Mapping the Dance of the Stars: Unveiling the Dynamics of N-Body Simulations

Gauss T. Chang *

December 17, 2024

Abstract

The N-body problem lies at the heart of understanding complex gravitational interactions, from planetary systems to galactic dynamics. This article explores the computational approach to solving the N-body problem, leveraging the power of Fortran for efficient simulation. I analyze the evolution of the system's energy and angular momentum to ensure conservation laws are upheld, a crucial benchmark for the accuracy of numerical methods. Additionally, statistical analyses provide insights into the emergent behaviors and stability of the system, further illustrating the robustness of the simulation.

This work sheds light on how N-body simulations continue to transform our understanding of the cosmos, offering a gateway to unraveling the mysteries of celestial mechanics.

I Introduction

The N-body problem, which seeks to determine the motions of N distinct masses under mutual gravitational influence, has long been central to our understanding of celestial mechanics and astrophysical systems. While closed-form solutions are limited to special cases, computational simulations now serve as essential tools for studying the evolution of complex systems such as star clusters, planetary systems, and galaxies.

Despite its importance, the N-body problem poses significant computational and numerical challenges. The complexity of direct pairwise force evaluations scales as $O(N^2)$, becoming prohibitively expensive for large N. Additionally, the need to maintain physical fidelity—such as conserving energy and angular momentum—demands careful selection of numerical integration methods and collision-handling procedures.

This article focuses on a computational ap-

*Department of Physics, National Taiwan University, Taipei, Taiwan b09501028@ntu.edu.tw

proach that harnesses both the classical laws governing gravitational interaction and modern approximation techniques to accelerate computations. We implement and test a Barnes–Hut algorithm to reduce computational complexity to approximately $O(N \log N)$, and we monitor key invariants such as total energy and angular momentum to assess the quality of the simulation. Furthermore, we explore performance improvements and discuss approaches to scaling these methods to larger systems.

In the following sections, we detail the theoretical foundations and algorithms employed (Section II), describe the simulation setup (Section III), present the results, and evaluate the performance and fidelity of my approach.

II Algorithm

In order to model the gravitational interactions among N bodies, we start from fundamental physical laws and incorporate efficient computational techniques. This section first reviews Newton's Law of Gravitation, the force model at the core of my simulation, and then introduces the Barnes–Hut algorithm, which significantly reduces the computational burden for large N.

II.I Newton's Law of Gravitation

The gravitational force between two point masses m_1 and m_2 separated by a distance vector $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$ is given by

$$\mathbf{F} = -G\frac{m_1 m_2}{|\mathbf{r}|^3}\mathbf{r}, \tag{1}$$

where G is the gravitational constant. This force is attractive and acts along the line connecting the two masses. In an N-body system, each particle experiences forces from every other particle, leading to a set of coupled, nonlinear ordinary differential equations. For large N, evaluating each pairwise force directly leads to an $O(N^2)$ complexity, a severe limitation for computational studies of large-scale structures.

To handle dynamical evolution, we employ numerical integration methods. In this study, we utilize time-stepping integrators that update particle velocities and positions incrementally. Ensuring that these integrators preserve essential physical invariants, such as total energy and angular momentum, is key to obtaining meaningful and stable long-term results.

II.II Barnes–Hut Algorithm[1]

The Barnes–Hut algorithm addresses the scaling issue inherent in direct N-body simulations. Rather than evaluating every particle-particle interaction, the algorithm builds a hierarchical tree structure to group particles that are sufficiently distant into a single effective mass at their combined center of mass. By treating distant clusters of particles as single entities, the force calculation for a given particle becomes more efficient, reducing the typical complexity to $O(N \log N)$.

The core idea is to recursively subdivide the computational domain into cells (quadrants in 2D or octants in 3D), constructing a tree that represents the spatial distribution of particles. Each internal node of the tree corresponds to a cell containing a group of particles, characterized by their total mass and center of mass. During the force calculation phase, if a node is sufficiently far away relative to its size, it can be used as a single approximation source. Otherwise, the tree is traversed further down to obtain finer-grained information.

