi-body system. The use of these physical principles ensures that the simulation not only predicts the paths of the planets but also provides insights into the nature of their orbital mechanics.

Newton's Laws of Motion

Second Law ($F=ma$)

Newton's Second Law of Motion states that the acceleration of an object is directly proportional to the net force acting on it and inversely proportional to its mass. This can be expressed as:

$$a = \frac{F}{M}$$

Where:

- a is the acceleration of the object.
- F is the net force acting on the object.
- M is the mass of the object.

In the context of our simulation, Newton's Second Law is used to calculate the acceleration of Earth and Jupiter due to gravitational forces. By knowing the forces acting on these planets, we can determine their accelerations and subsequently update their velocities and positions over time.

Universal Gravitation

Newton's Law of Universal Gravitation describes the attractive force between two masses. It is given by:

$$F = G \frac{M_1 M_2}{r^2}$$

Where:

- F is the gravitational force between two masses.
- G is the gravitational constant ($6.67430 \times 10^{-11} \text{m}^3 \text{kg}^{-1} \text{s}^{-2}$).
- M_1 and M_2 are the masses of the two objects.
- r is the distance between the centers of the two masses.

In our simulation, we use this law to compute the gravitational forces exerted by the Sun on Earth and Jupiter, as well as the gravitational force between Earth and Jupiter. These forces are then used to determine the accelerations of the planets, which are essential for updating their velocities and positions according to Newton's Second Law.

Kepler's Laws of Planetary Motion

First Law: Ellipses

Kepler's First Law states that the orbit of a planet around the Sun is an ellipse, with the Sun located at one of the two foci. This means that the distance between the planet and the Sun varies over the course of the orbit.

In our simulation, while we calculate the gravitational interactions and update positions numerically, the resulting orbits of the planets naturally conform to elliptical shapes, as described by this law. The elliptical nature of the orbits emerges from the gravitational forces and the initial conditions set for the simulation.

Second Law: Equal Areas

Kepler's Second Law, or the Law of Equal Areas, states that a line segment joining a planet and the Sun sweeps out equal areas during equal intervals of time. This can be expressed as:

$$\frac{dA}{dt} = \text{constant}$$

Where:

- A is the area swept out by the line segment joining the planet and the Sun.
- t is time.

This law implies that planets move faster when they are closer to the Sun and slower when they are farther from the Sun. In our simulation, by updating the positions and velocities of the planets based on gravitational forces, we inherently respect this law. The changing speeds of Earth and Jupiter as they move along their orbits reflect this principle.

Third Law (Harmonies)

Kepler's Third Law states that the square of the orbital period of a planet is directly proportional to the cube of the semi-major axis of its orbit. This can be expressed as:

$$\frac{T^2}{a^3} = \text{constant}$$

Where:

- T is the orbital period of the planet.
- a is the semi-major axis of the orbit.

In our simulation, we can compare the simulated orbital periods and the distances of Earth and Jupiter from the Sun to verify that they satisfy Kepler's Third Law. This comparison provides a check on the accuracy of our numerical methods and initial conditions.

Mathematical Principles

This project employs several key mathematical principles to simulate the orbits of Earth and Jupiter around the Sun. The primary mathematical concepts involved include differential equations, numerical integration methods, and vector mathematics.

Differential Equations

Differential equations describe the relationship between the rates of change of quantities and the quantities themselves. In the context of orbital mechanics, we use differential equations to model the motion of the planets under gravitational forces. The relevant equations are second-order differential equations for the positions of the planets:

$$\frac{d^2r}{dt^2} = a$$

Where:

- r is the position vector of the planet.
- a is the acceleration vector.

Given Newton's Second Law and the Law of Universal Gravitation, the acceleration a is determined by the gravitational forces acting on the planet:

$$a = G \frac{M_{sun} r}{r^3} + G \frac{M_{other} (r_{other} - r)}{(r_{other} - r)^3}$$

where:

- G is the gravitational constant.
- M_{sun} is the mass of the Sun.
- M_{other} is the mass of the other planet (e.g., Jupiter when calculating Earth's acceleration).
- r is the position vector of the planet.
- r_{other} is the position vector of the other planet.

Numerical Integration Methods

To solve these differential equations numerically, we use Euler's method, a simple yet effective numerical integration technique. Euler's method approximates the solution of a differential equation by discretizing time into small steps and updating the quantities iteratively. The basic principle of Euler's method is to use the current value of a quantity and its derivative to estimate the next value.

For our simulation, we use Euler's method to update the positions and velocities of the planets at each time step:

1. **Velocity Update:** Using the current acceleration to update the velocity:

$$v_{new} = v_{current} + a \cdot \Delta t$$

2. **Position Update:** Using the updated velocity to update the position:

$$r_{new} = r_{current} + v \cdot \Delta t$$

where:

- v is the velocity vector.
- a is the acceleration vector.
- Δt is the time step.

