Fast Multipole Methods for the Two Dimensional N Body Problem

Zheng Li[1]

[1]*Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA, 02139*
zhli@mit.edu

(Dated: December 8, 2015)

In this work, we implement both serial and parallel fast multipole methods for the two dimensional N body problem. For the purpose of comparison, the Barnes-Hut algorithm and a direct method are also implemented. The evolution of galaxies are presented as a demonstration.

INTRODUCTION

The N body problem arises in the situation where it is desired to evaluate the pairwise interactions in a large group of particles and that, provided with the initial states, to compute the position and momentum of each of the particles at a certain time. One example of the N body problem could be to simulate the evolution of a galaxy in its own gravitational field. Another example could be to model the behaviors of plasma by computing the Coulomb interactions among the ions and electrons.

Naively, it costs $O(N^2)$ to compute the pairwise interactions directly and precisely. However, if some target particles are well separated from a group of source particles, we can approximate the potential field from the sources by a Laurent series centered at a certain point within the area of the sources. This is called a multipole expansion, which is equivalent to project the randomly distributed sources into a series of canonical configurations. In this scenario, we can reduce the cost by eliminating the computation of the contribution of each single source.

Barnes and Hut utilize the multipole expansion method, and successfully reduce the cost to $O(NlogN)$. [1] In the Barnes-Hut algorithm, the space is represented by a hierarchical tree which is constructed by dividing the space into subspace recursively and self-similarly. The multipole expansion coefficients of each node are computed at each level when constructing the tree. After that, for each particle, a traversal of the tree from its root is performed. If a node is well separated from the target particle, the contribution of the potential from the node is evaluated by its multipole expansion and it is not necessary to go to the next level. Otherwise, we visit the children of the node. If a leaf node is visited and it is still not well separated from the particle, the contribution of the node is computed directly. In our process, the condition of well separation is that the distance from the target to the center of the node is larger than 1.5 $\times$ the size of the node.

The fast multipole method (FMM) is also based on a hierarchical tree and the multipole expansion of each node, but it is more sophisticated and the cost is further reduced to $O(N)$. [2] In the FMM, firstly at the leaf nodes, the multipole expansion coefficients are computed directly. Then by a upward traversal, the multipole expansion coefficients of a node are computed by shifting and adding the multipole expansions of its children. (Multipole to Multipole, M2M) Afterward, a downward traversal from the root is performed. For each target node, we find the nodes that are well separated from the target at the same level without considering the children of the well separated nodes at the previous level. Then their multipole expansions are transformed into a local expansion (Multipole to Local, M2L) and add to the existed local expansion of the target node. The local expansion contains the contributions of all the well separated nodes and can be passed to the next level by shifting from the center of the parent to the centers of the children. (Local to Local, L2L) Inside the leaf nodes, the potentials of the particles are evaluated by adding the local expansions and the direct interactions from the nearest leaf nodes.

It is worth noting that for both the Barnes-Hut algorithm and the FMM, the relative errors are well controlled by the order of the multipole expansion and the condition of well separation .

It is straightforward to compute the evolution as long as the potential of each particle can be evaluated. We compute the potentials near the particle to obtain the negative local gradient, which is just the force field. Using Newton's second law, the change of the momentum as well as the position of each particle can be easily obtained.

NUMERICAL RESULTS

In this section, we perform some results from the numerical experiments in two dimension. The experiments are executed on an Intel Xeon E5-1650 v2@3.7 GHz machine, and the environment is Julia 0.4.1 unless there is a specific indication.

Relative Errors. We first compute the relative errors of the Barnes-Hut algorithm and the FMM versus the orders of the multiple expansions p with the condition of well separation not varied. The relative error is

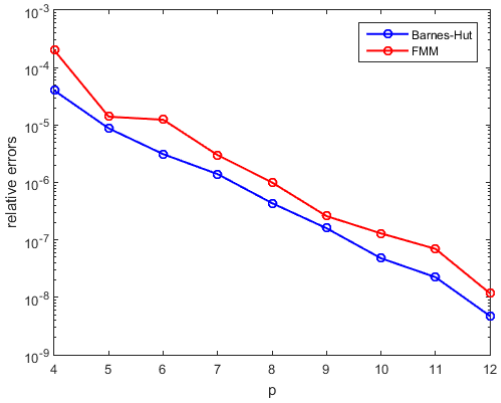

FIG. 1: the relative errors versus the order of the multipole expansions. $N = 500$.

