Streamlining Simulations of Many Interacting Objects Using Multipole Methods

Aaron T. Hutchins, Christopher F. Kane, Meghan K. Lentz, Patrick Miles, Walter Freeman

ABSTRACT

We are investigating the time-evolution of systems that can be modeled as many particles obeying classical force laws. While these systems can be simulated in principle using very simple algorithms which compute all forces between particles exactly, these algorithms run in $O(N^2)$ time per timestep and become extremely slow for large numbers of particles. We are developing an algorithm based on the rapid multipole method, in which the force from distant clusters of particles is approximated by a multipole expansion, to greatly accelerate these simulations, which we will apply to systems involving solar system formation, fluid dynamics, and crystal growth and defect formation. Here we discuss our current simulation of stellar objects, how it has been optimized, and further optimizations to be investigated.

INTRODUCTION

- **Pair-wise Newtonian calculation is expensive and slow.**
- **A multipole method can be used to approximate the effect of well-separated objects on a given object at a fraction of the computational cost.**
- **Omelyan integration results in fourth-order precision.**

For large numbers of particles the calculation time becomes prohibitive. This prevents us from making visually useful and smooth simulations. To decrease this time, and thus create faster simulations, we can make approximations using a multipole expansion.

The systems we are investigating are governed by central potential relationships for particle separation. We chose to first utilize multipole methods on interacting celestial objects that obey Newton's Law of Universal Gravitation:

$$\mathbf{F}_{21} = -G\frac{m_1 m_2}{|\mathbf{r}_{12}|^2}\hat{\mathbf{r}}_{12}$$

where $\mathbf{F}_{21}$ is the force applied on object 2 exerted by object 1, G is the gravitational constant, m_1 and m_2 are the masses of objects 1 and 2, $|r_{12}| = |r_2 - r_1|$ is the distance between objects 1 and 2, and $\hat{\mathbf{r}}_{12}$ is the unit vector from object 1 to object 2. This force was integrated into both classical and multipole integration methods.

Our integration algorithm is based on the work of Omelyan, Mryglod and Folk.[1] We run our force algorithm to make small position and velocity updates forward and backwards in time, based on derived coefficients. After four velocity and three position updates a full timestep has elapsed. This calculation method provides fourth order precision. We previously investigated two other algorithms, second-order Leapfrog integration (an adaptation of Verlet integration[2]), and fourth-order Forest-Ruth integration.[3] Omelyan was chosen because it has higher order precision than Leapfrog and handles rapid changes in trajectory resulting from near-collisions better than Forest-Ruth.

MULTIPOLE METHOD

- **New data types were constructed to hold planet and region parameters**
- **Method takes small steps forward and backward in time for a net single timestep progression**

All our code is written in C, with C++ operator overloading used to create a Cartesian vector data type. The regions are stored in a large, static array. Each region structure contains information about its children and the objects is contains. Lists of planets are stored in linked lists. Additionally, planets' positions, velocities, accelerations, masses and radii are stored.

Starting with a method proposed by Beatson and Greengaurd[4] we partitioned the three-dimensional space in which our particles exist into regions. If a region is found to be well separated from a particle our multipole method allows the region's effect on the particle to be approximated by the first few terms in the multipole expansion of the particles in that region. This expansion need be calculated only once for each region; once this is done, the approximate effect of that region (which may contain hundreds or thousands of objects) on distant particles can be calculated in only a few operations. To decide what regions were well-separated a displacement parameter α on the characteristic scale of a region side length was adjusted and the accuracy of the resulting simulations were compared. A particle is well-separated if the distance to the center of mass of that region, divided by the width of that region, is greater than α. The method farther partitions in descending levels (see *Figure II*). Beginning with the first level (the trivial case where one region contains all space), the number of regions in the *n*'th level are 8^n.

MULTIPOLE METHOD (CONT.)

Figure I (Right) Steps in force evaluation

For each region our algorithm decides whether or not to make a multipole approximation of its effect on a given object. If a region is found to only contain one object the force is calculated directly; otherwise the region is checked to see if it is well-separated from the object. If the region is well-separated, a multipole approximation is applied. If the region is not well-separated from the particle, it is partitioned further, descending down the region tree until reaching regions that are well-separated, contain only one particle, or lie at the bottom of the tree. At the bottom of the tree the effects from each planet in the region are calculated pair-wise.