By carefully tuning parameters such as the opening angle θ, one can achieve a balance between accuracy and performance. A smaller θ yields more accurate results at the cost of additional computations, whereas a larger θ trades off accuracy for speed.

II.III Time Integration

In this work, we start with a simple Euler method for time integration. For each particle, given a timestep Δt:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \dot{\mathbf{x}}(t) \cdot \Delta t \tag{2}$$

$$\dot{\mathbf{x}}(t + \Delta t) = \dot{\mathbf{x}}(t) + \ddot{\mathbf{x}}(t) \cdot \Delta t \tag{3}$$

However, the Euler method, while straightforward, is only first-order accurate in Δt and can lead to significant energy drift over long timescales. To improve accuracy and better conserve physical invariants such as total energy and angular momentum, we incorporate higher-order integration schemes derived from the Taylor expansion of position and velocity at $t + \Delta t$:

$$\begin{aligned}\mathbf{x_1} &= \mathbf{x_0} + \frac{1}{2}\left(\dot{\mathbf{x}}_\mathbf{0} + \dot{\mathbf{x}}_\mathbf{1}\right)\Delta t \\ &\quad + \frac{1}{12}\left(\mathbf{a_0} - \mathbf{a_1}\right)\Delta t^2 + O\left(\Delta t^5\right) \qquad (4)\\ \dot{\mathbf{x}}_\mathbf{1} &= \dot{\mathbf{x}}_\mathbf{0} + \frac{1}{2}\left(\mathbf{a_0} + \mathbf{a_1}\right)\Delta t \\ &\quad + \frac{1}{12}\left(\dot{\mathbf{a}}_\mathbf{0} - \dot{\mathbf{a}}_\mathbf{1}\right)\Delta t^2 + O\left(\Delta t^5\right) \qquad (5)\end{aligned}$$

Such higher-order and symplectic integrators (e.g., velocity Verlet) provide better long-term stability and minimize numerical artifacts that arise from approximation errors. In future enhancements, we may employ adaptive timestepping or even more sophisticated integrators to robustly handle close encounters, large N systems, and complex dynamical scenarios.

II.IV Elastic Collision

In my simulation, particles are modeled as spherical masses that can collide elastically. A collision occurs when the distance between two particles a and b is less than or equal to the sum of their radii:

$$|\mathbf{x}_a - \mathbf{x}_b| \leq (r_a + r_b) \tag{6}$$

Once a collision is detected, we resolve it by treating the encounter as a one-dimensional elastic collision along the line connecting the particle centers. Define the unit normal vector:

$$\mathbf{n} = \frac{\mathbf{x}_a - \mathbf{x}_b}{|\mathbf{x}_a - \mathbf{x}_b|} \tag{7}$$

The relative velocity of the particles along this normal direction is:

$$v_{\text{rel},n} = (\mathbf{v}_a - \mathbf{v}_b) \cdot \mathbf{n}. \tag{8}$$

If $v_{\text{rel},n}$ is positive, the particles are already moving apart, and no collision response is needed. Otherwise, we compute an impulse J that, when applied equally and oppositely to the two particles, conserves both momentum and kinetic energy. For masses m_a and m_b, the magnitude of the impulse is:

$$J = \frac{2, v_{\text{rel},n}}{\frac{1}{m_a} + \frac{1}{m_b}}. \tag{9}$$

The post-collision velocities are then updated as follows:

$$\mathbf{v}'_a = \mathbf{v}_a - \frac{J}{m_a}\mathbf{n}, \ \mathbf{v}'_b \quad = \mathbf{v}_b + \frac{J}{m_b}\mathbf{n}. \tag{10}$$

This formulation ensures that the collision is perfectly elastic. Kinetic energy and linear momentum along the collision line are conserved, and the particles no longer overlap following the collision response. Such a treatment, while idealized, is sufficient for maintaining physically consistent behavior in the simulated system and preventing unrealistic particle interpenetrations.

