

Efficient N-Body Simulation Using the Barnes-Hut Algorithm

Abeen Bhattacharya / Karaen Senthilkumar

December 9, 2025

What Is a General N -Body Problem?

- A general N -body system consists of N point masses, each exerting a force on every other:

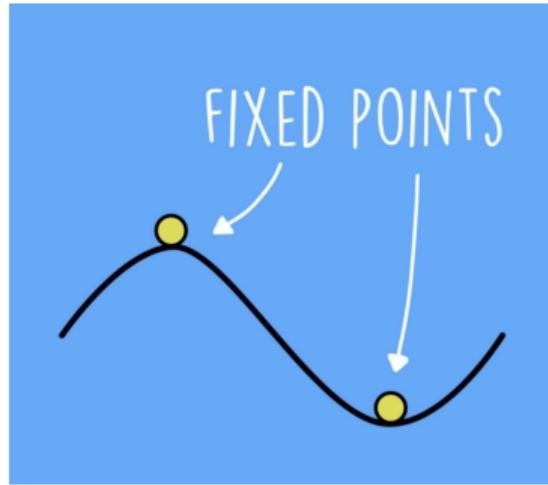
$$\{\mathbf{r}_i(t), \mathbf{v}_i(t), m_i\}_{i=1}^N, \quad m_i \mathbf{a}_i = \sum_{j \neq i} Gm_i m_j \frac{\mathbf{r}_j - \mathbf{r}_i}{\|\mathbf{r}_j - \mathbf{r}_i\|^3}.$$

- How do positions and velocities evolve over time in a way that remains numerically stable and respects conservation laws (energy, momentum)?

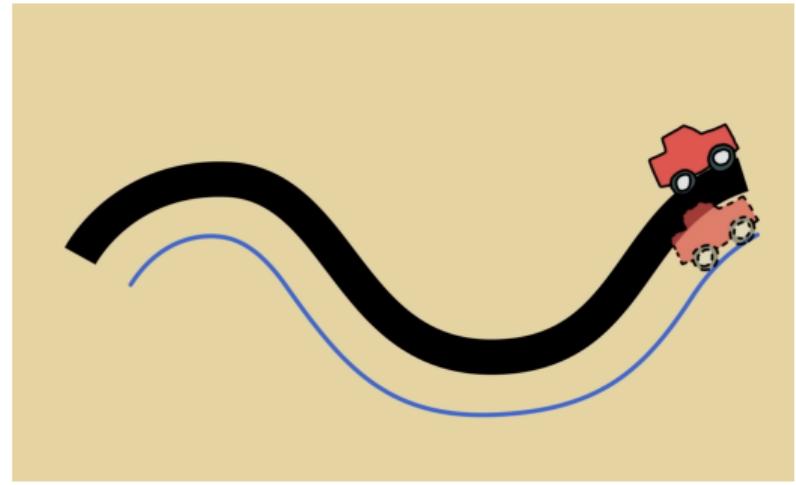
From Newton to the Solar System: A Birthday Competition

- 1687: Newton's laws of motion and universal gravitation solve the **two-body problem** exactly: closed-form conic orbits (ellipses, parabolas, hyperbolas).
- 1880s: To celebrate King Oscar II's birthday, a prize problem is posed:
 - *Given Newton's equations, is the Solar System stable for all time, or can tiny perturbations throw planets out of their orbits?*
- At first glance, it looks like a straightforward extension:
 - Two-body solutions are neat and integrable.
 - With three or more bodies, one might hope for a slightly messier but still closed-form solution.
- Reality: even adding **one** more body (the general three-body problem) fundamentally breaks the integrability and exposes chaotic behavior.

Equilibria



Unstable (left) and Stable (right) fixed points.

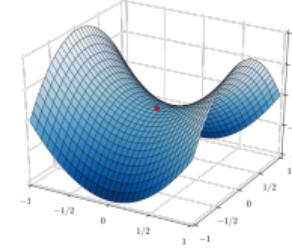
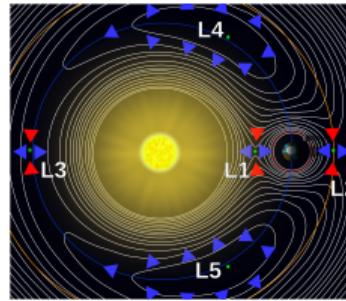


small inaccuracies in initial conditions lead to only
small deviations in trajectories

- For a long time it was assumed that *small inaccuracies in initial conditions lead to only small deviations in trajectories*; the three-body problem showed that this intuition can break down.

Special Three-Body Equilibria and Saddle Points

- Mathematicians realized that there are additional equilibrium configurations:
 - Euler (collinear points) and Lagrange (triangular points) identified equilibrium configurations in the restricted three-body problem.
 - These give the famous Lagrange points L1-L5 used in mission design (JWST, SOHO, etc.).
- However, many of these equilibria are **saddle points** in the gravitational potential:
 - Stable in some directions, unstable in others.
 - Tiny perturbations can grow over time, making long-term prediction extremely sensitive to initial conditions.