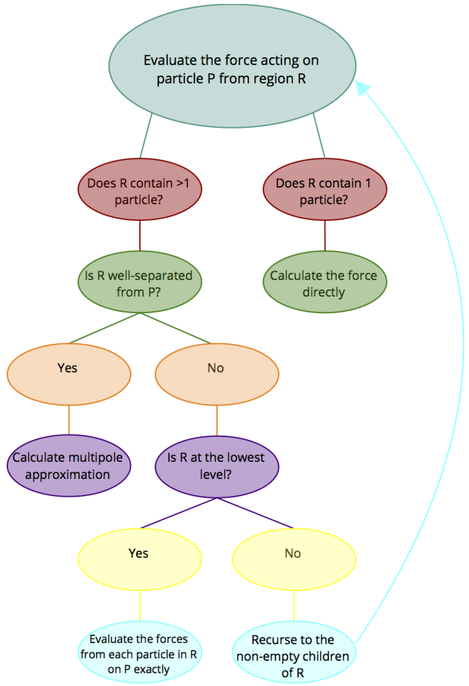
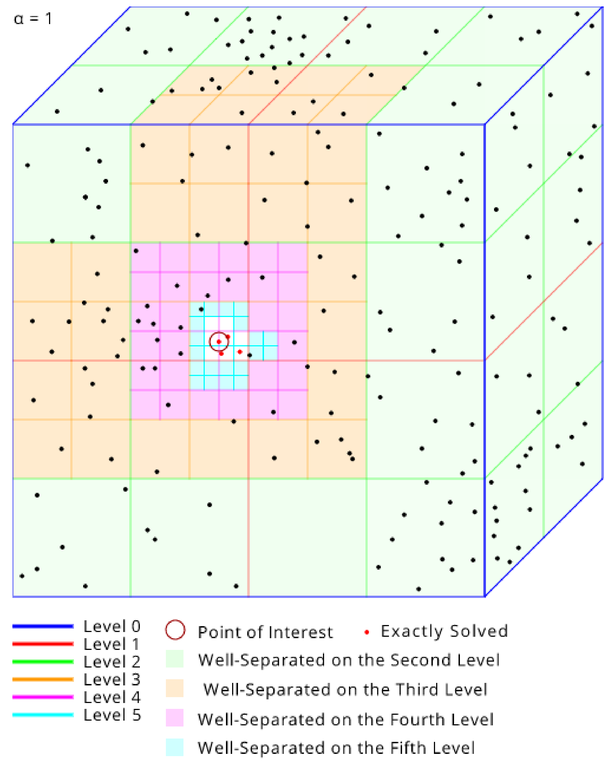


Figure II (Below) A model showing the subdivision of regions through six levels based on our multipole expansion



RESULTS

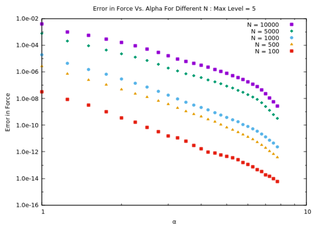


Figure III (Above)

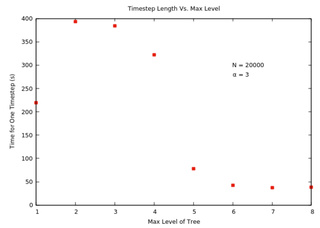


Figure IV (Above)

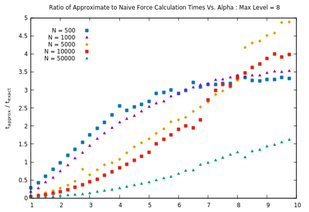


Figure V (Above)

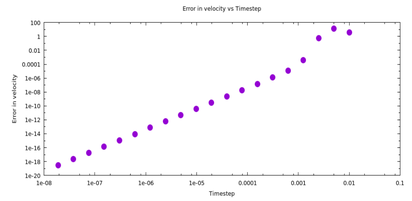


Figure VI (Above)

Figure III shows that if alpha -> ∞ , the error in the force calculation goes to zero. This in turn tells us that our algorithm reduces to the naïve pairwise force evaluation method. *Figure IV* shows that for large N, the algorithm is much more efficient if more levels are used. Additionally, we see that there is a speed gain cutoff as max level increases. Notice that using one level is faster until five levels are used. For one level, all regions are not well separated from the beginning and we jump right to using the naïve method. For higher levels, the naïve method is still used in the majority of the force calculations, but the algorithm must waste time traversing down the tree to find this out. *Figure V* shows that our algorithm is more efficient as N gets very large. We see it is slower than the naïve method if N is small for small values of alpha. This is evidence that using a complex algorithm to do a simple task is not always as efficient as using a simple algorithm for the same task. *Figure VI* shows that our code does in fact exhibit 4th-order scaling, as expected with the Omelyan integrator.

FUTURE WORK

- **Incorporating higher moments will give us more accurate results.**
- **Regions can be dynamically allocated to reduce unused memory.**
- **OpenMP will be incorporated to make efficient use of multicore systems.**
- **Future simulations include more extensive accretion disk studies and the models of molecular dynamics.**

The results shown here are preliminary, and serve as a proof of concept of the algorithm and a starting implementation. Going forward, we will add the analysis of higher moments in the multipole expansion for each region. This will allow us to use lower values of α while retaining accuracy. Likewise, switching to a dynamically-allocated region tree will drastically reduce memory use and allow deeper partitioning of densely-populated regions without wasting memory allocating regions that contain no planets. Additionally, adapting the code to use OpenMP will allow efficient use of multicore CPU's which are increasingly prevalent.

We anticipate that using this algorithm we can extract meaningful physics from simulations of 10^5 particles. The discussion and demonstration here discuss Newtonian gravity; the original conceit of this project was to use this algorithm to create a model of the early solar system and the accretion of planetesimals into planets. However, any central force can be addressed in this way; this method is well-suited to simulations of molecular dynamics, which we will investigate and apply to investigate crystal structure, fluid flow, and thermodynamics.

REFERENCES

[1] I. P. Omelyan, M. Mryglod, and R. Folk. "Optimized Forest-Ruth- and Suzuki-like Algorithms for Integration of Motion in Many-Body Systems." Cornell University Library. 2001. Web.

[2] Verlet, Loop. "Computer 'Experiments' on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules." Physical Review. 159: 98-103. 1967.

[3] Young, Peter. "The leapfrog method and other "symplectic" algorithms for integrating Newton's laws of motion." University of California, Santa Cruz, Young Research Site. 2014.

[4] Beatson, Rick, and Leslie Greengard. "A Short Course on Fast Multipole Methods." New York University Math Department Website. Oxford University Press, 1997. Web.