Simulating Galaxy and Planet Formation Using N-body Simulation

Yunus Sorgunçay[1]
Department of Astronautical Engineering, Istanbul Technical University

I. Abstract

This report explores gravitational dynamics through numerical simulation, emphasizing a comparative analysis of MATLAB-based custom implementations and the REBOUND astrophysical software package for solving N-body problems. Given the computational complexity and nonlinear nature of gravitational interactions, accurate modeling of celestial bodies demands robust numerical methods and computational tools. The study examines various N-body scenarios, from simple two- and three-body problems to more intricate planetary accretion and galactic formation systems. Simulations in MATLAB offer explicit control and clarity regarding computational methods, allowing for detailed validation of numerical integration techniques such as Leapfrog and Runge-Kutta. In contrast, simulations with REBOUND provide advanced integrators specifically designed for astrophysical accuracy and computational efficiency. Important findings include assessments of energy and momentum conservation, numerical stability over extended simulations, and computational performance comparisons between the two software approaches. Results highlight REBOUND's superior efficiency and robustness for complex scenarios, whereas MATLAB provides transparency and adaptability crucial for educational and theoretical validations. Finally, higher scale simulations of galaxy and planet formations, including their conservation of Total Energy and Momentum are observed.

II. Introduction and Objectives

1. Purpose and Motivation of the Project

The intricate processes governing the formation of galaxies and planetary systems have long been central to astrophysical research. These phenomena are fundamentally driven by gravitational interactions among numerous celestial bodies, making N-body simulations an indispensable tool for studying such systems. By simulating the gravitational forces between individual particles or bodies, N-body simulations enable researchers to model the dynamical evolution of systems over vast timescales, providing insights that are otherwise unattainable through observational or experimental means.

One of the primary motivations for this project is to address the complex, nonlinear nature of gravitational systems. While two-body problems can be solved analytically using Newton's laws, the introduction of additional bodies results in interactions that are both increasingly chaotic and computationally demanding. N-body simulations circumvent these challenges by employing numerical methods to approximate the trajectories and interactions of multiple bodies, enabling the exploration of a wide range of astrophysical phenomena.

By leveraging N-body simulations, this project seeks to deepen our understanding of these astrophysical processes, focusing on both the **formation of planetary systems** through accretion and the **evolution of galaxies** through large-scale gravitational interactions. The insights gained from these models contribute to a broader understanding of the mechanisms shaping the universe, while also refining computational techniques and tools for future studies.

Building on previous reports' foundations, this second-term report expands the investigation by performing detailed comparative analyses of N-body problem implementations using two distinct computational frameworks: MATLAB and REBOUND. MATLAB, widely employed for numerical analysis across various engineering and scientific

[1] E-mail: sorguncay21@itu.edu.tr
Advisor: Cuma Yarım, Dr. Lecturer, Istanbul Technical University, yarim@itu.edu.tr

fields, offers general-purpose numerical solvers and versatile algorithmic customization [23]. On the other hand, REBOUND provides specialized astrophysical tools, optimized for gravitational interactions and featuring methods explicitly tailored to celestial mechanics [2]. By analyzing how these tools handle gravitational dynamics, numerical accuracy, conservation laws, and computational efficiency, this report aims to elucidate the strengths and limitations of general versus specialized software approaches.

2. Background on the N-body Problem

2.1. Definition and Complexity

The **N-body problem** is a classical challenge in gravitational dynamics, where the goal is to determine the motions of N bodies interacting under their mutual gravitational influence. The problem is governed by **Newton's laws of motion** and the **law of universal gravitation** [3], which provide a mathematical framework for predicting the trajectories of celestial bodies. For a system with N particles, the gravitational force on each particle is affected by the combined influence of all others, leading to a set of coupled differential equations.

While the equations for the gravitational interactions are conceptually simple, the complexity of solving the N-body problem grows significantly as N increases. For a system with N particles, the number of pairwise interactions scales as $O(N^2)$, making direct computation increasingly infeasible for large systems [4]. For example, simulating the dynamics of a galaxy containing millions of stars requires calculating billions of gravitational interactions at every timestep.

This computational challenge necessitates the use of **numerical approximations** [4]. Methods such as **leapfrog integration** [15], **Runge-Kutta methods** [16], and more advanced techniques like **Barnes-Hut tree algorithms** [6] or **fast multipole methods** [18] are employed to optimize the calculation of gravitational forces and manage the computational load. These approximations enable the simulation of large systems over extended periods, albeit at the cost of reduced precision in some cases.

The N-body problem is not just computationally intensive but also exhibits **chaotic behavior** in many cases [5], meaning that small changes in initial conditions can lead to vastly different outcomes. This inherent sensitivity adds another layer of complexity, making long-term predictions particularly challenging.

2.2. Historical Context

The study of the N-body problem has its roots in the foundational work of **Isaac Newton**, who first formulated the laws of motion and gravitation in the late 17th century [3]. Newton's equations allowed for precise analytical solutions to the **two-body problem**, such as the motion of a planet around a star. However, when a third body is introduced, the system becomes **nonlinear**, and exact solutions are generally unattainable, marking the beginning of the quest for approximate methods.

During the 18th century, **Leonhard Euler** [9], **Joseph-Louis Lagrange** [7], and **Pierre-Simon Laplace** [8] made significant contributions to celestial mechanics, particularly in the context of restricted three-body systems. Euler introduced analytical solutions for specific cases, such as collinear configurations, while Lagrange identified equilibrium points (now known as **Lagrangian points**) in the gravitational interaction between three bodies. Laplace further refined perturbation theory to analyze the stability of planetary orbits, laying the groundwork for understanding complex gravitational systems.

In the late 19th century, **Henri Poincaré** revolutionized the field by demonstrating the inherent **chaos** present in N-body systems [10]. His work showed that even small perturbations in initial conditions could lead to dramatic changes in a system's evolution, making long-term predictions practically impossible in many cases. Poincaré's insights not only highlighted the unpredictability of multi-body systems but also laid the foundation for the modern study of **dynamical chaos** and **nonlinear systems**.

The advent of modern computers in the 20th century transformed the study of the N-body problem, enabling the development of **numerical simulation techniques** [11]. Researchers could now model systems with thousands or millions of particles, opening new avenues for exploring phenomena such as **galaxy formation**, **star cluster dynamics**, and **planetary system evolution**. Today, the N-body problem continues to be a cornerstone of computational astrophysics, driving advancements in both numerical algorithms and our understanding of the universe.

3. Objectives of the Project

The primary aim of this project is to explore the dynamics of galaxy and planetary system formation through the application of **N-body simulations**. It is structured around three key objectives as shown in Table 1.

Table 1. Project Objectives

Objective	Description	Focus Areas
Objective 1: Development	Development and Implementation of N-body Simulations	Design and implement numerical algorithms for simulating gravitational interactions between bodies using MATLAB and REBOUND.
Objective 2: Comparison	Comparative Analysis Between MATLAB and REBOUND	Perform simulations using identical initial conditions and parameters within MATLAB and REBOUND to compare the numerical outcomes.
Objective 3: Evaluation	Evaluation of Numerical Accuracy	Analyze and compare how effectively MATLAB and REBOUND calculate various aspects of bodies in the simulated N-body systems.

Through achieving these objectives, the report aims to provide a clear understanding of the advantages and drawbacks of general-purpose computational software versus specialized astrophysical simulation tools in handling N-body problems. These insights will assist in choosing appropriate computational strategies for future research and simulations within astrophysics and related fields.

III. Methods and Resources

A. Methods and Techniques for N-body Simulations

To simulate the gravitational dynamics of multi-body systems, this project employs a range of **numerical methods** tailored to the specific requirements of planetary and galactic models. Each technique is chosen based on its ability to balance computational efficiency and accuracy, ensuring robust simulation outcomes.

1. Numerical Integration Techniques

To address the challenges mentioned before, a variety of numerical methods have been developed, each with its advantages and trade-offs:

- **Leapfrog Integration [15]**: A symplectic method well-suited for long-term simulations of planetary systems, preserving energy and angular momentum over extended periods.
- **Runge-Kutta Methods [16]**: High-accuracy methods for general-purpose integration, though computationally expensive for large systems.
- **Barnes-Hut Algorithm [6]**: A tree-based approach that reduces computational cost to $O(NlogN)$ by approximating the gravitational influence of distant particles as a single force, enabling the simulation of large systems like galaxies.

- **Fast Multipole Method [18]**: Similar to the Barnes-Hut algorithm but with greater precision, allowing efficient force calculations for massive systems.
- **Hybrid Methods [19]**: Techniques like WHFAST and IAS15, used in **REBOUND**, combine speed and accuracy by adapting integration schemes based on the problem's requirements.

2. Conservations and Calculation Equations

In N-body simulations, the conservation laws of physics play a critical role in validating the accuracy of numerical methods and understanding the dynamics of multi-body systems. These simulations adhere to fundamental conservation principles such as mass, momentum, energy, and angular momentum. Below, we provide a detailed explanation of the conserved quantities and the governing force equations that underpin the simulations.

2.1. Total Mass Conservation

The total mass of an N-body system remains constant throughout the simulation, as no mass is created or destroyed. The total mass is given by:

$$M = \sum_{i=1}^{n} m_i \tag{1}$$

Where:

- m_i is the mass of the *i*-th body.
- *n* is the total number of bodies in the system.

This principle ensures that the mass distribution in the system is accurately maintained, which is crucial for calculating gravitational interactions.

2.2. Center of Mass Conservation

The center of mass (COM) of the system remains constant or moves uniformly if no external forces act on the system. The COM is defined as:

$$\vec{R} = \frac{1}{M} \sum_{i=1}^{n} m_i \vec{r_i} = \text{Constant} \tag{2}$$

Where $\vec{r_i}$ is the position vector of the *i*-th body. This property ensures that the system's overall motion is correctly represented in the simulation.

2.3. Momentum Conservation

The total linear momentum of the system is conserved, provided no external forces are acting. [14] The total momentum is given by:

$$\vec{P} = \sum_{i=1}^{n} m_i \vec{v_i} \tag{3}$$

Where $\vec{v_i}$ is the velocity vector of the *i*-th body. If the system is isolated, $\vec{P} = Constant$, this implies the center of mass moves with a uniform velocity unless acted upon by external forces.

2.4. Angular Momentum Conservation

The angular momentum of the system about the center of mass is conserved in the absence of external torques [14]. The angular momentum is defined as:

$$\vec{L} = \sum_{i=1}^{n} \vec{r}_i \times m_i \vec{v}_i \tag{4}$$

Where $\vec{r}_i \times \vec{v}_i$ is the cross product of position and velocity vectors. This conservation law is critical for accurately modeling rotational dynamics in galaxies and planetary systems.

2.5. Force Equations

The gravitational force between two bodies is governed by Newton's law of gravitation [3]:

$$\vec{F}_{ij} = -\frac{G m_i m_j}{r_{ij}^3} \vec{r}_{ij} \tag{5}$$

Where:

- $\vec{F}_{ij}$ is the force on body *i* due to body *j*.
- *G* is the gravitational constant.
- $\vec{r}_{ij} = \vec{r}_j - \vec{r}_i$ is the position vector of particle *j* relative to particle *i*.
- $r_{ij} = \left|\vec{r}_{ij}\right|$ is the magnitude of the distance.

The net force on a body is the vector sum of all individual forces:

$$\vec{F}_i = \sum_{j \neq i} \vec{F}_{ij} \tag{6}$$

These forces drive the acceleration of the bodies according to Newton's second law [3]:

$$\vec{a}_i = \frac{\vec{F}_i}{m_i} \tag{7}$$

2.6. Energy Conservation

The total energy of an N-body system, consisting of kinetic energy (*T*) and potential energy (*U*), is conserved in an isolated system [14]:

$$E_{\text{total}} = T_{\text{kinetic}} + U_{\text{potential}} \tag{8}$$

- **Kinetic Energy**:
$$T_{\text{kinetic}} = \sum_{i=1}^{n} \frac{1}{2} m_i |\vec{v}_i|^2 \tag{9}$$

- **Potential Energy**:
$$U_{\text{potential}} = -\sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{G m_i m_j}{r_{ij}} \tag{10}$$

The conservation of total energy provides an indication of the numerical accuracy of the simulation. Deviations in energy over time can suggest the accumulation of numerical errors, highlighting areas for potential refinement in the simulation parameters or methods.

B. Software Tools for N-body Simulations

The development and analysis of N-body simulations require a combination of general-purpose computational libraries and specialized software tailored to handle gravitational dynamics. These tools enable the efficient modeling, simulation, and interpretation of multi-body systems, providing the computational framework necessary for addressing complex astrophysical phenomena. Table 2 summarizes the key characteristics, applications, advantages, and limitations of some of the available software tools for use right now.

Table 2. Comparison of Software Tools

Aspect	SciPy / NumPy [12, 13]	Astropy [17]	Gadget-2 [1]	REBOUND [2]
Description	Foundational Python libraries for scientific computing, offering tools for numerical calculations and data manipulation.	Python library for astronomical data processing, focused on units, coordinates, and astrophysical constants.	Specialized code for large-scale cosmological simulations, combining tree codes with smoothed particle hydrodynamics.	Flexible and modular N-body simulation package for planetary and galactic dynamics.
Primary Focus	General-purpose scientific computation and data manipulation.	Astronomical data analysis and unit consistency.	Large-scale cosmological simulations, including baryonic and dark matter dynamics.	Planetary and galactic simulations with high accuracy and modularity.
Applications	Prototyping numerical methods. Data analysis for simulation outputs.	Unit conversions and consistency. Processing simulation outputs in astronomical contexts.	Modeling galaxy formation and evolution. Simulating dark matter and baryonic matter systems.	Planetary system formation and evolution. Small-scale galaxy or star cluster simulations.
Advantages	Simple and efficient for small-scale systems or exploratory studies. Integrates seamlessly with other Python libraries.	Tailored for astronomers with built-in support for data formats and coordinate systems. Well-documented and actively maintained.	Optimized for parallel computations, enabling simulations with millions of particles. Supports hydrodynamics and collisionless dynamics.	Modular design allows for easy customization. High accuracy with advanced integrators (e.g., WHFAST, IAS15). Built-in visualization tools for interactive/static outputs.
Limitations	Not optimized for large-scale simulations, where computational efficiency is critical.	Primarily focused on data analysis rather than direct simulation.	Requires detailed configuration and precise parameter tuning for simulations, which can demand significant preparation time.	Less optimized for extremely large systems compared to Gadget-2. Limited support for hydrodynamics.
Best Use Cases	Initial prototyping and testing of algorithms. Post-processing of simulation data.	Analyzing simulation outputs in the context of astronomical observations.	Large-scale studies of galaxy formation, mergers, and structure evolution.	Detailed planetary formation simulations. Small/medium-scale galaxy or star cluster studies.

C. Chosen Software and Rationale

1. Why REBOUND?

For this project, **REBOUND** [2] was selected as the primary simulation tool due to its ability to balance **efficiency, usability, and flexibility**, making it an ideal choice for studying both planetary formation and galactic dynamics. REBOUND's modular design, Python integration, and extensive feature set allow it to meet the specific

requirements of this study, which involve both small-scale high-precision simulations and larger-scale explorations of gravitational interactions.

2. Balance Between Efficiency and Usability
REBOUND is optimized for a wide range of gravitational simulations, offering:

- **Symplectic Integrators**: Algorithms such as **WHFAST** and **IAS15** ensure long-term stability and high accuracy, critical for preserving conserved quantities like energy and angular momentum over extended simulations [19].
- **Flexible Force Calculations**: REBOUND supports **direct methods** for small-scale, high-precision simulations and **tree-based methods** for efficient force calculations in larger systems.
- **Customizability**: Users can easily modify the simulation setup to explore different astrophysical scenarios, from planetary accretion to the evolution of galaxy-scale structures.

Unlike other tools designed for cosmological scales (e.g., **Gadget-2**), REBOUND focuses on making simulations approachable without sacrificing computational performance, making it well-suited for medium-sized systems typical of this project.

3. Modular Structure for Customized Simulations
One of REBOUND's standout features is its **modular framework**, which allows for extensive customization:

- **Boundary Conditions**: Users can define open, periodic, or reflective boundaries, accommodating a variety of astrophysical contexts.
- **Collision Detection**: REBOUND supports **direct collisional dynamics**, enabling the modeling of planetary accretion processes where bodies merge upon contact.
- **Multiple Integration Schemes**: The ability to switch between integrators (e.g., **leapfrog**, **WHFAST**, **IAS15**) ensures that simulations can be tailored to the specific requirements of the system being studied.
- **Scalability**: While not optimized for the largest cosmological systems, REBOUND handles simulations with thousands of particles efficiently, making it ideal for galaxy formation and planetary system evolution studies.

This modularity ensures that the tool can be adapted to address both the specific needs of this project and any future extensions.

4. Python Integration
REBOUND's seamless integration with Python provides several advantages:

- **Ease of Setup**: Simulations can be initialized with minimal effort, using Python's straightforward syntax for defining particle properties (mass, position, velocity) and system parameters.
- **Post-Processing and Visualization**:
    - Python libraries such as **Matplotlib [21]**, **NumPy [13]**, and **Astropy [17]** can be used to process and visualize simulation results, enabling detailed analysis of trajectories, energy conservation, and emergent structures.
    - REBOUND also includes built-in visualization tools for rendering real-time simulations and generating snapshots.
- **Rapid Prototyping**: Python's versatility enables researchers to quickly prototype and iterate on simulation setups, facilitating exploratory studies and hypothesis testing.
- **Extensibility**: The Python ecosystem allows for easy integration with additional tools for machine learning, statistical analysis, and data management.

REBOUND's combination of **efficiency**, **modularity**, and **Python integration** makes it the ideal software for this project. Its ability to model small- and medium-scale gravitational systems with high precision and flexibility ensures that the project's objectives—ranging from planetary accretion to galaxy formation—can be achieved effectively. Moreover, its user-friendly design and compatibility with Python streamline the workflow, from simulation setup to in-depth analysis, enabling meaningful insights into the processes shaping our universe.

IV. Implementation and Experiments

This section presents a comparative implementation of N-body simulations using both **MATLAB** and the **REBOUND** Python package for four distinct test cases: **2-body, 3-body, 5-body, and 10-body problems**. Each test scenario utilizes consistent initial conditions in both environments to ensure a fair comparison. The final positions, velocities, and trajectory visualizations are recorded for each case to evaluate the accuracy, stability, and output similarity of the two platforms.

Finally, two larger test cases in **REBOUND**, one for **100-body galaxy formation and 101-body planet formation** will be conducted to simulate smaller scale simulations relative to realistic situations. The reason for not going for larger scale simulations than these is mainly because I was limited with my own computer's computing power and the aforementioned simulations were the most plausible ones for this project considering the computation time I had to complete my simulations.

A. How Will Collisions Be Handled?

In our N-body simulation using the REBOUND library, collisions between particles are handled using the **"merge"** method. This method assumes that when two particles come into contact, they undergo a **perfectly inelastic collision**, meaning they coalesce into a single, larger body. [19]

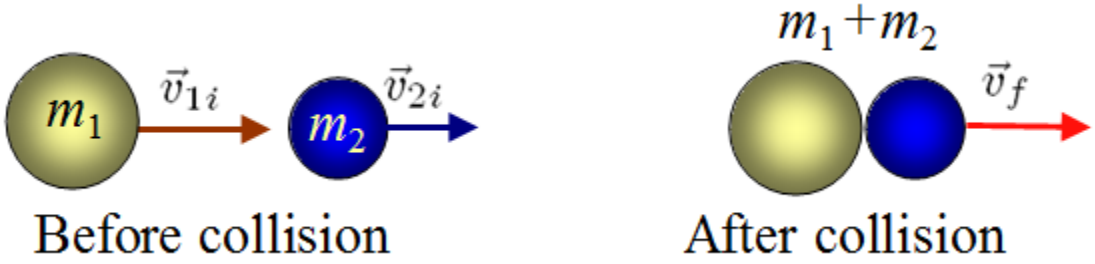


Figure 1. Perfectly Inelastic Collision (Credit: Mini Physics) [24]

This approach is physically justified in scenarios such as:

- **Protoplanetary disks**, where dust and planetesimals accrete into larger bodies,
- **Galaxy mergers**, where stars or clusters are treated as softened particles and merging simplifies the system's evolution.

A.1. Physical Principles and Conservation Laws

In a perfectly inelastic collision: [24]

- Linear momentum **is conserved,**
- Kinetic energy is ***not*** **conserved,**
- The two bodies stick together and move with a common final velocity.

A.2. Linear Momentum Conservation

Let two particles have:

- Masses m_1, m_2
- Velocities $\vec{v}_1$, $\vec{v}_2$

After collision (merging), the resulting body has mass:

$$m = m_1 + m_2 \tag{11}$$

And velocity:

$$\vec{v}_{\text{final}} = \frac{m_1\vec{v}_1 + m_2\vec{v}_2}{m_1 + m_2} \tag{12}$$

So, total linear momentum before and after is:

$$\vec{p}_{\text{before}} = m_1\vec{v}_1 + m_2\vec{v}_2 = \vec{p}_{\text{after}} = m\vec{v}_{\text{final}} \tag{13}$$

A.3. Kinetic Energy Loss

Kinetic energy before collision:

$$KE_{\text{before}} = \frac{1}{2}m_1|\vec{v}_1|^2 + \frac{1}{2}m_2|\vec{v}_2|^2 \tag{14}$$

Kinetic energy after collision:

$$KE_{\text{after}} = \frac{1}{2}(m_1 + m_2)|\vec{v}_{\text{final}}|^2 \tag{15}$$

Since $|\vec{v}_{\text{final}}|^2$ is generally **less** than the mass-weighted average of $|\vec{v}_1|^2$ and $|\vec{v}_2|^2$, some kinetic energy is **lost;** this is often interpreted as conversion into heat, deformation, or internal energy (which REBOUND does not track [19]).

B. Two-Body Simulation

B.1. Analytical Setup of the Two-Body Problem

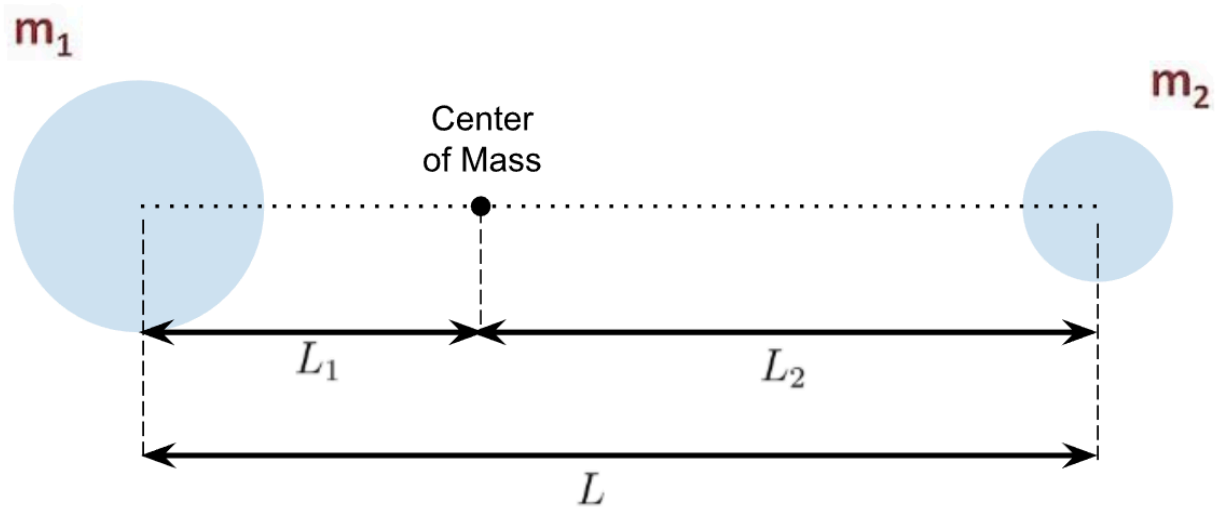


Figure 2. Illustration of the two-body problem

In the two-body test case, both bodies are initially placed such that their center of mass lies at the origin. This symmetric setup ensures a stable circular orbit when initial velocities are assigned appropriately. The system is governed by Newton's laws of motion and universal gravitation, assuming no external forces.

Let m_1 and m_2 be the masses of the two bodies, and L be the **total separation** between them. The **center of mass-relative distances** from each body to the origin are given by:

$$L_1 = \frac{m_2}{m_1 + m_2} \cdot L \qquad L_2 = \frac{m_1}{m_1 + m_2} \cdot L \tag{16}$$

These expressions ensure that the system's center of mass remains fixed at the origin.

B.2. Angular Velocity and Orbital Period

The required **angular velocity** ω for circular orbits in a two-body system is:

$$\omega = \sqrt{\frac{G(m_1 + m_2)}{L^3}} \tag{17}$$

This formula ensures that the centripetal acceleration balances the gravitational force for both masses, leading to circular orbits.

From this, the **orbital period** of the system (the time required for one complete revolution) can be derived as:

$$T = \frac{2\pi}{\omega} \tag{18}$$

This period is used as the total simulation time to complete one full orbit in both REBOUND and MATLAB simulations.

B.3. Initial Velocities

The **initial velocities** are set perpendicular to the radial line connecting the masses, with magnitudes proportional to their distances from the center of mass:

$$v_1 = \omega \cdot L_1 \qquad v_2 = \omega \cdot L_2 \tag{19}$$

Where:

- v_1 and v_2 are tangential velocities assigned to bodies 1 and 2 respectively.
- The velocity vectors are directed in opposite directions (one clockwise, one counter-clockwise) to ensure conservation of angular momentum.