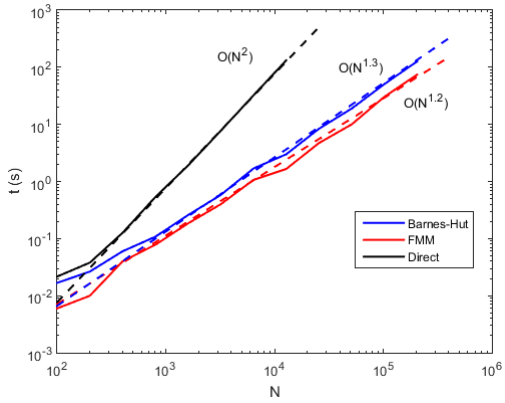

FIG. 2: the execution time of the three methods versus the problem sizes. p=8.

defined as follows.

$$\epsilon_{BH/FMM} = max \left| \frac{\phi_{BH/FMM} - \phi_{dir}}{\phi_{dir}} \right| \qquad (1)$$

where *dir* indicates the direct method which is considered to be precise. The size of the problem is fixed at $N = 500$.

In Fig. 1, the exponential relations of the relative errors and p are clearly observed. This phenomenon is consistent with the work of Greengard *et al.* [2] We also notice that the Barnes-Hut algorithm always performs better than the FMM in terms of the relative error at the same p. We believe that the difference mainly comes from the M2L step, where the extra truncation, besides the multipole expansion, happens.

Execution Time. We present the execution time of the three methods in Fig. 2. For all the investigated problem sizes, we have both the FMM and the Barnes-Hut algorithm faster than the direct method, and the FMM is slightly faster than the Barnes-Hut algorithm. The FMM and the Barnes-Hut algorithm both show approximately O(N) time cost, while the direct method is approximately O(N^2), as expected.

Linear fits show that the FMM and the Barnes-Hut algorithm have the cost of O($N^{1.2}$) and O($N^{1.3}$), respectively. Both are higher than the theoretical values, which are O(N) and O($NlogN$). The extra cost, especially for the FMM, may come from complex function calls and deep recursions.

Parallel Speedup. The FMM is also parallelized for 2 and 4 cores. The parallelization is based on the division of the space. For example, in the case of 4 cores, the data of the whole tree is sent to all the processors, but each processor only operates the FMM for one child of the root. It is obvious that the M2M, the L2L, and the evaluation of the potentials can be achieved in an embarrassingly parallel scenario. However, the data movement is required for the M2L, which calls for the M2M results from all the cores. Since we find the downward traversal is much more time consuming than the tree construction and the M2M, these two steps are still executed in serial to reduce the data movement.

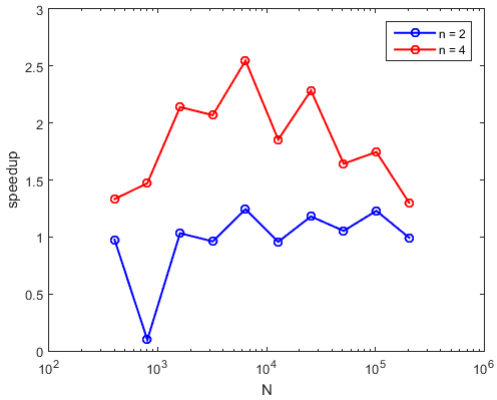

FIG. 3: the speedup of the FMM from the parallelization versus the problem sizes. Number of CPU cores $n = 2, 4$

The speedup compared to the serial code is presented in Fig.3. Basically, the parallelization of 2 cores does not have a significant advantage, while a speedup of approximate 2 is observed for 4 cores. We find the overheads mainly come from the data movement before and after the execution, as well as the serial M2M and the tree construction steps.

Conclusively, the Barnes-Hut algorithm and the FMM both have good error bounds, and their cost scale as O($N^{1+\delta}$), which is much better than the direct method. The FMM, in our investigated range, is always better than the Barnes-Hut algorithm, and can be further improved by parallelization.

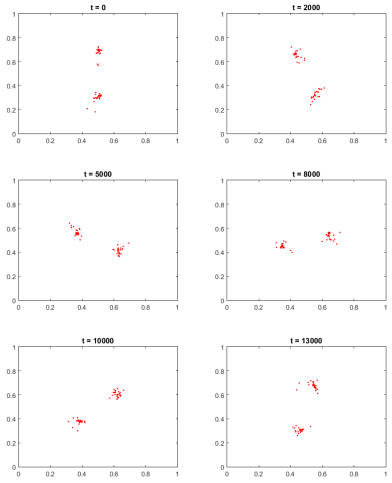

FIG. 4: the evolution of the two galaxies. $v = 3$. $\delta_t = 10^{-5}$.

