Journal of Computational Physics **179,** 27–42 (2002)
doi:10.1006/jcph.2002.7026

A Hierarchical $\mathcal{O}(N)$ Force Calculation Algorithm

Walter Dehnen

Max-Planck-Institut für Astronomie, Königstuhl 17, D-69117 Heidelberg, Germany
E-mail: dehnen@mpia.de

Received December 7, 2000; revised February 13, 2002

A novel code for the approximate computation of long-range forces between N mutually interacting bodies is presented. The code is based on a hierarchical tree of cubic cells and features mutual cell–cell interactions which are calculated via a Cartesian Taylor expansion in a symmetric way, such that total momentum is conserved. The code benefits from an improved and simple multipole acceptance criterion that reduces the force error and the computational effort. For $N \gtrsim 10^4$, the computational costs are found empirically to rise sublinearly with N. For applications in stellar dynamics, this is the first competitive code with complexity $\mathcal{O}(N)$; it is faster than the standard tree code by a factor of 10 or more. © 2002 Elsevier Science (USA)

Key Words: n-body simulations; tree code; fast multipole method; adaptive algorithms.

1. INTRODUCTION

In N-body simulations of stellar dynamics (or any other dynamics incorporating long-range forces), the computation, at every time step, of the gravitational forces between N mutually interacting bodies dominates the operational effort. In many situations, such as studies of collisionless stellar systems, the error of these simulations is dominated by the noise in the distribution of the bodies, whose number N is just a numerical parameter. Therefore, instead of computing the forces exactly by direct summation over all pairs of bodies, one may use approximate but much faster methods, allowing substantially larger N and hence significantly reduced noise.

In stellar dynamics, one of the most commonly used approximate methods is the Barnes and Hut tree code [1] and its clones. With these methods, one first sorts the bodies into a hierarchical tree of cubic cells and precomputes multipole moments of each cell. Next, the force at any body's position and generated by the contents of some cell is calculated by a multipole expansion if the cell is well-separated from the body; otherwise the forces generated by the cell's child nodes are taken. This technique reduces the number of interactions per body to $\mathcal{O}(\log N)$ and hence results in an overall complexity of $\mathcal{O}(N \log N)$.

Another technique used frequently, for instance in molecular dynamics, is Greengard and Rokhlin's [2, 3] fast multipole method (FMM) and its variants. Traditionally, these methods first sort the bodies into a hierarchy of nested grids, precompute multipole moments of each cell, and then compute the forces between grid cells by a multipole expansion, usually in spherical harmonics. That is, cells are not only sources but also sinks, which formally reduces the complexity to $\mathcal{O}(N)$, although the author is not aware of an empirical demonstration in three dimensions.[1]

Currently, no useful implementation of FMM for application in stellar dynamics exists. In fact, it has been shown [5] that, for this purpose, the FMM in its traditional form cannot compete with the tree code. The reason, presumably, is twofold: first, because stellar systems are very inhomogeneous, nonadaptive methods and are less useful; second, FMM codes are traditionally designed for high accuracy, whereas in collisionless stellar dynamics a relative force error of few 10^{-3} is often sufficient.

Here, we describe in detail a new code, designed for application in the low-accuracy regime, that combines the tree code and FMM, taking the best of each. In order to be fully adaptive, we use a hierarchical tree of cubic cells. The force is calculated employing *mutual* cell–cell interactions, in which both cells are source and sink simultaneously. Whether a given cell–cell interaction can be executed or must be split is decided using an improved multipole-acceptance criterion (MAC). The new code is a further development of the code presented in [6], which in turn may be considered an improvement of a code given in [7]. It is substantially faster than the tree code and empirically shows a complexity of $\mathcal{O}(N)$ or even less.

The paper is organized as follows. In Section 2, the numerical concepts are presented. Section 3 describes the algorithm. In Section 4 the force errors are empirically assessed for some typical stellar dynamical test cases. Section 5 presents empirical measurements of the complexity and performance, also in comparison to other methods. Finally, Section 6 sums up and concludes.

2. APPROXIMATING GRAVITY

The goal is to approximately compute the gravitational potential Φ and its derivative, the acceleration, at all body positions $\boldsymbol{x}_i$ and generated by all other $N-1$ bodies

$$\Phi(\boldsymbol{x}_i) = -\sum_{j\neq i} \mu_j \, g(\boldsymbol{x}_i - \boldsymbol{x}_j), \tag{1}$$

where μ_i is the weight of the ith body. Consider two cells A and B (see Fig. 1), with centers of mass z_A and z_B, respectively. The Greens function describing the mutual interaction between a body at position $\boldsymbol{x}$ in cell A and a body at position $\boldsymbol{y}$ in cell B may be Taylor expanded about the separation $\boldsymbol{R} \equiv z_\mathrm{A} - z_\mathrm{B}$,

$$g(\boldsymbol{x}-\boldsymbol{y}) = \sum_{n=0}^{p} \frac{1}{n!}(\boldsymbol{x}-\boldsymbol{y}-\boldsymbol{R})^{(n)} \odot \boldsymbol{\nabla}^{(n)} g(\boldsymbol{R}) + \mathcal{R}_p(g), \tag{2}$$

where p is the order of the expansion, while $\mathcal{R}_p$ denotes the Taylor series remainder (see also Appendix A). Here, we follow Warren and Salmon [7] by using the notational shorthand

[1] One of the most recent members of the FMM family, presented by Cheng *et al.* [4], still does not show clear $\mathcal{O}(N)$ behavior at $N = 10^6$ (see, e.g., Table I of their paper).

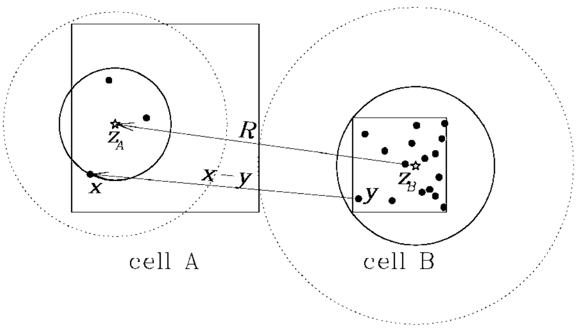

FIG. 1. Two interacting cells (boxes). Bodies are shown as solid dots. The stars indicate the positions of the centers of mass, the solid and dotted circles around which have radii $r_{\max}$ and r_{crit}, respectively, for $\theta = 0.5$.

