Computational Analysis of N-body Simulations in Astrophysics

April 1, 2024

Abstract

This paper explores the dynamic evolution of computational analysis using N-body simulations in astrophysics, with a focus on gravitational interactions. Our model is built on Newton's law of universal gravitation, which states that every point mass attracts every other point mass with a force proportional to the product of their masses and inversely proportional to the square of the distance between them. We simplify celestial bodies as point masses, neglecting factors such as rotational dynamics, relativistic effects, and non-gravitational forces, thereby simplifying the model while retaining the core elements necessary for simulating the orbital dynamics of N-body systems. The simulation employs numerical integration to evolve the positions and velocities of each body over time, allowing us to observe complex dynamical behaviors arising from gravitational interactions, such as orbital resonances, chaos, and the long-term stability of multi-body systems.

1 Introduction

In this simulation, we explore the dynamical evolution of a system of celestial bodies governed by Newtonian mechanics, primarily focusing on gravitational interactions. The foundation of our model is Newton's law of universal gravitation, which posits that every point mass attracts every other point mass by a force acting along the line intersecting both points. The force is proportional to the product of their masses and inversely proportional to the square of the distance between them:

$$F = G\frac{m_1 m_2}{r^2} \tag{1}$$

where F is the magnitude of the gravitational force between two masses, G is the gravitational constant, m_1 and m_2 are the masses of the objects, and r is the distance between their centers.

For the purpose of this simulation, we approximate the celestial bodies as point masses, neglecting factors such as rotational dynamics, relativistic effects, and non-gravitational forces, which allows for a simplified yet insightful exploration of their orbital dynamics. This approach, while reducing complexity, retains the core elements necessary to model the trajectories and interactions within an N-body system, such as a solar system or a cluster of stars.

The simulation employs numerical integration to evolve the positions and velocities of each body over time. Given the initial conditions—positions, velocities, and masses of all bodies—the simulation iterates over discrete time steps, calculating the net gravitational force on each body from every other body and updating their velocities and positions accordingly. This method, although computationally intensive, enables the examination of complex dynamical behaviors that arise from gravitational interactions, such as orbital resonances, chaos, and the long-term stability of multi-body systems.

In essence, our simulation is a tool to investigate the celestial mechanics of an N-body system under the influence of gravity, providing insights into the structural and temporal evolution of such systems. By abstracting away from the specifics of the implementation, we focus on the physics at play and the emergent behaviors that characterize these fascinating dynamical systems.

2 Program Structure

This project consists of three files: `unit.py`, `particle3D.py`, and `solar_system.txt`. The `solar_system.txt` file primarily records the initial positions of various planets. Unit5.py is the main program logic for a computational modeling exercise. It imports the Particle3D class and contains functions for reading data from a file, computing separations between particles, computing forces and potential energy, computing energy deviation, and performing the main time integration loop. Additionally, it includes code for plotting trajectories, energy changes, and verifying Kepler's Third Law.

2.1 unit5.py

`unit5.py` is a Python script that performs simulations of the solar system dynamics. It includes several functions:

- `read_data()`: Reads `particle3D` objects from an input file and returns a list of `particle3D` objects.
- `compute_separations(particles)`: Computes the separations between particles and returns a 3D array.
- `compute_forces_potential(particles, separations)`: Computes forces and potential energy between particles and returns them.
- `compute_energy_deviation(energy, initial_energy)`: Computes the deviation in energy and returns it.
- `compute_distance(p1, p2)`: Computes the distance between two particles.
- `main()`: The main function of the script, which orchestrates the simulation. It reads input arguments, initializes the system, performs time integration, computes energy, and writes output files. It also plots trajectories and verifies Kepler's Third Law.

The Particle3D class defines point particles in 3D space. It includes attributes for storing the particle's label, mass, position, and velocity. The class provides methods for computing kinetic energy, linear momentum, updating position (both first and second order), updating velocity, and reading particle information from a text file.

2.2 unit5.py

`unit5.py` simulates the dynamics of the solar system using numerical integration techniques. It computes trajectories, energy, and orbital parameters such as periods and apsides of various celestial bodies. The simulation is based on physical principles and mathematical models implemented in the script.