B.4. Initial Conditions

- **Masses:** $[m_1, m_2] = [2 \times 10^{20}, 5 \times 10^{20}]$ kg
- **Distance between bodies:** $L = 10^{10}$ m
- **Initial Positions:**
  - r_1: $[x_1, y_1, z_1] = [-L_1, 0, 0]$ m
  - r_2: $[x_2, y_2, z_2] = [L_2, 0, 0]$ m
- **Initial Velocities:**
  - v_1: $[vx_1, vy_1, vz_1] = [0, -\omega \times L_1, 0]$ m/s
  - v_2: $[vx_2, vy_2, vz_2] = [0, \omega \times L_2, 0]$ m/s
- **Integrators:**
  - MATLAB: ode45
  - REBOUND: ias15
- **Timespan:** $T = \frac{2\pi}{\omega}$ seconds
- Check Appendix C-1, C-1.1 and C-1.2

B.5. Output

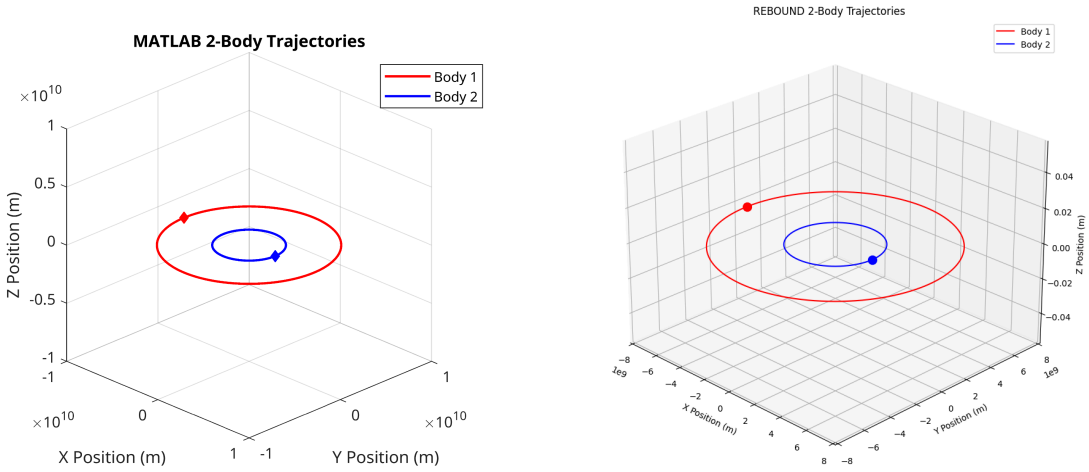
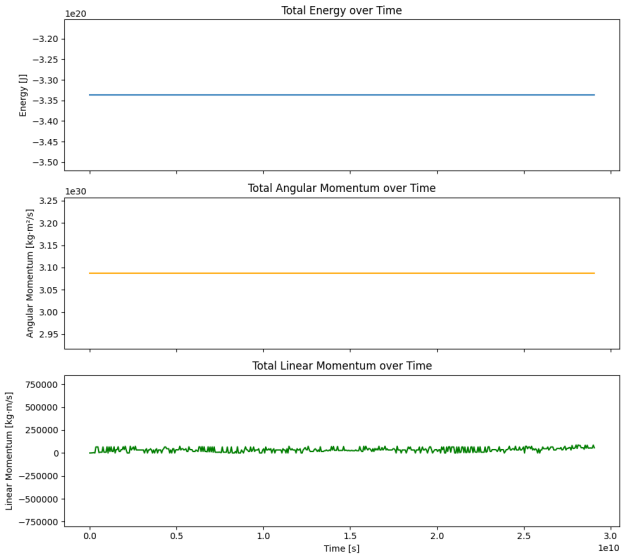


Figure 3. Trajectory plots of the two-body problem in MATLAB (left) and REBOUND (right) [Appendix C-1] [Appendix C-1.1]

```
Final positions and velocities (MATLAB):
Body 1:
  Final Position [m]: x = -7.1429e+09, y = 1.8519e-02, z = 0.0000e+00
  Final Velocity [m/s]: vx = -3.9998e-12, vy = -1.5439e+00, vz = 0.0000e+00
Body 2:
  Final Position [m]: x = 2.8571e+09, y = -7.4048e-03, z = 0.0000e+00
  Final Velocity [m/s]: vx = 1.6012e-12, vy = 6.1757e-01, vz = 0.0000e+00
```

```
Final positions and velocities (REBOUND):
Body 1:
  Final Position [m]: x = -7.1429e+09, y = -3.1918e-04, z = 0.0000e+00
  Final Velocity [m/s]: vx = 6.9732e-14, vy = -1.5439e+00, vz = 0.0000e+00

Body 2:
  Final Position [m]: x = 2.8571e+09, y = 1.3238e-04, z = 0.0000e+00
  Final Velocity [m/s]: vx = -2.7893e-14, vy = 6.1757e-01, vz = 0.0000e+00
```

Figure 4. Final positions and velocities of each body in MATLAB (top) and REBOUND (bottom)




```
Initial Energy: -3.3371e+20 J
Final Energy:   -3.3371e+20 J
Δ Energy:       0.0000e+00 J
Relative Change: 0.0e+00 (0.000e+00%)
```

Figure 5. Total Energy, Momentum plots over time (left) and ΔEnergy change information (right)

C. Three-Body Simulation

C.1. Initial Conditions

- **Masses:** $[m_1, m_2, m_3] = [2 \times 10^{20}, 2 \times 10^{20}, 1 \times 10^{20}]$ kg
- **Initial Positions:**
  - $r_1 = [-5 \times 10^9, 0, 0]$ m
  - $r_2 = [5 \times 10^9, 0, 0]$ m
  - $r_3 = [0, 0, 0]$ m
- **Timespan:** 3×10^{10} seconds
- **Integrators:**
  - MATLAB: ode45
  - REBOUND: ias15
- **Initial Velocities:**
  - $v_1 = [-1, 0.5, 1]$ m/s
  - $v_2 = [1, 2, 0]$ m/s
  - $v_3 = [0, 0, 0.2]$ m/s
- Check Appendix C-2, C-2.1 and C-2.2

C.2. Output

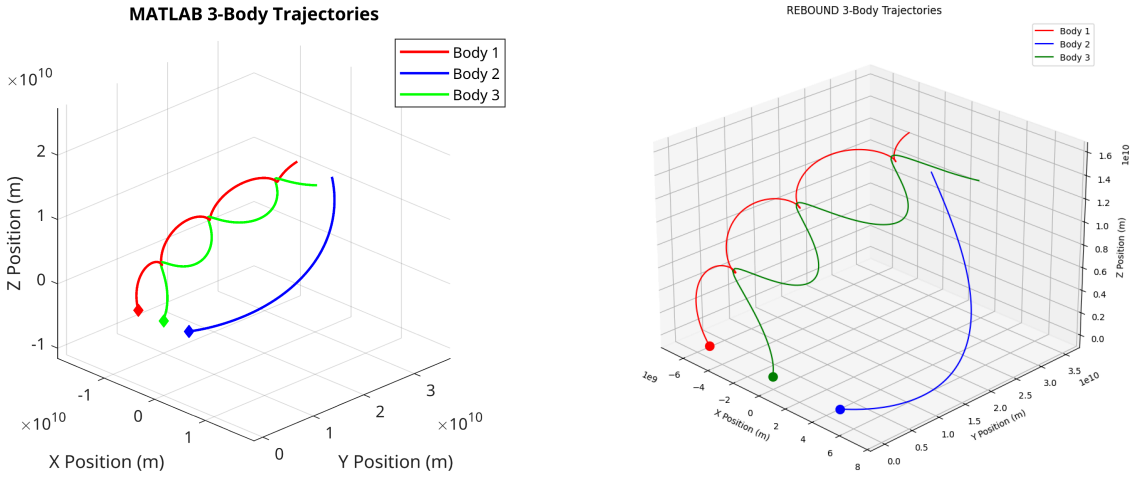
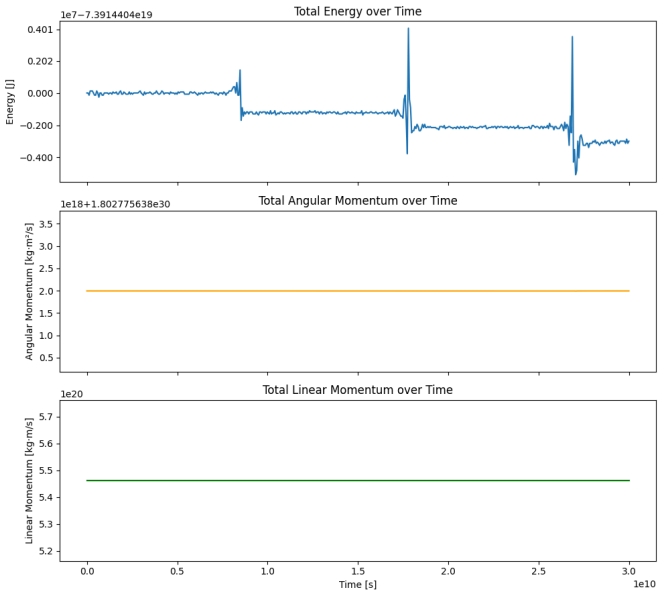


Figure 6. Trajectory plots of the three-body problem in MATLAB (left) and REBOUND (right) [Appendix C-2] [Appendix C-2.1]

```
Final positions and velocities (MATLAB):
Body 1:
  Final Position [m]: x = -5.8505e+08, y = 2.7069e+10, z = 1.5663e+10
  Final Velocity [m/s]: vx = 1.2900e-01, vy = 1.5416e+00, vz = 3.6716e-01
Body 2:
  Final Position [m]: x = -2.0113e+09, y = 3.5335e+10, z = 1.0089e+10
  Final Velocity [m/s]: vx = -4.8590e-01, vy = 1.7508e-01, vz = 8.7424e-01
Body 3:
  Final Position [m]: x = 5.1927e+09, y = 2.5192e+10, z = 1.4498e+10
  Final Velocity [m/s]: vx = 7.1380e-01, vy = 1.5666e+00, vz = -2.8280e-01
```

```
Final positions and velocities (REBOUND):
Body 1:
  Final Position [m]: x = -5.8505e+08, y = 2.7069e+10, z = 1.5663e+10
  Final Velocity [m/s]: vx = 1.2900e-01, vy = 1.5416e+00, vz = 3.6716e-01

Body 2:
  Final Position [m]: x = -2.0113e+09, y = 3.5335e+10, z = 1.0089e+10
  Final Velocity [m/s]: vx = -4.8590e-01, vy = 1.7508e-01, vz = 8.7424e-01

Body 3:
  Final Position [m]: x = 5.1927e+09, y = 2.5192e+10, z = 1.4498e+10
  Final Velocity [m/s]: vx = 7.1380e-01, vy = 1.5666e+00, vz = -2.8280e-01
```

Figure 7. Final positions and velocities of each body in MATLAB (top) and REBOUND (bottom)




```
Initial Energy: -7.3914e+19 J
Final Energy:   -7.3914e+19 J
Δ Energy:       -3.0147e+06 J
Relative Change: 4.1e-14 (4.100e-12%)
```

Figure 8. Total Energy, Momentum plots over time (left) and ΔEnergy change information (right)

D. Five-Body Simulation

D.1. Initial Conditions

- **Masses:** $[m_1, m_2, m_3, m_4, m_5] = [2 \times 10^{20}, 2.5 \times 10^{20}, 10^{20}, 3 \times 10^{20}, 1.5 \times 10^{20}]$ kg
- **Initial Positions:**
  - $r_1 = [-5 \times 10^9, 0, 0]$ m
  - $r_2 = [5 \times 10^9, 0, 7 \times 10^8]$ m
  - $r_3 = [0, 10^9, 2 \times 10^9]$ m
  - $r_4 = [0, 3 \times 10^9, 0]$ m
  - $r_5 = [10^9, 0, 4 \times 10^9]$ m
- **Timespan:** 10^{10} seconds
- **Initial Velocities:**
  - $v_1 = [-0.4, 0.7, 2]$ m/s
  - $v_2 = [2, 0, 0]$ m/s
  - $v_3 = [0, 1.7, -1]$ m/s
  - $v_4 = [0, -1.2, 0.2]$ m/s
  - $v_5 = [-0.5, 0, 0.9]$ m/s
- **Integrators:**
  - MATLAB: ode45
  - REBOUND: ias15
- Check Appendix C-3, C-3.1 and C-3.2

D.2. Output

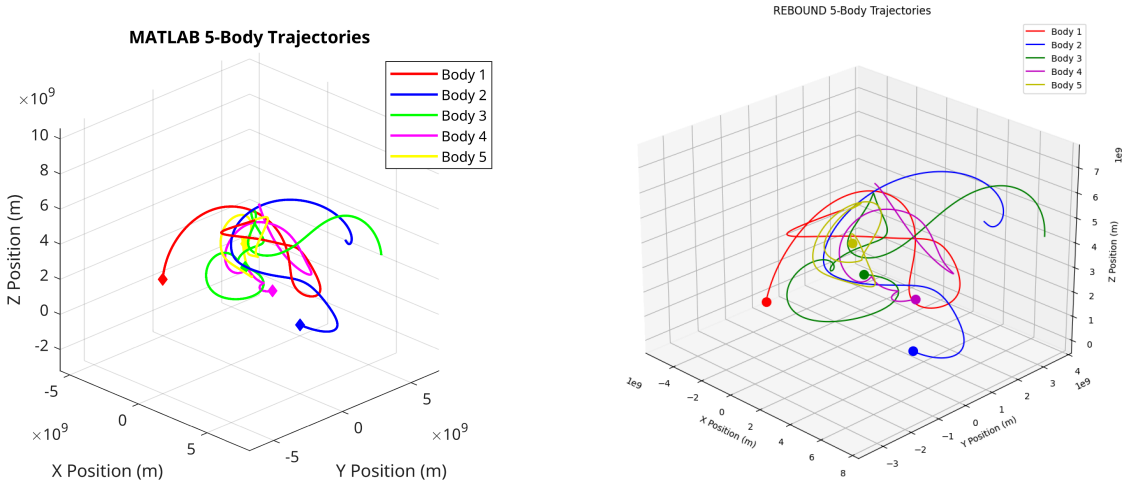
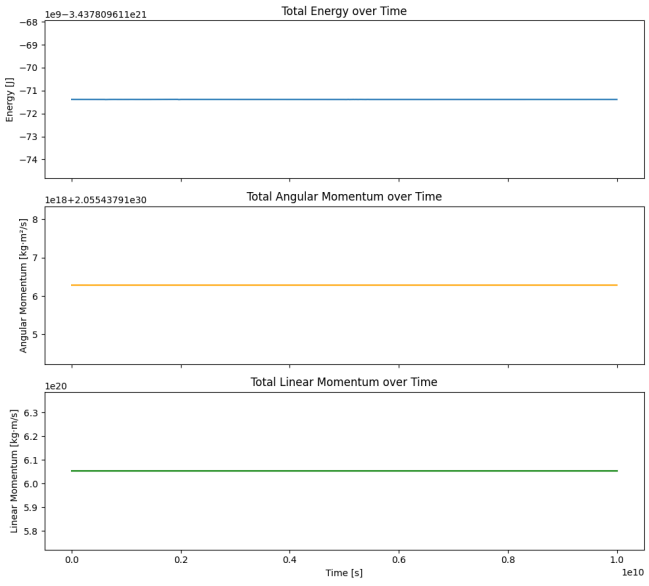


Figure 9. Trajectory plots of the five-body problem in MATLAB (left) and REBOUND (right) [Appendix C-3] [Appendix C-3.1]

```
Final positions and velocities (MATLAB):
Body 1:
  Final Position [m]: x = 3.3156e+09, y = -1.5830e+09, z = 6.6270e+09
  Final Velocity [m/s]: vx = 8.0144e-01, vy = 3.8422e+00, vz = 1.5129e-01
Body 2:
  Final Position [m]: x = 6.7127e+09, y = 1.5805e+09, z = 5.5897e+09
  Final Velocity [m/s]: vx = 4.3090e-03, vy = -5.4647e-01, vz = 7.3922e-01
Body 3:
  Final Position [m]: x = 7.4659e+09, y = 3.4655e+09, z = 4.3411e+09
  Final Velocity [m/s]: vx = 1.3788e+00, vy = -1.0064e+00, vz = -2.3668e+00
Body 4:
  Final Position [m]: x = 1.3678e+09, y = 6.3587e+08, z = 6.1976e+09
  Final Velocity [m/s]: vx = 1.4864e+00, vy = -2.0526e+00, vz = 2.3677e+00
Body 5:
  Final Position [m]: x = 2.3453e+09, y = -7.7215e+08, z = 6.0585e+09
  Final Velocity [m/s]: vx = -2.6678e+00, vy = 2.3066e-01, vz = -1.2912e+00
```

```
Final positions and velocities (REBOUND):
Body 1:
  Final Position [m]: x = 3.3156e+09, y = -1.5830e+09, z = 6.6270e+09
  Final Velocity [m/s]: vx = 8.0144e-01, vy = 3.8422e+00, vz = 1.5129e-01

Body 2:
  Final Position [m]: x = 6.7127e+09, y = 1.5805e+09, z = 5.5897e+09
  Final Velocity [m/s]: vx = 4.3090e-03, vy = -5.4647e-01, vz = 7.3922e-01

Body 3:
  Final Position [m]: x = 7.4659e+09, y = 3.4655e+09, z = 4.3411e+09
  Final Velocity [m/s]: vx = 1.3788e+00, vy = -1.0064e+00, vz = -2.3668e+00

Body 4:
  Final Position [m]: x = 1.3678e+09, y = 6.3587e+08, z = 6.1976e+09
  Final Velocity [m/s]: vx = 1.4864e+00, vy = -2.0526e+00, vz = 2.3677e+00

Body 5:
  Final Position [m]: x = 2.3453e+09, y = -7.7215e+08, z = 6.0585e+09
  Final Velocity [m/s]: vx = -2.6678e+00, vy = 2.3066e-01, vz = -1.2912e+00
```

Figure 10. Final positions and velocities of each body in MATLAB (top) and REBOUND (bottom)




```
Initial Energy: -3.4378e+21 J
Final Energy:   -3.4378e+21 J
Δ Energy:       -2.0972e+06 J
Relative Change: 6.1e-16 (6.100e-14%)
```

Figure 11. Total Energy, Momentum plots over time (left) and ΔEnergy change information (right)

E. Ten-Body Simulation

E.1. Initial Conditions

- **Masses:** $[m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9, m_{10}]$ = [2e20, 2.5e20, 1e20, 3e20, 1.5e20, 1.8e20, 2.2e20, 1.2e20, 2.7e20, 1.4e20] kg
- **Initial Positions:** (r_x = posx)
  - pos1 = [-5e9, 0, 0] m
  - pos2 = [5e9, 0, 7e8] m
  - pos3 = [0, 1e10, 2e9] m
  - pos4 = [0, 3e9, 0] m
  - pos5 = [1e9, 0, 4e9] m
  - pos6 = [-3e9, 4e9, 0] m
  - pos7 = [2e9, -2e9, 3e9] m
  - pos8 = [-4e9, -1e9, -1e9] m
  - pos9 = [0, 0, -5e9] m
  - pos10= [3e9, 5e9, -3e9] m
- **Timespan:** 2×10^9 seconds (+ Momentum and Energy plots until 2×10^{10} seconds)
- **Initial Velocities:** (v_x = velx)
  - vel1 = [-0.4, 0.7, 2] m/s
  - vel2 = [2.0, 0.0, 0] m/s
  - vel3 = [0.0, 1.7, -1] m/s
  - vel4 = [0.0, -1.2, 0.2] m/s
  - vel5 = [-0.5, 0.0, 0.2] m/s
  - vel6 = [0.3, 0.6, 0] m/s
  - vel7 = [-0.7, 0.9, -1.5] m/s
  - vel8 = [1.0, -1.0, 0] m/s
  - vel9 = [0.0, 0.0, 0.8] m/s
  - vel10 = [-0.4, 1.3, -0.9] m/s
- **Integrators:**
  - MATLAB: ode45
  - REBOUND: ias15
- Check Appendix C-4, C-4.1 and C-4.2

E.2. Output

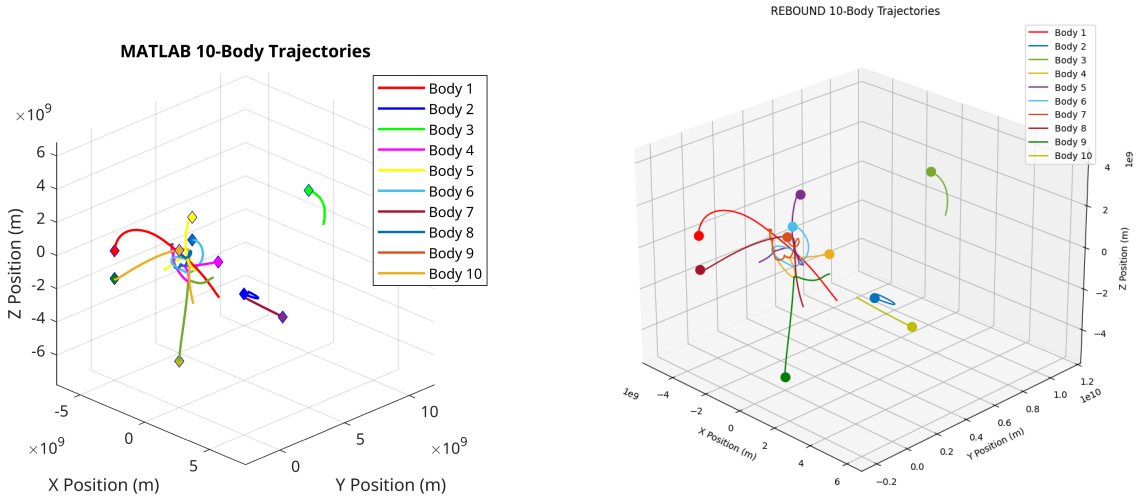
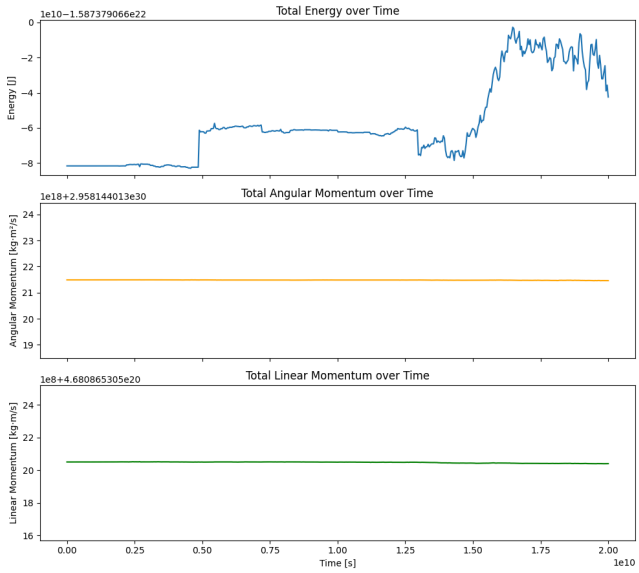


Figure 12. Trajectory plots of the ten-body problem in MATLAB (left) and REBOUND (right) [Appendix C-4] [Appendix C-4.1]

```
Final positions and velocities (MATLAB):
Body 1:
  Final Position [m]: x = 2.3089e+09, y = 7.8282e+08, z = -6.7031e+08
  Final Velocity [m/s]: vx = 6.9939e+00, vy = -2.3070e+00, vz = -1.9992e+00
Body 2:
  Final Position [m]: x = 4.7936e+09, y = 4.9400e+08, z = 6.1816e+08
  Final Velocity [m/s]: vx = -2.8753e+00, vy = 6.0546e-01, vz = -3.5737e-01
Body 3:
  Final Position [m]: x = 1.7778e+06, y = 1.1130e+10, z = -4.6110e+08
  Final Velocity [m/s]: vx = 2.2362e-02, vy = -5.7410e-01, vz = -1.3460e+00
Body 4:
  Final Position [m]: x = -1.7616e+09, y = 1.2625e+09, z = 9.5629e+08
  Final Velocity [m/s]: vx = 4.2643e+00, vy = -2.8173e+00, vz = 4.2753e-01
Body 5:
  Final Position [m]: x = -2.9598e+09, y = 1.7400e+09, z = -1.0740e+09
  Final Velocity [m/s]: vx = -6.1918e+00, vy = 2.7376e+00, vz = -4.4848e+00
Body 6:
  Final Position [m]: x = -8.7521e+08, y = 1.7342e+09, z = 2.4328e+08
  Final Velocity [m/s]: vx = -2.6987e+00, vy = 4.9554e+00, vz = -2.6400e+00
Body 7:
  Final Position [m]: x = -1.9354e+09, y = 1.2824e+09, z = 1.0260e+09
  Final Velocity [m/s]: vx = -5.4346e+00, vy = 3.2730e+00, vz = 8.6992e+00
Body 8:
  Final Position [m]: x = 1.1010e+09, y = -5.3214e+06, z = -1.1752e+09
  Final Velocity [m/s]: vx = 5.7198e+00, vy = -2.7369e+00, vz = -5.6283e+00
Body 9:
  Final Position [m]: x = 2.1267e+09, y = 5.3488e+08, z = 5.7381e+08
  Final Velocity [m/s]: vx = 2.0853e+00, vy = 6.5012e-01, vz = 1.0398e+00
Body 10:
  Final Position [m]: x = 3.3647e+08, y = 4.4649e+09, z = -2.4184e+09
  Final Velocity [m/s]: vx = -2.0840e+00, vy = -2.4224e+00, vz = 1.9592e+00
```

```
Final positions and velocities (REBOUND):
Body 1:
  Final Position [m]: x = 2.3089e+09, y = 7.8282e+08, z = -6.7031e+08
  Final Velocity [m/s]: vx = 6.9939e+00, vy = -2.3070e+00, vz = -1.9992e+00

Body 2:
  Final Position [m]: x = 4.7936e+09, y = 4.9400e+08, z = 6.1816e+08
  Final Velocity [m/s]: vx = -2.8753e+00, vy = 6.0546e-01, vz = -3.5737e-01

Body 3:
  Final Position [m]: x = 1.7778e+06, y = 1.1130e+10, z = -4.6110e+08
  Final Velocity [m/s]: vx = 2.2362e-02, vy = -5.7410e-01, vz = -1.3460e+00

Body 4:
  Final Position [m]: x = -1.7616e+09, y = 1.2625e+09, z = 9.5629e+08
  Final Velocity [m/s]: vx = 4.2643e+00, vy = -2.8173e+00, vz = 4.2753e-01

Body 5:
  Final Position [m]: x = -2.9598e+09, y = 1.7400e+09, z = -1.0740e+09
  Final Velocity [m/s]: vx = -6.1918e+00, vy = 2.7376e+00, vz = -4.4848e+00

Body 6:
  Final Position [m]: x = -8.7521e+08, y = 1.7342e+09, z = 2.4328e+08
  Final Velocity [m/s]: vx = -2.6987e+00, vy = 4.9554e+00, vz = -2.6400e+00

Body 7:
  Final Position [m]: x = -1.9354e+09, y = 1.2824e+09, z = 1.0260e+09
  Final Velocity [m/s]: vx = -5.4346e+00, vy = 3.2730e+00, vz = 8.6992e+00

Body 8:
  Final Position [m]: x = 1.1010e+09, y = -5.3214e+06, z = -1.1752e+09
  Final Velocity [m/s]: vx = 5.7198e+00, vy = -2.7369e+00, vz = -5.6283e+00

Body 9:
  Final Position [m]: x = 2.1267e+09, y = 5.3488e+08, z = 5.7381e+08
  Final Velocity [m/s]: vx = 2.0853e+00, vy = 6.5012e-01, vz = 1.0398e+00

Body 10:
  Final Position [m]: x = 3.3647e+08, y = 4.4649e+09, z = -2.4184e+09
  Final Velocity [m/s]: vx = -2.0840e+00, vy = -2.4224e+00, vz = 1.9592e+00
```

Figure 13. Final positions and velocities of each body in MATLAB (left) and REBOUND (right)




```
Initial Energy: -1.5874e+22 J
Final Energy:   -1.5874e+22 J
Δ Energy:       3.9257e+10 J
Relative Change: 2.5e-12 (2.500e-10%)
```

Figure 14. Total Energy, Momentum plots over time (left) and ΔEnergy change information (right)

F. 100-Body Galaxy Formation Simulation

F.1. Galaxy Formation: Background and Scope

Galaxies are vast gravitational systems composed of stars, gas, dust, dark matter, and other celestial objects. Their formation is driven by the **gravitational collapse of matter in the early universe**, followed by hierarchical mergers and accretion. Over time, regions with higher density attract more mass, leading to the creation of protogalaxies, which evolve into spiral, elliptical, or irregular galaxies depending on angular momentum, environment, and merger history.