Poincaré and the Onset of Chaos

- Henri Poincaré tackled King Oscar II's competition by studying the general three-body problem.
- His key result:
 - There is **no general closed-form solution** for the three-body problem in terms of elementary functions and standard integrals.
 - The motion can display extreme sensitivity to initial conditions, bifurcations, and complex invariant sets in phase space.
- This work effectively launched the modern view of **chaos** in deterministic systems.
- Consequence for us:
 - Exact formulas for realistic N -body systems do not exist.
 - Even with perfect equations, long-term prediction is limited by numerical resolution and initial-condition uncertainty.
- Therefore, realistic modeling requires **numerical simulation** and careful error control rather than explicit analytic solutions.

From Analytic Dreams to Numerical Reality

- For real gravitational systems with many bodies:
 - **No general closed-form solution** exists (even for three bodies).
 - We instead solve Newton's equations numerically over discrete timesteps.
- Basic time-stepping idea:
 - Pick a timestep Δt .
 - At each step:
 - ① Compute gravitational forces on each body from all others.
 - ② Update velocities (e.g., via Euler or Verlet integration).
 - ③ Update positions using the new velocities.
 - Repeat this loop to advance the system in time.
- This transforms the continuous-time differential equations into a sequence of **force evaluations + updates**.
- The main numerical challenges:
 - Stability and accuracy over long horizons.
 - Managing the $O(N^2)$ force-evaluation cost as N grows.

Computational Complexity and the Need for Approximation

- Direct summation:
 - For each body i , sum contributions from all $j \neq i$.
 - Cost per timestep: $O(N^2)$ force evaluations.
- Approximations emerge as the only way forward:
 - **Mean-field methods:** treat the system as a smooth density field.
 - **Tree codes (Barnes-Hut):** group distant bodies into “super-particles” using a hierarchical spatial tree.
 - **Fast multipole methods:** more sophisticated expansions with $O(N)$ asymptotic complexity.
- Our project focuses on the Barnes-Hut tree approach, which offers:
 - Good tradeoff between simplicity, controllable error, and speed.
 - Natural parallelism that maps well onto GPU architectures.

Our Research Problem (High-Level)

- **Computational bottleneck:** Direct $O(N^2)$ force calculation quickly becomes intractable for large N (e.g., 10^5 - 10^7 bodies).
- **Physical constraint:** We cannot simply “cheat” on physics; we care about:
 - Long-term orbital stability (no artificial explosions or damping).
 - Small energy and angular momentum drift over many timesteps.
 - Correct qualitative behavior near sensitive regions (e.g., saddle points).
- **Research question:** How far can we push a tree-based approximation (Barnes-Hut) on GPUs while *still* preserving the essential physics of gravitational systems?
- **Approach in one sentence:** Build and compare a GPU-accelerated Barnes-Hut solver against direct CPU/GPU baselines, and quantify the tradeoff between speed and physical fidelity.

Why This Matters in Practice

- **Astrophysics and cosmology:**
 - Galaxy mergers, star-cluster evolution, dark matter halo formation.
 - Need to track millions to billions of particles with long-range gravity.
- **Space mission design:**
 - Spacecraft near Lagrange points, multi-body trajectories, formation flying.
 - Stability near saddle points requires accurate force modeling.
- **Plasmas and charged particles:**
 - Long-range Coulomb interactions in accelerators and fusion devices.
- Efficient N -body solvers like Barnes-Hut are therefore foundational tools across physics, engineering, and visualization.

Methodology Overview

- **Goal:** Implement and compare multiple N -body solvers to understand the speed vs. accuracy tradeoff:
 - CPU direct solver ($O(N^2)$) as a correctness reference.
 - GPU direct solver to isolate pure parallel speedup.
 - GPU Barnes-Hut solver to exploit approximate $O(N \log N)$ complexity.
- **Dimensionality:** Start with a 2D simulation for clarity and visualization; design data structures so they extend naturally to 3D (octrees).
- **Core questions:**
 - At what N does Barnes-Hut on GPU outperform direct GPU?
 - How does the opening-angle parameter θ affect energy drift and orbit quality?
 - What GPU-specific optimizations (memory layout, divergence control) matter most?

Direct $O(N^2)$ Baseline (CPU & GPU)

- **Algorithm:** For each timestep:

- ① For each body i :

- Initialize net force $\mathbf{F}_i = \mathbf{0}$.
- Loop over all $j \neq i$ and accumulate pairwise forces:

$$\mathbf{F}_i += G \frac{m_i m_j}{\|\mathbf{r}_j - \mathbf{r}_i\|^3} (\mathbf{r}_j - \mathbf{r}_i).$$

- ② Update velocities and positions using a chosen integrator (e.g., velocity-Verlet).

- **CPU implementation:**

- Straightforward nested loops; good for correctness and debug.
- Serves as a numerical ground truth for small to medium N .

