PHYSICS CLUB
OCTOBER STEM SCHOOL

COMPASS

N-Body Gravitational Simulation Using Barnes–Hut and Particle–Mesh Methods

Kerlos Hanna, Mohamed Ahmed

February 8, 2026

STEM October Physics Club, COMPASS Computational Physics Program

Nomenclature

N	:	particles number; computational complexity
θ	:	Barnes–Hut opening angle
$\boldsymbol{r}_i$	:	position vector of particle i
$\boldsymbol{F}_i$	:	net gravitational force on particle i
$\rho(\boldsymbol{x})$	:	mass density field in Particle–Mesh simulations

Abstract

The gravitational N-body problems became the discussion of modern astrophysics. However, it's constrained by the huge computational cost of direct brute force calculations. Therefore, this paper examined two widely used efficient algorithms the Barnes–Hut (BH) algorithm and the Particle–Mesh (PM) method. The Barnes-Hut reduces the complexity by approximating distant particles group using their centre of amass, while the Particle-Mesh solves Poisson's equation on a grid using Fast Fourier Transforms to efficiently capture big-scale systems. Various simulations from 14 to 500 particles were examined and evaluated based on their trajectory accuracy, energy drift and collisions behaviour.

1 Introduction

The N-Body gravitational problem has been the cornerstone of computational physics, as it is able to model the dynamics of large-scale systems of planets and stars. It is a continuous numerical calculation using Newton's equations of motion, while computing the gravitational force pairwise between every two particles (e.g., planets). Despite the accuracy of this formulation, the classical computational solution results in a cost of $O(N^2)$. While feasible for small values of N, the cost increases dramatically as the number of particles grows. In earlier stages of this work, a direct (brute-force) simulation was implemented for a 14-body star–planet system. Although the physical phenomena were captured correctly, the simulation required nearly four hours to run.

As the demand for modelling more complex exoplanetary systems increases, there is an urgent need for more efficient computational techniques. Therefore, two of the most widely used methods in modern astrophysics were implemented and compared in the next few sections: the Barnes–Hut (BH) tree algorithm and the Particle–Mesh (PM) method.

Both methods reduce the computational cost of gravitational calculation; however, they do so through fundamentally different principles. The Barnes–Hut algorithm organises particles into a hierarchical quadtree (in two dimensions), which allows distant clusters to be approximated by their centres of mass, reducing the time complexity to $O(N \log N)$. In contrast, the Particle–Mesh method discretises space into a grid and solves for the gravitational potential from Poisson's equation using Fast Fourier Transforms, enabling simulations to scale up to millions of particles. That caused the reduction in computational time to $O(N + M \log M)$.

2 Methodology

2.1 Shockwave Model

Direct particle collisions can lead to unphysical singularities in the calculation of the force. Therefore, a shockwave model was implemented to simulate these singularities. The model calculates the velocity change experienced by a particle during a collision and is modeled using Eq. 2.1:

$$\Delta v = S \, \frac{M_{\text{two}}}{M_{\text{target}}} \, e^{-d/L} \, \hat{r} \tag{2.1}$$

This formulation aims to ensure realistic momentum transfer while preventing the singular acceleration of a particle, leading to a more reliable simulation. Computationally, collisions are detected when the distance between particles falls below a prescribed collision threshold (the distance used in this work is 0.01).

2.2 Particle–Mesh (PM) Algorithm

In the Particle–Mesh (PM) method, the discrete particle distribution is first mapped onto a uniform computational mesh to obtain a continuous mass density field. This is implemented through a Cloud-In-Cell (CIC) interpolation scheme, which distributes each particle's mass to the surrounding grid points in proportion to its relative position, resulting in a smoothed short-range force representation. The resulting density field, $\rho(\mathbf{x})$, serves as the source term for Poisson's equation (Eq. 2.2):

$$\nabla^2 \Phi = 4\pi G \rho. \tag{2.2}$$

Solving this equation directly is computationally expensive; therefore, the PM method employs a Fast Fourier Transform (FFT) technique to convert the convolution into a simple multiplication in Fourier space, enabling accurate and efficient calculation of the gravitational potential. After obtaining the potential field $\Phi(\mathbf{x})$, forces on the mesh are computed using finite-difference approximations of the gradient, and these forces are then interpolated back to the particle positions using the same CIC scheme. The PM approach offers a controllable trade-off between accuracy and efficiency through the mesh resolution N_{g}^2. However, the mesh cannot resolve individual collisions or the shockwave structure, so the PM method is combined with additional handlers. In this work, it is integrated with a simple Leapfrog integrator, which provides long-term simulations with good conservation of energy and angular momentum.