In the real universe, even **dwarf galaxies** (the smallest type) typically contain **millions to billions of stars** and span **several hundred to a few thousand light-years** in radius. For instance, ultra-faint dwarf galaxies can have radii of just **100–300 light-years**, while larger dwarf ellipticals may span up to **5,000 light-years**. [25] Despite their size, they are crucial for studying galaxy formation in a cosmological context, particularly because they are thought to resemble the **earliest building blocks** of larger galaxies.

Figure 15. The galaxy M101 as seen by NASA's Hubble Space Telescope (Credit: NASA, ESA, CXC, SSC, and STScI) [26]

Given these enormous scales, simulating galaxy formation in full physical detail requires the use of **supercomputers** and parallel N-body codes like GADGET or Arepo. However, for this study, the simulation is constrained by **personal computing limitations**. As an undergraduate student, I carried out the simulation on a **consumer-grade laptop**, without access to high-performance computing clusters. This required simplifying assumptions; notably, reducing the system to **100 gravitational bodies** and scaling the galaxy's radius down to a few **light-years (on the order of 10^16 meters)**.

While such a model is not representative of the full complexity of galactic dynamics, it provides valuable qualitative insights into **self-gravitating systems**, **orbital evolution**, and **mass clustering**. It also serves as an effective tool for understanding fundamental gravitational behavior in a manageable and visual way.

F.2. Initial Conditions

- **Initial Positions, Velocities and Masses:** bodies.csv [Appendix D-1.2]
- **Timespan:** 1×10^{15} seconds
- **Integrator:** ias15
- Check Appendix D-1.1, D-1.2 and D-1.3

F.3. Output

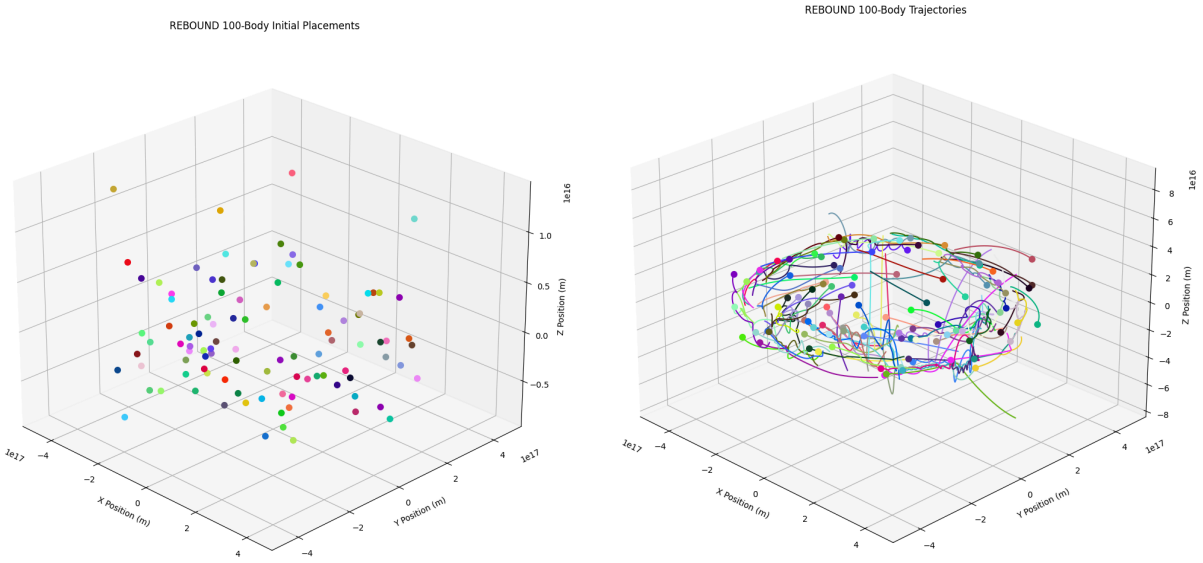
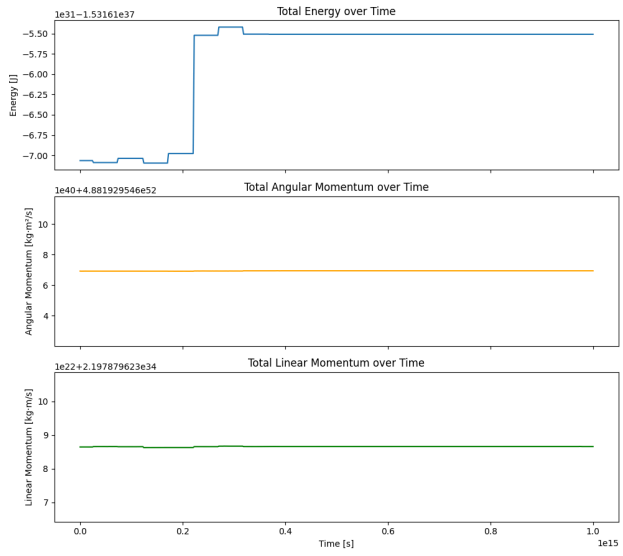


Figure 16. Initial placements of the 100 bodies (left) and their trajectories after simulation (right) [Appendix D-1.1]




```
Initial Energy: -1.5316e+37 J
Final Energy:   -1.5316e+37 J
Δ Energy:        1.5557e+31 J
Relative Change: 1.0e-06 (1.000e-04%)
```

Figure 17. Total Energy, Momentum plots over time (left) and ΔEnergy change information (right)

G. 101-Body Planet Formation Simulation

G.1. Planet Formation Simulation: Background

As mentioned earlier, simulating gravitational systems on a personal computer requires a reduction in scale and complexity. Following the galaxy formation simulation, a second experiment was conducted to model **planet formation**, specifically focusing on the **early stages of a protoplanetary disk** — the flattened, rotating disk of gas and dust surrounding a young star.

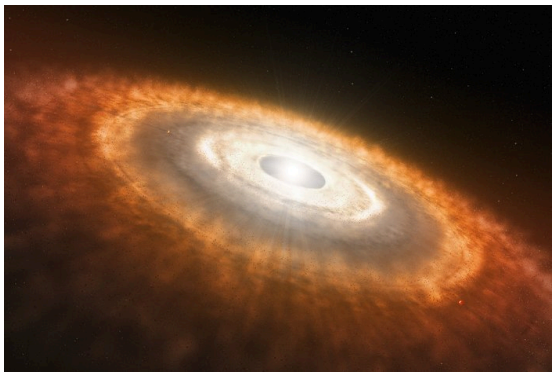


Figure 18. Artist's impression of a baby star still surrounded by a protoplanetary disc in which planets are forming. (Credit: ESO/L. Calçada) [27]

In real astrophysical environments, these disks typically span from a fraction of an astronomical unit (AU) to **tens or even hundreds of AU**, with embedded solid particles gradually coalescing into **planetesimals** and eventually full-sized planets through accretion and collisions. The bodies involved in these early stages range in size from **dust grains** to **moon- and Mars-sized embryos**. Over time, through a combination of **gravitational interactions** and **inelastic collisions**, some bodies grow while others are ejected or absorbed. [28]

In this simulation, 100 bodies were distributed across a rotating disk around a central solar-mass star to emulate this environment. The disk extends up to **30 AU**, and the bodies were given masses consistent with **early-stage planetary embryos** (approximately 10^22–10^24 kg). Again, due to computational constraints, many physical details such as gas drag, turbulence, and detailed material composition are omitted. However, the simplified model is sufficient to illustrate the **emergence of larger bodies through mergers** and the **dynamical evolution of a protoplanetary system** under gravity alone.

G.2. Initial Conditions

- **Initial Positions, Velocities and Masses:** bodies.csv [Appendix D-2.2]
- **Timespan:** 1×10^{11} seconds
- **Integrator:** ias15
- Check Appendix D-2.1, D-2.2 and D-2.3

G.3. Output

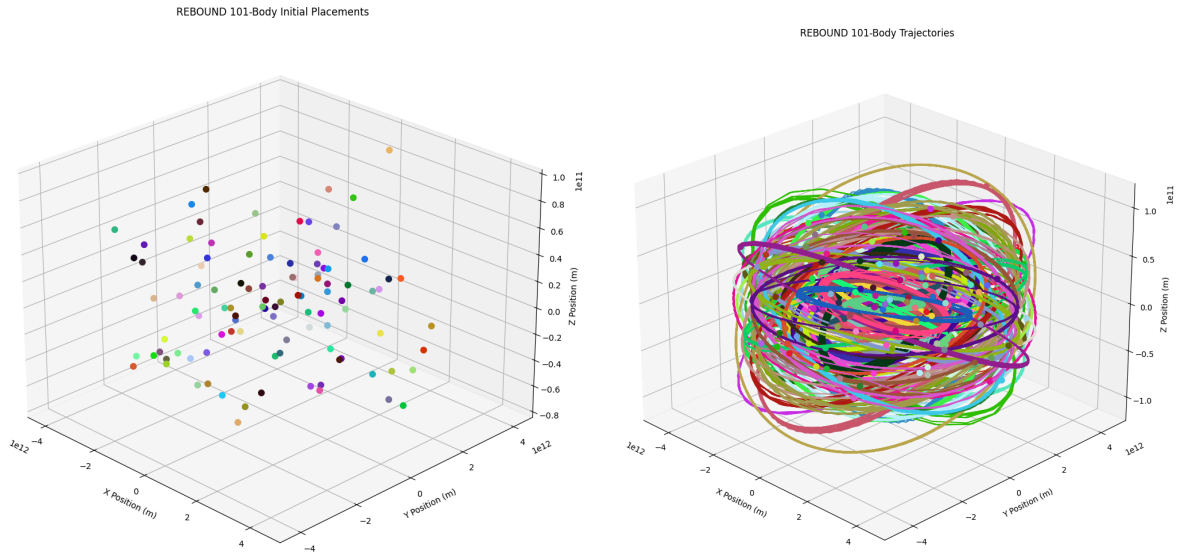
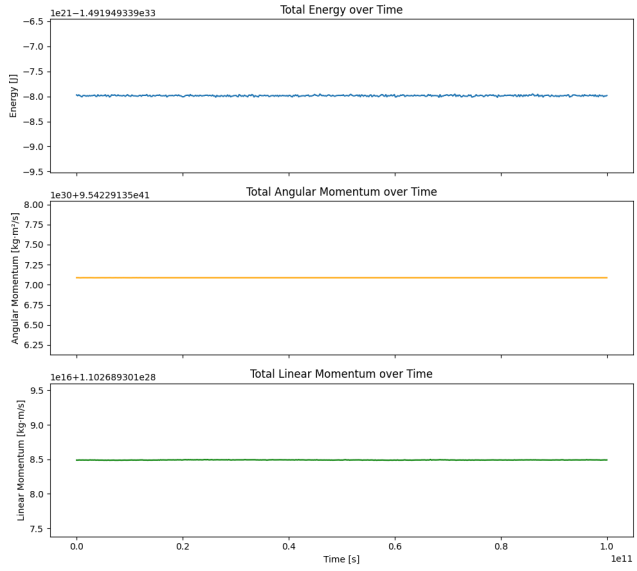


Figure 19. Initial placements of the 101 bodies (left) and their trajectories after simulation (right) [Appendix D-2.1]




```
Initial Energy: -1.4919e+33 J
Final Energy:   -1.4919e+33 J
Δ Energy:       -1.3259e+19 J
Relative Change: 8.9e-15 (8.900e-13%)
```

Figure 20. Total Energy, Momentum plots over time (left) and ΔEnergy change information (right)

V. Results and Conclusion

A. Beginning Simulations

The comparative simulations conducted using **MATLAB** and **REBOUND** for the 2-body, 3-body, 5-body, and 10-body gravitational systems demonstrated highly consistent results in terms of particle trajectories, final positions, and velocities. In all four cases, the overall orbital behavior, symmetry, and conservation properties were closely matched between the two platforms, validating the physical correctness of both implementations.

Despite the high level of agreement, **minor numerical differences** were observed between the final output states, especially in long-term simulations with a greater number of interacting bodies. These discrepancies can be attributed to several factors:

- ➔ **Numerical Integration Methods**: MATLAB utilizes variable-step solvers like `ode45`, which are Runge-Kutta-based, while REBOUND's `IAS15` integrator is a high-order adaptive integrator optimized specifically for gravitational N-body problems. Although both are highly accurate, slight differences in time stepping and rounding behavior may accumulate over time.

- ➔ **Floating-Point Precision and Error Accumulation**: Differences in internal precision handling (e.g., floating-point arithmetic in Python vs. MATLAB) can lead to tiny variations, particularly when dealing with inverse-square gravitational force calculations over thousands of integration steps.

- ➔ **Time Sampling and Output Intervals**: Both simulations use continuous integration, but the way output timesteps are selected (e.g., uniform vs. adaptive) can result in differences in exactly when positions and velocities are recorded, affecting visual alignment.

Nevertheless, the **qualitative and quantitative alignment** of the outputs from both environments confirms that both implementations accurately model Newtonian gravitational dynamics.

These findings demonstrate that both platforms are reliable for educational and research purposes in classical gravitational systems, with REBOUND offering a more specialized toolkit for astrophysical studies, while MATLAB provides flexible prototyping and algorithmic control.

B. Extended Simulations: Galaxy and Planet Formation

Building on these small-N validation cases, two extended simulations involving **100-body systems** were conducted using REBOUND: one modeling a **small-scale galaxy formation**, and another simulating the **early stages of planet formation** in a protoplanetary disk. These larger-scale systems showcased more complex and emergent gravitational behavior, including multi-body clustering, angular momentum redistribution, and collision-driven body growth.

In the galaxy formation simulation, particles initialized in a rotating disk within a few light-years of scale evolved under mutual gravitational attraction. Despite the limited particle count compared to real galaxies, the simulation successfully reproduced large-scale structural features such as **central mass concentration**, **spiral-like arcs**, and **long-range orbital coherence**. Mergers occurred naturally over time, reducing the number of bodies and forming more massive central structures — a behavior consistent with hierarchical galaxy formation models.

The planet formation simulation modeled 100 planetesimals orbiting a central solar-mass star within a 30 AU disk. Over time, several inelastic mergers between bodies led to the formation of larger planetary embryos, echoing the process of **core accretion**. The overall system remained gravitationally bound and dynamically stable over the simulated time span, with several bodies migrating slightly inward or outward due to momentum exchange and gravitational perturbations.

Although these simulations simplify many physical processes (including but not limited to: omitting gas drag, fragmentation, and radiative effects), they provide valuable qualitative insights into the fundamental gravitational interactions that drive structure formation in astrophysical systems. They also demonstrate REBOUND's practical capability to model such complex N-body dynamics on modest computational resources, even without parallelization or GPU acceleration.

VI. References

[1] Springel, V., "The Cosmological Simulation Code Gadget-2," *Monthly Notices of the Royal Astronomical Society*, Vol. 364, No. 4, 2005, pp. 1105–1134. doi:10.1111/j.1365-2966.2005.09655.x.

[2] Rein, H., & Liu, S.-F., "REBOUND: An Open-Source Multi-Purpose N-Body Code," *Astronomy and Astrophysics*, Vol. 537, 2012, A128. doi:10.1051/0004-6361/201118085.

[3] Newton, I., *Philosophiæ Naturalis Principia Mathematica*, 1687.

[4] Press, W. H., et al., *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, Cambridge, 2007, pp. 713–723.

[5] Chandrasekhar, S., *An Introduction to the Study of Stellar Structure*, Dover Publications, 1967, pp. 27–35.

[6] Barnes, J., & Hut, P., "A Hierarchical O(N log N) Force-Calculation Algorithm," *Nature*, Vol. 324, 1986, pp. 446–449. doi:10.1038/324446a0.

[7] Lagrange, J.-L., *Mécanique Analytique*, 1788.

[8] Laplace, P.-S., *Traité de Mécanique Céleste*, Vols. I–V, 1799–1825.

[9] Euler, L., "Essay on the Physical and Mathematical Principles of the Theory of Universal Gravitation," 1760.

[10] Poincaré, H., "Sur le problème des trois corps et les équations de la dynamique," *Acta Mathematica*, Vol. 13, 1890, pp. 1–270. doi:10.1007/BF02392506.

[11] Arnold, V. I., Kozlov, V. V., & Neishtadt, A. I., *Mathematical Aspects of Classical and Celestial Mechanics*, Springer-Verlag, Berlin, Germany, 2006, pp. 45–67.

[12] Virtanen, P., et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, Vol. 17, 2020, pp. 261–272. doi:10.1038/s41592-019-0686-2.

[13] Harris, C. R., et al., "Array Programming with NumPy," *Nature*, Vol. 585, 2020, pp. 357–362. doi:10.1038/s41586-020-2649-2.

[14] Binney, J., & Tremaine, S., *Galactic Dynamics*, 2nd ed., Princeton University Press, Princeton, NJ, 2008, pp. 23–45.

[15] Hut, P., Makino, J., & McMillan, S., "Building a Better Leapfrog Integrator," *Astrophysical Journal Letters*, Vol. 443, 1995, pp. L93–L96. doi:10.1086/187843.

[16] Butcher, J. C., *Numerical Methods for Ordinary Differential Equations*, 3rd ed., Wiley, Hoboken, NJ, 2016, pp. 112–132.

[17] Astropy Collaboration, "The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package," *Astronomy & Astrophysics*, Vol. 637, 2018, A119. doi:10.1051/0004-6361/201843577.

[18] Darve, E. (2000). *The fast multipole method: numerical implementation*. Journal of Computational Physics, 160(1), 195-240.

[19] REBOUND Documentation https://rebound.readthedocs.io/en/latest/

[20] NASA: The Milky Way Galaxy https://imagine.gsfc.nasa.gov/science/objects/milkyway1.html

[21] Matplotlib Documentation https://matplotlib.org/stable/index.html

[22] REBOUND Video 3: Integrators https://www.youtube.com/watch?v=QW5a-iH62dQ

[23] MATLAB Documentation https://www.mathworks.com/help/matlab/index.html

[24] Mini Physics. (n.d.). *Elastic & Inelastic Collisions.* https://www.miniphysics.com/types-of-collision.html

[25] Australian Astronomical Observatory. (2004). *Astronomers discover dozens of mini-galaxies.* https://web.archive.org/web/20180427015808/https://www.aao.gov.au/public/mediarelease/astronomers-discover-dozen-of-mini-galaxies

[26] Harvard University. (n.d.). *Galaxy Formation and Evolution.* https://www.cfa.harvard.edu/research/topic/galaxy-formation-and-evolution

[27] ESO/L. Calçada. (2009). *Burning lithium inside a star (artist's impression)* https://www.eso.org/public/images/eso0942a/

[28] Harvard University. (n.d.). *Planet Formation.* https://www.cfa.harvard.edu/research/topic/planet-formation

VII. Appendices

A. Project Schedule

Simulating Galaxy and Planet Formation Using N-body Simulation
Yunus Sorgunçay - 13/06/2025

Project Schedule

Lecture :	UZB 4901E														UZB 4902E												
Week :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Completed Tasks																												
Literature Review and Research																												
Conducted a thorough review of relevant literature on the N-body problem																												
Investigated key algorithms and foundational theories																												
Tool Selection and Familiarization																												
Evaluated simulation tools such as REBOUND and Gadget-2																												
Tested basic functionality and simple scenarios using REBOUND																												
Initial Simulation Planning																												
Defined project goals and initial conditions for simulations																												
Simulation Implementation																												
Develop scripts and configurations for N-body simulations using REBOUND																												
Run small-scale test simulations to validate the setup and initial conditions																												
UZB 4901E Final Report Writing																												
Document findings and progress, tool evaluations, planned methodologies, and preliminary results																												
Preparation for Advanced Simulations																												
Scale simulations for larger systems																												
Optimize parameters and initial conditions for efficient computation																												
Analyze test data for expected patterns																												
Advanced Simulation Runs																												
Execute simulations with finalized configurations, focusing on specific goals																												
Collect data for analysis																												


Data Analysis and Modeling
Process simulation results
UZB 4902E Interim Report Writing
Model Validation
Compare results with theoretical predictions and other studies
Adjust parameters or methods as necessary
UZB 4902E Final Report Writing and Preparing of Presentation
Compile all findings
Prepare a presentation summarizing the project

B. Statement of Ethics

I hereby declare that this report is my own work except for the sources that have been properly referenced, and completely compliant with the ethics in university life that have been published in https://odek.itu.edu.tr/en/code-of-honor/code-of-honor by ITU Dean of Students. I also declare that it has not been submitted to any other degree at any university.

