JOURNAL OF COMPUTATIONAL PHYSICS **143,** 29–48 (1998)
ARTICLE NO. CP985949

A Comparison between the Fast Multipole Algorithm and the Tree-Code to Evaluate Gravitational Forces in 3-D

R. Capuzzo-Dolcetta* and P. Miocchi†

**Istituto Astronomico, Università "La Sapienza," Rome, Italy, and †Dipartimento di Fisica, Università "La Sapienza," Rome, Italy*
E-mail: dolcetta@astrmb.rm.astro.it

Received March 24, 1997; revised February 10, 1998

We present tests of comparison between our versions of the Fast Multipole Algorithm (FMA) and the tree-code to evaluate gravitational forces in particle systems. We have optimized Greengard's original version of FMA allowing for a more efficient criterion of *well-separation* between boxes, to improve the *adaptivity* of the method (which is very important in highly inhomogeneous situations) and to permit the *smoothing* of gravitational interactions. The results of our tests indicate that the tree-code is 2–4 times faster than the FMA for clumped distributions and 3–9 times for homogeneous distributions, at least in the interval of N here investigated ($N \leq 2 \cdot 10^5$) and at the same level of accuracy (error $\sim 10^{-3}$). This order of accuracy is generally considered as the best compromise between CPU-time consumption and precision for astrophysical simulations. Moreover, the claimed linear dependence on N of the CPU-time of the FMA is not confirmed and we give a "theoretical" explanation for that.

1. INTRODUCTION

The availability of fast computers is allowing rapid development of simulations of large N-body systems in Astrophysics, as well as in other fields of physics where the behaviour of large systems of particles is investigated.

The heaviest computational part of a dynamical simulation of such systems (composed by point masses and/or smoothed particles representing a gas) is the evaluation of the *long-range* force, such as the gravitational one, acting on every particle and due to all the other particles of the system. Astrophysically realistic simulations require very large N (greater than 10^5), making the direct $\sim N^2$ pair gravitational interaction evaluations too slow to perform. To overcome this problem various approximate techniques to compute gravitational interactions have been proposed.

0021-9991/98 $25.00

Among them, the tree-code algorithm proposed by Barnes and Hut (hereafter BH, see [1]) is now widely used in Astrophysics because it does not require any spatial fixed grid (like, for example, methods based on the solution of Poisson's equation). This allows it to follow very inhomogeneous and variable in time situations, typical of self-gravitating systems out of equilibrium. In fact its intrinsic capability to give a rapid evaluation of forces allows the dedication of more CPU-time to follow fast dynamical evolution, in contrast to other higher accuracy methods that are more suitable for other physical situations, e.g., molecular dynamics for polar fluids, where the Coulomb term is present.

While for the tree-code the CPU-time requirement scales as $N \log_8 N$, in the recently proposed Fast Multipole Algorithm (hereafter the FMA, see [9]) this time is claimed to scale as N, at least in quasi-homogeneous 2-D particle distributions. Were this linear behaviour confirmed in 3-D highly non-uniform cases, the FMA would really be appealing for use in astrophysical simulations.

In this paper, we compare CPU-times of our own implementations of the adaptive 3-D FMA and the tree-code to evaluate gravitational forces among N particles in three different (uniform and clumped) spatial configurations.

Detailed descriptions of the tree-code and FMA can be found in [1, 9, 10]. For the purpose of this paper (the performance comparison of the two above mentioned methods in astrophysically realistic situations) we built our own computer versions of the tree-code and FMA. Our tree-code was written following at most the [11] prescription but for the short-range component of the interaction force (see [12]), while our FMA is slightly different from the original proposed by Greengard [9], to make it more efficient in non-uniform situations where adaptivity is important, and to include the *smoothing* of interactions which is quite useful in astrophysical simulations, as we will describe in Section 3.

In Sections 2 and 3 we briefly review the algorithms and give some details of our implementations, in particular for the FMA, while in Section 4 the comparison of the FMA and the tree-code CPU-time performances is presented and discussed.

2. THE TREE-CODE

We have implementated our own version of the BH *tree-code* (see [1] or [10]) which is well described in [12] (see also [11]); we will give here only a brief discussion of the improvements we have introduced.

In the tree-code the cubic volume of side ℓ that encloses all the particles is subdivided in 8 cubic *boxes* (in 3-D) and each of them is furtherly subdivided in 8 *children boxes* and so on. The subdivision goes on recursively until the smallest boxes (the so-called *terminal boxes*) have only one particle inside. Moreover the subdivision is local and *adaptive* in the sense that it is locally as more refined as the density is higher. The logical internal representation of this picture is a "tree data structure," from that the denomination *tree-code*.

The gravitational force on a given particle in $\mathbf{r}$ is then computed considering the contribution of the "clusters" of particles contained in boxes that are sufficiently distant from the particle, that is, in those boxes which satisfy the *open-angle* criterion,

$$\frac{\ell_l}{d} < \theta \tag{1}$$

with d the distance between the particle and the center of mass of the cluster, $\ell_l = \ell/2^l$ the box size at level l of refinement, and $\theta > 0$ a parameter a priori fixed. This contribution is

evaluated by means of a truncated multipole expansion that permits the representation of the set of n particles contained in the box with a unique "entity" identified by a relatively low number of attributes (the total mass, the center of mass, the quadrupole moment, ..., that is, the set of multipole coefficients), that are stored, in a previous step, in the tree data structure. In this way the evaluation of interactions is speeded-up compared to a direct pair-to-pair evaluation. Usually in tree-codes second order expansions are used, that is, up to the quadrupole term, this approximation being sufficient in typical astrophysical simulations. Those boxes that do not satisfy the condition (1), will be "opened" and their *children* boxes will be considered. This "tree descending" continues until one reaches a terminal box whose contribution to the field will be calculated *directly*.

The parameter θ and the order of truncation of the expansion permit control of the error made in the evaluation of the field in the generic particle position. With the second order of approximation we used and with θ in the interval [0.7, 1] one obtains a relative error less than about 1%, as we will see in the following.

In our version of the tree-code we considered a *gravitational smoothing*. This means that each particle is represented by a β-spline (a polynomial function, with a compact support, differentiable up to the second order) that gives a Newtonian potential outside the sphere centered at the position of the particle and of radius 2ϵ, while inside that sphere the field is *smoothed*; see, for example, [11, 12]. This smoothing avoids divergence in the accelerations during too close approaches between particles whose trajectories would be not well integrated in time by the usual numerical procedures (like, for example, the *leap-frog* scheme). In the particular case of our *static* test, i.e., without considering any time evolution, the smoothing would only be an unnecessary complication; consequently in these tests we fix $\epsilon = 0$ for every particle (exactly Newtonian potential).

However, because of the presence, in general, of a smoothing of the field we have modified the open angle criteria in the following way: the box of size ℓ with center of mass at $\mathbf{R}$ is "sufficiently distant" from the particle in $\mathbf{r}$, i.e., its potential can be expanded in a multipole series, if

$$|\mathbf{R} - \mathbf{r}| > \max\{\ell/\theta, 2\epsilon\}, \tag{2}$$

where 2ϵ is the gravitational smoothing "radius" of the particle in $\mathbf{r}$.

3. THE FAST MULTIPOLE ALGORITHM

In a way similar to the tree-code, the FMA is based on an approximation of the gravitational field produced by a set of sufficiently distant particles over a generic particle. It uses the same logical "octal tree-structure" of the hierarchical subdivision of the space as the tree-code, with the only difference that instead of stopping the subdivision when boxes contain single particles, the recursive subdivision ends up at boxes containing no more than a fixed number $s > 1$ of particles. This reduces the CPU-time required for the calculation. We still call *terminal boxes* the boxes located at the "leaves" of the logical tree-structure. Also the FMA uses the truncated *multipole expansion*, but in a different and more complicated form. The advantage is that of a more rigorous control of the truncation error.