in which $\boldsymbol{x}^{(n)}$ indicates the n-fold outer product of the vector $\boldsymbol{x}$ with itself, while $\odot$ denotes a tensor inner product. When inserting (2) into (1), whereby restricting the sum over j to all bodies within cell B, we obtain for the potential at every position $\boldsymbol{x}$ in cell A and generated by all bodies in cell B [7]

$$\Phi_{\text{B}\to\text{A}}(\boldsymbol{x}) = -\sum_{m=0}^{p} \frac{1}{m!}(\boldsymbol{x} - \boldsymbol{z}_{\text{A}})^{(m)} \odot \mathbf{C}^{m,p}_{\text{B}\to\text{A}} + \mathcal{R}_p(\Phi_{\text{B}\to\text{A}}), \tag{3}$$

$$\mathbf{C}^{m,p}_{\text{B}\to\text{A}} = \sum_{n=0}^{p-m} \frac{(-1)^n}{n!} \nabla^{(n+m)} g(\boldsymbol{R}) \odot \mathbf{M}^n_{\text{B}}, \tag{4}$$

$$\mathbf{M}^n_{\text{B}} = \sum_{\boldsymbol{y}_i \in \text{B}} \mu_i (\boldsymbol{y}_i - \boldsymbol{z}_{\text{B}})^{(n)}. \tag{5}$$

The summation over m in Eq. (3) represents the evaluation of gravity, represented by the field tensors $\mathbf{C}^{m,p}_{\text{B}\to\text{A}}$, at the evaluation point $\boldsymbol{x}$ within the sink cell A. The computation of the field tensors via the summation over n in Eq. (4) represents the interaction between sink cell A and source cell B, represented by its multipole moments $\mathbf{M}^n_{\text{B}}$.

The symmetry between $\boldsymbol{x}$ and $\boldsymbol{y}$ of the Taylor expansion (2) has the important consequence that if Eqs. (3) and (4) are used to compute $\nabla\Phi_{\text{B}\to\text{A}}$ *and* $\nabla\Phi_{\text{A}\to\text{B}}$, Newton's third law is satisfied by construction. For instance, the sum over all forces of N bodies vanishes within floating point accuracy.

Note that the highest-order multipole moments, $\mathbf{M}^{n=p}$, contribute only to the coefficients $\mathbf{C}^{0,p}$ and, hence, affect only Φ but not $\nabla\Phi$. Since in stellar dynamics the acceleration rather than the potential is to be computed, one may well ignore $\mathbf{M}^p$, reducing CPU-time and -memory requirements.

The formulae used in the standard tree code can be obtained by setting $\boldsymbol{x} = \boldsymbol{z}_{\text{A}}$, corresponding to body sinks. In this case, potential and acceleration are approximated by $-\mathbf{C}^{0,p}_{\text{B}\to\text{A}}$ and $\mathbf{C}^{1,p}_{\text{B}\to\text{A}}$, respectively.

2.1. Gravity between Well-Separated Cells

In our implementation, we stick to a third-order expansion ($p = 3$), ignoring octopoles $\mathbf{M}^3_{\text{B}}$ (see remark above). The dipole $\mathbf{M}^1_{\text{B}}$ vanishes by construction and the Taylor-series coefficients for spherical Greens functions read (with Einstein's sum convention)

$$\begin{aligned}
\mathsf{C}^0_{\mathrm{B}\to\mathrm{A}} &= \mathsf{M}_\mathrm{B}\left[D^0 + \frac{1}{2}\tilde{\mathsf{M}}^2_{\mathrm{B}ii}D^1 + \frac{1}{2}R_iR_j\tilde{\mathsf{M}}^2_{\mathrm{B}ij}D^2\right],\\
\mathsf{C}^1_{\mathrm{B}\to\mathrm{A},i} &= \mathsf{M}_\mathrm{B}\left[R_i\left(D^1 + \frac{1}{2}\tilde{\mathsf{M}}^2_{\mathrm{B}jj}D^2 + \frac{1}{2}R_jR_k\tilde{\mathsf{M}}^2_{\mathrm{B}jk}D^3\right) + R_j\tilde{\mathsf{M}}^2_{\mathrm{B}ij}D^2\right],\\
\mathsf{C}^2_{\mathrm{B}\to\mathrm{A},\,ij} &= \mathsf{M}_\mathrm{B}[\delta_{ij}D^1 + R_iR_jD^2],\\
\mathsf{C}^3_{\mathrm{B}\to\mathrm{A},\,ijk} &= \mathsf{M}_\mathrm{B}[(\delta_{ij}R_k + \delta_{jk}R_i + \delta_{ki}R_j)D^2 + R_iR_jR_kD^3],
\end{aligned} \tag{6}$$

where $\mathsf{M}_\mathrm{B} \equiv \mathsf{M}^0_\mathrm{B}$ is the mass of cell B and $\tilde{\mathsf{M}}^2_\mathrm{B} \equiv \mathsf{M}^2_\mathrm{B}/\mathsf{M}^0_\mathrm{B}$ its *specific* quadrupole moment, while

$$D^m \equiv \left(\frac{1}{r}\frac{\partial}{\partial r}\right)^m g(r)\bigg|_{r=|\boldsymbol{R}|}. \tag{7}$$

In practice, we calculate the coefficients $\tilde{\mathbf{C}}^m_{\mathrm{B}\to\mathrm{A}} \equiv \mathsf{M}_\mathrm{A}\mathbf{C}^m_{\mathrm{B}\to\mathrm{A}}$, because these obey the symmetry relations $\tilde{\mathbf{C}}^2_{\mathrm{B}\to\mathrm{A}} = \tilde{\mathbf{C}}^2_{\mathrm{A}\to\mathrm{B}}$ and $\tilde{\mathbf{C}}^3_{\mathrm{B}\to\mathrm{A}} = -\tilde{\mathbf{C}}^3_{\mathrm{A}\to\mathrm{B}}$ (for $p=3$), which arise from the mutuality of gravity. Since we consider always both directions of any interaction, exploiting these relations substantially reduces the operational effort.

2.2. Accumulating Taylor Coefficients

For each cell, after the coefficients of all its interactions have been accumulated as $\tilde{\mathbf{C}}^m_\mathrm{A} = \sum_\mathrm{B}\tilde{\mathbf{C}}^m_{\mathrm{B}\to\mathrm{A}}$, where the sum includes all interaction partners B of cell A, we transform back to $\mathbf{C}^m_\mathrm{A} = \tilde{\mathbf{C}}^m_\mathrm{A}/\mathsf{M}_\mathrm{A}$. Next, for each body, the Taylor series of all relevant cells (those that contain the body) have to be accumulated by first translating to a common expansion center and then adding coefficients.

