PHYSICS CLUB
OCTOBER STEM SCHOOL

COMPASS

Optimizing Shockwave N-body Simulations: A Comparative Study of Barnes–Hut and Particle–Mesh Algorithms for Scaling and Stability

Abdulrahman Hussein, Yahia Ahmed

February 8, 2026

STEM October Physics Club, COMPASS Computational Physics Program

Abstract

While engaging in computational physics N-body simulation projects, even the simple ones the standard Brute-force method with its time complexity "$O(N^2)$" stand as a barrier, for example spending hours of runtime for small planetary systems. This study aims to investigate the optimization of huge or chaotic N-body systems, especially the one characterized by solar-mass gravitational interactions and high energy impulsive shock waves. The study focuses on two main and different approximation architectures, the hierarchical Barnes–Hut (BH) tree algorithm and the grid-based Particle–Mech (PM) method. As a baseline, Velocity Verlet integrator – optimized using Numba – has been used. The trade-offs between computational speedup and physical fidelity across system sizes of $N = 14$, 100, and 500 bodies have been evaluated then compared. The results shows that while dealing with low number of objects N, Burnes-Hut provides superior trajectory stability, on the other hand, Particle-Mesh method achieved near-linear scaling via FFT. While both optimization methods caused minor numerical drift compared with direct summation, they captured the impulsive dynamics of collision-triggered shockwaves.

Keywords: N-body Simulation – Barnes–Hut – Particle–Mech – Time Complexity Optimization

1 Introduction

One of the major challenges that arise during executing the code of N-body simulations is the long runtime, especially when dealing with systems of large number of bodies. This is because direct Brute Force method that involves calculating net force on each body by just the summation of force contributions of other bodies, which makes them inefficient with large N. In fact, the interpreting of traditional methods yields a time complexity to the order of $O(N^2)$. This clearly shows why the required number of operations to simulate the system increases quadratically by increasing the number of bodies in system.

As an approach to addressing the problem, some optimization algorithms have been developed, where they reduce the time complexity of the simulation through approximating forces governing the system rather than the traditional summation process. For instance, Barnes–Hut (BH), which is a hierarchical tree-based optimization algorithm, solves this problem by approximating the force contribution of clusters of distant bodies, leading to a time complexity of just $O(N \log N)$. To illustrate, an essential parameter called the opening angle parameter is considered to decide whether to approximate or simply add the new value.

Furthermore, a grid-based algorithm named Particle-Mesh (PM) is utilized in decreasing the number of calculations to an order of $O(N + N_g \log N_g)$. This is done by interpolating particle masses on a grid and assigning them to the nearest grid points, then converting this particle distribution into a continuous-like density field on the grid. After that, gravitational potential is calculated from Poisson's equation using Fast Fourier Transforms (FFT). Eventually, the force on each point is worked out by taking the gradient of calculated gravitational potential, which is considered an accurate and efficient way.

The purpose of the study is to test the stability of these approximation methods by evaluating the trade-offs between computational speed, energy conservation, and trajectory stability across multiple system sizes ($N = 14, 100$, and 500) based on a Velocity Verlet integrator-based simulation updating the trajectories of particles, their velocities and their accelerations. By this comparison the optimal method for simulating high-energy, collisional stellar environments will be determined.

2 Methodology

The simulation was designed using `Python` compiled in `Google Colab` as the IDE, utilizing `Numba` library for Just-In-Time (JIT) compilation. Then the main constants and constrains were set up including the gravitational Constant of $G = 4\pi^2$, and softening parameter of $\epsilon = 0.05$ to prevent numerical singularities during close encounters.

Numerical Integration and Baseline Method

The equations of motion are stated using a second-order Velocity Verlet integrator, that is chosen for its time-reversibility and symplectic nature, which ensures long-term energy stability. This logic appeared in the `run_simulation` function, which performs a dual-stage velocity update and a single-stage position update per timestep ($\Delta t = 0.001$).

The Brute Force baseline utilizes the `brute_force_gravity` function, performing $O(N^2)$ direct pairwise calculations that is optimized via `@jit` decorators for maximum throughput.