The FMA takes advantage of the possibility to build a Taylor expansion—the so-called *local expansion*—of the potential in a neighbourhood of a point P, knowing the coefficients of the *multipole expansion* that gives the potential in P (see Appendix B). This local

expansion is used to evaluate the acceleration of particles near P, instead of re-evaluating the multipole expansions of the field due to all collections of distant particles as it happens with the tree-code.

Furthermore, besides particle-particle and particle-box interactions, the FMA considers also a box-box interaction which is estimated by a truncated multipolar expansion, if and only if the boxes are sufficiently distant from each other, that is, if they satisfy the so-called *well-separation* criterion (which substitutes the open-angle criterion of the tree-code). Boxes which are not well-separated will be "opened" and their children boxes considered. In this way one can control the error introduced by the truncation of the multipolar expansion, giving this error, as we will see, a well defined upper bound.

In fact we know that if we have a set of k particles at $P_i = (\rho_i, \alpha_i, \beta_i)$ (we will use spherical coordinates) having masses m_i, which is enclosed in a sphere A with center in the origin and radius a, and another set of l particles at $Q_i = (r_i, \theta_i, \phi_i)$ enclosed in a sphere B centered in Q_0 with radius b, then the approximated potential generated by the set of particles in A at the position of the ith particle in B is given by a multipole expansion truncated at the order p:

$$\tilde{\Phi}_p(Q_i) \equiv -G \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m}{r_i^{n+1}} Y_n^m(\theta_i, \phi_i). \tag{3}$$

The functions $Y_n^m(\theta, \phi)$ are the spherical harmonics and

$$M_n^m \equiv \sum_{i=1}^{k} m_i \rho_i^n Y_n^{-m}(\alpha_i, \beta_i). \tag{4}$$

Now one can show (see [9]) that, for any $p \geq 1$,

$$|\Phi(Q_i) - \tilde{\Phi}_p(Q_i)| \leq \frac{\mathcal{A}}{r_i - a} \left(\frac{a}{r_i} \right)^{p+1}, \tag{5}$$

where Φ is the exact potential and $\mathcal{A} \equiv G \sum_{i=1}^{k} m_i$. This result gives the possibility to limit the error introduced by approximating Φ with the expression (3), if the two sets of particles are sufficiently distant.

Let d be the distance between the centers of the two spheres and let us define $\sigma \equiv d - a - b$ as the *separation* between them. Obviously the set of particles in B is such that $r_i \geq a + \sigma$ for any $i = 1, \ldots, l$. Then from (5) we find

$$\Delta\Phi \equiv \max_{i=1,\ldots,l} \{|\Phi(Q_i) - \tilde{\Phi}_p(Q_i)|\} \leq \frac{\mathcal{A}}{\sigma} \left(\frac{a}{\sigma + a} \right)^{p+1}. \tag{6}$$

The maximum error of the truncated multipole expansion depends on two quantities: the order of the expansion p and the separation σ.

In his original algorithm, Greengard introduced the concept of *well-separation* between two boxes of the *same level* of refinement, i.e., with the same size ℓ, to control the truncation error. Suppose that such boxes are circumscribed by two spheres A and B, with equal radii $a = b = \ell\sqrt{3}/2$; these two boxes are *well-separated* if, and only if, their separation is such

that $\sigma > a$, i.e., $d > 3a$. In this way the upper bound on the error will be, from Eq. (6),

$$\Delta\Phi \leq \frac{\mathcal{A}}{a}\left(\frac{1}{2}\right)^{p+1}. \tag{7}$$

To implement this criterion, Greengard imposed simply that two boxes of the same level are *well-separated* if, and only if, there are at least *two layers of boxes* of that level between them, in such a way to satisfy the inequality $d > 3a$ and, finally, Eq. (7).

Note that Greengard's well-separation criterion refers exclusively to boxes at the same level of refinement, limiting the efficiency of the algorithm especially in the case of non-uniform distributions of particles.

We have modified this criterion to make it more efficient (in the *adaptive* form of the algorithm) to face astrophysical typical distributions very far from uniformity. Let us briefly explain our modification.

First of all, every box is associated with a sphere that is not merely the sphere circumscribing the box but the *smallest* one containing all its particles. So, for terminal boxes, it is the smallest sphere concentric to the box that contains all its particles and for a non-terminal box of size ℓ, the sphere is also concentric to the box but the radius r is calculated recursively, knowing the radii r_i $(i = 1, \ldots, 8)$ of the spheres associated to each of its *children unempty boxes*, via the formula

$$r = \max_{i=1,\ldots,8}\{r_i\} + \ell\sqrt{3}/2. \tag{8}$$

This sphere contains all the spheres associated to the children boxes.[1] The value of the radius of these spheres is stored in the "tree" data structure together with the other data pertinent to the box and it is much more representative of the "size" of the set of particles in the box, than the mere box size ℓ, for controlling the truncation error.

Now consider the box $\hat{A}$ centered in the origin O and containing the collection of k particles seen before. Let A be its associated sphere with radius a (see Fig. 1). Let the other set of particles at Q_i, $i = 1, \ldots l$, be enclosed in the box $\hat{B}$ whose associated sphere is the sphere B centered in Q_0 and with radius b. The radii a, b have been evaluated by Eq. (8). Let us suppose that the separation of the two spheres is such that $\sigma > \delta \cdot a$, with $\delta > 0$ a fixed parameter (see again Fig. 1). Obviously the set of particles in B is such that, from Eq. (6),

$$\Delta\Phi \leq \frac{\mathcal{A}}{\delta \cdot a}\left(\frac{1}{\delta+1}\right)^{p+1} \tag{9}$$

and we can note that the parameter δ works in a way similar to θ in the tree-code.

Hence, our criterion of *well-separation* is the following: once we fix the parameter δ, we define two boxes as *well-separated* if the distance between their centers d is such that

$$d > a + b + \delta \cdot a, \tag{10}$$

that is, if the separation between their associated spheres is $\sigma > \delta \cdot a$. In this way the evaluation of the interaction between the sets of particles contained into such boxes will be affected by a truncation error which is bounded by Eq. (9).

[1] This is to preserve the same error bound as in Eq. (9), for the truncated expansions obtained translating and composing the local and the multipole expansions as discussed in Appendix B.

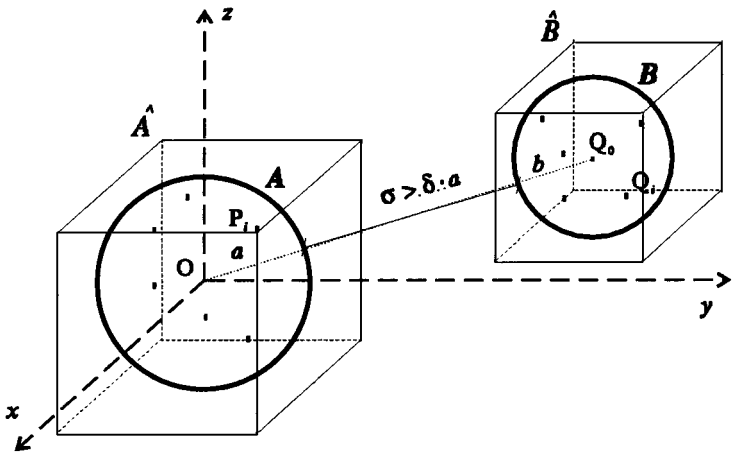

FIG. 1. Example of two well-separated boxes, $\hat{A}$ and $\hat{B}$ (see text). A and B are the associated spheres enclosing the two sets of particles.

The implementation of this criterion is very easy because it consists just in verifying Eq. (10) where the radii of the spheres associated to the boxes have been already calculated and stored, together with all the other data of each box, in the phase of the algorithm in which the tree-structure is built.

