John von Neumann Institute for Computing


# Long-Range Interactions in Many-Particle Simulation

Paul Gibbon, Godehard Sutmann

published in

*Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms*, Lecture Notes,
J. Grotendorst, D. Marx, A. Muramatsu (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **10**, ISBN 3-00-009057-6, pp. 467-506, 2002.



http://www.fz-juelich.de/nic-series/volume10

# Long-Range Interactions in Many-Particle Simulation

**Paul Gibbon and Godehard Sutmann**

John von Neumann Institute for Computing
Central Institute for Applied Mathematics
Research Centre Jülich, 52425 Jülich, Germany
*E-mail: {p.gibbon, g.sutmann}@fz-juelich.de*

Numerical algorithms for accelerating the computation of $N$-body problems dominated by long-range inter-particle forces are reviewed. For periodic systems, the optimised Ewald lattice sum with an $O(N^{3/2})$ scaling forms a reference standard against which the newer, potentially faster Particle-Particle Particle-Mesh and Fast Multipole Methods can be measured. The more general $N$-body problem with arbitrary boundary conditions is also described, in which various multipole methods are now routinely used instead of direct summation for particle numbers in excess of $10^4$. These techniques are described in a tutorial fashion and rough comparisons given of their respective computational performance on scalar and parallel machine architectures.

## 1 Introduction

Until relatively recently, the general solution of the N-body problem – computing the trajectories of many mutually interacting particles – was considered intractable, except for small systems, or for particle assemblies in which the interaction potential is either physically or artificially truncated. Over the past half century, however, our definition of 'small' has stretched from a few dozen to several thousand bodies, thanks to advances in computing power. On the other hand, this increase in manageable system size is *not* as dramatic as one might expect from Moore's 'Law', in which processing power has doubled every 18 months or so since the 1960s.

A brief inspection of the typical N-body force law will reveals why this is so. Consider a classical system of $N$ bodies with charges $q_i$ and masses $m_i$ interacting via a central potential $V$:

$$m_i \frac{d^2\mathbf{r}_i}{dt^2} = -q_i \nabla_i V \qquad\qquad i = 1, 2, 3 \ldots N, \tag{1}$$

where

$$V(\boldsymbol{r}_i) = \sum_{j \neq i}^{N} \frac{q_j}{\mid \boldsymbol{r}_i - \boldsymbol{r}_j \mid}. \tag{2}$$

To compute one iteration of the ensemble trajectory $\boldsymbol{r}_i(t)$ described by the equation of motion (1), we require $N(N-1)$ operations. With the aid of Newton's third law (action=reaction), we can exploit the symmetry of the potential to reduce the operation count by one half, but this still leaves us with an asymptotic scaling of $O(N^2)$ for large $N$. In other words, we need a 100-fold increase in computing power in order to increase the simulation size by an order of magnitude. This dispiriting fact of $N$-body life held up large-scale simulation of many-particle systems until the early 1980s, when a number of *algorithmic* advances reduced the computational effort to complexities ranging from $O(N^{3/2})$ down to $O(N)$, depending on the context of the problem.

In this article we present a tutorial survey of these techniques, which can be broadly classified into three categories: (i) Ewald summation, (ii) particle-mesh methods, and (iii) hierarchical or multipole methods. The Ewald method[1] is restricted to fully or partially periodic systems, but has been widely adopted for studies of condensed matter – ionic salts, molecules in solvent etc. – where it is important to eliminate surface effects which would arise in a small, isolated system. Particle-mesh codes[2,3] are actually more widespread outside the MD community, especially in astrophysics, plasma physics and electrical engineering, but form a vital component of the so-called Particle-Mesh-Ewald (PME) method developed some 10 years ago by Darden[4]. Multipole methods[5], which come in two main flavours – 'Fast Multipole Methods' and 'Tree-Codes' respectively – are based on the observation that distant charges (or masses, in the case of gravity) may be grouped together and substituted by a single multipole expansion, leading to a substantial saving in the number of interactions necessary to sum the potential or force.

All of these techniques for accelerating $N$-body force summation have recently been subjected to intense re-examination in order to produce codes suitable for parallel computer architectures. In the final section, some of the important design considerations for $N$-body simulation on parallel machines will be discussed, and an attempt is made to compare the relative performance of the most commonly used methods.

## 2 Ewald Summation

The technique of Ewald summation is hugely popular in contemporary molecular dynamics simulation, even though it applies to a special case: namely, periodic systems. By this we mean that the simulation region or 'cell' is effectively replicated in all spatial directions, so that particles leaving the cell reappear at the opposite boundary. For systems governed by a short-ranged potential – say Lennard-Jones or hard spheres – it is sufficient to take just the neighbouring simulation volumes into account, leading to the 'minimum-image' configuration shown in Fig. 1. The potential seen by the particle at $\boldsymbol{r}_i$ is summed over all other particles $\boldsymbol{r}_j$, or their periodic images $(\boldsymbol{r}_j \pm \boldsymbol{n})$, where $\boldsymbol{n} = (i_{\hat{x}}, i_{\hat{y}}, i_{\hat{z}})L$, with $i_\alpha = 0, \pm 1, \pm 2 ... \pm \infty$, whichever is closest. More typically, this list is further restricted to particles lying within a sphere centred on $\boldsymbol{r}_i$[6]. For long-range potentials, this arrangement is inadequate because the contributions from more distant images at $2L, 3L$ etc., are no longer negligible. One might argue that these contributions should more-or-less cancel, which they nearly do, but one has to take care in which order to do the sum: a simple example serves to illustrate the problem. Consider a system of two oppositely charged ions, periodically extended to form an infinite one-dimensional line of charges, each separated by a distance $R$ – Fig. 2. The potential energy of the reference ion with charge $-q$ is:

$$\begin{aligned} U &= -2q^2 \left( \frac{1}{R} - \frac{1}{2R} + \frac{1}{3R} - \frac{1}{4R} ... \right) \\ &= -\frac{2q^2}{R} \left( 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} ... \right) \\ &= -\frac{2q^2}{R} \log 2 \end{aligned} \tag{3}$$

The factor $2 \log 2$ is the Madelung constant, which is of central importance in the theory of ionic crystals[7,8]. The series in (3) is actually *conditionally convergent*; the result depends

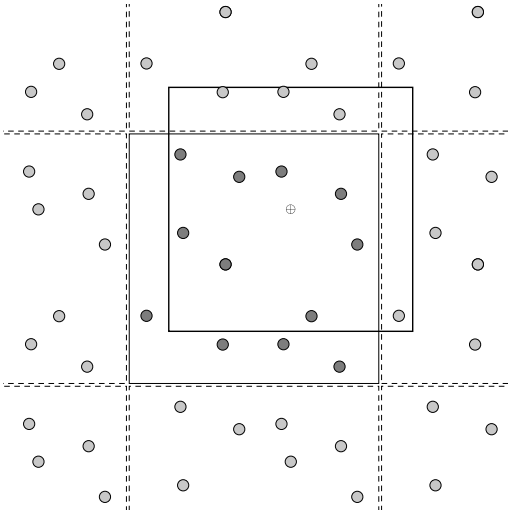


Figure 1. Periodic boundary conditions for simulation region (centre, dark-shaded particles at positions $\boldsymbol{r}_j$), showing 'minimum-image' box for reference ion $\oplus$ at position $\boldsymbol{r}_i$ containing nearest periodic images (light-shaded particles at positions $\boldsymbol{r}_j \pm \boldsymbol{n}$).

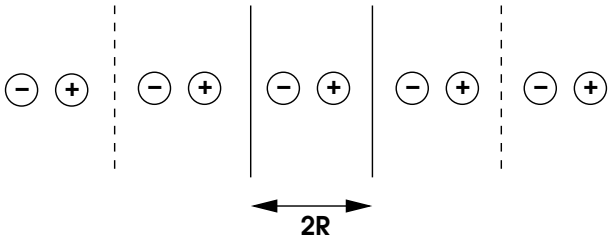



Figure 2. Infinite 1D lattice

on the summation order. To illustrate this, we can choose a different ordering, for example:

$$\begin{aligned}
&1 \quad + \frac{1}{3} - \frac{1}{2} + \frac{1}{5} \qquad + \frac{1}{7} - \frac{1}{4} + \frac{1}{9} \quad + \ldots \\
= 1 \quad &- \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} \quad - \frac{1}{6} + \frac{1}{7} - \frac{1}{8} + \frac{1}{9} \quad + \ldots \\
&+ \frac{1}{2} \qquad - \frac{1}{4} \qquad + \frac{1}{6} \qquad - \frac{1}{8} + \ldots \\
= \sum_n \frac{(-1)^{n-1}}{n} &+ \frac{1}{2} \sum_n \frac{(-1)^{n-1}}{n} = \frac{3}{2} \log 2,
\end{aligned}$$

giving us 50% more potential energy than we had before!

In three dimensions, determination of the Madelung constant – and hence the lattice potential energy – is non-trivial because successive terms in the series must be arranged so that positive and negative contributions nearly cancel. This is exactly the problem we are faced with when evaluating the potential on our reference ion in Fig. 1, albeit for an irregular assortment of charges: in what order should we sum over image boxes?

An intuitive and elegant way of doing this is to build up sets of images contained within successively larger spheres surrounding the simulation region[25] – Fig. 3. According to this

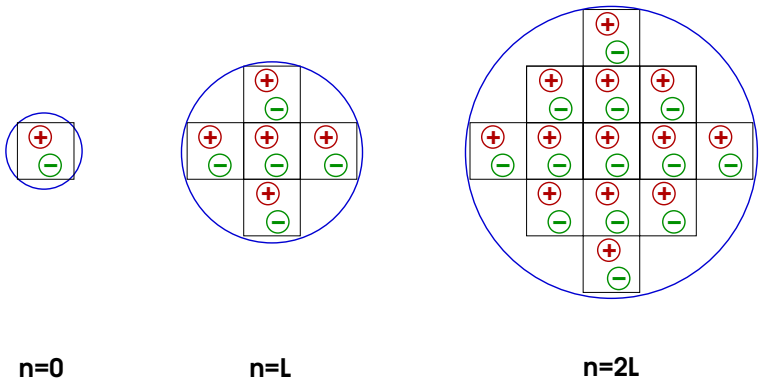



Figure 3. Constructing a convergent sum over periodic images. (Adapted from Allen & Tildesley)

scheme, the potential is expressed mathematically as:

$$V_s(r_i) = \sum_{\boldsymbol{n}} {}' \sum_{j=1}^{N} \frac{q_j}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid}, \tag{4}$$

where $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ and $\boldsymbol{n} = (i_{\hat{x}}, i_{\hat{y}}, i_{\hat{z}})L$, with $i_\alpha = 0, \pm 1, \pm 2 ... \pm \infty$ as before. The prime in the sum over $\boldsymbol{n}$ indicates that the $j = i$ term is omitted for the primary cell $\boldsymbol{n} = 0$. Taking the image cells in the order prescribed by Fig. 3 does ensure that the sum (4) converges to the correct value, but only slowly. Strictly speaking, (4) is a conditionally convergent series; by contrast, potentials with a radial fall-off steeper than $\sim r^{-3}$ are *absolutely* convergent[9].

### 2.1 Periodic Lattice Sum

As it stands, the summation over image boxes implied by (4) makes the prospects of speeding up our $N$-body problem look rather grim: we have turned an $O(N^2)$ problem into one requiring $N_{box} \times N^2$ operations! Ewald got around this by recasting the potential as the sum of two rapidly converging series: one in real space; the other in reciprocal, or $k$-space:

$$V_E(r_i) = \sum_{\boldsymbol{n}}{}' \sum_{j=1}^{N} q_j \frac{\mathrm{erfc}(\alpha \mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid)}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid} + \frac{4\pi}{L^3} \sum_{\boldsymbol{k} \neq 0} \sum_{j} q_j \exp\left(\frac{- \mid k \mid^2}{4\alpha^2}\right) \exp\{i\boldsymbol{k} \cdot (\boldsymbol{r}_j - \boldsymbol{r}_i)\} - \frac{2\alpha}{\pi^{1/2}} q_i \qquad (5)$$

The term $\alpha$ is an arbitrary, but important parameter, which governs the relative convergence rates of the two main series. The last term is a 'self-potential' which cancels an equivalent contribution in the $k$-space sum. It is not immediately obvious why the double series (5) should be equivalent to (4). However, we can begin to make physical sense of it by noting that

$$\mathrm{erfc}(x) = 1 - \frac{2}{\pi^{1/2}} \int_0^x e^{-t^2} dt.$$

Thus, what we actually have is an expression of the form:

$$V_E(r_i) = V_s(r_i) - \sum_{\boldsymbol{n}} f(\boldsymbol{n}) + \sum_{\boldsymbol{k}} g(\boldsymbol{k}). \qquad (6)$$

In other words, to get (5) from (4), we just use the trick of adding and subtracting an *additional* series, summed in real space and $k$-space respectively. In the Ewald sum, this new series is in fact the lattice sum for a Gaussian charge distribution

$$\rho(r) = A \exp(-\alpha^2 r^2). \qquad (7)$$

The first two terms in (6) combine to give the rapidly converging real-space sum in (5) – as illustrated schematically in Fig. 4.

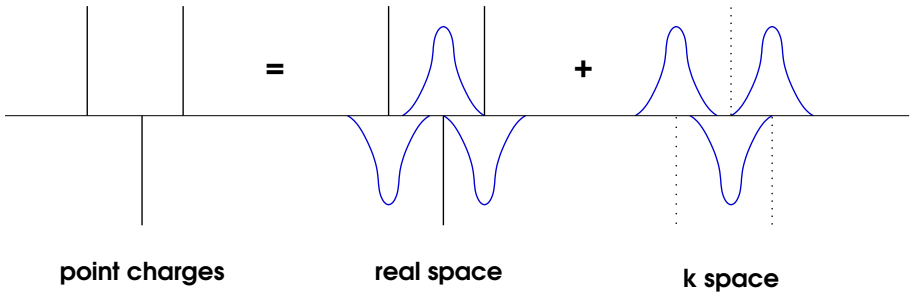



Figure 4. Splitting the sum for point charges into two rapidly convergent series for Gaussian-shaped charges.

The choice of kernel for the charge-smearing function is actually not too critical, and mainly influences the convergence characteristics of the final series. A comparison of several alternative functions was made some time ago by Heyes[10]; for tutorial purposes here, however, we will stick to the simple Gaussian distribution originally used by Ewald himself:

$$\sigma(r) = \frac{\alpha^3}{\pi^{3/2}} \exp(-\alpha^2 r^2), \qquad (8)$$

which is normalised such that

$$\int_0^\infty \sigma(r)dr = 1.$$

Note that $\alpha$ determines the width and height of the spreading function, and hence the effective size of the charges.