In contrast to expansions in spherical harmonics, the translation of the Cartesian expansion (3) to a different center z is straightforward. Let $\mathbf{C}^{m,p}_0$ be the coefficients for expansion center z_0; then the coefficients for expansion center z_1 are

$$\mathbf{C}^{m,p}_1 = \sum_{n=0}^{p-m}\frac{1}{n!}(z_0 - z_1)^{(n)} \odot \mathbf{C}^{m+n,p}_0. \tag{8}$$

2.3. The Multipole Acceptance Criterion (MAC)

For the expansion (3) to converge, we must have $|\boldsymbol{x} - \boldsymbol{y} - \boldsymbol{R}| < |\boldsymbol{R}|$ for all body–body interactions "caught" by a single cell–cell (or cell–body) interaction (see Appendix A). In order to ensure this, we first obtain, for each node, an upper limit r_max for the distance of any body within the node from the center of mass (bodies naturally have $r_\mathrm{max} = 0$). We take r_max to be either the distance from the cell's center of mass to its most distant corner [8], or [9]

$$\max_{\text{child nodes } i}\{r_{i,\mathrm{max}} + |z - z_i|\}, \tag{9}$$

whichever is smaller. Then $|\boldsymbol{x} - \boldsymbol{y} - \boldsymbol{R}| < \theta|\boldsymbol{R}|\ \forall\boldsymbol{x} \in \mathrm{A}$ and $\boldsymbol{y} \in \mathrm{B}$, i.e., the nodes are *well-separated* if

$$|\boldsymbol{R}| > r_{\mathrm{A,crit}} + r_{\mathrm{B,crit}} \quad \text{with} \quad r_\mathrm{crit} = r_\mathrm{max}/\theta. \tag{10}$$

The *tolerance parameter* θ controls the accuracy of the approximation: for Newtonian forces, the error made in d dimensions by the pth-order expansion of the form (3) is (Eq. (A.7))

$$|\mathcal{R}_p(\nabla\Phi_{\mathrm{B}\to\mathrm{A}})| \leq \frac{(p+1)\theta^p}{(1-\theta)^2}\,\frac{\mathsf{M}_\mathrm{B}}{R^2} \tag{11}$$

$$\propto \frac{\theta^{p+2}}{(1-\theta)^2}\, r_{\mathrm{B,max}}^{d-2} \propto \frac{\theta^{p+2}}{(1-\theta)^2}\, \mathsf{M}_\mathrm{B}{}^{(d-2)/d}, \tag{12}$$

where $\mathsf{M}_\mathrm{B} \propto r_{\mathrm{B,max}}^d$ has been assumed. Since $\mathsf{M}_\mathrm{B}/R^2$ is, to lowest order, the acceleration due to the interaction, Eq. (11) tells us that with constant θ, which is standard tree-code practice, the *relative* error introduced by every single interaction is approximately constant. Equation (12), however, shows that for the most important case of $d = 3$ this practice results in larger *absolute* errors for interactions with bigger and hence on average more massive cells. This implies that these interactions will dominate the *total* error of any body's acceleration. It is, therefore, expedient to balance the absolute errors of the individual interactions, which can be approximately achieved by using a mass-dependent tolerance parameter,

$$\frac{\theta^{p+2}}{(1-\theta)^2} = \frac{\theta_{\min}^{p+2}}{(1-\theta_{\min})^2}\left(\frac{\mathsf{M}}{\mathsf{M}_{\mathrm{tot}}}\right)^{(2-d)/d}, \tag{13}$$

where $\mathsf{M}_{\mathrm{tot}}$ is the mass of the root cell, while $\theta_{\min} = \theta(\mathsf{M}_{\mathrm{tot}})$ is the new tolerance parameter. If $r_{\mathrm{A,max}} \sim r_{\mathrm{B,max}}$ and $\mathsf{M}_\mathrm{A} \sim \mathsf{M}_\mathrm{B}$, this method results in approximately constant absolute acceleration errors. Note that Eq. (13) results in a very weak decrease of θ with increasing mass: $\theta \propto \mathsf{M}^{-1/15}$ for $d = 3$, $p = 3$, and $\theta \ll 1$.

Instead of Eq. (12), one can obtain a stricter error limit incorporating the first $p + 1$ multipole moments (see Appendix A), which may be turned into a MAC [7]. However, our choice (13) is (i) much simpler and (ii) overcomes already the main disadvantage of $\theta = \mathrm{const}$, the variations of the absolute individual errors.

2.4. Direct Summation

For small N the exact force computation via direct summation is not only more accurate than approximate methods but also more efficient. Therefore, we replace the approximate technique by direct summation whenever the latter results in higher accuracy at the same efficiency (see Appendix B for more details).

If an interaction is executed by direct summation, the Taylor coefficients of the interacting cell(s) are not affected, but the coefficients $\tilde{\mathbf{C}}^0$ and $\tilde{\mathbf{C}}^1$ of the bodies within the cell(s) are accumulated.

3. THE ALGORITHM

3.1. Tree Building and Preparation

In the first stage, a hierarchical tree of cubic cells is built, as described in [1], although cells containing s or fewer bodies are not divided. Next, the tree is linked such that every cell holds the number of cell children[2], a pointer to its first cell child, as well as the number of body children, of all body descendants, and a pointer to the first body child. Since cell as well as body children are contiguous in memory (to arrange this we actually use tiny copies of the bodies, called "souls," which only hold a pointer to their body and its mass, position, acceleration, and potential) these data allow fast and easy access to all cell and body children of any given cell and to all bodies contained in it.

Next, the masses M, centers of mass z, radii $r_{\max}$ and r_{crit}, and specific quadrupole moments $\tilde{\mathsf{M}}_0^2$ of each cell are computed in a recursive way from the properties of the child nodes.

3.2. A New Generic Tree-Walk Algorithm

One important new feature of our code is the mutual treatment of all interactions: both interacting nodes are source and sink simultaneously. The standard tree walk, as for instance implemented by the generic algorithm given in [7], and the usual FMM coefficient accumulation algorithm, e.g., in [4], contain an inherent asymmetry between sinks and sources and thus cannot be used for our purposes. Instead, our algorithm approximates the forces in two steps: an *interaction phase*, incorporating Eq. (4), and an *evaluation phase*, incorporating Eq. (3).

3.2.1. *The Interaction Phase*

Because of the mutuality of the interactions, we cannot accumulate the Taylor coefficients $\tilde{\mathbf{C}}^n$ "on the walk," but each node must accumulate the coefficients of all its interactions in its own private memory. Cells need storage for $\tilde{\mathbf{C}}^0$ to $\tilde{\mathbf{C}}^3$, while bodies only need to accumulate $\tilde{\mathbf{C}}^0$ and $\tilde{\mathbf{C}}^1$, i.e., potential and acceleration. The accumulation of these coefficients is done using the following algorithm with the root cell for arguments A and B.