III Simulation Setup

For my simulations, we begin by generating an initial distribution of N particles placed within a cubic volume of side length L. Each particle is assigned:

- **Mass:** Chosen randomly between 1 and 2, ensuring a modest mass contrast.
- **Position:** Initialized from a uniform distribution within the domain

$$-\frac{L}{2} \leq x, y, z \leq \frac{L}{2}$$

- **Velocity:** Drawn from a small uniform range (e.g., -1 to 1), providing a non-zero initial kinetic state that allows the system to evolve dynamically.
- **Radius:** Assigned between 0.001 and 0.05, enabling detection and handling of elastic collisions when particles come into close proximity.

These initial conditions, as illustrated in the project slides, serve as a simple yet flexible baseline for exploring gravitational dynamics. The time domain is discretized into steps of size Δt. At each timestep, we:

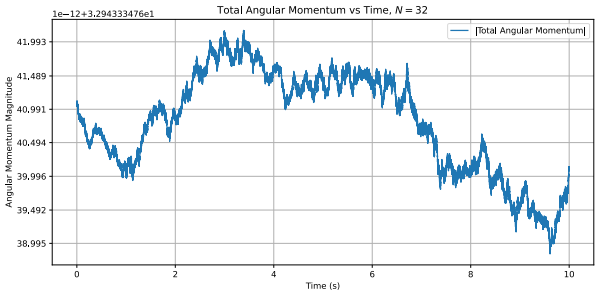
1. **Build the Barnes–Hut tree:** Particles are sorted into a hierarchical structure for efficient force approximations.
2. **Compute Forces:** Gravitational forces on each particle are derived using the Barnes–Hut approximation. Close pairs are treated at a finer resolution, while distant groups of particles are represented by their center of mass.
3. **Handle Collisions:** If two particles overlap (based on their radii), we resolve their encounter with an elastic collision model, preserving linear momentum and ensuring a physically plausible outcome.

4. **Update Kinematics:** Particle positions and velocities are integrated forward in time using a chosen time integrator. We record total energy and angular momentum after each update to monitor numerical fidelity.

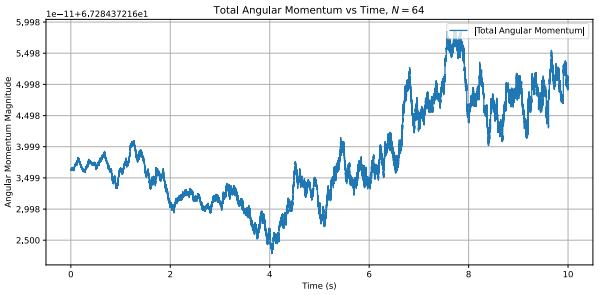
By proceeding in this manner, we can visualize the initial random configurations evolving into complex structures, as demonstrated in my slide presentations and supplementary animations. These setups, while simplified, are an effective proving ground for testing algorithmic performance, verifying energy and momentum conservation, and identifying numerical drift in large-N simulations.

IV Results

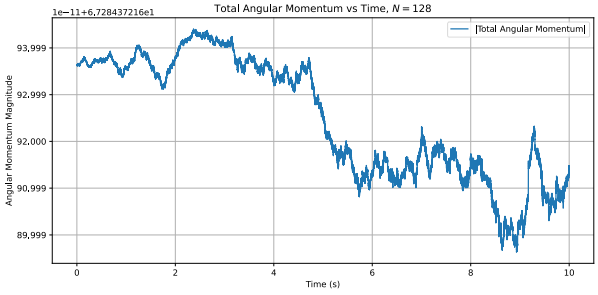
IV.I Angular Momentum



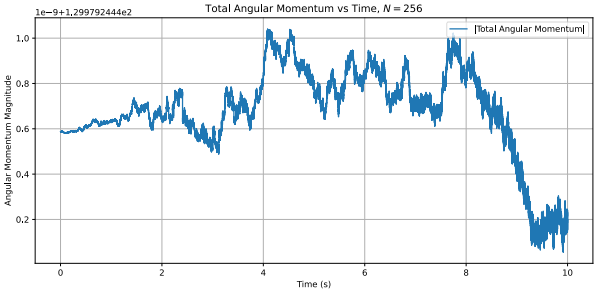
(a) Angular momentum v.s. time, $N = 32$



(b) Angular momentum v.s. time, $N = 64$



(c) Angular momentum v.s. time, $N = 128$



(d) Angular momentum v.s. time, $N = 256$

Figure 1: Angular momentum v.s. time for different particle number.