Yunus Sorgunçay

C. Two-, Three-, Five-, Ten-Body Problem Codes

1. two_body.m (MATLAB)

```
clear; clc; close all;

%% Constants and Parameters
G = 6.6742867e-11;           % Gravitational Constant [N·m²/kg²]
numBodies = 2;               % Number of bodies

% Masses of the bodies [kg]
masses = [2e20, 5e20];

% Characteristic length (initial distance between bodies) [m]
L = 1e10;

%% Initial Conditions Setup
% Compute positions relative to the system's center of mass
Lm1 = masses(2)/(masses(1) + masses(2)) * L;
Lm2 = masses(1)/(masses(1) + masses(2)) * L;

% Angular velocity for circular orbit [rad/s]
omega = sqrt(G * sum(masses) / L^3);

% Initial positions for each body
pos1 = [-Lm1, 0, 0];
pos2 = [Lm2, 0, 0];

% Initial velocities for stable circular orbit
vel1 = [0, -omega * Lm1, 0];
vel2 = [0, omega * Lm2, 0];

% Initial state vector
initialConditions = [pos1, vel1, pos2, vel2];

%% Simulation Time Span
orbitalPeriod = 2*pi / omega;  % Orbital period for the two-body system
timeSpan = [0, orbitalPeriod];

% Solver accuracy options
solverOptions = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);

% Solve equations of motion
[time, solution] = ode45(@(t, y) nBodyEquations(t, y, masses, G, numBodies),...
                         timeSpan, initialConditions, solverOptions);

%% Visualization of Trajectories
figure; hold on; grid on;

% Trajectory plots for each body
plot3(solution(:,1), solution(:,2), solution(:,3), 'r', 'LineWidth', 1.5);
plot3(solution(:,7), solution(:,8), solution(:,9), 'b', 'LineWidth', 1.5);

% Initial position markers
plot3(solution(1,1), solution(1,2), solution(1,3), 'dr', 'MarkerFaceColor','r');
plot3(solution(1,7), solution(1,8), solution(1,9), 'db', 'MarkerFaceColor','b');
```

```
% Formatting the plot
axis equal;
axisLimit = 1 * L;
xlim([-axisLimit, axisLimit]);
ylim([-axisLimit, axisLimit]);
zlim([-axisLimit, axisLimit]);

xlabel('X Position (m)');
ylabel('Y Position (m)');
zlabel('Z Position (m)');
title('MATLAB 2-Body Trajectories');
legend('Body 1','Body 2');

view([45, 25]); % set view angle

% Extract final positions and velocities
finalState = solution(end, :);

fprintf('Final positions and velocities (MATLAB):\n');
for i = 1:numBodies
    idx = (i-1)*6;
    fprintf('Body %d:\n', i);
    fprintf('  Final Position [m]: x = %.4e, y = %.4e, z = %.4e\n', ...
        finalState(idx+1), finalState(idx+2), finalState(idx+3));
    fprintf('  Final Velocity [m/s]: vx = %.4e, vy = %.4e, vz = %.4e\n\n', ...
        finalState(idx+4), finalState(idx+5), finalState(idx+6));
end

%% Function Defining the N-body System of Equations
function dydt = nBodyEquations(~, y, masses, G, numBodies)
    % Initialize derivatives vector
    dydt = zeros(6*numBodies,1);

    % Loop over each body to calculate forces and accelerations
    for i = 1:numBodies
        force = [0;0;0];  % Initialize gravitational force vector
        idx_pos_i = (i-1)*6 + (1:3); % Position indices
        idx_vel_i = (i-1)*6 + (4:6); % Velocity indices

        position_i = y(idx_pos_i);

        % Compute gravitational interaction with other bodies
        for j = 1:numBodies
            if i ~= j
                idx_pos_j = (j-1)*6 + (1:3);
                position_j = y(idx_pos_j);

                r_vec = position_j - position_i; % Vector from body i to j
                r_mag = norm(r_vec);             % Distance magnitude

                % Newton's law of gravitation
                force = force + (G * masses(i) * masses(j) / r_mag^3) * r_vec;
            end
        end
```

```
        % Assign velocity components as derivatives of position
        dydt(idx_pos_i) = y(idx_vel_i);

        % Assign acceleration components (force divided by mass)
        dydt(idx_vel_i) = force / masses(i);
    end
end
```

1.1 rebound2body.py (REBOUND)

```
import rebound
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random

# Constants and Parameters
G = 6.6742867e-11          # Gravitational constant [N·m²/kg²]
num_bodies = 2             # Number of bodies

# Load CSV
body_data = pd.read_csv("bodies.csv")

# Optional: use only the number of bodies you want
body_data = body_data.iloc[:num_bodies]

# Setup simulation
sim = rebound.Simulation()
sim.integrator = "ias15"
sim.dt = 1e-6 # small time-step
sim.units = ('m', 's', 'kg')  # units
sim.G = G
sim.collision = "direct"
sim.collision_resolve = "merge"

# Adding bodies with initial conditions
for index, row in body_data.iterrows():
    sim.add(
        m=row['mass'],
        x=row['x'], y=row['y'], z=row['z'],
        vx=row['vx'], vy=row['vy'], vz=row['vz']
    )

# Simulation timespan (one orbital period) [s]
orbital_period = 2 * np.pi / np.sqrt(6.6742867e-11 * 7e20 / 1e10**3)
N_outputs = 500
times = np.linspace(0, orbital_period, N_outputs)

# Arrays to store positions for plotting
positions = np.zeros((num_bodies, N_outputs, 3))

energy_vals = []
momentum_vals = []
linear_momentum_vals = []
```

```
# Integrate simulation
for i, t in enumerate(times):
    sim.integrate(t)

    # Store energy and angular momentum
    energy_vals.append(sim.energy())
    Lx, Ly, Lz = sim.angular_momentum()
    momentum_vals.append(np.sqrt(Lx**2 + Ly**2 + Lz**2))

    # Compute total linear momentum
    px = py = pz = 0.0
    for p in sim.particles:
        px += p.m * p.vx
        py += p.m * p.vy
        pz += p.m * p.vz
    total_p = (px**2 + py**2 + pz**2)**0.5
    linear_momentum_vals.append(total_p)

    for j, particle in enumerate(sim.particles):
        positions[j, i, :] = [particle.x, particle.y, particle.z]

# Plot 3D trajectories
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

# Generate random RGB colors for each body
colors = [(random.random(), random.random(), random.random()) for _ in range(num_bodies)]

for i in range(num_bodies):
    ax.plot3D(positions[i,:,0], positions[i,:,1], positions[i,:,2], color=colors[i])
    ax.scatter(positions[i,0,0], positions[i,0,1], positions[i,0,2], color=colors[i], marker='o',
s=100)

print("\nFinal positions and velocities (REBOUND):")
for i, particle in enumerate(sim.particles):
    print(f"Body {i+1}:")
    print(f"  Final Position [m]: x = {particle.x:.4e}, y = {particle.y:.4e}, z =
{particle.z:.4e}")
    print(f"  Final Velocity [m/s]: vx = {particle.vx:.4e}, vy = {particle.vy:.4e}, vz =
{particle.vz:.4e}\n")

initial_energy = energy_vals[0]
final_energy = energy_vals[-1]
delta_energy = final_energy - initial_energy

print(f"Initial Energy: {initial_energy:.4e} J")
print(f"Final Energy:   {final_energy:.4e} J")
print(f"Δ Energy:       {delta_energy:.4e} J")
rel_change = abs(delta_energy / initial_energy)
sig_fig_format = f"{rel_change:.1e}"
print(f"Relative Change: {sig_fig_format} ({float(sig_fig_format)*100:.3e}%)")

# Formatting plot
ax.set_xlabel('X Position (m)')
ax.set_ylabel('Y Position (m)')
ax.set_zlabel('Z Position (m)')
```

```
ax.set_title(f'REBOUND {num_bodies}-Body Trajectories')
ax.view_init(elev=25, azim=-45)
plt.tight_layout()
plt.show()

# Plot Energy, Angular Momentum, and Linear Momentum
fig, ax = plt.subplots(3, 1, figsize=(10, 9), sharex=True)

# Energy plot
ax[0].plot(times, energy_vals)
ax[0].set_ylabel("Energy [J]")
ax[0].set_title("Total Energy over Time")

# Angular momentum plot
ax[1].plot(times, momentum_vals, color="orange")
ax[1].set_ylabel("Angular Momentum [kg·m²/s]")
ax[1].set_title("Total Angular Momentum over Time")

# Linear momentum plot
ax[2].plot(times, linear_momentum_vals, color="green")
ax[2].set_ylabel("Linear Momentum [kg·m/s]")
ax[2].set_xlabel("Time [s]")
ax[2].set_title("Total Linear Momentum over Time")
ax[2].set_ylim(-8e5, 8.5e5)

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

1.2 bodies.csv (REBOUND)

```
mass,x,y,z,vx,vy,vz
2e20,-7.142857142857143e+09,0,0,0,-1.543915095278420,0
5e20,2.857142857142857e+09,0,0,0,0.617566038111368,0
```

2. three_body.m (MATLAB)

```
clear; clc; close all;

%% Constants and Parameters
G = 6.6742867e-11;           % Gravitational Constant [N·m²/kg²]
numBodies = 3;               % Number of bodies

% Masses of the bodies [kg]
masses = [2e20, 2e20, 1e20];

% Initial positions for each mass [m]
pos1 = [-5e9, 0, 0];
pos2 = [5e9, 0, 0];
pos3 = [0, 0, 0];

% Initial velocities for each mass [m/s]
vel1 = [-1, 0.5, 1];
vel2 = [1, 2, 0];
```

```
vel3 = [0, 0, 0.2];

% Initial condition vector
initialConditions = [pos1, vel1, pos2, vel2, pos3, vel3];

%% Simulation Time Span [s]
timeSpan = [0, 3e10];

% Set solver accuracy
solverOptions = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);

% Solve the system of ODEs
[time, solution] = ode45(@(t, y) nBodyEquations(t, y, masses, G, numBodies),...
                         timeSpan, initialConditions, solverOptions);

%% Visualization
figure; hold on; grid on;
% Trajectories
plot3(solution(:,1), solution(:,2), solution(:,3),'r','LineWidth',1.5);
plot3(solution(:,7), solution(:,8), solution(:,9),'b','LineWidth',1.5);
plot3(solution(:,13), solution(:,14), solution(:,15),'g','LineWidth',1.5);

% Initial positions markers
plot3(solution(1,1), solution(1,2), solution(1,3),'dr','MarkerFaceColor','r');
plot3(solution(1,7), solution(1,8), solution(1,9),'db','MarkerFaceColor','b');
plot3(solution(1,13), solution(1,14), solution(1,15),'dg','MarkerFaceColor','g');

% Plot formatting
axis equal;
axis auto; % automatically fits axis to data

% Positions extraction
x_positions = solution(:, 1:6:end);
y_positions = solution(:, 2:6:end);
z_positions = solution(:, 3:6:end);

padding_factor = 0.1; % 10% padding

% Axis min-max calculations
x_min = min(x_positions(:)); x_max = max(x_positions(:));
y_min = min(y_positions(:)); y_max = max(y_positions(:));
z_min = min(z_positions(:)); z_max = max(z_positions(:));

% Range calculations
x_range = x_max - x_min;
y_range = y_max - y_min;
z_range = z_max - z_min;

% Use largest range for equal axes
max_range = max([x_range, y_range, z_range]);

% Calculate midpoints for each axis
x_mid = (x_max + x_min)/2;
y_mid = (y_max + y_min)/2;
z_mid = (z_max + z_min)/2;
```

```
half_range_with_padding = (max_range / 2) * (1 + padding_factor);

% Cube-like axis limits around midpoints
xlim([x_mid - half_range_with_padding, x_mid + half_range_with_padding]);
ylim([y_mid - half_range_with_padding, y_mid + half_range_with_padding]);
zlim([z_mid - half_range_with_padding, z_mid + half_range_with_padding]);

xlabel('X Position (m)');
ylabel('Y Position (m)');
zlabel('Z Position (m)');
title('MATLAB 3-Body Trajectories');
legend('Body 1','Body 2','Body 3');

view([45, 25]); % set 3D view angle

% Extract final positions and velocities
finalState = solution(end, :);

fprintf('Final positions and velocities (MATLAB):\n');
for i = 1:numBodies
    idx = (i-1)*6;
    fprintf('Body %d:\n', i);
    fprintf('  Final Position [m]: x = %.4e, y = %.4e, z = %.4e\n', ...
        finalState(idx+1), finalState(idx+2), finalState(idx+3));
    fprintf('  Final Velocity [m/s]: vx = %.4e, vy = %.4e, vz = %.4e\n\n', ...
        finalState(idx+4), finalState(idx+5), finalState(idx+6));
end

%% Function defining the system of ODEs
function dydt = nBodyEquations(~, y, masses, G, numBodies)
    % Initialize derivatives vector
    dydt = zeros(6*numBodies,1);

    for i = 1:numBodies
        force = [0;0;0]; % Ensure column vector format
        idx_pos_i = (i-1)*6 + (1:3);
        idx_vel_i = (i-1)*6 + (4:6);

        position_i = y(idx_pos_i);

        % Compute gravitational forces from other bodies
        for j = 1:numBodies
            if i ~= j
                idx_pos_j = (j-1)*6 + (1:3);
                position_j = y(idx_pos_j);

                r_vec = position_j - position_i;
                r_mag = norm(r_vec);

                force = force + (G * masses(i) * masses(j)/r_mag^3) * r_vec;
            end
        end

        % Derivatives for position (velocity components)
        dydt(idx_pos_i) = y(idx_vel_i);
```

```
        % Derivatives for velocity (acceleration components)
        dydt(idx_vel_i) = force / masses(i);
    end
end
```

2.1 rebound3body.py (REBOUND)

```
import rebound
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random

# Constants and Parameters
G = 6.6742867e-11          # Gravitational constant [N·m²/kg²]
num_bodies = 3             # Number of bodies

# Load CSV
body_data = pd.read_csv("bodies.csv")

# Optional: use only the number of bodies you want
body_data = body_data.iloc[:num_bodies]

# Setup simulation
sim = rebound.Simulation()
sim.integrator = "ias15"
sim.dt = 1e-6 # small time-step
sim.units = ('m', 's', 'kg')  # units
sim.G = G
sim.collision = "direct"
sim.collision_resolve = "merge"

# Adding bodies with initial conditions
for index, row in body_data.iterrows():
    sim.add(
        m=row['mass'],
        x=row['x'], y=row['y'], z=row['z'],
        vx=row['vx'], vy=row['vy'], vz=row['vz']
    )

# Simulation timespan (s)
N_outputs = 500
times = np.linspace(0, 3e10, N_outputs)

# Arrays to store positions for plotting
positions = np.zeros((num_bodies, N_outputs, 3))

energy_vals = []
momentum_vals = []
linear_momentum_vals = []

# Integrate simulation
for i, t in enumerate(times):
    sim.integrate(t)
```

```
    # Store energy and angular momentum
    energy_vals.append(sim.energy())
    Lx, Ly, Lz = sim.angular_momentum()
    momentum_vals.append(np.sqrt(Lx**2 + Ly**2 + Lz**2))

    # Compute total linear momentum
    px = py = pz = 0.0
    for p in sim.particles:
        px += p.m * p.vx
        py += p.m * p.vy
        pz += p.m * p.vz
    total_p = (px**2 + py**2 + pz**2)**0.5
    linear_momentum_vals.append(total_p)

    for j, particle in enumerate(sim.particles):
        positions[j, i, :] = [particle.x, particle.y, particle.z]

# Plot 3D trajectories
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

# Generate random RGB colors for each body
colors = [(random.random(), random.random(), random.random()) for _ in range(num_bodies)]

for i in range(num_bodies):
    ax.plot3D(positions[i,:,0], positions[i,:,1], positions[i,:,2], color=colors[i])
    ax.scatter(positions[i,0,0], positions[i,0,1], positions[i,0,2], color=colors[i], marker='o',
s=100)

print("\nFinal positions and velocities (REBOUND):")
for i, particle in enumerate(sim.particles):
    print(f"Body {i+1}:")
    print(f"  Final Position [m]: x = {particle.x:.4e}, y = {particle.y:.4e}, z =
{particle.z:.4e}")
    print(f"  Final Velocity [m/s]: vx = {particle.vx:.4e}, vy = {particle.vy:.4e}, vz =
{particle.vz:.4e}\n")

initial_energy = energy_vals[0]
final_energy = energy_vals[-1]
delta_energy = final_energy - initial_energy

print(f"Initial Energy: {initial_energy:.4e} J")
print(f"Final Energy:   {final_energy:.4e} J")
print(f"Δ Energy:       {delta_energy:.4e} J")
rel_change = abs(delta_energy / initial_energy)
sig_fig_format = f"{rel_change:.1e}"
print(f"Relative Change: {sig_fig_format} ({float(sig_fig_format)*100:.3e}%)")

# Formatting plot
ax.set_xlabel('X Position (m)')
ax.set_ylabel('Y Position (m)')
ax.set_zlabel('Z Position (m)')
ax.set_title(f'REBOUND {num_bodies}-Body Trajectories')
ax.view_init(elev=25, azim=-45)
plt.tight_layout()
plt.show()
```

```
# Plot Energy, Angular Momentum, and Linear Momentum
fig, ax = plt.subplots(3, 1, figsize=(10, 9), sharex=True)

# Energy plot
ax[0].plot(times, energy_vals)
ax[0].set_ylabel("Energy [J]")
ax[0].set_title("Total Energy over Time")

# Angular momentum plot
ax[1].plot(times, momentum_vals, color="orange")
ax[1].set_ylabel("Angular Momentum [kg·m²/s]")
ax[1].set_title("Total Angular Momentum over Time")

# Linear momentum plot
ax[2].plot(times, linear_momentum_vals, color="green")
ax[2].set_ylabel("Linear Momentum [kg·m/s]")
ax[2].set_xlabel("Time [s]")
ax[2].set_title("Total Linear Momentum over Time")

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

2.2 bodies.csv (REBOUND)

```
mass,x,y,z,vx,vy,vz
2e20,-5e9,0,0,-1,0.5,1
2e20,5e9,0,0,1,2,0
1e20,0,0,0,0.0,0,0.2
```

3. five_body.m (MATLAB)

```
clear; clc; close all;

%% Constants and Parameters
G = 6.6742867e-11;           % Gravitational Constant [N·m²/kg²]
numBodies = 5;               % Number of bodies

% Masses of the bodies [kg]
masses = [2e20, 2.5e20, 1e20, 3e20, 1.5e20];

% Initial positions for each mass [m]
pos1 = [-5e9, 0, 0];
pos2 = [5e9, 0, 7e8];
pos3 = [0, 10e8, 2e9];
pos4 = [0, 3e9, 0];
pos5 = [1e9, 0, 4e9];

% Initial velocities for each mass [m/s]
vel1 = [-0.4, 0.7, 2];
vel2 = [2, 0, 0];
vel3 = [0, 1.7, -1];
vel4 = [0, -1.2, 0.2];
```

```
vel5 = [-0.5, 0, 0.9];

% Initial condition vector
initialConditions = [pos1, vel1, pos2, vel2, pos3,...
    vel3, pos4, vel4, pos5, vel5];

%% Simulation Time Span [s]
timeSpan = [0, 1e10];

% Set solver accuracy
solverOptions = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);

% Solve the system of ODEs
[time, solution] = ode45(@(t, y) nBodyEquations(t, y, masses, G, numBodies),...
                         timeSpan, initialConditions, solverOptions);

%% Visualization
figure; hold on; grid on;
% Trajectories
plot3(solution(:,1), solution(:,2), solution(:,3),'r','LineWidth',1.5);
plot3(solution(:,7), solution(:,8), solution(:,9),'b','LineWidth',1.5);
plot3(solution(:,13), solution(:,14), solution(:,15),'g','LineWidth',1.5);
plot3(solution(:,19), solution(:,20), solution(:,21),'m','LineWidth',1.5);
plot3(solution(:,25), solution(:,26), solution(:,27),'y','LineWidth',1.5);

% Initial positions markers
plot3(solution(1,1), solution(1,2), solution(1,3),'dr','MarkerFaceColor','r');
plot3(solution(1,7), solution(1,8), solution(1,9),'db','MarkerFaceColor','b');
plot3(solution(1,13), solution(1,14), solution(1,15),'dg','MarkerFaceColor','g');
plot3(solution(1,19), solution(1,20), solution(1,21),'dm','MarkerFaceColor','m');
plot3(solution(1,25), solution(1,26), solution(1,27),'dy','MarkerFaceColor','y');

% Plot formatting
axis equal;
axis auto; % automatically fits axis to data

% Positions extraction
x_positions = solution(:, 1:6:end);
y_positions = solution(:, 2:6:end);
z_positions = solution(:, 3:6:end);

padding_factor = 0.1; % 10% padding

% Axis min-max calculations
x_min = min(x_positions(:)); x_max = max(x_positions(:));
y_min = min(y_positions(:)); y_max = max(y_positions(:));
z_min = min(z_positions(:)); z_max = max(z_positions(:));

% Range calculations
x_range = x_max - x_min;
y_range = y_max - y_min;
z_range = z_max - z_min;

% Use largest range for equal axes
max_range = max([x_range, y_range, z_range]);
```

```
% Calculate midpoints for each axis
x_mid = (x_max + x_min)/2;
y_mid = (y_max + y_min)/2;
z_mid = (z_max + z_min)/2;

half_range_with_padding = (max_range / 2) * (1 + padding_factor);

% Cube-like axis limits around midpoints
xlim([x_mid - half_range_with_padding, x_mid + half_range_with_padding]);
ylim([y_mid - half_range_with_padding, y_mid + half_range_with_padding]);
zlim([z_mid - half_range_with_padding, z_mid + half_range_with_padding]);

xlabel('X Position (m)');
ylabel('Y Position (m)');
zlabel('Z Position (m)');
title('MATLAB 5-Body Trajectories');
legend('Body 1','Body 2','Body 3','Body 4','Body 5');

view([45, 25]); % set 3D view angle

% Extract final positions and velocities
finalState = solution(end, :);

fprintf('Final positions and velocities (MATLAB):\n');
for i = 1:numBodies
    idx = (i-1)*6;
    fprintf('Body %d:\n', i);
    fprintf('  Final Position [m]: x = %.4e, y = %.4e, z = %.4e\n', ...
        finalState(idx+1), finalState(idx+2), finalState(idx+3));
    fprintf('  Final Velocity [m/s]: vx = %.4e, vy = %.4e, vz = %.4e\n\n', ...
        finalState(idx+4), finalState(idx+5), finalState(idx+6));
end

%% Function defining the system of ODEs
function dydt = nBodyEquations(~, y, masses, G, numBodies)
    % Initialize derivatives vector
    dydt = zeros(6*numBodies,1);

    for i = 1:numBodies
        force = [0;0;0]; % Ensure column vector format
        idx_pos_i = (i-1)*6 + (1:3);
        idx_vel_i = (i-1)*6 + (4:6);

        position_i = y(idx_pos_i);

        % Compute gravitational forces from other bodies
        for j = 1:numBodies
            if i ~= j
                idx_pos_j = (j-1)*6 + (1:3);
                position_j = y(idx_pos_j);

                r_vec = position_j - position_i;
                r_mag = norm(r_vec);

                force = force + (G * masses(i) * masses(j)/r_mag^3) * r_vec;
            end
```

```
        end

        % Derivatives for position (velocity components)
        dydt(idx_pos_i) = y(idx_vel_i);

        % Derivatives for velocity (acceleration components)
        dydt(idx_vel_i) = force / masses(i);
    end
end
```

3.1 rebound5body.py (REBOUND)

```
import rebound
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random

# Constants and Parameters
G = 6.6742867e-11          # Gravitational constant [N·m²/kg²]
num_bodies = 5             # Number of bodies

# Load CSV
body_data = pd.read_csv("bodies.csv")

# Optional: use only the number of bodies you want
body_data = body_data.iloc[:num_bodies]

# Setup simulation
sim = rebound.Simulation()
sim.integrator = "ias15"
sim.dt = 1e-6 # small time-step
sim.units = ('m', 's', 'kg')  # units
sim.G = G
sim.collision = "direct"
sim.collision_resolve = "merge"

# Adding bodies with initial conditions
for index, row in body_data.iterrows():
    sim.add(
        m=row['mass'],
        x=row['x'], y=row['y'], z=row['z'],
        vx=row['vx'], vy=row['vy'], vz=row['vz']
    )

# Simulation timespan (s)
N_outputs = 500
times = np.linspace(0, 1e10, N_outputs)

# Arrays to store positions for plotting
positions = np.zeros((num_bodies, N_outputs, 3))

energy_vals = []
momentum_vals = []
linear_momentum_vals = []
```

```
# Integrate simulation
for i, t in enumerate(times):
    sim.integrate(t)

    # Store energy and angular momentum
    energy_vals.append(sim.energy())
    Lx, Ly, Lz = sim.angular_momentum()
    momentum_vals.append(np.sqrt(Lx**2 + Ly**2 + Lz**2))

    # Compute total linear momentum
    px = py = pz = 0.0
    for p in sim.particles:
        px += p.m * p.vx
        py += p.m * p.vy
        pz += p.m * p.vz
    total_p = (px**2 + py**2 + pz**2)**0.5
    linear_momentum_vals.append(total_p)

    for j, particle in enumerate(sim.particles):
        positions[j, i, :] = [particle.x, particle.y, particle.z]

# Plot 3D trajectories
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

# Generate random RGB colors for each body
colors = [(random.random(), random.random(), random.random()) for _ in range(num_bodies)]

for i in range(num_bodies):
    ax.plot3D(positions[i,:,0], positions[i,:,1], positions[i,:,2], color=colors[i])
    ax.scatter(positions[i,0,0], positions[i,0,1], positions[i,0,2], color=colors[i], marker='o',
s=100)

print("\nFinal positions and velocities (REBOUND):")
for i, particle in enumerate(sim.particles):
    print(f"Body {i+1}:")
    print(f"  Final Position [m]: x = {particle.x:.4e}, y = {particle.y:.4e}, z =
{particle.z:.4e}")
    print(f"  Final Velocity [m/s]: vx = {particle.vx:.4e}, vy = {particle.vy:.4e}, vz =
{particle.vz:.4e}\n")

initial_energy = energy_vals[0]
final_energy = energy_vals[-1]
delta_energy = final_energy - initial_energy

print(f"Initial Energy: {initial_energy:.4e} J")
print(f"Final Energy:   {final_energy:.4e} J")
print(f"Δ Energy:       {delta_energy:.4e} J")
rel_change = abs(delta_energy / initial_energy)
sig_fig_format = f"{rel_change:.1e}"
print(f"Relative Change: {sig_fig_format} ({float(sig_fig_format)*100:.3e}%)")

# Formatting plot
ax.set_xlabel('X Position (m)')
ax.set_ylabel('Y Position (m)')
```

```
ax.set_zlabel('Z Position (m)')
ax.set_title(f'REBOUND {num_bodies}-Body Trajectories')
ax.view_init(elev=25, azim=-45)
plt.tight_layout()
plt.show()

# Plot Energy, Angular Momentum, and Linear Momentum
fig, ax = plt.subplots(3, 1, figsize=(10, 9), sharex=True)

# Energy plot
ax[0].plot(times, energy_vals)
ax[0].set_ylabel("Energy [J]")
ax[0].set_title("Total Energy over Time")

# Angular momentum plot
ax[1].plot(times, momentum_vals, color="orange")
ax[1].set_ylabel("Angular Momentum [kg·m²/s]")
ax[1].set_title("Total Angular Momentum over Time")

# Linear momentum plot
ax[2].plot(times, linear_momentum_vals, color="green")
ax[2].set_ylabel("Linear Momentum [kg·m/s]")
ax[2].set_xlabel("Time [s]")
ax[2].set_title("Total Linear Momentum over Time")

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

3.2 bodies.csv (REBOUND)