Vector Mathematics

Vector mathematics is essential for accurately modeling the positions, velocities, and accelerations of the planets in three-dimensional space. Key vector operations used in the project include:

1. **Vector Addition and Subtraction:** To calculate the relative positions and velocities of the planets.

$$\mathbf{r}_{relative} = \mathbf{r}_{jupiter} - \mathbf{r}_{earth}$$

2. **Magnitude of a Vector:** To calculate distances between celestial bodies.

$$|\mathbf{r}| = \sqrt{x^2 + y^2 + z^2}$$

3. **Normalization of a Vector:** To determine the direction of the gravitational force.

$$\hat{\mathbf{r}} = \frac{\mathbf{r}}{|\mathbf{r}|}$$

Discretization

Discretization involves breaking continuous variables into discrete steps to facilitate numerical computation. In this project, we discretize time into small intervals (Δt) to apply Euler's method effectively. The accuracy of the simulation depends on the size of these time steps; smaller time steps generally result in more accurate simulations but require more computational effort.

Initial Conditions Calculation

The initial conditions, such as positions and velocities, are crucial for accurate simulations. For example, the initial velocity of Earth at perihelion is calculated using the vis-viva equation:

$$v = \sqrt{GM\left(\frac{2}{r} - \frac{1}{a}\right)}$$

where:

- v is the orbital speed.
- G is the gravitational constant.
- M is the mass of the Sun.
- r is the distance from the Sun at perihelion.
- a is the semi-major axis of the orbit.

Equations

Differential Equations for the Position of the Earth

Under the assumption that the Earth and the Sun are the only bodies in the Solar System, the motion of the Earth can be described by Newton's Second Law of Motion and Newton's Law of Universal Gravitation.

Let $\mathbf{r} = (x, y)$ be the position vector of the Earth relative to the Sun, where x and y are the x and y coordinates of the Earth, respectively.

The gravitational force exerted by the Sun on the Earth is given by:

$$\mathbf{F} = -g \frac{M_{sun} M_{earth}}{r^2} \hat{\mathbf{r}}$$

where:

- G is the gravitational constant.
- M_{sun} is the mass of the Sun.
- M_{earth} is the mass of the Earth.
- $r = \sqrt{x^2 + y^2}$ is the distance between the Earth and the Sun.
- $\hat{\mathbf{r}} = \frac{\mathbf{r}}{r} = (\frac{x}{r}, \frac{y}{r})$ is the unit vector in the direction of \mathbf{r} .

The acceleration of the Earth due to this force is:

$$\mathbf{a} = \frac{\mathbf{F}}{M_{earth}} = -G \frac{M_{sun}}{r^2} \hat{\mathbf{r}}$$

Therefore, the differential equations for the x and y components of the Earth's acceleration are:

$$a_x = -G \frac{M_{sun}}{r^2} \frac{x}{r} = -G \frac{M_{sun} x}{(x^2 + y^2)^{3/2}}$$

$$a_y = -G \frac{M_{sun}}{r^2} \frac{y}{r} = -G \frac{M_{sun} y}{(x^2 + y^2)^{3/2}}$$

Using these accelerations, the second-order differential equations for the x and y components of the Earth's position are:

$$\frac{d^2 x}{dt^2} = -G \frac{M_{sun} x}{(x^2 + y^2)^{3/2}}$$

$$\frac{d^2 y}{dt^2} = -G \frac{M_{sun} y}{(x^2 + y^2)^{3/2}}$$

Difference Equations for Numerical Simulation

To simulate the system, we convert the continuous differential equations into discrete difference equations using a numerical method such as Euler's method. Let (Δt) be the time step for the simulation.

Euler's Method

In Euler's method, the position and velocity of the Earth are updated iteratively using the following difference equations:

1. Calculate the distance \mathbf{r}

$$r = \sqrt{x^2 + y^2}$$

1. Calculate the accelerations:

$$a_x = -G \frac{M_{\text{sun}} x}{(x^2 + y^2)^{3/2}}$$

$$a_y = -G \frac{M_{\text{sun}} y}{(x^2 + y^2)^{3/2}}$$

2. Update the velocities:

$$v_{x,\text{new}} = v_{x,\text{current}} + a_x \Delta t$$

$$v_{y,\text{new}} = v_{y,\text{current}} + a_y \Delta t$$

3. Update the positions:

$$x_{\text{new}} = x_{\text{current}} + v_{x,\text{current}} \Delta t$$

$$y_{\text{new}} = y_{\text{current}} + v_{y,\text{current}} \Delta t$$

These difference equations provide a step-by-step method to approximate the continuous motion of the Earth around the Sun.

Initial Conditions

Accurately determining the initial conditions for the simulation is crucial for producing realistic orbital paths for Earth and Jupiter. These conditions include the initial positions and velocities of the planets at a specific point in their orbits. In this project, we use the perihelion (the point in the orbit where the planet is closest to the Sun) to define the initial positions and velocities. Here, we explain how we obtained these initial values.

Initial Positions

The initial positions of Earth and Jupiter at perihelion are well-documented astronomical data:

- **Earth's perihelion distance:** 1.471×10^{11} meters
- **Jupiter's perihelion distance:** 7.785×10^{11} meters

These distances are the shortest distances between the Sun and the respective planets.

