Computer Programming on Geosciences

N-body Simulation

Gauss Chang[†]

December 17, 2024

[†] Department of Physics
NATIONAL TAIWAN UNIVERSITY

Outline

1. Background
2. Algorithm
3. Simulation Setup
4. Results
5. Conclusion
6. Discussion
7. Bibliography

Background

Background: Inspiration

Background: N-Body Problem

The n-body problem seeks to predict the motions of a group of celestial objects interacting gravitationally.

It involves determining their interactive forces and true orbital trajectories over time, given their initial positions, velocities, and other properties.

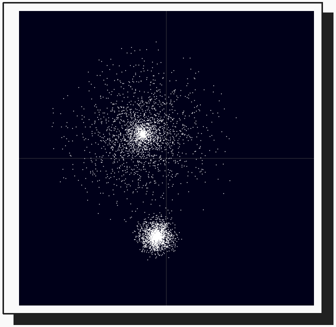

Figure 1: The N body problem.

Background: N-Body Problem

- Classical Mechanics: Focuses on celestial dynamics like planetary motion and star clusters, using Newtonian physics.
- General Relativity: Adds complexity by considering spacetime distortions caused by massive objects.
- Applications: Includes understanding the solar system, star clusters, and galaxy dynamics.
- Simplified Cases: The restricted three-body problem is a notable special case.
- Solving the n-body problem often requires numerical simulations due to the lack of general analytical solutions.

Algorithm

Algorithm: Newton's Law of Gravitation

$$\mathbf{F} = \frac{G m_1 m_2}{|\mathbf{r_1} - \mathbf{r_2}|^3}(\mathbf{r_1} - \mathbf{r_2}) \tag{1}$$

- G: Gravitational constant
- m_1, m_2: Masses of the two bodies
- $\mathbf{r_1}, \mathbf{r_2}$: Position vectors of the bodies
- The force $\mathbf{F}$ exerted on m_1 by m_2 is attractive and along the line connecting the two masses.
- Elastic Collision.

Algorithm: Barnes–Hut Algorithm[1]

- **Motivation:** Direct N-body simulations are $O(N^2)$, which is costly for large N.
- **Key Idea:**
  - Use a hierarchical spatial decomposition (quadtree in 2D, octree in 3D).
  - Group distant particles into cells and approximate their collective influence by a single mass located at their center of mass.
  - Elastic Collision

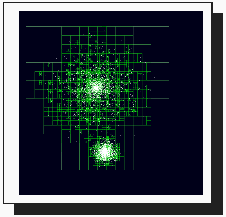
Figure 2: BarnesHut tree.

ALGORITHM: BARNES–HUT ALGORITHM

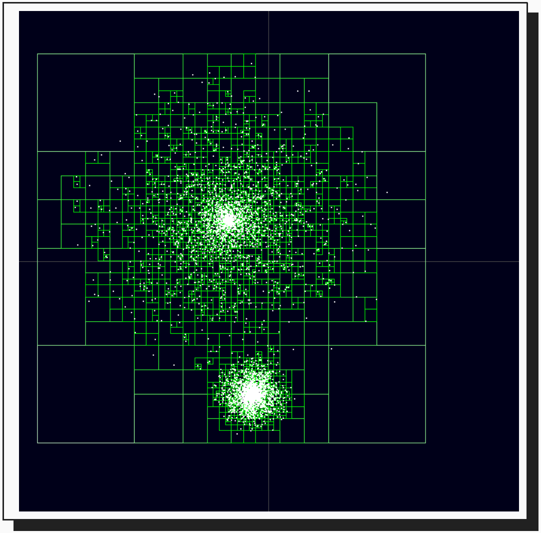
Figure 3: BarnesHut tree (2D).