3 Methods

3.1 Physical Model

Our model consists of a simplified solar system, focusing on primary celestial bodies, including the Sun, planets, and selected moons. The gravitational forces between these bodies are calculated using Newton's law of universal gravitation.

3.2 Numerical Integration

We employ the Verlet integration method for its simplicity and effectiveness in conserving energy within the system. The positions and velocities are updated iteratively using:

$$\vec{r}(t + \Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{2}\vec{a}(t)\Delta t^2$$

3.3 Initial Conditions

The initial positions and velocities are set based on astronomical data, with the center-of-mass velocity of the system corrected to prevent drift.

Table 1: Initial Conditions for Solar System Simulation

Body	Mass	X (AU)	Y (AU)	Z (AU)	Vx	Vy	Vz
Sun	332946.000	0.000	0.000	0.000	0.000	0.000	0.000
Mercury	0.055	0.081	-0.448	-0.044	0.022	0.006	-0.001
Venus	0.815	-0.711	-0.111	0.040	0.003	-0.020	-0.000
Earth	1.000	-0.486	-0.888	0.000	0.015	-0.008	0.000
Moon	0.012	-0.486	-0.886	0.000	0.014	-0.008	0.000
Mars	0.107	-1.479	0.764	0.052	-0.006	-0.011	-0.000
Jupiter	317.800	4.483	2.109	-0.109	-0.003	0.007	0.000
Saturn	95.160	8.521	-4.833	-0.255	0.002	0.005	-0.000
Uranus	14.540	12.950	14.776	-0.113	-0.003	0.002	0.000
Neptune	17.150	29.798	-2.495	-0.635	0.000	0.003	-0.000
Pluto	0.002	16.581	-30.527	-1.527	0.003	0.001	-0.001
1P/Halley	0.000	-19.927	27.173	-9.964	0.000	0.000	0.000

3.4 Algorithm unit5.py

4 Convergence

Detail the convergence tests conducted to ensure the accuracy and reliability of your simulation results. Explain the criteria used for determining convergence and present the findings, preferably through graphs or tables. Discuss the implications of these tests for the credibility of your simulation data. Run the Python script named unit3.py with the input file $solar_system.txt$, outputting the results to output.txt, using 20 timesteps with a timestep size of 15, 10 timesteps with a timestep size of 30, 10 timesteps with a timestep size of 40, 10 timesteps with a timestep size of 50.

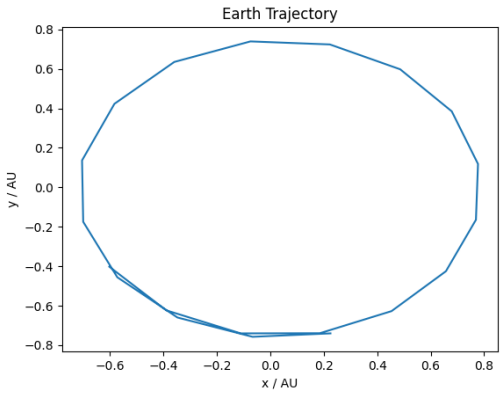

Figure 1: 20 * 15

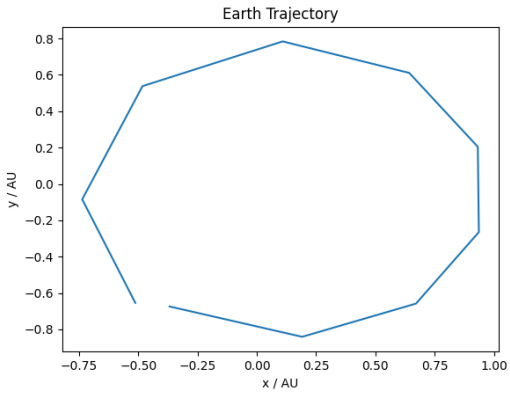

Figure 2: 10*30

Obviously, in the Earth-Moon-Sun-Mercury system, when viewed over approximately one year with a timestep of 20 or more, the trajectory of the Earth no longer converges.

