Methods for efficient computer simulations of charged systems

Axel Arnold
Institute for Computational Physics
Universität Stuttgart

August 2010

1 1. Introduction

1.1 Electrostatics

We assume N charges $q_i e$ at positions r_i. This corresponds to a charge density

$$\rho e = \sum_{i=1}^{N} q_i e \delta(r - r_i). \tag{1.1}$$

If the charges move slowly, then we can ignore magnetic forces, and the force on each particle is

$$F_i = q_i e E(r_i) \tag{1.2}$$

and the total electrostatic energy

$$U_E = \int \rho(r) e E(r) dr, \tag{1.3}$$

where E is the electrostatic field with the properties

$$\nabla \cdot E = \rho e/\epsilon \tag{1.4}$$

$$\nabla \times E = 0. \tag{1.5}$$

The latter implies that E is the gradient of an up to a constant unique scalar electrostatic potential Φ such that $E = -\nabla\Phi$. Inserting into (1.4) gives

$$-\nabla^2 \Phi = \rho e/\epsilon, \tag{1.6}$$

which is a defining differential equation for Φ. For a single charge, it has the solution:

$$-\nabla^2 \frac{q_i e}{4\pi\epsilon} \frac{1}{r - r_i} = q_i e \delta(r - r_i)/\epsilon \tag{1.7}$$

It is convenient to define the *Bjerrum length*

$$l_\mathrm{B} = \frac{e^2}{4\pi\epsilon k_\mathrm{B} T}, \tag{1.8}$$

which is the distance at which two unit charges feel an electrostatic interaction equal to $1 k_\mathrm{B} T$; this means for example that by spontaneous thermal fluctuations, two oppositely charged unit charges can be separated by a distance up to l_B. The Bjerrum length in water at room temperature is 0.7nm due to its large relative permittivity of $\epsilon/\epsilon_0 \approx 80$.

Then the electrostatic forces and potential are

$$F_i = k_\mathrm{B} T \, l_\mathrm{B} \sum_{j \neq i} \frac{q_i q_j}{|r_i - r_j|^3} (r_i - r_j) \quad \text{and} \quad U_E = \frac{1}{2} k_\mathrm{B} T \, l_\mathrm{B} \sum_{i=1}^{N} \sum_{j \neq i} \frac{q_i q_j}{r_i - r_j}, \tag{1.9}$$

where the (infinite) self interactions of point charges are omitted. The differential equation for Φ is

$$-\nabla^2 \Phi = 4\pi \frac{l_\mathrm{B} k_\mathrm{B} T}{e} \rho. \tag{1.10}$$

In the following, the Bjerrum length and $k_\mathrm{B} T$will be omitted.

1.2 Boundary conditions and conditional convergence

To avoid finite size effects, computer simulations are typically performed with periodic boundary conditions. This means, we consider an infinite set of charges $q_i e$ at positions $r_i + n$, where

$$n = (kl_x, ll_y, ml_z) \quad \forall\, k, l, m \in \mathbb{Z} \tag{1.11}$$

and l_x, l_y, l_z are the periodic cell sizes in the three spatial dimensions. In the following, we use the short notation $n \in l\mathbb{Z}^3$, and $V = l_x \cdot l_y \cdot l_x$ denotes the volume of the periodic cell. We assume *charge neutrality*, that is

$$\sum_{i=1}^{N} q_i = 0. \tag{1.12}$$

We consider the electrostatic energy of the N primary charges with all others, that is

$$U_E = \frac{1}{2} \sum_{i=1}^{N} \sum_{n \in l\mathbb{Z}^3} \sum_{j=1}^{N}{}' \frac{q_i q_j}{r_i - r_j - n}, \tag{1.13}$$

where $'$ denotes that for $n = 0$, the $j = i$-term is omitted. While being the obvious extension of Eqn. (1.9) to a periodic system, this is in fact not a well defined equation. The reason is that the sum is not absolutely convergent; the value of U_E depends on the order of summation over k, l, m. Since we assume that our N particles are representative for the bulk of a large, macroscopic sample that is approximated by the periodic sum, a reasonable choice is a summation in radially ascending shells:

$$U_E = \frac{1}{2} \sum_{i=1}^{N} \sum_{S=0}^{\infty} \sum_{n^2=S} \sum_{j=1}^{N}{}' \frac{q_i q_j}{r_i - r_j - n}. \tag{1.14}$$

Different summation orders are possible, which result in different values for U_E. In addition, U_E can be defined via different routes. For example, one can solve the differential Eqn. (1.10) under periodic boundary conditions, then

$$U_E = \frac{1}{2} \sum_{i=1}^{N} q_i \Phi(r_i). \tag{1.15}$$

This definition of the electrostatic potential is called *intrinsic*. Finally, one can use *convergence factors* to define U_E:

$$U_E := \lim_{\beta \to 0} \frac{1}{2} \sum_{i=1}^{N} \sum_{S=0}^{\infty} f(S, \beta) \sum_{n^2=S} \sum_{j}{}' \frac{q_i q_j}{r_i - r_j - n}, \tag{1.16}$$

where the convergence factor $f(S, \beta)$ has the following properties:

1. $f(S, \beta)$ is a continuous function for $\beta \geq 0$
2. $f(S, 0) = 1$
3. $f(S + 1, \beta) \leq f(S, \beta)$ for all S, β
4. $0 \leq f(S, \beta) \leq 1$ for all S, β.

That is, *formally* we recover the original series for $\beta = 0$. However, if at fixed $\beta > 0$, $f(S, \beta)$ decays fast enough (for example exponentially), then

$$\sum_{S=0}^{\infty} f(S, \beta) \sum_{n^2=S} \sum_{j=1}^{N}{}' \frac{q_i q_j}{r_i - r_j - n}, \tag{1.17}$$

is absolutely convergent and well defined, so that also the limit (1.16) is well defined.

The convergence factor approach is the most general one, since it allows to obtain the same results as the other approaches; we will use this approach to prove some of the methods for evaluating the Coulomb sum. For example, the convergence factor $e^{-\beta S^2}$ corresponds to summation in ascending spherical shells. The convergence factor $e^{-\beta|r+S|^2}$ leads to the intrinsic potential.

However, the difference between summation orders is not as big as expected. Smith has shown that for example

$$U_{\text{spherical}} = U_{\text{intrinsic}} + \frac{2\pi}{3V} M^2 \tag{1.18}$$

$$U_{\text{slabwise}} = U_{\text{intrinsic}} + \frac{2\pi}{V} M_z^2, \tag{1.19}$$

where $U_{\text{spherical}}$ denotes the summation result for summation in ascending shells, U_{slabwise} diskwise summation in two dimensions, and after that symmetric summation in the remaining dimension, and $U_{\text{intrinsic}}$ the intrinsic solution.

Note that the *dipole term* $\frac{2\pi}{3V} M^2$ has in fact a physical interpretation, if written as

$$\frac{2\pi}{(2\epsilon_r' + 1)V} M^2, \tag{1.20}$$

where ϵ_r' is the dielectric permittivity of a medium that is supposed to surround the crystal (remember that we interpret the crystal as growing in ascending shells). For vacuum, $\epsilon_r' = 1$, so that we recover the term as in Eqn. (1.18). Another important value are metallic boundary conditions, for which the term vanishes. In case the system contains free ions, this is the only reasonable boundary condition, since otherwise this term imposes a harmonic potential that drives the particles back to an imaginary central cell.

Note also that for this term the unfolded positions r_i matter. For calculating the energy (1.13), it seems that the position r_i only matter up to multiples of l_x, l_y and l_z. After all, we are investigating a large, periodic crystal, and shifting r_i for example by l_x in the x-direction, the internal structure of the crystal will not change. However, the dipole term will change; moreover, it will be discontinuous, even if particle coordinates are modified only by multiples of l_*. Therefore, it is important to use unfolded, that is continuous coordinates for its calculation.

Instead of three dimensional periodicity, one can also consider two- or onedimensionally periodic systems; this is used for studying inter-/surfaces or wires and nanotubes. One can also generalize the periodicity to non-orthorombic simulation boxes. Another problem arises when for example calculating solvation free energies. In this case, a single charge is added to the system, which consequently becomes non-neutral. As one can easily, the energy of a non-neutral system is necessarily infinite. However, one can assume a homogeneous neutralizing background, which being homogeneous does not generate forces on the particles. Therefore, the dynamics of the system is unbiased by the background. But to calculate the energy, this background is necessary, since otherwise the result depends on the applied method, and even on non-physical tuning parameters.

Literature

[1] 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. R. Soc. Lond. A*, 373:27–56, 1980.

[2] E. R. Smith. Electrostatic energy in ionic crystals. *Proc. R. Soc. Lond. A*, 375:475–505, 1981.

[3] J. D. Jackson. *Classical Electrodynamics*. Wiley, New York, 3rd edition, 1999.

2 Ewald method

The Ewald method is *the* classical method for calculating electrostatic interactions in computer simulations with periodic boundary conditions. It dates back to 1921 and was original invented by Ewald to calculate crystal energies, especially the Madelung constant (by hand, not by computer, of course!). By carefully tuning the method parameters, the method nevertheless has a computational scaling of $N^{3/2}$ and is therefore significantly faster than a plain all-with-all N^2-force calculation. Nowadays, there are several other methods scaling like $N \log N$ or even N, but all are much less elegant than the Ewald methods, and have rather large overheads, so that for small systems, Ewald's method is still a good choice. Moreover, unlike its mesh-extensions, the Ewald method converges exponentially, that is to double the amount of valid digits in the result, one just needs to double the computational effort.