```
mass,x,y,z,vx,vy,vz
2e20,-5e9,0,0,-0.4,0.7,2
2.5e20,5e9,0,7e8,2.0,0.0,0
1e20,0,10e8,2e9,0.0,1.7,-1
3e20,0,3e9,0,0.0,-1.2,0.2
1.5e20,1e9,0,4e9,-0.5,0.0,0.9
```

4. ten_body.m (MATLAB)

```
clear; clc; close all;

%% Constants and Parameters
G = 6.6742867e-11;           % Gravitational Constant [N·m²/kg²]
numBodies = 10;               % Number of bodies

% Masses of the bodies [kg]
masses = [2e20, 2.5e20, 1e20, 3e20, 1.5e20, ...
          1.8e20, 2.2e20, 1.2e20, 2.7e20, 1.4e20];

% Initial positions for each mass [m]
pos1 = [-5e9,   0,     0];
pos2 = [ 5e9,   0,     7e8];
pos3 = [ 0,     1e10,  2e9];
```

```
pos4 = [ 0,     3e9,   0];
pos5 = [ 1e9,   0,     4e9];
pos6 = [-3e9,   4e9,   0];
pos7 = [ 2e9,  -2e9,   3e9];
pos8 = [-4e9,  -1e9,  -1e9];
pos9 = [ 0,     0,    -5e9];
pos10= [ 3e9,   5e9,  -3e9];

% Initial velocities for each mass [m/s]
vel1  = [-0.4,  0.7,   2];
vel2  = [ 2.0,  0.0,   0];
vel3  = [ 0.0,  1.7,  -1];
vel4  = [ 0.0, -1.2,   0.2];
vel5  = [-0.5,  0.0,   0.2];
vel6  = [ 0.3,  0.6,   0];
vel7  = [-0.7,  0.9,  -1.5];
vel8  = [ 1.0, -1.0,   0];
vel9  = [ 0.0,  0.0,   0.8];
vel10 = [-0.4,  1.3,  -0.9];

% Initial condition vector
initialConditions = [pos1, vel1, pos2, vel2, pos3,...
    vel3, pos4, vel4, pos5, vel5, pos6, vel6, pos7, vel7, pos8,...
    vel8, pos9, vel9, pos10, vel10];

%% Simulation Time Span [s]
timeSpan = [0, 0.2e10];

% Set solver accuracy
solverOptions = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);

% Solve the system of ODEs
[time, solution] = ode45(@(t, y) nBodyEquations(t, y, masses, G, numBodies),...
                         timeSpan, initialConditions, solverOptions);

%% Visualization
figure; hold on; grid on;
% Trajectories
plot3(solution(:,1), solution(:,2), solution(:,3),'r','LineWidth',1.5);
plot3(solution(:,7), solution(:,8), solution(:,9),'b','LineWidth',1.5);
plot3(solution(:,13), solution(:,14), solution(:,15),'g','LineWidth',1.5);
plot3(solution(:,19), solution(:,20), solution(:,21),'m','LineWidth',1.5);
plot3(solution(:,25), solution(:,26), solution(:,27),'y','LineWidth',1.5);
plot3(solution(:,31), solution(:,32), solution(:,33),0,0.45,0.74,'LineWidth',1.5);
plot3(solution(:,37), solution(:,38), solution(:,39),0.93, 0.69, 0.13,'LineWidth',1.5);
plot3(solution(:,43), solution(:,44), solution(:,45),0.00, 0.50, 0.00,'LineWidth',1.5);
plot3(solution(:,49), solution(:,50), solution(:,51),0.75, 0.75, 0.00,'LineWidth',1.5);
plot3(solution(:,55), solution(:,56), solution(:,57),0.49, 0.18, 0.56,'LineWidth',1.5);

% Initial positions markers
plot3(solution(1,1), solution(1,2), solution(1,3),'db','MarkerFaceColor','r');
plot3(solution(1,7), solution(1,8), solution(1,9),'db','MarkerFaceColor','b');
plot3(solution(1,13), solution(1,14), solution(1,15),'db','MarkerFaceColor','g');
plot3(solution(1,19), solution(1,20), solution(1,21),'db','MarkerFaceColor','m');
plot3(solution(1,25), solution(1,26), solution(1,27),'db','MarkerFaceColor','y');
plot3(solution(1,31), solution(1,32), solution(1,33),'db','MarkerFaceColor',[0,0.45,0.74]);
```

```
plot3(solution(1,37), solution(1,38), solution(1,39),'db','MarkerFaceColor',[0.93, 0.69, 0.13]);
plot3(solution(1,43), solution(1,44), solution(1,45),'db','MarkerFaceColor',[0.00, 0.50, 0.00]);
plot3(solution(1,49), solution(1,50), solution(1,51),'db','MarkerFaceColor',[0.75, 0.75, 0.00]);
plot3(solution(1,55), solution(1,56), solution(1,57),'db','MarkerFaceColor',[0.49, 0.18, 0.56]);

% Plot formatting
axis equal;
axis auto; % automatically fits axis to data

% Positions extraction
x_positions = solution(:, 1:6:end);
y_positions = solution(:, 2:6:end);
z_positions = solution(:, 3:6:end);

padding_factor = 0.1; % 10% padding

% Axis min-max calculations
x_min = min(x_positions(:)); x_max = max(x_positions(:));
y_min = min(y_positions(:)); y_max = max(y_positions(:));
z_min = min(z_positions(:)); z_max = max(z_positions(:));

% Range calculations
x_range = x_max - x_min;
y_range = y_max - y_min;
z_range = z_max - z_min;

% Use largest range for equal axes
max_range = max([x_range, y_range, z_range]);

% Calculate midpoints for each axis
x_mid = (x_max + x_min)/2;
y_mid = (y_max + y_min)/2;
z_mid = (z_max + z_min)/2;

half_range_with_padding = (max_range / 2) * (1 + padding_factor);

% Cube-like axis limits around midpoints
xlim([x_mid - half_range_with_padding, x_mid + half_range_with_padding]);
ylim([y_mid - half_range_with_padding, y_mid + half_range_with_padding]);
zlim([z_mid - half_range_with_padding, z_mid + half_range_with_padding]);

xlabel('X Position (m)');
ylabel('Y Position (m)');
zlabel('Z Position (m)');
title('MATLAB 10-Body Trajectories');
legend('Body 1','Body 2','Body 3','Body 4','Body 5','Body 6','Body 7','Body 8','Body 9','Body 10');

view([45, 25]); % set 3D view angle

% Extract final positions and velocities
finalState = solution(end, :);

fprintf('Final positions and velocities (MATLAB):\n');
for i = 1:numBodies
    idx = (i-1)*6;
    fprintf('Body %d:\n', i);
```

```
    fprintf('  Final Position [m]: x = %.4e, y = %.4e, z = %.4e\n', ...
        finalState(idx+1), finalState(idx+2), finalState(idx+3));
    fprintf('  Final Velocity [m/s]: vx = %.4e, vy = %.4e, vz = %.4e\n\n', ...
        finalState(idx+4), finalState(idx+5), finalState(idx+6));
end

%% Function defining the system of ODEs
function dydt = nBodyEquations(~, y, masses, G, numBodies)
    % Initialize derivatives vector
    dydt = zeros(6*numBodies,1);

    for i = 1:numBodies
        force = [0;0;0]; % Ensure column vector format
        idx_pos_i = (i-1)*6 + (1:3);
        idx_vel_i = (i-1)*6 + (4:6);

        position_i = y(idx_pos_i);

        % Compute gravitational forces from other bodies
        for j = 1:numBodies
            if i ~= j
                idx_pos_j = (j-1)*6 + (1:3);
                position_j = y(idx_pos_j);

                r_vec = position_j - position_i;
                r_mag = norm(r_vec);

                force = force + (G * masses(i) * masses(j)/r_mag^3) * r_vec;
            end
        end

        % Derivatives for position (velocity components)
        dydt(idx_pos_i) = y(idx_vel_i);

        % Derivatives for velocity (acceleration components)
        dydt(idx_vel_i) = force / masses(i);
    end
end
```

4.1 rebound10body.py (REBOUND)

```
import rebound
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random

# Constants and Parameters
G = 6.6742867e-11          # Gravitational constant [N·m²/kg²]
num_bodies = 10             # Number of bodies

# Load CSV
body_data = pd.read_csv("bodies.csv")

# Optional: use only the number of bodies you want
```

```
body_data = body_data.iloc[:num_bodies]

# Setup simulation
sim = rebound.Simulation()
sim.integrator = "ias15"
sim.dt = 1e-6 # small time-step
sim.units = ('m', 's', 'kg')  # units
sim.G = G
sim.collision = "direct"
sim.collision_resolve = "merge"

# Adding bodies with initial conditions
for index, row in body_data.iterrows():
    sim.add(
        m=row['mass'],
        x=row['x'], y=row['y'], z=row['z'],
        vx=row['vx'], vy=row['vy'], vz=row['vz']
    )

# Simulation timespan (s)
N_outputs = 500
times = np.linspace(0, 2e10, N_outputs)

# Arrays to store positions for plotting
positions = np.zeros((num_bodies, N_outputs, 3))

energy_vals = []
momentum_vals = []
linear_momentum_vals = []

# Integrate simulation
for i, t in enumerate(times):
    sim.integrate(t)

    # Store energy and angular momentum
    energy_vals.append(sim.energy())
    Lx, Ly, Lz = sim.angular_momentum()
    momentum_vals.append(np.sqrt(Lx**2 + Ly**2 + Lz**2))

    # Compute total linear momentum
    px = py = pz = 0.0
    for p in sim.particles:
        px += p.m * p.vx
        py += p.m * p.vy
        pz += p.m * p.vz
    total_p = (px**2 + py**2 + pz**2)**0.5
    linear_momentum_vals.append(total_p)

    for j, particle in enumerate(sim.particles):
        positions[j, i, :] = [particle.x, particle.y, particle.z]

# Plot 3D trajectories
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

# Generate random RGB colors for each body
```

```
colors = [(random.random(), random.random(), random.random()) for _ in range(num_bodies)]

for i in range(num_bodies):
    ax.plot3D(positions[i,:,0], positions[i,:,1], positions[i,:,2], color=colors[i])
    ax.scatter(positions[i,0,0], positions[i,0,1], positions[i,0,2], color=colors[i], marker='o',
s=100)

print("\nFinal positions and velocities (REBOUND):")
for i, particle in enumerate(sim.particles):
    print(f"Body {i+1}:")
    print(f"  Final Position [m]: x = {particle.x:.4e}, y = {particle.y:.4e}, z =
{particle.z:.4e}")
    print(f"  Final Velocity [m/s]: vx = {particle.vx:.4e}, vy = {particle.vy:.4e}, vz =
{particle.vz:.4e}\n")

initial_energy = energy_vals[0]
final_energy = energy_vals[-1]
delta_energy = final_energy - initial_energy

print(f"Initial Energy: {initial_energy:.4e} J")
print(f"Final Energy:   {final_energy:.4e} J")
print(f"Δ Energy:       {delta_energy:.4e} J")
rel_change = abs(delta_energy / initial_energy)
sig_fig_format = f"{rel_change:.1e}"
print(f"Relative Change: {sig_fig_format} ({float(sig_fig_format)*100:.3e}%)")

# Formatting plot
ax.set_xlabel('X Position (m)')
ax.set_ylabel('Y Position (m)')
ax.set_zlabel('Z Position (m)')
ax.set_title(f'REBOUND {num_bodies}-Body Trajectories')
ax.view_init(elev=25, azim=-45)
plt.tight_layout()
plt.show()

# Plot Energy, Angular Momentum, and Linear Momentum
fig, ax = plt.subplots(3, 1, figsize=(10, 9), sharex=True)

# Energy plot
ax[0].plot(times, energy_vals)
ax[0].set_ylabel("Energy [J]")
ax[0].set_title("Total Energy over Time")

# Angular momentum plot
ax[1].plot(times, momentum_vals, color="orange")
ax[1].set_ylabel("Angular Momentum [kg·m²/s]")
ax[1].set_title("Total Angular Momentum over Time")

# Linear momentum plot
ax[2].plot(times, linear_momentum_vals, color="green")
ax[2].set_ylabel("Linear Momentum [kg·m/s]")
ax[2].set_xlabel("Time [s]")
ax[2].set_title("Total Linear Momentum over Time")

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

4.2 bodies.csv (REBOUND)

```
mass,x,y,z,vx,vy,vz
2e20,-5e9,0,0,-0.4,0.7,2
2.5e20,5e9,0,7e8,2.0,0.0,0
1e20,0,1e10,2e9,0.0,1.7,-1
3e20,0,3e9,0,0.0,-1.2,0.2
1.5e20,1e9,0,4e9,-0.5,0.0,0.2
1.8e20,-3e9,4e9,0,0.3,0.6,0
2.2e20,2e9,-2e9,3e9,-0.7,0.9,-1.5
1.2e20,-4e9,-1e9,-1e9,1.0,-1.0,0
2.7e20,0,0,-5e9,0.0,0.0,0.8
1.4e20,3e9,5e9,-3e9,-0.4,1.3,-0.9
```

D. 100, 101-Body Problem Codes & Final Characteristics (REBOUND ONLY)

1. 100 Body Galaxy Formation Simulation

1.1 100body_galaxy_simulation.py

```
import rebound
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random

# Constants and Parameters
G = 6.6742867e-11          # Gravitational constant [N·m²/kg²]
num_bodies = 100             # Number of bodies

# Simulation timespan (s)
N_outputs = 500
times = np.linspace(0, 1e15, N_outputs)

# Load CSV
body_data = pd.read_csv("bodies.csv")

# Optional: use only the number of bodies you want
body_data = body_data.iloc[:num_bodies]

# Setup simulation
sim = rebound.Simulation()
sim.integrator = "ias15"
sim.dt = 1e-6 # small time-step
sim.units = ('m', 's', 'kg')  # units
sim.G = G
sim.collision = "direct"
sim.collision_resolve = "merge"

# Adding bodies with initial conditions
for index, row in body_data.iterrows():
    sim.add(
        m=row['mass'],
        x=row['x'], y=row['y'], z=row['z'],
        vx=row['vx'], vy=row['vy'], vz=row['vz']
```

```
    )

# Arrays to store positions for plotting
positions = np.zeros((num_bodies, N_outputs, 3))

energy_vals = []
momentum_vals = []
linear_momentum_vals = []

# Integrate simulation
for i, t in enumerate(times):
    sim.integrate(t)

    # Store energy and angular momentum
    energy_vals.append(sim.energy())
    Lx, Ly, Lz = sim.angular_momentum()
    momentum_vals.append(np.sqrt(Lx**2 + Ly**2 + Lz**2))

    # Compute total linear momentum
    px = py = pz = 0.0
    for p in sim.particles:
        px += p.m * p.vx
        py += p.m * p.vy
        pz += p.m * p.vz
    total_p = (px**2 + py**2 + pz**2)**0.5
    linear_momentum_vals.append(total_p)

    for j, particle in enumerate(sim.particles):
        positions[j, i, :] = [particle.x, particle.y, particle.z]

# Plot 3D trajectories
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

# Generate random RGB colors for each body
colors = [(random.random(), random.random(), random.random()) for _ in range(num_bodies)]

for i in range(num_bodies):
    ax.plot3D(positions[i,:,0], positions[i,:,1], positions[i,:,2], color=colors[i])
    ax.scatter(positions[i,0,0], positions[i,0,1], positions[i,0,2], color=colors[i], marker='o',
s=50)
    # ax.scatter(positions[i,-1,0], positions[i,-1,1], positions[i,-1,2], color=colors[i],
marker='x', s=50)

print("\nFinal positions and velocities (REBOUND):")
for i, particle in enumerate(sim.particles):
    print(f"Body {i+1}:")
    print(f"  Final Position [m]: x = {particle.x:.4e}, y = {particle.y:.4e}, z =
{particle.z:.4e}")
    print(f"  Final Velocity [m/s]: vx = {particle.vx:.4e}, vy = {particle.vy:.4e}, vz =
{particle.vz:.4e}\n")

initial_energy = energy_vals[0]
final_energy = energy_vals[-1]
delta_energy = final_energy - initial_energy
```

```
print(f"Initial Energy: {initial_energy:.4e} J")
print(f"Final Energy:   {final_energy:.4e} J")
print(f"Δ Energy:       {delta_energy:.4e} J")
rel_change = abs(delta_energy / initial_energy)
sig_fig_format = f"{rel_change:.1e}"
print(f"Relative Change: {sig_fig_format} ({float(sig_fig_format)*100:.3e}%)")

# Formatting plot
ax.set_xlabel('X Position (m)')
ax.set_ylabel('Y Position (m)')
ax.set_zlabel('Z Position (m)')
ax.set_title(f'REBOUND {num_bodies}-Body Trajectories')
ax.view_init(elev=25, azim=-45)
plt.tight_layout()
plt.show()

# Plot Energy, Angular Momentum, and Linear Momentum
fig, ax = plt.subplots(3, 1, figsize=(10, 9), sharex=True)

# Energy plot
ax[0].plot(times, energy_vals)
ax[0].set_ylabel("Energy [J]")
ax[0].set_title("Total Energy over Time")

# Angular momentum plot
ax[1].plot(times, momentum_vals, color="orange")
ax[1].set_ylabel("Angular Momentum [kg·m²/s]")
ax[1].set_title("Total Angular Momentum over Time")

# Linear momentum plot
ax[2].plot(times, linear_momentum_vals, color="green")
ax[2].set_ylabel("Linear Momentum [kg·m/s]")
ax[2].set_xlabel("Time [s]")
ax[2].set_title("Total Linear Momentum over Time")

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

1.2 bodies.csv

```
mass,x,y,z,vx,vy,vz
7.854705528807635e+30,-3.343429490876588e+17,-2.999537584866527e+17,-3920168746863057.5,178.2612919
827695,-198.69864731949167,0.5123869681921225
9.423823398786262e+30,-3.963723048941534e+17,7.549982226370651e+16,1898166977146147.5,-49.948367919
48856,-262.2277658985435,4.610957842195626
2.8329895332918785e+30,-2.5460763233999363e+17,1.0004466868438874e+17,-6509112554186496.0,-97.62522
999496123,-248.45030717307012,5.480410237449638
3.918345983866617e+30,-2.954281149923516e+17,2.0529250636562944e+16,-7390603188590669.0,-18.5051570
9370554,-266.3002062088936,5.343426087775348
9.1123207728085e+30,-2.4515739942758416e+17,-1.772841399293838e+17,172743816508240.88,156.423281107
59698,-216.30995768455892,1.231277619147746
7.23261434490186e+29,2.6177218552186032e+17,-1.827463766554968e+17,-5559667537368110.0,152.80397961
119962,218.88166775895616,-2.387486034499131
7.502064637469726e+30,-2.404283733793178e+17,-8.321812570347562e+16,-563758425075879.0,87.313042377
50264,-252.25913917387882,-2.236240645727801
```

```
4.079157799266707e+30,-2.2542904412494787e+17,2.8945280376980646e+17,1880341541914897.2,-210.6060659004147,-164.02233284498826,-5.729658259614186
7.564296846325697e+30,-3.5268625284436365e+17,1.1231421015765118e+17,-5721834068780440.0,-81.00068340535506,-254.35630511907382,-0.7623156331601366
9.397060884035235e+30,-1.8733199515301862e+17,-4.065646424631445e+17,-5843647796555617.0,242.44386969471702,-111.71038767014018,-1.44836939881202
2.468865020002301e+29,1.446690407157611e+17,-1.1713727728014304e+17,396391148991262.1,167.98068278165925,207.4626011554119,-0.8616259576185381
9.462723864541927e+30,4.322770461509607e+17,-1.1282723131946917e+17,-1560045009692487.8,67.41528260955265,258.28941197160236,-1.2018091357204985
6.679763631774596e+30,-2.4002659293222563e+17,-1.3866668716398374e+17,-1982290559520678.5,133.53422033127686,-231.14249428973451,3.9905202846919763
3.791559601002336e+30,2.4996712187273933e+17,-7.704516709848299e+16,-507685819584740.7,78.62722319521802,255.09998125944915,1.9224906660751622
7.399604672006321e+30,7.3470272795055e+16,-9.977194233617186e+16,4673022751783331.0,214.9507412434069,158.28587904444245,-9.862384368069435
7.688243999791485e+30,9312999101462000.0,-2.0164433129896918e+17,-2185893554806012.2,266.65814199963137,12.315679894606532,-0.23079620257082695
7.970340722090709e+30,7.295574956143723e+16,2.2490006049152346e+17,-5337700991827312.0,-253.9166616457051,82.36849886129067,-0.6333522059390875
9.724183568234544e+30,2.2527517746405306e+17,2.893039007954193e+16,-2897452509205524.5,-34.00215515301259,264.7680005418185,1.4677813019405148
8.911219297276061e+30,-1.7506542759058662e+17,3.7090918602717997e+17,-2988218314696729.5,-241.40393781768051,-113.94024518175922,7.522981359981199
9.104545572351985e+30,-1.8097994065709136e+17,1.6725059371317634e+16,-408509722604174.06,-24.56451197847879,-265.80975417997945,2.9477811028109846
5.117126257740141e+30,1.801988778223776e+17,3.7703303414063334e+17,-2482046477082527.0,-240.84801109102892,115.11071287876531,0.21840140952945017
1.3546904402394887e+30,2.6645861286731066e+17,-5.1963735514262296e+16,3707153926622179.0,51.09553179782823,262.00665506100995,-2.0014384999214836
6.165816681650635e+30,-2.139903327112675e+17,2.90750546596794e+17,1.1269950701083676e+16,-214.99068863657692,-158.2316165168403,-0.9144827091898071
2.3061612372744742e+30,2.861508605801722e+17,-1.0637269632815582e+16,4662721290235753.0,9.916371650164614,266.7581418424896,-0.08659284801221495
3.584575961093458e+30,2.7726097356240954e+17,-3.067672670746019e+17,-3529963447163484.5,198.0405656765442,178.9921087994561,-3.5498225082379844
7.01754465623818e+30,1.815910855949603e+17,335938913422799.9,68597965007510.164,-0.4938358255816474,266.9419352446134,-0.6290367905741304
3.485373402361492e+30,-4.4572044829707e+17,-2.8986791910714124e+16,934105967964448.0,17.323621392197275,-266.37967791823337,-1.9085035524836251
8.327170478270394e+30,2.4809790448254275e+17,1.8328776156601382e+17,-13976833891215.6,-158.61838559046498,214.70549228800954,-0.8080256754855026
5.953568927271674e+30,1.145626990760085e+17,2.7107919598416832e+17,2569339147512893.5,-245.8857562867386,103.91552107967576,1.8496721869636765
8.844581295781479e+30,1.0343191280516283e+17,-3.900796570791008e+17,5472192151297562.0,258.0258607275482,68.41707288221137,-1.238793765842116
9.862909235366136e+30,-5.318038962491857e+16,4.1849892054680134e+17,-6137610267457162.0,-264.8128768455183,-33.65086808333914,-4.803854643442029
3.166701081042563e+30,3.21265237581313e+17,-3.4428230351044685e+17,3743395997971546.5,195.16799619871483,182.11999869794744,2.396258391691144
7.688305309387805e+30,-1.820068150289772e+17,-6.397166637962622e+16,4324003751474879.5,88.51639300822454,-251.8394108053295,1.3716712056489353
3.9515099700346504e+30,-4.3841043778341645e+17,1.4108377564870216e+17,-7959509013925201.0,-81.77409474248783,-254.1087131426402,-0.6747989339578251
4.8236145486786 08e+30,-3.479938419648405e+17,-7.123467966569325e+16,682297008168178.2,53.53328900885196,-261.5194593789469,2.846115895847908
```