Our criterion of *well-separation* is more efficient than Greengard's original one, having the following features:

- it depends upon the internal distribution of particles in the box (via the radius of the associated sphere);
- it can involve boxes of different level of refinement, having different sizes, so to improve efficiency in non-uniform situations and to make unnecessary those complicated tricks conceived to shorten the list of the well-separated boxes like, for example, the mechanism of "parental conversion" (see [4]);
- it can be tuned in such a way to obtain the desired accuracy, via the parameter δ;
- it can be easily modified in order to allow a *gravitational smoothing* to be applied to the particles, as we will see below.

Note also, see Eq. (7), that to control the truncation error, Greengard varied p according to the desired accuracy. In our version we have, instead, fixed $p = 2$ and varied the parameter δ in order to obtain the same accuracy that is usually obtained with the tree-code in astrophysical simulations. One has to take care with keeping reasonably low the execution CPU-time (small compared with the human time scale!) and this is obtained at a price of a certain loss in accuracy in the evaluation of interactions. In fact the main characteristics of astrophysical simulations of gravitating systems are:

(1) they require a great number of particles (usually more than 10^4) for a rather large duration of the simulation[2] and, in particular,

(2) due to the intrinsic instability they offer a very wide distribution of time scales.

So, while simulating polar fluids in molecular dynamics one has to face "microscopic" time scales more narrowly distributed (just because molecules tend to repulse each other

[2] Several *dynamical times*, that is, many times the typical time scale of the entire system, such as the sound crossing time for a collisional system or the core-crossing time for a collisionless one.

due to the presence of short-range interactions, like the Lennard–Jones potential) and one can work with expansions truncated up to eighth order or more; in astrophysical simulations one prefers to limit the precision at a lower but reasonable level and, on the other hand, to be able to process systems which are highly dynamical. Therefore one usually works with expansions truncated up to the second order (in some cases even the first order) that, in the tree-code, correspond to considering up to the quadrupole moment.

For the mass density of the single particle we used the same β-*spline* profile as we did in our tree-code. In this way the potential is exactly Newtonian outside the sphere of radius[3] $2\epsilon_i$ centered in $\mathbf{r}_i$, so it can be expanded in multipole series *only in this region*. Hence the interaction with a particle inside the sphere of radius $2\epsilon_i$ must be necessarily evaluated by means of a *direct* summation. This requirement, which would be very difficult to incorporate in Greengard's criterion, has been considered via a little arrangement of our *well-separation* criterion (10); that is, two boxes ($\hat{A}$ and $\hat{B}$) are *well-separated* if, and only if, their spheres (A with radius a and B with radius b, respectively) are such that

$$d \geq a + b + \max\{\delta \cdot a, 2\epsilon_{\hat{A}}\}, \tag{11}$$

where d is the distance between the centers of the two spheres.

The smoothing length $\epsilon_{\hat{A}}$ used for the box $\hat{A}$ is another quantity stored in the tree data structure and it is given by this simple recursive scheme: if $\hat{A}$ is terminal, then $\epsilon_{\hat{A}} \equiv \max_i\{\epsilon_i\}$, where ϵ_i is the smoothing length of the particle i contained in $\hat{A}$, otherwise $\epsilon_{\hat{A}} \equiv \max_{\hat{C}}\{\epsilon_{\hat{C}}\}$, with the maximum taken over all the unempty children boxes $\hat{C}$ of $\hat{A}$. Anyway in the following comparison tests, we let $\epsilon = 0$ (as for the tree-code), i.e., no softening, because we are not interested in the dynamical evolution of the system.

For a more detailed description of our own implementation of the FMA see Appendixes B and C.

4. CODES PERFORMANCE COMPARISON

4.1. *A Suitable Set of Particle Distributions*

To compare the CPU-time spent by the two algorithms described in the previous sections, we ran the codes on a DEC Alpha 200-4/233 workstation with three different distributions of particles (which are always assumed to have the same mass).

(1) In the first case N particles have been distributed *uniformly* at random in a sphere of unitary radius.

(2) In the second case a set of N particles has been distributed, with a Monte–Carlo method, in a unitary sphere in such a way to discretize the "clumped" density profile,

$$\rho(r) = \frac{\rho_0}{[1 + (r/r_c)^2]^{5/2}}, \tag{12}$$

with $r_c = 0.2$ (obviously $\rho = 0$ for $r > 1$). This is known as the Schuster [15] profile (or as the Plummer profile in some texts); it corresponds to a polytropic sphere (of index 5) at equilibrium (see [2]) and represents a good approximation to the density distribution of various stellar systems.

[3] In principle each particle is allowed to have its own smoothing length. This is useful when one has to simulate gravitational interactions between both *collisionless* and *collisional* particles (i.e., fluid elements).

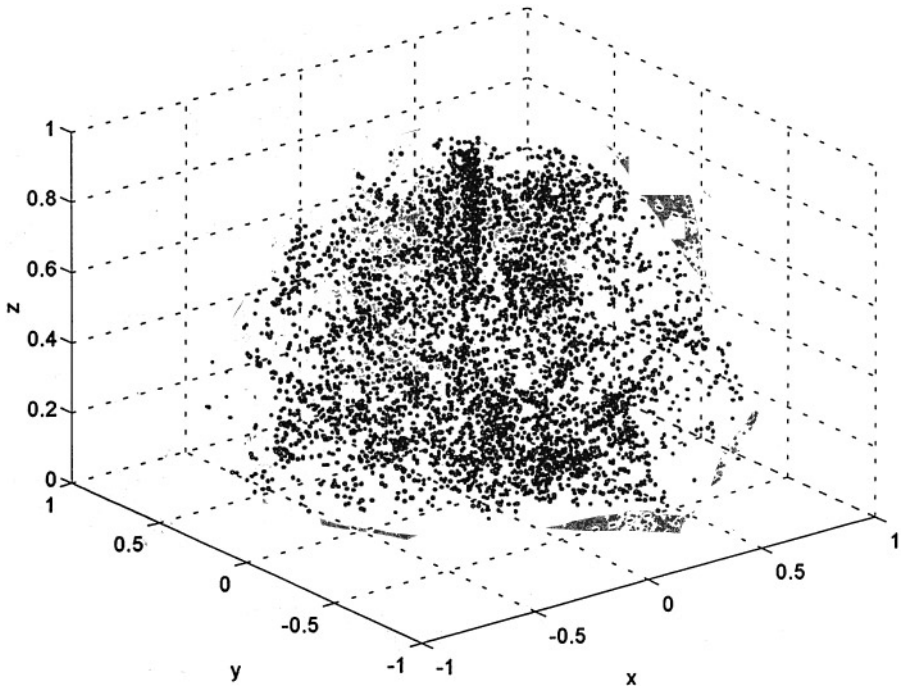

FIG. 2. Sample of $N = 4827$ galaxies in the Northern galactic hemisphere (Leda catalog). The unit on the axes corresponds to about 80 Mpc and the origin is on the Earth.

(3) In the third case we distributed N particles according to the density distribution of a sample of $n = 4827$ galaxies in the Northern galactic hemisphere (see Fig. 2), taken from the Leda catalog (see [5]). Other than the case $N = n$, five more different sets with $N = 2n, 4n, 8n, 20n$ and $40n$ particles have been placed at random, but always reproducing the same density distribution traced by the galaxies. We will call these distributions "cosmological."