The main trick of the Ewald method is to split the two main problems of the Coulomb potential sum - the singularities at the positions of the charges, and the slow decay of the potential, which makes it conditionally convergent. To this aim, the delta distributions that represent the charge densities of the charges are replaced by Gaussian charge clouds of width α^{-1}:

$$q_i \delta(r - r_i) = q_i \left[\delta(r - r_i) - \rho_{\text{Gauss}}(r - r_i)\right] + q_i \, \rho_{\text{Gauss}}(r - r_i) \tag{2.1}$$

where the Gaussian cloud is defined as

$$\rho_{\text{Gauss}}(r) = \left(\frac{\alpha}{\sqrt{\pi}}\right)^3 e^{-\alpha^2 r^2}. \tag{2.2}$$

To make use of this splitting, we calculate the electrostatic energy via the electrostatic potential, that is as

$$U_E = \frac{1}{2} \sum_{j=1}^{N} q_j \Phi(r_j), \tag{2.3}$$

where

$$\Phi(r) = \sum_{j=1}^{N} \sum_{n \in l\mathbb{Z}^3}{}' \frac{q_j}{|r - r_j - n|}. \tag{2.4}$$

The $'$ on the sum denotes that the (infinite) terms with $r = r_j$ are omitted.

Fourier space sum

Although the sum over the Gaussians still leads to an only conditionally convergent sum, we can however solve the differential equation for the electrostatic potential, since unlike the original charge distribution, the Gaussians are smooth, and (1.10) can be solved for a periodic and smooth solution. Such a function can be conventiently represented as Fourier series:

$$f(r) = \frac{1}{l^3} \sum_{k \in \frac{2\pi}{l}\mathbb{Z}^3} \widehat{f}(k) e^{ikr}, \tag{2.5}$$

where

$$\widehat{f}(k) = \int_V dr \, f(r) e^{-ikr} \tag{2.6}$$

is the Fourier transform of f. Note that for simplicity we assume a cubic periodic cell V of size $l \times l \times l$. Since for any smooth periodic function

$$\nabla_r f(r) = \frac{1}{l^3} \sum_{k \in \frac{2\pi}{l}\mathbb{Z}^3} \widehat{f}(k) \nabla_r e^{ikr} = \frac{1}{l^3} \sum_{k \in \frac{2\pi}{l}\mathbb{Z}^3} ik\widehat{f}(k) e^{ikr}, \tag{2.7}$$

Poisson's equation for the Fourier series of the electrostatic potential Φ transforms into

$$k^2 \widehat{\Phi}(k) = 4\pi \widehat{\rho}(k). \tag{2.8}$$

We now consider the charge density given by the sum of all Gaussians, which has the Fourier series

$$\widehat{\rho}_G(k) = \int_V dr\, e^{-ikr} \sum_{j=1}^{N} q_j \sum_{n \in l\mathbb{Z}^3} \left(\frac{\alpha}{\sqrt{\pi}}\right)^3 e^{-\alpha^2 (r - r_j - n)^2} = \sum_{j=1}^{N} q_j e^{-ikr_j} e^{-k^2/4\alpha^2} \tag{2.9}$$

Note that

$$\widehat{\rho}_G(0) = \int_V dr \sum_{j=1}^{N} q_j \sum_{n \in l\mathbb{Z}^3} \left(\frac{\alpha}{\sqrt{\pi}}\right)^3 e^{-\alpha^2 (r - r_j - n)^2} = \sum_{j=1}^{N} q_j = 0 \tag{2.10}$$

due to charge neutrality; therefore, Eqn. (2.8) trivially holds, but does not allow to determine $\Phi(0)$. However, normalization requires that this constant term is 0. By insertion of Eqn. (2.9) into (2.8), we obtain the electrostatic potential due to the Gaussians Φ_G, and from this the interaction of the charges with these Gaussians:

$$U_F = \frac{1}{2} \sum_{j=1}^{N} q_j \Phi_G(r_j) = \frac{1}{2} \sum_{j=1}^{N} q_j \sum_{k \neq 0} e^{-ikr_j} \frac{4\pi}{k^2} \widehat{\rho}_G(k) = \frac{1}{2V} \sum_{k \neq 0} \left| \sum_{i=1}^{N} q_i e^{ikr_j} \right|^2 \frac{4\pi}{k^2} e^{-k^2/4\alpha^2}. \tag{2.11}$$

Real space sum

What remains is to calculate the contribution due to the interaction between the charges and the real, discrete charges and the charge-inverted Gaussians. The potential of the inverted Gaussians rapidly screens the potential of the point-like charges, so that the sum of both over the lattice is absolutely and quickly convergent; therefore, we can perform it efficiently in real space.

For this, we need is the potential also due to a single Gaussian. Again, we use Poisson's equation, but this time in real space:

$$-\frac{1}{r} \frac{\partial^2 r \Phi_{\text{Gauss}}(r)}{\partial r^2} = 4\pi \rho_{\text{Gauss}}. \tag{2.12}$$

Integration on both sides gives

$$-\frac{\partial r \Phi_{\text{Gauss}}(r)}{\partial r} = -\int_r^\infty dr 4\pi r \rho_{\text{Gauss}} = -\left(\frac{\alpha}{\sqrt{\pi}}\right)^3 \int_r^\infty dr 4\pi r e^{-\alpha^2 r^2} = -2\frac{\alpha}{\sqrt{\pi}} e^{-\alpha^2 r^2}. \tag{2.13}$$

And a second integration gives

$$\Phi_{\text{Gauss}}(r) = \frac{2\alpha}{\sqrt{\pi} r} \int_0^r dr e^{-\alpha^2 r^2} = \frac{\text{erf}(\alpha r)}{r}, \tag{2.14}$$

where erf denotes the error function.

We now compute the real space energy

$$\begin{aligned} U_R =& \frac{1}{2} \sum_{i \neq j} q_j q_i \left[\frac{1}{|r_i - r_j|} - \Phi_{\text{Gauss}}(|r_i - r_j|) \right] + \frac{1}{2} \sum_i q_i^2 \Phi_{\text{Gauss}}(0) \\ &+ \frac{1}{2} \sum_{i,j} q_j q_i \sum_{n \neq 0} \left[\frac{1}{|r_i - r_j - n|} - \Phi_{\text{Gauss}}(|r_i - r_j - n|) \right] \end{aligned} \tag{2.15}$$

where the first line are the contributions due to the primary image $n = 0$, in which we have to omit the singular self-energy of the point charges. We need therefore to evaluate two terms:

$$\Phi_{\text{Gauss}}(0) = \lim_{r\to 0} \Phi_{\text{Gauss}}(r) = \frac{\alpha}{\sqrt{\pi}} \tag{2.16}$$

and

$$\frac{1}{r} - \Phi_{\text{Gauss}}(r) = \frac{\operatorname{erfc}(\alpha r)}{r} \tag{2.17}$$

The complementary error function erfc decays exponentially fast. By choosing α sufficiently large, one can obtain U_R easily by a summation only over nearest images of each pair of charges.

The total (intrinsic) Coulomb energy is then approximately given by

$$\begin{aligned} U_E = U_{\text{intrinsic}} \approx \quad & \frac{1}{2V} \sum_{0<|k|^2<K^2} \left| \sum_{i=1}^{N} q_i e^{ikr_j} \right|^2 \frac{4\pi}{k^2} e^{-k^2/4\alpha^2} \\ & + \frac{1}{2} \sum_{\substack{i\neq j \\ |r_i - r_j|_{\text{m.i.}} < r_c}} q_i q_j \frac{\operatorname{erfc}(\alpha |r_i - r_j|_{\text{m.i.}})}{|r_i - r_j|_{\text{m.i.}}} \\ & - \frac{\alpha}{\sqrt{\pi}} \sum_i q_i^2, \end{aligned} \tag{2.18}$$

where we have introduced the computationally necessary truncations in k-space by cutoff K and in real space by cutoff r_c. The force on particle i can be easily obtained by taking the derivative with respect to r_i, since all sums are absolutely and fast converging.

2.1 Error estimates

What are now optimal values for K, R and α? Or given values, what is the numerical error we make? From Eqn. (2.18), it is easy to see that the real space cutoff should be inversely proportional to α, while the k-space cutoff should be proportional. This allows us to estimate the computational effort of the Ewald summation: Using a cell list-like approach and assuming homogeneously distributed particles at a fixed density, the computation time for the evaluation of the real space sum is Nr_c^3. At constant density, the volume is proportional to the number of particles. Therefore, the number of k-vectors we need to sum over in Eqn. (2.18) grows proportional to NK^3, and the computation time of the Fourier space sum is N^2K^3. Inserting the expected scalings with α, we get a total computation time $N\alpha^{-3} + N^2\alpha^3$, which is minimal for $\alpha = N^{-1/6}$. Therefore, the optimal scaling of the Ewald sum is $N^{3/2}$. Note that with increasing system size, the optimal Ewald cutoff grows. For reasonably large systems, the time spent for the Ewald real space sum will always dominate the calculation of the short-ranged forces.

This alone does not yet allow to reliably choose the parameters; for that, we need quantitative error estimates. However, there is no unique or optimal measure of accuracy. In molecular dynamics simulations, the main interest lies in force errors, which we will consider here, while in Monte Carlo simulations, one is concerned with the errors in the energy. One can be interested in either absolute or relative errors; we will consider the first. The reason is again practical — most applications, particulary in soft matter research, include a considerable level of thermal noise on the forces. As long as the absolute error in the electrostatic force is significantly below this noise level, the errors should not influence the system.

In the following, we discuss the force error estimates by Kolafa and Perram for the Ewald sum. Giving a general expression for the expected error is difficult; there are always pathological cases, in which the errors of the used methods are unusually high. However, this are only a few special configurations, which we will rarely encounter in a thermal simulation. Therefore, we assume the

most common case, namely that the charges are homogeneously and randomly distributed within the periodic cell V. Then, our goal is to calculate the root mean square (RMS) error

$$\Delta F = \sqrt{\langle (F^{\text{exact}} - F^{\text{Ewald}})^2 \rangle} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \Delta F_i^2}, \tag{2.19}$$

where $\Delta F_i = F_i^{\text{exact}} - F_i^{\text{Ewald}}$ denotes the error in the force on particle number i.