```
2.4747833448786237e+30,-1.0604096252786018e+17,-1.5675905467705664e+17,2370933136132254.5,221.10521724474071,-149.56845781486803,0.8585773298855156
8.728525959134725e+30,3.958965438147684e+17,-7.792650159409576e+16,-862259964541824.5,51.55451512545779,261.9167284389084,-5.070814130217599
7.294291855621756e+30,-4.599260407482477e+17,-8.846681559992021e+16,-3835626191471617.0,50.42209274128697,-262.1370886188198,-2.2608219462480377
3.965872299585533e+30,-2.7563326086565962e+17,-6040642556794494.0,-5151814113133270.0,5.848773379901569,-266.87831031373173,2.4001983245032683
6.795355187605461e+30,-5.3712171180787864e+16,-2.217532039065041e+17,-1543151597774438.5,259.44035799160764,-62.84060240917851,0.2481679367806988
1.1109545738796694e+30,-2.8137513903450157e+17,6.772436522964298e+16,587569278209980.6,-62.46660364750364,-259.5306611841993,-1.879569175281333
7.746082897877237e+30,-3.6804014055855386e+17,-2.2111762064245786e+17,5820643358309314.0,137.47489491386077,-228.82065888897552,-1.4441986763635402
6.98775899842679e+29,-6.426088566751458e+16,-7.898245516140755e+16,-5799587099375486.0,207.06523195544796,-168.47026556991744,0.16473684705723732
3.34877525048116e+30,1.5138112327222947e+17,-1.1480499527569742e+17,-2951330947072850.0,161.30439422180754,212.69492958389404,1.6578072490498186
9.726259420652215e+30,-3.964632222364196e+17,1.751355482932051e+17,6841240819249507.0,-107.86482484453494,-244.17907409771985,-5.008716428118688
6.768332312888978e+30,-4.07382736719356e+17,2.065512536467814e+17,1926416484466448.8,-120.71552564292168,-238.08822426721372,-1.2851479649945345
4.670969903803161e+30,1.2466715720394717e+17,-4.951209889211552e+16,9795442167979818.0,98.53100099809377,248.09248781104918,-3.9063240646401303
8.544992045267683e+30,2.339128672791295e+16,-2.703909853173060 5e+17,1853782561405186.5,265.949088359090 35,23.007022122184555,0.7760280942152411
9.413323267444235e+30,2.388970808852286e+17,2.1757522077573018e+17,-369841963366343.06,-179.74395180320516,197.35843649595776,1.412466941946921
2.5234967896691667e+30,5.072003019094564e+16,3.625900833502918e+17,1007200307624065.6,-264.36845601055023,36.98053721296843,-2.8997823250264676
9.90183533543628e+30,-1.0500796064028658e+17,-3.790765749757709e+17,2290354555738066.5,257.25465237613446,-71.26208316346204,2.1680743712155337
8.201563897966279e+30,-2.6226105840243357e+17,-1.984703364912584e+17,-6405321407293488.0,161.08572443682465,-212.8605882934306,-0.9041820417729692
1.7219566757005227e+30,3.0847810505725216e+17,1.2236739084589074e+17,-5224642700398416.0,-98.42954106064168,248.13275904755784,-0.9686097411871861
3.0348535733467094e+30,-8.243182300838242e+16,-4.357801960214183e+17,829007838167096.5,262.29108946514225,-49.61476648288014,2.469852405037995
4.521900045019489e+30,-1.072905558953244e+17,-9.74864029562379e+16,5483365931037097.0,179.5139987890573,-197.56762109395424,-3.411908809650647
3.356556909186819e+30,3.957875961248756e+17,-1.7158478913740726e+17,5653470377428662.0,106.1783064652059,244.9171449738519,-3.439156345689482
2.4368568049976553e+30,-1.7611749706279258e+17,-1.6305549970409357e+17,-369093078480329.8,181.35306990902026,-195.88084311778928,1.4216022398975938
6.284514679411738e+30,1.22049222146479e+16,-1.2317358890642776e+17,-5391661824032157.0,265.6415077127613,26.321664955942943,2.018086045487308
6.601043034340335e+30,1.465152817762293e+17,3.1499802343036755e+17,-4077580187619742.0,-242.04095877428387,112.5807041263615,-3.6220344517521195
1.426338752975458e+30,2.7573937963341526e+17,2.3906060937616048e+17,-2655072622147546.5,-174.86484594262183,201.69414051832314,-1.7411822890380342
1.824108591955361e+30,-9.413599862946622e+16,-1.5475683213345235e+17,-1214222473668472.0,228.0635618667483,-138.72725908996333,0.8076296797170258
3.056623926585341e+30,-1.9943663460256525e+17,4.212581775179763e+17,-2525870849599173.5,-241.26958057508207,-114.2244726627615,1.5981885826625908
5.386666205587308e+30,-4.405695856921652e+17,1.53965333412706e+16,-633659373607166.2,-9.323112935963849,-266.77953488066737,1.5729500993459353
```

```
8.760371663510909e+30,1.6938786162728963e+17,-9.483614402570486e+16,-3298789151737831.0,130.4069074
600558,232.9211865689045,-2.3886070708034928
2.4973702628829302e+30,3.619524788676113e+17,1.0282810289714685e+17,-5618089388698795.0,-72.9496958
0707675,256.7811958606333,-6.637815980071549
7.808720785978237e+30,2.8636104469218192e+17,-2.5432114312984013e+17,-1714449920609709.5,177.260415
75318638,199.59204812058388,0.10447324008875856
7.592580451718313e+30,-2.3179713440963834e+17,3.4208644668624224e+17,1181333726529527.5,-220.988249
99427518,-149.74122355059637,-2.7149663444659318
6.844151168213602e+30,-8.982931226498531e+16,4.486233034980664e+17,-3713864391916206.5,-261.7468025
729149,-52.41041889564965,0.9685452609605792
8.981270035039122e+30,-3.879440008211657e+16,-9.459199778659877e+16,4167250776811234.5,246.97825040
485566,-101.29158155008932,-1.180133344429352
3.988381783437102e+30,-2.5987773295284136e+16,-1.7001001178972925e+17,-5531862787358806.0,263.87728
15337312,-40.336347830832615,0.6737510389874208
7.050447153662501e+30,-1.450204063904948e+17,-9122106508960284.0,-2939886766378017.5,16.75815024075
0645,-266.41584987847546,-0.4927554807270947
7.717407769132711e+30,3.701805410778286e+17,1.2397220621308958e+17,6386015692039828.0,-84.770645260
16946,253.12482764070467,-0.9242764354161559
9.582694974975267e+29,1.6558206984268582e+17,4.070049191205708e+17,-3643671347939813.0,-247.2631106
71479,100.59420841691563,0.7530793060694113
3.7107779558709585e+30,-4.621056304079799e+16,-2.995185059232775e+17,-1162540688219957.5,263.820967
20352297,-40.703045706182635,1.8977252968297451
7.028566810369305e+30,-1.3791647210615704e+17,-4.001658120626022e+17,964028394418141.4,252.37406522
42737,-86.98029586139619,4.215854507448064
1.309506956812427e+30,-2.8031540934550323e+17,3.1215557830405286e+17,3164091997836795.0,-198.614207
6932214,-178.35536764660569,-0.938002606994017
2.360226962203492e+30,-7.691038575697579e+16,-2.3534418816392128e+17,-3932641695604995.0,253.736748
08159086,-82.92106691873443,-2.975648374925828
4.238305288588274e+30,-2.8550438573431926e+17,2.0249681386316813e+17,2210300768449714.2,-154.431645
80837416,-217.7363254922771,3.7205036310079245
1.858845912508658e+30,1.5052652857302826e+17,-6.9749351005390936e+16,241901749204079.03,112.2298014
2864797,242.20386524010758,-2.5675543735569093
7.354460302639872e+30,-3.632261597160554e+17,2.855208291063064 3e+17,532853584463627.25,-164.9687881
4846954,-209.86552742836864,-0.8771109820614679
3.976541649990319e+30,-9.232741926021453e+16,-2.4243565312925962e+17,6792172860810.162,249.46432400
785596,-95.00416682054235,2.530773968019699
8.262612207115865e+30,1.979019679792005e+17,-1.060187764359872e+17,-833659720639534.4,126.055788723
92785,235.3044385368674,-3.3619711880913252
1.9115764383510005e+30,1.1954134760943349e+17,4.552769533431982e+17,8265763017809975.0,-258.1906091
5961734,67.79269877905476,1.1107040486575726
1.720396746823735e+30,-1.1172547241827277e+17,-2.377021465317051e+17,-1114488510567670.1,241.587054
11883454,-113.55146827824485,-3.359566682658861
4.5188142857571966e+30,5.993234774737249e+16,-3.8316362397314144e+17,3743513670754217.5,263.7356748
5926185,41.25208446238419,-1.2367752415790791
9.807099306025716e+30,2.945113643722034e+17,-2.5523754174240458e+17,-3036384639335627.5,174.8265181
381701,201.72736359167055,-1.582506387960276
1.7393138001709218e+30,3.603882207661471e+17,2.165483430790486e+17,-2796538648892822.5,-137.4879464
6995594,228.8128170392482,-0.759305114788728
8.828047439511355e+30,1.2086465385833506e+17,-2.308394399536158e+17,5338247692008115.0,236.48756335
68133,123.82194088092443,0.6536995108921476
6.336522679164339e+30,-1.2101268135232616e+17,1.388810098657212e+17,2580827030371998.0,-201.2591615
8793683,-175.36530598405906,3.03287641792649
9.311589670270912e+30,2.295963613058987e+16,1.1599948520869774e+17,-5511191417535855.0,-261.8623346
2030674,51.83009138680388,-2.929316061840667
8.417191084950064e+30,2.6375022748284768e+17,-3.0860694272333786e+17,57998670777619.305,202.9277488
4469824,173.43174281193453,1.7824739043607725
```

```
6.953706996979272e+30,-1.4769883731666486e+17,-3.461463501820588e+17,-3368464290791362.0,245.525293
62192283,-104.76435871912987,1.4357057294639075
6.364777632379967e+30,-4.000252487037642e+16,4.005700036043911e+17,-2508951904946264.5,-265.6211818
595864,-26.5259950516185,0.21982230861163418
7.85445298972399e+30,-3.762164123646596e+17,-2.6192336690369085e+17,1.3380898170465838e+16,152.5227
7616857785,-219.0777109059519,0.20778399564793493
3.8150842667940305e+30,1.2580463605762776e+17,3.069646628126593e+17,2336168869910514.0,-247.0033233
940397,101.23042476578574,-1.2173080985831606
4.568116471777161e+30,-4.2101156793601075e+17,-1.2216005480311064e+17,2619591151735271.0,74.3874473
5504109,-256.36838405303206,-0.5736244595908527
1.164512376278206e+30,1.1143616000265333e+17,-3.440592029562872e+17,5979154693631864.0,253.95429713
246054,82.25238983713427,-1.4544137976739009
6.891536659679046e+30,3.091378863297511e+17,-5344072306801869.0,-841300009711077.9,4.61394865657442
8,266.90251430767086,-1.0613313038957064
9.520107087635427e+30,2.6064158051261174e+17,-1.9788002238164624e+17,-2121572207239089.2,161.415022
4919327,212.6109855578747,-7.5229987864683
2.706559047853158e+30,-7.272916716651312e+16,2.8299671246678832e+17,-8132530271916604.0,-258.540943
7332504,-66.44411997673066,1.5818355259539136
```

1.3 Galaxy Formation Final Positions and Velocities:

Body 1:
  Final Position [m]: x = -1.2450e+17, y = -3.2431e+17, z = -1.7347e+16
  Final Velocity [m/s]: vx = 1.1036e+02, vy = 8.4590e+01, vz = -1.8135e+02

Body 2:
  Final Position [m]: x = -3.9893e+17, y = -1.5454e+17, z = 3.6692e+15
  Final Velocity [m/s]: vx = 1.9730e+02, vy = -2.1277e+02, vz = -1.4282e+01

Body 3:
  Final Position [m]: x = -3.6271e+17, y = -2.1684e+17, z = 1.0169e+15
  Final Velocity [m/s]: vx = -7.5515e+01, vy = -3.4869e+02, vz = -3.8551e+00

Body 4:
  Final Position [m]: x = -2.3928e+17, y = -2.9600e+17, z = 4.6563e+15
  Final Velocity [m/s]: vx = 1.7967e+02, vy = -5.0886e+02, vz = 2.1145e+00

Body 5:
  Final Position [m]: x = -7.2103e+16, y = -2.9285e+17, z = -1.0498e+16
  Final Velocity [m/s]: vx = 9.3609e+01, vy = 1.4119e+01, vz = -6.1127e+01

Body 6:
  Final Position [m]: x = 3.6583e+17, y = -3.5613e+16, z = 2.0289e+16
  Final Velocity [m/s]: vx = 3.7437e+01, vy = 4.4588e+02, vz = 1.3437e+02

Body 7:
  Final Position [m]: x = -1.1540e+17, y = -3.8192e+17, z = 1.5570e+15
  Final Velocity [m/s]: vx = 2.9726e+02, vy = -4.9482e+02, vz = -5.5757e+01

Body 8:
  Final Position [m]: x = -4.0608e+17, y = 1.6169e+17, z = -6.1875e+15
  Final Velocity [m/s]: vx = -1.6310e+02, vy = -1.4712e+02, vz = 8.0793e+01

Body 9:
  Final Position [m]: x = -3.9173e+17, y = -1.3195e+17, z = 4.3408e+15
  Final Velocity [m/s]: vx = 6.6000e+01, vy = -4.3808e+02, vz = 1.4167e+02

Body 10:
  Final Position [m]: x = 1.0815e+17, y = -3.7984e+17, z = 3.9348e+15
  Final Velocity [m/s]: vx = 2.0216e+02, vy = 9.5679e+01, vz = 5.3053e+01

Body 11:
  Final Position [m]: x = 2.7307e+17, y = 1.1622e+17, z = 5.1134e+16
  Final Velocity [m/s]: vx = 2.2978e+02, vy = 3.4204e+02, vz = 3.6330e+01

Body 12:
  Final Position [m]: x = 3.7034e+17, y = 1.4064e+17, z = -5.1365e+15
  Final Velocity [m/s]: vx = -3.0240e+02, vy = 2.5183e+02, vz = 5.8951e+00

Body 13:
  Final Position [m]: x = -1.3241e+17, y = -3.0647e+17, z = 6.3652e+15
  Final Velocity [m/s]: vx = 6.4191e+00, vy = -3.5364e+02, vz = -2.2959e+01

Body 14:
  Final Position [m]: x = 2.6754e+17, y = 1.9228e+17, z = -1.0280e+16
  Final Velocity [m/s]: vx = -8.5037e+01, vy = 1.8705e+02, vz = 4.8754e+01

Body 15:
  Final Position [m]: x = 3.3700e+17, y = 1.0026e+17, z = 1.7075e+15
  Final Velocity [m/s]: vx = 2.4524e+02, vy = 3.1806e+02, vz = 1.4216e+01

Body 16:
  Final Position [m]: x = 2.1720e+17, y = -2.0114e+17, z = -1.3256e+16
  Final Velocity [m/s]: vx = 2.2374e+02, vy = 4.3789e+02, vz = -5.5791e+02

Body 17:
  Final Position [m]: x = -1.5123e+17, y = 3.2724e+17, z = -4.0743e+15
  Final Velocity [m/s]: vx = -2.2092e+02, vy = 8.3995e+01, vz = -2.6599e+00

Body 18:
  Final Position [m]: x = 2.2657e+17, y = 1.5894e+17, z = 1.4812e+15
  Final Velocity [m/s]: vx = 3.3925e+02, vy = 3.7797e+02, vz = -1.1298e+02

Body 19:
  Final Position [m]: x = -4.1313e+17, y = 1.4900e+17, z = 9.6336e+15
  Final Velocity [m/s]: vx = -1.0892e+02, vy = -1.7039e+02, vz = -3.4166e+01

Body 20:
  Final Position [m]: x = -1.4719e+17, y = -3.5496e+17, z = 4.3859e+15
  Final Velocity [m/s]: vx = 1.4439e+02, vy = -3.4381e+02, vz = 5.5268e+01

Body 21:
  Final Position [m]: x = -1.3480e+17, y = 3.4083e+17, z = -9.9708e+14
  Final Velocity [m/s]: vx = -5.0342e+02, vy = -1.1096e+02, vz = 7.9679e+01

Body 22:
  Final Position [m]: x = 1.8451e+17, y = 1.7487e+17, z = 7.5029e+15
  Final Velocity [m/s]: vx = -1.6082e+02, vy = 1.2459e+02, vz = 5.6258e+01

Body 23:
  Final Position [m]: x = -4.0682e+17, y = 1.4585e+17, z = 2.5596e+16
  Final Velocity [m/s]: vx = -2.7689e+02, vy = -2.2103e+02, vz = -5.6197e+01

Body 24:
  Final Position [m]: x = 2.3620e+17, y = 1.5699e+17, z = -3.1771e+14
  Final Velocity [m/s]: vx = -1.1740e+02, vy = -3.0714e+02, vz = 3.9353e+01

Body 25:
  Final Position [m]: x = 4.4192e+17, y = 3.6375e+16, z = 5.2299e+16
  Final Velocity [m/s]: vx = 2.4106e+01, vy = 3.5401e+02, vz = 1.2109e+01

Body 26:
  Final Position [m]: x = 1.8007e+17, y = 1.6554e+17, z = -1.8775e+16
  Final Velocity [m/s]: vx = -4.8532e+01, vy = 5.2068e+01, vz = -7.7960e+01

Body 27:
  Final Position [m]: x = -3.0936e+17, y = -2.8146e+17, z = -2.3998e+15
  Final Velocity [m/s]: vx = 4.2916e+02, vy = -2.4747e+02, vz = -5.4557e+00

Body 28:
  Final Position [m]: x = 4.7183e+16, y = 3.2612e+17, z = -5.2757e+14
  Final Velocity [m/s]: vx = -1.7620e+02, vy = 2.5831e+01, vz = -1.2806e+00

Body 29:
  Final Position [m]: x = -1.2565e+17, y = 3.5172e+17, z = -2.0333e+15
  Final Velocity [m/s]: vx = -7.3059e+02, vy = -5.6190e+01, vz = 8.6159e+01

Body 30:
  Final Position [m]: x = 2.8345e+17, y = -2.1113e+17, z = 3.7649e+14
  Final Velocity [m/s]: vx = 3.9827e+01, vy = 2.8085e+02, vz = -1.2155e+01

Body 31:
  Final Position [m]: x = -3.2845e+17, y = 3.3018e+17, z = 3.0148e+15
  Final Velocity [m/s]: vx = -3.8704e+02, vy = -7.6473e+01, vz = 5.2155e+00

Body 32:
  Final Position [m]: x = 3.9795e+17, y = 9.1838e+16, z = -5.7195e+16
  Final Velocity [m/s]: vx = 6.5299e+01, vy = 5.7137e+02, vz = -9.7217e+01

Body 33:
  Final Position [m]: x = -9.5934e+16, y = -3.3631e+17, z = 1.0714e+16
  Final Velocity [m/s]: vx = -2.7850e+01, vy = -1.1678e+02, vz = 3.9665e+01

Body 34:
  Final Position [m]: x = -4.3107e+17, y = -1.1967e+17, z = -9.4466e+14
  Final Velocity [m/s]: vx = 3.1021e+01, vy = 1.6628e+01, vz = -4.5943e+01

Body 35:
  Final Position [m]: x = -2.3134e+17, y = -3.0509e+17, z = 4.8506e+15
  Final Velocity [m/s]: vx = 1.0463e+01, vy = -2.7102e+02, vz = 1.8171e+01

Body 36:
  Final Position [m]: x = 2.2215e+17, y = -2.7083e+17, z = -6.5395e+15
  Final Velocity [m/s]: vx = 5.1055e+02, vy = -4.5935e+00, vz = -9.9466e+00

Body 37:
  Final Position [m]: x = 3.5238e+17, y = 1.3518e+17, z = -3.5972e+15
  Final Velocity [m/s]: vx = -5.4950e+01, vy = 1.0724e+02, vz = -1.1623e+01

Body 38:
  Final Position [m]: x = -2.9870e+17, y = -2.7602e+17, z = -1.6418e+15
  Final Velocity [m/s]: vx = 1.7523e+02, vy = -7.7792e+01, vz = 1.0673e+01

Body 39:
  Final Position [m]: x = -2.1065e+17, y = -2.9437e+17, z = -1.7118e+15

Final Velocity [m/s]: vx = 2.1403e+02, vy = -1.7332e+02, vz = -1.3322e+01

Body 40:
Final Position [m]: x = 1.9098e+17, y = -2.1647e+17, z = 6.6563e+16
Final Velocity [m/s]: vx = 7.3023e+01, vy = -1.7725e+02, vz = 3.7876e+02

Body 41:
Final Position [m]: x = -3.5525e+17, y = -1.9331e+17, z = 2.6024e+15
Final Velocity [m/s]: vx = -1.9172e+02, vy = -2.7482e+02, vz = 2.6707e+01

Body 42:
Final Position [m]: x = -1.3178e+17, y = -4.1148e+17, z = 8.8277e+15
Final Velocity [m/s]: vx = 3.2999e+02, vy = 3.2391e+01, vz = -3.6552e+01

Body 43:
Final Position [m]: x = 2.2094e+17, y = -2.5520e+17, z = 8.3802e+16
Final Velocity [m/s]: vx = 1.4259e+02, vy = -9.8784e+01, vz = 1.5486e+02

Body 44:
Final Position [m]: x = 2.9079e+17, y = 1.2852e+17, z = 6.3713e+15
Final Velocity [m/s]: vx = 5.8391e+01, vy = 3.2016e+02, vz = -1.1624e+02

Body 45:
Final Position [m]: x = -4.0581e+17, y = -1.2427e+17, z = -3.6663e+15
Final Velocity [m/s]: vx = 1.3399e+02, vy = -4.6287e+02, vz = 5.7092e+00

Body 46:
Final Position [m]: x = -3.9641e+17, y = -1.1950e+17, z = -4.4627e+15
Final Velocity [m/s]: vx = -1.7819e+02, vy = -4.4955e+01, vz = -1.1715e+02

Body 47:
Final Position [m]: x = 2.6923e+17, y = 1.4142e+17, z = -8.3842e+15
Final Velocity [m/s]: vx = 5.6735e+01, vy = 3.1360e+02, vz = -1.2521e+02

Body 48:
Final Position [m]: x = 2.2247e+17, y = -1.8087e+17, z = -1.1316e+16
Final Velocity [m/s]: vx = 3.1671e+02, vy = 1.2866e+02, vz = -2.6431e+02

Body 49:
Final Position [m]: x = 4.7019e+16, y = 3.5251e+17, z = -1.2446e+15
Final Velocity [m/s]: vx = -2.4472e+02, vy = 1.1727e+02, vz = -6.8424e-01

Body 50:
Final Position [m]: x = -1.8985e+17, y = 3.1887e+17, z = -4.7028e+15
Final Velocity [m/s]: vx = -1.2944e+02, vy = -1.1385e+02, vz = -3.9472e+00

Body 51:
Final Position [m]: x = 1.1215e+17, y = -4.0039e+17, z = -2.6758e+15
Final Velocity [m/s]: vx = 3.7120e+02, vy = 4.8212e+01, vz = -2.0341e+01

Body 52:
Final Position [m]: x = -5.9532e+16, y = -3.1826e+17, z = 2.5316e+15
Final Velocity [m/s]: vx = 7.7256e+01, vy = -2.3169e+02, vz = 2.5660e+01

Body 53:
Final Position [m]: x = 1.6039e+17, y = 3.3210e+17, z = -2.8436e+15
Final Velocity [m/s]: vx = -3.8514e+02, vy = 1.1006e+02, vz = -5.0377e+00

Body 54:

Final Position [m]: x = 1.4698e+17, y = -2.0454e+17, z = -2.4301e+16
Final Velocity [m/s]: vx = 4.0804e+02, vy = 3.9578e+02, vz = 5.4623e+00

Body 55:
Final Position [m]: x = 1.7069e+17, y = -3.4346e+17, z = -6.0712e+15
Final Velocity [m/s]: vx = 3.7251e+02, vy = -2.3794e+02, vz = -8.4393e+00

Body 56:
Final Position [m]: x = 3.7686e+17, y = 7.4480e+16, z = 3.1126e+15
Final Velocity [m/s]: vx = -1.9508e+02, vy = 1.4050e+02, vz = -4.5261e+00

Body 57:
Final Position [m]: x = -1.2359e+17, y = -3.8216e+17, z = -1.4451e+15
Final Velocity [m/s]: vx = 5.3355e+02, vy = -5.2564e+02, vz = 4.7975e+01

Body 58:
Final Position [m]: x = 2.1668e+17, y = -2.0483e+17, z = -1.1400e+16
Final Velocity [m/s]: vx = -1.6638e+01, vy = 1.7380e+02, vz = 4.2295e+02

Body 59:
Final Position [m]: x = -1.2768e+17, y = 3.5078e+17, z = -1.7933e+15
Final Velocity [m/s]: vx = 1.6452e+02, vy = 2.5866e+02, vz = -1.1682e+02

Body 60:
Final Position [m]: x = -1.8899e+16, y = 4.4368e+17, z = 1.6868e+15
Final Velocity [m/s]: vx = -2.9151e+02, vy = 1.5325e+02, vz = -9.0777e-01

Body 61:
Final Position [m]: x = 2.3744e+17, y = -2.6846e+17, z = 2.1205e+15
Final Velocity [m/s]: vx = 4.7720e+02, vy = 1.0316e+02, vz = -1.0043e+01

Body 62:
Final Position [m]: x = -4.1028e+17, y = 1.4332e+17, z = 8.6963e+15
Final Velocity [m/s]: vx = -1.1708e+02, vy = -5.2627e+02, vz = 3.4863e+02

Body 63:
Final Position [m]: x = -3.2755e+17, y = -2.3599e+17, z = -2.4762e+12
Final Velocity [m/s]: vx = 2.1070e+02, vy = -2.1267e+02, vz = 2.8654e+00

Body 64:
Final Position [m]: x = 2.2372e+17, y = 1.6061e+17, z = 3.8847e+15
Final Velocity [m/s]: vx = 1.3528e+01, vy = -2.8553e+02, vz = 1.2110e+02

Body 65:
Final Position [m]: x = 1.9615e+17, y = 2.8450e+17, z = -9.6776e+15
Final Velocity [m/s]: vx = -2.6581e+02, vy = 5.2450e+01, vz = -2.2163e+01

Body 66:
Final Position [m]: x = 3.7304e+17, y = -4.3238e+16, z = -1.1265e+16
Final Velocity [m/s]: vx = -4.4759e+00, vy = 4.5579e+02, vz = 9.5749e+01

Body 67:
Final Position [m]: x = -3.7402e+17, y = 1.5636e+17, z = -1.7661e+16
Final Velocity [m/s]: vx = -6.5479e+01, vy = -2.0929e+02, vz = -9.2144e+01

Body 68:
Final Position [m]: x = -3.3533e+17, y = 3.3030e+17, z = 3.0680e+15
Final Velocity [m/s]: vx = -1.7939e+01, vy = -3.1930e+02, vz = 1.9204e+00

Body 69:
  Final Position [m]: x = 2.1838e+17, y = -2.0195e+17, z = -1.3939e+16
  Final Velocity [m/s]: vx = 3.7834e+02, vy = -4.1682e+02, vz = 2.7645e+02

Body 70:
  Final Position [m]: x = 2.0836e+17, y = -3.0048e+17, z = -6.8859e+15
  Final Velocity [m/s]: vx = 1.4195e+02, vy = -5.5531e+01, vz = -5.7551e+01

Body 71:
  Final Position [m]: x = -1.1261e+17, y = -3.6193e+17, z = 1.2663e+16
  Final Velocity [m/s]: vx = 1.7172e+02, vy = -2.5299e+02, vz = 3.8067e+01

Body 72:
  Final Position [m]: x = 1.9301e+17, y = 3.0721e+17, z = 1.8021e+15
  Final Velocity [m/s]: vx = -2.0852e+02, vy = 1.0411e+02, vz = 4.5687e+00

Body 73:
  Final Position [m]: x = -1.2970e+17, y = 3.7663e+17, z = 3.0422e+14
  Final Velocity [m/s]: vx = -3.3892e+02, vy = -2.9577e+02, vz = 4.9184e+00

Body 74:
  Final Position [m]: x = 2.0753e+17, y = -1.8371e+17, z = 1.4715e+16
  Final Velocity [m/s]: vx = 2.4720e+02, vy = -1.6993e+02, vz = -3.3471e+01

Body 75:
  Final Position [m]: x = 1.4106e+17, y = -3.8207e+17, z = 7.8045e+15
  Final Velocity [m/s]: vx = 2.7412e+02, vy = 3.1706e+02, vz = -9.9059e+01

Body 76:
  Final Position [m]: x = -2.5398e+17, y = 1.5545e+17, z = -7.2751e+16
  Final Velocity [m/s]: vx = 1.0415e+02, vy = -1.3070e+02, vz = -9.0505e+01

Body 77:
  Final Position [m]: x = 2.4111e+17, y = -2.5112e+17, z = -6.0731e+15
  Final Velocity [m/s]: vx = 4.0291e+02, vy = 6.7669e+01, vz = -7.5709e+01

Body 78:
  Final Position [m]: x = -4.5290e+17, y = -8.6894e+16, z = 3.5131e+14
  Final Velocity [m/s]: vx = -1.1470e+02, vy = -4.4436e+02, vz = -1.2773e+01

Body 79:
  Final Position [m]: x = 2.7719e+17, y = 1.3542e+17, z = -2.7227e+15
  Final Velocity [m/s]: vx = 1.5230e+02, vy = 2.7833e+02, vz = 1.4693e+02

Body 80:
  Final Position [m]: x = -4.4624e+17, y = -6.2148e+15, z = 7.3872e+14
  Final Velocity [m/s]: vx = 6.9063e+00, vy = -3.6000e+02, vz = 5.4432e-01

Body 81:
  Final Position [m]: x = 2.1978e+17, y = -1.9132e+17, z = -2.0576e+16
  Final Velocity [m/s]: vx = 3.1894e+02, vy = -2.2384e+02, vz = -1.9207e+02

Body 82:
  Final Position [m]: x = 2.6734e+17, y = 1.6646e+17, z = 3.8548e+15
  Final Velocity [m/s]: vx = 4.8806e+01, vy = 3.1707e+02, vz = 1.0261e+02

Body 83:
  Final Position [m]: x = -1.3680e+17, y = 4.0031e+17, z = 9.1182e+13
  Final Velocity [m/s]: vx = -2.2374e+02, vy = -2.4345e+02, vz = -1.9910e+01

Body 84:
  Final Position [m]: x = 2.2391e+17, y = -2.4814e+17, z = 3.5370e+15
  Final Velocity [m/s]: vx = 3.1841e+02, vy = 4.9986e+01, vz = 1.0681e+02

Body 85:
  Final Position [m]: x = 2.4894e+17, y = -2.0307e+17, z = -1.4058e+15
  Final Velocity [m/s]: vx = -7.1697e+01, vy = 3.1044e+02, vz = -2.8166e+01

Body 86:
  Final Position [m]: x = 3.7218e+17, y = -2.6728e+16, z = 1.4020e+16
  Final Velocity [m/s]: vx = -1.4814e+01, vy = 2.2097e+02, vz = -6.7721e+01

Body 87:
  Final Position [m]: x = 1.5458e+17, y = 3.2943e+17, z = -1.7824e+14
  Final Velocity [m/s]: vx = -1.5669e+02, vy = -4.5829e+01, vz = -1.0985e+01

Body 88:
  Final Position [m]: x = 3.1935e+17, y = -6.1899e+16, z = -3.3148e+15
  Final Velocity [m/s]: vx = 2.2730e+02, vy = 2.1393e+02, vz = -1.5424e+01

Body 89:
  Final Position [m]: x = -3.2487e+17, y = -7.7014e+16, z = 1.9987e+15
  Final Velocity [m/s]: vx = -2.3255e+02, vy = -2.5858e+02, vz = -4.1507e+00

Body 90:
  Final Position [m]: x = -2.3824e+17, y = 1.5046e+17, z = -5.0142e+15
  Final Velocity [m/s]: vx = -2.6979e+02, vy = 3.4052e+01, vz = 1.0748e+00

Body 91:
  Final Position [m]: x = 3.6156e+17, y = -2.1217e+16, z = -9.0885e+15
  Final Velocity [m/s]: vx = -2.0046e+02, vy = 2.6455e+02, vz = 1.0556e+02

Body 92:
  Final Position [m]: x = 1.2621e+17, y = -3.9275e+17, z = 7.7375e+15
  Final Velocity [m/s]: vx = 1.1236e+02, vy = -5.5390e+01, vz = 8.2063e+01

Body 93:
  Final Position [m]: x = -2.8226e+17, y = 2.4521e+17, z = -2.0702e+16
  Final Velocity [m/s]: vx = -2.1840e+02, vy = -2.1909e+02, vz = 5.1310e+00

Body 94:
  Final Position [m]: x = -1.1340e+17, y = -3.7047e+17, z = -7.0946e+15
  Final Velocity [m/s]: vx = 2.9275e+02, vy = 1.0509e+02, vz = 1.8038e+02

Body 95:
  Final Position [m]: x = -1.3546e+17, y = 3.5444e+17, z = -4.8870e+14
  Final Velocity [m/s]: vx = -9.6672e+01, vy = -2.6545e+02, vz = -3.3627e+01

Body 96:
  Final Position [m]: x = -2.4580e+17, y = -3.0840e+17, z = 2.7461e+15
  Final Velocity [m/s]: vx = 3.7219e+02, vy = -6.4067e+01, vz = -1.1498e+00

Body 97:
  Final Position [m]: x = 3.5846e+17, y = -2.7090e+17, z = 4.7377e+15
  Final Velocity [m/s]: vx = 2.1962e+02, vy = 2.1192e+02, vz = -1.3548e+00

Body 98:
  Final Position [m]: x = 2.1640e+17, y = 1.5927e+17, z = -9.7018e+15

Final Velocity [m/s]: vx = -2.0653e+02, vy = 1.8615e+02, vz = -3.4747e+01

Body 99:
Final Position [m]: x = 3.6183e+17, y = -2.5710e+16, z = -5.8048e+15
Final Velocity [m/s]: vx = 1.1507e+02, vy = 8.3259e+01, vz = -6.8913e+01

Body 100:
Final Position [m]: x = -3.4950e+17, y = 2.0481e+17, z = -1.1006e+15
Final Velocity [m/s]: vx = -3.1409e+02, vy = -1.1381e+02, vz = 6.0937e+00

2. 101 Body Planet Formation Simulation

2.1 101body_planet_simulation.py