Let us begin with the real-space sum depicted in Fig. 4. To obtain this, we just subtract the lattice sum for the smeared-out charges from the original point-charge sum, thus:

$$\begin{aligned} V_r(r_i) &= \sum_{\boldsymbol{n}}{}' \sum_{j=1}^{N} \frac{q_j}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid} \left[ 1 - \int_0^\infty \sigma(r - r_{ij}) d^3r \right] \\ &= \sum_{\boldsymbol{n}}{}' \sum_{j} q_j \left[ \frac{1}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid} - \frac{4\alpha^3}{\pi^{1/2} \mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid} \int_0^{|\boldsymbol{r}_{ij}+\boldsymbol{n}|} r^2 \exp(-\alpha^2 r^2) dr \right. \\ &\quad \left. - \frac{4\alpha^3}{\pi^{1/2}} \int_{|\boldsymbol{r}_{ij}+\boldsymbol{n}|}^\infty r \exp(-\alpha^2 r^2) dr \right] \end{aligned}$$

The second term in the above expression can be integrated by parts to give an error function $\mathrm{erf}(\alpha|\boldsymbol{r}_{ij} + \boldsymbol{n}|)$, plus a term which exactly cancels the third term. Carrying out this simplification, we finally get:

$$V_r(r_i) = \sum_{\boldsymbol{n}}{}' \sum_{j=1}^{N} q_j \frac{\mathrm{erfc}(\alpha \mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid)}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid}. \tag{9}$$

The reciprocal-space sum takes a little more work. First, we consider the charge density of the whole lattice at some arbitrary position $r$:

$$\rho(r) = \sum_j q_j \delta(r - r_j). \tag{10}$$

Since the lattice is periodic, we can express this equivalently as a Fourier sum:

$$\rho(r) = L^{-3} \sum_j \sum_{\boldsymbol{k}} f(\boldsymbol{k}) \exp(-i\boldsymbol{k} \cdot \boldsymbol{r}), \tag{11}$$

where $\boldsymbol{k} = 2\pi/L(i_{\hat{x}}, i_{\hat{y}}, i_{\hat{z}})$; $i_\alpha = 0, 1, 2, ..$etc., and

$$f(\boldsymbol{k}) = \int_{L^3} \rho(r) \exp(i\boldsymbol{k} \cdot \boldsymbol{r}) d^3r, \tag{12}$$

where the integration is now restricted to the unit cell volume $V = L^3$. Substituting (10) into (12) and making use of a standard identity picks out modes corresponding to the point charges:

$$f(\boldsymbol{k}) = \sum_j q_j \exp(i\boldsymbol{k} \cdot \boldsymbol{r}_j) \tag{13}$$

Turning now to the *smeared* charge distribution:

$$\begin{aligned} \rho'(r) &= \sum_j q_j \sigma(r - r_j) \\ &= \int_V \rho(r - r') \sigma(r') d^3r', \end{aligned} \tag{14}$$

we observe that this is just the convolution of $\rho(r)$ with $\sigma(r)$, which we know can be expressed in Fourier space as[11]:

$$\rho'(r) = \frac{1}{L^3} \sum_{\boldsymbol{k}}{}' f(\boldsymbol{k})\phi(\boldsymbol{k},\alpha)\exp(-i\boldsymbol{k}\cdot\boldsymbol{r}), \tag{15}$$

where $\phi(\boldsymbol{k},\alpha)$ is the Fourier transform of the charge-smearing function $\sigma(r)$, i.e.:

$$\phi(\boldsymbol{k},\alpha) = \exp\left(\frac{-\mid \boldsymbol{k} \mid^2}{4\alpha^2}\right). \tag{16}$$

We are now equipped to express the potential due to the smeared charges in $k$-space. At the reference position $r_i$, this is:

$$\begin{aligned} V_k(r_i) &= \int_0^\infty \frac{\rho'(r_i + r)}{r} d^3r \\ &= \frac{1}{L^3} \sum_{\boldsymbol{k}}{}' f(\boldsymbol{k})\phi(\boldsymbol{k},\alpha)\exp(-i\boldsymbol{k}\cdot\boldsymbol{r}_i) \int_0^\infty \frac{\exp(-i\boldsymbol{k}\cdot\boldsymbol{r})}{r} d^3r. \end{aligned}$$

The integral on the right of the above expression is a standard one[12], and evaluates to $4\pi/k^2$. Combining this with the earlier results (13) and (16) for $f(\boldsymbol{k})$ and $\phi(\boldsymbol{k},\alpha)$ respectively, we have finally:

$$V_k(r_i) = \frac{4\pi}{L^3} \sum_{\boldsymbol{k}}{}' \sum_j q_j \exp\{i\boldsymbol{k}\cdot(\boldsymbol{r}_j - \boldsymbol{r}_i)\} \frac{\exp\left(\frac{-|\boldsymbol{k}|^2}{4\alpha^2}\right)}{\mid \boldsymbol{k} \mid^2}. \tag{17}$$

This potential includes an unphysical 'self-term' corresponding to a smeared out charge centered at $r_i$, which needs to be subtracted off:

$$\begin{aligned} V_s(r_i) &= q_i \int_0^\infty \sigma(r) d^3r \\ &= \frac{4\pi q_i \alpha^3}{\pi^{3/2}} i \int_0^\infty r\exp(-\alpha^2 r^2) \\ &= \frac{2\alpha}{\pi^{1/2}} q_i. \end{aligned} \tag{18}$$

Adding together our partial results (9), (17) and (18), we recover the Ewald sum quoted before in Equation 5. The equivalent expression for the force (or more correctly the electric field) can be found by direct differentiation with respect to the vector between the reference particle $i$ and particle $j$:

$$\begin{aligned} \boldsymbol{E}(\boldsymbol{r}_i) &= -\frac{\partial V_E(\boldsymbol{r}_i)}{\partial \boldsymbol{r}_{ij}} \\ &= \sum_{\boldsymbol{n}}{}' \sum_j \frac{q_j \boldsymbol{r}_{ij,\boldsymbol{n}}}{r^3_{ij,\boldsymbol{n}}} \left[\mathrm{erfc}(\alpha r_{ij,\boldsymbol{n}}) + \frac{2\alpha r_{ij,\boldsymbol{n}}}{\sqrt{\pi}} \exp(-\alpha^2 r^2_{ij,\boldsymbol{n}})\right] \\ &+ \frac{4\pi}{L^3} \sum_{\mathbf{k}\neq 0} \sum_j q_j \frac{\boldsymbol{k}}{k^2} \exp\left(\frac{-k^2}{4\alpha^2}\right) \sin\left(\boldsymbol{k}\cdot\boldsymbol{r}_{ji}\right). \end{aligned} \tag{19}$$

In the above expression, we have used the shorthand notation $\boldsymbol{r}_{ij,\boldsymbol{n}} \equiv \boldsymbol{r}_{ij} + \boldsymbol{n}$ and $\boldsymbol{r}_{ji} \equiv \boldsymbol{r}_j - \boldsymbol{r}_i$. More sophisticated derivations of lattice sums can be found in Leeuw *et al.*[9], who consider more general systems surrounded by a dielectric medium, by Heyes[10], who considers an arbitrary charge-spreading function, and by Perram *et al.*[13], who derive expressions for other types of potential (force-laws). A detailed analysis of the cutoff errors incurred by real-space and $k$-space sums has been made by Kolafa and Perram[14]; for the special 2D case by Solvason *et al.*[15].

### 2.2 Scaling

In replacing (4) by (5) we immediately reap the benefits of rapid convergence. This can be seen more clearly when we make use of the previous results to compute the total potential energy of the system, summing over all charges, $q_i$:

$$\begin{aligned}\Phi_T &= \frac{1}{2}\sum_i q_i[V_r(r_i) + V_k(r_i) - V_s(r_i)] \\ &= \frac{1}{2}\sum_{\boldsymbol{n}}{}' \sum_{i=1}^{N}\sum_{j=1}^{N} q_i q_j \frac{\mathrm{erfc}(\alpha \mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid)}{\mid \boldsymbol{r}_{ij} + \boldsymbol{n} \mid} \\ &\quad \frac{2\pi}{L^3}\sum_{\boldsymbol{k}}{}' \sum_{i=1}^{N}\sum_{j=1}^{N} q_i q_j \exp\{i\boldsymbol{k}\cdot(\boldsymbol{r}_j - \boldsymbol{r}_i)\}\frac{\exp\left(\frac{-|\boldsymbol{k}|^2}{4\alpha^2}\right)}{\mid \boldsymbol{k} \mid^2} - \frac{\alpha}{\pi^{1/2}}\sum_{i=1}^{N} q_i^2. \quad (20)\end{aligned}$$

Simple experimentation with the Ewald sums[16] soon reveals a range of parameters in which one or both of the partial sums can be restricted. The example in Fig. 5, constructed for 40 randomly distributed positive and negative charges, shows that neither real-space nor $k$-space parts can be neglected in the range $\alpha L = 1$–10, even though large summation volumes were taken: $|\boldsymbol{n}|_{max} = 12, h^2 = (k_{max}/2\pi)^2 = 700$, on each side. Although we will come to the question of optimisation shortly, we can also get an idea of where to make cutoffs by successively truncating the two sums – Fig. 6. Inspection of these curves confirms the consensus choice found in the literature of $\alpha L \sim$ 2–5, which allows the real-space sum to be restricted to a couple of box lengths ($\mid \boldsymbol{r}_j + \boldsymbol{n} \mid \leq 2L$), while maintaining reasonable accuracy. In fact, for the curve $n = 2, h^2 = 100$, the potential energy is accurate to better than $10^{-6}$ in the range $\alpha L = 2$ to $\alpha L = 9$.

The qualitative observations above still do not tell us how the overall computational effort scales with $N$, because the cutoff point in both sums may vary. Fixing $\mid \boldsymbol{r}_j + \boldsymbol{n} \mid \leq L$ – i.e., adopting the minimum image convention – would of course lead to an $O(N^2)$ scaling once more. The arbitrariness of the parameter $\alpha$ raises the question of whether one can choose the cutoffs $n_{max}$ and $k_{max}$ in either sum to reduce the overall effort. This seems a tall order, but just such a recipe was derived by Perram *et al.*[13], who showed that there does exist an optimal choice of parameters which reduces the scaling to $O(N^{3/2})$.

The trick is to weight the summation towards the $k$-space sum, thereby restricting the number of particle pairs which have to be considered in real space. Fincham[17] gives an intuitive proof of Perram's $N^{3/2}$ scaling, which we reproduce here. First, we suppose that both sums are to converge to some small value, depending on the accuracy requirement of the application. We set this 'small' value equal to $\exp(-p)$. For the real-space sum (9),

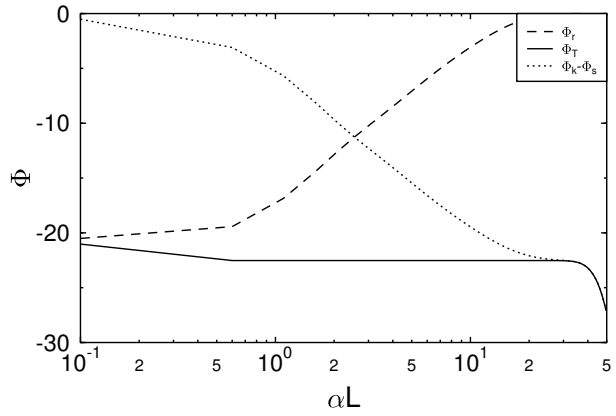



Figure 5. Convergence of real- (dashed) and $k$-space (dotted) Ewald potentials for different values of $\alpha$.

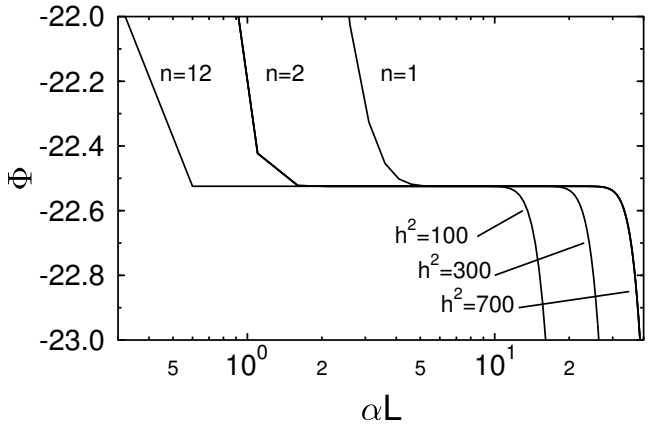



Figure 6. Ewald potential energy for different cutoffs in real- and $k$-space ($h^2 \equiv (k/2\pi)^2$).

this implies that at some cutoff radius $R$, we may write

$$\text{erfc}(\alpha r)\,\big|_{r=R} \sim \exp(-\alpha^2 R^2) = \exp(-p)\ .$$

From this we immediately obtain a constraint on $\alpha$, namely:

$$\alpha = p^{1/2}/R. \tag{21}$$

Applying the same convergence criterion to the $k$-space sum, we have for some cutoff

wave-vector $K$:

$$\exp\left(-\frac{K^2}{4\alpha^2}\right) \sim \exp(-p),$$

or

$$p = \frac{K^2}{4\alpha^2}.$$

Thus, making use of our first constraint (21), we obtain

$$K = 2\alpha p^{1/2} = \frac{2p}{R}. \tag{22}$$

Once the accuracy (via $p$) and the cutoff radius $R$ have been chosen, Equations 21 and 22 specify $K$ and $\alpha$. It remains for us to find a cutoff radius in the real-space sum that minimises the execution time. To estimate the latter, we assume that all $N$ charges are distributed uniformly in a cubic box with side $L$, so that the number density $n = N/L^3$ remains constant as $N$ is varied. The number of ions contained in a cutoff sphere is then:

$$N_c = \frac{4\pi}{3}R^3 n.$$

Hence, the execution time for the real-space sum can be approximated by:

$$T_r \simeq \frac{1}{2}N\frac{4\pi}{3}R^3 n t_r, \tag{23}$$

where $t_r$ is the time needed to evaluate a single interaction pair. For the $k$-space sum, we have a total volume, using (22), of

$$\frac{4\pi}{3}K^3 = \frac{4\pi}{3}\frac{8p^3}{R^3},$$

whereby wave-vectors are chosen according to $\boldsymbol{k} = 2\pi(i_{\hat{x}}, i_{\hat{y}}, i_{\hat{z}})/L$ and $(i_{\hat{x}}, i_{\hat{y}}, i_{\hat{z}})$ is the usual integer triple. The volume per reciprocal point is just $(2\pi/L)^3$, so the number of points in the cutoff sphere is

$$N_k = \frac{4\pi}{3}\left(\frac{p}{\pi}\right)^3 \frac{N}{nR^3}, \tag{24}$$

and the overall execution time for the $k$-space sum is then

$$T_k = \frac{1}{2}\frac{4\pi}{3}\left(\frac{p}{\pi}\right)^3 \frac{N}{nR^3}t_k. \tag{25}$$