4.2. *The Appropriate Choice of Refinement Levels*

The tree-code and the FMA in our implementations are both *adaptive*; this means that the number of levels of refinement needed to reach terminal boxes is a *local* quantity. Given a distribution of N particles, let l be the maximum of the number of subdivisions (taken over all the space inside the "root" box) that an algorithm based on a tree structure achieves: $l_{\rm tc}$ for the tree-code and $l_{\rm FMA}$ for the FMA. Obviously l depends on the ratio between the minimum particle-particle distance and the size ℓ of the root box containing all the particles. Moreover it is always $l_{\rm tc} > l_{\rm FMA}$, because $l_{\rm FMA}$ depends also on the number of particles (s) the FMA leaves in terminal boxes and, being in general $s > 1$, it performs less subdivisions in comparison with the tree-code.

To make homogeneous comparisons with codes of other authors, we need to limit the spatial subdivision. In fact the value of l strongly affects the rapidity of the algorithms. For instance, we verified that, given the Schuster clumped profile and with a fixed number of particles used to discretize it, passing from a distribution such as $l_{\rm tc} = 12$ and $l_{\rm FMA} = 11$ (with $s = 10$) to another one such as $l_{\rm tc} = 8$ and $l_{\rm FMA} = 7$, one has a speedup factor of

about 2. Moreover the memory requirement is also affected by l especially in a recursive implementation as ours, because at each recursive subroutine call a great amount of stack memory is allocated each time for the automatic variables and the number of recursive calls depends on the number of levels of refinement. Hence, we verified that it is impossible, with an usual workstation, to refine more than the 11th level with $N > 10^5$ particles.

For all these reasons we decided to generate particle positions in such a way to keep fixed l for both the codes. This does not mean that the algorithms become non-adaptive! The number of levels of refinement is still *local*, but it is limited by appropriate restrictions on the positions that the particles can assume.

We used the following restriction: each particle must be set at a point of a regular grid. Such a grid has a step $\ell/2^L$, with L a fixed integer. In this way the maximum "depth" reached in the space subdivision is limited by the inequality $l_{\text{FMA}} \leq l_{\text{tc}} \leq L$.

Thus the uniform and clumped distributions have been generated in such a way to give $l_{\text{tc}} \leq 8$ and $l_{\text{FMA}} \leq 7$ (with $s \geq 10$ in the FMA). Instead we found that the $N = n$ original galaxy distribution gave $l_{\text{tc}} = 10$ and $l_{\text{FMA}} = 9$. Consequently in generating all the cosmological distributions with $N > n$, we kept the same limit for the maximum levels of refinement in the space subdivision.

Of course during the time evolutions of such systems, l tends to grow due to the gravitational instability, but in this context we are only interested in giving homogeneous and comparable performance tests in the evaluation of forces only.

4.3. *The Results of the Comparison Tests*

The order of accuracy chosen was the same for both the tree-code and the FMA. For accuracy we mean how close, in modulus, the evaluated forces are to those calculated "exactly" by a direct, *particle-particle* (PP) method, which is affected only by the numerical error of the computer (due to the finite number of digits). Consequently we define the *relative error* of the calculation ε as

$$\varepsilon \equiv \frac{1}{N} \sum_i^N \frac{\left| a_i - a_i^{PP} \right|}{a_i^{PP}}, \tag{13}$$

where a_i is the modulus of the acceleration of the ith particle estimated by each of the two algorithms and a_i^{PP} that computed by the PP method. The error on the *direction* of the forces is much lower then the error on the modulus we have defined above, and, as it is usual in the N-body numerical method, it is not considered at all in performance tests being negligible.

Figure 3 gives the relative error of the tree-code and of the FMA for the various distributions. The error is about the same for both the algorithms. An averaged (on all the particles) relative error less than 1% (this is the order of magnitude of the error generally admitted in astrophysical simulations) and almost constant is obtained with the following settings for the various parameters:

- in the FMA we fixed $\delta = 2.5$ and varied s in the range 10–50 considering, as we have said, up to the second order term in the multipole expansion ($p = 2$);
- in the tree-code we varied θ in the range 0.65–0.75.

The CPU-time spent to calculate the accelerations vs the number N of particles is shown in Fig. 4. The CPU-time for the *particle-particle* method is also shown as a reference. Both

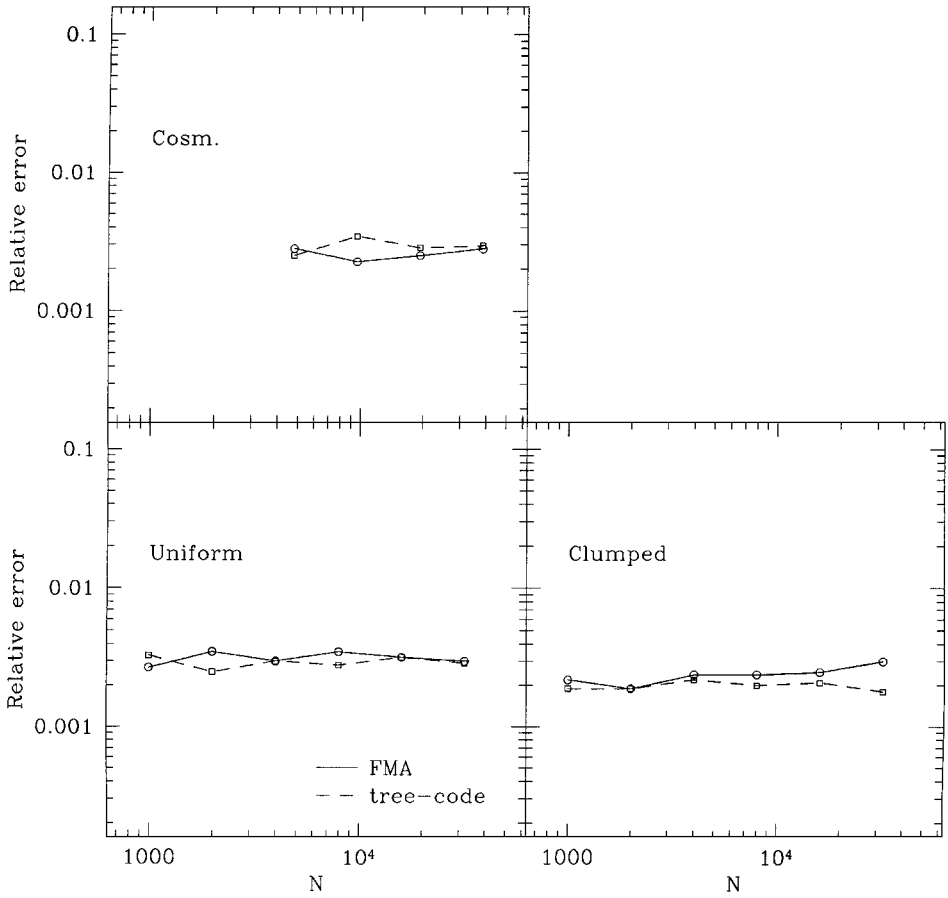

FIG. 3. Average relative error vs number of particles for both algorithms.

the algorithms are slower to compute forces in the non-uniform models than in the uniform one, and for the tree-code this is more evident. This is clearly due to the more complicated and non-uniform spatial subdivision in boxes that affects mostly the tree-code due to the *finer* and *deeper* subdivision of the space it uses. Anyway the tree-code is shown to be faster than the FMA for all the distributions and for N varying in the range we tested. As expected, the behaviour of the CPU-time, t, vs N for the tree-code is well fitted by the logarithmic law

$$t = \alpha N \log N + \beta, \tag{14}$$

where α and β are given in Table 1. In our opinion, this logarithmic law must be followed by the FMA, too, as we will explain in Subsection 4.4.

However, let us observe the CPU-time for the uniform case: it is not easy to distinguish at first sight a logarithmic behaviour from a linear one; furthermore we can presume, as we can observe in Fig. 4, that the FMA must show a more complicated behaviour due to the presence of the parameter s (the maximum number of particles left in terminal boxes) and to that s was varied.

Blelloch and Narlikar [3] have obtained similar "undulations" in the behaviour of the CPU-time of their version of the FMA, while the behaviour of the tree-code was much "cleaner," as in our tests.