Angular momentum is a critical conserved quantity in gravitational systems, reflecting the rotational invariants of the system's motion. In principle, a well-posed N-body simulation with accurate numerical integration and appropriate timestep selection should preserve total angular momentum over time.

Figures 1a–1d show the evolution of the to-

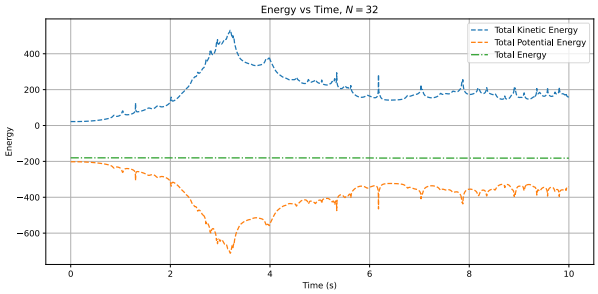
tal angular momentum magnitude as a function of time for simulations with $N = 32, 64, 128$, and 256 particles. Overall, the angular momentum remains remarkably stable throughout the simulations. Although minor fluctuations are present, it is important to consider them in the context of the scale of the system's total angular momentum.

Our simulations maintain a total angular momentum on the order of 10^1, and the observed fluctuations are on the order of 10^{-9} to 10^{-11}. In other words, the relative change in angular momentum is extremely small, on the order of 10^{-10} relative to the magnitude of the total angular momentum. Such tiny deviations are well within acceptable bounds for numerical simulations involving floating-point arithmetic and are likely attributable to truncation errors, round-off errors, and other numerical artifacts inherent in large-scale computations.

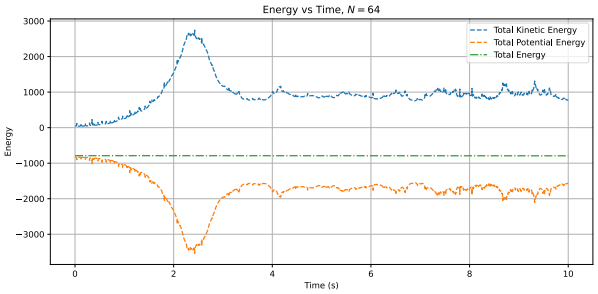
Unlike total energy, which can drift more significantly under certain conditions (especially at larger N or after numerous collision events), angular momentum appears robust and shows no trend of systematic loss or gain over time. This stability suggests that our choice of integration scheme, timestep, and collision handling mechanism effectively preserves rotational invariants. Further refinement of the integration methods or employing adaptive time-stepping could reduce these minuscule fluctuations even more, but for the purposes of this study, the current level of angular momentum conservation is fully satisfactory.

In conclusion, the angular momentum results confirm that the numerical approach and algorithms implemented here are both stable and reliable, providing a solid foundation for exploring more complex gravitational systems.

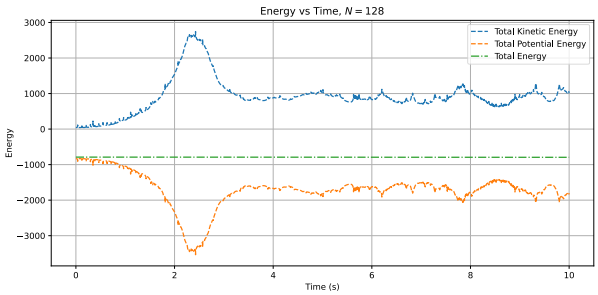
IV.II Total Energy



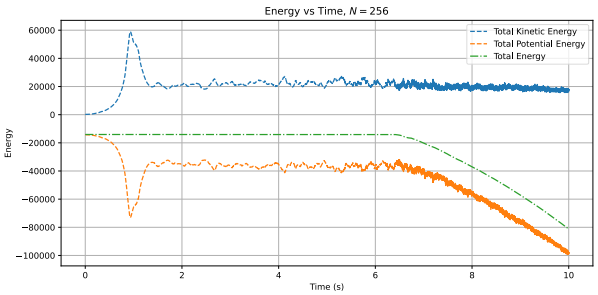
(a) Total energy v.s. time, $N = 32$



(b) Total energy v.s. time, $N = 64$



(c) Total energy v.s. time, $N = 128$



(d) Total energy v.s. time, $N = 256$

Figure 2: Total energy v.s. time for different particle number.

