

The Fast Multipole Method

This method for solving the N -body problem in time of $\mathcal{O}(N)$ was introduced by L. Greengard and V. Rokhlin, *J. Comput. Phys.*, **73**, 325 (1987).

The tree-code algorithms scale like $\mathcal{O}(N \log N)$. Because $\log N$ increases very slowly with N , for example $\log_{10} 10^9 = 9$, the difference between $\mathcal{O}(N \log N)$ and $\mathcal{O}(N)$ is not dramatic, unlike the difference between $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$. In practical applications, tree code algorithms are generally sufficiently efficient, and not much is gained by using the fast multipole method. Nevertheless, the reduction from $\mathcal{O}(N \log N)$ behavior to $\mathcal{O}(N)$ is remarkable theoretically: the fast multipole method has been named one of the “Top Ten Algorithms of the Twentieth Century” in a recent issue of *Computers in Science and Engineering*.

The duality principle

The fast multipole method is based on what the authors call a *duality principle*. This gives a relation between multipole expansions about two different points in space.

Consider two clusters of particles, one in a box called A whose center is taken to be the origin $z = 0$ in the complex plane, and the other in a box called B whose center is located at point z_B .

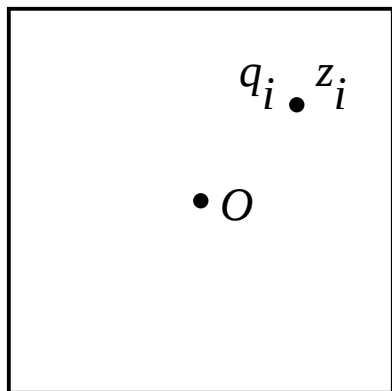
In the tree-code algorithm, we computed the multipole moments

$$a_0 = \sum_{i=1}^{N_A} q_i, \quad a_1 = \sum_{i=1}^{N_A} q_i z_i, \quad \dots$$

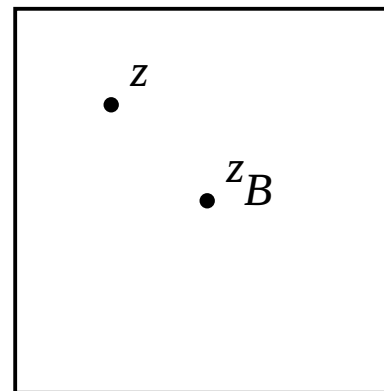
about the center of box A due to the N_A charges q_i located at positions z_i inside box A . The potential and field due to these charges at point z in box B where then given by

$$U(z) = a_0 \log(z) - \frac{a_1}{z} + \dots, \quad E(z) = \frac{a_1}{z^2} + \dots$$

The effects of the N_A charges in box A on the charges in box B are approximated by this *far field* multipole expansion.



Box A



Box B

Greengard and Rokhlin pointed out that there is a different *local field* multipole expansion for $U(z)$ and $E(z)$. Recall the derivation of the far field expansion:

$$U(z) = \sum_{i=1}^{N_A} q_i \log(z - z_i) = a_0 - \frac{a_1}{z} + \dots$$

The potential can be approximated in a different way as follows:

$$\begin{aligned} q_i \log(z - z_i) &= q_i \log[(z - z_B) + (z_B - z_i)] \\ &= q_i \log(z_B - z_i) + q_i \log\left(1 + \frac{z - z_B}{z_B - z_i}\right) \\ &= q_i \log(z_B) + q_i \log\left(1 - \frac{z_i}{z_B}\right) + q_i \left(\frac{z - z_B}{z_B - z_i}\right) + \dots \\ &= q_i \log(z_B) - \frac{q_i z_i}{z_B} + \dots + (z - z_B) \left(\frac{q_i}{z_B} + \frac{q_i z_i}{z_B^2}\right) + \dots \end{aligned}$$

Note that z_B is the displacement between the centers of A and B , which is assumed to be large compared with $z - z_B$ (the displacement between the center of B and a point Z inside B) as well as z_i (the displacement of a charge inside A from the center of A at $z = 0$). The expansion above is called a *near field* expansion because it involves z_B the location of the center of the box B which also contains the point z of interest. This local field expansion can be written

$$U(z) = b_0 + b_1(z - z_B) + \dots, \quad E(z) = \frac{dU}{dz} = b_1 + \dots$$

where

$$b_0 = a_0 \log(z_B) - \frac{a_1}{z_B}, \quad b_1 = \left(\frac{a_0}{z_B} + \frac{a_1}{z_B^2} \right).$$

Note that the local field expansion coefficients b_i can be obtained from the far field multipole moments a_i . This has been derived here for b_0 and b_1 , but the relation can be generalized to higher coefficients by retaining higher order terms in the Taylor series expansions. The relations are not simple, but they can be derived in principle to any order.

Merging and shifting expansions

In the tree-code algorithm, a hierarchy of boxes is constructed in the form of a quadtree: each box is subdivided into four children. The duality principle can be used to speed up the calculation of the multipole moments in the boxes of the quadtree.