2.3 Barnes–Hut Algorithm

The Barnes–Hut (BH) technique reduces the computational cost of the gravitational calculation from $O(N^2)$ to $O(N \log N)$, thereby significantly increasing simulation speed (up to roughly 770× in medium-scale simulations). Instead of computing the gravitational force exerted by each particle on every other particle, it approximates the combined influence of distant particles as a single pseudo-body located at their center of mass.

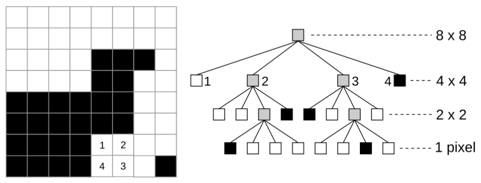

Figure 1: Schematic illustration of a quadtree used in the Barnes–Hut algorithm.

The BH algorithm recursively subdivides the simulation domain into a hierarchical tree structure (a quadtree in two dimensions), as shown in Figure 1. Initially, the entire domain is represented by a single square cell. Whenever a cell contains more than one particle, it is subdivided into four equal child cells. This recursive partitioning continues until each leaf cell contains at most one particle or is empty. Each cell stores the total mass of the particles it contains and the corresponding center of mass.

When evaluating the gravitational force on a given particle, the simulation traverses the quadtree and, for each cell, decides whether it is sufficiently far away to be approximated as a single mass using the multipole acceptance criterion, as demonstrated in Eq. 2.3.

$$\frac{s}{d} < \theta, \tag{2.3}$$

where s is the width of the cell, d is the distance from the particle to the cell's center of mass, and θ is a user-defined accuracy parameter. If the criterion is fulfilled, the cell contributes one approximated force calculation; otherwise, the traversal continues to the children of that cell and their individual contributions are evaluated.

3 Results and Discussion

The following section represents a detailed comparison of the simulation using the classical (brute-force), Barnes–Hut (BH) and Particle–Mesh (PM) methods. The comparison mainly focuses on trajectory evolution, collision dynamics, heatmaps and energy drift.

3.1 Two-Dimensional Trajectory

For small systems ($N = 14$), all three methods approached nearly the same orbital paths (Figure 2). This confirms the correctness of preserving the essential physical dynamics for both methods. However, after long integration times, some minor deviations were observed due to the accumulation of numerical error from both methods, rather than an algorithmic deficiency.

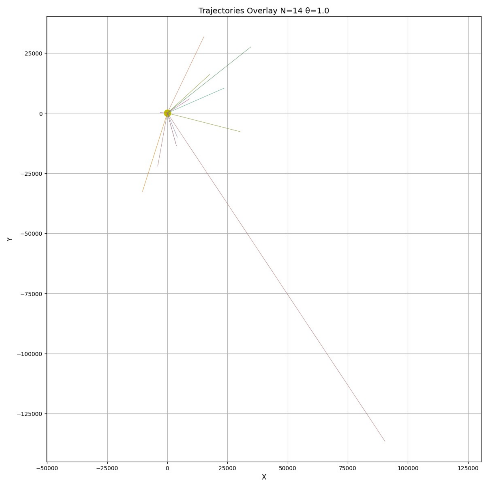

Figure 2: Two-dimensional trajectories for $N = 14$ by Barnes–Hut.

As the system size increased to $N = 100$, a qualitative difference between methods becomes more observable. The Barnes–Hut (BH) simulation (Figure 3) maintained a stable structure while significantly reducing computation time due to the hierarchical force approximation. The resulting trajectory path was smooth with minimal loss in physical fidelity. In contrast, the Particle–Mesh (PM) offered smoother and more diffused trajectories at smaller spatial scales, as demonstrated in Figure 4.