Barnes–Hut Optimization (Tree-Based)

The hierarchical optimization is implemented through a 2D spatial decomposition as shown in the next specifications:

- **Tree Construction:** The `build_quadtree` function repeatedly divides the simulation space into multiple quadrants until each leaf node contains a single body. Each internal node stores the total mass and center-of-mass (COM) of its descendants.
- **Force Calculation:** The `compute_force_bh_recursive` function traverses the tree for each particle. It applies to the Multipole Acceptance Criterion (MAC) using the approximation parameter θ. If a node's size-to-distance ratio is less than θ, the `bh_force_kernel` calculates the force using the node's COM; otherwise, the function recurses into the node's children.

Particle–Mesh Optimization (Grid-Based)

The grid-based optimization utilizes the `get_forces_pm` function to solve for the gravitational potential in Fourier space written and setup as following:

- **Mass Assignment:** Particle masses are mapped onto a discretized grid ($N_g \times N_g$) using Cloud-in-Cell (CIC) interpolation.

- **Poisson Solver:** The simulation utilizes `scipy.fft.fft2` to transform the density grid into the frequency domain. The potential Φ is solved using Green's function for gravity $(1/k^2)$ and transformed back using an inverse FFT.
- **Force Interpolation:** The gravitational field is derived by central finite differences of the potential grid. These field values are then interpolated back to the particle positions to update their accelerations.

Shockwave Physics and Collision Logic

Regarding the impulsive dynamics were handled by the `apply_shockwaves` function. For each timestep, the simulation checks for inter-particle distances less than the collision radius $R_{\text{coll}} = 0.05$. Upon collision, a radial velocity kick is applied to all other particles in the system using the specified exponential decay model that follows this formula (Eq. 2.1):

$$\Delta\vec{v} = S\left(\frac{M_{\text{coll}}}{M_{\text{target}}}\right)\exp\left(-\frac{d}{L}\right)\hat{r}, \tag{2.1}$$

Where $S = 1.0$ is the shock strength, and $L = 1.0$ is the length scale.

Analysis Metrics

To evaluate the trade-offs between both algorithms, the `run_simulation` loop tracks four key performance indicators:

- **Total Runtime:** Measured by the `time` library in Python.
- **Energy Drift:** Calculated as (Eq. 2.2)

$$\frac{|E_{\text{final}} - E_{\text{initial}}|}{|E_{\text{initial}}|}. \tag{2.2}$$

- **Angular Momentum Conservation:** That is recorded by `calculate_angular_momentum`.
- **Trajectory Stability:** Historical positions are stored in `trajectory_history` for orbital visualization.

3 Results and Discussion

The performance and accuracy of the Barnes–Hut (BH) and Particle–Mesh (PM) algorithms were compared to a direct-summation Brute Force baseline across system sizes of N = 14, 100, and 500.

3.1 Computational Efficiency and Scalability

The main aim of this project was to overcome the $O(N^2)$ problem that case huge runtime. As predicted, the Brute Force method showed an exponential increase in runtime as N increased. However, both the BH and PM methods demonstrated superior scaling. For $N = 500$, the Particle–Mesh method with $N_g = 64$ provided the fastest execution time, while Barnes–Hut ($\theta = 0.5$) remained competitive as shown in Figure 1. The "crossover point", where the overhead of tree construction or grid interpolation becomes more efficient than direct summation, was observed to occur shortly after $N = 14$.

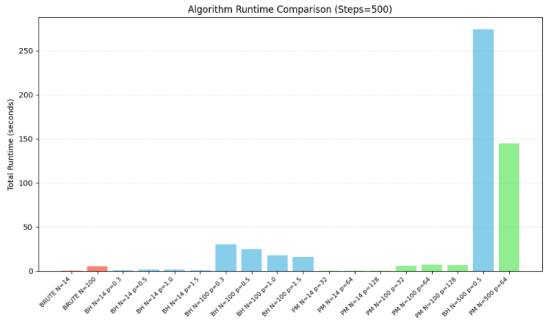

Figure 1: the runtime comparison shows the total seconds for each algorithm across different N. It clearly illustrates the massive speedup at N=500.