- **GPU direct implementation:**

- Map one thread to each body i ; inside each thread, loop over all j .
- Use structure-of-arrays layout for positions and velocities to maximize coalescing.
- This already provides significant speedup but retains $O(N^2)$ scaling.

Barnes-Hut Tree Structure (Quadtree in 2D)

- **Core idea:** Replace groups of distant bodies with a single “super-particle” representing their total mass and center of mass.
- **Data structure:** Quadtree (2D) / Octree (3D):
 - Start from a root node covering the entire simulation domain.
 - Recursively subdivide each node into 4 quadrants (2D) until:
 - Each leaf contains at most one body, or
 - A maximum depth is reached.
 - Each internal node stores:
 - Total mass of all particles in that region.
 - Center-of-mass position.
 - Pointers/indices to its children.
- **Result:** A hierarchical spatial partition that allows multi-resolution approximations of the gravitational field.

Barnes-Hut Force Approximation

- For each body i , traverse the quadtree and decide, node by node, whether to:
 - **Approximate** a subtree as one super-particle, or
 - **Open** the node and descend to its children for finer resolution.
- **Opening-angle criterion:**
 - Let s be the size of a node (e.g., side length of its bounding square).
 - Let d be the distance from body i to the node's center of mass.
 - If $\frac{s}{d} < \theta$, treat the node as a single aggregated mass; otherwise, recurse into children.
 - θ controls the accuracy-speed tradeoff: small θ = high accuracy, lower speed; large θ = more approximation, higher speed.
- **Complexity:** In typical distributions, the number of nodes visited per body scales like $O(\log N)$, leading to an overall $O(N \log N)$ method.

CUDA Implementation Strategy

- **Data layout:**

- Store particles in structure-of-arrays form: separate arrays for x, y, v_x, v_y, m .
- Store quadtree nodes in a flat array with integer indices for children to avoid pointer chasing.

- **Tree construction:**

- Compute a global bounding box for all particles.
- Insert particles into the quadtree; in CUDA, use:
 - Atomic operations or carefully designed parallel insertion schemes.
 - Morton codes / space-filling curves as an optional improvement for locality.
- Post-process tree to compute mass and center of mass for each internal node.

- **Force kernel:**

- One thread per body traverses the tree (using an explicit stack in registers/shared memory).
- Apply the opening-angle test and accumulate forces from accepted nodes or leaves.
- Minimize warp divergence by ordering particles with similar spatial locations (and hence similar traversal paths).

Time Integration and Stability

- **Integrator choice:**

- Use a symplectic or nearly-symplectic method (e.g., velocity-Verlet) rather than naive forward Euler:
 - Better long-term energy behavior.
 - More robust orbital dynamics.

- **Update step (schematic):**

- ① Given positions and velocities at time t , build quadtree and compute forces $\mathbf{F}_i(t)$.
- ② Half-step velocity update, full-step position update, second half-step velocity update (Verlet).
- ③ Optionally adapt Δt based on maximum acceleration or local dynamical timescales (future work).

- **Diagnostics:**

- Track total energy and angular momentum over time.
- Visualize representative orbits (especially near saddle regions) to check for artifacts from excessive approximation.

Validation and Evaluation Plan

- **Correctness tests:**
 - Small- N systems (2-4 bodies) with known behavior:
 - Compare trajectories and energies between CPU direct, GPU direct, and GPU Barnes-Hut.
 - Expect near machine-precision agreement when θ is small.
 - Simple hierarchical systems (e.g., one star with a cluster of small bodies) to test center-of-mass approximations.
- **Scaling experiments:**
 - Sweep N from 10^2 to as large as GPU memory allows.
 - Measure wall-clock time per timestep for:
 - CPU direct vs GPU direct vs GPU Barnes-Hut.
 - Identify crossover points where the tree method becomes faster than direct.
- **Accuracy vs speed tradeoff:**
 - Vary θ and monitor:
 - Energy drift over long runs.
 - Qualitative changes in orbit structure.
 - Runtime per timestep.
 - Choose recommended operating regimes (e.g., $\theta \approx 0.5$) where physics is still trustworthy and speedups are significant.

References

-  J. Barnes and P. Hut (1986). A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(6096), 446–449.
-  M. Burtscher and K. Pingali (2011). Efficient CUDA implementation of the tree-based Barnes–Hut N-body algorithm. *GPU Computing Gems: Emerald*, Chap. 6.
-  H.-H. Wu (2024). High-performance CUDA implementation of N-body simulation with Barnes–Hut. Project report. <https://hsin-hung.github.io/N-body-simulation/report.pdf>
-  T. Ventimiglia and K. Wayne. The Barnes–Hut algorithm (online explainer). <https://medium.com/@hsinhungw/optimizing-n-body-simulation-with-barnes-hut-algorithm-and-cuda-c76e78228c28>
-  Wikipedia. Barnes–Hut simulation overview and pseudocode.