

Project Proposal: Barnes-Hut N-Body Simulation

1. The Classic N-Body

The n-body simulation at its core uses an incredibly simple algorithm to simulate the gravitational impact of bodies on each other. The simplest way of approaching this is to go body by body, sum the force vector that every other body applies on it due to gravity, and once you have all the final force vectors step forward.

The force applied on a given object can be described by the following formula, where, for a body i , $\vec{F}$ is the total force vector, G is the universal gravitational constant, m is mass, j is the body being compared to (note that we never compare a body to itself), $\vec{r}$ are the position vectors, and ϵ is a softening parameter, used to prevent bodies from getting incredibly close to each other, at which point their forces begin to approach infinity, and they may “slingshot” away. [1]

$$\vec{F}_i = G m_i \sum_{j \neq i} \frac{m_j (\vec{r}_j - \vec{r}_i)}{|\vec{r}_j - \vec{r}_i|^3 + \epsilon^2}$$

Once the total force has been calculated as shown above for all the bodies, we can use leapfrog integration to ‘step forward’. [2]

This method has a $O(n^2)$ computation time, and we can quickly see that it’s an ideal algorithm for parallelization. However, many algorithms go beyond simply running these in parallel for every core available. Firstly, if we limit our optimization by assigning one body to one core, then given x cores, and n bodies, we only actually get $O((n^2)/x)$ given that n is significantly larger than x . Most modern processors are limited to 24-32 cores, and even specialized hardware only have a few hundred cores, whereas these simulations can have a number of bodies several magnitudes higher.

2. Barnes-Hut Approximation Algorithm

The main idea behind the Barnes-Hut algorithm is recursively dividing the volume being simulated up into cells using an octree (when simulating three dimensions, or a quadtree in two dimensions). An octree is simply a tree data structure where every node has eight children, thus you can essentially envision a cube being divided into 8 parts. This division is continued until every leaf node has 0 or 1 bodies. Each node stores the position and value of its center of mass.

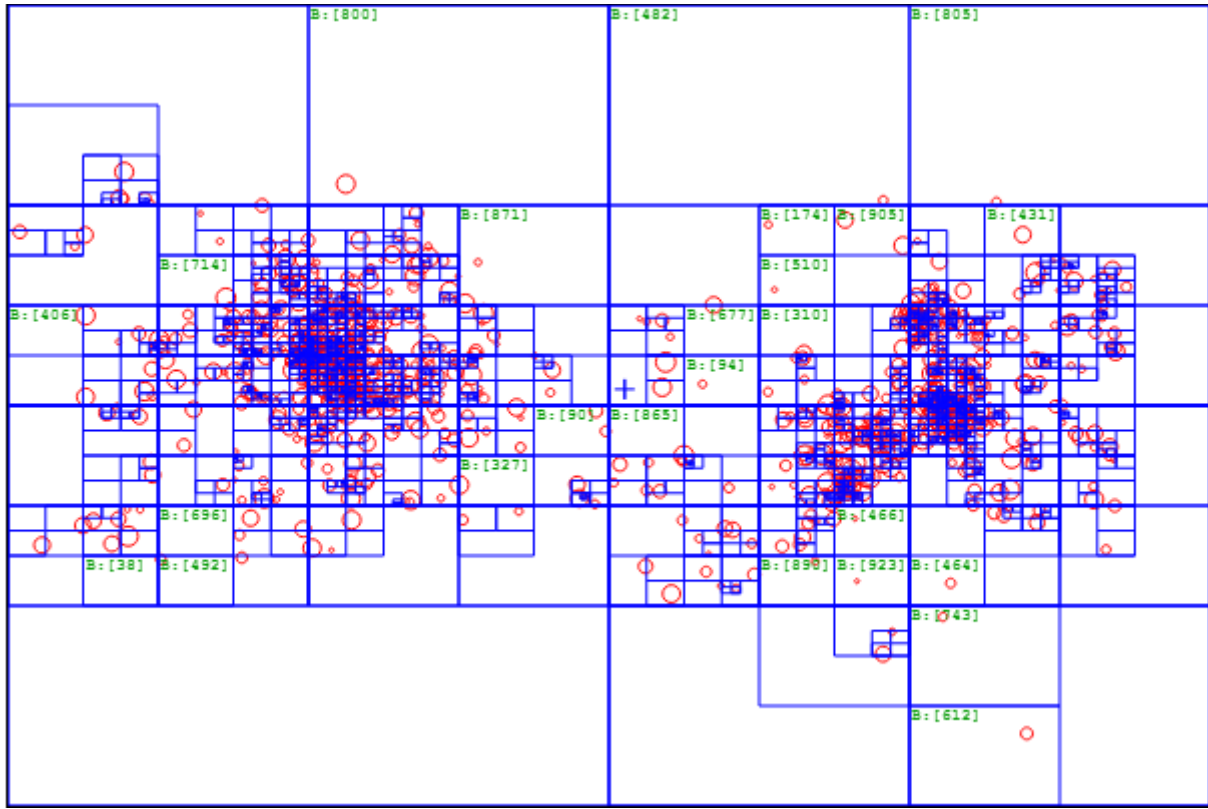


Fig 1. An example of quadtree cells used in a 2 dimensional barnes-hut simulation [3]

To calculate the total force applied on a single body, the tree is traversed in a depth first manner, and whenever the center of a node is further away than a set distance the dividing stops, and the node is treated as a single body. A node is considered far away when the value s/d is less than some predetermined value θ , where s is the length of the side of the node being considered and d is the distance between the body and the node's center of mass. The greater the value of θ the faster the simulation but the lower the accuracy. This continues until the entire tree is accounted for [4]. Using this method the runtime for a serial implementation is reduced from $O(n^2)$ to $O(n \log n)$.

3. Proposed Implementation

I will implement a three dimensional n-body simulation using the Barnes-Hut approximation algorithm while parallelizing the various node traversals to speed up the algorithm.

The user will input a csv, with four Doubles per body, one for each dimension, and one for the mass. At the beginning of each time step, I can construct the octree of a given step using par and pseq [5]. To prevent excessive memory buildup due to thunks, strict evaluation will be used for loading in the values. From there, the easiest way is to process as many bodies in parallel as possible. This again can be accomplished using par and pseq. The final algorithmic complexity should be around $O((n \log n) / x)$ where x is the number of cores available.

Synthetic data is easy to acquire by simply generating random values within a reasonable range. Additionally, values representing the solar system can be used to compare against known ephemeris data to verify accuracy.

References

- [1] “N-Body Simulation,” *Wikipedia*, Jan. 15, 2022. https://en.wikipedia.org/wiki/N-body_simulation (accessed Oct. 18, 2026).
- [2] “Leapfrog Integration,” *Wikipedia*, Apr. 06, 2020. https://en.wikipedia.org/wiki/Leapfrog_integration (accessed Oct. 18, 2026).
- [3] “Barnes-Hut Simulation Overlay,” *Wikimedia.org*, Jan. 03, 2012. https://commons.wikimedia.org/wiki/File:Barnes-Hut_simulation_overlay.png (accessed Oct. 18, 2026).
- [4] Wikipedia Contributors, “Barnes–Hut Simulation,” *Wikipedia*, Oct. 12, 2020. https://en.wikipedia.org/wiki/Barnes%E2%80%93Hut_simulation (accessed Oct. 18, 2026).
- [5] “Par and Seq,” *HaskellWiki*, Sept. 09, 2008. https://wiki.haskell.org/Par_and_seq (accessed Oct. 18, 2026).