Algorithm: Barnes–Hut Algorithm

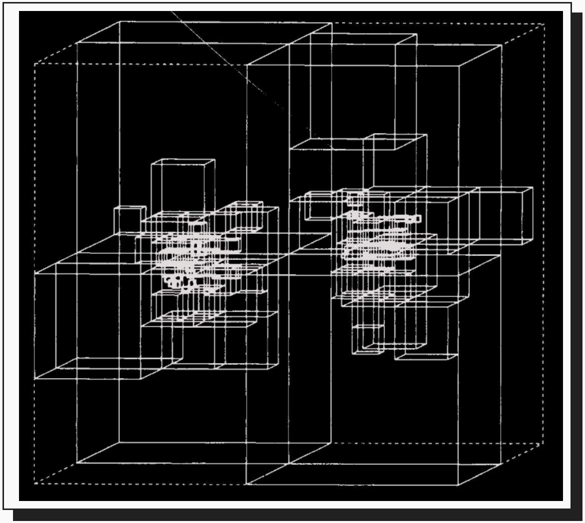

Figure 4: BarnesHut tree (3D).[1]

Algorithm: Barnes–Hut Algorithm

1. **Build the Tree:**
   - Start with the entire simulation domain as the root.
   - Recursively subdivide the domain into smaller cells until each leaf node contains only a few particles.
2. **Compute Mass Properties:**
   - For each node, compute total mass and center of mass of all particles in its subtree.
3. **Force Calculation:**
   - For a given particle, traverse the tree.
   - If a node is sufficiently ”far away,” approximate all its particles by the node’s center of mass.
   - Otherwise, descend into its children for finer resolution.

Algorithm: Direct N-Body vs. Barnes–Hut

- **Direct N-Body Simulation:**
  - Complexity: $O(N^2)$
  - Each particle interacts directly with every other particle.
  - Provides high accuracy but is computationally expensive.
- **Barnes–Hut Approximation:**
  - Complexity: $O(N \log N)$ on average
  - Trades a small loss in accuracy for a significant gain in efficiency.
  - Accuracy can be tuned via the opening angle θ: a smaller θ means more accuracy but higher cost.

Algorithm: Time Integration

- Simple 'Euler method', which updates the position and velocity for a given particle by *timestep* Δt via

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

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

- From the Taylor expansion of the position and its time derivatives at time $t+\Delta t$

$$\mathbf{x_1} = \mathbf{x_0} + \frac{1}{2}\left(\dot{\mathbf{x}}_\mathbf{0} + \dot{\mathbf{x}}_\mathbf{1}\right)\Delta t + \frac{1}{12}\left(\mathbf{a_0} - \mathbf{a_1}\right)\Delta t^2 + O\left(\Delta t^5\right) \tag{4}$$

$$\dot{\mathbf{x}}_\mathbf{1} = \dot{\mathbf{x}}_\mathbf{0} + \frac{1}{2}\left(\mathbf{a_0} + \mathbf{a_1}\right)\Delta t + \frac{1}{12}\left(\dot{\mathbf{a}}_\mathbf{0} - \dot{\mathbf{a}}_\mathbf{1}\right)\Delta t^2 + O\left(\Delta t^5\right) \tag{5}$$

Simulation Setup

Simulation Setup: Initial Condition

Generate N particle randomly.

- Mass is randomly assigned between $1 \sim 2$.
- Position (r_x, r_y, r_z) is randomly assigned between $-\frac{L}{2} \sim \frac{L}{2}$.
- Velocity (v_x, v_y, v_z) is randomly assigned between $-1 \sim 1$.
- Radius (r) is randomly assigned between $0.001 \sim 0.05$.
- No initial acceleration.

Simulation Setup: Initial Condition

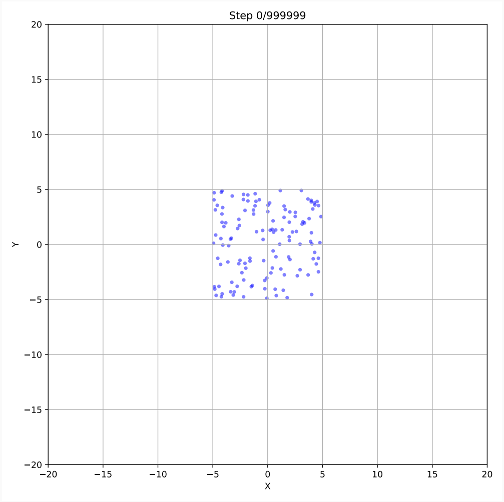

Figure 5: Initial particle distribution.

Results

Results: Metrics for Evaluate Simulation Results

I use the two conservation laws to check if the simulation has broken.

- Conservation of total energy, $\sum E = \sum (T + V)$ of the system.
- Conservation of total angular Momentum, $\sum \mathbf{L}$.