ALGORITHM 1 (INTERACT(NODE A, NODE B)).

```
try to perform the interaction between NODEs $A$ and $B$;      // see Appendix B
if (it cannot be performed)
  if ($A = B$)                                            // split cell self-interaction
    for (all pairs $\{a, b\}$ of child NODEs of $A$)
      INTERACT($a, b$);
  else if ($r_{\max}(A) > r_{\max}(B)$)                            // split bigger node
    for (all child NODEs $a$ of $A$)
      INTERACT($a, B$);
  else
    for (all child NODEs $b$ of $B$)
      INTERACT($A, b$);
```

Thus, if an interaction cannot be executed, using the formulae of the last section—see Appendix B for details—it is split. In case of a mutual interaction, the bigger node is divided and up to eight new mutual interactions are created, while a self-interaction of a cell results in up to 36 new interactions between its child nodes. In practice, we use a nonrecursive code incorporating a stack of interactions.

[2] Hereafter "child" means a direct subnode of a cell, while "descendant" refers to any node contained within a cell, including the children, the grandchildren, and so on.

3.2.2. *The Evaluation Phase*

Finally, the Taylor coefficients relevant for each body are accumulated and the expansion is evaluated at every body's position. After transforming $\tilde{\mathbf{C}}$ to $\mathbf{C}$ for every cell and body, this is done by the following recursive algorithm, which is initially called with the root cell and an empty Taylor series as arguments.

ALGORITHM 2 (EVALUATE GRAVITY(CELL A, TAYLOR SERIES T_0)).

```
$T_A$ = TAYLOR SERIES due to the $\mathbf{C}^n$ of CELL $A$;
translate center of $T_0$ to center of mass of $A$;      // using equation (8)
$T_A$ += $T_0$;                                          // add up coefficients
for (all BODY children of $A$) {
    evaluate $T_A$ at BODY's position;                   // as in equation (3)
    add to BODY's potential and acceleration;
}
for (all CELL children $C$ of $A$)
    EVALUATE GRAVITY($C$, $T_A$);                        // recursive call
```

Thus, the coefficients of the Taylor series that is eventually evaluated at some body's position have been added up from all hierarchies of the tree and hence account for all interactions of all cells that contain the body.

4. ERROR ASSESSMENT

Two types of force errors are involved in collisionless N-body simulations of stellar dynamics. One is the unavoidable error between the smooth force field of the underlying stellar system modeled and the forces *estimated* from the positions of N bodies (which are sampled from this stellar system). This *estimation error* can be reduced by increasing the number N of bodies in conjunction with a careful *softening*: the Newtonian Greens function $g(\boldsymbol{x}) = G/|\boldsymbol{x}|$ is replaced with a nonsingular function that approaches the Newtonian form for $|\boldsymbol{x}|$ larger than the softening length ϵ. But at fixed N, it cannot be decreased below a certain optimum value [10, 11].

The other type of error is introduced by an approximate rather than exact computation of these estimated forces. While this *approximation error* can be reduced to (almost) any size (at the price of increasing computational effort), it is sufficient to reduce it well below the level of the estimation error. We now first assess the approximation error alone and then consider the total error.

4.1. Accuracy of the Approximation

We estimate the accuracy of the approximated forces for various choices of the parameters controlling the algorithm and for two typical astrophysical situations: a spherical Hernquist [12] model galaxy, which has density and force per unit mass

$$\rho(\boldsymbol{x}) = \frac{\mathsf{M}_{\text{tot}}}{2\pi} \frac{r_0}{|\boldsymbol{x}|(r_0 + |\boldsymbol{x}|)^3}, \quad \boldsymbol{F}(\boldsymbol{x}) = -\frac{\boldsymbol{x}}{|\boldsymbol{x}|} \frac{G\mathsf{M}_{\text{tot}}}{(r_0 + |\boldsymbol{x}|)^2}, \tag{14}$$

and a group of five such galaxies at different positions and with various scale radii r_0. In either case, we sample $N = 10^4$, 10^5, and 10^6 bodies. We use standard Plummer softening,

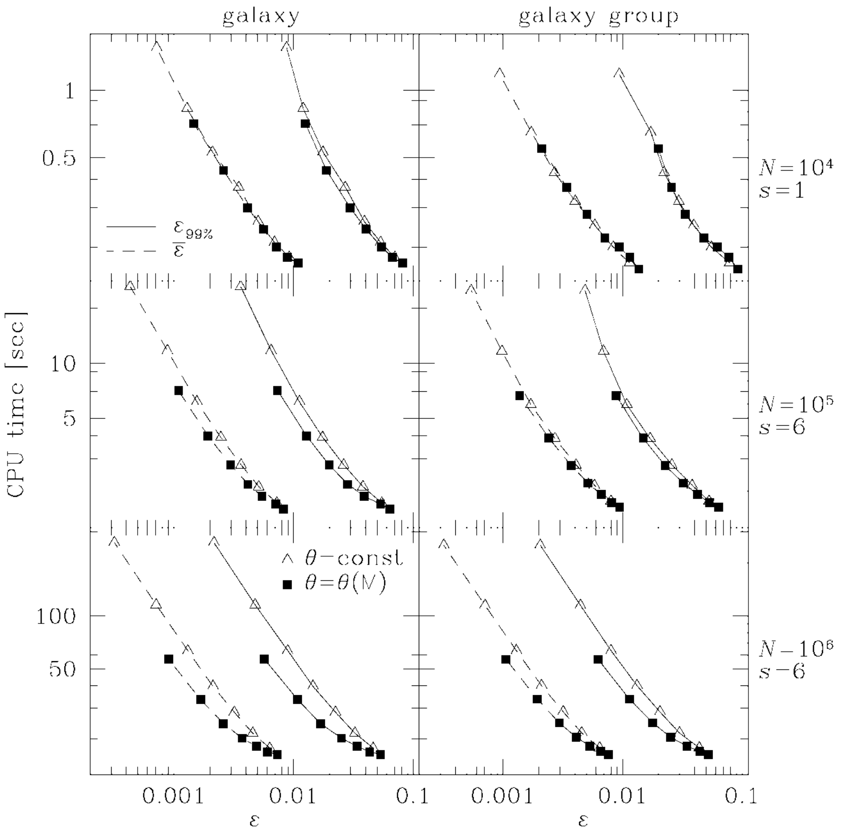

FIG. 2. The mean (dashed) and 99th percentile (solid) relative force error versus the CPU time consumed on a PC (Pentium III/933MHz/Linux) for approximating the forces of a galaxy (left) and a group of galaxies (right), sampled with a total of 10^4 (top), 10^5 (middle), or 10^6 (bottom) bodies. We used constant (open triangles) or mass-dependent tolerance parameters (solid squares). The symbols along each curve correspond, from left to right, to θ, $\theta_{\min} = 0.3$, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. Cells containing s or fewer bodies have not been divided.

where $g(r) = G/\sqrt{r^2+\epsilon^2}$, with softening lengths ϵ chosen so as to minimize the estimation error [11]. In order to single out the approximation error, we compare the approximated forces with those obtained by a computation via direct summation (in double precision; in case of $N = 10^6$, for the first 10^5 bodies only). As a measure for the relative error, we compute for each body [5]

$$\varepsilon \equiv |a_{\text{approx}} - a_{\text{direct}}|/a_{\text{direct}}, \tag{15}$$

where a denotes the magnitude of the acceleration. Figure 2 plots the mean relative error, $\bar{\varepsilon}$, and that at the 99th percentile, $\varepsilon_{99\%}$, versus the CPU time needed by an ordinary PC (Pentium III/933MHz/Linux/compiler: gcc version 2.95.2) for both constant and mass-dependent tolerance parameter. This figure allows several interesting observations.