It is reasonable to assume that the error in the force on particle i can be written as

$$\Delta F_i = q_i \sum_{j \neq i} q_j \chi_{ij}, \tag{2.20}$$

that is, we assume that the error on F_i is a sum of errors stemming from the $N-1$ interactions with the other charges. χ_{ij} is a pairwise error, which is algorithm dependent; we assume that

$$\langle \chi_{ij} \cdot \chi_{ik} \rangle = \delta_{jk} \left\langle \chi_{ij}^2 \right\rangle = \delta_{jk} \, \chi^2, \tag{2.21}$$

i. e. that the contributions from different particles are uncorrelated and that the magnitude is independent of the particle properties, except for the charge prefactor. The first assumption is justified only for random systems, as we have assumed. Inserting into Eqn. (2.20) gives

$$\left\langle \Delta F_i^2 \right\rangle = q_i^2 \sum_{j \neq i} \sum_{k \neq i} q_j q_k \langle \chi_{ij} \cdot \chi_{ik} \rangle = q_i^2 \, \chi^2 \sum_{j=1}^{N} q_j^2, \tag{2.22}$$

which shows that the RMS force error has the form

$$\Delta F \approx \frac{\sum q_i^2}{\sqrt{N}} \chi, \tag{2.23}$$

where the factor χ is the only constant that is algorithm-dependent. We further simplify the problem by assuming that the real and Fourier space errors are uncorrelated; this is sensible, since the parts are calculated by very different types of algorithms. In this case, we can calculate their contributions separately:

$$\Delta F^2 = \Delta F_{\text{real}}^2 + \Delta F_{\text{Fourier}}^2 \approx \frac{\sum q_i^2}{\sqrt{N}} \left(\chi_{\text{real}} + \chi_{\text{Fourier}} \right). \tag{2.24}$$

By replacing the sums over n and k by an integral, both errors χ_{real} and χ_{Fourier} can be estimated. The resulting errors are:

$$\Delta F_{\text{real}} \approx \frac{\sum q_i^2}{\sqrt{N}} \frac{2}{\sqrt{r_c V}} e^{-\alpha^2 r_c^2} \qquad \text{and} \tag{2.25}$$

$$\Delta F_{\text{Fourier}} \approx \frac{\sum q_i^2}{\sqrt{N}} \frac{2\alpha}{\sqrt{\pi K V}} e^{-K^2/4\alpha^2}. \tag{2.26}$$

This error estimates do not only allow to estimate the error at given K, r_c and α, but can also be used to tune the Ewald sum optimally for given error goal ΔF. To this aim, one loops over Fourier-cutoffs K and determines the minimal α numerically from Eqn. (2.26), such that the RMS error is below $\Delta F/\sqrt{2}$. Then, Eqn. (2.25) allows to determine the minimal cutoff r_c, so that the real space error is also below $\Delta F/\sqrt{2}$. Now, one just needs to try through a couple of Fourier space cutoffs to see which combination of K and r_c is optimal for the given implementation and hardware.

2.2 Neutralization

As mentioned before, it is possible to consider systems with a net charge by assuming a homogeneous background that fills the entire simulation and neutralizes the system. This background contributes to the dipole term, which in a non-neutral system otherwise would not be translationally invariant. If the primary simulation box, and with that the neutralizing background, is located in $[0, l]^3$, then its dipole moment is $-\sum_{i=1}^{N} q_i l/2(1, 1, 1)$. In addition, the background contributes a constant, position dependent energy

$$U_N = -\frac{\pi}{2\alpha^2 V}\left(\sum_{i=1}^{N} q_i\right)^2, \tag{2.27}$$

which can for example be easily calculated by integrating the implicit pair interaction

$$U_{ij} = \frac{1}{V}\sum_{0<|k|^2<K^2} q_i q_j e^{ik(r_i-r_j)}\frac{4\pi}{k^2}e^{-k^2/4\alpha^2} + q_i q_j \sum_{n\in l\mathbb{Z}^3}\frac{\operatorname{erfc}(\alpha|r_i - r_j - n|}{|r_i - r_j - n|} \tag{2.28}$$

in r_i over the whole periodic cell V. Only the second sum contributes, and gives the above term. Note that this contribution depends on α, therefore it must not be neglected when calculating the energy of nonneutral systems, otherwise U_E will depend on the tuning parameter α; in the worst case, that is, if the algorithm is tuned for the underlying hardware, the result can be even hardware dependent.

Literature

[1] D. Frenkel. Colloidal systems: Playing tricks with designer "atoms". *Science*, 296:65, November 2002.

[2] 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. R. Soc. Lond., A*, 373(1752):27–56, oct 1980.

[3] Jiri Kolafa and John W. Perram. Cutoff errors in the Ewald summation formulae for point charge systems. *Molecular Simulation*, 9(5):351–368, 1992.

3 Mesh-accelerated Ewald methods (P^{3}M)

The Fourier transformations involved in Eqn. (2.18) are the most time consuming part of the Ewald sum. The mesh-accelerated Ewald methods are based on the idea to modify the problem in such a way that it permits application of the Fast Fourier Transformation (FFT). This reduces the complexity of the reciprocal part of the Ewald sum to $\mathcal{O}(N \log N)$ and allows for a constant cutoff in the real space, so that the real space computation time scales like $\mathcal{O}(N)$.

Performing the Fourier transformations in the reciprocal space part of the Ewald sum by FFT routines is by no means straightforward, and consists of four main steps:

1. The point charges with continuous coordinates have to be replaced by a grid based charge density, since the FFT is a discrete and finite transformation.

2. The potential has to be calculated in the discrete Fourier space by solving Poisson's equation; that is, by multiplication of the Fourier transformed charge density with the Green's function. It is neither obvious nor true that the best grid approximation to the continuum solution of the Poisson equation is achieved by using the continuum Green's function $4\pi/k^2$.

3. The electric field has to be calculated by differentiation from the electric potential. There are at least three possibilities for implementing this differentiation, which differ in accuracy and speed.

4. Finally, the forces on the particle have to calculated from the electric field that is known only on the discrete grid points. This can – under certain circumstances – lead to unwanted violations of Newton's third law. They can be anything between harmless and disastrous.

There exist three major mesh-based Ewald summation methods – similar in spirit but different in detail, namely in how the four steps above are performed. The oldest is the original particle-particle–particle-mesh (P^{3}M) method of Hockney and Eastwood, and then there are two variants, namely the Particle Mesh Ewald (PME) method of Darden *et al.* and an extension of the latter by Essmann *et al.*, which is usually referred to as Smooth Particle Mesh Ewald (SPME). Deserno *et al.* have shown how the three methods differ in detail, and it was demonstrated that the oldest method, namely the original P^{3}M algorithm is actually the most accurate one. Since in addition error estimates exist, this mesh method should be the preferred method of choice, and will be introduced here.

3.1 P^{3}M in a Nutshell

The P^{3}M method maps the system onto a mesh, such that the necessary Fourier transformations can be accomplished by Fast Fourier routines. At the same time the simple Coulomb Green function $4\pi/k^2$ is adjusted to make the result of the mesh calculation most closely resemble the continuum solution.

The first step, i.e., generating the mesh based charge density $\rho_{\mathbb{M}}$ (defined at the mesh points $r_p \in \mathbb{M}$), is carried out with the help of a charge assignment function W:

$$\rho_{\mathbb{M}}(r_p) = \frac{1}{h^3} \sum_{i=1}^{N} q_i W(r_p - r_i). \tag{3.1}$$

Here h is the mesh spacing, and the number of mesh points $N_{\mathbb{M}} = L/h$ along each direction should preferably be a power of two, since in this case the FFT is most efficient. The charge assignment

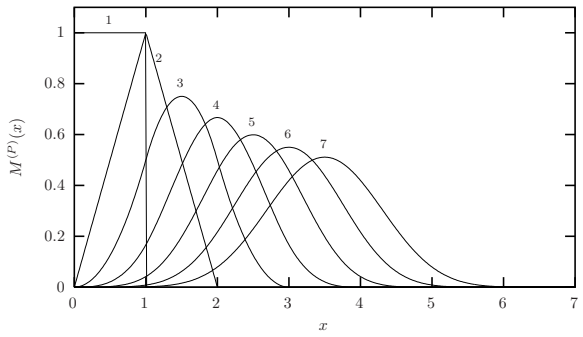


Figure 3.1: Sketch of the first 7 cardinal-B-splines $M^{(P)}(x)$, parameterized by P. Note that the charge assignment functions $W^{(P)}(x)$ for the P^{3}M algorithm are just the "centered" B-splines.

function is classified according to its order P, i.e. between how many grid points – per coordinate direction – each charge is distributed. For W a cardinal B-spline is chosen, which is a piecewise polynomial function of weight one. The order P gives the number of sections in the function. The first 7 cardinal-B-splines are sketched in Fig. 3.1. Their Fourier transforms are

$$\widehat{W}(k) = h^3 \left(\frac{\sin(\frac{1}{2}k_x h)}{\frac{1}{2}k_x h} \frac{\sin(\frac{1}{2}k_y h)}{\frac{1}{2}k_y h} \frac{\sin(\frac{1}{2}k_z h)}{\frac{1}{2}k_z h} \right)^P \tag{3.2}$$

The second and third step, i. e. solving Poisson's equation and deriving the mesh-based electric field $E(r_p)$ from it, happen simultaneously. There exist several alternatives for implementing the differentiation on a lattice; here we will restrict ourselves to the case of ik-*differentiation*, that is, multiplying the Fourier transformed potential with ik. In this case $E(r_p)$ can be written as

$$E(r_p) \;=\; \overleftarrow{\mathrm{FFT}} \left[-\mathrm{i}k \;\times\; \widehat{G}_{\mathrm{opt}} \;\times\; \overrightarrow{\mathrm{FFT}} \left[\rho_{\mathbb{M}}\right] \right] (r_p). \tag{3.3}$$