Record $\sum E$ and $\sum \mathbf{L}$ in every simulation step, and plot against time.

Results: Energy v.s. Time

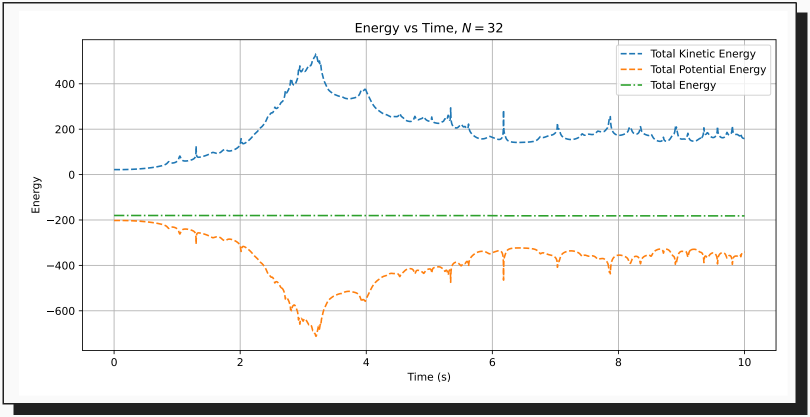

Figure 6: Energy v.s. Time when $n = 32$.

Results: Energy v.s. Time

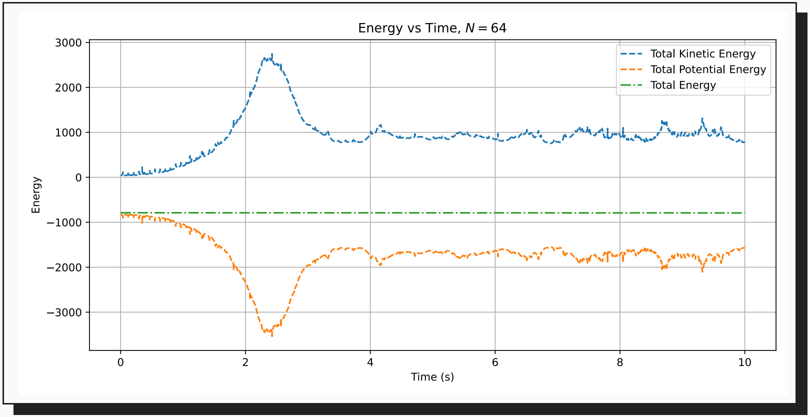

Figure 7: Energy v.s. Time when $n = 64$.

Results: Energy v.s. Time

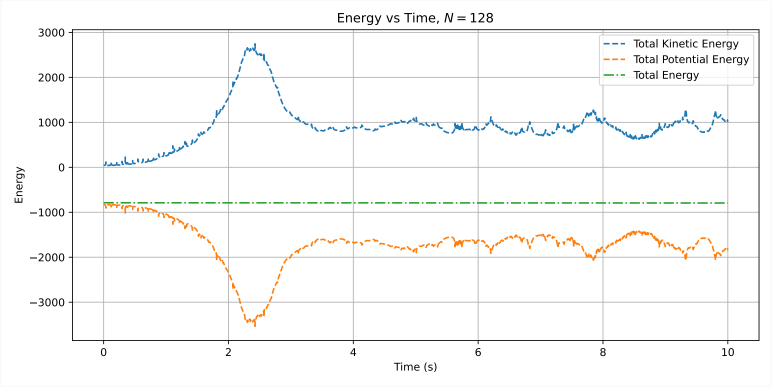

Figure 8: Energy v.s. Time when $n = 128$.

Results: Energy v.s. Time

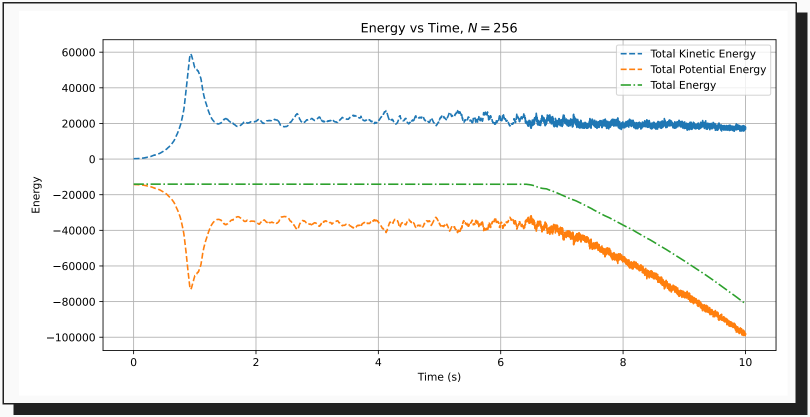

Figure 9: Energy v.s. Time when $n = 256$.