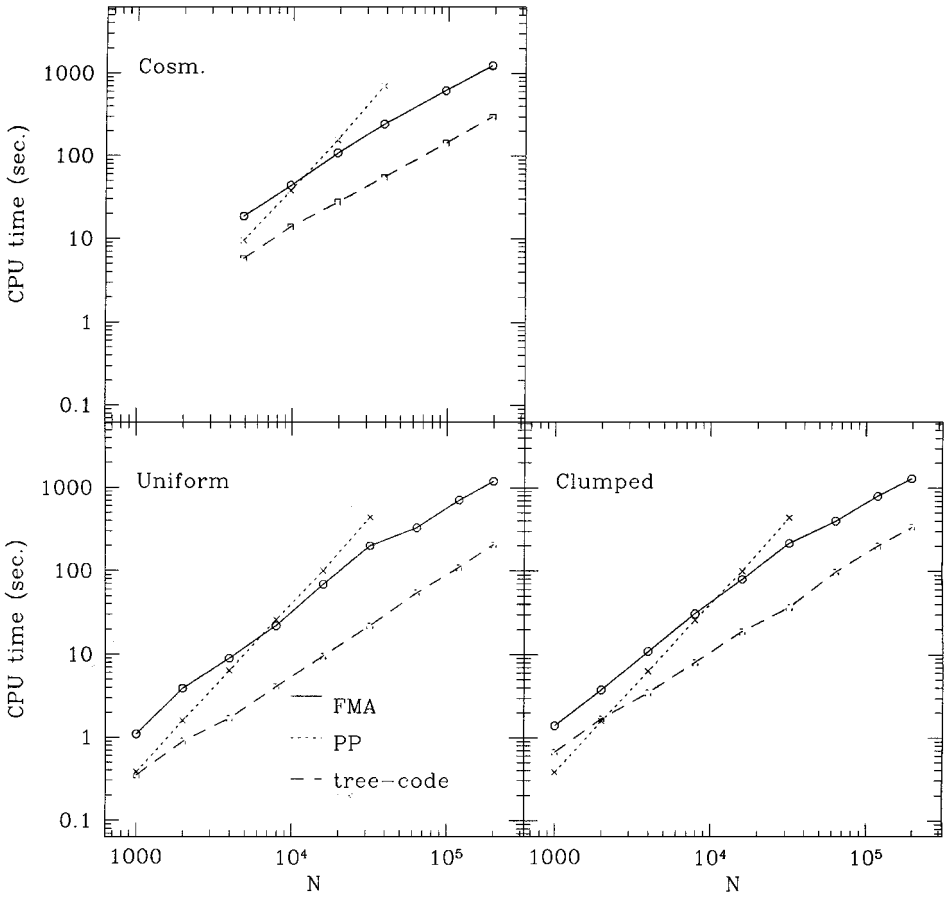

FIG. 4. CPU-time (on an Alpha 200-4/233 machine) vs the number of particles for both the algorithms and for the direct particle-particle method.

Coming back to the accuracy, we noted that the CPU-time of both algorithms, but especially that of the FMA, depends strongly on the error. For example, we found, for the cosmological $N = 8n$ distribution, that doubling the average error on the evaluation of the forces means reducing the FMA CPU-time to about 40% in comparison with the more accurate case.

We can see that our FMA becomes faster than the PP method for $N > 11{,}000$ in the Schuster model, for $N > 7{,}000$ in the uniform case and for $N > 12{,}000$ in the cosmological distributions.

TABLE 1

Values Obtained for the Parameters α and β, Fitting the Behaviour of the CPU-Time for the Tree-Code (t.c.) and the FMA with the Law $t = \alpha N \log_8 N + \beta$

Uniform			Clumped			Cosmological		
Alg.	α	β	Alg.	α	β	Alg.	α	β
t.c.	10^{-4}	-0.1	t.c.	$3 \cdot 10^{-4}$	-0.2	t.c.	$3 \cdot 10^{-4}$	0.6
FMA	$8 \cdot 10^{-4}$	-2	FMA	10^{-3}	-2	FMA	10^{-3}	-4

To make a comparison with codes of other authors, let us consider the FMA implemented in 3-D by Schmidt and Lee [14] (although this code *is not adaptive*) following the original Greengard algorithm and, in particular, his well-separation criterion. Their FMA is *vectorized* and it runs on a CRAY Y-MP, but anyway we do not make a direct comparison, but rather compare our "CPU-time ratio" (that is, the ratio between the CPU-time consumed by the FMA and that consumed by the direct PP method), with the same ratio as obtained by the codes of Schmidt and Lee, *at the same order of magnitude of the error on the forces*.

So we can note (see [14]) that for $N = 10^4$ uniformly distributed particles, a truncation of eighth order ($p = 8$) and *five* levels of refinement, they obtain a CPU-time of 418 s for their FMA, against 11 s spent by the *direct* PP method, with a relative error on the forces of about $1.4 \cdot 10^{-3}$. Thus they obtain a ratio $t_{\mathrm{FMA}}/t_{PP} \sim 40$, while with our codes we see that for the uniform distribution with $N = 10^4$ and *seven* levels of refinement, we have an error on the forces that is $\sim 3 \cdot 10^{-3}$ (see Fig. 3) with a CPU-time ratio $t_{\mathrm{FMA}}/t_{PP} \sim 0.8$, i.e., our FMA spends about 80% of the CPU-time spent by the PP method (see Fig. 4). This ratio is certainly lower if we use only five levels of refinement, also in the case of an error on the forces close to their one.

This seems a very good result.

4.4. *Scaling of CPU-Times versus N*

Greengard and other authors [4, 9, 13] assert that the FMA would exhibit a *linear* scaling of the CPU-time vs N. We tried to fit t_{FMA} as a function of N, with various laws, the linear included. The result is that a *logarithmic* behaviour like that of Eq. (14), gives the best fit, as for the tree-code.

From Fig. 4 and Table 1 we can see that the tree-code is roughly 2–4 *times faster* than the FMA in the "astrophysical" more clumped distributions, while in uniform situations (with the order of accuracy we fixed) the FMA goes even slower.

Note, in fact, how the difference of performances reduces slightly passing to more clumped distributions; this is because, as we have said, the FMA is less sensitive to the degree of uniformity of the distribution of the particles than the tree-code which uses a finer subdivision of the space in boxes (as it corresponds to $s = 1$).

The higher speed of the tree-code compared to the FMA is easily understood, at least in 3-D, since while in theory the FMA is more efficient and less "redundant" in managing information (remember the use of the Taylor expansion of the potential on near bodies), in practice this "potential" greater efficiency pays the price of a certain quantity of computational "complications." This carries the method to a negative total balance in terms of speed with respect to the competing tree-code.

How can we interpret the CPU-time scaling?

The logarithmic behaviour of the tree-code is explained by a simple estimate of the number of operations needed by the various steps of the algorithm (see, e.g., [1, 10, 12]). It is roughly given by the product between the number of particles N by the number of "bodies" (boxes or particles), about $\log_8 N$, which contribute to the force on each particle. Hence $t_{\mathrm{tc}} \sim N \log_8 N$.

The logarithmic behaviour of the FMA can be similarly understood when one reconsiders carefully the cost of each step of the algorithm.