For simplicity, we place both planets on the x-axis at the start of the simulation, meaning their y-coordinates are initially zero:

- **Earth's initial position:**
 - $x_{\text{earth}} = 1.471 \times 10^{11}$ meters
 - $y_{\text{earth}} = 0$ meters

- **Jupiter's initial position:**

- $x_{jupiter} = 7.785 \times 10^{11}$ meters
- $y_{jupiter} = 0$ meters

Initial Velocities

The initial velocities of Earth and Jupiter at perihelion can be calculated using the vis-viva equation, which provides the orbital speed of a body in a gravitational field:

$$v = \sqrt{GM\left(\frac{2}{r} - \frac{1}{a}\right)}$$

where:

- v is the orbital speed.
- G is the gravitational constant.
- M is the mass of the Sun.
- r is the distance from the Sun at perihelion.
- a is the semi-major axis of the orbit.

The semi-major axis is the average of the maximum and minimum distances of the planet from the Sun.

- **For Earth:**

- Semi-major axis $a_{earth} = 1.495 \times 10^{11}$ meters (average Earth-Sun distance).
- Perihelion distance $r_{earth} = 1.471 \times 10^{11}$ meters.

Using the vis-viva equation:

$$v_{earth} = \sqrt{(6.6743 \times 10^{-11}) \times (1.989 \times 10^{30}) \left(\frac{2}{1.471 \times 10^{11}} - \frac{1}{1.495 \times 10^{11}} \right)} \approx 30,286 \text{ m/s}$$

Thus, the initial velocity of Earth at perihelion is approximately 30,286 m/s.

- **For Jupiter:**

- Semi-major axis $a_{jupiter} = 7.785 \times 10^{11}$ meters.
- Perihelion distance $r_{jupiter} = 7.785 \times 10^{11}$ meters.

Using the vis-viva equation:

$$v_{jupiter} = \sqrt{(6.6743 \times 10^{-11}) \times (1.989 \times 10^{30}) \left(\frac{2}{7.785 \times 10^{11}} - \frac{1}{7.785 \times 10^{11}} \right)} \approx 13,070 \text{ m/s}$$

Thus, the initial velocity of Jupiter at perihelion is approximately 13,070 m/s.

For both Earth and Jupiter, the initial velocities are directed along the y-axis, as they are at their closest approach to the Sun moving perpendicular to the line connecting them to the Sun.

- **Earth's initial velocity:**

- $vx_{earth} = 0 \text{ m/s}$
- $vy_{earth} = 30,286.4 \text{ m/s}$

- **Jupiter's initial velocity:**

- $vx_{jupiter} = 0 \text{ m/s}$
- $vy_{jupiter} = 13,070 \text{ m/s}$

Summary of Initial Conditions

- **Earth:**

- Position: $(x_{earth}, y_{earth}) = (1.471 \times 10^{11} \text{ m}, 0 \text{ m})$
- Velocity: $(vx_{earth}, vy_{earth}) = (0 \text{ m/s}, 30,286.4 \text{ m/s})$

- **Jupiter:**

- Position: $(x_{jupiter}, y_{jupiter}) = (7.785 \times 10^{11} \text{ m}, 0 \text{ m})$
- Velocity: $(vx_{jupiter}, vy_{jupiter}) = (0 \text{ m/s}, 13,070 \text{ m/s})$

These initial conditions provide the starting points for our numerical simulation, allowing us to accurately model the subsequent motion of Earth and Jupiter as they orbit the Sun, taking into account their mutual gravitational interactions.

Methodology

The methodology section details the step-by-step approach used to simulate the orbital motion of Earth and Jupiter around the Sun. This involves setting up the differential equations that govern their motion and solving these equations using numerical integration.

Differential Equations

The motion of Earth and Jupiter is governed by second-order differential equations derived from Newton's Second Law of Motion and Newton's Law of Universal Gravitation. These equations describe how the positions and velocities of the planets change over time due to gravitational forces.

Position and Velocity

We decompose the second-order differential equations into first-order differential equations for position and velocity:

For Earth:

$$\frac{d\mathbf{r}_{earth}}{dt} = \mathbf{v}_{earth}$$

$$\frac{d\mathbf{v}_{earth}}{dt} = \mathbf{a}_{earth}$$

For Jupiter:

$$\frac{d\mathbf{r}_{jupiter}}{dt} = \mathbf{v}_{jupiter}$$

$$\frac{d\mathbf{v}_{jupiter}}{dt} = \mathbf{a}_{jupiter}$$

Where:

- \mathbf{r} represents the position vectors of Earth and Jupiter.
- \mathbf{v} represents the velocity vectors.
- \mathbf{a} represents the acceleration vectors.

The accelerations are determined by the gravitational forces acting on the planets:

For Earth:

$$\mathbf{a}_{earth} = -G \frac{M_{sun} \mathbf{r}_{earth}}{r_{earth}^3} + G \frac{M_{jupiter} (\mathbf{r}_{jupiter} - \mathbf{r}_{earth})}{(r_{jupiter} - r_{earth})^3}$$

For Jupiter:

$$\mathbf{a}_{earth} = -G \frac{M_{sun} \mathbf{r}_{jupiter}}{r_{jupiter}^3} + G \frac{M_{earth} (\mathbf{r}_{earth} - \mathbf{r}_{jupiter})}{(r_{earth} - r_{jupiter})^3}$$

Numerical Integration

To solve these differential equations, we use Euler's method, a straightforward numerical integration technique.