```
import rebound
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random

# Constants and Parameters
G = 6.6742867e-11          # Gravitational constant [N·m²/kg²]
num_bodies = 101             # Number of bodies

# Simulation timespan (s)
N_outputs = 500
times = np.linspace(0, 1e11, N_outputs)

# Load CSV
body_data = pd.read_csv("bodies.csv")

# Optional: use only the number of bodies you want
body_data = body_data.iloc[:num_bodies]

# Setup simulation
sim = rebound.Simulation()
sim.integrator = "ias15"
sim.dt = 1e-6 # small time-step
sim.units = ('m', 's', 'kg')  # units
sim.G = G
sim.collision = "direct"
sim.collision_resolve = "merge"

# Adding bodies with initial conditions
sim.add(m=1.989e30, x=0, y=0, z=0, vx=0, vy=0, vz=0) # massive central star
for index, row in body_data.iterrows():
    sim.add(
        m=row['mass'],
        x=row['x'], y=row['y'], z=row['z'],
        vx=row['vx'], vy=row['vy'], vz=row['vz']
    )

# Arrays to store positions for plotting
positions = np.zeros((num_bodies, N_outputs, 3))

energy_vals = []
```

```
momentum_vals = []
linear_momentum_vals = []

# Integrate simulation
for i, t in enumerate(times):
    sim.integrate(t)

    # Store energy and angular momentum
    energy_vals.append(sim.energy())
    Lx, Ly, Lz = sim.angular_momentum()
    momentum_vals.append(np.sqrt(Lx**2 + Ly**2 + Lz**2))

    # Compute total linear momentum
    px = py = pz = 0.0
    for p in sim.particles:
        px += p.m * p.vx
        py += p.m * p.vy
        pz += p.m * p.vz
    total_p = (px**2 + py**2 + pz**2)**0.5
    linear_momentum_vals.append(total_p)

    for j, particle in enumerate(sim.particles):
        positions[j, i, :] = [particle.x, particle.y, particle.z]

# Plot 3D trajectories
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

# Generate random RGB colors for each body
colors = [(random.random(), random.random(), random.random()) for _ in range(num_bodies)]

for i in range(num_bodies):
    ax.plot3D(positions[i,:,0], positions[i,:,1], positions[i,:,2], color=colors[i])
    ax.scatter(positions[i,0,0], positions[i,0,1], positions[i,0,2], color=colors[i], marker='o',
s=50)
    # ax.scatter(positions[i,-1,0], positions[i,-1,1], positions[i,-1,2], color=colors[i],
marker='x', s=50)

print("\nFinal positions and velocities (REBOUND):")
for i, particle in enumerate(sim.particles):
    print(f"Body {i+1}:")
    print(f"  Final Position [m]: x = {particle.x:.4e}, y = {particle.y:.4e}, z =
{particle.z:.4e}")
    print(f"  Final Velocity [m/s]: vx = {particle.vx:.4e}, vy = {particle.vy:.4e}, vz =
{particle.vz:.4e}\n")

initial_energy = energy_vals[0]
final_energy = energy_vals[-1]
delta_energy = final_energy - initial_energy

print(f"Initial Energy: {initial_energy:.4e} J")
print(f"Final Energy:   {final_energy:.4e} J")
print(f"Δ Energy:       {delta_energy:.4e} J")
rel_change = abs(delta_energy / initial_energy)
sig_fig_format = f"{rel_change:.1e}"
print(f"Relative Change: {sig_fig_format} ({float(sig_fig_format)*100:.3e}%)")
```

```
# Formatting plot
ax.set_xlabel('X Position (m)')
ax.set_ylabel('Y Position (m)')
ax.set_zlabel('Z Position (m)')
ax.set_title(f'REBOUND {num_bodies}-Body Trajectories')
ax.view_init(elev=25, azim=-45)
plt.tight_layout()
plt.show()

# Plot Energy, Angular Momentum, and Linear Momentum
fig, ax = plt.subplots(3, 1, figsize=(10, 9), sharex=True)

# Energy plot
ax[0].plot(times, energy_vals)
ax[0].set_ylabel("Energy [J]")
ax[0].set_title("Total Energy over Time")

# Angular momentum plot
ax[1].plot(times, momentum_vals, color="orange")
ax[1].set_ylabel("Angular Momentum [kg·m²/s]")
ax[1].set_title("Total Angular Momentum over Time")

# Linear momentum plot
ax[2].plot(times, linear_momentum_vals, color="green")
ax[2].set_ylabel("Linear Momentum [kg·m/s]")
ax[2].set_xlabel("Time [s]")
ax[2].set_title("Total Linear Momentum over Time")

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

2.2 bodies.csv

```
mass,x,y,z,vx,vy,vz
5.5427706979966595e+23,3506453208046.993,-1910761974659.002,77145289750.52805,2758.8838629732386,5062.847858733433,3.894265561765288
6.098442716505168e+23,2695210892004.7295,3240031718618.201,-20356558189.801846,-4314.706070438273,3589.1756028251716,49.1979342198361
9.237496918310586e+23,-1215706762690.7769,2443379248779.1763,34392163761.23765,-6244.237481547893,-3106.82909251926,-163.00412890884513
1.5012086835822655e+23,494337294850.11743,749898566680.8749,-6758984784.553311,-10150.367836040892,6691.178781699708,-34.0298795315993
5.0722455690240844e+23,-2555755427556.285,-108774619146.93912,68837831039.17197,306.32285924437434,-7197.32522382635,-143.55658447844277
5.343338351754706e+23,649417342187.1436,315190773114.8056,10703490299.024717,-5921.208821333089,12200.026216771808,-37.63395159902496
4.201874395595084e+23,1277328291976.9155,-1232057723269.5417,-15581344186.252197,6004.358351936957,6224.981714123039,15.573282316398155
7.742886968699744e+23,-2513247484577.797,-2078339709216.3635,3190103469.0924225,4065.891950581718,-4916.709560063945,155.07588016694245
4.097850767850947e+23,-1747770255318.2217,-2409851491671.686,-18533011297.510536,5405.782859835318,-3920.6011332155676,-64.76327306211246
7.414477456446105e+23,864063222789.0308,-425905629500.7431,-13948670379.646267,5190.059664530685,10529.421002156338,92.46720533711549
```

```
7.504775166860182e+23,1438836892141.1504,273586705535.08438,4514172500.019934,-1778.3997576078689,9352.892989505812,24.48787866800756
6.328677505227548e+23,-1496358668846.9622,3071715495674.906,2952541078.5718937,-5603.660525865318,-2729.772993937954,71.50330020972294
6.209159711958556e+23,3120213000059.8823,-1023669465580.5708,4347424435.607813,1982.0121799258613,6041.30569292139,2.5715136731661654
1.1633048120537142e+23,-249213566804.27527,-4121433208613.2124,-2094011096.2645857,5659.881589204809,-342.2399944727467,-2.080063684225706
3.7342514946734613e+23,1974026557768.8943,1934213928797.0405,12976507195.865807,-4850.556175279733,4950.39693768891,-86.85013228737328
1.809075216446877e+23,1090631847975.5431,-3455600408093.0015,-17051538558.860773,5772.036267437686,1821.7287410298268,70.1922148656983
3.203337319049133e+23,-128809868011.3992,328769381044.5871,-172102454.91910884,-18053.46870069415,-7073.240558764092,12.915687481855707
9.364493975515471e+23,-3519967329255.7163,-1505515854731.7388,31104114710.382526,2315.6737438822993,-5414.154821460429,20.960573443416997
6.87022901075785e+23,-1418321828273.679,2994735563035.2275,-13176089541.639357,-5720.377204591967,-2709.1994216039957,53.71287479375227
2.1807720980699517e+23,-3537693290330.395,555880087511.596,31560495100.123035,-945.0977213211206,-6014.7250145104845,68.78258356533264
9.275816833231578e+23,-915683725698.9122,2504369698956.494,-7520286491.057161,-6626.75206267144,-2422.9685499542334,23.408906622014587
3.39300147200648e+23,-536644877302.18585,-1704051990121.09,23121598740.948997,8222.015470592216,-2589.3003904646685,122.54693786756768
2.559673131250052e+23,-439495797617.67096,4157477306197.6147,-27714203807.944828,-5603.837707079427,-592.39364196202,67.07789730161797
9.63928025748155e+23,-1576314546624.293,-3546056809335.286,-19585579104.211136,5344.568487622554,-2375.7997983254127,1.8013340633705117
8.438678480098613e+23,-3077404796482.766,-1407289817690.0261,-48885982083.83838,2604.8007690212353,-5696.073601687624,-58.51259258433425
3.036480114628158e+23,-2161039720092.5042,2649813244032.4697,-7128897424.699079,-4828.697258134816,-3938.015856261318,53.143789368096016
2.4255742290565477e+23,-3370484223665.698,101024427493.03403,49842741228.57128,-187.98175858780604,-6271.647040027247,-91.78076434515508
9.205286275942509e+23,-2300450483980.0503,3146338474642.304,20370101741.95903,-4711.126981449711,-3444.548141248395,109.41773154059354
7.498642107846891e+23,1746658275388.265,214424231099.81058,23986803341.33038,-1058.302920880133,8620.73071285687,56.02323864915189
6.225379225445672e+22,1377563024265.668,3927618886916.9746,-72452250382.82184,-5329.247694329568,1869.1667349181546,-108.48588355694515
4.6173182768204735e+23,-1962272698654.7278,3928035901363.0835,41645881283.38547,-4918.874752739967,-2457.251888164826,-82.04383084182405
4.9257088982319656e+23,-255242364083.0477,-1718449692562.8425,-8099570644.5842705,8646.575169667683,-1284.2809987858004,-120.38118616839547
9.173035061816086e+23,2859364303338.6465,-434251453984.1956,-11605659195.303125,1017.2554951545475,6698.202213332954,-41.843089734232436
4.739472439502892e+23,-3123963515135.5186,2239377465219.05,27275005187.096508,-3423.9300172531935,-4776.4312262694075,-6.45708235220634
1.8473895862833583e+23,-3802479112509.664,406187475262.3559,17291806523.03482,-625.8219733845984,-5858.563168171263,22.686574621901553
6.639995186692614e+23,4242218502255.375,1289669617830.9507,-7818676517.5632715,-1591.5365922902097,5235.174873845092,108.82164405881171
7.413151421386195e+23,47603672824.2805,-3939166718606.135,-7404151036.394388,5804.576076451636,70.14659702568375,117.23214243152493
7.223500496463721e+23,-811501370849.2231,302111561358.2824,5987009921.259841,-4319.922659014372,-11603.737188976258,2.1551071186951756
```

```
1.2638325442815349e+23,3792215511960.735,560811981636.0796,-21056866233.342,-860.8971861079159,5821.394282335983,-12.050252704167933
7.76806582542927e+23,1939233290024.8074,-2944818669818.5137,-31561970539.71101,5124.574276833076,3374.6543163053084,-110.83268378083945
4.531762741453568e+23,-2135058121033.2903,-1598161553855.8022,-40007660945.22419,4227.853590204031,-5648.185642137732,12.285622211767697
4.5469073866722694e+23,2139400156011.2432,1835343932815.5798,-18442773487.124645,-4468.3373546544235,5208.594129272178,17.764049433387704
7.454450605576281e+23,3445984470735.0728,-1742843848024.6277,-16777806334.295824,2646.183511101818,5232.085075382624,45.0541615651858
1.668535123347896e+22,-3368524064981.3574,2785880388727.585,3222803901.866221,-3512.1241922777754,-4246.655710259839,6.011904518426931
5.121684391595042e+23,-1352097530415.2644,1112696223030.513,16891641029.321806,-5532.737537263546,-6723.129463129965,-8.690391908784997
9.419647814467515e+23,-524414862693.1513,1151772362564.4902,8046597093.957622,-9321.231384154624,-4244.061096906933,-69.44962476846807
5.885608003202723e+23,2048853120204.6313,440838823610.1651,18111331920.61906,-1674.1451425310222,7780.7972330117145,2.307211055057165
1.2444771733631736e+23,-1490050451681.9844,-458484055575.0939,-9577960296.071032,2713.803529836938,-8819.726937586789,-91.2465169658429
6.175410627558458e+23,-2399762473768.9546,-72180302921.2157,29492313130.364643,223.55893041568302,-7432.614025921371,-164.69070588271737
2.0009769472714615e+23,-4113826419632.9746,952726779651.7307,-25881881239.811794,-1265.0339252609115,-5462.353000482166,-40.306651712115155
4.936786393603655e+22,2223000097998.196,2022417965130.2798,22049680868.591312,-4472.565281302964,4916.151473169563,-97.80567930823793
8.002701509473704e+23,1622812966820.8813,-2076508601834.3184,-31962460031.19371,5592.17272059676,4370.340867198832,-95.30257227690174
6.185857998568405e+23,1821409286419.9084,-927245780545.8318,31954911224.972576,3656.302176869022,7182.154805801666,-74.45085510229907
4.337631160678652e+23,-1655941907939.6592,-1255470999163.3386,-9754845942.346949,4828.7918112430525,-6369.08278270226,-107.96912387540512
2.645415556648411e+23,3397634317935.198,-2017184610414.1592,-11781898287.957338,2959.0487237865896,4984.058196991437,11.675627002451685
8.370895903927054e+23,500548230829.0025,3779652526614.632,91131853975.12679,-5849.668209240825,774.6852527988198,-0.9016787945381777
3.090609934883735e+23,2832944219731.832,-2561780722226.6187,12844989882.337454,3954.201888371045,4372.752627155767,21.138053820291564
4.839663430930087e+22,-524167323712.7829,439314278348.1899,-15756014608.664484,-8949.243527719398,-10677.779576881238,173.19022449801028
1.418028598109713e+23,2114807538455.9307,1219248132232.5361,42859206123.85622,-3683.270156311253,6388.697498739484,-97.71665409403599
8.919992374494341e+23,1134730901458.992,163032056800.75952,-7232189597.263556,-1530.3780663266743,10651.692169339336,24.15770320674953
6.258624009468967e+23,1819200806114.8447,682855665910.3245,-30069632463.245373,-2904.65036279426,7738.300412917758,-86.37166667710417
6.743596010959364e+22,1253328990700.1858,1464063797039.534,14166500151.950108,-6304.796691185454,5397.295179016,20.895662328889923
2.339457778273833e+23,75508055448.51282,-2900600889743.6904,-32634768686.33055,6761.695555986498,176.01955676587644,-16.316956684993144
3.260377462399979e+23,10114950280.683367,-395447479342.74335,2165230819.94884,18313.139825887225,468.42250487960007,241.3127340770064
9.329938814688688e+23,-757886428557.7277,2559998216239.111,1127877341.995149,-6761.373670707596,-2001.701919529074,-26.206076811414395
1.4523263382865287e+23,4467082433211.427,-243099831418.19778,-28658722775.86959,296.0095636605309,5439.325540361331,14.00313524992038
```

```
2.3863416432251313e+22,2420654728529.979,-684414602907.497,-31962107631.376373,1976.4691745762814,6990.425734499497,-156.23007623798875
2.092747889431005e+23,-1146385328314.4126,112529715743.88174,3043792831.442931,-1048.740632129676,-10683.94126771808,12.185145160153107
4.7104294414152436e+23,-919901815944.7694,-2232874350013.1797,-27320875691.523254,6855.266612430197,-2824.2396198974934,19.416085850502295
9.322222107117287e+23,-459834694917.5231,2741667476652.402,34138332513.68849,-6815.170945468226,-1143.0460036483373,-25.006510745795893
9.207622699085732e+23,1695365625322.82,-2863864827671.615,21250163172.43063,5434.829950942605,3217.338957228021,-66.82035283419586
2.656623117337504e+23,-834492639030.8278,1534982358253.941,1685596575.4959002,-7658.19956853968,-4163.377600928865,13.060028753426069
3.856543759123538e+23,-879861258184.105,-2070744332697.259,7692587154.967091,7069.622991307059,-3003.8896071329273,102.19701736604985
2.5840494693158018e+23,-1805883353682.5747,750959335779.3049,18755492548.532017,-3163.3695417438785,-7607.171420345203,-212.79780745522092
9.899511919276274e+23,1737932338726.1675,-3553357470455.0264,-19173621831.80597,5204.0430022085275,2545.275757043843,-6.3395574160986525
5.0075005993195044e+23,-3052385991714.6904,353180798116.2392,9179625022.234406,-755.4858694485752,-6529.331427820885,-6.8230070548518755
2.8575432520541096e+23,-28425378058.908985,-1429575131357.711,2360683233.7440333,9633.586644844165,-191.55225663647303,57.73790881660828
2.565685350992863e+22,-3910003754136.7993,915387801069.9373,-8756618145.167652,-1310.626925585971,-5598.235188762793,-95.28059521974359
6.63515261792011e+23,-1171592815138.3242,4097079308845.8833,39719901094.05051,-5366.378039241864,-1534.559006587284,-24.053462393320636
8.413839769845166e+23,-1787024438208.426,-3489772741999.4473,-29450713206.884712,5179.290258843549,-2652.1836662135665,-61.850357312338346
8.536074079425558e+23,-2010912904985.4443,3975275338800.16,-33443727308.045193,-4871.056094900272,-2464.048079019207,18.418360270705932
7.714499873890978e+23,1516120373489.6672,-3888490545080.975,13589742024.067585,5254.534717122366,2048.7402619290892,63.39833538457608
1.673761928200324e+23,-3079306407821.9185,-2257181299477.6943,30793282025.28048,3486.066640987063,-4755.784282002366,39.63804936049871
3.378748862119759e+23,471361450331.154,-923388111060.8502,12136486380.07532,10078.607664045581,5144.821629103849,-24.517694204687178
3.3698693811864705e+23,-1137902948822.9517,727579818113.1344,-10973411055.592464,-5340.689875959497,-8352.604906446468,236.41934848316782
7.58044259192524e+23,648300109544.8904,1335699164548.849,22368541424.61076,-8506.742355249844,4128.8653516837185,38.36219425779492
1.0579887746488259e+23,-1632295649527.5942,-93648697230.0252,-32077566152.070194,516.1231982143895,-8996.021044439141,18.03905733250755
6.92738022216169e+23,417512980987.0734,1561796208391.9014,-24805467251.308514,-8754.365758902448,2340.293391039761,8.30537891085404
6.374704371076046e+23,3928513387676.635,-289519501577.91296,-14210557136.798727,426.6693414468001,5789.510588577225,22.914430192944987
5.243474466481589e+22,-848744085478.0242,-833610754584.3085,-624299793.8692383,7402.071932604561,-7536.448802431594,84.48111366137911
3.421664902424112e+23,3357640040515.004,1481249536044.6335,-59688839908.879776,-2427.5872862705714,5502.762415030132,44.9712800447486
2.4485187959585803e+23,258668447107.2883,-1689784387908.641,10108847654.574532,8710.841382930143,1333.436283139731,-47.29582683301328
7.756253465255933e+23,-1301443420091.9014,-2737930905113.1323,-26657057145.378807,5976.616275273395,-2840.9146159761576,-35.69921975584328
9.438742325463212e+23,2045452254087.3528,-3308851852604.364,-38605387167.562744,4968.97994645779,3071.7033233135257,-52.87386547551059
```

```
4.402750693024406e+23,560623625492.2523,-1770928804812.1365,37660807466.24908,8059.553720882636,255
1.4160787113706,-67.46697821505052
3.3993621846178e+23,-3754692540736.05,-1371850931436.352,14835222225.304178,1977.6566408757064,-541
2.754744320877,44.336354785591666
1.1007614259469398e+23,963840653277.7816,3826308915149.3564,-3587004143.895379,-5624.599330482808,1
416.826924677803,-10.329813181827976
6.0381514417686115e+23,-3353151619038.3594,-2670669837674.0835,53067166912.62168,3466.9724431809445
,-4352.947001168123,59.72668098893764
7.66278591054588e+23,909350766514.4365,1742715318137.2585,-9646438039.764187,-7285.69403929066,3801
.6831494315297,35.66993859690378
9.565315802000941e+23,-2034940721883.856,-1594028072709.519,5074027621.1321,4419.170825554112,-5641
.525907756028,79.33504070657573
```