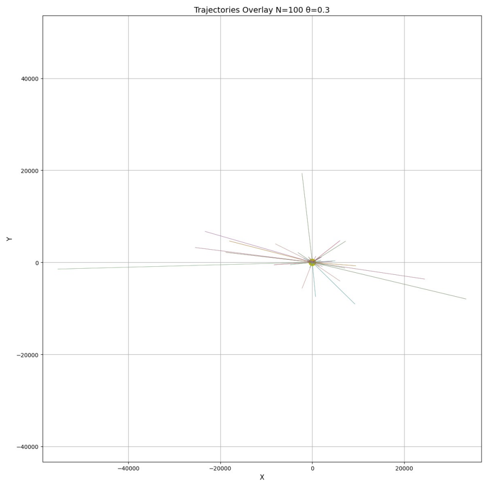

Figure 3: Two-dimensional trajectories for $N = 100$ using the Barnes–Hut method.

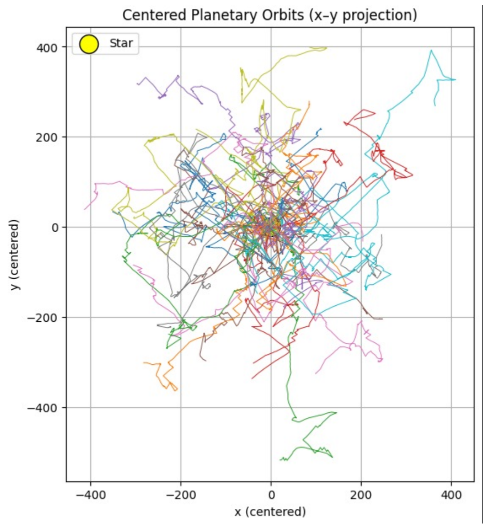

Figure 4: Two-dimensional trajectories for $N = 100$ using the Particle–Mesh method.

When the particles exceeded $N = 500$, the limitations of classical methods become clearly observed; the brute-force method was no longer computationally feasible. In comparison, the Barnes–Hut continued to produce coherent and stable trajectories. Also, the Particle–Mesh method exhibited smooth and collective trajectories with more complex strucutre shown in (Figure 5). The local orbitals were averaged out by the mesh resolution, resulting in reduced spatial detail. However, the overall structure remained stable, demonstrating great efficiency in large-scale simulations despite reduced small-scale resolution.

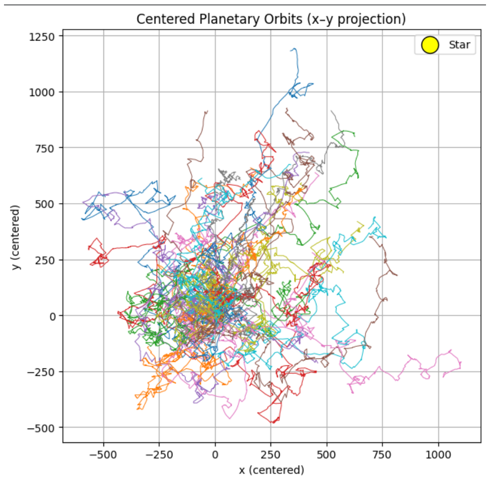

Figure 5: Two-dimensional trajectories for $N = 500$ using the Particle–Mesh method.

3.2 Collision Heatmap

A collision heatmap was generated for different simulations by recording particle interactions occurring when the inter-particle separation fell below 0.01. The map highlights regions of frequent collisions. The Particle–Mesh method effectively captured the overall spatial distribution of interactions. Compared to the Barnes–Hut simulation, which preserves more local detail, the mesh-based method produced more diffused structures while still reflecting the system's collective dynamics (Figure 6).

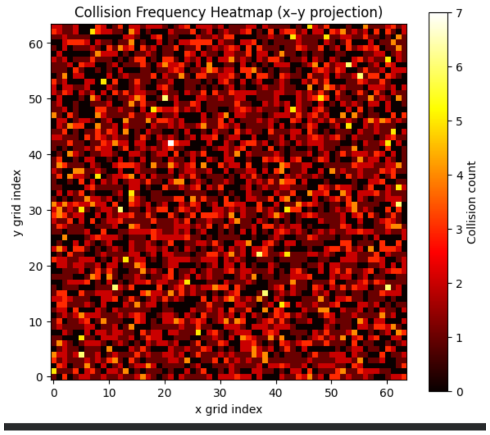

Figure 6: Collision heatmap for the Particle–Mesh simulation (collision threshold: 0.01, N=500).