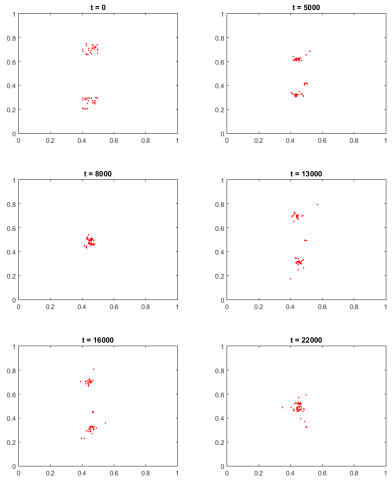

FIG. 5: the evolution of the two galaxies. $v = 0.1$. $\delta_t = 10^{-5}$.

EVOLUTION OF TWO GALAXIES: A DEMONSTRATION

As a demonstration of the N body problem, we present a model based on the FMM to simulate the evolution of two galaxies. In the model, each galaxy has 25 stars. The initial velocities of the two galaxies are opposite and approximately vertical to the line connecting the two galaxies.

The velocity, v, is tuned here and we clearly observe different behaviors of the galaxies. When v is large and the centrifugal force balances the gravitation, (Fig. 4) the galaxies basically keep rotating with their distance unchanged. Otherwise, when v is small, (Fig. 5) the galaxies are similar to a dipole system. The distance of them oscillates. When v is in the intermediate regime, Fig. 6) the galaxies rotate with each other and the distance also changes periodically over time. In each cycle, they seem to 'merge' at some point. However, since there is no damping in this system, they will still split.

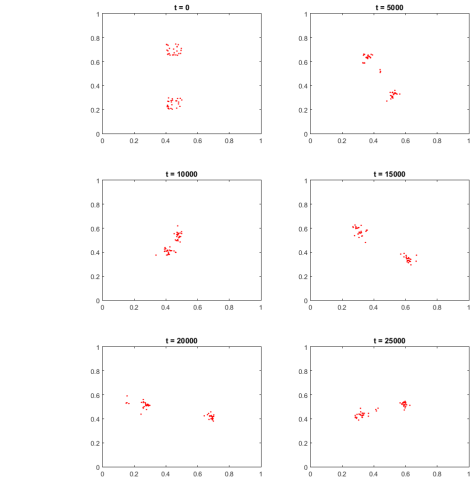

FIG. 6: the evolution of the two galaxies. $v = 1.8$. $\delta_t = 10^{-5}$.

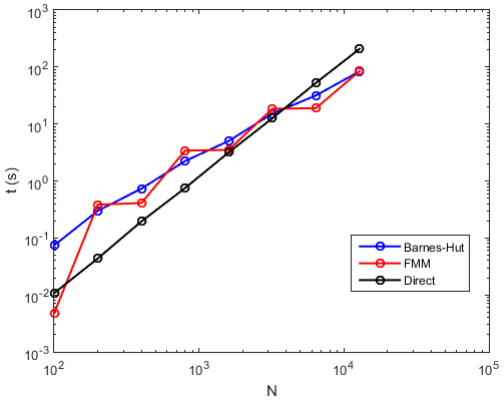

FIG. 7: the execution time of the three methods in MATLAB.

SUPPLEMENTARY MATERIALS

Implementation in MATLAB. We also implement the same algorithms in MATLAB R2015a, based on the MATLAB 'handle class'. However, the speeds are extremely slow (at least 50 × slower than Julia) for both the Barnes-Hut algorithm and the FMM. Also, the scaling of $O(N)$ cannot be clearly observed. This should be somehow expected since MATLAB is well known to have large overheads when executing function calls and recursions.

ACKNOWLEDGMENT

Z. L. thanks Prof. Alan Edelman for his thought-provoking 18.337 lectures and very helpful advice on the parallelization. Z. L. also thanks Dr. Alex Townsend for delivering the outstanding 18.336 lectures and the helpful discussion on the project.

[1] A hierarchical O($NlogN$) force-calculation algorithm
J. Barnes and P. Hut, Nature, **324**, 446 (1986).
[2] A short course on fast multipole methods
R. Beatson and L. Greengard