Algorithm 1 Main Simulation Algorithm

procedure MAIN
    Initialize simulation parameters and read input file $solor_system.txt$
    Perform main time integration loop for n_{step} timesteps
    Compute initial forces and potential energy using Newton's law of universal gravitation:

$$F = \frac{G \cdot m_1 \cdot m_2}{r^2}$$
$$U = -\frac{G \cdot m_1 \cdot m_2}{r}$$

    where G is the gravitational constant, m_1 and m_2 are the masses of the particles, and r is the separation between them.
    Initialize arrays for storing results: positions, velocities, energies
    Perform main time integration loop for n_{step} timesteps
    for $i = 0$ to $n_{\text{step}} - 1$ **do**
        Update particle positions using the Verlet integration algorithm:

$$\mathbf{r}_{\text{new}} = \mathbf{r}_{\text{old}} + \mathbf{v}_{\text{old}} \cdot dt + \frac{1}{2}\mathbf{a}_{\text{old}} \cdot dt^2$$

        Compute new separations and forces between particles
        Update particle velocities using the Verlet integration algorithm:

$$\mathbf{v}_{\text{new}} = \mathbf{v}_{\text{old}} + \frac{1}{2}(\mathbf{a}_{\text{old}} + \mathbf{a}_{\text{new}}) \cdot dt$$

        Compute kinetic energy:

$$KE = \frac{1}{2}mv^2$$

        Compute total energy: $E_{\text{total}} = KE + U$
        Write output data to file
    end for
    Compute energy deviation and apsides
    Plot trajectories and energy
    Verify Kepler's Third Law by fitting the periods and semi-major axes using linear regression
end procedure

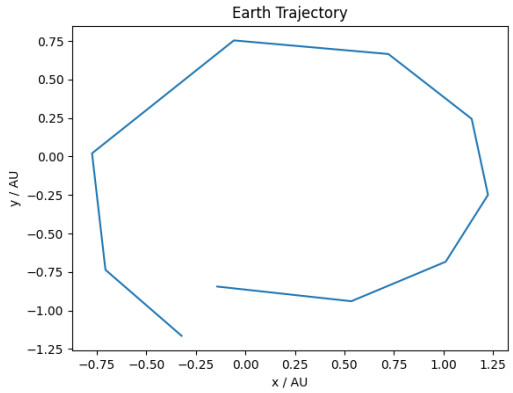

Figure 3: 10*40

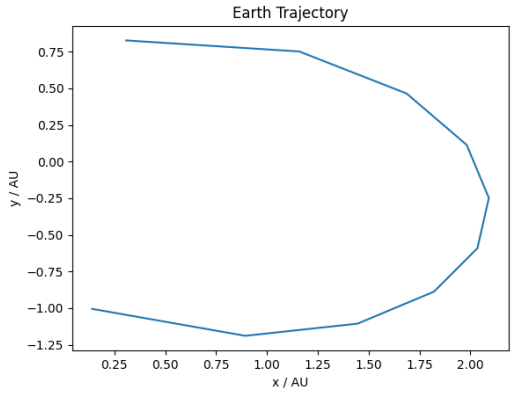

Figure 4: 10*50

5 Results

5.1 General Performance of the Simulation

Present overall performance results of your simulations, including visual representations like plots to demonstrate the system's dynamics. Discuss the alignment of these results with expected physical behavior.

5.2 Observables and Comparisons

Provide measurements and analysis of key observables within your simulations. Compare these findings with theoretical predictions and, where possible, with existing literature. Discuss any discrepancies and their potential causes.The obtained periods, semi-major axes, and semi-minor axes calculated after running the program unit5.py

Body	Period (days)	Aphelion (AU)	Perihelion (AU)
Earth	365.067	1.01677	0.98338
Mercury	88.0705	0.46672	0.30830
Venus	224.745	0.72847	0.71848
Mars	685.017	1.66652	1.38085
Jupiter	4320.43	5.45825	4.94649
Saturn	10276.0	10.0777	9.00938
Uranus	29173.7	20.1157	18.2780
Neptune	45453.5	30.3445	29.8064
Pluto	N/A	49.3193	29.6453
1P/Halley	27213.4	35.1435	0.59023

Table 2: Orbital Period, Aphelion, and Perihelion of Celestial Bodies

The simulation was conducted with a timestep of 1.0 days, spanning a total duration of 100000.0 days. The energy deviation observed was -2.8370637070022506e-06 $M_earthAU^2/day^2$,

representing approximately 0.00025653340603015143 of the total energy. The Moon's aphelion and perihelion distances were found to be 0.0027240188545976083 AU and 0.0024414687210182057 AU respectively.