The total time for the Ewald summation is just the sum of (23) and (25):

$$T_{tot} = \frac{1}{2}\frac{4\pi}{3}\left[NnR^3 t_r + \left(\frac{p}{\pi}\right)^3 \frac{N}{nR^3}t_k\right].$$

If we fix the accuracy requirement $p$, the only free parameter remaining is $R$. The obvious thing to do is therefore to set $dT/dR = 0$, whereupon we find:

$$R_{opt} = \left(\frac{p}{\pi}\right)^{1/2}\left(\frac{t_k}{t_r}\right)^{1/6}\frac{N^{1/6}}{n^{1/3}}, \tag{26}$$

and

$$T_{opt} = 2T_r = 2T_k = \frac{4\pi}{3} N^{3/2} \left(\frac{p}{\pi}\right)^{3/2} (t_r t_k)^{1/2}. \tag{27}$$

The optimal configuration is thus equally divided (in terms of computation time) between real- and $k$-space sums. Assuming $t_r \sim t_k$, and stipulating a fairly conservative accuracy of $\exp(-p) \sim 5 \times 10^{-5}$, or $p = \pi^2$, we find from (26)

$$R_{opt} \simeq \pi^{\frac{1}{2}} L N^{-1/6},$$

with

$$\alpha L \simeq \frac{KL}{2\pi} = \pi^{\frac{1}{2}} N^{1/6}.$$

To illustrate this with a concrete example: for a system of 10000 particles, we would choose $R = 0.38L$ – less than the minimum-image box length – and $\alpha L = 8.2$.

For small systems, say $N < 10^4$, the conventional Ewald summation technique encapsulated by Equation 5 together with the simple optimisation recipe dictated by Equations 26, 21 and 22 is widely regarded as the standard method for periodic systems. The reciprocal-space sum itself can be optimised further by exploiting symmetries to reduce the number of lattice vectors needed[16,18]. For larger systems, however, even the reduced $O(N^{3/2})$ cost-scaling eventually becomes prohibitive, forcing us to seek faster alternatives. Nevertheless, the direct Ewald method is still an important benchmark for assessing the performance of other, newer methods, some of which we will describe in the following sections.

## 3 Particle-Mesh Techniques

The deployment of a mesh or grid to speed up $N$-body calculations has long been standard practice in fields outside the traditional molecular dynamics arena, such as astrophysics and plasma physics, where the need to follow macroscopic trends over long timescales usually takes precedence over high accuracy or fine-grained spatial resolution. The term 'macroscopic' is loosely applied here to indicated that simulation charges (or masses) may represent a large number of physical entities. For example, a galaxy containing $10^{11}$ stars, each with mass $M$ equal to one solar mass $M_\odot$, can be simulated by a system of $10^4$ stars each weighing $10^7$ $M_\odot$. It turns out that this seemingly crude approximation works very well as long as one is interested in large-scale, collective effects, such as waves or instabilities with wavelengths on the same order as the system size itself. For *statistical* purposes, however, it is still highly desirable to use as many 'particles' as possible, which leaves one with the same type of computational challenge faced by someone wishing to follow the dynamics of a protein, say.

Since arbitrarily high accuracy was not (and is still not necessarily) a priority in many such applications, the $O(N)$ particle-mesh techniques, pioneered by Bunemann[19], Dawson[20], Hockney[3] and Birdsall[2] in the 1960s, quickly replaced direct summation as the workhorse computational tool in these fields.

### 3.1 Particle-in-Cell Codes

The subject of grid-based particle methods is too vast to do justice to within the confines of this review. For one thing, the representation of discrete charges on a grid introduces artificial structure factors, which in turn give rise to characteristic kinetic behaviour – such as modifications in the dielectric constant – storing up nasty surprises for the unwary. A quantitative understanding of such effects affords a certain amount of background theory, for which we heartily recommend the two definitive texts by Hockney & Eastwood[3] and Birdsall & Langdon[2]. Nonetheless, it will prove instructive to review the basic concepts of the particle-mesh (PM) method. In doing so, we hope to clarify some of the ambiguous terminology which has crept into the MD literature on this subject.

Formally, the PM method can be derived by rigorous application of kinetic theory, simplifying the $N$-body problem with $6N$ degrees of freedom via the introduction of a smooth distribution function $f(\boldsymbol{r}, \boldsymbol{v}, t)$, obeying the kinetic Vlasov-Boltzmann equation[21]:

$$\frac{\partial f}{\partial t} + \boldsymbol{v}\cdot\frac{\partial f}{\partial \boldsymbol{x}} + q\boldsymbol{E}\cdot\frac{\partial f}{\partial \boldsymbol{v}} = \left.\frac{\partial f}{\partial t}\right|_c . \tag{28}$$

The term on the RHS is a collision term, describing the transfer of momentum between particles due to close encounters. Herein lies an important difference between PM and MD: in MD, collisions are treated automatically as part of the computation, whereas in a PM code, some approximate collision *model* must be introduced. Construction of a physically sensible and computationally stable model is fiendishly difficult, and luckily need not concern us here: for the time-being we will assume that our particle system is purely collision*less*, and set $\partial f/\partial t|_c = 0$.

The distribution function $f(\boldsymbol{r}, \boldsymbol{v})$ is 6-dimensional, so the general solution of (28) is still intractable for most practical purposes. Even for problems reducible to a 1D geometry, one typically still needs to retain 2 or 3 velocity components in order to incorporate the appropriate electron motion and its coupling to Maxwell's equations, which effectively results in a 3- or 4-dimensional 'Vlasov'-code. In the particle-mesh technique, the distribution function is represented instead by a large number of discrete 'macro-particles', each carrying a fixed charge $q_i$ and mass $m_i$ – Fig. 7, usually chosen so that the charge-to-mass ratio corresponds to a physical counterpart, for example $e/m_e$. This ensures that the particle trajectories in the simulation match those which would be followed by real electrons and ions.

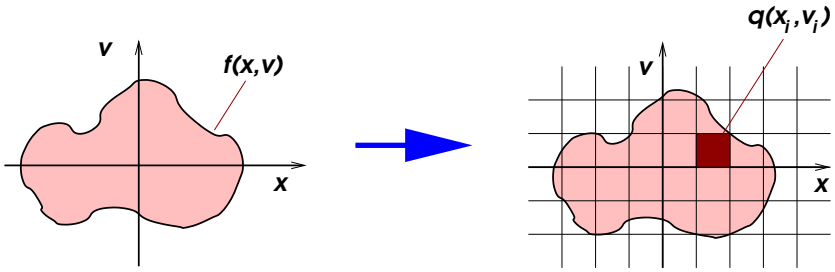



Figure 7. Correspondence between Vlasov and particle-in-cell representation of phase-space.

The particles are moved individually in Lagrangian fashion according to the equation of motion:

$$\frac{d}{dt}(\boldsymbol{v}_i) = \frac{q_i}{m_i}\boldsymbol{E} \qquad\qquad i = 1, ..., N \tag{29}$$

The density source needed to compute the electric field is obtained by mapping the local particle positions onto a grid via a weighting function $W$:

$$\rho(\boldsymbol{r}) = \sum_j q_j W(\boldsymbol{r}_j - \boldsymbol{r}), \qquad j = 1, ..., N_{\text{cell}} \tag{30}$$

where $W(\boldsymbol{r_j} - \boldsymbol{r})$ is a function describing the effective shape of the particles. Often it is sufficient to use a linear weighting for $W$ – originally dubbed the 'Cloud-in-Cell' scheme by its inventors Birdsall & Fuss[22] – although other more accurate methods are also possible. Once $\rho(\boldsymbol{r})$ is defined at the grid points, we can proceed to solve Poisson's equation to obtain the new electric field. This is then interpolated back to the particle positions so that we can go back to the particle push step (29) and complete the cycle – Fig. 8.

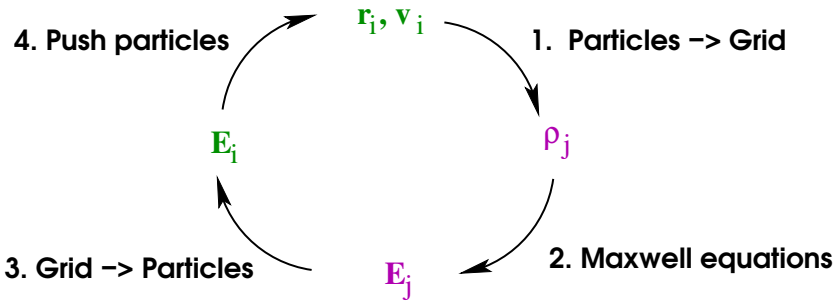



Figure 8. Schematic illustration of the particle-in-cell algorithm.

Because of its simplicity and ease of implementation, the PIC-scheme sketched above is currently one of the most widely used plasma simulation methods. It is particularly suitable for the study of *kinetic* or *non-Maxwellian* effects. The simplest variation of this technique is a '1D1V'-configuration: 1 space coordinate plus 1 velocity, the numerical behaviour of which was first examined by Dawson some forty years ago[23].

The heart of the code is based on the following difference equations:

Particle pusher:

$$v_i^{n+\frac{1}{2}} = v_i^{n-\frac{1}{2}} + \frac{q_i}{m_i}E_i^n \Delta t,$$
$$x_i^{n+1} = x_i^n + v_i^{n+\frac{1}{2}}\Delta t, \tag{31}$$

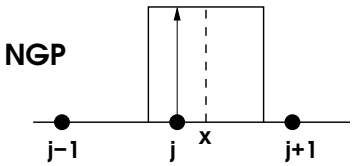
Density gather:

$$\rho_j^{n+1} = \sum_i q_i W(x_i - x_j),$$
$$S = 1 - \frac{\mid x_i - x_j \mid}{\Delta x}, \tag{32}$$

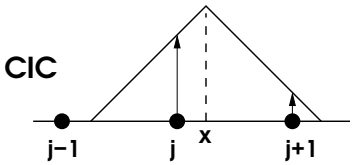
Field integration:

$$E_{j+\frac{1}{2}}^{n+1} = E_{j-\frac{1}{2}}^{n+1} + \rho_j^{n+1}\Delta x. \tag{33}$$

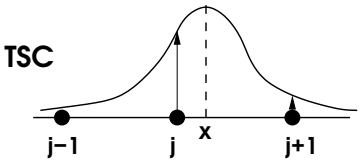
Notice that this scheme uses one of the simplest weighting schemes, namely linear interpolation, with the effect that the particles have an effective size of $2\Delta x$. Other weighting schemes are listed below in Fig. 9. The choice of scheme is, not surprisingly, a choice between speed and accuracy – see Hockney & Eastwood, Chapter 5 for a comprehensive discussion[3]. We will return to this issue when we discuss the Particle-Mesh-Ewald method in Section 3.3. The subscript on $W$ refers to the number of grid points along each axis




$$W_1(x) = 1, \qquad \mid x \mid < 1$$




$$W_2(x) = 1 - |x|, \qquad \mid x \mid < 1$$




$$W_3(x) = \begin{cases} \dfrac{3}{4} - x^2, & |x| < \frac{1}{2} \\ \dfrac{1}{2}(\dfrac{3}{2} - |x|)^2, & \frac{1}{2} < |x| < \frac{3}{2} \end{cases}$$

Figure 9. Charge assignment schemes: a) Nearest-grid-point, b) Cloud-in-cell and c) Triangular-shaped-cloud.

contributing to the charge-spreading. In all cases, $x \equiv (x_i - x_j)/\Delta x$ and $W(x) = 0$ outside the ranges indicated. The weighting function is extended into 2 and 3 dimensions by forming the product along each direction. For example, the popular CIC scheme in 3D takes the form of an 8-point weighting function:

$$W_2(\boldsymbol{r}) = (1 - |x|)(1 - |y|)(1 - |z|), \qquad |x|, |y|, |z| < 1.$$

The difference scheme for the Poisson equation is also readily generalised to 3 dimensions:

$$6\phi_{j,k,l} - (\phi_{j+1,k,l} + \phi_{j-1,k,l} + \phi_{j,k+1,l} + \phi_{j,k-1,l} + \phi_{j,k,l+1} + \phi_{j,k,l-1}) = 4\pi\Delta^2\rho_{j,k,l}, \quad (34)$$

where the grid spacing is now assumed to be the same in all directions: $\Delta_x = \Delta_y = \Delta_z = \Delta$. Given that we can solve (34), the electric field is then obtained from:

$$E_{j,k,l} = -\frac{\phi_{j+1,k,l} - \phi_{j-1,k,l}}{2\Delta} - \frac{\phi_{j,k+1,l} - \phi_{j,k-1,l}}{2\Delta} - \frac{\phi_{j,k,l+1} - \phi_{j,k,l-1}}{2\Delta} \quad (35)$$

For periodic systems, Equation 34 can be solved using fast Fourier transforms (FFTs), by first computing $\tilde{\rho}(k)$ and then applying the inverse transform to $\tilde{\phi}(k) = \tilde{\rho}(k)/k^2$ to recover $\phi(r)$ in real space. This procedure results in $N_g \log N_g$ operations, where $N_g$ is the total number of grid points.

Having obtained new field values on the mesh, these are then interpolated back to the particles using the *same* weighting scheme as for the charge assignment:

$$\boldsymbol{E}_i = \sum_{jkl} E_{j,k,l} W(r_i - r_{j,k,l}).$$

For example, in the CIC scheme in Fig. 9, a particle will receive field contributions from its nearest 8 grid points, weighted according to the volume-overlap between the mesh cell and a cube of side $\Delta/2$ centered on the particle.