3.3 Energy Drift and Conservation Behaviour

Figure 7 shows the total energy drift for the Particle–Mesh (PM) simulation at $N = 500$; the system showed an increase in energy deviation monotonically with the simulation step. Each step indicates cumulative numerical errors inside the discretisation of the grid. Despite the magnitude of the energy drift, the system remains stable through the simulation.

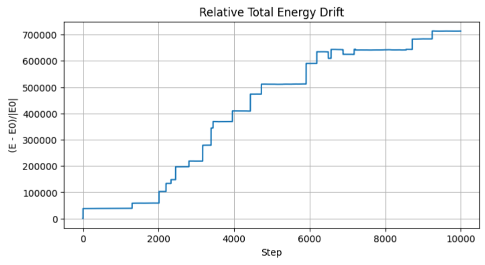

Figure 7: Total energy drift for the Particle–Mesh simulation at $N = 500$.

At the same time, the Barnes–Hut (BH) showed a strong dependence of energy conservation on the opening angle parameter (θ). Smaller values of θ lead to improved force accuracy and reduced energy drift, but with more computational time. Two simulations were examined with the same number of particles ($N = 100$) but different parameter values: $\theta = 1.5$ achieved an energy drift of 2.94×10^2, while $\theta = 1.0$ obtained an energy drift of 1.32×10^1 (Figures 8 and 9).

Compared to the PM simulation results, the BH method provides superior energy conservation. The larger energy drift observed in the PM simulation comes from the smoothing of short-range forces from grid-based solvers, which reduces the efficiency of energy conservation. Angular momentum further supports that, as PM momentum shows smoother long-term behaviour (Figure 10).

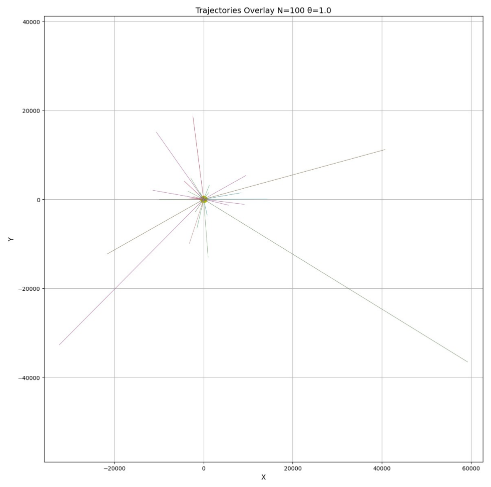

Figure 8: Barnes–Hut trajectory for $N = 100$ with $\theta = 1.0$ (Average runtime: 500.50 s, Energy drift: 1.32×10^1, Collisions: 1.0).

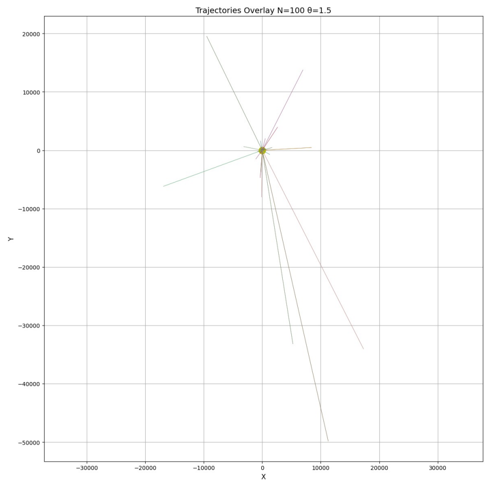

Figure 9: Barnes–Hut trajectory for $N = 100$ with $\theta = 1.5$ (Average runtime: 474.76 s, Energy drift: 2.94×10^2, Collisions: 1.0).