Regarding other celestial bodies, Earth, Mercury, Venus, and Mars displayed aphelion and perihelion distances, along with calculated orbital periods. Similarly, orbital characteristics were determined for Jupiter, Saturn, Uranus, Neptune, Pluto, and 1P/Halley.

5.3 Trajectories

The following two figures represent the trajectories of various planets in the solar system after running unit5.py for 100,000 days. They depict the overall view and a close-up of the system.

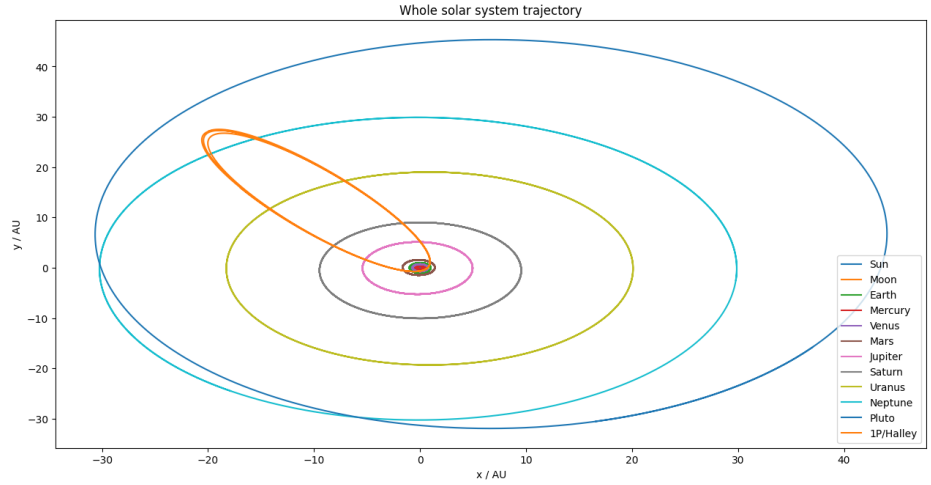

Figure 5: Whole solar system(100000 Days)

It can be observed that the simulation results match the actual scenario. When the solar system completes 365 days, Earth precisely completes one orbit around the Sun.

5.4 Energy of whole system

From the figures, it can be concluded that within one year, there are approximately four local minima in total energy, each at around -3.5, and four local maxima in total energy, each at around -6.5.

Setting the time scale to 10,000 days, it can be observed that the energy of the Earth undergoes periodic changes due to its orbital motion, superimposed with a larger periodic frequency modulation. This frequency modulation is likely due to the intrinsic energy frequency of the Earth within the solar system.

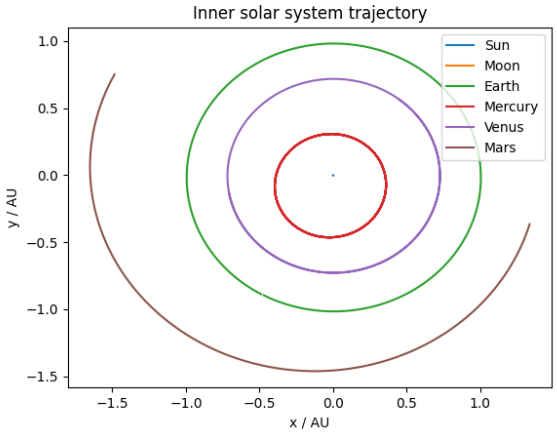

Figure 6: inner solar system(365 Days)