In words, $E(r_p)$ is the *backward* finite Fourier transform of the product of $-ik$, the *forward* finite Fourier transform of the mesh based charge density $\rho_{\mathbb{M}}$ and the so-called optimal influence function $\widehat{G}_{\mathrm{opt}}$, given by

$$\widehat{G}_{\mathrm{opt}}(k) = \frac{\mathrm{i}k \cdot \sum_{m\in\mathbb{Z}^3} \widehat{U}^2(k + \frac{2\pi}{h}m) \widehat{R}(k + \frac{2\pi}{h}m)}{|k|^2 \left[\sum_{m\in\mathbb{Z}^3} \widehat{U}^2(k + \frac{2\pi}{h}m)\right]^2}, \tag{3.4}$$

where the true, analytic reference force is derived from (2.18) as

$$\widehat{R}(k) := \; -\mathrm{i}k \frac{4\pi}{k^2} e^{-k^2/4\alpha^2} \tag{3.5}$$

and the dimensionless Fourier transform of the B-spline is

$$\widehat{U}(k) := \widehat{W}(k)/h^3. \tag{3.6}$$

The last step of P^{3}M, the back-interpolation of the forces onto the particles, is performed done again using the B-splines. The force on particle i is determined as

$$F_i = q_i \sum_{r_p\in\mathbb{M}} E(r_p) W(r_i - r_p). \tag{3.7}$$

The sum extends over the complete mesh $\mathbb{M}$; however, since the B-splines have compact carrier, the sum in fact only extends over a small vicinity of r_i.

Although the presented formulas (3.1–3.7) look somewhat complicated, they are rather easy to implement step by step. If the real space cutoff r_c is chosen small enough, so that the real space contribution (the second sum in Eqn. (2.18)) can be calculated in order $\mathcal{O}(N)$, the complete algorithm is of order $\mathcal{O}(N \log N)$.

3.2 The error measure of Hockney and Eastwood

While the real space error estimate of Kolafa and Perram of course also applies to the P^{3}M real space sum, the four steps involved in any particle mesh calculation introduce completely different errors than the simple k-space truncation of the standard Ewald sum. In fact, being a discrete Fourier transform, the P^{3}M k-space sum is *not* truncated at all. However, there are new sources of errors, originating, e. g., from discretization, interpolation or aliasing[1] problems. Since these contributions are not independent of each other (reducing one might enhance another), the only reasonable demand is the minimization of the *total* error at given computational effort.

The most interesting ingredient of the P^{3}M method is the optimal influence function from Eqn. (3.4). It is constructed such that the result of the mesh calculation is as close as possible to the solution of the original continuum problem. More precisely, the P^{3}M method is derived from the requirement that the resulting Fourier space contribution to the force minimizes the the following error measure Q:

$$Q := \frac{1}{h^3} \int_{h^3} dr_1 \int_V dr \big[F(r; r_1) - R(r)\big]^2 \tag{3.8}$$

$F(r; r_1)$ is the Fourier space contribution of the force between two unit charges at positions r_1 and $r_1 + r$ as calculated by the P^{3}M method (note that due to broken rotational and translational symmetry this does in fact depend on the coordinates of *both* particles), and $R(r)$ is the corresponding exact reference force (whose Fourier transform is just Eqn. (3.5)). The inner integral over r scans all particle separations, whereas the outer integral over r_1 averages over all possible locations of the first particle within a mesh cell. Obviously, up to a factor L^{-3} this expression is just the mean square error in the force for two unit charges, in other words, the quantity χ^2 from Eqn. (2.22). Inserting into Eqn. (2.23), the RMS force error of an N particle system is given by

$$\Delta F \approx \sum q_i^2 \sqrt{\frac{Q}{NV}}. \tag{3.9}$$

It is important to realize that Hockney and Eastwood not only provide a closed expression for the optimal influence function $\widehat{G}_{\text{opt}}$, but also a closed expression for the corresponding "optimal error" $Q_{\text{opt}} = Q[\widehat{G}_{\text{opt}}]$:

$$Q_{\text{opt}} = \frac{1}{L^3} \sum_{k \in \mathbb{M}} \Bigg\{ \sum_{m \in \mathbb{Z}^3} \Big| \widehat{R}(k + \frac{2\pi}{h} m) \Big|^2 \tag{3.10}$$

$$- \frac{\Big| \mathrm{i} k \cdot \sum_{m \in \mathbb{Z}^3} \widehat{U}^2(k + \frac{2\pi}{h} m) \widehat{R}^*(k + \frac{2\pi}{h} m) \Big|^2}{|k|^2 \Big[\sum_{m \in \mathbb{Z}^3} \widehat{U}^2(k + \frac{2\pi}{h} m) \Big]^2} \Bigg\}. \tag{3.11}$$

Here, the asterisk denotes the complex conjugate.

Admittedly, Eqn. (3.11) looks rather complicated. Still, in combination with Eqn. (3.9) it gives the RMS force error of the Fourier contribution of the P^{3}M method. After all, the computation of Q_{opt} and that of $\widehat{G}_{\text{opt}}$ are quite similar. It should be emphasized that the formula (3.11) for the optimal

[1] A finite grid cannot represent arbitrarily large k-vectors. Instead, they are folded back into the first "Brillouin zone" and distort there the true spectrum. This effect is usually referred to as "aliasing".

Q-value and the optimal influence function (3.4) are of a very general nature, and can be applied to for different charge assignment functions, reference forces or other differentiation schemes.

With the real space error estimate by Kolafa and Perram and the k-space error estimate by Hockney and Eastwood at hand, it is easy to determine the optimal value of the splitting parameter α *a priori* just from the system parameters N, $\sum q_i^2$ and L and the tuning parameters of the algorithm r_c, $N_{\mathbb{M}}$, P. Just like for the standard Ewald method, a good approximation to the optimal α can be obtained by requiring the real and k-space RMS force errors to be equal, and a similar tuning routine can be applied, although now the two parameters $N_{\mathbb{M}}$ and P both need to be tried out.

3.3 Parallelization

Being an order $\mathcal{O}(N \log N)$-method, P^{3}M and other mesh-based Ewald methods are well suited to study even large systems with many thousands of particles. This quickly raises the question of parallelization, i. e. employing N_P processors to compute the electrostatic energy simultaneously. Just as with the charge interpolation and differentation, there are several ways of parallelization; we briefly present here one that we found to scale rather well even on large computers with hundreds of processors.

The real space part of the Ewald sum is a short-ranged potential, for which several good parallelization strategies exist. In fact, every parallel Molecular Dynamics code has such a strategy built in, or rather, is built around such a strategy. And to be able to scale up to hundreds of processors, almost all codes use a domain decomposition and cell lists, often combined with Verlet lists. These methods reach the ideal N/N_P scaling with modern fast networks, at least in weak scaling, i. e. constant number of particles per processor. Moreover, communication only occurs between neighboring processors in a 3D toroidal structure, which is efficiently handled by most hardware.

The domain decomposition strategy also allows to conveniently split up the work load of charge assignment and force interpolation, by using the same domain decomposition also for the k-space mesh. What remains, is computing the 3D Fourier transform. The Fourier transforms are typically performed by highly efficient libraries, e. g. the excellent FFTW. However, this library at present does not scale very well when it comes to parallel 3D Fourier transforms. And it is the parallelization of the 3D Fourier transform, which is in fact the major bottle neck. To understand this, one has to see that a 3D Fourier transform of a mesh of $N_{\mathbb{M}}^3$ total points consists of three times performing $N_{\mathbb{M}}^2$ 1D Fourier transforms of length $N_{\mathbb{M}}$. To perform the latter, all data should be available on one processor. This means, that to perform the 1D Fourier transforms along the different axes, one has to completely exchange the data between all the processors, requiring rather inefficient global communication.

To our knowledge the most efficient way to implement the 3D Fourier transform is as follows:

1. redistribute the 3D domain-decomposed mesh such that the processors form a 2D mesh in the x, y-plane, and each processor obtains all data of its domain, in particulary always has one or more *full* columns along the z-axis. Calculate the Fourier transforms along the z-axis.

2. redistribute such that the processors form a 2D mesh in the x, z-plane and have full columns along the y-axis. Calculate the Fourier transforms along the y-axis.

3. redistribute such that the processors form a 2D mesh in the y, z-plane and have full columns along the y-axis. Calculate the Fourier transforms along the x-axis.

After applying the optimal influence function and the difference operator,

4 distribute the data back into the original, 3D domain-decomposed mesh.

Literature

[1] M. Deserno and C. Holm. How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines. *J. Chem. Phys.*, 109:7678, 1998.

[2] M. Deserno and C. Holm. How to mesh up Ewald sums. II. An accurate error estimate for the Particle-Particle-Particle-Mesh algorithm. *J. Chem. Phys.*, 109:7694, 1998.

[3] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles.* IOP, London, 1988.

[4] FFTW website, `http://www.fftw.org`.

4 MMM, MMM2D, MMM1D

The algorithm MMM[1] by R. Strebel and R. Sperb will be the first non-Ewald type method that we will consider, although it is still based on a Fourier transform. However, MMM uses the convergence factor $e^{-\beta|r_{ij}+n|}$, which as discussed in the introduction, gives the intrinsic energy

$$U_E = U_{\text{intrinsic}} = \frac{1}{2}\lim_{\beta\to 0}\sum_{n\in l\mathbb{Z}^3}\sum_{i,j=1}^{N}{}' \frac{q_i q_j e^{-\beta|r_{ij}+n|}}{|r_{ij}+n|} =: \frac{1}{2}\lim_{\beta\to 0}\sum_{i,j=1}^{N} q_i q_j \phi_\beta(r_{ij}), \tag{4.1}$$

where the screened pairwise potential is defined as

$$\phi_\beta(r) = \sum_{n\in l\mathbb{Z}^3}{}' \frac{e^{-\beta|r+n|}}{|r+n|}. \tag{4.2}$$

This potential is called screened, since from the physical point of view the Coulomb interaction is replaced by a screened Coulomb interaction with screening length $1/\beta$. $U_E N$ is the energy in the limit of infinite screening length. This also explains why this solution recovers the intrinsic solution without polarization contribution — even if we assume a very large screening length, any polarization at infinite distance is still screened by it.