Particle-mesh or particle-in-cell simulation typically uses many particles per cell, $N \gg N_g$, to keep field quantities as smooth as possible. The main computational cost is therefore not in the field solver, but in the integrator (or 'particle pusher', as it is commonly known), which is just $O(N)$. In electrodynamics, the pusher can be much more complicated than the simple linear acceleration implied by (31), often containing magnetic fields and relativistic factors. The drawback of mesh methods is that the spatial resolution is limited to distances $r \geq \Delta$, no matter how densely-packed the particles are, since these have an effective size $\sim \Delta$. In some sense, the particle-mesh technique is the ideal algorithm for long-range forces, because short-range interactions are automatically excluded! Of course this is not particularly helpful in the context of molecular dynamics, which is based on the ability to follow individual particle trajectories – including short-range encounters – explicitly and accurately.

### 3.2 P$^3$M: Particle-Particle, Particle-Mesh

Ideally, one would like to have the best of both worlds offered by pure MD and PM respectively: high resolution of individual encounters, combined with a rapid mesh-based evaluation of long-range forces. This is precisely the philosophy behind the particle-particle, particle-mesh (P$^3$M) method developed primarily by Eastwood in the 1970s[24]. The interparticle force is initially split into two contributions:

$$\boldsymbol{F}_{ij} = \boldsymbol{F}_{ij}^{PP} + \boldsymbol{F}_{ij}^{PM}, \tag{36}$$

where the PP part is finite only over a few interparticle spacings, up to some cutoff radius $r_c$; the PM part is assumed to be temporally and spatially smooth enough to be computed on a mesh – Fig. 10.

The question which immediately arises is: how do we implement this splitting in practice? Clearly, the short-range (PP) sphere should be as small as possible to minimise the number of direct PP interactions. If it is too small, however, spatial resolution will be lost which cannot be compensated for by the PM calculation. This is because PM-codes filter out all modes with $|\boldsymbol{k}| \geq \pi/\Delta$, where $\Delta$ is the mesh size.

The second issue concerns the matching of the force and potential contributions across the artificial boundary created by the cutoff sphere. We have already seen that the introduction of a grid causes the particles to acquire a finite form factor which depends on the details of the charge assignment scheme. We can illustrate how force splitting works by

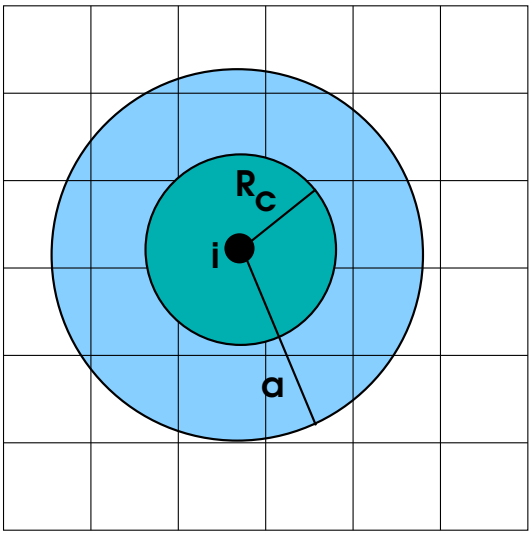



Figure 10. Force calculation using the $P^3M$ splitting scheme.

taking the simplest of the schemes in Fig. 9, NGP, and working out its associated interparticle force. In three dimensions, the NGP scheme is equivalent to replacing point charges by spheres of radius $a/2$, with a uniform density given by:

$$\rho(r) = \begin{cases} \dfrac{6q}{\pi a^3}, & r < a/2 \\ 0, & r \geq a/2 \end{cases} \tag{37}$$

Elementary electrostatics shows that the force between two such spheres along the axis joining their centres is given by:

$$F_{\text{sphere}}(r) = \begin{cases} \dfrac{q^2}{a^2}\left(\dfrac{8r}{a} - \dfrac{9r^2}{a^2} + \dfrac{2r^4}{a^4}\right), & r < a \\ \dfrac{q^2}{r^2}, & r \geq a \end{cases} \tag{38}$$

The natural force-splitting choice in this case is to take the short-range cutoff $R_c = a$, and to set the PP force inside this sphere equal to the *difference* between the Coulomb force and this effective force contribution arising from the mesh points:

$$F^{PP}(r) = \begin{cases} F_c(r) - F_{\text{sphere}}(r), & r < a \\ 0, & r \geq a \end{cases}$$

This modified force-law is illustrated in Fig. 11. Note that we are not changing the physics here: the ultimate goal is still to match the exact Coulomb law for all $r$; that is, when short- and long-range components are added together. Of course, we could guarantee

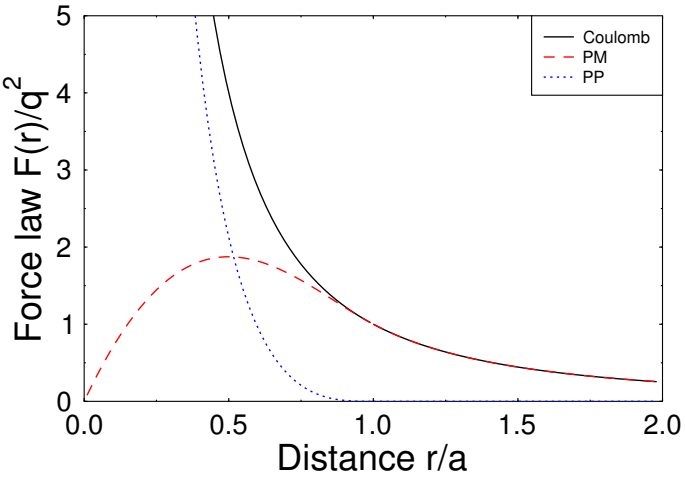



Figure 11. $P^3M$ force-laws for short-range (PP, dotted line) and long-range (PM, dashed line) contributions.

this by summing over the *particles*:

$$F_i^{PM} = \sum_{j=1}^{N} F_{\text{sphere}}(r_i - r_j),$$

but then we would be back to square one, saddled with an $O(N^2)$ force-summation! The challenge is to arrange the fast, but approximate, particle-mesh calculation in such a way as to minimize the difference between the forces evaluated on the mesh and the 'exact' reference force given by (38).

Fortunately the particle-mesh algorithm outlined earlier (31)–(33) gives us plenty of leeway to achieve this goal. All four of the intermediate steps necessary to compute the 'mesh' forces – charge assignment, potential solver, differencing, and back-interpolation to the particles – present an opportunity to improve the matching of $F_i^{PM}$ to the reference force. The details of this minimisation process are quite complex however, and the reader is referred to Hockney & Eastwood[3], Chapter 8. In the public version of their $P^3M$ code[24], charge assignment is performed using the TSC function, rather than NGP, which is found to give better overall performance. Once the PM parameters have been fixed, the calculation proceeds as in conventional MD: linked neighbour-lists are used to narrow down the search effort in the short-range calculation[25,6], and the velocities and positions are updated using a standard integrator.

The P$^3$M algorithm in its original form as advocated by Hockney & Eastwood has not been widely adopted for production MD simulation, despite its promising scaling characteristics for large particle number. This is perhaps due to the uncertainty created by the admittedly complicated force-splitting procedure, leaving the user with less than 100% confidence in its accuracy. Recent comparisons of P$^3$M with other techniques[26] has

demonstrated that these doubts are largely unfounded, however, so we can perhaps expect a rejuvenation of this method in other fields too.

A more transparent version of $P^3M$ has been implemented for 'dense plasma' (neutral electron-ion) systems by Nishihara *et al.*[27]. In their scheme, the short- and long-range contributions are organised by the mesh itself – Fig.12. In this example, the Coulomb forces on particle $i$ are summed 1 over the other particles within its own cell plus those (e.g.: $j_1$) in the neighbouring 26 cells (hatched region). The forces from particles outside ($j_2$) are computed from the mesh (shaded region), but *excluding* the charges inside the hatched region. The splitting is thus even more artificial than before, but actually easier to implement.

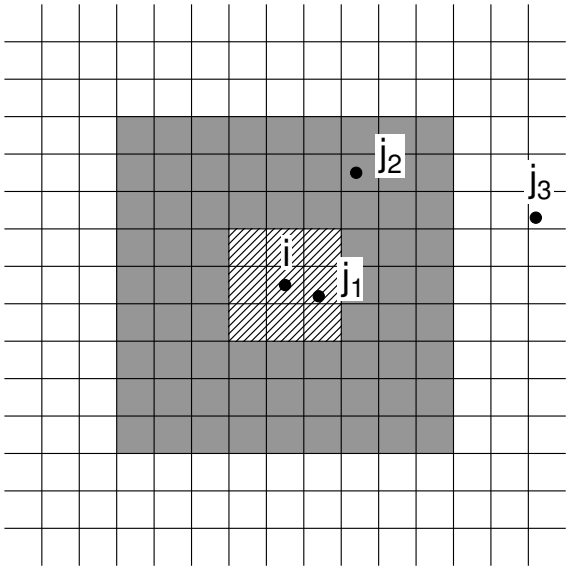



Figure 12. $P^3M$ scheme using grid-based force-splitting.

### 3.3 Particle-Mesh-Ewald

By now the reader will probably have noticed the striking similarity between the classical Ewald method – summarised by Eq. 6 – and the force- or potential-splitting principle behind $P^3M$. This correspondence is perhaps surprising because in a sense, the two methods approach the $N$-body problem from different philosophical viewpoints: the Ewald method seeks to perform an exact, periodic lattice-sum by deliberately introducing a charge-spreading function to speed up convergence; $P^3M$ is a means of rapidly evaluating the long-range force contribution through the use of a grid, due to which the charges automatically acquire a finite size.

In Section 2.2 we saw that the optimal configuration for the Ewald sum is achieved when the computational effort is equally shared between the real- and reciprocal-space

sums. In P$^3$M, the bias is shifted even further towards the particle-mesh calculation (which is also usually performed in reciprocal space) to guarantee something like an $O(N)$ scaling. In 1993, Darden, York and Pedersen[4] realised that the Ewald sum could be recast in the P$^3$M form by using a large value for the convergence parameter $\alpha$. This accelerates the convergence of the real-space sum, which can then be restricted to a smaller cutoff radius $R_c$, and ensures that the main contribution is computed in $k$-space. According to Eq. 24, this means summing over a large number of $k$-vectors, $N_k \sim N/nR_c^3$ for *each particle*.

At this point, Darden *et al.* took the analogy with P$^3$M a step further, arguing that Gaussian-shaped charges could be equally well mapped onto a regular grid and the resulting potential computed by a FFT. Assuming the charges can be adequately represented on a $m \times m \times m$ mesh with a fixed number of particles per cell, i.e.: $m^3 \propto N$, then the scheme will scale as $N \log N$, the time it takes to perform the Fourier transform. Thus, instead of evaluating (17), we first compute the FFT of the gridded density, which by inspection of (15) is given by:

$$\tilde{\rho}_{jkl}(k) = \frac{1}{L^3} \exp\left(-\frac{k^2}{4\alpha^2}\right) \sum_i q_i \exp(i\boldsymbol{k} \cdot \boldsymbol{r}_i) \tag{39}$$

The potential in $k$-space is then just the density multiplied by an 'influence' function $\tilde{G}(k)$:

$$\tilde{\phi}_{jkl}(k) = \tilde{G}(k)\tilde{\rho}(k), \tag{40}$$

where

$$\tilde{G}(k,\alpha) = \frac{\exp(-k^2/4\alpha^2)}{k^2}, \tag{41}$$

and the fields are given simply by:

$$\begin{aligned} \tilde{E}_{jkl}(\boldsymbol{k}) &= -i\boldsymbol{k}\tilde{\phi}_{jkl}(k) \\ &= -i\boldsymbol{k}\tilde{G}(k,\alpha)\tilde{\rho}(k). \end{aligned} \tag{42}$$

The inverse FFT then yields the potential and fields at the grid points, which can then be interpolated back to the particle positions[28].

In its original form, the particle-mesh-Ewald (PME) algorithm of Darden *et al.* stuck with Gaussian charge shapes for consistency with the real-space part of the Ewald sum (which is evaluated conventionally according to Eq. 9). Unfortunately, this has the drawback that a large number of points – typically $8^3 \sim 100$ – are needed to map the charges onto the grid, creating a new numerical bottleneck. The advantage of the P$^3$M scheme is that it restricts the charge distribution to 8 (CIC) or 27 (TSC) grid points respectively. A number of works have since appeared[29,26] which fully adopt the P$^3$M concept of a narrow, finite assignment function. Just as in P$^3$M, the errors incurred by this procedure can be compensated by modifying the influence function $\tilde{G}(k)$ – (41) – to include the discreteness effects of the grid[30]. For consistency, the same charge shape should be used for the real-space sum, which, as shown by Heyes[10], will still guarantee rapid convergence. PME can thus be formally regarded as a special case of P$^3$M, though we have deliberately treated the methods separately here because of their independent historical development.

Many implementations and applications of the PME method have already appeared[31], including a sophisticated variation based on B-spline interpolation[32], assessments of its performance relative to crude cutoff methods[33] as well as multipole methods[28].

## 4 Multipole Methods

So far in this review, we have dealt exclusively with periodic systems, which can be neatly handled by some form of Ewald summation. As we have seen in the previous two sections, there are essentially two choices in this case: the direct, optimised scheme of Perram *et al.*[13], or a grid-based P$^3$M/PME scheme[4,29]. There is, of course, a large number of $N$-body problems for which periodic boundaries are completely *in*appropriate, for example: galaxy dynamics, electron-beam transport, large proteins[31], and any number of problems with complex geometries. So how does one get round the $N^2$-bottleneck if there is no symmetry to exploit?

Two new approaches to this problem were put forward in the mid-1980s, the first from Appel[34] and Barnes & Hut[35], who proposed $O(N \log N)$-schemes based on hierarchical grouping of distant particles; the second from Greengard & Rohklin[36], who went one better, devising an $O(N)$ solution with rounding-error accuracy. These two methods – known today as the 'hierarchical tree algorithm' and the 'fast multipole method' respectively – have revolutionised $N$-body simulation in a much broader sense than the specialised periodic methods discussed earlier. They offer a generic means of accelerating the computation of many-particle systems governed by central, long-range potentials.

Although the FMM is currently more widespread in molecular dynamics than the Barnes-Hut (BH) tree algorithm, we nevertheless give both methods an equal airing here. For one thing, the methods are conceptually very similar (and are therefore related); secondly, both FMM and BH are still evolving, and it is likely that some hybrid, adaptive scheme may eventually prevail as a competitive alternative to, say, PME even for periodic simulations with moderate numbers of particles.