1. For the same $\theta = \text{const}$ and stellar system, the errors decrease with increasing N. This is because, at constant *relative* force error per individual interaction (as is the case for $\theta = \text{const}$), the total error of some body's force scales with the inverse square root of the number of individual interactions contributing, which increases with N.

2. At the same operational effort, as measured by the CPU time, the mass-dependent tolerance parameter employing Eq. (13) results in smaller errors than $\theta = \text{const}$, for $N > 10^4$. This advantage becomes more pronounced for larger N, because, with $\theta = \text{const}$, the *absolute* force errors of individual interactions contributing to some body's force vary stronger with increasing N (due to the larger range of cell masses), such that balancing them becomes more beneficial.

3. Finally, a relative error of $\varepsilon_{99\%}$ of a few percent or $\bar{\varepsilon}$ of a few 10^{-3} at $N = 10^5$, which is commonly accepted to be sufficient in astrophysical contexts [5], requires a tolerance parameter $\theta = \text{const} \simeq 0.65$ or $\theta_{\min} \simeq 0.5$.

4.2. The Total Force Error

The important question here is for which choices of θ is the approximation error negligible compared to the estimation error? To answer this question, we performed some experiments using samples of $N = 10^4$, 10^5, and 10^6 bodies drawn from a Hernquist model and computed the mean averaged squared error (MASE) of the force (per unit mass) [13]:

$$\text{MASE}(\boldsymbol{F}) = \left\{ \frac{1}{\mathsf{M}_{\text{tot}}} \sum_i \mu_i (\hat{\boldsymbol{F}}_i - \boldsymbol{F}(\boldsymbol{x}_i))^2 \right\}. \tag{16}$$

Here, $\hat{\boldsymbol{F}}_i$ and $\boldsymbol{F}(\boldsymbol{x})$ are, respectively, the approximately estimated force for the ith body and the true force field of the stellar system. The curly brackets denote the ensemble average over many possible random realizations of the same underlying stellar density by N bodies. We used $10^7/N$ ensembles and computed the $\text{MASE}(\boldsymbol{F})$[3] for various values of θ, but always at optimum ϵ [11]. As one might already have guessed from the behavior of the approximation error in Fig. 2, the relative increase of the $\text{MASE}(\boldsymbol{F})$ is negligible: even for $\theta = 0.9$, the approximation error contributes less than 1%, in agreement with the findings of [13]. Based on this result, one may advocate the use of tolerance parameters larger than $\theta_{\min} \simeq 0.5$. However, the distribution of approximation errors is not normal, and the rms error, which is essentially measured by (the square root of) the $\text{MASE}(\boldsymbol{F})$, may well underestimate the danger of using large θ. We therefore cannot recommend using $\theta_{\min} \gtrsim 0.7$.

5. PERFORMANCE TESTS

5.1. Scaling with N

We measured the CPU time consumption for both constant $\theta = 0.65$ and $\theta = \theta(\mathsf{M})$ with $\theta_{\min} = 0.5$, using $s = 6$ in either case. Figure 3 plots the consumed time (averaged over many experiments) per body versus N plotted on a logarithmic scale. For the case of $\theta = \theta(\mathsf{M})$, Table I gives the number of cells as well as the number of individual body–body (B–B), cell–body (C–B) and cell–cell (C–C) interactions, where the latter two are split into those done via a Taylor expansion and direct summation (subscripts "app" and "dir"), respectively.

[3] The force field of the Hernquist model has a central singularity, causing a 100% force error at $r = 0$, which cannot be resolved by N-body methods. In our experiments, we have therefore restricted the summation in Eq. (16) to $|\boldsymbol{x}_i| > \epsilon/2$.

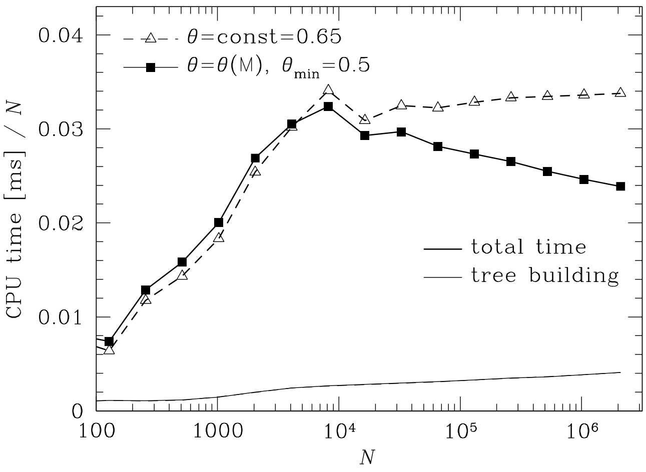

FIG. 3. CPU time consumption per body (Pentium III/933MHz PC) versus N for the test case of a galaxy group; $s = 6$, and $\theta = \mathrm{const} = 0.65$ or $\theta_{\min} = 0.5$.

The tree code requires $\mathcal{O}(N \log N)$ operations, corresponding to a rising straight line in Fig. 3. For our code, however, there is a turnover at $N \sim 10^4$, above which the CPU time per body approaches a constant[4] (for $\theta = \mathrm{const}$) or even decreases with N (for $\theta = \theta(\mathsf{M})$), i.e., the total number of operations becomes $\mathcal{O}(N)$ or less, which is also evident from the number of interactions in Table I.