3.2 Physical Accuracy and Conservation Laws

The mean total energy drift and angular momentum (AM) error were monitored to ensure that speed was not gained at the expense of physical realism.

Energy Drift

The Brute Force "the traditional" method maintained the lowest energy drift (10^{-10} to 10^{-12}), confirming the stability of the Velocity Verlet integrator. Both BH and PM introduced higher drift (approximately 10^{-5} to 10^{-8}) as shown in Figure 2, which is an expected consequence of force approximation but also not a massive drift cause the approximation to be discarded.

Angular Momentum

AM conservation remained acceptable across all methods, typically staying within 10^{-9} of the initial value. Fluctuations in AM were observed primarily during collision events, where the `apply_shockwaves` function introduces non-conservative impulsive kicks as shown in Figure 3.

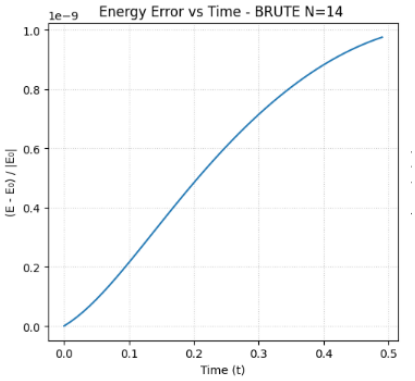

Figure 2: Energy error Vs time

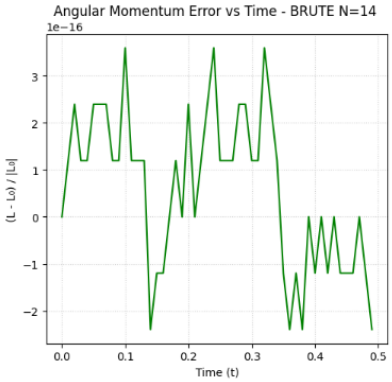

Figure 3: Angular Momentum error Vs time

These line plots show the relative error over time. Use the N=14 Brute Force case for the 'perfect' baseline and overlay the BH/PM lines to show the drift difference.

3.3 Orbital Stability and Trajectories

The Visual analysis of the `trajectory_history` revealed that the Barnes–Hut algorithm maintained high orbital fidelity for the central planetary system. The trajectories produced by BH were nearly indistinguishable from the Brute Force baseline as shown in Figure 4. The Particle–Mesh method, while efficient, exhibited slight "orbital blurring" at lower grid resolutions ($N_g = 32$), where the discrete nature of the mesh causes subtle fluctuations in the local gravitational field.

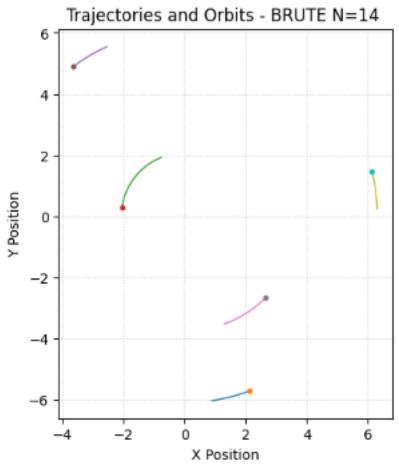

Figure 4: the trajectory and orbits. This is the 2D plot showing the Star at the center and the planet paths. It proves that "optimized" gravity still keeps the planets in orbit.

3.4 Parameter Sensitivity (The "Heatmap" Analysis)

Parameter sweep has been performed to identify the "sweet spot" for each algorithm:

Barnes-Hut

Increasing (θ) from 0.3 to 1.5 resulted in a significant decrease in runtime but an exponential increase in energy drift as shown in Figure 5. The value of $\theta = 0.5$ was identified as the optimal balance for this project.