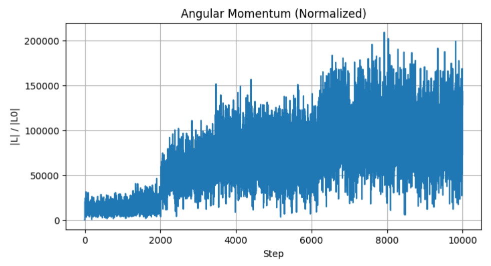

Figure 10: Angular momentum behaviour for the Particle–Mesh method over long integration times.

4 Conclusion

To conclude, the study focused on optimizing the simulations using Barnes–Hut and Particle–Mesh algorithms. The simulations explored systems ranging from 14 to 500 particles, and it demonstrates a comparison between the efficiency and time complexity of these algorithms compared to the classical approach with a shockwave model.

Bibliography

[1] Barnes, J., and Hut, P.: A Hierarchical $O(N \log N)$ Force-Calculation Algorithm, *Nature*, Vol. 324, 1986, pp. 446–449.

[2] Binney, J., and Tremaine, S.: *Galactic Dynamics*, 2nd ed., Princeton University Press, Princeton, 2008.

[3] Benz, W., and Asphaug, E.: Catastrophic Disruptions Revisited, *Icarus*, Vol. 142, 1999, pp. 5–20.

[4] Burtscher, M., and Pingali, K.: An Efficient CUDA Implementation of the Barnes–Hut N-Body Algorithm, GPU Computing Gems Emerald Edition, Morgan Kaufmann, Burlington, MA, 2011.

[5] Cooley, J. W., and Tukey, J. W.: An Algorithm for the Machine Calculation of Complex Fourier Series, *Math. Comput.*, Vol. 19, 1965, pp. 297–301.

[6] Dehnen, W., and Read, J. I.: N-Body Simulations of Gravitational Dynamics, *Eur. Phys. J. Plus*, Vol. 126, 2011, pp. 1–28.

[7] Hairer, E., Lubich, C., and Wanner, G.: *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer, Berlin, 2006.

[8] Hockney, R. W., and Eastwood, J. W.: *Computer Simulation Using Particles*, Taylor & Francis, New York, 1988.

[9] Lam, S. K., Pitrou, A., and Seibert, S.: Numba: A LLVM-Based Python JIT Compiler, Proc. 2nd Workshop on the LLVM Compiler Infrastructure in HPC, 2015.

[10] Landau, L. D., and Lifshitz, E. M.: *Mechanics*, 3rd ed., Pergamon Press, Oxford, 1976.

[11] Lichtenberg, A. J., and Lieberman, M. A.: *Regular and Chaotic Dynamics*, Springer, New York, 1992.

[12] Monaghan, J. J.: Smoothed Particle Hydrodynamics, *Annu. Rev. Astron. Astrophys.*, Vol. 30, 1992, pp. 543–574.

[13] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P.: *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, Cambridge, 2007.

[14] Richardson, D. C., Bottke, W. F., and Love, S. G.: Tidal Distortion and Disruption of Binary Asteroids, *Icarus*, Vol. 134, 1998, pp. 47–76.

[15] Sanz-Serna, J. M.: Symplectic Integrators for Hamiltonian Problems, *Acta Numer.*, Vol. 1, 1992, pp. 243–286.

[16] Springel, V.: The Cosmological Simulation Code GADGET-2, *Mon. Not. R. Astron. Soc.*, Vol. 364, 2005, pp. 1105–1134.

[17] Sundar, H., et al.: Bottom-Up Construction and 2:1 Balance Refinement of Linear Octrees, *SIAM J. Sci. Comput.*, Vol. 30, 2008, pp. 2675–2708.

[18] Van der Walt, S., Colbert, S. C., and Varoquaux, G.: The NumPy Array: A Structure for Efficient Numerical Computation, *Comput. Sci. Eng.*, Vol. 13, 2011, pp. 22–30.

[19] Virtanen, P., et al.: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, *Nat. Methods*, Vol. 17, 2020, pp. 261–272.

[20] Yoshida, H.: Construction of Higher-Order Symplectic Integrators, *Phys. Lett. A*, Vol. 150, 1990, pp. 262–268.