In order to understand these scalings, let us first consider the simpler case of $\theta = \mathrm{const}$ and a homogeneous distribution of bodies. Then, eightfolding N is equivalent to arranging eight copies of the old root cell into the octants of the new root cell [1], and the total number of interactions rises from N_I to $8N_I + N_+$, where N_+ interactions are needed for the mutual forces between these octants. In terms of a differential equation, this gives

$$\frac{\mathrm{d}N_I}{\mathrm{d}N} \simeq \frac{N_I}{N}\frac{\Delta \ln N_I}{\Delta \ln N} \approx \frac{N_I}{N} + \frac{N_+}{N 8 \ln 8}, \tag{17}$$

where the first term on the right-hand side accounts for the *intradomain* and the second for

TABLE I
Number of Cells and of Interactions for $\theta_{\min}$ = 0.5, s = 6, and the Same Test Case as in Fig. 3

N	N_{cells}	B–B	C–B$_{\mathrm{app}}$	C–B$_{\mathrm{dir}}$	C–C$_{\mathrm{app}}$	C–C$_{\mathrm{dir}}$	Total
1,000	370	210	3,746	2,049	4,057	1,600	11,662
3,000	1,090	472	16,870	5,184	30,101	4,472	57,099
10,000	3,558	753	61,873	11,914	163,554	11,887	249,981
30,000	10,607	494	170,634	24,984	572,322	20,102	790,474
100,000	35,282	112	429,035	73,350	1,954,508	53,466	2,516,146
300,000	106,065	66	1,041,836	205,821	5,457,445	137,138	6,859,114
1,000,000	353,342	1	2,918,326	621,802	16,105,065	393,311	19,023,391
3,000,000	1,060,650	2	7,771,584	1,689,115	42,974,890	1,047,627	53,645,379

[4] The slow rise can be entirely attributed to the tree building, which is an $\mathcal{O}(N \log N)$ process.

the *interdomain* interactions. Equation (17) has the solution

$$N_I \simeq c_0 N + \frac{N}{8\ln 8}\int \frac{N_+}{N^2}\,\mathrm{d}N. \tag{18}$$

In the tree code, every body requires a constant number of additional interactions, i.e., $N_+ \propto N$, and the second term in (18) becomes $\propto N \ln N$ dominating $N_I(N)$. However, in the new code, N_+ grows sublinearly for large N, since a constant number of cell–cell interactions accounts for most new interactions of all bodies. In this case, the second term on the right-hand side of Eq. (18) also grows sublinearly with N and N_I will eventually be dominated by the first term. That is, in contrast to the tree code, the interdomain interactions are neglible at large N when compared to the intradomain interactions. The transition value of N will depend on the tolerance parameter θ and the distribution of bodies.

Another way of estimating the scaling of the computational costs with N is similar to the FMM approach [2, 4]: On each level l of the tree there are $\sim 8^l$ cells of mass $\mathsf{M} \sim 8^{-l}$, i.e., $n(\mathsf{M}) \propto \mathsf{M}^{-2}$. Each cell has $\propto \theta^{-3}$ interactions, and thus

$$N_I \propto \int \frac{d\mathsf{M}}{\mathsf{M}^2\theta^3}. \tag{19}$$

Hence, for $\theta = \text{const}$, $N_I \propto 1/\mathsf{M}_{\min} \propto N$, while a shallower scaling results if $\theta(\mathsf{M})$ increases toward smaller masses. Empirically we find for $N > 30{,}000$ that the CPU time used by the force computation alone (without tree building) is very well fit by the power law $\propto N^{0.929\pm0.001}$.

5.2. Comparison with Other Methods Used in Stellar Dynamics

For various computational techniques used in stellar dynamics, Fig. 4 plots N versus the CPU time consumption normalized by N, both on logarithmic scales. An ordinary direct method running on a general-purpose computer is slower than our code for any $N \gtrsim 100$. The GRAPE-5 system [14] obtains an $\sim$100 times higher performance by wiring elementary gravity into special-purpose hardware and, for $N \lesssim 10^4$, is faster than any other method.

The new code presented here is the fastest method running solely on general-purpose computers and the only one faster than $\mathcal{O}(N \log N)$. In particular, it out-performs the popular tree code by a factor of 10 and more. At $30{,}000 \lesssim N \lesssim 3 \times 10^7$, the only technique that requires less CPU time is a combination of the tree code with a GRAPE-5 board [14, 15]. Here, the speedup due to the usage of special-purpose hardware does not quite reach that for direct methods, because of tree building and other overheads that cannot be done on the GRAPE board.

5.3. Comparison with Fast Multipole Methods

Because our code relies on cell–cell interactions, it may be considered a variant of FMM, introduced by Greengard and Rokhlin [2, 3]. However, it differs in several ways from most implementations of FMM: (i) the expansions are centered on the cells' centers of mass instead of the geometrical centers; (ii) a Cartesian Taylor series is used instead of an expansion in spherical harmonics; (iii) the interaction partners are determined by a multipole acceptance criterion rather than by their mutual grid position; (iv) the mutuality

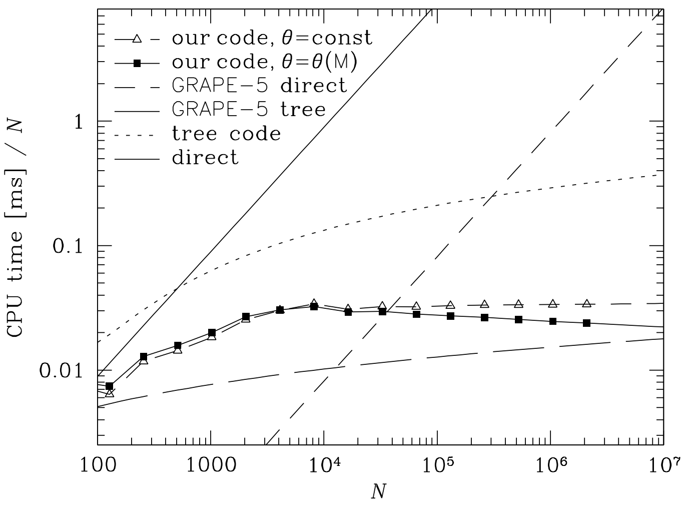

FIG. 4. CPU time per body versus N for various techniques. The direct, tree, and our codes were all compiled and run with the same equipment. For the new code, we use $\theta = 0.65$ and $\theta_{\min} = 0.5$, while for the tree code $\theta = 0.8$ and $p = 3$. The scalings for the GRAPE-5 methods are taken from [14]. The accuracy requirements for the approximate (nondirect) methods are adapted for astrophysical applications.

of interactions is fully exploited; and (v) the expansion order p is fixed and the accuracy controlled by the parameter θ.

To assess the effect of these differences, we compared our code directly with the 3D adaptive FMM code by Cheng *et al.* [4]. We performed a test identical to one reported by these authors (N bodies randomly distributed in a cube) on an identical computer (a Sun UltraSPARC with 167MHz) using the same error measure (Eq. (57) of [4]),

$$E = \left[\sum_i (\Phi_i - \tilde{\Phi}_i)^2 \Big/ \sum_i \Phi_i^2\right]^{1/2}, \tag{20}$$

where Φ and $\tilde{\Phi}$ are the potential computed by direct summation and the approximate method, respectively (unfortunately, Cheng *et al.* do not give the more relevant error of the accelerations). Table II gives, in the last three columns, the CPU time (T_{approx}) in seconds and error E for our code with $\theta = 1$ and $s = 6$ as well as the time needed for direct summation in 64-bit precision (T_{direct}); columns 2–4 report the data from Table I of [4]. On average our code is faster by more than a factor of 10 and twice as accurate (even though we compromised the approximation of the potential by omitting the octopole contributions).