Euler's method approximates the solutions by discretizing time into small steps and iteratively updating the positions and velocities of the planets.

Euler's Method

In each time step Δt , Euler's method updates the position and velocity using the following formulas:

For position:

$$\mathbf{r}_{new} = \mathbf{r}_{current} + \mathbf{v}_{current} \cdot \Delta t$$

For velocity:

$$v_{new} = v_{current} + a \cdot \Delta t$$

In our simulation, we perform the following steps iteratively:

1. Calculate the distances:

- Distance from the Sun to Earth: $r_{earth} = \sqrt{x_{earth}^2 + y_{earth}^2}$
- Distance from the Sun to Jupiter: $r_{jupiter} = \sqrt{x_{jupiter}^2 + y_{jupiter}^2}$
- Distance between Earth and Jupiter: $r_{earth-jupiter} = \sqrt{(x_{jupiter} - x_{earth})^2 + (y_{jupiter} - y_{earth})^2}$

2. Calculate the accelerations:

- For Earth:

$$a_{x,earth} = -G \frac{M_{sun} x_{earth}}{r_{earth}^3} + G \frac{M_{jupiter} (x_{jupiter} - x_{earth})}{r_{earth-jupiter}^3}$$

$$a_{y,earth} = -G \frac{M_{sun} y_{earth}}{r_{earth}^3} + G \frac{M_{jupiter} (y_{jupiter} - y_{earth})}{r_{earth-jupiter}^3}$$

- For Jupiter:

$$a_{x,jupiter} = -G \frac{M_{sun} x_{jupiter}}{r_{jupiter}^3} + G \frac{M_{earth} (x_{earth} - x_{jupiter})}{r_{earth-jupiter}^3}$$

$$a_{y,jupiter} = -G \frac{M_{sun} y_{jupiter}}{r_{jupiter}^3} + G \frac{M_{earth} (y_{earth} - y_{jupiter})}{r_{earth-jupiter}^3}$$

3. Update the velocities:

- For Earth:

$$v_{x,earth,new} = v_{x,earth,current} + a_{x,earth} \cdot \Delta t$$

$$v_{y,earth,new} = v_{y,earth,current} + a_{y,earth} \cdot \Delta t$$

- For Jupiter:

$$v_{x,jupiter,new} = v_{x,jupiter,current} + a_{x,jupiter} \cdot \Delta t$$

$$v_{y,jupiter,new} = v_{y,jupiter,current} + a_{y,jupiter} \cdot \Delta t$$

4. Update the positions:

- For Earth:

$$x_{earth,new} = x_{earth,current} + v_{x,earth,current} \cdot \Delta t$$

$$y_{earth,new} = y_{earth,current} + v_{y,earth,current} \cdot \Delta t$$

- For Jupiter:

$$x_{jupiter,new} = x_{jupiter,current} + v_{x,jupiter,current} \cdot \Delta t$$

$$y_{jupiter,new} = y_{jupiter,current} + v_{y,jupiter,current} \cdot \Delta t$$

By repeating these steps for each time interval, we simulate the continuous motion of Earth and Jupiter around the Sun, taking into account their mutual gravitational interactions. Euler's method, while simple, provides a practical means of approximating the complex dynamics of planetary orbits.

Implementation

`initial_vals.py` provides foundational data, while `two_body.py` and `three_body.py` use this data to simulate and visualize the orbits of Earth and Jupiter using numerical integration techniques. The two scripts illustrate different scenarios: `two_body.py` models Earth's orbit ignoring Jupiter, while `three_body.py` includes the gravitational influence of Jupiter, providing a more complex and accurate representation of the planetary dynamics.

Explanation of Files

▼ `initial_vals.py`

This file provides the necessary constants and initial values used in both `two_body.py` and `three_body.py` scripts. It defines the gravitational constant, masses, distances, and orbital periods of Earth and Jupiter.

```
import numpy as np
```

- **Importing numpy:** `numpy` is a library used for numerical computations. It provides support for arrays and a variety of mathematical functions.

```
G = 6.67430e-11
m_sun = 1.989e30
```

- **Gravitational constant:** `G` is a fundamental constant used in the calculation of gravitational forces.
- **Mass of the Sun:** `m_sun` is the mass of the Sun, which is used to calculate gravitational forces exerted by the Sun.

```
m_earth = 5.972e24
```

- **Mass of the Earth:** This is the mass of the Earth.

```
earth_perihelion = 1.471e8
earth_aphelion = 1.521e8
earth_semi_major = (earth_perihelion + earth_aphelion) / 2
```

```
earth_period = 365.256 * 24 * 3600
```

- **Earth's orbital distances:**

- `earth_perihelion` : Closest distance between Earth and the Sun.
- `earth_aphelion` : Farthest distance between Earth and the Sun.
- `earth_semi_major` : Average distance of Earth from the Sun (semi-major axis).