It can be estimated (see [9]) that

$$t_{\mathrm{FMA}} \sim aN + bN_{ter} + cN_{ne}, \tag{15}$$

that is, the CPU-time is a *linear function* of N, N_{ter} (the number of all *terminal* boxes), and N_{ne} (the number of all non-empty boxes). Greengard [9] in his final considerations on the scaling of the FMA in the adaptive 2-D version (the 3-D case is similar) estimates the number of these types of boxes (see Lemmas 2.6.4 and 2.6.5 in [9]) to be

$$N_{ter} \sim 4 \cdot L \cdot \frac{N}{s} \tag{16}$$

$$N_{ne} \sim 5 \cdot L \cdot \frac{N}{s}, \tag{17}$$

where L is the total *number of subdivisions* needed to reach terminal boxes. This L is identified by Greengard with $L \sim \log_2(1/\Delta)$, where Δ, *a priori* fixed, is the *spatial resolution* that one wants to reach in the simulation. With Δ fixed, L would be *independent* of N, thus N_{ter} and N_{ne}, in Eq. (15), are quantities linear in N. Then the FMA CPU-time estimated by Eq. (15) is linear in N too.

The crucial point is that a constant Δ (and L) only allows manipulation of distributions of particles with $\Delta < r_{min}$, with r_{min} the minimum distance between a pair of particles (see observation 2.5.1 in [9]).

Obviously, in 3-D, $r_{min} \sim N^{-1/3}$, so that $\Delta < r_{min}$ implies

$$L \gtrsim \frac{1}{2} \log_2 N = \log_8 N. \tag{18}$$

Substituting this inequality in the expressions (16) and (17) for N_{ne} and N_{ter}, one obtains from Eq. (15) that $t_{\mathrm{FMA}} \gtrsim N \log_8 N$, that is, the same "natural" scaling of the tree-code.

The FMA, to be competitive with the tree-code in astrophysical simulations (with the numbers of particles currently used), needs substantial improvements. Some of these improvements have already been achieved, but only in a molecular dynamics context, where a higher accuracy is requested (see [7]). To get an indication of the direction where one has to proceed for such an improvement, we made a "profile" of the CPU-time spent by the various parts of the algorithm. This profile is shown in some detail in the Appendix A.

We note that the particular *implementation* of the forces evaluation by means of the various truncated expansions (together with the manipulations operated on them) makes the difference with the tree-code and deserves to be further optimized. In fact, in the tree-code the same multipolar expansions are used, but in a way much more optimized than in the FMA, because in the tree-code one truncates *a priori* the expansion at the quadrupole ($p = 2$) moment. Moreover one uses *real coefficients* (the mass, center of mass, etc.) which satisfy simpler composition and translation rules than the multipolar and Taylor *complexes* coefficients used in the FMA. We think a similar way to proceed has to be introduced in an "astrophysical" (i.e., with a priori fixed p) Fast Multipole Algorithm, in order to make it competitive with the tree-code.

5. CONCLUSIONS

Realistic astrophysical simulations require a large amount of computation to evaluate gravitational forces. Many codes which give a rapid numerical evaluation of the force field have been proposed in the literature. In this paper we compare two of these codes: one (the BH tree-code) has been largely used in astrophysics for ten years; the other (the Greengard's FMA [9]) has given promising results in the field of molecular dynamics.

We have so implemented our own optimized *serial* versions of both the tree-code and *adaptive*, 3-D, FMA.

The results of our comparison tests indicate the tree-code as faster than the FMA over all the interval of total number of particles ($N \leq 2 \cdot 10^5$) allowed by the central memory capacity of a "typical" workstation. This maximum value of N, which could appear low with respect to modern parallel simulations where it can reach 10^7, is anyway meaningful because even in fully parallel codes *each* processor cannot manipulate more than about 10^4–10^5 particles. The problems in the parallelization of the two codes are comparable, due to their similar structure; thus the higher speed of the tree-code, here verified in a serial context, should be confirmed in the parallel implementation and it seems a valid reason to concentrate efforts for the most efficient parallelization of the tree-code.

At the end of this paper we have discussed the dependence of the FMA CPU-time on N and given an explanation for why its behaviour is similar to that of the tree-code.

APPENDIX A: ANALYSIS OF THE CPU-TIME CONSUMPTION FOR THE FMA

Table 2 shows the CPU-time spent by the various subroutines in our FMA implementation, during a run with $N = 1.2 \cdot 10^5$ particles distributed according to the clumped density profile. The routine BUILD builds the tree structure setting all the pointers and the links among parent-children boxes, classifying boxes in terminal and non-terminal, and so on. The routine EVAL stores in this structure all the data related to each box (the various multipolar and Taylor coefficients, the radius of the associated sphere, etc.).

The functions ARMR and ARMI give, respectively, the real and the imaginary part of the various spherical harmonics needed in the expansions. The functions GARMij give instead the real and the imaginary part ($j = R, I$, respectively) of all components ($i = X, Y, Z$) of the gradient of the spherical harmonics.

The routine FORCE1 evaluates the force acting on a given box due to all the rest of the system, while FORCE2 evaluates the force acting on a box A due to another box B, considering all the possible cases (both boxes, only one, or none are terminal; they are, or not, well-separated, etc.). All these routines are *recursive*, except the functions.

It is evident that most of the CPU-time is spent by FORCE2 (80% of the total). Thus, in this CPU-time profile, this routine has been split into four parts in order to investigate better what part is the most CPU-time expensive. These parts are listed in Table 2 as:

TABLE 2
Percentages of the Total CPU-Time Spent by the Different Routines of the FMA in the Run with $N = 1.2 \cdot 10^5$ Particles in the Clumped Case

Routine	t (%)	Routine	t (%)
FORCE2.transf	35	GARMYR	2
FORCE2.direct	22	GARMYI	2
FORCE2.mult	18	GARMZR	2
FORCE2.other	5	GARMZI	2
ARMR	4	BUILD	1
ARMI	2	FORCE1	0.5
GARMXR	2	MAIN	0.4
GARMXI	2	EVAL	0.1

Note. The total CPU-time has been 800 s.

- FORCE2.direct, which performs the direct calculations of the forces on the particles in A due to those in B (when both boxes are terminal):
- FORCE2.transf which transforms the multipole coefficients pertinent to a well-separated box B into Taylor coefficients of the box A;
- FORCE2.mult which evaluates the forces due to the box B on the particles in A by means of the multipole expansion (when A is terminal); and, finally,
- FORCE2.other representing the rest of the routine.

Note that the "approximate" calculations (both box-box and box-"terminal box" interactions via multipolar expansion) use about the 53% of the total CPU-time, while direct force evaluations (both "terminal box"-"terminal box" interactions) use 22%. Finally, note that the phase of tree structure construction and data storage (BUILD + EVAL) takes only about 1% of the total CPU-time.

APPENDIX B: MANIPULATION OF MULTIPOLE EXPANSIONS IN THE FMA

Here we briefly describe the three theorems, due to Greengard [9], that permit the manipulation, in 3-D, of the various series expansions used in the algorithm, and that are useful for the deeper and formal description of our own implementation that follows in the next Appendix C.

We have said that one can "transform" the multipole expansion (3) into a *local expansion* useful to evaluate the field about a given point P. More precisely, given the set of k particles at P_i in the sphere A (associated to the box $\hat{A}$, see Fig. 1) which produce the gravitational potential $\Phi(Q_i)$ over the set of particles in the sphere B (box $\hat{B}$) with center Q_0, then one can show that in the vicinity of Q_0 the approximated potential

$$\Psi_p(Q_i) = -G \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta_i', \phi_i')(r_i')^j, \tag{B1}$$

where $Q_i - Q_0 = (r_i', \theta_i', \phi_i')$, differs from the exact $\Phi(Q_i)$ of an amount bounded by the same expression that appears in the r.h.s. of Eq. (9). In this truncated *local expansion* the coefficients are given by

$$L_j^k = \sum_{n=0}^{p} \sum_{m=-n}^{n} S_{j,k,n,m}^{(1)} M_n^m Y_{j+n}^{m-k}(\alpha_i, \beta_i)(\rho_i)^{-j-n-1}, \tag{B2}$$

M_n^m being the *same* one that appears in the expression (3) and $S^{(1)}$ a matrix of coefficients (see Appendix D).