Results: Angular Momentum v.s. Time

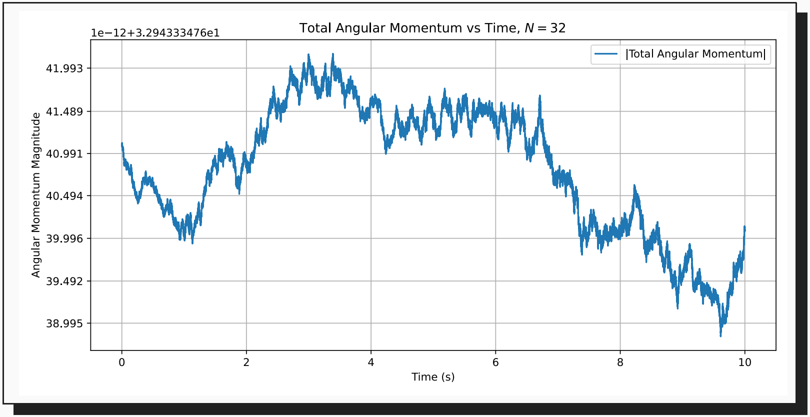

Figure 10: Angular Momentum v.s. Time when $n = 32$.

Results: Angular Momentum v.s. Time

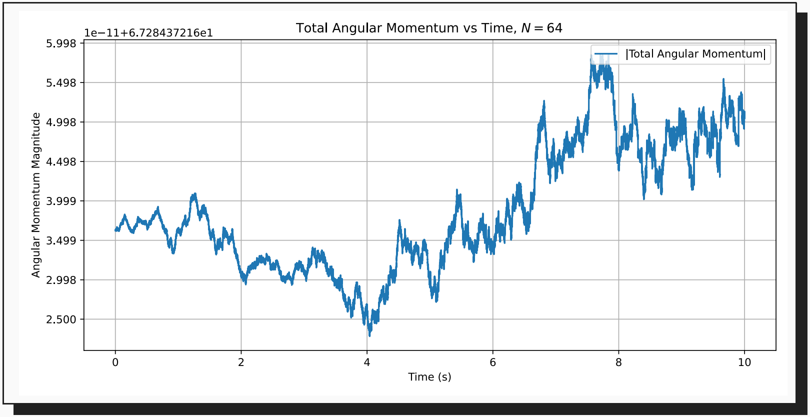

Figure 11: Angular Momentum v.s. Time when $n = 64$.

Results: Angular Momentum v.s. Time

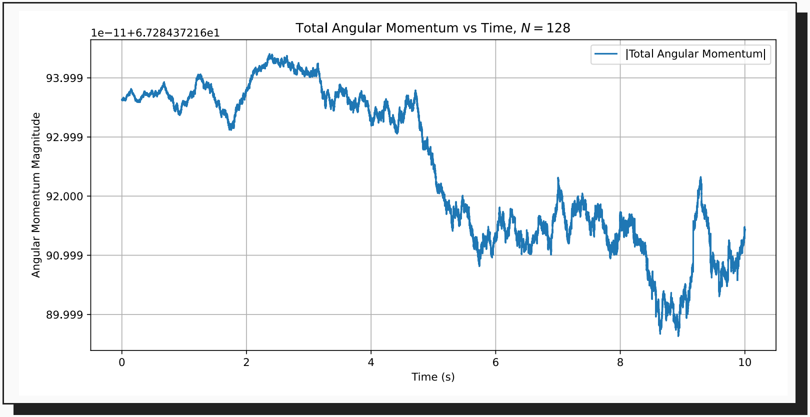

Figure 12: Angular Momentum v.s. Time when $n = 128$.