For particles sufficiently separated in the z–axis we can use the the *Poisson formula*

$$\sum_{n\in l\mathbb{Z}} f(x+n) = \frac{1}{l}\sum_{k\in\frac{2\pi}{l}\mathbb{Z}} \widehat{f}\,(k)\, e^{\mathrm{i}kx} \tag{4.3}$$

using the transforms

$$\begin{aligned} \widehat{\left(\frac{e^{-\beta\sqrt{c^2+x^2}}}{\sqrt{c^2+x^2}}\right)} &= 2\,\mathrm{K}_0\left(c\sqrt{\beta^2+k^2}\right) \quad \text{and} \\ \widehat{\left(\mathrm{K}_0\left(\beta\sqrt{c^2+x^2}\right)\right)} &= \pi\frac{e^{-c\sqrt{\beta^2+k^2}}}{\sqrt{\beta^2+k^2}}, \end{aligned} \tag{4.4}$$

where K_0 is the so-called modified Bessel function of order 0. This allows us to Fourier transform the potential along both x and y, while the summation in z can be performed directly by a geometric series. We obtain the *far formula* as

$$\begin{aligned} \phi_\beta(r) &= \frac{2}{l_x}\sum_{p\in\frac{2\pi}{l_x}\mathbb{Z}}\left(\sum_{l,m\in l\mathbb{Z}}\mathrm{K}_0\left(\sqrt{\beta^2+p^2}\sqrt{(y+l)^2+(z+m)^2}\right)\right)e^{\mathrm{i}px} \\ &= \frac{2\pi}{l_x l_y}\sum_{p,q\in\frac{2\pi}{l}\mathbb{Z}}\sum_{m\in l_z\mathbb{Z}}\frac{e^{-\sqrt{\beta^2+p^2+q^2}|z+m|}}{\sqrt{\beta^2+p^2+q^2}}e^{\mathrm{i}px}e^{\mathrm{i}qy} \\ &= \frac{2\pi}{l_x l_y}\sum_{\substack{p,q\in\frac{2\pi}{l}\mathbb{Z}\\ p^2+q^2>0}}\frac{e^{f_{pq}z}+e^{f_{pq}(l_z-z)}}{f_{pq}\left(e^{f_{pq}l_z}-1\right)}e^{\mathrm{i}px}e^{\mathrm{i}qy} \\ &\quad + \frac{2\pi}{l_x l_y}\left(\frac{1}{l_z}z^2 - z + \frac{l_z}{6}\right) + \frac{\pi}{V}\beta^{-2} + \mathcal{O}_{\beta\to 0}(\beta), \end{aligned} \tag{4.5}$$

[1] Even the authors of the method have no idea what this acronym stands for.

where we abbreviated $f_{pq} = \sqrt{p^2+q^2}$. Note the singularity in β; this singularity prohibits us from exchanging the limit and the sum in Eqn. (4.1). However, this singularity is independent of the particle coordinates; therefore, when summing it up over a charge neutral set of particles, this singularity exactly cancels out.

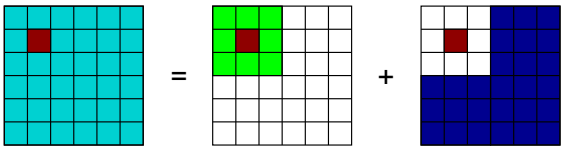
The advantage of this far formula is that it allows for a product decomposition into components of the particles, just as in the Ewald sum. Therefore one just has to calculate the sum over all these exponentials on the left side and on the right side and multiply them together, which can be done in $\mathcal{O}(N)$ computation time. As can be seen easily, the convergence of the series is excellent as long as z is sufficiently large. By symmetry one can choose the coordinate with the largest distance as z to optimise the convergence. However, if particles are spatially really close, we need a different formula.

For sufficiently small distance we obtain after some lengthy maths the *near formula* as

$$\begin{aligned}
\phi(r) = &\frac{4\pi}{l_x l_y} \sum_{\substack{p,q\in\frac{2\pi}{l}\mathbb{Z}\\ p,q>0}} \frac{\cosh(f_{pq}|z|)}{f_{pq}\left(e^{f_{pq}l_z}-1\right)} e^{\mathrm{i}px} e^{\mathrm{i}qy} \\
&+ \frac{4}{l_x} \sum_{p\in\frac{2\pi}{l_x}\mathbb{Z}} \sum_{l\in l_y\mathbb{Z}} \left(\mathrm{K}_0(p\sqrt{(y+l)^2+z^2}) + \mathrm{K}_0(p\sqrt{(y+l)^2+z^2})\right) cos(px) \\
&- \frac{2}{l_x} \sum_{n\geq 1} \frac{b_{2n}(2\pi)^{2n}}{2n(2n)!} \Re\left[\left(\frac{z+\mathrm{i}y}{l_y}\right)^{2n}\right] \\
&+ \frac{1}{l_x} \sum_{n\geq 0} \binom{-\frac{1}{2}}{n} \frac{\left(\psi^{(2n)}(1+x/l_x)+\psi^{(2n)}(1-x/l_x)\right)}{(2n)!} \left(\frac{y^2+z^2}{l_x^2}\right)^n \\
&- 2\log(4\pi) +' \frac{1}{|r|} + \frac{\pi}{V}\beta^{-2} + \mathcal{O}_{\beta\to 0}(\beta),
\end{aligned} \tag{4.6}$$

where b_n denotes the Bernoulli numbers, and $\psi^{(m)}(x)$ the polygamma function of m-th order. Note that the self contribution $1/r$ is added explicitely, so that it can be easily omitted for the calculation of the self interaction term. Eqn. (4.6) is derived using the same convergence factor approach as used for Eqn. (4.5), and consequently the same singularity in β is obtained. This is important since otherwise the charge neutrality argument does not hold and the limit $\beta \to 0$ could not be performed.

The near formula does not split as nicely as the far formula, therefore one cannot straightforwardly exploit the far formula, as we need to consider particle pairs according to their spatial positions differently. Nevertheless, we will explain now how to obtain a computational order of $\mathcal{O}(N \log N)$, that is, the same as the P^{3}M method. A simple implementation segments the simulation box in $B = S^3$ smaller boxes or cells. For all particles the interactions within the cell itself and the 26 neighbouring cells are treated using the near formula, while for the rest the far formula is used. In two dimensions this looks like this:



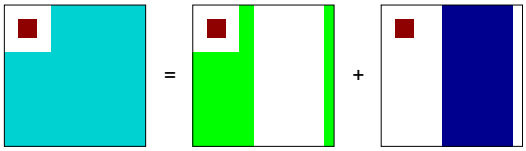
The interactions of the red cell with the light green cells is done via the near formula, while all the dark blue cells are treated using the far formula. One first determines the product decomposition components for each cell and then adds them up over all pairs of cells which are not neighbours.

For this simple approach, we can again estimate the computation time. Here, the cutoff of the far formula summation has to be chosen inversely proportional to the particle distance, as can be easily seen from Eqn. (4.5). Using the algorithm described above, the minimal distance of two particles calculated with the far formula is l_z/S. Therefore for a constant pairwise error the Fourier space cutoff

R has to be chosen proportional to S. This leads to a calculation time for the far formula of $\mathcal{O}(NS^2)$. The near formula has to be used for $\mathcal{O}(N^2S^{-3})$ particle pairs. Since the calculation time for the near formula is practically parameter independent, this is also the scaling of the calculation time. The total computation time has a minimum for $S \sim N^{1/5}$, resulting in an overall computation time scaling of $\mathcal{O}(N^{\frac{7}{5}})$. Upper error bounds can be found easily by approximating the sums by integrals, similar to the MMM2D/1D methods described below.

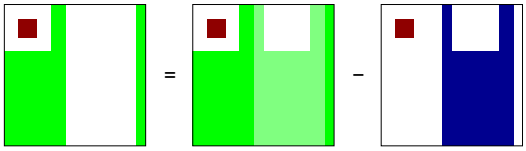
To decrease the computational effort of MMM down to $\mathcal{O}(N \log(N))$ as promised, we use the periodicity in all spatial dimensions to maximize the minimal distance at which the far formula has to be applied. This is an important point, since it explains, why for partially periodic dimensions MMM does not achieve the same favorable scaling, although the approach itself can be transferred, as we will see.

In the following we assume that the number of cells per side is a power of 2, i. e. $S = 2^L$. The main idea is to increase l_z/S not by decreasing S, but rather by increasing l_z. This is possible due to the periodic boundary conditions. The idea will be presented graphically in two dimensions, and as before, we calculate the interactions with the small red cell. First the primary simulation cell will be divided into small cells along the x–coordinate:



The particles in the right part are far away and can be calculated using the far formula with a reasonably small cutoff.

The other half will now be calculated using a cell length of $l_x/2$. This introduces artificial particles in the right part, which are just copies of the particles in the left side. Their contribution can be subtracted easily together with the contribution of the real particles in the right half.



Now we are left with a cell containing roughly $N/2$ particles (provided the simulation cell is filled homogeneously) and cell dimensions $l_x/2 \times l_y \times l_z$. For this system we apply the same trick again to one of the other axes, e. g. y, then to z and again to x and so on, until the calculation using the near formula is more efficient than another subdivision. The subdivision occurs in the coordinate which occurs in the exponential. Therefore the shift of the particle coordinates is actually only a multiplication.