Another theorem allows us to calculate M_n^m in a *recursive* manner. Let us consider the *partition* of the set of k particles in the box $\hat{A}$, in the $q \leq 8$ subsets each of them enclosed in the spheres associated to the children boxes of $\hat{A}$ and with centers in $(\rho^{(1)}, \alpha^{(1)}, \beta^{(1)}), \ldots, (\rho^{(q)}, \alpha^{(q)}, \beta^{(q)})$. Let $M_n^{m(1)}, M_n^{m(2)}, \ldots, M_n^{m(q)}$ be the multipole coefficients calculated for each of the subsets of particles. If all these spheres are enclosed in the sphere A (this is automatically satisfied because of Eq. (8)), the coefficients M_j^k given by

$$M_j^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} S_{j,k,n,m}^{(2)} M_{j-n}^{k-m^{(q)}} Y_n^{-m}\left(\alpha^{(q)}, \beta^{(q)}\right)\left(\rho^{(q)}\right)^n \tag{B3}$$

(see the Appendix D for the matrix $S^{(2)}$) give a potential

$$\tilde{\Phi}_p(P) = -G \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} Y_n^m(\theta, \phi) \tag{B4}$$

(where $P = (r, \theta, \phi)$ is a generic point *outside* the sphere A), which *well approximates* the exact potential $\Phi(P)$ generated by *all* the k particles in the sphere A. In fact if P is the position of a particle in the *well-separated* box $\hat{B}$, then one can show that

$$|\tilde{\Phi}(P) - \Phi(P)| \leq \left(\frac{G \sum_i^k m_i}{r - a} \right) \left(\frac{a}{r} \right)^{p+1} < \frac{G \sum_i^k m_i}{\delta \cdot a} \left(\frac{1}{\delta + 1} \right)^{p+1}. \tag{B5}$$

The truncation error has the same upper bound given by (9).

So we can compute M_j^k associated to the box $\hat{A}$ containing the total set of particles knowing only those pertinent to the q subsets in each children box. In the tree-code the same happens, but there the analogous theorem—the "quadrupole composition theorem"—has been developed *specifically* for the quadrupole moment by Goldstein [8] (those for the monopole and the dipole are obvious). In the FMA this theorem works for coefficients of any order and it has an upper error bound.

Thus, once the M_n^m have been calculated for all *terminal* boxes using the definition (4), by means of (B3) we can compute recursively the coefficients of *parent* boxes ascending the tree-structure. This coefficients will be transformed, when needed, in the local expansion coefficients (as we will see in more detail in Appendix C) necessary to calculate forces by (B1). In this way, we will be sure that the error made in approximating the "true" potential with the various expansions, will always be bounded by Eq. (B5).

The last theorem relates to the translation and composition of the *local expansion* coefficients (briefly *Taylor coefficients*). In this case the rules of composition work, in a certain sense, inversely. That is, given the coefficients L_j^k relative to the set of k particles in the sphere A, such that the potential $\Psi_p(P)$ ($P = (r, \theta, \phi)$ is a point *inside* A) given by the truncated local expansion about the origin O,

$$\Psi_p(P) \equiv -G \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta, \phi) r^j, \tag{B6}$$

is such that

$$|\Psi_p(P) - \Phi(P)| < \frac{G \sum_i^k m_i}{\delta \cdot a} \left(\frac{1}{\delta + 1} \right)^{p+1}, \tag{B7}$$

then at the same point but with another origin $Q \in A$, we have the equality

$$\Psi_p(P) = -G \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^{Q^k} Y_j^k(\theta', \phi')(r')^j, \tag{B8}$$

where $P - Q = (r', \theta', \phi')$ and where the new translated coefficients are

$$L_j^{Q^k} = \sum_{n=j}^{p} \sum_{m=-n}^{n} S_{j,k,n,m}^{(3)} L_n^m Y_{n-j}^{m-k}(\alpha, \beta) \rho^{n-j} \tag{B9}$$

being $O - Q = (\rho, \alpha, \beta)$ (for the matrix $S^{(3)}$ see Appendix D).

Thus given the L_j^k for a box, we can compute the Taylor coefficients for all the unempty *children boxes* using the above formula, with the new origin Q at the center of each children boxes. The process has to be recursively iterated until we reach terminal boxes. But how can we obtain the coefficients L_j^k of a box B "the first time" (not knowing those of its parent box)? Obviously they will be calculated by means of Eq. (B2), transforming the multipole coefficients of *sufficiently distant* boxes, that is, of boxes that are *well-separated* from B.

APPENDIX C: FORMAL DESCRIPTION OF THE FMA ALGORITHM

Here we describe in deeper detail our version of the FMA algorithm that is slightly different from the original Greengard algorithm in the adaptive implementation. This differences regard mainly the way interactions between distant boxes and the set of particles in a terminal box are calculated. Moreover, as we have said, we have modified the Greengard *well-separation* criterion to take into account the presence of a *smoothing* of the interaction that in astrophysical simulations, contrary to molecular dynamics, must be included.

Let us first introduce some useful definitions: in the following s indicates the maximum number of particles in the *terminal boxes* and the script letters refer to collections of boxes while simple capitals letters refer to a single box.

- l_{max} is the maximum level of refinement reached in the space subdivision;
- l_A indicates the level of box A, whereas the level of the *root box* R, that is, the box containing all the particles, is 0;
- $\mathcal{B}(l)$ is the set of all boxes at level l of refinement;
- $M(A)$ is the *parent* box of box A;
- $\mathcal{C}(A)$ is the set of all *children boxes* of box A;
- $\mathcal{C}(\mathcal{S}) \equiv \cup_{A\in\mathcal{S}}\mathcal{C}(A)$ is the set of all children boxes of each box in the set $\mathcal{S}$;
- $\mathcal{X}(A)$ indicates the set of boxes, of level l_A or $l_A + 1$, that are *NOT well separated* from box A. The set contains all the *brothers* of box A, but not A itself;
- $\mathcal{T}$ is the set of *terminal boxes*, that is, such boxes that have no children because they contain less than $s + 1$ particles, so they have not been subdivided;
- n_A, with $A \in \mathcal{T}$, is the number of particles inside the terminal boxes A (obviously $n_A \leq s$);
- d_{AB} represents the distance between the geometrical centers of boxes A and B;
- ϵ_A is the length of the *gravitational smoothing* relative to the box A;
- r_A is the radius of the sphere that contains all the particles in the box A and that is concentric to it (see text).

The notation "do $n = a, b$" (with $b > a$ integers) means that all passages included between this statement and the correspondent "end do" are repeated $b - a + 1$ times and every time the integer variable n takes the values $a, a + 1, \ldots, b - 1, b$, like in Fortran; while the notation "do $A \in \mathcal{S}$" means, in this case, that every time the loop is executed the *box* A represents one of the various boxes in the set $\mathcal{S}$. So the statements between "do" and the related "end do" are repeated Card$\{\mathcal{S}\}$ times and each time with a different box $A \in \mathcal{S}$. For example, if $\mathcal{S} = \{A_1, A_2, A_3, \ldots, A_n\}$, the box A is A_1 the first time the loop is executed, A_2, the second time, and so on. However, the order the boxes have in the set $\mathcal{S}$ has no importance in the algorithm. On the contrary in the first case of "do ... end do," the order in the values that n takes every time is important. Another notation is the "do while *condition*"

meaning that the statements between this "do while ..." and the correspondent "end do" will be executed *while* the logical condition keeps *true*.

Calculate recursively the multipole coefficients for all the boxes of each level, starting from the terminal boxes. This procedure is the same as that in the tree-code, but with the difference that in the FMA the multipole expansion is calculated with the origin in the geometrical center of the boxes, so the first order coefficient (the dipole moment) does not vanish. Another complication is that in the FMA these coefficients are necessarily complex quantities. For terminal boxes use Eq. (4), while for the others use Eq. (B3).