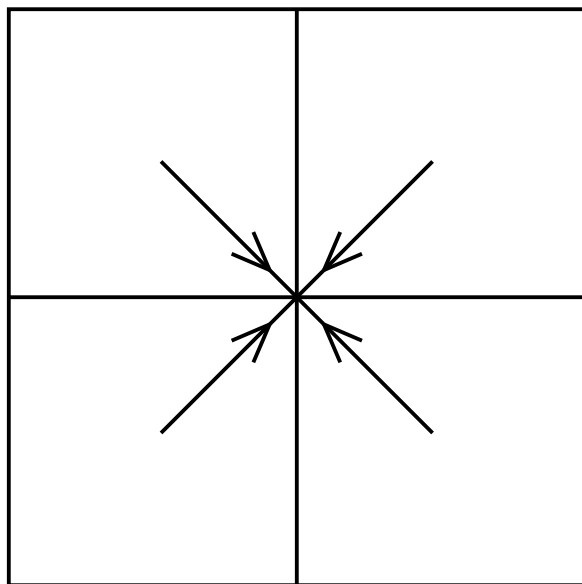
Suppose the multipole moments have been computed for the four children of a given parent box. Let the center of the parent box is taken to be at $z = 0$, and c_k is the position of the center of child k relative to its parent. The net charge and dipole moment of the 4 child boxes are given by

$$a_{0_k} = \sum q_{i_k}, \quad a_{1_k} = \sum q_{i_k}(z_{i_k} - c_k),$$

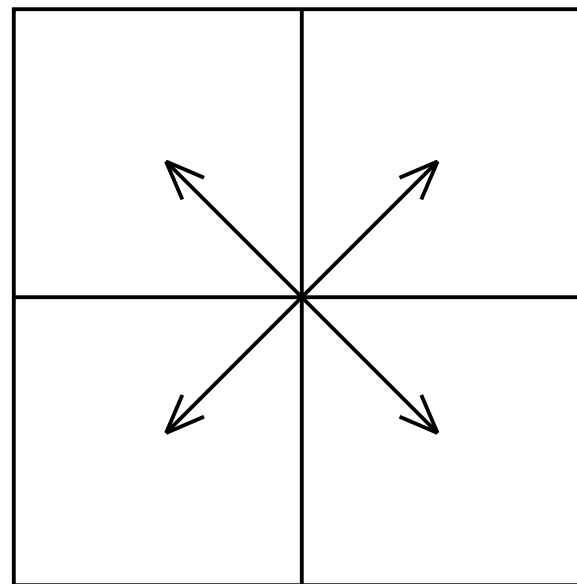
where q_{i_k} and z_{i_k} are the charges in box k and their positions relative to the center $z = 0$ of the parent box. The charge and dipole moment of the parent box are then given by

$$a_0 = \sum_{k=1}^4 a_{0_k}, \quad a_1 = \sum_{k=1}^4 [a_{1_k} + a_{0_k} c_k].$$

This *merging expansion* can be used to avoid computing the multipole coefficients at the parent level if this has already been done for the children!



Merging Expansion



Shifting Expansion

The second important result concerns local expansions. Suppose that the local expansion coefficients has been computed for the parent box. A *shifting expansion* can then be used to compute the local expansion coefficients for all 4 children of the parent. For example, we have taken the center of the parent box to be $z_B = 0$. The local field expansion for the potential due to charges in a distant box else where is then given by

$$U(z) = b_0 + b_1 z .$$

The local expansion for each child k of the parent is then given by *shifting* the center of expansion:

$$U_k(z) = (b_0 + b_1 c_k) + b_1 (z - c_k) .$$

This shifting expansion can be used to avoid computing the local field coefficients for the children if it has already been done for the parents at any given level!

Outline of the fast multipole algorithm

- In the tree-code algorithm, boxes were constructed starting at the coarsest level $\ell = 0$ and proceeding to finer levels. At each level, multipole coefficients were computed for each box. In the fast multipole algorithm, we begin at the finest level of boxes: multipole coefficients are computed for all boxes at this level, which requires time of $\mathcal{O}(N)$. Multipole moments at all parent levels are computed using the merging expansion, which requires time of $\mathcal{O}(N)$ because in a tree of depth $\log N$ there are only of $\mathcal{O}(N)$ boxes.
- In the tree-code algorithm, boxes at a given level were iterated over: for each box, the interaction list was constructed and the multipole moment of the box was interacted with all of the charges in the interaction list. In the fast multipole algorithm, the far field expansion is converted into a *local* expansion about the center of each box in the interaction list, which once again requires time of $\mathcal{O}(N)$.
- After the two steps above, there is a local expansion about the center of each box at each level. Beginning at the coarsest level 0, these expansions are *shifted* to the childrens' level and added to the childrens' local expansions. At the finest level of subdivision, a single local expansion will have been created in each box which describes the interaction of *all* charges outside the near neighborhood. This procedure once again requires time of $\mathcal{O}(N)$.
- The interactions are now evaluated at the finest level:
 - The cumulative local expansion in each box is interacted with all the charges in that box. This requires time of $\mathcal{O}(N)$.
 - The near neighbor interactions are evaluated by interacting each charge with all of the other charges in its near neighborhood. This also requires time of $\mathcal{O}(N)$.

It is clear from this whole procedure will scale like $\mathcal{O}(N)$. It does however require more storage than the tree-code procedure. In the tree-code algorithm, boxes were considered one at a time and so the multipole coefficients did not need to be stored. In the fast multipole method, the merging and shifting steps require that the multipole

moments and local expansion coefficients be stored.

A Fast Multipole Code in 2-D

The fast multipole algorithm can be implemented with a relatively simple code for a system of charges in 2-D, and compared with the tree-code program