In pseudo code the algorithm for a subdivision step looks like this:

for each (p, q)
  for each cell c
    calculate the coefficients ξ_c **of the product decomposition of cell** c
  end
  for each row r
    calculate $\Xi_r = \sum_{c \text{ in row } r} \xi_c$
  end
  for each row r
    calculate $\Xi_r = \sum_{\text{rows } s \text{ further than } l_z/2 \text{ from row } r} \Sigma_s$
    add to Ξ_r **the** Σ_r **of the rows closer than** $l_z/2$ **but not adjacent**
      for row r **multiplied by the shift factor**
  end
  for each cell c
    use the Ξ_r **and the** σ_c **of the adjacent rows of all cells** c'
      not adjacent to cell c **multiplied by the shift factor**
      to calculate the contribution to the energy
      from the far particles and the artificial images
  end
end

This algorithm can be further optimised with respect to computation time, but even in the present form one can see that it its implementation is more demanding than the Ewald or multipole methods. The implementation is even more complex as the presented code should work with all three coordinates symmetrically for reuse in the subdivision steps. The calculation time for each of the subdivision steps is proportional to half of the number of particles left, i. e. $N/2, N/4, N/8, \ldots$. To maintain a constant calculation time of the near formula, one has to ensure that $2^L \sim N$ or $L \sim \log N$. Therefore the overall computation time is $\mathcal{O}(NL) = \mathcal{O}(N \log N)$.

4.1 MMM2D

MMM can also be applied to systems with only two periodic dimensions and one non-periodic one, which we assume to be z without loss of generality. This method we call MMM2D, since although the formulas are very similar to MMM, the algorithmic implementation is different.

The *far formula* of MMM2D is

$$\phi(r) = \frac{2\pi}{l_x l_y} \sum_{\substack{p,q \in \frac{2\pi}{l}\mathbb{Z} \\ p^2+q^2>0}} \frac{e^{-f_{pq}|z|}}{f_{pq}} e^{\mathrm{i}px} e^{\mathrm{i}qy} \; - \; \frac{2\pi}{l_x l_y}|z|, \tag{4.7}$$

omitting again a constant singularity in β. As before the far formula is well convergent for $|z| \gg 0$,

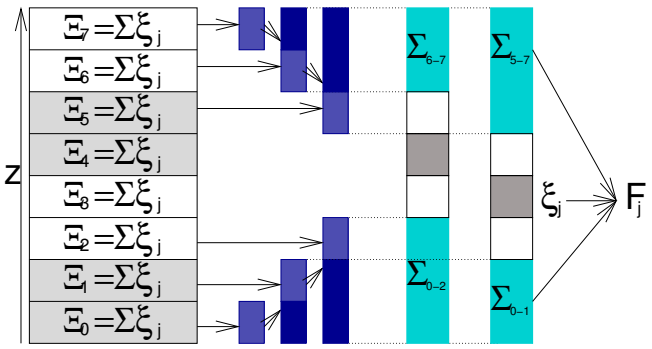


Figure 4.1: Schema of the evaluation of the MMM2D far formula. We need the sums Ξ_c, and their sums Σ over all slices above and below each slice. These can be computed in two loops generating the sums from above and below. The blue rectangles represent the growing sums from above and below. The dark grey slices on the right side mark two exemplary slices for which the far sum is calculated using the turquoise sums over the slices above and below.

but does not converge for small $|z|$. In this case, we use the *near formula*

$$\begin{aligned}\phi(r) = &\frac{4}{l_x}\sum_{l\in l_y\mathbb{Z}}\sum_{p\in\frac{2\pi}{l_x}\mathbb{N}}\left(\mathrm{K}_0(p\sqrt{(y+l)^2+z^2})+\mathrm{K}_0(p\sqrt{(y-l)^2+z^2}\right)\cos(px)\\ &-\frac{2}{l_x}\sum_{n\geq 1}\frac{b_{2n}(2\pi)^{2n}}{2n(2n)!}\Re\left[\left(\frac{z+\mathrm{i}y}{l_y}\right)^{2n}\right]\\ &-\frac{1}{l_x}\sum_{n\geq 0}\binom{-\frac{1}{2}}{n}\frac{\psi^{(2n)}(2+\frac{l_x}{x})+\psi^{(2n)}(2-\frac{l_x}{x})}{(2n)!}\left(\frac{y^2+z^2}{l_x^2}\right)^n\\ &+\frac{1}{\sqrt{(x+l_x)^2+y^2+z^2}}+\frac{1}{\sqrt{(x-l_x)^2+y^2+z^2}}+{\sum}'\frac{1}{|r|}-\frac{2}{l_x}\log\left(4\pi\frac{l_x}{l_y}\right).\end{aligned} \tag{4.8}$$

We cannot apply the same algorithmic tricks as used in MMM for 3D periodic boundary conditions, since these tricks require the periodicity of all coordinates. However, the far formula still allows for a product decomposition and can be evaluated in linear time with respect to the number of particles with constant cutoffs.

To draw advantage from this linear computation time scaling, one splits the system into B equally sized layers along the nonperiodic z–coordinate, as depicted in Figure 4.1. Now for all particle in one slice, the interactions with the neighboring slices are calculated using the near formula, while for the further distant slices the far formula is used. This ensures that the far formula is used only for particle pairs more distant than l_z/B, where l_z in this case is the total system height, not the period in z (which is non-periodic!).

Assume we calculate the forces for a slice c. For the slices far from c, that is, the ones that are treated using the far formula, we make use of the product decomposition. This means, that we need to calculate the sum of eight terms of the form

$$\xi_j^{\pm,\pm,\pm} = e^{\pm f_{pq} z_j} e^{\pm \mathrm{i}px_j} e^{\pm \mathrm{i}qy_j} \tag{4.9}$$

over all particles j in the slices far above and below c. As shown in Figure 4.1, the calculation of these sums can be done two loops, one summing up the slices from below, one from above. This is also convenient for parallelization: each computing node only needs the total sums from all nodes above and below, plus in addition the sums of the two slices that are directly adjacent to it. These values are very few and can be easily broadcasted.

There is one technically important point: The term $\exp(-f_{pq}|z|)$ is obviously always smaller than 1. However, the product decomposition requires the calculation of $e^{\pm f_{pq} z_j}$ which might be both extremely large or low. Both cases cause numerical problems, since the exponents of IEEE floating point numbers are limited. This problem can be avoided by normalizing the sums with appropriate prefactors. That is, one calculates for example for each slice

$$\Xi_c^{\pm,\pm,\pm} = \sum_{j \in \text{slice}c} e^{\pm f_{pq}(z_j - c\,h_l)} e^{\pm \mathrm{i} p x_j} e^{\pm \mathrm{i} q y_j}, \tag{4.10}$$

where $h_l = h/B$ is the height of a slice, and we assume that that slice 0 is located at zero. In other words, for each slice, we normalize the coordinates to the bottom coordinate of the slice. When calculating the sums over all slices above and below, we shift the bases accordingly. For the sum from below, we shift for example relative to the top slice:

$$\Sigma_s^{+,\pm,\pm} = \sum_{c=0}^{s} \Xi_c^{+,\pm,\pm} e^{f_{pq}(c-s)h_l} = \Sigma_{s-1}^{+,\pm,\pm} e^{f_{pq} h_l} + \Xi_s^{+,\pm,\pm}. \tag{4.11}$$

It is easy to see that for the slices below, only the terms with positive exponent are needed, for the once above, only the once with negative exponent.

In pseudo code the (non-parallel) MMM2D algorithm looks like this:

for each (p, q)
  for each slice c
    for each particle j
      calculate $\xi_j^{\pm,\pm,\pm} = e^{\pm f_{pq}(z_j - c h_l)} e^{\pm \mathrm{i} p x_j} e^{\pm \mathrm{i} q y_j}$
    end
    calculate $\Xi_c^{\pm,\pm,\pm} = \sum \xi_j^{\pm,\pm,\pm}$
  end
  for slice $s = 0 \ldots B-1$
    $\Sigma_s^{+,\pm,\pm} = \Sigma_{s-1}^{+,\pm,\pm} e^{f_{pq} h_l} + \Xi_s^{+,\pm,\pm}$
  end
  for slice $s = B-1 \ldots 0$
    $\Sigma_s^{-,\pm,\pm} = \Sigma_{s+1}^{-,\pm,\pm} e^{-f_{pq} h_l} + \Xi_s^{-,\pm,\pm}$
  end
  for each slice c
    for each particle j
      calculate its interaction with far cells using $\xi_j^{\pm,\pm,\pm}$, $\Sigma_{c-2}^{+,\pm,\pm}$ **and** $\Sigma_{c+2}^{+,\pm,\pm}$
      for each particle i **in slices** $c-1, \ldots, c+1$
        calculate interaction of particle j **with particle** i **using near formula**
    end
  end
end

Let us estimate the computation time scaling of MMM2D. Assuming that N/B particles are located in every slice, one can show that the far formula scales like $\mathcal{O}(B^2 N) + \mathcal{O}(B^3)$. The time for the calculation of the near formula for a single particle pair is nearly constant, leading to a scaling of $\mathcal{O}(N(2N/B))$. Minimizing the total time with respect to the number of slices B leads to $B \propto N^{1/3}$, yielding an asymptotic optimal computational time of $\mathcal{O}(N^{\frac{5}{3}})$.