### 4.1 The Barnes-Hut Tree Algorithm

An inherent inefficiency in direct force-summation is that one does not distinguish near-neighbours from more distant particles; each pair evaluation requires the same computational effort, even though the individual contributions of distant particles may be negligibly small. Introduction of an artificial cutoff radius can separate out the important and less important partners, but this procedure only works well for short-range potentials; for Coulomb potentials, errors will accumulate as a result of abrupt truncation.

In 1986, Barnes and Hut[35] introduced a scheme in which the physical space is systematically divided up so as to establish and maintain a relationship between each particle and its neighbours. The resulting 'tree' structure can then be used to group distant clusters of particles into a single charge or mass, thereby reducing the number of interactions in the force/potential calculation. These codes are sometimes described as oct-tree codes to distinguish them from so-called binary tree codes[37,38], based on nearest neighbour *pairs*. Although binary trees might reflect the structure of the system more closely, the Barnes and Hut method is by far the most commonly used method due to its conceptual simplicity and easy, low-overhead tree construction.

There is no single correct way to go about the tree-building process, but one method which produces an identical structure to the original BH scheme is as follows[39]. First, a root cell is created containing all simulation particles – Fig.13a) This cell is then divided into eight equally sized subcells – Fig.13b). For each subcell, one asks whether it contains none, one, or more than one particle.

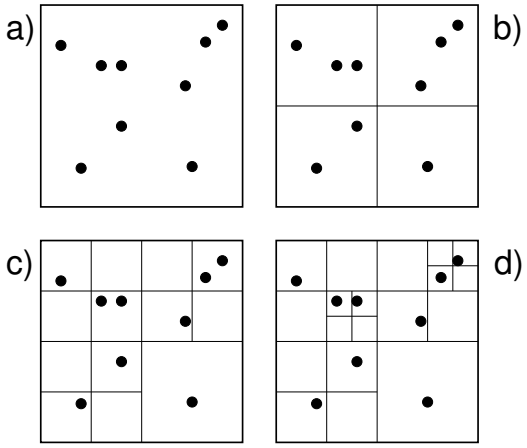



Figure 13. Step-by-step division of space for a simple 2-D particle distribution.

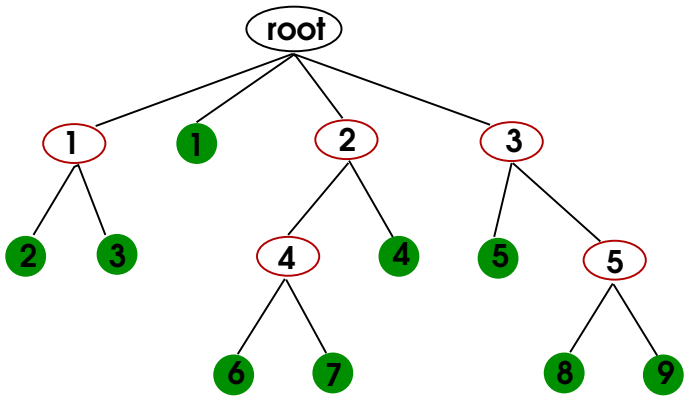



Figure 14. Tree data structure corresponding to Fig. 13d)

If the cell is empty, this cell is ignored; if there is one particle in the cell, this is stored as a 'leaf' node in the tree structure; if there are more particles in a cell, this cell is stored as a 'twig' node and subdivided further. The subdivision process continues until there are no cells with more than one particle left, which ultimately leads to Fig.13d). The division of space just described is not used as a grid in the particle-mesh sense, but rather as a *bookkeeping* structure. At each division step, the tree data structure is augmented with the twig-nodes belonging to next level down in the hierarchy – Fig. 14. Each node in the tree is associated with a cubic volume of space containing a given number of particles; empty cells are not stored. Pointers to the *parents* of each leaf and twig node are also kept in the tree structure.

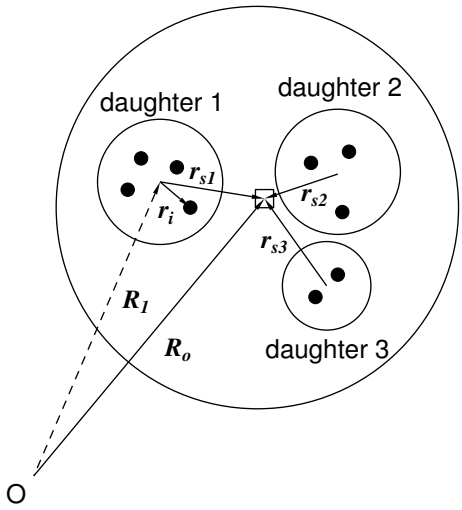



Figure 15. Origin shift for the multipole calculation: the circles symbolize the pseudoparticles (twig-nodes); $\mathbf{r}_i$ is a vector from a constituent particle to the centre of charge of the daughter node and $\mathbf{r}_{s1}, \mathbf{r}_{s2}$, etc. are the shifting vectors to the new origin $O'$ (□), which is the centre of charge of the parent node.

Once the tree is in place, the twig nodes (represented by the 5 ellipses in Fig. 14) can be 'loaded' with information about their physical contents, like their centres of charge,

$$\boldsymbol{r}_{coc} = \frac{\sum_i \mid q_i \mid \mathbf{r}_i}{\sum_i \mid q_i \mid} \tag{43}$$

and their multipole moments:

$$M = \sum_i q_i$$

$$D_\alpha = \sum_i q_i r_{i\alpha} \tag{44}$$

$$Q_{\alpha\beta} = \sum_i q_i (3 r_{i\alpha} r_{i\beta} - r_i^2 \delta_{\alpha\beta}) \tag{45}$$

This information will be needed later for the force calculation when the twigs are treated as pseudoparticles. This loading of twig-nodes, can be performed very rapidly by propagating information up the tree level-by-level, from individual particles (leaves), through the intermediate twigs until the root is reached.

To do this, we make use of the already-computed multipole moments of the daughter cell to calculate the moments of the parent cell. Each individual $\mathbf{r}_i$ in the sum of a daughter moment is shifted by the same vector $\mathbf{r}_{sd}$. For example,

$$\sum_i q_i x_i \quad \rightarrow \quad \sum_i q_i x_i - x_{sd} \sum_i q_i$$

$$\sum_i q_i x_i^2 \quad \rightarrow \quad \sum_i q_i x_i^2 - 2 x_{sd} \sum_i q_i x_i + x_{sd}^2 \sum_i q_i$$

$$\sum_i q_i x_i y_i \qquad \rightarrow \qquad \sum_i q_i x_i y_i - x_{sd} \sum_i q_i y_i - y_{sd} \sum_i q_i x_i + x_{sd} y_{sd} \sum_i q_i$$

and so forth. These results are used later to calculate the contribution of selected twig-nodes to the total force or potential.

Obviously, the tree-building process incurs a certain overhead in an $N$-body code, so it is natural to ask how much. A rough estimate of how many divisions are needed to reach a typical cell, starting from the root, can be obtained from the average size of a cell containing one or more particles. The average volume of such a cell is the volume of the root cell $V$ divided by the number of simulation particles $N$. Moreover, the average length of a cell is a power of $V^{1/3}/2$. Therefore,

$$\left(\frac{1}{N}\right)^{1/3} = \left(\frac{1}{2}\right)^{x},$$

which means that the height $x$ of the tree is of the order

$$\log_2 N^{1/3} = \frac{1}{3 \log 2} \log N \simeq \log N. \tag{47}$$

Starting from the root, an average of $\log N$ divisions are necessary to reach a given leaf. The tree contains $N$ leaves, therefore the time required to construct the tree is $O(N \log N)$. In practice, tree-building actually comprises only 3–5% of the total force calculation, so it can be performed every timestep.

The tree structure provides the means to distinguish between close particles and distant particles without actually calculating the distance between every particle. The force between near-neighbours is calculated directly, whereas more distant particles are grouped together to pseudoparticles. An *interaction list* is thus built for each particle by traversing the tree from node-to-node and deciding whether to accept the node as-is, or subdivide further. There are actually a number of so-called 'multipole acceptance criteria' (MACs) of varying complexity[40], the simplest of which is the original '$s/d$' criterion introduced by Barnes and Hut[35]. Beginning at the root of the tree, the 'size' of the current node (or twig), $s$, is compared with its distance from the particle, $d$. If the ratio $s/d$ is smaller than some preset value, $\theta$, then the internal structure of the pseudoparticle is ignored and it is added to the interaction list for that particle. Otherwise, this node is resolved into its daughter nodes, each of which is recursively examined according to $s/d$ and, if necessary, subdivided. Fig. 16 illustrates this comparison at two different stages of the tree-walk. This continues until all nodes have been examined, i.e.: when we have returned to the root.

The result of this procedure is an interaction list, the length of which depends both on $N$ and $\theta$, the multipole acceptance parameter – Fig. 17. The case $\theta$ = 0 is equivalent to computing all particle–particle interactions; which is exact but rather pointless, because the operation count is again $O(N^2)$. This is in fact slower than direct PP because of the tree-building and traversal overheads. A practical choice using this MAC proves to be in the range $\theta = 0.3$–1.0, depending on the application.

The asymptotic scaling of this algorithm can be estimated by considering the average number of interactions $n_{int}$ in a spherical, homogeneous distribution surrounding a test particle. For nonzero $\theta$, it can be shown that[39]:

$$n_{int} \sim \log N / \theta^2, \tag{48}$$

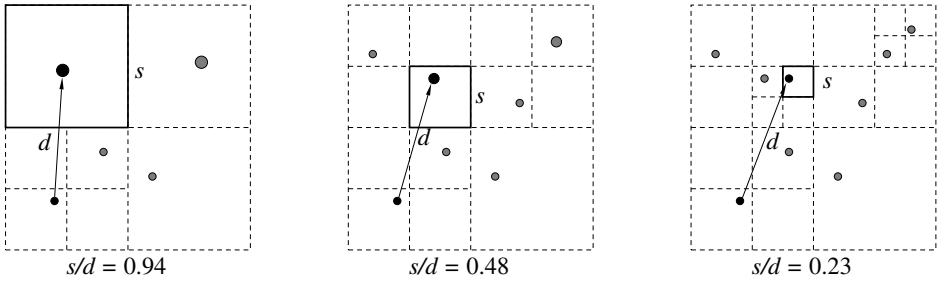



Figure 16. The relation $s/d$ for different levels of the tree.

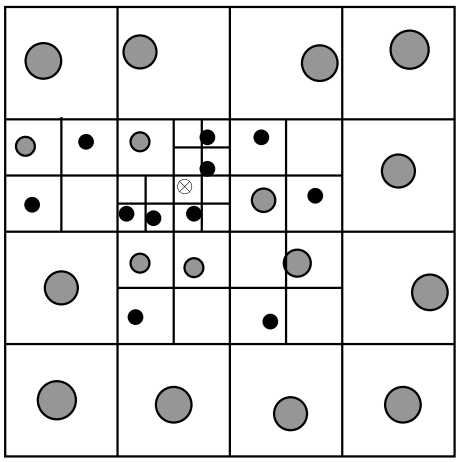


Figure 17. Interaction list generated for particle marked $\otimes$, using multipole acceptance parameter $\theta = 1.0$. The black circles are single charges; the shaded circles multipole expansions.

so the time required to calculate the force on a given particle is $O(\log N)$, which means the number of operations to compute the force on all $N$ bodies will scale as $O(N \log N)$.

Having determined the interaction lists for each particle, all that remains is to compute the potentials and forces. However, we have already anticipated making use of multipole expansions to take account of the charge *distribution* inside the pseudoparticle terms. Referring to Fig. 18, the potential at particle $P$ due to the pseudoparticle is the sum of the potentials $\Phi_i$ due to the particles contained in the cell,

$$\Phi(\mathbf{R}) = \sum_i \Phi_i(\mathbf{R} - \mathbf{r}_i),$$

where $\mathbf{r}_i$ is the vector from the particle to the centre of mass and the origin is, for simplicity, the individual particle on which the force of the pseudoparticle is calculated. Here we

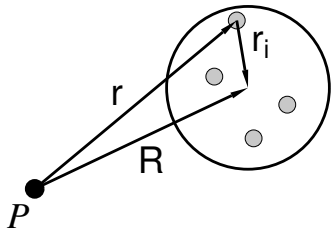



Figure 18. Geometry of multipole expansion.

consider a $1/r$-potential, therefore

$$\begin{aligned}\Phi_i(\mathbf{R}-\mathbf{r}_i) &= \frac{q_i}{|\mathbf{R}-\mathbf{r}_i|} \\ &= \frac{q_i x_i}{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}}.\end{aligned}$$

Expanding this potential about $\boldsymbol{R}$ up to quadrupole order, gives:

$$\begin{aligned}\Phi(\mathbf{R}) &= \sum_i q_i \left[1 - \mathbf{r}_i\frac{\partial}{\partial \mathbf{r}} + \frac{1}{2}\mathbf{r}\,\mathbf{r}\frac{\partial}{\partial \mathbf{r}}\frac{\partial}{\partial \mathbf{r}} + ...\right]\frac{1}{R} \\ &= \sum_i q_i \left[1 - x_i\frac{\partial}{\partial x} - y_i\frac{\partial}{\partial y} - z_i\frac{\partial}{\partial z}\right. \\ &\quad +\frac{1}{2}x_i^2\frac{\partial}{\partial x}\frac{\partial}{\partial x} + \frac{1}{2}y_i^2\frac{\partial}{\partial y}\frac{\partial}{\partial y} + \frac{1}{2}z_i^2\frac{\partial}{\partial z}\frac{\partial}{\partial z} \\ &\quad +\frac{1}{2}x_i y_i\left(\frac{\partial}{\partial x}\frac{\partial}{\partial y} + \frac{\partial}{\partial y}\frac{\partial}{\partial x}\right) \\ &\quad +\frac{1}{2}y_i z_i\left(\frac{\partial}{\partial y}\frac{\partial}{\partial z} + \frac{\partial}{\partial z}\frac{\partial}{\partial y}\right) \\ &\quad \left.+\frac{1}{2}x_i z_i\left(\frac{\partial}{\partial x}\frac{\partial}{\partial z} + \frac{\partial}{\partial z}\frac{\partial}{\partial x}\right)\right]\frac{1}{R} \\ &= \sum_i q_i \left[\frac{1}{R} + x_i\frac{x}{R^3} + y_i\frac{y}{R^3} + z_i\frac{z}{R^3}\right. \\ &\quad +\frac{1}{2}x_i^2\left(-\frac{1}{R^3} + \frac{3x^2}{R^5}\right) + \frac{1}{2}y_i^2\left(-\frac{1}{R^3} + \frac{3y^2}{R^5}\right) \\ &\quad +\frac{1}{2}z_i^2\left(-\frac{1}{R^3} + \frac{3z^2}{R^5}\right) \\ &\quad \left.+x_i y_i\left(\frac{3xy}{R^5}\right) + y_i z_i\left(\frac{3yz}{R^5}\right) + x_i z_i\left(\frac{3xz}{R^5}\right)\right]\end{aligned}$$

This can be rearranged to give:

$$\Phi(\boldsymbol{R}) = \frac{M}{R} + \sum_{\alpha} \frac{r_\alpha D_\alpha}{R^3} + \sum_{\alpha\beta} \frac{1}{2} Q_{\alpha\beta} \frac{r_\alpha r_\beta}{R^5}, \tag{50}$$

where $M, D_\alpha$ and $Q_{\alpha\beta}$ are the monopole, dipole and quadrupole moments of the pseudoparticles, defined previously in (44). The indices $\alpha$ and $\beta$ refer to the 1 components $x, y, z$, so that $\boldsymbol{r}_{i\alpha} = (x_i, y_i, z_i)$ etc.