What causes this enormous difference? Since for the direct summation the timings are much more similar, we can rule out differences in hardware, compiler, and so forth, as a cause. An important clue is the fact that our code cannot compete with the Cheng *et al.* FMM in the regime $E \lesssim 10^{-6}$. While our code tries to reach this goal by decreasing θ, FMM obtains it by increasing p. In general, the accuracy as well as the performance are controlled by both the order p and the choice of interaction partners, parameterized in our code by θ. Hence, maximal efficiency at given accuracy is obtained at a unique choice of (p, θ). Apparently, low orders of p are optimal for low accuracies, while high accuracies are most efficiently obtained with high orders, instead of an increasing number of interactions (decreasing θ).

TABLE II
Comparison with FMM: Timing Results for Bodies Uniformly Distributed in a Cube

N	T^a_{FMM}	T^a_{direct}	E^a	T^b_{approx}	T^c_{direct}	E^b
20,000	13.3	233	7.9×10^{-4}	0.97	136	3.7×10^{-4}
50,000	27.7	1,483	5.2×10^{-4}	2.64	924	3.3×10^{-4}
200,000	158	24,330	8.4×10^{-4}	10.77	14,694	3.4×10^{-4}
500,000	268	138,380	7.0×10^{-4}	29.42	91,134	3.7×10^{-4}
1,000,000	655	563,900	7.1×10^{-4}	58.34	366,218	3.5×10^{-4}

[a] Using FMM; data from Table I of Cheng *et al.* [4].
[b] Using the code presented here on a computer identical to that used by Cheng *et al.*
[c] Using our own implementation of direct summation on the same computer.

Thus, traditional FMM is less useful in the low-accuracy regime, such as is needed in stellar dynamics, in agreement with earlier findings [5], and our code may be called a variant of FMM optimized for low accuracy. Clearly, however, a code for which p and θ can be adapted simultaneously would be superior to both.

6. DISCUSSION AND SUMMARY

Our code for the approximate computation of mutual long-range forces between N bodies extends the traditional Barnes and Hut [1] tree code by including cell–cell interactions, similar to fast multipole methods (FMM). However, unlike most implementations of FMM, our code is optimized for comparably low accuracy, which is sufficient in stellar dynamical applications (see Section 5.3).

As a unique feature, our code exploits the mutual character of gravity: both nodes of any interaction are sink and source simultaneously. This results in exact conservation of Newton's third law and substantially reduces the computational effort, but requires a novel tree-walking algorithm (see Section 3.2) which preserves the natural symmetry of each interaction. Note that the generic algorithm given in Section 3.2 is not restricted to long-range force approximations but can be used for any task that incorporates mutuality, for instance neighbor and collision-partner searching.

6.1. Complexity

The new code requires $\mathcal{O}(N)$ or less operations (Fig. 3) for the approximate computation of the forces of N mutually gravitating bodies. For stellar dynamicists, this is the first competitive code better than $\mathcal{O}(N \log N)$. A complexity of $\mathcal{O}(N)$ was expected for methods based on cell sinks (implying cell–cell interactions) [2, 4, 7] but, to my knowledge, hardly ever shown empirically in three dimensions.

Our code obtains a complexity of less than $\mathcal{O}(N)$ by employing a mass-dependent tolerance parameter θ. The traditional $\theta = \text{constant}$ results in equal *relative* errors of each interaction, such that the total error of any body's force is dominated by the interactions with the most massive cells. By slightly increasing the tolerance parameter for less massive cells, we obtain (approximately) equal *absolute* errors, resulting in a total force error lower than that in the traditional method for the same number of interactions. Thus, at the same error, we require, fewer interactions. The additional interactions, arising when increasing N, occur at ever less massive cells and hence at ever larger tolerance parameters. This causes the computational costs to rise sublinearly with $N \gtrsim 10^4$ (depending on the accuracy requirements).

6.2. Performance

We have shown that on general-purpose computers our code out-performs any competitor code commonly used in the field of stellar dynamics. A recent adaptive 3D implementation of FMM [4] is also out-performed by a factor of 10 (see Section 5.3), which is related to the fact that traditional FMM codes appear to be good only in the high-accuracy regime. The code presented here was optimized for low accuracies, but by increasing the expansion order p, one can easily obtain a version suitable for high accuracies, and it remains to be seen how it would perform compared to traditional FMM.

Currently, the only faster method appears to be a GRAPE-supported tree code [15], which uses the special-purpose GRAPE hardware [14]. Unfortunately, unlike the tree code, our code cannot be combined with the current GRAPE hardware. There are, however, no conceptual obstacles against hard-wiring Eq. (16) into special-purpose hardware, which should yield a speedup comparable to that of tree to GRAPE tree, i.e., a factor of $\sim$50.

6.3. Publication of the Code

Our code is written in C++, also includes a purely two-dimensional version (not described here), and is linkable to C and FORTRAN programs. The code is publicly available from the author upon request.

A full N-body code based on this force algorithm is available under the NEMO [16] package (http://bima.astro.umd.edu/nemo).

APPENDIX A

Here, we give an error estimate for the Taylor series approximation of gravity. Using the integral form of the Taylor-series remainder, we find for the remainder in Eq. (2), with $\boldsymbol{\Delta} \equiv (\boldsymbol{x} - \boldsymbol{z}_{\mathrm{A}}) - (\boldsymbol{y} - \boldsymbol{z}_{\mathrm{B}}) = \boldsymbol{x} - \boldsymbol{y} - \boldsymbol{R}$,

$$\mathcal{R}_p(g) = \frac{\boldsymbol{\Delta}^{(p+1)}}{p!} \odot \int_0^1 dt\,(1-t)^p \boldsymbol{\nabla}^{(p+1)} g(\boldsymbol{R} + \boldsymbol{\Delta} t), \tag{A.1}$$

$$\boldsymbol{\nabla}\mathcal{R}_p(g) = \frac{\boldsymbol{\Delta}^{(p)}}{(p-1)!} \odot \int_0^1 dt\,(1-t)^{p-1}\, \boldsymbol{\nabla}^{(p+1)} g(\boldsymbol{R} + \boldsymbol{\Delta} t). \tag{A.2}$$