Conservation of total energy, $E_{\text{total}} = E_{\text{kinetic}} + E_{\text{potential}}$, is a stringent test of the numerical fidelity of N-body simulations. In principle, the gravitational N-body system should conserve its total energy if integrated perfectly and if collisions are handled in a physically consistent manner. However, numerical approximations, finite timestep sizes, and the accumulation of round-off and truncation errors can introduce energy drift over the course of the simulation.

Figures 2a–2d compares total kinetic, potential, and total energy for $N = 32, 64, 128$, and 256. For the smaller systems ($N = 32$ and $N = 64$), the total energy remains relatively stable over time, with only modest fluctuations that do not significantly diverge from the initial values. This indicates that, for these system sizes, our chosen timestep and integration schemes effectively preserve total energy on the timescales examined.

As we increase N to 128, we begin to see more pronounced variability in total energy, though the system still appears to retain rough energy balance. The fluctuations can be attributed to the increasing complexity of gravitational interactions and more frequent close encounters. Nevertheless, the energy still does not exhibit a catastrophic drift away from its initial level.

At $N = 256$, however, energy conservation

begins to break down more noticeably. Over time, the total energy curve drifts significantly, indicating that the numerical errors and collision-related adjustments accumulate faster than they can be mitigated. This behavior points toward the need for more robust integration methods, smaller timesteps, or improved collision handling as N grows large. The rapidly evolving and increasingly crowded environment of a high-N system challenges the stability of the current approach, causing greater deviations from exact energy conservation.

In summary, while total energy is reasonably well conserved for smaller systems, higher particle numbers lead to more substantial energy drift. Future improvements, such as adaptive timestep selection or higher-order symplectic integrators, may be necessary to maintain better energy conservation in large-scale N-body simulations.

V Discussion

Acknowledgments

This is a final report for a course on the Computer Programming on Geosciences, written by a senior student from the Department of Physics at National Taiwan University. Please note that I have tried my best to eliminate typos and errors. So if there are any, they are unintentional, and I ask for your understanding and forgiveness.

Author Declarations

Conflict Of Interest

The authors have no conflicts to disclose.

References

[1] Josh Barnes and Piet Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(6096):446–449, December 1986.

[2] W. Dehnen and J. I. Read. N-body simulations of gravitational dynamics. *The European Physical Journal Plus*, 126(5):55, May 2011.

[3] Lars Nylons, Mark Harris, and Jan Prins. Fast n-body simulation with cuda. *GPU gems*, 3:62–66, 2007.

[4] Sverre J Aarseth. Direct methods for n-body simulations. In *Multiple time scales*, pages 377–418. Elsevier, 1985.

[5] Joachim Gerhard Stadel. *Cosmological N-body simulations and their analysis*. University of Washington, 2001.

[6] Erich Elsen, Vaidyanathan Vishal, Mike Houston, Vijay Pande, Pat Hanrahan, and Eric Darve. N-body simulations on gpus. *arXiv preprint arXiv:0706.3060*, 2007.

[7] Daniel J Eisenstein and Piet Hut. Hop: a new group-finding algorithm for n-body simulations. *The Astrophysical Journal*, 498(1):137, 1998.

[8] Edmund Bertschinger and James M Gelb. Cosmological n-body simulations. *Computers in Physics*, 5:164–175, 1991.

[9] Evangelista Athanassoula. On the nature of bulges in general and of box/peanut bulges in particular: input from n-body simulations. *Monthly Notices of the Royal Astronomical Society*, 358(4):1477–1488, 2005.