The corresponding electric field can be obtained directly from (50) by differentiating with respect to $\boldsymbol{R}$:

$$\mathbf{E}(\mathbf{R}) = -\frac{\partial}{\partial \mathbf{R}} \Phi(\boldsymbol{R}), \tag{51}$$

which gives for the each component $\gamma$:

$$E_\gamma = \frac{r_\gamma}{R^3} M + \sum_{\alpha} \left( \frac{3 r_\alpha r_\gamma}{R^5} - \frac{r_\alpha \delta_{\alpha\gamma}}{R^3} \right) D_\alpha + \sum_{\alpha\beta} \frac{5 r_\alpha r_\beta r_\gamma}{R^7} \cdot \frac{1}{2} Q_{\alpha\beta} - \sum_{\alpha} \frac{r_\alpha}{R^5} Q_{\alpha\gamma}.$$

Expanding the multipole moments and the summations, we arrive at a somewhat more convenient form for implementing in a code[5]:

$$\begin{aligned}
E_x = & \frac{x}{R^3} \sum_i q_i. \\
& - \left( \frac{1}{R^3} - \frac{3x^2}{R^5} \right) \cdot \sum_i q_i x_i + \frac{3xy}{R^5} \cdot \sum_i q_i y_i + \frac{3xz}{R^5} \cdot \sum_i q_i z_i. \\
& + \left( \frac{15x^3}{R^7} - \frac{9x}{R^5} \right) \cdot \frac{1}{2} \sum_i q_i x_i^2 + \left( \frac{15xy^2}{R^7} - \frac{3x}{R^5} \right) \cdot \frac{1}{2} \sum_i q_i y_i^2 \\
& + \left( \frac{15xz^2}{R^7} - \frac{3x}{R^5} \right) \cdot \frac{1}{2} \sum_i q_i z_i^2 + \left( \frac{15x^2 y}{R^7} - \frac{3y}{R^5} \right) \cdot \sum_i q_i x_i y_i \\
& + \left( \frac{15x^2 z}{R^7} - \frac{3z}{R^5} \right) \cdot \sum_i q_i x_i z_i + \left( \frac{15xyz}{R^7} \right) \cdot \sum_i q_i y_i z_i.
\end{aligned} \tag{53}$$

The $y$- and $z$-components can be found by cyclic rotation. Compared with the direct force calculation, Equation 53 above contains 8 additional multipole terms which must be evaluated for each twig-node in the interaction list. Clearly, this necessitates a certain overhead for the BH tree algorithm, so that it will only be more efficient above a certain particle number. Where this breakeven point is exactly, depends mainly on the accuracy desired, i.e. on the choice of $\theta$. We defer comparison of multipole schemes until later (Fig.24), but for the moment, we note that the standard tree code relies very much on a trade-off between speed and accuracy – Fig. 19.

### 4.2 The Fast-Multipole Method (FMM)

The Fast-Multipole Method was developed by Greengard & Rohklin, shortly after the Barnes–Hut algorithm appeared[36,41,42]. In some sense, therefore, the FMM can be thought

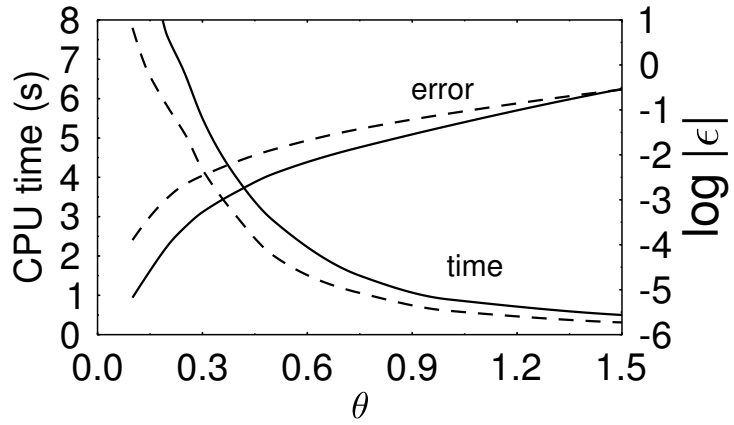



Figure 19. Trade-off between CPU time per integration step and average force error as a function of multipole acceptance parameter using monopole terms only (dashed lines) and quadrupole terms (solid lines).

of as an elegant refinement of the BH tree-code, but in fact it was developed independently. The FMM makes rigorous use of the fact that a multipole expansion to infinite order contains the total information of a particle distribution. As in the BH algorithm, the interaction between near-neighbours is calculated by direct particle–particle force summation, and more distant particles are treated separately. However, the distinction between these two contributions is obtained in a different way. In FMM the distant region is treated as a *single* 'far-field' contribution, which is calculated by a high-order multipole expansion.

By forming high-order multipole moments at the lowest level of the tree and carefully combining and shifting these centres up and down the tree, Greengard and Rokhlin showed that the $N$-body problem can in principle be reduced to an order $O(N)$ algorithm[36]. Arbitrary accuracy (e.g., within numerical rounding error) can be assured *a priori* by taking sufficient terms in the expansion. Because of this, it is essential to find a concise mathematical representation of the multipoles and their shifting theorems. The first FMM codes were two dimensional[36,43] and exploited a convenient complex variable notation to represent the potentials and fields. The 3D formulation uses spherical harmonics instead and was also derived by Greengard[41], and later implemented by Schmidt and Lee[44] for periodic systems.

Like the tree algorithm, FMM starts with a box big enough to contain all the simulation particles, and this box is subsequently subdivided into boxes of length $d/2^r$ ($r = 0, 1, 2, ...$) equivalent to $8^r$ equal sized subvolumes ($4^r$ in two dimensions). In contrast to the tree method, however, this is done for *every box* up to a given maximum refinement level $R$, regardless of the number of particles it contains – Fig. 20. This maximum refinement level $R$ is chosen so that the number of boxes is approximately equal to the number of the simulated particles $N$. This means that assuming the $N$ particles are more or less homogeneously distributed, the maximum refinement level (in 3D) must be chosen as

$$R = \log_8 N.$$

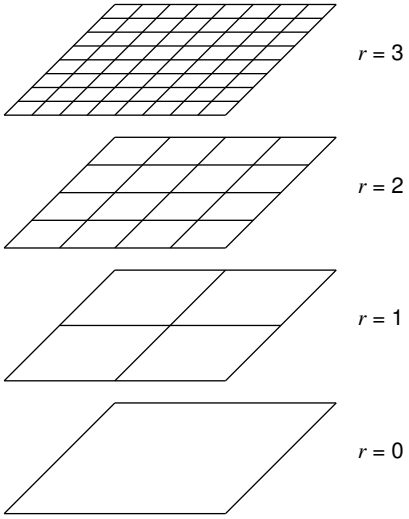



Figure 20. Division of the simulation box in a fast multipole code.

The FMM tree structure is used to build 'near-neighbour' lists of boxes at each refinement level $r$. Near-neighbours are defined as the box itself and any box at the same level with which it shares a boundary point. By contrast, a box on the same level which is *not* in a near-neighbour list is *well separated*: A local multipole expansion made about the centre of this box will then automatically be valid. Using the usual conventions[45] for spherical harmonics and units, the generalized version of (50) is:

$$\Phi(\mathbf{r}) = 4\pi \sum_{l,m} \frac{M_{lm} Y_{lm}(\theta, \phi)}{(2l+1) r^{l+1}} \tag{54}$$

in spherical coordinates $(r, \theta, \phi)$ relative to an origin $O$, with multipole moments given by:

$$M_{lm} = \sum_i q_i r_i^l Y_{lm}^*(\theta_i, \phi_i), \tag{55}$$

where the charges now have the coordinates $(r_i, \theta_i, \phi_i)$. As in the Barnes-Hut algorithm, multipole moments are first computed for each box at the highest refinement level $R$, but this time relative to the *centre of the box* rather than the centre of charge of the particles. The maximum number of terms $L$ in the multipole expansion is chosen such that[36,43]:

$$(\sum_i \mid q_i \mid) 2^{-L} \leq \epsilon, \tag{56}$$

where $\epsilon$ is the desired precision.

Next, the multipole moments on the next coarsest refinement level $r = R - 1$ are calculated, and just as for the tree method, the shifted multipole moments of the daughter cells can be used to obtain the moments of the parent cell. Shifting the origin $O$ to $O'$ by

a translation vector $\mathbf{r}_t$, a transformation of the multipole moments is needed to obtain the expansion in terms of the new vector $\mathbf{r}' = \mathbf{r} - \mathbf{r_t}$ relative to $O'$. This transformation is given by:

$$M'_{l'm'} = \sum_{l,m} T^{MM}_{l'm',lm} M_{lm}, \tag{57}$$

with the transformation matrix

$$T^{MM}_{l'm',lm} = 4\pi \frac{(-r_t)^{l'-1} Y^*_{l'-l,m'-m}(\theta_t, \phi_t) a'_{l'-l,m'-m} a_{lm} (2l'+1)}{2(l+1)[2(l'-l)+1] a_{l'm'}},$$

and $a_{lm}$ is defined as

$$a_{lm} = (-1)^{l+m} \frac{2(l+1)^{1/2}}{[4n(l+m)!(l-m)!]^{1/2}}.$$

This is just a generalization of the 1 shifting relations used for the (quadrupole order) tree

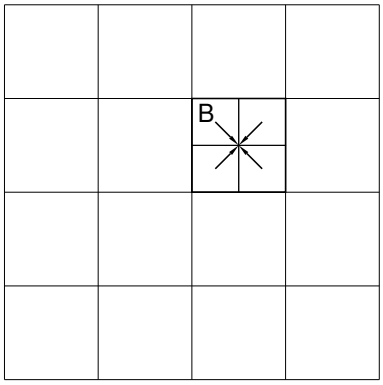



Figure 21. Shifting of the multipole expansion from the daughters on level $r = 3$ to the parents up $r = 2$ during the upward pass.

code, with the main difference that the moments are calculated relative to the centre of the cell (see Fig. 21). In this fashion, the moment expansions are carried up to to root level $r = 0$, which ultimately contains an $L$-term multipole expansion of the whole system.

So far, apart some hair-raising mathematics, there is little to choose between a standard tree code and the FMM. At this point, we could just use our $L$-term multipole expansions for the 'box-to-particle' strategy of the tree algorithm. In the FMM, however, we include the extra step of evaluating 'box-box' interactions. To do this, we perform a downward pass (from $r = 0$ to $r = R$), in which a careful distinction is made in the interaction lists for the boxes. There are actually *three* regions: the near field, the interactive field and the far field. The near field consists of the neighbouring cells; the far field is the entire simulation box *excluding* the cell in question and its neighbours. The interactive field is the part of the far field that is contained in the near field of this cell's *parents* – Fig. 22.

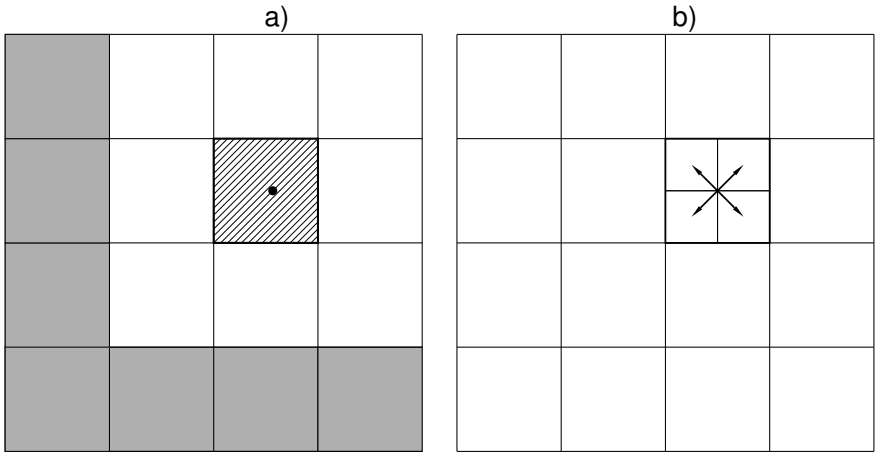



Figure 22. a) List of well-separated boxes (interactive field) which contribute to local expansion of the hatched box in b) at level $r = 2$.