- **Orbital period of the Earth:**

- `earth_period` : the time it takes for the Earth to complete one orbit around the Sun, converted into seconds.

```
x_earth = 0.0 # TODO
y_earth = 0.0 # TODO
vx_earth = 0 # TODO
vy_earth = 0.0 # TODO
```

- **Initial conditions for Earth:** These variables will store the initial position and velocity components of the Earth. They are marked as TODOs, indicating that their values need to be calculated or specified.

```
m_jupiter = 1.898e27
```

- **Mass of Jupiter:** This is the mass of Jupiter.

```
jupiter_perihelion = 7.4059e8
jupiter_aphelion = 8.1636e8
jupiter_semi_major = (jupiter_perihelion + jupiter_aphelion) / 2

jupiter_period = 4332.589 * 24 * 3600
```

- **Jupiter's orbital distances:**

- `jupiter_perihelion` : Closest distance between Jupiter and the Sun.
- `jupiter_aphelion` : Farthest distance between Jupiter and the Sun.
- `jupiter_semi_major` : Average distance of Jupiter from the Sun (semi-major axis).

- **Orbital period of Jupiter:**

- `jupiter_period` : the time it takes for Jupiter to complete one orbit around the Sun, converted into seconds.

```
x_jupiter = 0.0 # TODO
y_jupiter = 0.0 # TODO
```

```
vx_jupiter = 0.0 # TODO
vy_jupiter = 0.0 # TODO
```

- **Initial conditions for Jupiter:** These variables will store the initial position and velocity components of Jupiter. They are marked as TODOs, indicating that their values need to be calculated or specified.

▼ two_body.py

This file simulates the two-body problem, specifically Earth orbiting the Sun, ignoring the gravitational influence of Jupiter.

- Imports the initial values from `initial_vals.py`.
- Sets the initial conditions for Earth, calculates its velocity, and simulates its orbit around the Sun over one year.
- Uses Euler's method for numerical integration to update Earth's position and velocity at each time step.
- Plots the results showing Earth's position over time and its orbit around the Sun.

```
import data.initial_vals as vals
import numpy as np
import matplotlib.pyplot as plt
```

- **Importing libraries and constants:**

- `data.initial_vals`: Imports the initial values and constants from `initial_vals.py`.
- `numpy`: Used for numerical calculations.
- `matplotlib.pyplot`: Used for plotting graphs.

```
x_earth = vals.earth_perihelion
y_earth = 0.0
vx_earth = 0.0
vy_earth = np.sqrt(vals.G * vals.m_sun * ((2 / vals.earth_perihelion) -
(1 / vals.earth_semi_major)))
```

- **Setting Earth's initial conditions:**

- `x_earth`: Earth's initial x-coordinate, set to its perihelion distance.
- `y_earth`: Earth's initial y-coordinate, set to 0.
- `vx_earth`: Earth's initial x-velocity, set to 0.
- `vy_earth`: Earth's initial y-velocity, calculated using the vis-viva equation.

```
t_span = 31558149.8 * 3
dt = 60 * 60
```

```
n_steps = int(t_span / dt)
```

- **Simulation parameters:**

- `t_span`: Total time span for the simulation (three years).
- `dt`: Time step (one hour).
- `n_steps`: Number of simulation steps, calculated by dividing the total time span by the time step.
 - This value is converted to an integer using `int()`, preventing a fractional number of steps in the simulation.

```
x_positions = np.zeros(n_steps)
y_positions = np.zeros(n_steps)
vx = vx_earth
vy = vy_earth
x = x_earth
y = y_earth
```

- **Arrays for storing positions and velocities:** These arrays will store the x and y positions of Earth at each time step.

- `x_positions`, `y_positions`: Arrays to store Earth's position at each step.
- `vx`, `vy`, `x`, `y`: Variables to hold Earth's current velocity and position.

```
for i in range(n_steps):
    r = np.sqrt(x**2 + y**2)
    ax = -vals.G * vals.m_sun * x / r**3
    ay = -vals.G * vals.m_sun * y / r**3
    vx += ax * dt
    vy += ay * dt
    x += vx * dt
    y += vy * dt
    x_positions[i] = x
    y_positions[i] = y
```

- **Euler's method for numerical integration:**

- Loop through each time step to update Earth's position and velocity.
- `r`: Distance between Earth and the Sun.
- `ax`, `ay`: Acceleration components due to the Sun's gravitational force.
- `vx`, `vy`: Update Earth's velocity components.
- `x`, `y`: Update Earth's position components.

- Store the updated positions in `x_positions` and `y_positions` arrays.

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7))
```

- **Setting up the plots:**

- Create two subplots side-by-side.

```
ax1.plot(np.linspace(0, t_span, n_steps), x_positions, label='x_earth')
ax1.plot(np.linspace(0, t_span, n_steps), y_positions, label='y_earth')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Position (m)')
ax1.legend()
ax1.set_title('Position vs Time (Two-Body System)')
ax1.grid()
```

- **Plotting position vs. time:**

- Plot Earth's x and y positions over time.
- Label the axes and add a legend and title.