Results: Angular Momentum v.s. Time

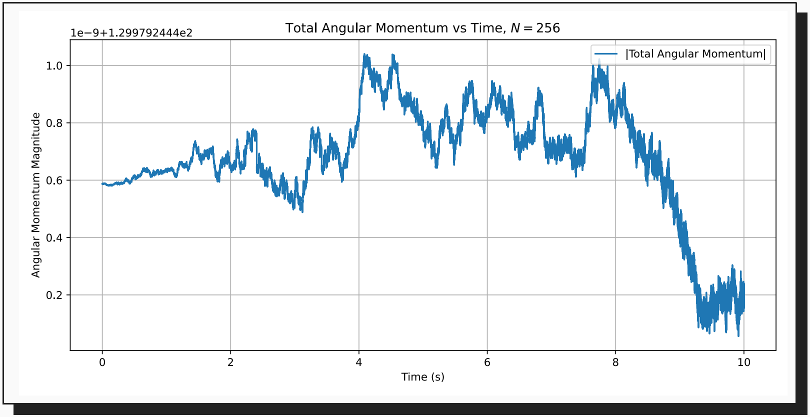

Figure 13: Angular Momentum v.s. Time when $n = 256$.

Results: Animation

`https://www.youtube.com/watch?v=QOyjhbGaQ2A`

Conclusion

Conclusion: Conservation Of Energy

- Energy conservation breaks down for $N \geq 256$.
  - Happens when N is large, a while after simulation.
  - Sometimes after a large amount of collision.
  - Finer time-stepping can temporarily fix the issue.
- Implications:
  - While acceptable for small N, these issue may limit the suitability of the algorithm for high-precision, large N studies requiring strict conservation.

Conclusion: Conservation Of Angular Momentum

- Observed very slight fluctuations in angular momentum across all n:
  - Angular momentum conservation is much more stable than total rnergy.
  - Possible reason is collision does not affect angular momentum.

CONCLUSION: FUTURE IMPROVEMENTS

- Future improvements:
  - Implement adaptive time-stepping to better handle interactions in larger systems.
  - Explore higher precision arithmetic to reduce numerical drift.

Discussion

Discussion: Comparison with Actual Research

- In this project, I managed to push my particle number to $N \leq 256$, but it has already been done 60 years ago.

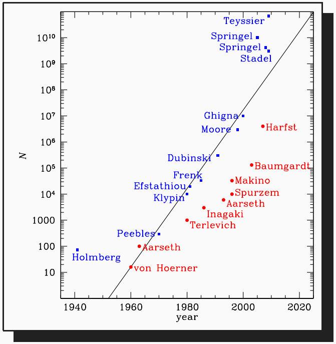

Figure 14: The increase in particle number over the past 80 years.[2]

Discussion: Ways to Accelerate Simulation

For many simulations, such as the **n-body**, multiple nested loops are used. And Python is very slow in looping.

- Use ‘for’ loop instead of ‘while’ loop. ‘for’ loop is slightly faster.
- Use “Numba” decorator.
- Harness PyTorch GPU (not so useful in I/O intensive work.)
- Migrate entire project to C/C++. ✓
- Migrate entire project to Fortran.

Discussion: Ways to Accelerate Simulation

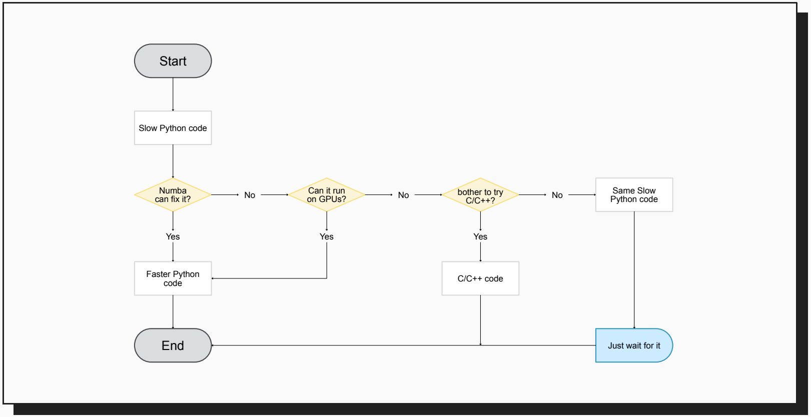

Figure 15: Flow chart if the Python code is slow.

Bibliography

Bibliography I

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

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

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

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

Bibliography II

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

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

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

BIBLIOGRAPHY III

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

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.