2.3 Planet Formation Final Positions and Velocities:

Body 1:
  Final Position [m]: x = -1.7802e+08, y = 5.1957e+08, z = -1.0543e+06
  Final Velocity [m/s]: vx = 9.3655e-03, vy = -1.8237e-02, vz = -3.3397e-04

Body 2:
  Final Position [m]: x = -7.9553e+10, y = 3.9125e+12, z = -4.9230e+10
  Final Velocity [m/s]: vx = -5.8368e+03, vy = -1.0761e+02, vz = -7.9520e+01

Body 3:
  Final Position [m]: x = 4.0470e+12, y = 1.4050e+12, z = -2.8365e+10
  Final Velocity [m/s]: vx = -1.7714e+03, vy = 5.2792e+03, vz = 4.7981e+01

Body 4:
  Final Position [m]: x = 2.2763e+12, y = -1.5103e+12, z = 1.0665e+10
  Final Velocity [m/s]: vx = 3.8430e+03, vy = 5.8156e+03, vz = 1.8672e+02

Body 5:
  Final Position [m]: x = -8.0199e+11, y = -3.9853e+11, z = 5.5292e+09
  Final Velocity [m/s]: vx = 5.4331e+03, vy = -1.0914e+04, vz = 7.4170e+01

Body 6:
  Final Position [m]: x = -1.8377e+12, y = -1.8343e+12, z = 5.8873e+09
  Final Velocity [m/s]: vx = 5.0565e+03, vy = -5.0365e+03, vz = -2.3071e+02

Body 7:
  Final Position [m]: x = 6.9788e+11, y = -1.8419e+11, z = 9.4928e+09
  Final Velocity [m/s]: vx = 3.4677e+03, vy = 1.3106e+04, vz = 1.0097e+02

Body 8:
  Final Position [m]: x = 1.7304e+12, y = 4.2484e+11, z = 4.2120e+09
  Final Velocity [m/s]: vx = -2.1380e+03, vy = 8.3591e+03, vz = 8.4572e+01

Body 9:
  Final Position [m]: x = -6.9316e+11, y = -3.1943e+12, z = 5.5567e+10
  Final Velocity [m/s]: vx = 6.2212e+03, vy = -1.3490e+03, vz = 1.1216e+02

Body 10:
  Final Position [m]: x = -2.0943e+12, y = 2.1026e+12, z = 3.0566e+10
  Final Velocity [m/s]: vx = -4.7339e+03, vy = -4.7401e+03, vz = -3.5567e+01

Body 11:
  Final Position [m]: x = 2.3069e+11, y = -9.3480e+11, z = -1.4874e+10

Final Velocity [m/s]: vx = 1.1394e+04, vy = 2.8233e+03, vz = -6.5585e+01

Body 12:
Final Position [m]: x = 5.2966e+11, y = -1.3612e+12, z = -1.2674e+10
Final Velocity [m/s]: vx = 8.8666e+03, vy = 3.5080e+03, vz = 1.1638e+01

Body 13:
Final Position [m]: x = -1.3297e+12, y = 3.1433e+12, z = 1.9112e+09
Final Velocity [m/s]: vx = -5.7494e+03, vy = -2.4377e+03, vz = 7.0901e+01

Body 14:
Final Position [m]: x = -2.3993e+11, y = -3.3029e+12, z = 2.4975e+09
Final Velocity [m/s]: vx = 6.3158e+03, vy = -4.4475e+02, vz = 1.1572e+01

Body 15:
Final Position [m]: x = -2.3826e+12, y = 3.4699e+12, z = -1.2553e+10
Final Velocity [m/s]: vx = -4.6511e+03, vy = -3.1580e+03, vz = -5.3654e+01

Body 16:
Final Position [m]: x = -1.4018e+11, y = -2.7890e+12, z = 1.6501e+10
Final Velocity [m/s]: vx = 6.8782e+03, vy = -3.5359e+02, vz = 7.4843e+01

Body 17:
Final Position [m]: x = -2.3740e+12, y = 2.7448e+12, z = -7.1244e+09
Final Velocity [m/s]: vx = -4.5692e+03, vy = -3.9611e+03, vz = -7.1936e+01

Body 18:
Final Position [m]: x = -2.2639e+10, y = 3.5291e+11, z = -1.8796e+08
Final Velocity [m/s]: vx = -1.9350e+04, vy = -1.2335e+03, vz = 8.3987e+00

Body 19:
Final Position [m]: x = -3.6768e+12, y = 1.4359e+12, z = 3.8668e+10
Final Velocity [m/s]: vx = -2.1378e+03, vy = -5.3880e+03, vz = 1.9019e+01

Body 20:
Final Position [m]: x = 2.3275e+11, y = -3.2735e+12, z = 1.9262e+10
Final Velocity [m/s]: vx = 6.3460e+03, vy = 4.4908e+02, vz = -4.8762e+01

Body 21:
Final Position [m]: x = -3.5139e+12, y = -6.6816e+11, z = 4.6050e+10
Final Velocity [m/s]: vx = 1.1422e+03, vy = -5.9854e+03, vz = 4.3484e+01

Body 22:
Final Position [m]: x = -4.5600e+11, y = -2.6044e+12, z = 1.0572e+10
Final Velocity [m/s]: vx = 6.9679e+03, vy = -1.1939e+03, vz = -2.9526e+01

Body 23:
Final Position [m]: x = -1.7432e+12, y = 3.7704e+11, z = -1.9277e+10
Final Velocity [m/s]: vx = -1.8071e+03, vy = -8.4442e+03, vz = 1.3409e+02

Body 24:
Final Position [m]: x = -1.5203e+11, y = -4.1765e+12, z = 2.8446e+10
Final Velocity [m/s]: vx = 5.6269e+03, vy = -1.9801e+02, vz = -6.6087e+01

Body 25:
Final Position [m]: x = 3.6947e+12, y = -1.1096e+12, z = 8.3026e+09
Final Velocity [m/s]: vx = 1.6715e+03, vy = 5.6282e+03, vz = 1.7356e+01

Body 26:

Final Position [m]: x = 3.2276e+12, y = 9.8750e+11, z = 4.6190e+10
Final Velocity [m/s]: vx = -1.8462e+03, vy = 5.9929e+03, vz = 6.3125e+01

Body 27:
Final Position [m]: x = -2.6199e+12, y = 2.1726e+12, z = -1.1738e+09
Final Velocity [m/s]: vx = -3.9896e+03, vy = -4.8206e+03, vz = 5.5151e+01

Body 28:
Final Position [m]: x = 3.3453e+12, y = -5.1185e+11, z = -5.1297e+10
Final Velocity [m/s]: vx = 9.6541e+02, vy = 6.1932e+03, vz = 8.6173e+01

Body 29:
Final Position [m]: x = 1.7308e+12, y = 3.5047e+12, z = -4.6698e+10
Final Velocity [m/s]: vx = -5.2222e+03, vy = 2.5571e+03, vz = 9.9243e+01

Body 30:
Final Position [m]: x = -1.5670e+12, y = 8.1376e+11, z = -1.6629e+10
Final Velocity [m/s]: vx = -4.0164e+03, vy = -7.6777e+03, vz = -5.8108e+01

Body 31:
Final Position [m]: x = -8.9211e+11, y = -4.0761e+12, z = 8.8003e+10
Final Velocity [m/s]: vx = 5.5175e+03, vy = -1.2009e+03, vz = 8.6737e+01

Body 32:
Final Position [m]: x = -1.5605e+12, y = 4.0886e+12, z = 4.5221e+10
Final Velocity [m/s]: vx = -5.1522e+03, vy = -1.9575e+03, vz = -7.9918e+01

Body 33:
Final Position [m]: x = 1.6946e+12, y = 4.2466e+11, z = -9.9265e+09
Final Velocity [m/s]: vx = -2.1284e+03, vy = 8.4249e+03, vz = 8.4420e+01

Body 34:
Final Position [m]: x = -1.7713e+11, y = 2.8873e+12, z = -1.4041e+10
Final Velocity [m/s]: vx = -6.7621e+03, vy = -4.1170e+02, vz = 3.8581e+01

Body 35:
Final Position [m]: x = 2.3429e+12, y = -3.0085e+12, z = -3.1915e+10
Final Velocity [m/s]: vx = 4.6505e+03, vy = 3.6228e+03, vz = -5.9122e+00

Body 36:
Final Position [m]: x = -3.3455e+12, y = 2.1196e+12, z = 2.5057e+10
Final Velocity [m/s]: vx = -3.1409e+03, vy = -4.8444e+03, vz = 4.5039e+01

Body 37:
Final Position [m]: x = -1.8680e+12, y = -4.0215e+12, z = -6.1560e+10
Final Velocity [m/s]: vx = 4.9646e+03, vy = -2.2979e+03, vz = -7.8294e+01

Body 38:
Final Position [m]: x = 2.6236e+12, y = -2.9903e+12, z = 5.3758e+10
Final Velocity [m/s]: vx = 4.3361e+03, vy = 3.8415e+03, vz = 3.6814e+01

Body 39:
Final Position [m]: x = 8.6577e+11, y = 1.6545e+10, z = -5.5750e+09
Final Velocity [m/s]: vx = -2.2942e+02, vy = 1.2377e+04, vz = 3.0618e+01

Body 40:
Final Position [m]: x = -3.7612e+12, y = 7.8720e+11, z = 1.7205e+10
Final Velocity [m/s]: vx = -1.1973e+03, vy = -5.7442e+03, vz = 2.5648e+01

Body 41:
  Final Position [m]: x = -3.5040e+12, y = -4.2648e+11, z = 7.1388e+10
  Final Velocity [m/s]: vx = 7.4375e+02, vy = -6.0848e+03, vz = -5.3907e+00

Body 42:
  Final Position [m]: x = -1.6237e+12, y = -2.1180e+12, z = -3.3930e+10
  Final Velocity [m/s]: vx = 5.5904e+03, vy = -4.3139e+03, vz = 5.7131e+01

Body 43:
  Final Position [m]: x = 1.6262e+12, y = -2.2992e+12, z = -9.2321e+09
  Final Velocity [m/s]: vx = 5.6215e+03, vy = 3.9522e+03, vz = -4.9836e+01

Body 44:
  Final Position [m]: x = -1.3726e+12, y = 3.5805e+12, z = 9.0539e+08
  Final Velocity [m/s]: vx = -5.4916e+03, vy = -2.1240e+03, vz = -1.9209e+01

Body 45:
  Final Position [m]: x = -4.2091e+12, y = 1.1946e+12, z = 5.5900e+09
  Final Velocity [m/s]: vx = -1.5044e+03, vy = -5.2965e+03, vz = 4.3940e+00

Body 46:
  Final Position [m]: x = 1.6152e+12, y = 6.3600e+11, z = -4.3875e+08
  Final Velocity [m/s]: vx = -3.1152e+03, vy = 8.1600e+03, vz = 9.5489e+01

Body 47:
  Final Position [m]: x = 1.0047e+12, y = 7.7045e+11, z = 1.0327e+10
  Final Velocity [m/s]: vx = -6.2256e+03, vy = 8.1289e+03, vz = 4.5011e+01

Body 48:
  Final Position [m]: x = -9.4810e+10, y = -2.0836e+12, z = -2.8523e+09
  Final Velocity [m/s]: vx = 7.9593e+03, vy = -3.6416e+02, vz = 6.3774e+01

Body 49:
  Final Position [m]: x = -1.5660e+11, y = -1.5494e+12, z = -1.7748e+10
  Final Velocity [m/s]: vx = 9.1885e+03, vy = -9.2441e+02, vz = 2.3447e+01

Body 50:
  Final Position [m]: x = -2.1528e+12, y = -1.0865e+12, z = 1.5213e+09
  Final Velocity [m/s]: vx = 3.3461e+03, vy = -6.6228e+03, vz = -1.8928e+02

Body 51:
  Final Position [m]: x = -3.3463e+12, y = -2.5504e+12, z = -4.0815e+10
  Final Velocity [m/s]: vx = 3.4136e+03, vy = -4.4734e+03, vz = 9.4349e+00

Body 52:
  Final Position [m]: x = -1.1072e+12, y = 2.7892e+12, z = -3.6428e+10
  Final Velocity [m/s]: vx = -6.1787e+03, vy = -2.4681e+03, vz = -6.9225e+01

Body 53:
  Final Position [m]: x = -2.0235e+12, y = -1.7005e+12, z = 1.9558e+10
  Final Velocity [m/s]: vx = 4.5427e+03, vy = -5.4626e+03, vz = -1.0782e+02

Body 54:
  Final Position [m]: x = -1.0439e+12, y = -1.7505e+12, z = 1.9773e+10
  Final Velocity [m/s]: vx = 6.9452e+03, vy = -4.1270e+03, vz = 1.2444e+02

Body 55:
  Final Position [m]: x = -1.6680e+12, y = 1.2831e+12, z = 2.2888e+10
  Final Velocity [m/s]: vx = -4.8516e+03, vy = -6.2780e+03, vz = -8.2319e+01

Body 56:
  Final Position [m]: x = 2.9733e+12, y = 2.3701e+12, z = -6.2380e+09
  Final Velocity [m/s]: vx = -3.6722e+03, vy = 4.6325e+03, vz = 9.6290e+01

Body 57:
  Final Position [m]: x = 3.9699e+11, y = -3.8035e+12, z = -8.4242e+10
  Final Velocity [m/s]: vx = 5.8673e+03, vy = 6.0204e+02, vz = 4.4340e+01

Body 58:
  Final Position [m]: x = 3.8245e+12, y = 1.5397e+11, z = 7.4886e+09
  Final Velocity [m/s]: vx = -2.0204e+02, vy = 5.8780e+03, vz = 7.9581e+01

Body 59:
  Final Position [m]: x = -4.8639e+10, y = 6.8301e+11, z = -1.6984e+10
  Final Velocity [m/s]: vx = -1.3891e+04, vy = -9.9669e+02, vz = -9.9540e+01

Body 60:
  Final Position [m]: x = 2.0526e+12, y = 1.3302e+12, z = 3.7401e+10
  Final Velocity [m/s]: vx = -4.0134e+03, vy = 6.1747e+03, vz = -1.1239e+02

Body 61:
  Final Position [m]: x = 4.8650e+11, y = 1.0404e+12, z = -1.6199e+09
  Final Velocity [m/s]: vx = -9.7412e+03, vy = 4.5580e+03, vz = 7.0778e+01

Body 62:
  Final Position [m]: x = 1.4142e+12, y = 1.3957e+12, z = -1.6447e+10
  Final Velocity [m/s]: vx = -5.8049e+03, vy = 5.7439e+03, vz = 5.3641e+01

Body 63:
  Final Position [m]: x = -6.1560e+11, y = -1.8269e+12, z = -1.4502e+10
  Final Velocity [m/s]: vx = 7.8636e+03, vy = -2.6337e+03, vz = 8.4875e+00

Body 64:
  Final Position [m]: x = 1.7400e+12, y = -2.3168e+12, z = -3.0458e+10
  Final Velocity [m/s]: vx = 5.4186e+03, vy = 4.0607e+03, vz = 3.3506e+01

Body 65:
  Final Position [m]: x = -3.5656e+11, y = -1.7117e+11, z = -3.8192e+09
  Final Velocity [m/s]: vx = 7.9511e+03, vy = -1.6503e+04, vz = 1.9252e+02

Body 66:
  Final Position [m]: x = 1.5084e+12, y = 2.2170e+12, z = -1.7919e+09
  Final Velocity [m/s]: vx = -5.8031e+03, vy = 3.9757e+03, vz = -1.2231e+01

Body 67:
  Final Position [m]: x = -3.0628e+12, y = 3.2620e+12, z = 2.8808e+10
  Final Velocity [m/s]: vx = -3.9733e+03, vy = -3.7263e+03, vz = 1.3479e+01

Body 68:
  Final Position [m]: x = 1.9668e+12, y = -1.5608e+12, z = -1.3440e+10
  Final Velocity [m/s]: vx = 4.5112e+03, vy = 5.7070e+03, vz = -1.8091e+02

Body 69:
  Final Position [m]: x = 3.0425e+11, y = -1.1019e+12, z = 1.1331e+08
  Final Velocity [m/s]: vx = 1.0384e+04, vy = 2.8599e+03, vz = -3.1859e+01

Body 70:
  Final Position [m]: x = 4.6006e+11, y = -2.3538e+12, z = -1.7872e+10

Final Velocity [m/s]: vx = 7.2989e+03, vy = 1.4435e+03, vz = 6.2527e+01

Body 71:
Final Position [m]: x = 2.6006e+12, y = -9.3629e+11, z = -4.5408e+09
Final Velocity [m/s]: vx = 2.3460e+03, vy = 6.5270e+03, vz = 8.8360e+01

Body 72:
Final Position [m]: x = 3.2075e+12, y = 9.0990e+11, z = -2.8804e+10
Final Velocity [m/s]: vx = -1.7015e+03, vy = 6.0731e+03, vz = -5.2457e+01

Body 73:
Final Position [m]: x = -2.4511e+10, y = 1.7666e+12, z = 1.0160e+10
Final Velocity [m/s]: vx = -8.6742e+03, vy = -6.9747e+01, vz = 1.1215e+02

Body 74:
Final Position [m]: x = 2.2484e+12, y = 1.4513e+11, z = 2.2589e+10
Final Velocity [m/s]: vx = -4.9268e+02, vy = 7.6541e+03, vz = -7.1234e+01

Body 75:
Final Position [m]: x = 1.9812e+12, y = 3.2854e+11, z = 2.0387e+10
Final Velocity [m/s]: vx = -1.3723e+03, vy = 7.9629e+03, vz = 1.9233e+02

Body 76:
Final Position [m]: x = 5.6242e+11, y = 3.9032e+12, z = 2.6215e+10
Final Velocity [m/s]: vx = -5.7332e+03, vy = 8.1557e+02, vz = 1.9776e+00

Body 77:
Final Position [m]: x = -3.0178e+12, y = -5.8409e+11, z = 7.6230e+09
Final Velocity [m/s]: vx = 1.2476e+03, vy = -6.4511e+03, vz = -1.3163e+01

Body 78:
Final Position [m]: x = 1.0788e+12, y = 9.3672e+11, z = 3.2997e+09
Final Velocity [m/s]: vx = -6.3405e+03, vy = 7.2628e+03, vz = -4.8222e+01

Body 79:
Final Position [m]: x = 3.1480e+11, y = 4.0153e+12, z = 5.7591e+10
Final Velocity [m/s]: vx = -5.7180e+03, vy = 4.3792e+02, vz = -3.6699e+01

Body 80:
Final Position [m]: x = -3.8337e+12, y = 1.6051e+12, z = 4.1052e+09
Final Velocity [m/s]: vx = -2.2072e+03, vy = -5.2230e+03, vz = -4.0351e+01

Body 81:
Final Position [m]: x = 2.7856e+12, y = 2.7901e+12, z = 3.6330e+10
Final Velocity [m/s]: vx = -4.1103e+03, vy = 4.1115e+03, vz = 5.6400e+01

Body 82:
Final Position [m]: x = 1.1829e+12, y = -4.2984e+12, z = 3.5626e+10
Final Velocity [m/s]: vx = 5.2628e+03, vy = 1.4438e+03, vz = -1.2117e+01

Body 83:
Final Position [m]: x = -1.6923e+12, y = 3.8067e+12, z = -1.7885e+10
Final Velocity [m/s]: vx = -5.1653e+03, vy = -2.2903e+03, vz = -5.1011e+01

Body 84:
Final Position [m]: x = -9.2922e+11, y = 3.7021e+12, z = -2.8885e+10
Final Velocity [m/s]: vx = -5.7182e+03, vy = -1.4066e+03, vz = 4.4928e+01

Body 85:

Final Position [m]: x = -1.0371e+12, y = -1.7628e+10, z = -2.8013e+09
Final Velocity [m/s]: vx = 2.0733e+02, vy = -1.1310e+04, vz = 1.3243e+02

Body 86:
Final Position [m]: x = 9.8271e+11, y = 9.2730e+11, z = -2.9340e+10
Final Velocity [m/s]: vx = -6.8077e+03, vy = 7.2046e+03, vz = -1.2681e+02

Body 87:
Final Position [m]: x = 9.7315e+11, y = 1.1342e+12, z = 1.9401e+10
Final Velocity [m/s]: vx = -7.1518e+03, vy = 6.1064e+03, vz = 1.4606e+00

Body 88:
Final Position [m]: x = 8.8926e+11, y = 1.3716e+12, z = 1.2112e+10
Final Velocity [m/s]: vx = -7.5577e+03, vy = 4.9068e+03, vz = -1.7780e+02

Body 89:
Final Position [m]: x = -1.2275e+12, y = 1.0560e+12, z = -7.7253e+09
Final Velocity [m/s]: vx = -5.9000e+03, vy = -6.8637e+03, vz = 1.2974e+02

Body 90:
Final Position [m]: x = -3.9408e+12, y = -2.9025e+11, z = 3.4431e+10
Final Velocity [m/s]: vx = 4.3618e+02, vy = -5.7632e+03, vz = -2.1443e+01

Body 91:
Final Position [m]: x = 1.0687e+12, y = -5.2229e+11, z = 8.9803e+09
Final Velocity [m/s]: vx = 4.6354e+03, vy = 9.4899e+03, vz = -3.2859e+01

Body 92:
Final Position [m]: x = 2.2088e+12, y = 2.9210e+12, z = -3.4803e+10
Final Velocity [m/s]: vx = -4.7949e+03, vy = 3.6494e+03, vz = 9.2154e+01

Body 93:
Final Position [m]: x = -4.9759e+11, y = -1.6341e+12, z = 1.3035e+10
Final Velocity [m/s]: vx = 8.4497e+03, vy = -2.5408e+03, vz = -4.0529e+01

Body 94:
Final Position [m]: x = -2.4007e+12, y = 1.8665e+12, z = 2.0929e+10
Final Velocity [m/s]: vx = -4.0536e+03, vy = -5.2077e+03, vz = -4.9923e+01

Body 95:
Final Position [m]: x = 3.2848e+12, y = -2.0554e+12, z = -8.6016e+09
Final Velocity [m/s]: vx = 3.1298e+03, vy = 4.9096e+03, vz = -1.0314e+01

Body 96:
Final Position [m]: x = 9.1912e+11, y = 1.6148e+12, z = -3.7599e+10
Final Velocity [m/s]: vx = -7.3491e+03, vy = 4.1843e+03, vz = -6.9320e+01

Body 97:
Final Position [m]: x = -3.9910e+12, y = 9.0296e+10, z = 1.1337e+10
Final Velocity [m/s]: vx = -1.2940e+02, vy = -5.7707e+03, vz = 4.2296e+01

Body 98:
Final Position [m]: x = -2.9821e+12, y = -2.6305e+12, z = 3.1935e+09
Final Velocity [m/s]: vx = 3.8042e+03, vy = -4.3105e+03, vz = 4.5823e+01

Body 99:
Final Position [m]: x = -2.4983e+11, y = 4.2789e+12, z = -6.6878e+10
Final Velocity [m/s]: vx = -5.5577e+03, vy = -3.1617e+02, vz = 2.7468e+01

Body 100:
  Final Position [m]: x = 9.7632e+11, y = 1.6597e+12, z = -2.7749e+10
  Final Velocity [m/s]: vx = -7.0975e+03, vy = 4.3036e+03, vz = 3.8189e+00

Body 101:
  Final Position [m]: x = 2.2917e+12, y = 1.1897e+12, z = -5.9066e+09
  Final Velocity [m/s]: vx = -3.2953e+03, vy = 6.3457e+03, vz = -8.2048e+01