```
ax2.plot(x_positions, y_positions, label='Earth')
ax2.plot(0, 0, 'yo', label='Sun') # Plotting the Sun at the origin
ax2.set_xlabel('X Position (m)')
ax2.set_ylabel('Y Position (m)')
ax2.legend()
ax2.set_title('Orbit Trace (Two-Body System)')
ax2.grid()
```

- **Plotting the orbit trace:**

- Plot Earth's orbit around the Sun.
- Mark the Sun's position at the origin.
- Label the axes and add a legend and title.

```
plt.tight_layout()
plt.show()
```

- **Finalizing the plots:**

- Adjust the layout and display the plots.

▼ **three_body.py**

This file simulates the three-body problem, including the gravitational interactions between Earth, Jupiter, and the Sun.

- Imports the initial values from `initial_vals.py`.
- Sets the initial conditions for both Earth and Jupiter, including their velocities.
- Simulates their orbits around the Sun over one Jupiter year, taking into account the gravitational interactions between Earth, Jupiter, and the Sun.
- Uses Euler's method for numerical integration to update the positions and velocities of Earth and Jupiter at each time step.
- Plots the results showing the positions of both Earth and Jupiter over time and their orbits around the Sun.

```
import data.initial_vals as vals
import numpy as np
import matplotlib.pyplot as plt
```

- **Importing libraries and constants:**

- `data.initial_vals`: Imports the initial values and constants from `initial_vals.py`.
- `numpy`: Used for numerical calculations.
- `matplotlib.pyplot`: Used for plotting graphs.

```
x_earth = 1.47095e11
y_earth = 0.0
vx_earth = 0.0
vy_earth = 30286.4027

x_jupiter = 7.785e11
y_jupiter = 0.0
vx_jupiter = 0.0
vy_jupiter = 13070.0
```

- **Setting initial conditions:**

- Earth's initial position and velocity.
- Jupiter's initial position and velocity.

```
t_span = 31558118.4 * 32
dt = 60 * 60
n_steps = int(t_span / dt)
```

- **Simulation parameters:**

- `t_span`: Total time span for the simulation (32 Earth years).

- `dt`: Time step (one hour).
- `n_steps`: Number of simulation steps, calculated by dividing the total time span by the time step.

```
x_positions_earth = np.zeros(n_steps)
y_positions_earth = np.zeros(n_steps)
x_positions_jupiter = np.zeros(n_steps)
y_positions_jupiter = np.zeros(n_steps)
```

- **Arrays for storing positions:**

- `x_positions_earth`, `y_positions_earth`: Arrays to store Earth's position at each step.
- `x_positions_jupiter`, `y_positions_jupiter`: Arrays to store Jupiter's position at each step.

```
vx_earth_current = vx_earth
vy_earth_current = vy_earth
x_earth_current = x_earth
y_earth_current = y_earth

vx_jupiter_current = vx_jupiter
vy_jupiter_current = vy_jupiter
x_jupiter_current = x_jupiter
y_jupiter_current = y_jupiter
```

- **Initial values:**

- Set the current position and velocity of Earth and Jupiter to the initial values.

```
for i in range(n_steps):
    r_earth = np.sqrt(x_earth_current**2 + y_earth_current**2)
    r_jupiter = np.sqrt(x_jupiter_current**2 + y_jupiter_current**2)
    r_earth_jupiter = np.sqrt((x_jupiter_current - x_earth_current)**2 +
    (y_jupiter_current - y_earth_current)**2)

    ax_earth = -vals.G * vals.m_sun * x_earth_current / r_earth**3 + val
s.G * vals.m_jupiter * \
        (x_jupiter_current - x_earth_current) / r_earth_jupiter**3
    ay_earth = -vals.G * vals.m_sun * y_earth_current / r_earth**3 + val
s.G * vals.m_jupiter * \
        (y_jupiter_current - y_earth_current) / r_earth_jupiter**3

    vx_earth_current += ax_earth * dt
    vy_earth_current += ay_earth * dt
```

```

x_earth_current += vx_earth_current * dt
y_earth_current += vy_earth_current * dt

x_positions_earth[i] = x_earth_current
y_positions_earth[i] = y_earth_current

```

- **Updating Earth's position and velocity:**

- Calculate the distances: `r_earth`, `r_jupiter`, `r_earth_jupiter`.
- Calculate Earth's acceleration components: `ax_earth`, `ay_earth`.
- Update Earth's velocity components: `vx_earth_current`, `vy_earth_current`.
- Update Earth's position components: `x_earth_current`, `y_earth_current`.
- Store the updated positions in `x_positions_earth` and `y_positions_earth` arrays.