As for all the other methods, we are interested in error estimates for MMM2D. It is rather tedious to calculate the parameter χ^2 of formula (2.22), but one can relatively easily determine an upper bound

for it. The derivation of the error estimates is straight forward, and we present only the results here. For the maximal pairwise force error of the far formula, one obtains the upper bound

$$\tau_F^{\text{far}} = \frac{e^{-2\pi R|z|}}{|z|}\left(2\pi R + 2(\frac{1}{l_x}+\frac{1}{l_y}) + \frac{1}{|z|}\right), \tag{4.12}$$

where R is the (p,q) cutoff radius of the far formula. A similar expression can also be found for the energy error. For the near formula, one has separate error estimates for the sum of Bessel terms, the complex sum and the polygamma sum, which one can use to determine the individual cutoffs and by that limit the overall χ. For the Bessel sum, the maximal force error is

$$\tau_F^{\text{Bessel}} = \frac{16\pi}{l_x^2}\,\mathrm{K}_1(l_y L)\left(l_x\frac{e^{\pi l_y/l_x}}{\pi l_y}\left(l_x\frac{L+\frac{1}{l_y}}{\pi}-1\right) + \sum_{p=1}^{\lceil\frac{l_x L}{\pi}\rceil-1} p e^{-\pi l_y p/l_x}\right) \tag{4.13}$$

where the summation takes place over $0 < p < \frac{L}{\pi u_x}$ and $0 < l < \frac{l_x L}{2\pi p} + 1$. For the summation of the complex sum up to order N, one obtains the maximal error bound

$$\tau_F^{\text{Bernoulli}} = \frac{16}{l_x l_y}\left(\frac{y^2+z^2}{l_y^2}\right)^{N-1} \leq \frac{16\sqrt{2}}{l_x l_y} 2^{-N}. \tag{4.14}$$

The polygamma sum finally is a sum of alternating coefficients with monotonously decreasing value, and therefore by Leibniz' criterion, the absolute value of last term taken into account is an upper error bound.

Note that the error distribution of MMM2D is non-uniform in z; particles near the system boundaries in z will have smaller errors than particles in the center; therefore it is vital to tune MMM2D for rather small errors, especially since increasing MMM2D's performance is rather inexpensive.

4.2 MMM1D

For completeness, we also give the MMM equivalent algorithm for one dimensionally periodic systems, which can for example model nanotubes or wires. In this case, we have only a single coordinate for Fourier transformation, so that we are left with the Bessel function in the far formula. Therefore, neither near nor far formula can be split up, and the overall computational effort is still $\mathcal{O}(N^2)$ for a plain all-by-all summation. Nevertheless, we need both formulas, since none of them is convergent for all distances.

In the following, the periodic dimension is z without loss of generality. Then, the *far formula* is

$$\phi(r) = \frac{4}{l_z}\sum_{p\in\frac{2\pi}{l_z}\mathbb{N}} \mathrm{K}_0(p\sqrt{x^2+y^2})\cos(pz) - \frac{2}{l_z}\log\left(\frac{\sqrt{x^2+y^2}}{2l_z}\right) - \frac{2\gamma}{l_z} + \mathcal{O}(\beta), \tag{4.15}$$

where we as usual omitted a singular term of $\mathcal{O}(\log\beta)$. The *near formula* is given by

$$\begin{aligned}\phi(r) = &-\frac{1}{l_z}\sum_{n\geq 0}\binom{-\frac{1}{2}}{n}\frac{\left(\psi^{(2n)}(2+z/l_z)+\psi^{(2n)}(2-z/l_z)\right)}{(2n)!}\left(\frac{x^2+y^2}{l_z^2}\right)^n \\ &-\frac{2\gamma}{l_z} + \frac{1}{\sqrt{x^2+y^2+(z+l_z)^2}} + \frac{1}{\sqrt{x^2+y^2+(z-l_z)^2}} +' \frac{1}{|r|}.\end{aligned} \tag{4.16}$$

The evaluation is very simple, since one simply has to loop over all pairs of charges, and use the far or near formula, depending on the distance of the particles. Clearly, the resulting computational scaling of $\mathcal{O}(N^2)$ prohibits to employ the methods for systems larger than a couple of hundred particles. However, the numerical prefactor is rather small, so that for small systems, the method is rather competitive. Finally note, that of course there are error estimates for MMM1D, just like for MMM or MMM2D.

Literature

[1] R. Sperb. An alternative to ewald sums - part 1: Identities for sums. *Molecular Simulation*, 20(3):179–200, 1998.

R. Sperb. An alternative to ewald sums - part 2: The coulomb potential in a periodic system. *Molecular Simulation*, 22(3):199–212, 1999.

R. Sperb and R. Strebel. An alternative to ewald sums part 3: Implementation and results. Technical Report 2000-02, ETH Research Report, 2000.

[2] A. Arnold and C. Holm. MMM2D: A fast and accurate summation method for electrostatic interactions in 2d slab geometries. *Comput. Phys. Commun.*, 148(3):327–348, 1 November 2002.

[3] A. Arnold and C. Holm. A novel method for calculating electrostatic interactions in 2D periodic slab geometries. *Chem. Phys. Lett.*, 354:324–330, 2002.

[4] A. Arnold and C. Holm. MMM1D: A method for calculating electrostatic interactions in 1D periodic geometries. *J. Chem. Phys.*, 123(12):144103, September 2005.

5 Electrostatic Layer Correction (ELC)

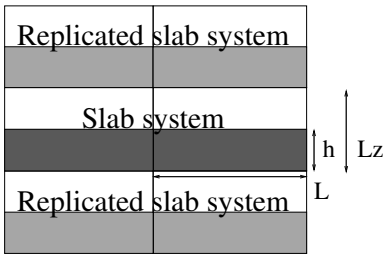


Figure 5.1: Schematic representation of a fully periodically replicated slab system

For large numbers of particles the computation time scaling should not have a power of N larger than 1. So far such a scaling is only known for methods for three dimensional periodicity. Therefore, there have been soon attempts to use a 3D Ewald sum for these slab problems. The main idea is to fill only parts of the simulation box with charges and to leave some space empty, in an attempt to decouple the interactions in the third dimension (compare also Fig. 5.1). Since each image layer is globally neutral, one hopes that their interactions decay as they become more and more distant, i.e. as the size of the gap is increased. In this way one could make use of any advanced 3D Ewald implementation, for example P^3M.

However, in a naive implementation, even large gap sizes will result in large errors. This is due to the order of summation for the three dimensional Coulomb sum, which is spherical by convention. This order implies that with increasing shell cutoff S, the number of image shells grows faster than the number of shells of the primary layer, namely $\mathcal{O}(S^3)$ versus $\mathcal{O}(S^2)$. In other words, we include the unwanted terms faster than the actually wanted terms. Also the image layers are not really infinite charged sheets, but are truncated due to the cut-off. Similarly, a fully periodic, that is, intrinsic, solution in the non-periodic direction can never lead to the correct potential. Yeh and Berkowitz already suggested that this problem can be solved by changing the order of summation to planewise summation. However, this approach requires a gap that is at least as large as the real slab system, which makes the computation still half as fast as a 3D system of the same number of charges.

In the following a new expression, called the electrostatic layer correction (ELC) term, for the error produced by the artificial images will be derived. From this expression one can estimate the error produced by the method proposed by Yeh et al., and one can tune the gap size according to the desired accuracy. But that is only a byproduct. Its primary advantage is that it can be used to numerically subtract the contributions of the unwanted image layers, which can be evaluated with computation time $\mathcal{O}(N)$. Therefore it can be combined with any conventional method for three dimensions without decreasing the computation time scaling. The term is very fast to evaluate, and allows one to use a gap between the layers that is typically not more than 10% of the slab height, thereby speeding up the computation time considerably.

As for MMM2D, we consider a system where the non-periodic dimension is z; however, for ELC, we need to require that all particles stay in a slab of dimension $0 \leq z \leq h < l_z$; l_z will the artificial periodicity that we introduce, and h the system height. We artificially replicate the system, that is, we introduce copies of the primary layer such that all charges also appear at position $r + n$, where $n \in 0 \times 0 \times l_z\mathbb{Z}$); this includes also the periodic images in the x, y-plane. Since we assume a charge

neutral system, the additional image layers (those layers above or below the original slab system) are charge neutral, too. Now let us consider the n^{th} image layer which has an offset of nl_z to the original layer. If nl_z is large enough, each particle in the n^{th} layer and its replicas in the x, y-plane can be viewed as constituting a homogeneous charged sheet of charge density $\sigma_j = \frac{q_j}{l_x l_y}$. The potential of such a charged sheet at distance z is $2\pi\sigma_j|z|$; the interaction of a pair of image layers located at $\pm nl_z$, $n \gg 0$ with a charge q_i in the primary layer is therefore

$$2\pi q_i \sum_{j=1}^{N} \sigma_j(|z_j - z_i + nl_z| + |z_j - z_i - nl_z|) = 4\pi q_i n l_z \sum_{j=1}^{N} \sigma_j = 0. \tag{5.1}$$

The only errors occurring are those coming from the approximation of assuming homogeneously charged, infinite sheets instead of discrete charges. This assumption should become better when increasing the distance nl_z from the central layer. Naturally, this argument only holds when using planewise summation. Yeh and Berkowitz stated that a gap size of at least h is normally sufficient to obtain an moderately accurate result. However, no theoretical estimates exist for the error introduced by the image layers, which could justify this statement; we will see that it is in fact not true. Therefore one might be forced to use even larger gaps to assure that no artifacts are produced by the image layers. One simple deducible artifact is that the pairwise error will be position dependant, similar to MMM2D. Particles in the middle of the slab will see no effect of the image layers due to symmetry, and particles near the surface will encounter for the same reason the largest errors, which is definitely an unwanted feature for studying surface effects.