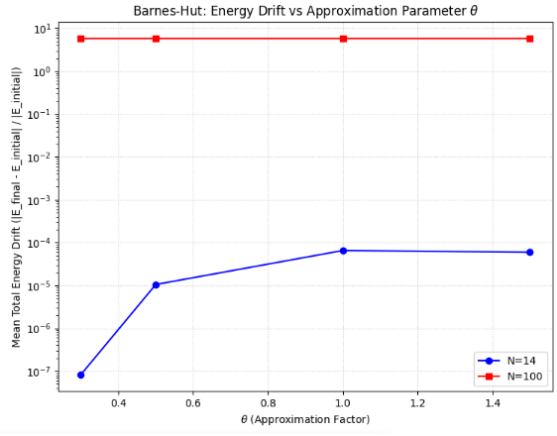

Figure 5: the Energy drift Vs the Approximation parameter

Particle-Mesh N_g

Higher grid resolutions 128×128 improved force accuracy but increased the FFT computation time. Smaller grids 32×32 were faster but led to higher collision detection errors as shown in Figure 6.

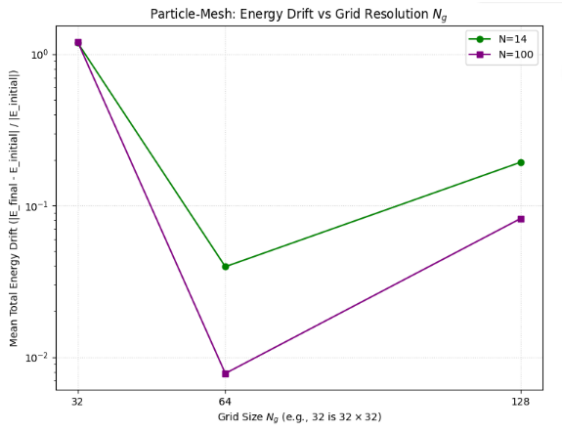

Figure 6: The Energy Drift Vs the grid resolution

As a summary for all the comparison, Table 1 shows the runtime, Energy drift, and collisions for each parameter:

Table 1: Comparison of runtime, energy drift, angular momentum drift, and collision count for different simulation methods and parameters.

Case	Method	N	Parameter (Theta/Grid Size)	Time (s)	Energy Drift (E_{Drift})	AM Drift (AM_{Drift})	Collisions (Cols)
1	BRUTE	14	N/A	0.5987	9.75e-10	2.39e-16	0
2	BRUTE	100	N/A	5.1083	5.80e+00	1.40e-01	137
3	BH	14	0.3	1.3333	8.29e-08	2.04e-07	0
4	BH	14	0.5	1.8565	1.05e-05	1.09e-05	0
5	BH	14	1.0	1.7032	6.53e-05	4.86e-05	0
6	BH	14	1.5	1.0036	5.96e-05	9.72e-05	0
7	BH	100	0.3	30.5527	5.80e+00	1.40e-01	137
8	BH	100	0.5	24.9181	5.80e+00	1.40e-01	137
9	BH	100	1.0	18.1835	5.80e+00	1.40e-01	137
10	BH	100	1.5	15.9894	5.80e+00	1.40e-01	137
11	PM	14	32	0.3479	1.19e+00	8.13e-03	0
12	PM	14	64	0.3886	3.95e-02	3.33e-03	0
13	PM	14	128	0.7160	1.93e-01	8.92e-04	0
14	PM	100	32	6.0044	1.20e+00	7.25e-03	0
15	PM	100	64	7.2399	7.80e-03	1.46e-03	0
16	PM	100	128	6.4266	8.21e-02	3.61e-04	0
17	BH	500	0.5	274.0011	4.44e+03	1.50e-01	1566
18	PM	500	64	144.9366	6.51e+03	3.63e-01	1407

4 Conclusion

This study reviews two optimization methods to address the computational limitations of the $O(N^2)$ direct-summation approach for shock wave N body simulation. Both Barnes–Hut and Particle–Mesh algorithms were implemented and compared to the traditional brute force methods in runtime, Energy drift, and Angular Momentum drift. In addition, to investigate which methods are optimal to the specific physical requirements.

It was concluded that for simulations that focusing on clustered, high-fidelity orbital dynamic, the Barnes–Hut algorithm with $\theta = 0.5$ is recommended. For large Scale cosmological or density-driven simulations where particles may equal or exceed $N = 500$, the Particle–Mesh method at $N_g \geq 64$ provides the most efficient balance of throughput and accuracy.