```

ax_jupiter = -vals.G * vals.m_sun * x_jupiter_current / r_jupiter**3
+ vals.G * vals.m_earth * \
    (x_earth_current - x_jupiter_current) / r_earth_jupiter*
*3
ay_jupiter = -vals.G * vals.m_sun * y_jupiter_current / r_jupiter**3
+ vals.G * vals.m_earth * \
    (y_earth_current - y_jupiter_current) / r_earth_jupiter*
*3

vx_jupiter_current += ax_jupiter * dt
vy_jupiter_current += ay_jupiter * dt

x_jupiter_current += vx_jupiter_current * dt
y_jupiter_current += vy_jupiter_current * dt

x_positions_jupiter[i] = x_jupiter_current
y_positions_jupiter[i] = y_jupiter_current

```

- **Updating Jupiter's position and velocity:**

- Calculate Jupiter's acceleration components: `ax_jupiter`, `ay_jupiter`.
- Update Jupiter's velocity components: `vx_jupiter_current`, `vy_jupiter_current`.
- Update Jupiter's position components: `x_jupiter_current`, `y_jupiter_current`.
- Store the updated positions in `x_positions_jupiter` and `y_positions_jupiter` arrays.

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7))
```

- **Setting up the plots:**

- Create two subplots side-by-side.

```
ax1.plot(np.linspace(0, t_span, n_steps), x_positions_earth, label='x_earth')
ax1.plot(np.linspace(0, t_span, n_steps), y_positions_earth, label='y_earth')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Position (m)')
ax1.legend()
ax1.set_title('Position vs Time for Earth and Jupiter')
ax1.grid()
```

- **Plotting position vs. time for Earth and Jupiter:**

- Plot Earth's x and y positions over time.
- Label the axes and add a legend and title.

```
ax1.plot(np.linspace(0, t_span, n_steps), x_positions_jupiter, label='x_jupiter', linestyle='--')
ax1.plot(np.linspace(0, t_span, n_steps), y_positions_jupiter, label='y_jupiter', linestyle='--')
ax1.legend()
```

- **Plotting position vs. time for Jupiter:**

- Plot Jupiter's x and y positions over time with dashed lines for distinction.
- Add a legend.

```
ax2.plot(x_positions_earth, y_positions_earth, label='Earth')
ax2.plot(x_positions_jupiter, y_positions_jupiter, label='Jupiter', linestyle='--')
ax2.plot(0, 0, 'yo', label='Sun') # Plotting the Sun at the origin
ax2.set_xlabel('X Position (m)')
ax2.set_ylabel('Y Position (m)')
ax2.legend()
ax2.set_title('Orbit Trace')
ax2.grid()
```

- **Plotting the orbit trace:**

- Plot Earth's and Jupiter's orbits around the Sun.
- Mark the Sun's position at the origin.

- Label the axes and add a legend and title.

```
plt.tight_layout()  
plt.show()
```

- **Finalizing the plots:**
 - Adjust the layout and display the plots.

Results and Discussion

Plots and Observations

Two-Body Simulation

In the two-body simulation (`two_body.py`), we modeled the orbit of Earth around the Sun without considering the gravitational influence of Jupiter. Below are the key plots and observations from this simulation:

- **Position vs. Time:** This plot shows how the x and y positions of Earth change over time during one year. The Earth starts at its perihelion (closest approach to the Sun) and follows an elliptical path.
- **Orbit Trace:** This plot shows the path of Earth around the Sun, forming an elliptical orbit as expected. The Sun is positioned at one of the foci of the ellipse.

Observations:

- The orbit is an ellipse with the Sun at one focus, which aligns with Kepler's First Law.
- The position vs. time graph indicates that Earth's velocity changes over its orbit, moving faster at perihelion and slower at aphelion, consistent with Kepler's Second Law.

Three-Body Simulation

In the three-body simulation (`three_body.py`), we included both Earth and Jupiter, considering their mutual gravitational influences along with the Sun's gravity. Below are the key plots and observations from this simulation:

- **Position vs. Time for Earth and Jupiter:** These plots show the x and y positions of both Earth and Jupiter over time. Both planets exhibit periodic motion, but their paths are more complex due to their gravitational interactions.
- **Orbit Trace:** This plot shows the orbits of Earth and Jupiter around the Sun. The orbits are more complex than in the two-body simulation, showing perturbations due to their mutual gravitational influences.

Observations:

- The orbits are still generally elliptical, but with noticeable deviations caused by the gravitational interactions between Earth and Jupiter.
- The perturbations in the orbit of Earth are more pronounced when Jupiter is closer, indicating the significant gravitational influence of the more massive Jupiter.

Comparison of Orbits

Comparing the results of the two-body and three-body simulations reveals several key differences:

- **Orbit Shape:** In the two-body simulation, Earth's orbit is a near-perfect ellipse. In the three-body simulation, Earth's orbit shows deviations from a perfect ellipse due to Jupiter's gravitational pull.
- **Position Changes:** The position vs. time plots for Earth in the three-body simulation show slight oscillations superimposed on the regular elliptical motion. These oscillations are absent in the two-body simulation.
- **Perturbations:** The three-body simulation highlights the perturbations in Earth's orbit due to Jupiter's influence. These perturbations cause the orbit to shift slightly, especially noticeable when Jupiter is near its closest approach to Earth.

Observations:

- Jupiter's gravitational influence causes significant perturbations in Earth's orbit, demonstrating the importance of considering multi-body interactions in orbital simulations.
- The perturbations can affect the long-term stability and predictability of Earth's orbit, an important factor in celestial mechanics.