Calculate the radius r of the sphere that contains all the particles in each box. If a box is *terminal* then $r \equiv \max_i\{|\mathbf{r}_i - \mathbf{R}|\}$, where $\mathbf{r}_i$ is the position of the particle i in the box and $\mathbf{R}$ is its center. If it is not a terminal box the radius is calculated by means of (8).

let $\mathcal{X}(R) = \emptyset$
do $l = 1, l_{max}$
  do $C \in \mathcal{B}(l)$
  let $\mathcal{X}(C) = \emptyset$
    if $l > 1$ **then** translate, if they exist, the coefficients of the local expansion of the parent box $M(C)$ about the center of box C using Eq. (B9).
    if $C \notin \mathcal{T}$ **then** [the box C isn't terminal]
      do $B \in \mathcal{X}(M(C))$
        if $d_{BC} \geq \max\{2\epsilon_B, \delta \cdot r_B\} + r_B + r_C$ **then**
          convert the multipole coefficients of box B to Taylor coefficients about the center of box C with Eq. (B2), because it is *well separated* from C and the sphere where the field generated by the masses in B is *smoothed, does not intersect* the sphere associated to C. Sum the Taylor coefficients to the pre-existent ones.
        else
          do $B_1 \in \mathcal{C}(B)$
            if $d_{B_1C} \geq \max\{2\epsilon_{B_1}, \delta \cdot r_{B_1}\} + r_{B_1} + r_C$ **then**
              convert the multipole coefficients of box B_1 to Taylor coefficients of box C with Eq. (B2), because it is *well separated* from C and the sphere where the field generated by the masses in B_1 is *smoothed, does not intersect* the sphere associated to C. Sum the Taylor coeff. to the pre-existent ones.
            else
              put B_1 into the collection $\mathcal{X}(C)$.
            end if
          end do
        end if
      end do
    else [the box C is *terminal*]
      let $\mathcal{A} = \mathcal{X}(M(C))$
      do while $\mathcal{A}$ **not empty**
        do $B \in \mathcal{A}$
          if $B \in \mathcal{T}$ **then** [the box B is *terminal*]
            eliminate B **from the set** $\mathcal{A}$
            do $i = 1, n_C$

do $j = 1, n_B$

Sum directly (i.e., without any expansion) to the grav. field on the particle i that due to the particle j taking into account the *grav. smoothing*

end do

end do

else [the box B isn't terminal]

if $d_{BC} \geq \max\{2\epsilon_B, \delta \cdot r_B\} + r_B + r_C$ **then** [the box C is *well-sep.* from the box B and it is outside its smoothing sphere]

eliminate B **from the set** $\mathcal{A}$

Sum to the *Taylor coefficients* of the box C those obtained transforming the *multipole coefficients* of the box B by means of (B2).

end if

end if

end do

let $\mathcal{A} = \mathcal{C}(\mathcal{A})$ [Now indicate with $\mathcal{A}$ the collection of all the children boxes of each box in the precedent set $\mathcal{A}$. This means that we are descending the tree to the next level]

end do

do $i = 1, n_C$

Calculate the local expansion of the grav. field in the position of the particle i, using Eq. (B1) and the coefficients pertinent to the box C, summing to the accelerations calculated up to now.

end do

end if

end do

end do

Note that we have simplified the way forces on the particles in terminal boxes are evaluated. In Greengard's adaptive algorithm this is made by means of complicated passages and classifications of boxes into many several collections that are computationally expensive to build up.

In our opinion this complication is unnecessary, because when one has to consider a terminal box for which one has the long-range component of the potential in terms of Taylor coefficients (translated from those of its parent box), one has only to calculate the short-range forces on the n particles (with $n < s$) inside the terminal box, due to a certain set of near boxes and this can be done in the most efficient way by means of the same kind of passages that in the *tree-code* are used to evaluate the force on a single particle.

Suppose we have to evaluate forces on n_C particles in the terminal box C. When we deal with a non-terminal box B and this box is *not* well-separated from C, then it will be subdivided considering its children boxes and the subdivision is recursively repeated until we reach either terminal or well-separated boxes. The contribution due to terminal boxes will be calculated *directly*, that is, summing particle-particle interactions. The contribution due to well-separated boxes will be evaluated converting their multipole expansion coefficients into Taylor ones, summing them to the pre-existent coefficients of the box C and then, in a following passage, using these coefficients and the Taylor expansion to evaluate gravitational

forces at the points occupied by the particle in C. This is done in the last statements of the above description (from the "do while ..." forward).

APPENDIX D: THE MATRICES OF COEFFICIENTS

Defining $A_j^k \equiv (-1)^j[(j-k)!(j+k)!]^{-1/2}$, we have

$$S^{(1)}_{j,k,n,m} \equiv B^m_{k,n} A^m_n A^k_j / A^{m-k}_{j+n} \tag{D1}$$

$$S^{(2)}_{j,k,n,m} \equiv C^{k-m}_m A^m_n A^{k-m}_{j-n} / A^k_j \tag{D2}$$

$$S^{(3)}_{j,k,n,m} \equiv D^m_{n-j,m-k} A^{m-k}_{n-j} A^k_j / A^m_n, \tag{D3}$$

where

$$B^m_{k,n} \equiv (-1)^n \begin{cases} (-1)^{\min\{|m|,|k|\}} & \text{if } m \cdot k > 0 \\ 1 & \text{otherwise} \end{cases} \tag{D4}$$

$$C^s_r \equiv \begin{cases} (-1)^{\min\{|r|,|s|\}} & \text{if } r \cdot s < 0 \\ 1 & \text{otherwise} \end{cases} \tag{D5}$$

$$D^s_{n,m} \equiv (-1)^n \begin{cases} (-1)^m & \text{if } m \cdot s < 0 \\ (-1)^{s-m} & \text{if } m \cdot s > 0 \text{ and } |s| < |m| \\ 1 & \text{otherwise.} \end{cases} \tag{D6}$$

REFERENCES

1. J. Barnes and P. Hut, *Nature* **324**, 446 (1986).
2. J. Binney and S. Tremaine, *Galactic Dynamics* (Princeton Univ. Press, Princeton, NJ, 1987).
3. G. Blelloch and G. Narlikar, A practical comparison of N-body algorithms, 1995.
4. J. A. Board, Jr. and J. F. Leathrum, *The Parallel FMA in Three Dimension*, Technical Report, Dept. of Electrical Engineering, Duke University, 1992.
5. M. Di Nella and G. Paturel, *C. R. Acad. Sci. Paris Sec. II* **319**, 57, 1994.
6. J. W. Eastwood and R. W. Hockney, *Computer Simulation Using Particles* (Hilger, Bristol, UK, 1988).
7. D. Elliott and J. A. Board, *J. Sci. Comput.* **17**, 398 (1996).
8. H. Goldstein, *Classical Mechanics* (Addison–Wesley, Reading, MA, 1980).
9. L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, Ph.D. Thesis (MIT Press, Cambridge, MA, 1987).
10. L. Hernquist, *Ap. J. Suppl. S.* **64**, 715 (1987).
11. L. Hernquist and N. Katz, *Ap. J. Suppl. S.* **70**, 419 (1989).
12. P. Miocchi, Graduation Thesis, Department of Physics, University of L'Aquila, Italy, 1994.
13. J. K. Salmon and M. S. Warren, Astrophysical N-body simulations using hierarchical tree data structures, in *Supercomputing '92*, IEEE Comp. Soc., Los Alamitos, 1992.
14. K. Schmidt and M. A. Lee, *J. Stat. Phys.* **63**(5/6), 1223 (1991).
15. A. Schuster, *British Assoc. Report*, 427 (1883).