For Newtonian gravity,

$$\left\| \frac{1}{p!} \int_0^1 dt\,(1-t)^p\, \boldsymbol{\nabla}^{(p+1)} g(\boldsymbol{R} + \boldsymbol{\Delta} t) \right\| \le \frac{1}{(R - |\boldsymbol{\Delta}|)\, R^{p+1}}, \tag{A.3}$$

$$\left\| \frac{1}{(p-1)!} \int_0^1 dt\,(1-t)^{p-1}\, \boldsymbol{\nabla}^{(p+1)} g(\boldsymbol{R} + \boldsymbol{\Delta} t) \right\| \le \frac{(p+1)R - p|\boldsymbol{\Delta}|}{(R - |\boldsymbol{\Delta}|)^2\, R^{p+1}}. \tag{A.4}$$

For the summation over the source cell, we find

$$\left\| \sum_{y_i \in \mathrm{B}} \mu_i \boldsymbol{\Delta}^{(p)} \right\| \leq \sum_{k=0}^{p} \binom{p}{k} r_{\mathrm{A,max}}^{k} \left\| \mathsf{M}_{\mathrm{B}}^{(p-k)} \right\| \leq (r_{\mathrm{A,max}} + r_{\mathrm{B,max}})^p \mathsf{M}_{\mathrm{B}}, \tag{A.5}$$

with M_{B} the mass of cell B. With $\theta > (r_{\mathrm{A,max}} + r_{\mathrm{B,max}})/R$, we finally get

$$|\mathcal{R}_p(\Phi_{\mathrm{B}\to\mathrm{A}})| \leq \frac{\theta^{p+1}}{1-\theta} \frac{\mathsf{M}_{\mathrm{B}}}{R}, \tag{A.6}$$

$$|\mathcal{R}_p(\boldsymbol{\nabla}\Phi_{\mathrm{B}\to\mathrm{A}})| \leq \frac{(p+1)\theta^{p}}{(1-\theta)^2} \frac{\mathsf{M}_{\mathrm{B}}}{R^2}. \tag{A.7}$$

APPENDIX B

Here, we give the interaction details for Algorithm 1 in Section 3.2.1. Mutual body–body interactions are done by elementary gravity, while body self-interactions are ignored. Mutual interactions between nodes containing N_1 and N_2 bodies are treated as follows.

1. If $N_1 N_2 < N_{\mathrm{nn}}^{\mathrm{pre}}$, execute the interaction by direct summation; otherwise,
2. if the interaction is well-separated, execute it using Eq. (6); otherwise,
3. if $N_1 N_2 < N_{\mathrm{nn}}^{\mathrm{post}}$, execute the interaction by direct summation; otherwise,
4. the interaction cannot be executed but must be split.

Here, $N_{\mathrm{nn}}^{\mathrm{pre}}$ and $N_{\mathrm{nn}}^{\mathrm{post}}$ have different values depending on whether it is a cell–body (cb) or cell–cell (cc) interaction (see below). Cell self-interactions are done slightly differently.

1. If $N_1 < N_{\mathrm{cs}}$, execute the interaction by direct summation; otherwise,
2. the interaction cannot be executed but must be split.

The numbers $N_{\mathrm{cb}}^{\mathrm{pre}}$, $N_{\mathrm{cb}}^{\mathrm{post}}$, $N_{\mathrm{cc}}^{\mathrm{pre}}$, $N_{\mathrm{cc}}^{\mathrm{post}}$, and N_{cs} determine the usage of direct summation. After some experiments, I found the following values to result in the most efficient code at a given accuracy (but this certainly depends on the implementation details).

$$\begin{aligned} N_{\mathrm{cb}}^{\mathrm{pre}} &= 3, \quad N_{\mathrm{cc}}^{\mathrm{pre}} = 0, \\ N_{\mathrm{cb}}^{\mathrm{post}} &= 128, \quad N_{\mathrm{cc}}^{\mathrm{post}} = 16, \quad N_{\mathrm{cs}} = 64. \end{aligned} \tag{B.1}$$

Note that cell–body interactions with as many as 128 bodies in the cell hardly ever occur, since the interaction algorithm favors interactions between roughly equally sized nodes. Thus, cell–body interactions will almost always be executed.

ACKNOWLEDGMENTS

I thank J. Makino from Tokoy University for helpful discussions and R. Ibata and C. Pichon from Strasbourg Observatory for hardware support.

REFERENCES

1. J. E. Barnes and P. Hut, A hierarchical $\mathcal{O}(N \log N)$ force calculation algorithm, *Nature* **324**, 446 (1986).
2. L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, *J. Comput. Phys.* **73**, 325 (1987).

3. L. Greengard and V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, *Acta Numer.* **6**, 229 (1997).
4. H. Cheng, L. Greengard, and V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, *J. Comput. Phys.* **155**, 468 (1999).
5. R. Capuzzo-Colcetta and P. Miocchi, A comparison between the fast multipole algorithm and the tree code to evaluate gravitational forces in 3D, *J. Comput. Phys.* **143**, 29 (1998).
6. W. Dehnen, A very fast and momentum-conserving tree code, *Astrophys. J.* **536**, L39 (2000).
7. M. S. Warren and J. K. Salmon, A portable parallel particle program, *Comput. Phys. Commun.* **87**, 266 (1995).
8. J. K. Salmon and M. S. Warren, Skeletons from the tree code closet, *J. Comput. Phys.* **111**, 136 (1994).
9. W. Benz, R. L. Bowers, A. Q. W. Cameron, and W. H. Press, Dynamic mass exchange in doubly degenerate binaries. I. 0.9 and 1.2 $M_\odot$ stars, *Astrophys. J.* **348**, 647 (1990).
10. D. Merritt, Optimal smoothing for N-body codes, *Astron. J.* **111**, 2462 (1996).
11. W. Dehnen, Towards optimal softening in three-dimensional N-body codes—I. Minimizing the force error, *Mon. Not. R. Astron. Soc.* **324**, 273 (2001).
12. L. Hernquist, An analytic model for spherical galaxies and bulges, *Astrophys. J.* **356**, 359 (1990).
13. E. Athanassoula, E. Fady, J. C. Lambert, and A. Bosma, Optimal softening for force calculations in collisionless N-body simulations, *Mon. Not. R. Astron. Soc.* **314**, 475 (2000).
14. A. Kawai, T. Fukushige, J. Makino, and M. Taiji, GRAPE-5: A special purpose computer for N-body simulations, *Publ. Astron. Soc. Jpn.* **52**, 659 (2000).
15. J. Makino, A tree code with special purpose processor, *Publ. Astron. Soc. Jpn.* **43**, 621 (1991).
16. P. J. Teuben, The stellar dynamics toolbox NEMO, in *Astronomical Data Analysis Software and Systems IV*, edited by R. Shaw, H. E. Payne, and J. J. E. Hayes, ASP Conf. Series (1995), Vol. 77, p. 398.