Physical Interpretations

- **Kepler's Laws:** The two-body simulation results align well with Kepler's Laws of Planetary Motion:
 - The orbit of Earth is an ellipse with the Sun at one focus (First Law).
 - The area swept out by the line joining Earth and the Sun is constant over equal time intervals, indicating that Earth's speed varies as it moves along its orbit (Second Law).
 - The relationship between the orbital period and the semi-major axis can be observed, consistent with Kepler's Third Law.
- **Gravitational Interactions:** The three-body simulation shows the significant impact of gravitational interactions between planets:
 - Jupiter's large mass and gravitational influence cause noticeable perturbations in Earth's orbit, highlighting the complexity of multi-body systems.
 - These perturbations illustrate the principles of Newton's Third Law, where the gravitational force exerted by Jupiter on Earth is equal and opposite to the force exerted by Earth on Jupiter.
- **Stability and Chaos:** The introduction of a third body (Jupiter) into the simulation demonstrates how additional gravitational influences can affect the stability and predictability of orbits. While the orbits remain generally stable over the simulation period, the perturbations hint at the potential for more chaotic behavior over longer timescales.
- **Real-World Implications:** Understanding these interactions is crucial for accurate predictions of planetary positions and for planning space missions. The perturbations caused by gravitational interactions must be accounted for in long-term models of the solar system.

In summary, the two-body and three-body simulations provide valuable insights into the dynamics of planetary orbits. The results highlight the importance of considering multi-body gravitational interactions and demonstrate the principles of Kepler's and Newton's laws in governing celestial motion.

Conclusion

This project aimed to simulate the orbital dynamics of Earth and Jupiter around the Sun, highlighting the effects of their gravitational interactions. Through the use of differential equations and numerical integration, specifically Euler's method, we successfully modeled and visualized the complex motions of these celestial bodies.

Key Findings

1. Two-Body Simulation:

- The orbit of Earth around the Sun, modeled as a two-body problem, conforms to Kepler's Laws of Planetary Motion.
- Earth's elliptical orbit with the Sun at one focus demonstrates the predictive power of Kepler's First Law.
- The varying velocity of Earth, moving faster at perihelion and slower at aphelion, aligns with Kepler's Second Law.

2. Three-Body Simulation:

- Introducing Jupiter into the simulation as a third body revealed the significant perturbations in Earth's orbit due to Jupiter's gravitational influence.
- The mutual gravitational interactions between Earth and Jupiter led to deviations from a perfect elliptical orbit, showcasing the complexity of multi-body gravitational systems.
- Jupiter's gravitational pull caused noticeable oscillations in Earth's orbital path, underscoring the importance of considering all significant gravitational influences for accurate orbital predictions.

Physical and Mathematical Principles

The simulations relied on fundamental physical laws and mathematical techniques:

- **Newton's Laws of Motion** provided the framework for calculating the accelerations and forces acting on the planets.
- **Newton's Law of Universal Gravitation** allowed us to compute the gravitational forces between the Sun, Earth, and Jupiter.
- **Kepler's Laws of Planetary Motion** guided our expectations and interpretations of the orbital paths.
- **Numerical Integration (Euler's Method)** enabled the step-by-step approximation of the planets' positions and velocities over time.

Implications

Understanding the dynamics of planetary orbits has broad implications:

- **Celestial Mechanics:** Accurate models of planetary motion are essential for predicting the positions of celestial bodies, understanding the stability of the solar system, and exploring the potential for chaotic behavior in multi-body systems.
- **Space Exploration:** Precise orbital calculations are critical for mission planning, satellite deployment, and interplanetary navigation.
- **Astrophysical Research:** Insights into gravitational interactions and orbital perturbations contribute to our knowledge of planetary formation, migration, and long-term stability.

Improvements

Future improvements could include:

- **Advanced Numerical Methods:** Implementing more accurate numerical integration techniques, such as Runge-Kutta methods, to reduce errors and improve the precision of the simulations.
- **Extended Simulations:** Running the simulations over longer periods to observe potential long-term stability or chaos in the planetary orbits.
- **Inclusion of More Bodies:** Expanding the model to include additional planets or moons to further study the complexities of gravitational interactions in the solar system.

In conclusion, this project successfully demonstrated the principles of planetary motion through the simulation of Earth and Jupiter's orbits. The results emphasized the importance of considering multi-body gravitational interactions for accurate modeling and highlighted the fundamental physical laws governing celestial mechanics.

References

Williams, D. R. (n.d.). *Sun Fact Sheet*. NASA.
<https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html>

Sun Fact Sheet

*This is the adopted period at 16 deg. latitude - the actual rotation rate varies with latitude L as:

 <https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html>

Williams, D. R. (n.d.). *Earth Fact Sheet*. NASA.
<https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html>

Earth Fact Sheet


Planetary Fact Table
 - metric units

 <https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html>

Williams, D. R. (n.d.). *Jupiter Fact Sheet*. NASA.
<https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html>

Jupiter Fact Sheet

Planetary Fact Table
- metric units

 <https://nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html>