In this downward pass, each multipole expansion is converted into a *local expansion* (i.e., a Taylor expansion about the centre of all well-separated boxes at each level). Using the notation above, this can be expressed thus:

$$\Psi(\mathbf{r}) = 4\pi \sum_{l,m} L_{lm} r^l Y_{lm}(\theta, \phi), \tag{60}$$

where $L_{lm}$ are referred to as the local moments of the Taylor series expansion, obtained from the original multipole moments by the transformation:

$$L_{l'm'} = \sum_{l,m} T^{LM}_{l'm',lm} M_{lm}, \tag{61}$$

where the transformation matrix $T^{LM}_{l'm',lm}$ is now

$$T^{LM}_{l'm',lm} = 4\pi \frac{(-1)^{l+m} Y^*_{l'+l,m'-m}(\theta_t, \phi_t) a_{l,m} a_{l'm'}}{r_t^{l'+l+1}(2l+1)(2l'+l) a_{l'+l,m'-m}}. \tag{62}$$

The region of validity of the multipole expansions that contribute to the local expansion consists of all boxes at the same level which are not near neighbours. At levels $r = 0$ and $r = 1$, there are no boxes which fulfill this requirement, so we just set

$$\Psi_0 = \Psi_1 = 0$$

From $r = 2$ to $r = R$ the following operations can now be performed: For each box on level $r$, the local expansion of the parent box is shifted to the centre of each of its daughters as in Fig. 22b). In other words,

$$L'_{l'm'} = \sum_{l,m} T^{LL}_{l'm',lm} L_{lm},$$

with

$$T^{LL}_{l'm',lm} = 4\pi \frac{r_t^{l-l'} Y_{l'-l,m'-m}(\theta_t, \phi_t) a_{l'm'} a_{l-l',m-m'}}{(2l'+1)[2(l-l')+1] a_{lm}}.$$

In this local expansion of the daughter cell there are now boxes missing. These are the boxes that do not touch the current daughter cell, but do not contribute to the local expansion of the parent cell either – in other words, the boxes of the *interactive* field – Fig. 23a). Their contribution has to be added to the local expansion of the daughter cell. The result is the local expansion due to all particles in all boxes at the same level which are not near neighbours. However, the ultimate aim is to evaluate the potential or force not on the box, but rather on the particles inside the box. Therefore, once the highest refinement level is reached, the local expansions at the individual particle locations are evaluated.

Finally, the remaining interactions with the particles in neighbouring boxes and the box itself are added by direct summation of the particle–particle interactions – Fig. 23b).

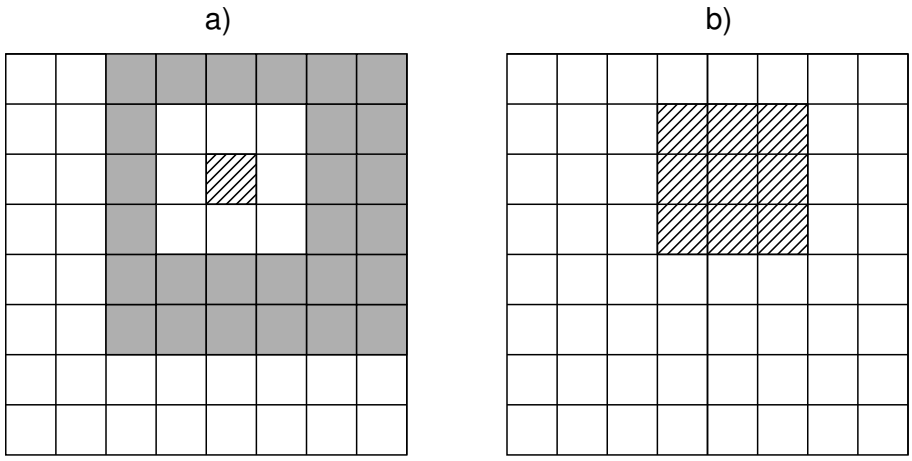



Figure 23. a) List of well-separated boxes (interactive field – grey) which contribute to the local expansion of the hatched box at level $r = 3$. b) Near-neighbour boxes (hatched region) for the direct particle–particle sum at the finest level (in this case $R = 3$)

Some recent improvements to these formulations – in which the central multipole transformations are optimised – have been proposed by Petersen et al.[46] and White and Head-Gordon[47]. A new implementation for high-precision quantum chemistry applications has been reported by Dachsel[48] at this Winter School.

At first sight it seems that FMM codes with a computation time proportional to $N$ must be superior to tree codes – which scale as $N \log N$ – and particle–particle codes – which scale either as $N^2$ or $N^{3/2}$ depending on the boundary conditions. However, FMM codes have a large overhead due to the multipole and Taylor expansions, so one has to ask instead: at what point is it appropriate to use an FMM code instead of a PP or tree code? This is not a simple question to answer, because the computation time depends not only on the number of particles $N$, but also on how accurately the calculation should be performed.

For the tree code, accuracy is determined by the tolerance parameter $\theta$, for the FMM code the number of multipole terms $L$ and the refinement level $R$ have the same function. Schmidt and Lee[44] showed that the overall polynomial dependence of the computation on $N$, $L$, and $R$ is given by

$$P = aBL^2 + bNL^2 + cBL^4 + dB\left[g_1 + g_2\left(\frac{N}{B}\right) + g_3\left(\frac{N}{B}\right)^2\right], \qquad (65)$$

where $a, b, c, g_1, g_2$, and $g_3$ are machine-dependent parameters and $B$ is the number of boxes ($8^R$) on the highest refinement level. Interestingly, Eq. 65 shows that it is crucial that ratio $N/B$ is kept small, because only then is the $N^2$ dependence actually removed, leaving a dominant $O(N)$ scaling. In practice, this means increasing the number of levels as $N$ is increased, typically leading to a ratchet-like timing curve.

One of the assumptions of the FMM is that the distribution of the particles is more or less homogeneous. Nonuniform particle distributions would either require a high refinement level or a large number of terms in the multipole expansion, both resulting in higher overheads. Therefore, the original form of the fast multipole method is not very suitable for nonuniform distributions, or for dynamical systems which may develop large density contrasts over the course of time. To cope with this, adaptive fast multipole codes[49–51] have been developed, which are basically hybrids between the BH and FM algorithms.

## 5 Performance and Parallelism

Comparison of $N$-body algorithms is a dangerous business because unless one is completely impartial, there is a tendency to neglect one's least favourite scheme. A classic example is the direct $N$-particle summation, which can be performed in a time $N(N-1)/2$ by exploiting the 'action-reaction' symmetry of the Coulomb force-law. Overlooking this fact immediately makes the naive PP scheme a factor of 2 slower than it needs to be. Less trivial optimisations are also often neglected in the Ewald method for periodic systems, leading to wildly differing conclusions for the 'crossover' points at which alternative schemes – PME, FMM etc. – are faster.

### 5.1 Open Systems

Because periodicity necessitates additional complexity and overheads which differ for the Ewald and multipole schemes, we first consider the simpler case of open boundaries, where the system is surrounded by an infinite vacuum with no external forces present. Here we will compare the pure PP algorithm against the Barnes-Hut[35] tree algorithm (BH) and the standard Greengard FMM[36]. To make the comparison meaningful we perform tests for fixed accuracy, defined by the relative RMS particle-particle force error, estimated as follows:

$$\varepsilon_\alpha = \left\{\frac{\sum_{i=1}^{N_{test}}\left(f_{\alpha i}^a - f_{\alpha i}^r\right)^2}{\sum_{i=1}^{N_{test}}\left(f_{\alpha i}^r\right)^2}\right\}^{\frac{1}{2}},$$

where $f_{\alpha i}^a$ and $f_{\alpha i}^r$ are the components of the forces on particle $i$, evaluated using the approximate and direct (reference) methods respectively. For homogeneous, symmetric

density distributions, we can average over the three force components:

$$\varepsilon_f = \frac{1}{3}\sum_{\alpha}\varepsilon_\alpha.$$

Note that this is a stronger measure than, say the relative error in potential energy

$$\varepsilon_{pot} = \left|\frac{\Phi^a - \Phi^r}{\Phi^r}\right|,$$

where individual errors can sometimes cancel each other. Generally, error estimates based on $\varepsilon_f$ are larger than those found with $\varepsilon_{pot}$, and represent a more reliable and stringent measure for dynamical applications[52,53].

Benchmarks for these three algorithms performed on a SunBlade Sparc machine are shown in Fig. 24. The initial distribution used was a neutral, homogeneous sphere of randomly placed positive and negative charges. With the BH algorithm, setting the multipole acceptance parameter $\theta$ to 0.2 and 0.5 resulted in average force errors of 0.1% and 1% respectively, almost independent of $N$ – see Fig. 19. To achieve equivalent precision with the FMM is not just a matter of choosing the appropriate number of multipoles, because the refinement level needs to be adjusted too as $N$ is increased. Esselink[53] investigated this effect in some detail, and we quote adjusted timings from his FMM code for 7- and 4-term expansions. The above curves reveal two noticeable features. First, the Barnes-Hut

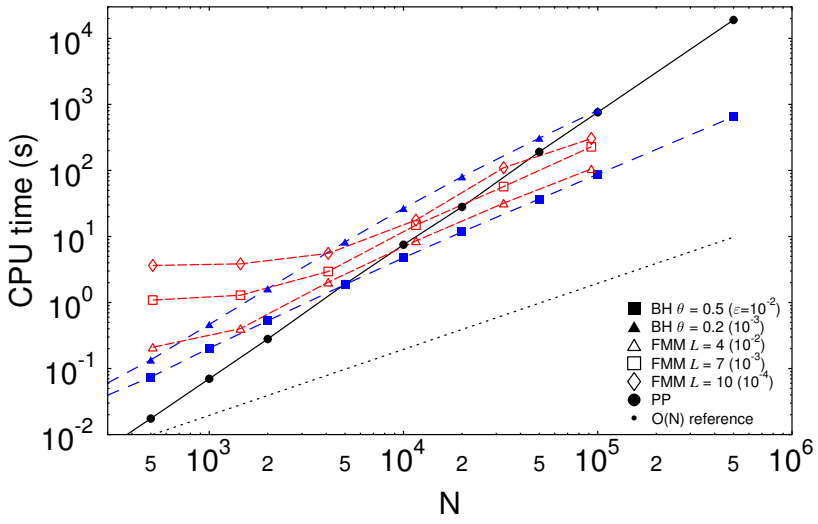



Figure 24. Comparison of computation time as a function of the number of simulation particles $N$ between particle–particle, hierarchical tree, and the fast multipole code for a 3-dimensional open system. The FMM timings are taken (and adjusted) from Esselink (1995)

algorithm is clearly hard to beat for 'low-precision' applications (where a force error of 1%

can be tolerated), displaying a breakeven point over PP of around 5000 for the BH code compared to about 10000 for FMM. If high precision is needed however (for, say Monte-Carlo configurational computations), then FMM comes into its own for particle numbers in excess of $10^5$.

### 5.2 Periodic Systems

As we hinted at before, timing comparisons for the classic, periodic lattice of charges with which we began this article have proved to be the most controversial in the past, with claimed breakeven points ranging from 300 to $10^5$! These discrepancies can usually be traced to differing levels of optimisation for both Ewald and FMM algorithms. Rather than attempt to implement a 'perfect' Ewald sum ourselves, we take timings from Esselink's paper[53], which describes several optimisations in some detail. The FMM timings, on the other hand, we draw from Schmidt & Lee's paper[44], where they also describe and implement a fully periodic version. The overhead caused by periodic boundaries is actually just a few percent if one uses the multipole expansion for the whole system for the periodic images. Figure 25 shows a comparison of the computation time required by the Ewald method, tree code, and fast multipole codes for a 3-dimensional, fully periodic system of charges. Extrapolating the FMM curves for these error levels (around $10^{-4}$ and $10^{-3}$ for the upper and lower curve respectively), one can estimate the breakeven point somewhere between $5\times10^4$ and $10^5$ - a little higher than for open systems. This value seems to be consistent with a more rigorous comparison by Solvason *et al.* for 2D periodic systems[15].

The other two curves shown are taken from a periodic tree code (Barnes-Hut-Ewald – BHE), developed by Pfalzner & Gibbon[54] for plasma physics applications. These appear to be comparable with timings reported for PME/P$^3$M codes[30,28,26], though the tree code is perhaps less accurate in the form used. Another interesting periodic BH derivation has been presented by Duan & Krasny[55], who also deduce a breakeven point of about $10^4$ particles.

### 5.3 Parallelisation Strategies

The widespread availability of parallel supercomputers has made $N$-body calculation very attractive as a simulation tool, even bringing direct $O(N^2)$ force summation into the realms of feasibility for a restricted set of problems[56]. The direct $N^2$ algorithm contains a natural parallelism, requiring a simple sharing of particles between the processors. Communication overheads can be minimised by passing partial results between processors in a ring (systolic loop), as described elsewhere in these proceedings[6].

The methods described in this review are all designed to yield a better *algorithmic scaling* than the brute force option, i.e. a speed-up relative to direct summation regardless of machine architecture. Having established a breakeven point for a given accuracy, it is important to know whether this will still hold on a massively parallel computer. In practice, we find that $N$-body algorithms have widely differing efficiencies when implemented in parallel, so we now briefly outline the main parallelisation strategies used to date.

The conventional Ewald method which we started with in Section 2 is quite easy to implement in parallel. Even with a modest number of processors, there is an immediate

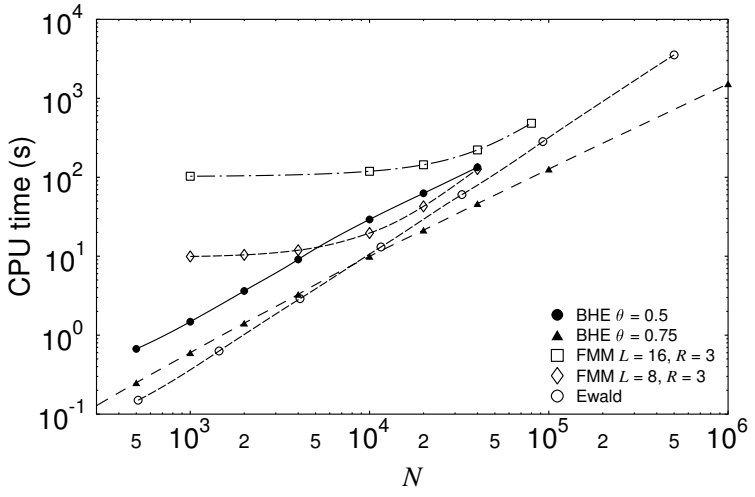



Figure 25. Comparison of computation time as a function of the number of simulation particles $N$ between particle–particle, hierarchical tree (BHE), and the fast multipole code for a neutral system of charges with periodic boundary conditions. The FMM timings are derived from Schmidt and Lee (1991).

opportunity for task sharing provided by the splitting of the sum into real- and reciprocal-space, which yields two more-or-less equally time-consuming contributions. The real-space part can be split further according to the standard PP recipe; the $k$-space sum can be reformulated as a particle sum multiplied by structure factors which can be computed independently on each processor[57].

Unfortunately, parallelism is not so clear-cut for the PME or P$^3$M methods: if anything it becomes less efficient, because the Ewald sum gets heavily and deliberately biased towards the now gridded Fourier component. On a scalar machine, the latter can be rapidly evaluated by FFTs, leading to the desired $O(N \log N)$-scaling. However, the binary, recursive nature of the FFT is not well suited to distributed-memory parallelism, and ultimately results in poor scalability[58]. An alternative P$^3$M scheme with very promising scaling characteristics has recently been suggested by Beckers *et al.* , in which the FFT is abandoned in favour of an iterative Poisson solver[59]. Parallel implementation of this scheme involves a standard spatial decomposition commonly found in fluid dynamics applications, in which a 'halo' of ghost cells a few grid points wide is placed around each processor domain. The halos act as communication buffers for grid information held by neighbouring processors needed for the local integration of Poisson's equation.