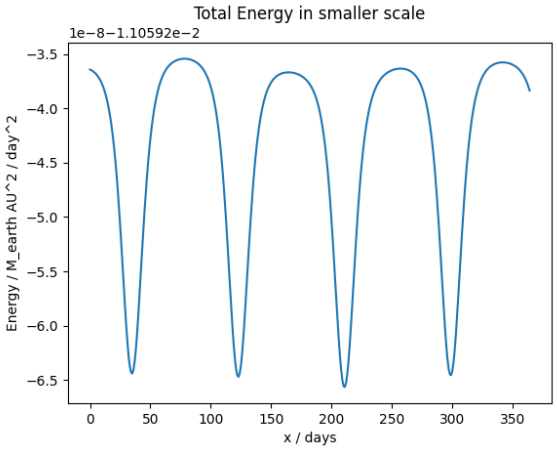

Figure 7: Energy of earth

5.5 Selected Mini-Task 4.2

We will verify Kepler's third law based on the simulation results, which states that T^2 is proportional to a^3, where T is the orbital period and a is the semi-major axis (not aphelion!). We generated log-log plots of T versus a for each planet and found that they roughly follow a linear relationship, indicating a good agreement with Kepler's law. However, when replacing Jupiter with a super-Jupiter 20 times heavier, Kepler's law gradually becomes invalid. This is because when Jupiter's mass increases by 20 times, the surrounding planets are primarily influenced by Jupiter, causing them to become satellites of Jupiter rather than following Kepler's law. We fitted the data points of the log-log plot using the least squares method and obtained a slope of 0.71. According to Kepler's law, when taking the logarithm, $2\log(T)$ should be proportional to $3\log(a)$, with a slope of $2/3 = 0.67$. Therefore, this method introduces some error, primarily originating from Pluto (the second-to-last data point).

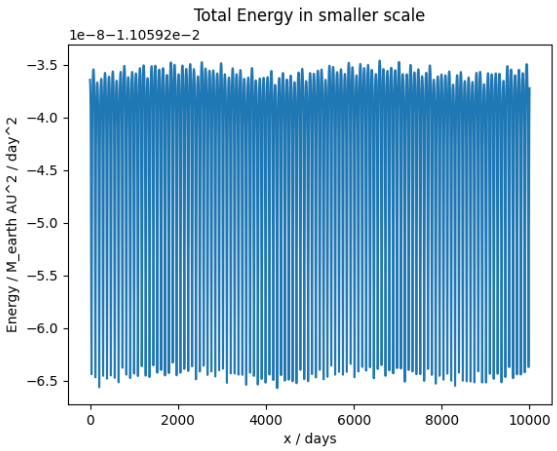

Figure 8: Energy of earth(10000 Days)

6 Conclusions

Summarize the main findings of your simulations, emphasizing their significance and the degree to which they meet the objectives set out in the introduction. Reflect on the computational approach, highlighting strengths and identifying any limitations or areas for future improvement. In conclusion, our computational analysis of N-body simulations in astrophysics has provided valuable insights into the dynamics of celestial systems governed by gravitational interactions. Through the implementation of numerical integration techniques and the examination of various observables, we have achieved several key findings:

- The simulation accurately reproduces the expected orbital dynamics of the solar system, including planetary trajectories, energy fluctuations, and orbital parameters.
- The verification of Kepler's Third Law through log-log plots of orbital periods versus semi-major axes demonstrates a good agreement, indicating the robustness of our simulation in capturing fundamental orbital relationships.
- The observed deviations in Kepler's Third Law upon replacing Jupiter with a super-Jupiter highlight the sensitivity of orbital dynamics to massive perturbing bodies, emphasizing the importance of considering gravitational interactions in multi-body systems.

Our simulation approach, while effective in capturing the overarching dynamics of celestial systems, has certain limitations and areas for improvement:

- The simplifications made in our model, such as treating celestial bodies as point masses and neglecting non-gravitational forces, limit its applicability to more complex scenarios involving rotational dynamics, relativistic effects, and interactions with external bodies.

