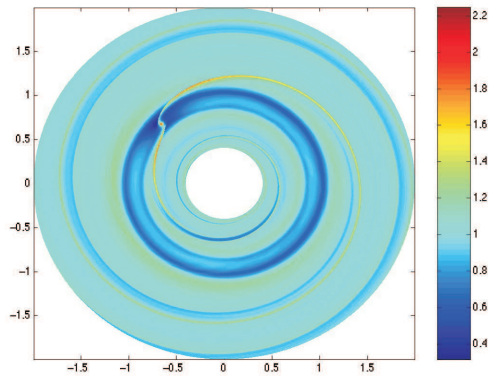
A fast algorithm for disk self-gravity interactions in proto-planet simulations

Matthew Buoni, mjbuoni@gmail.com
Shengtai Li, sli@lanl.gov

Proto-planet simulations seek to study and understand the physical processes involved in planet and solar system formation. There has been a growing interest in recent years in such simulations due to discoveries of 210 extra-solar planets (the first confirmed in 1988) with properties very different from our own. In the simulations studied here, planets are modeled as discrete point masses embedded in a two-dimensional disk of gas revolving around a central star. Because of the long time scales involved in planet formation, well-resolved simulations take days or weeks to run even on the fastest parallel computers. Computationally, one of the main challenges is calculating gravitational interactions of the gas disk with itself (disk self-gravity) efficiently. If performed directly, this calculation takes orders of magnitude longer than all other calculations combined. In this paper, we describe our algorithm that accelerates the calculation of these interactions. We implement a parallel tree-code specifically optimized and tailored to the disk geometry used in our simulations [1]. For a typical simulation, we find that our algorithm gives a speedup of 300 - 500 times compared to direct summation. Also, we describe the interaction forces and our model used to approximate and compute them efficiently.

Algorithm

We developed a parallel tree-code, specifically tailored and optimized for our disk geometry, to calculate disk self-gravity forces [1],[2]. Tree-codes accelerate force calculations by replacing particle-particle interactions with particle-cluster interactions, where a cluster is a group of particles in close proximity to one another. This reduces the number of interactions from $O(N^2)$ to $O(N\log N)$, since each particle interacts with $O(\log N)$ clusters. One of the major differences between a general parallel tree-code and our optimized code is that our tree structure is computed once initially and is unchanging in time. We can do this because our particle positions are fixed at grid cell centers, which are unchanging in time. Parallelization is done for a distributed memory architecture, with data distributed in concentric bands of equal numbers of grid points. Because grid points are more densely packed on the inner part of the disk, optimal load balancing is not automatic. A master-slave scheme is used to achieve nearly ideal load balancing even if some processors compute faster than others.

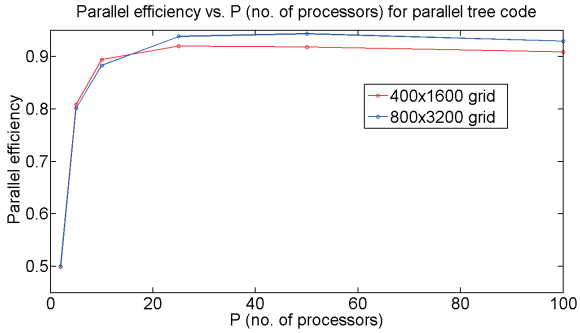


Snapshot of gas density during a simulation. Notice the low density ring near the planet (upper-left part of figure), which is due to gravitational collapse of mass onto the planet.

We tested the performance of our tree-code, both serial and parallel. Our serial tree-code is checked to have the proper $O(N\log N)$ complexity scaling (where N is the number of spatial grid points); this is important since a parallel algorithm is only as efficient as its corresponding serial algorithm. Parallel speedup is then checked and parallel efficiency computed. For the grid sizes of interest (400×1600 and 800×3200), we find the parallel efficiency to be greater than

A fast algorithm for disk self-gravity interactions in proto-planet simulations

90%, which is as good or better than any of the results given in the literature. Finally, when we compared our algorithm to direct summation, we find that it performs 300 and 500 times faster (for a 800×3200 grid), although it still accounts for around 90 % of the CPU time.



Measured parallel efficiency for parallel tree-code as a function of number of processors, P. Notice that efficiency is more than 90 % except for very small P (due to master-slave scheme) and is higher for larger grids (which have a higher computation to communication ratio).

Modeling interaction forces

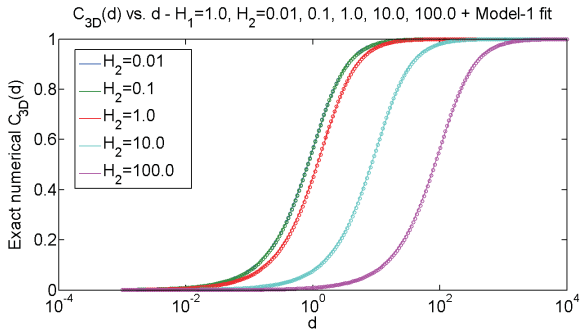
The interaction force between particles in our simulation is due to gravity; however, simply using $F(d) = \frac{-1}{d^2}$ is not desirable for the following two reasons. First, numerical singularities would occur when summing forces due to nearby particles. For this reason, a smoothing factor, ε, is usually introduced, i.e. $F(d) = \frac{-1}{d^2+\varepsilon^2}$. Second, even though we are performing two-dimensional simulations, mass is not located sharply in the plane of the disk, but is distributed normally in the third (height) dimension. Thus, the force between two points in the disk separated by a distance, d, is also a function of the scale heights of the disk at the two points, H_1 and H_2. It is given exactly by $F_{3D}(d) = C_{3D}(d)F(d)$, where $F(d)$ is the two-dimensional force already given and $C_{3D}(d)$ is a 3D correction factor which is shown to be

$$C_{3D}(d) = \frac{d^3}{2\pi H_1 H_2} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{e^{\frac{-z_1^2}{2H_1^2}} e^{\frac{-z_2^2}{2H_2^2}} dz_1 dz_2}{\left(d^2 + (z_1 - z_2)^2\right)^{\frac{3}{2}}}. \quad (1)$$

$C_{3D}(d)$ is very expensive to compute on the fly; also, pre-computing and storing it in main memory is prohibitively slow and uses excessive storage. A better strategy is to approximate $C_{3D}(d)$ with an accurate and computationally inexpensive model. We implement such a strategy which uses a model of the form

$$C_{3D}^{(\text{model})}(d) = \frac{x^p + x^{2p}}{2 + x^p + x^{2p}}, \quad (2)$$

where $x = \frac{d}{0.83(H_1^2+H_2^2)^{0.5}}$ and $p = 0.91$. Our model gives less than 1% error and requires computing just one **power** and one **square root** function, which is done on the fly. We plot $C_{3D}(d)$ and our model overtop in the figure below.



Three-dimensional correction factor for interaction forces used in disk self-gravity calculations. Our model is plotted overtop the exact data for comparison.

Acknowledgements

Los Alamos Report LA-UR-06-8030. This work was carried out under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396, and supported by DOE Office of Science Advanced Scientific Computing Research (ASCR) Program in Applied Mathematics Research.

References

[1] J. Dubinski, A parallel tree code, *New Astronomy*, **1**, 133-147 (1996).

[2] P. Miocchi, R. Capuzzo-Dolcetta, An efficient parallel tree-code for the simulation of self-gravitating systems, *Astronomy and Astrophysics*, **382**, 758-767 (2002).