Multipole-based $N$-body methods require somewhat more effort to implement on parallel architectures, but have nonetheless attracted considerable attention because of the potentially rewarding prospect of Giga-particle simulation which modern Teraflop machines would offer. A parallel version of the original, non-adaptive 2D FMM was proposed by Greengard himself as early as 1990[60]. This scheme, based on task-sharing in each of the separate near- and far-field stages of the FMM, works well on shared memory machines,

but less efficiently on distributed memory systems. Nonetheless, these ideas inspired the first parallel FMM for MD simulation of macromolecules, developed and demonstrated by the Duke University group in the early 1990s[50,61].

Of all the algorithms described here, the Barnes-Hut tree code probably presents the biggest challenge for parallel implementation. At first sight, the hierarchical data structure would seem to rule out parallelism altogether, but it was soon realised that both tree construction and interaction lists could be at least vectorised on a level-by-level basis[62–64], leaving a straightforward $N \times N_{list}$ force summation to contend with. However, all this only works with shared memory. On distributed memory machines, the tree structure either has to be global to all processors – a very expensive and wasteful option – or somehow divided up equally among them. The problem is that the access to the *whole* tree is in principle needed to build an interaction list. Searching for nonlocal nodes on remote processors using the pointer addressing schemes typical of scalar tree codes would entail a huge communication overhead. This problem was recognised by Salmon and Warren[65,66], who practically reinvented the BH algorithm by scrapping pointers in favour of a set of unique binary keys to represent particle and node coordinates. Domain decomposition is reduced to cutting a list of keys sorted into an appropriate order. The main drawback of this scheme is that memory locations are accessed by mapping the large number of possible keys onto a hash table; a process that risks 'collisions' – two or more keys yielding the same address. Various ways have been proposed to minimise this effect, including sorting addresses according to access frequency, and distributing 'work' rather than tree-nodes to remote processors[67].

The trend in machine architecture is towards ever larger processor arrays – currently approaching the 10000 mark[68], which puts a premium on algorithm efficiency. Even a 1% inefficiency due to a non-parallel component or communication overhead can lead to severe performance deterioration for 1000 processors or more[6]. Which of the above algorithms is best suited to large-scale computations on large machines is really an open question. The mesh- and multipole-based methods may have a mathematically superior scaling, but are harder to implement efficiently in parallel, and will thus continue to present a challenge as computers increase in size.

## 6 Summary

In this article we have attempted to provide a guide to the alternative methods available for accelerating the force or potential calculation for long-range $N$-body problems. Although it is impossible to give any strict recommendations, we can draw up some general rules-of-thumb as to which scheme to choose. For the special but important case of periodic systems, some form of $P^3M$ is widely acknowledged as being faster than the classical Ewald sum for reasonable accuracy ($10^{-4}$ relative force error, say) – in the range $N = 10^4$–$10^5$. Thereafter, it may be worth investing in FMM, particularly if very high precision is desired for one-shot configurational calculations, for example. The lesser known Barnes-Hut-Ewald schemes may also be competitive with either of these methods for all $N$, though this remains to be demonstrated conclusively. For open systems, multipole methods are the only alternative, and have become routine in astrophysical and plasma physics applications. For static, high-precision problems where $N > 10^5$, FMM is again hard to beat. On the other hand, dynamical applications (MD) call for force-accuracy comparable with the in-

tegration scheme (typically 0.1–1%), which favours the much simpler BH algorithm. This segregation of requirements does not rule out future hybrid, adaptive multipole schemes suitable for implementing on massively parallel architectures.

## Acknowledgments

One of us (PG) acknowledges discussions on the intricacies of optimised FMM with H. Dachsel.

## References

1. P. P. Ewald, *Die Berechnung optischer und elektrostatischer Gitterpotentiale*, Ann. Phys. **64**, 253 (1921).
2. C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation*, (McGraw-Hill, New York, 1985).
3. R. L. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, (McGraw-Hill, New York, 1981).
4. T. Darden, D. York, and L. Pedersen, *Particle mesh Ewald: an N.log(N) method for Ewald sums in large systems*, J. Chem. Phys. **98**, 10089–10092 (1993).
5. S. Pfalzner and P. Gibbon, *Many Body Tree Methods in Physics*, (Cambridge University Press, New York, 1996).
6. G. Sutmann, "Classical molecular dynamics", in *Quantum Simulations of Complex Many-Body Systems: From Theory to Applications*, J. Grotendorst, Ed. (NIC, Jülich, 2002).
7. E. Madelung, *Das elektrische Feld in Systemen von regelmässig angeordneten Punkt-Ladungen*, Phys. Z. **19**, 524–532 (1918).
8. C. Kittel, *Introduction to Solid State Physics*, (Wiley & Sons, New York, 5th edition, 1976).
9. S. W. De Leeuw, J. W. Perram, and E. R. Smith, *Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants*, Proc. Roy. Soc. Lon. **373**, 27–56 (1980).
10. D. M. Heyes, *Electrostatic potentials and fields in infinite point charge lattices*, J. Chem. Phys. **74**, 1924–1929 (1981).
11. M. L. Boas, *Mathematical Methods in the Physical Sciences*, (Wiley & Sons, New York, 1st edition 1966).
12. R. P. Feynman, R. B. Leighton, and M. Sands, *The Feynman Lectures on Physics* vol. 1, (Addison-Wesley, Reading, Mass.,1963).
13. J. W. Perram, H. G. Petersen, and S. W. D. Leeuw, *An algorithm for the simulation of condensed matter which grows as the $N^{3/2}$ power of the number of particles*, Mol. Phys. **65**, 875–893 (1988).
14. J. Kolafa and J. W. Perram, *Cutoff errors in the Ewald summation formulae for point charge systems*, Mol. Sim. **9**, 351–368 (1992).
15. D. Solvason, J. Kolafa, H. G. Petersen, and J. W. Perram, *A rigorous comparison of the Ewald method and the fast multipole method in two dimensions*, Comp. Phys. Commun. **87**, 307–318 (1995).

16. M. J. Sangster and M. Dixon, *Interionic potentials in alkali halides and their use in simulations of the molten salts*, Adv. in Phys. **25**, 247–342 (1976).
17. D. Fincham, *Optimisation of the Ewald sum for large systems*, Mol. Sim. **13**, 1–9 (1994).
18. G. Rajagopal and R. J. Needs, *An optimized Ewald method for long-ranged potentials*, J. Comp. Phys. **115**, 399–405 (1994).
19. O. Bunemann, *Dissipation of currents in ionized media*, Phys. Rev. **115**, 503–517 (1959).
20. J. M. Dawson, *Particle simulation of plasmas*, Rev. Mod. Phys. **55**, 403–447 (1983).
21. R. Dendy, Ed., *Plasma Physics: An Introductory Course*, (Cambridge University Press, Cambridge, 1993).
22. C. K. Birdsall and D. Fuss, *Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation*, J. Comp. Phys. **3**, 494–511 (1969).
23. J. M. Dawson, *One-dimensional plasma model*, Phys. Fluids **5**, 445–459 (1962).
24. J. Eastwood, R. W. Hockney, and D. N. Lawrence, *P3M3DP: The three-dimensional periodic particle-particle / particle-mesh program*, Comp. Phys. Commun. **19**, 215–261 (1980).
25. M. P. Allen and D. J. Tildesley, *Computer simulations of liquids*, (Oxford University Press, Oxford 1987).
26. E. L. Pollock and J. Glosli, *Comments on $P^3M$, FMM, and the Ewald method for large periodic Coulombic systems*, Comp. Phys. Commun. **95**, 93–110 (1996).
27. H. Furukawa and K. Nishihara, *Reduction in bremsstrahlung emission from host, dense binary-ionic-mixture plasmas*, Phys. Rev. A **42**, 3532–3543 (1990).
28. H. G. Petersen, *Accuracy and efficiency of the particle mesh Ewald method*, J. Chem. Phys. **103**, 3668–3679 (1995).
29. B. A. Luty, I. G. Tironi, and W. F. van Gunsteren, *Lattice-sum methods for calculating electrostatic interactions in molecular simulations*, J. Chem. Phys. **103**, 3014–3021 (1995).
30. B. A. Luty, M. E. Davis, I. G. Tironi, and W. F. van Gunsteren, *A comparison of Particle-Particle Particle-Mesh and Ewald methods for calculating electrostatic interactions in periodic molecular systems*, Mol. Sim. **14**, 11–20 (1994).
31. T. Schlick, R. D. Skeel, A. T. Brunger, L. V. Kale, J. A. Board Jr., J. Hermans, and K. Schulten, *Algorithmic challenges in computational molecular biophysics*, J. Comp. Phys. **151**, 9–48 (1999).
32. U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee and L. G. Pedersen, *A smooth particle mesh Ewald method*, J. Chem. Phys. **103**, 8577–8593 (1995).
33. D. M. York, T. A. Darden, and L. G. Pedersen, *The effect of long-range electrostatic interactions in simulations of macromolecular crystals: a comparison of the Ewald and truncated list methods*, J. Chem. Phys. **99**(10), 8345–8348 (1993).
34. A. Appel, *An efficient program for many-body simulation*, SIAM J. Sci. Statist. Comput. **6**, 85 (1985).
35. J. Barnes and P. Hut, *A hierarchical $O(N \log N)$ force-calculation algorithm*, Nature **324**, 446–449 (1986).
36. L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comp. Phys. **73**, 325–348 (1987).
37. W. H. Press, ", in *The use of supercomputers in stellar dynamics*, P. Hut and S. L. W.

McMillan, Eds. (Springer, New York 1986), p. 184.
38. W. Benz, *Applications of smooth particle hydrodynamics (SPH) to astrophysical problems*, Comp. Phys. Commun. **48**, 97–105 (1988).
39. L. Hernquist, *Hierarchical* N*-body methods*, Comp. Phys. Commun. **48**, 107–115 (1988).
40. J. K. Salmon and M. S. Warren, *Skeletons from the treecode closet*, J. Comp. Phys. **111**, 136–155 (1994).
41. L. Greengard, *The rapid evaluation of potential fields in particle systems*, MIT Press Cambridge, Mass. (1988).
42. L. Greengard, *The numerical solution of the* N*-body problem*, Computers in Physics pp. 142–152 Mar./Apr. (1990).
43. J. Ambrosiano, L. Greengard, and V. Rokhlin, *The fast multipole method for gridless particle simulation*, Comp. Phys. Commun. **48**, 117–125 (1988).
44. K. E. Schmidt and M. A. Lee, *Implementing the fast multipole method in three dimensions*, J. Stat. Phys. **63**, 1223–1235 (1991).
45. J. D. Jackson, *Classical Electrodynamics*, (Wiley, New York, 2nd edition, 1975).
46. H. G. Peterson, D. Soelvason, J. W. Perram, and E. R. Smith, *The very fast multipole method*, J. Chem. Phys. **101**, 8870–8876 (1994).
47. C. A. White and M. Head-Gordon, *Derivation and efficient implementation of the fast multipole method*, J. Chem. Phys. **101**, 6593–6605 (1994).
48. H. Dachsel, private communication. See also Poster by same author at the NIC Winter School (2002).
49. J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Stat. Comput. **9**, 669–686 (1988).
50. J. A. Board Jr., J. W. Causey, J. F. Leathrum Jr., A. Windemuth, and K. Schulten, *Accelerated molecular dynamics simulation with the parallel fast multipole algorithm*, Chem. Phys. Lett. **198**, 89–94 (1992).
51. H. Cheng, L. Greengard, and V. Rohklin, *A fast adaptive multipole algorithm in three dimensions*, J. Comp. Phys. **155**, 468–498 (1999).
52. L. Hernquist, *Performance characteristics of tree codes*, Astrophys. J. Supp. **64**, 715–734 (1987).
53. K. Esselink, *A comparison of algorithms for long-range interactions*, Comp. Phys. Commun. **87**, 375–395 (1995).
54. S. Pfalzner and P. Gibbon, *A hierarchical tree code for dense plasma simulation*, Comp. Phys. Commun. **79**, 24–38 (1994).
55. Z.-H. Duan and R. Krasny, *An Ewald summation based multipole method*, J. Chem. Phys. **113**, 3492–3495 (2000).
56. R. Spurzem, *Direct N-body simulations*, J. Comp. Appl. Math. **109**, 407–432 (1999).
57. R. K. Kalia, S. de Leeuw, A. Nakano and P. Vashishta, *Molecular dynamics simulations of Coulombic systems on ditributed-memory MIMD machines*, Comp. Phys. Commun. **74**, 316–326 (1993).
58. A. Gupta and V. Kumar, *The scalability of FFT on parallel computers*, IEEE Trans. Parallel Dist. Systems **4**, 922–932 (1993).
59. J. V. L. Beckers, C. P. Lowe, and S. W. de Leeuw, *An iterative PPPM method for simulating Coulombic systems on distributed memory parallel computers*, Mol. Sim. **20**, 369–383 (1998).

60. L. Greengard and W. D. Groop, *A parallel version of the fast multipole method*, Comput. Math. Applic. **20**, 63–71 (1990).
61. "Distributed parallel multipole tree algorithm", Duke University Technical Report (2000), `http://www.ee.duke.edu/research/SciComp/Docs/`
62. J. E. Barnes, *A modified tree code: don't laugh; it runs*, J. Comp. Phys. **87**, 161–170 (1990).
63. J. Makino, *Vectorization of a treecode*, J. Comp. Phys. **87**, 148–160 (1990).
64. L. Hernquist, *Vectorization of tree traversals*, J. Comp. Phys. **87**, 137–147 (1990).
65. M. S. Warren and J. K. Salmon, "A parallel hashed oct-tree $n$-body algorithm", in *Supercomputing '93* Los Alamitos (1993) IEEE Comp. Soc. pp. 12–21.
66. M. S. Warren and J. K. Salmon, *A portable parallel particle program*, Comp. Phys. Commun. **87**(266–290) (1995).
67. A. Grama, V. Kumar, and A. Sameh, *Scalable parallel formulations of the Barnes-Hut method for $N$-body simulations*, Parallel Comp. **24**, 797–822 (1998).
68. Top 500 computer sites, `http://www.top500.org/`