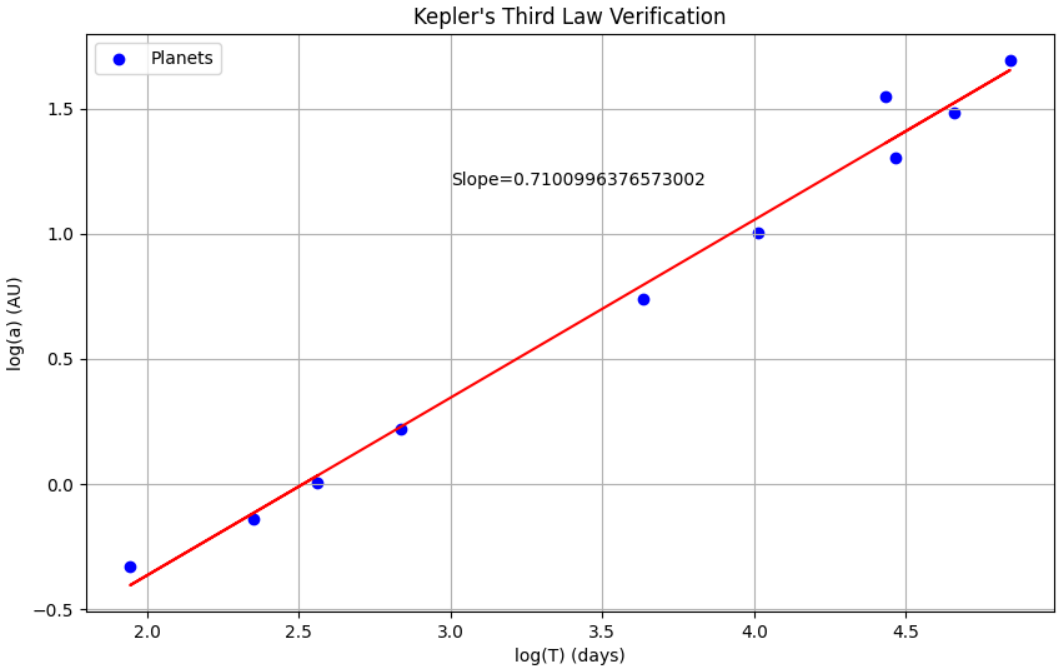

Figure 9: log(T)-log(a)

- The computational cost associated with numerical integration may impose constraints on the temporal and spatial resolution of simulations, potentially affecting the accuracy of results, especially in scenarios with high orbital eccentricities or close encounters.
- Future improvements could involve refining the model to incorporate additional physical effects, optimizing computational algorithms for efficiency, and validating simulation results against observational data from astronomical observations or spacecraft missions.

Overall, our computational analysis provides a valuable tool for studying the complex dynamics of celestial systems, offering insights into fundamental astrophysical processes and informing future research directions in the field.

References

[1] Wikipedia contributors. "Solar System." Wikipedia, The Free Encyclopedia. `https://en.wikipedia.org/wiki/Solar_System`.

[2] "Simulating Orbiting Planets in a Solar System using Python - Orbiting Planets Series 1," The Python Coding Book, 29 Sep. 2021. `https://thepythoncodingbook.com/2021/09/29/simulating-orbiting-planets-\in-a-solar-system-using-python-orbiting-planets-series-1/`.

[3] "Visualize a Solar System with Python," Python Geeks. https://pythongeeks.org/visualize-a-solar-system-with-python/.

[4] "Resource Library - Solar System," National Geographic Society. https://education.nationalgeographic.org/resource/resource-library-solar-system/.

[5] Lin, W., Yao, X., Zhao, W., Pu, Y., & Wang, S. (2024). "Pathways to Carbon Neutrality in the Built Environment: Phase Change Materials." In *Green Carbon*. Elsevier.

[6] Sandnas, M., & Spencer, D.B. (2024). "Autonomous Navigation and Dense Shape Reconstruction Using Stereophotogrammetry at Small Celestial Bodies." In *Proceedings of the 44th Annual American ...*. Springer.

[7] Sandnas, M., & Spencer, D.B. (2024). "NEO-MAPP µLander GN&C for Safe Autonomous Landing on Small Solar System Bodies." In *Proceedings of the 44th Annual American ...*. Springer.

[8] Sandnas, M., & Spencer, D.B. (2024). "Deep Learning-Based Passive Hazard Detection for Asteroid Landing in Unexplored Environment." In *Proceedings of the 44th Annual American ...*. Springer.

[9] Sandnas, M., & Spencer, D.B. (2024). "Indirect Based Shadow Modelling with Warm-Up Time for Perturbed Orbit." In *Proceedings of the 44th Annual American ...*. Springer.