The ELC term is the energy contribution of the artificially introduced image layers, that is

$$E_{lc} = -\frac{1}{2} \sum_{m'>0} \sum_{m=\pm m' l_z} \sum_{n \in (l\mathbb{Z})^2 \times m} \sum_{i,j=1}^{N} \frac{q_i q_j}{|r_i - r_j - n|}. \tag{5.2}$$

Since we assume that the image layers all are separated from the main layer by at least $l_z - h$, we can use the far formula of MMM2D to calculate the interaction of the primary layer with the image layer m as

$$E_{lc}(m) = -\frac{1}{2} \sum_{i,j=1}^{N} q_i q_j \frac{2\pi}{l_x l_y} \sum_{\substack{p,q \in \frac{2\pi}{l}\mathbb{Z} \\ p^2+q^2>0}} \frac{e^{-f_{pq}|z_i - z_j - ml_z|}}{f_{pq}} e^{\mathrm{i}px} e^{\mathrm{i}qy} \; - \; \frac{2\pi}{l_x l_y} |z_i - z_j - ml_z|. \tag{5.3}$$

The sum over m to obtain E_{lc} can be performed analytically. Note that the terms $\frac{2\pi}{l_x l_y}|z_i - z_j - ml_z|$ are exactly the homogeneous sheet potential, which we have seen to cancel out for charge neutral systems (see Eqn. (5.1)). The summation over m of the remaining sums over (p, q) is performed using the geometric series (as these sums are absolutely convergent, exchanging the summation over m and the summations over (p, q) is possible). For $m > 0$ we obtain, ignoring the terms independent of m,

$$\sum_{m \geq 1} e^{-f_{pq}|z + ml_z|} = \frac{e^{-f_{pq} z}}{e^{f_{pq} l_z} - 1}, \tag{5.4}$$

where $z = z_i - z_j$. For $m < 0$ we obtain

$$\sum_{m \leq -1} e^{-f_{pq}|z + ml_z|} = \frac{e^{f_{pq} z}}{e^{f_{pq} l_z} - 1}, \tag{5.5}$$

since we required that $|z| \leq l_z$.

Combining the two terms for $\pm m$ we obtain

$$E_{lc} = \sum_{i,j=1}^{N} q_i q_j \frac{2\pi}{l_x l_y} \sum_{\substack{p,q \in \frac{2\pi}{l}\mathbb{Z} \\ p^2+q^2>0}} \frac{\cosh(f_{pq}(z_i - z_j))}{f_{pq}(e^{f_{pq} l_z} - 1)} e^{\mathrm{i}p(x_i - x_j)} e^{\mathrm{i}q(y_i - y_j)} \tag{5.6}$$

The forces can be obtained by simple differentiation since the sums are absolutely convergent. Although the form in Eqn.(5.6) has a much better convergence than the original form in Eqn.(5.2), its main advantage is a linear computation time with respect to the number of particles N. This is achieved similar to the far formula of MMM2D, using the addition theorems for the exponentials rsp. hyperbolic cosine. For the calculation of the ELC term, no tricks such as the splitting of the simulation box into slices is necessary. In the terms of MMM2D, we just need to calculate the terms $\xi_j^{\pm,\pm,\pm}$ for each particle, and their total sums over all particles $\Xi_j^{\pm,\pm,\pm}$. Parallelization is also trival, since we deal with simple sums over all nodes, for which efficient reduction operations exist in the MPI standard.

In pseudo code the calculation of the ELC term on a single node looks like this:

for each (p,q)
  for each particle j
    calculate $\xi_j^{(\pm,\pm,\pm)}$
  end
  $\Xi^{(\pm,\pm,\pm)} = \sum_j \xi_j^{(\pm,\pm,\pm)}$
  for each particle j
    calculate particle interaction from $\xi_j^{(\pm,\pm,\pm)}$ **and** $\Xi^{(\pm,\pm,\pm)}$
  end
end

In a multiprocessor environment a MPI *reduce all* operation would be necessary to implement the sums $\Xi^{(\pm,\pm,\pm)}$.

5.1 Error estimates

Here, we adapt the error formulas for the far formula to our newly developed ELC formula given by Eqn. (5.6). We will show that using ELC the errors will be highest for the particles near the borders of the simulation box instead of lowest as for MMM2D. Therefore again the maximal pairwise error seems a more reasonable error estimate than the conventionally used RMS force error. But since for the widely used Ewald methods the RMS force error is the standard error measure, this error measure plays a more prominent role for ELC. It was already described for MMM2D how such an error measure can be estimated from the maximal pairwise error.

While the error bounds for MMM2D were only used to tune the algorithm, the error estimates for ELC can also be used to obtain an error bound for the slab–wise method of Yeh and Berkowitz, and hence one can determine "a priori" the necessary gap size to reach a preset precision. Therefore we also have to deal with small cutoffs, especially the case when no terms of E_{lc} are added.

The summation is performed only over all (p,q) vectors contained in $2\pi l_x\mathbb{Z} \times 2\pi l_y\mathbb{Z}$, where $p^2+q^2 \leq R^2$. An upper bound for the absolute value of the summands is

$$\left| \frac{\cosh(f_{pq}z)}{f_{pq}(e^{f_{pq}l_z}-1)} e^{\mathrm{i}px} e^{\mathrm{i}qy} \right| \leq e^{-2\pi f_{pq}l_z} \frac{\cosh(f_{pq}z)}{f_{pq}(1-e^{-f_{pq}l_z})} \leq e^{-2\pi f_{pq}l_z} \frac{\cosh(f_{pq}h)}{f_{pq}(1-e^{-f_{pq}l_z})}. \tag{5.7}$$

The sum over all these upperbounds can then be performed by an careful approximation of the sum by an integral. We find an upper bound for the maximal pairwise energy error as

$$\Delta E \leq \frac{1+2\frac{\frac{1}{l_x}+\frac{1}{l_y}}{R}}{e^{Rl_z}-1} \left(\frac{\exp(Rh)}{l_z-h} + \frac{\exp(-Rh)}{l_z+h} \right), \tag{5.8}$$

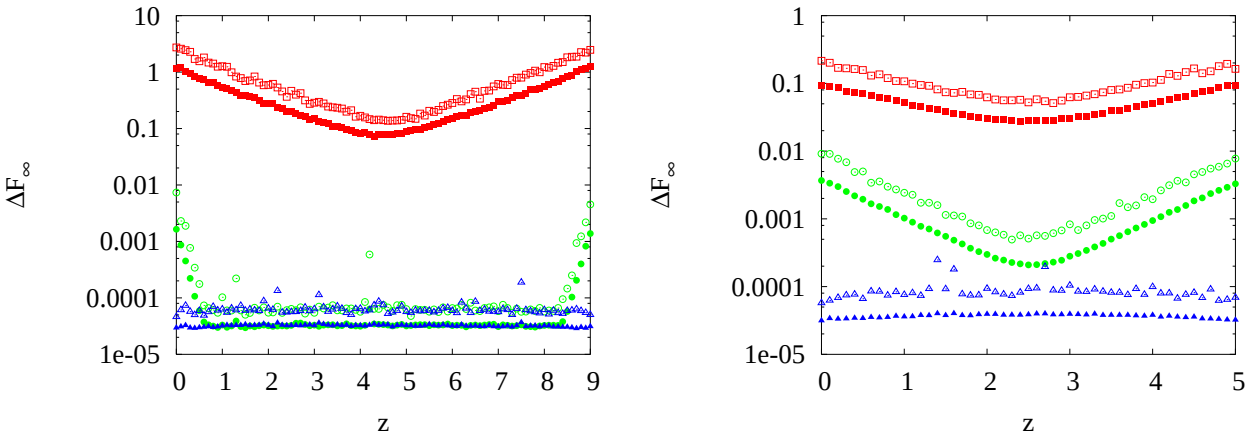


Figure 5.2: Absolute force error ΔF_∞ as a function of the particle z–coordinate for a homogeneous random system of 1000 particles in a box of size $10 \times 10 \times h$ with a system height $h = 9$ (left graph) and 5 (right graph). The red rectangles denote results for $R = 0$, the green circles for a maximal error of 10^{-2}, corresponding to $R = 1$ for $h = 9$ and $R = 0.1$ for $h = 5$, and the blue triangles for a maximal error of 10^{-4}, corresponding to $R = 1.8$ rsp. $R = 0.3$. Open symbols show the maximal error that occurred within all particles with similar z–coordinates, the filled symbols show the RMS force error of these particles.

and the RMS force error estimate for the forces is given by

$$\Delta F \leq \frac{\sum q_i^2}{\sqrt{N}} \frac{1}{2(e^{Rl_z}-1)} \left(\left(R + 2\left(\frac{1}{l_x}+\frac{1}{l_y}\right) + \frac{1}{l_z-h} \right) \frac{\exp(Rh)}{(l_z-h)} + \left(R + 2\left(\frac{1}{l_x}+\frac{1}{l_y}\right) + \frac{1}{l_z+h} \right) \frac{\exp(-Rh)}{(l_z+h)} \right). \tag{5.9}$$

An upper bound for the error produced by the method of Yeh and Berkowitz can be obtained by using $R = 2\pi \min(\frac{1}{l_x}, \frac{1}{l_y})$. Our error estimates show that the error drops exponential both with R and l_z. The decay in R means that it is easy to achieve high accuracies with our layer correction formula, while the decay in l_z shows that the slab-wise summation method can achieve good accuracies without increasing l_z too much. But the error formulas also show that for $l_z >> h$ not the fraction l_z/h dominates the error, but rather $Rl_z \sim l_z \min(\frac{1}{l_x}, \frac{1}{l_y})$. Therefore the actually important fractions are l_z/l_x and l_z/l_y and *not* l_z/h as was claimed by Yeh and Berkowitz[1].

Note that Eqn.(5.7) shows that the error in the potential or the force for a single particle will be largest if it is located near the gap, since there $|z_i - z_j|$ will be maximal, compare Fig 5.2. This effect will increase with increasing R, therefore, it is important to tune ELC to errors small enough to be negligible even at the system boundaries. Just measuring the overall RMS force error as for 3D systems might be misleading, especially, since for thin films or surfaces, mostly the particles at the surfaces are of interest.

If one assumes an mesh-based Ewald method to be used for the 3D periodic system, one can estimate the optimal l_z as follows. The computation time of the mesh method is proportional to $l_x l_y l_z$. For ELC, R is proportional to $1/(l_z - h)$, therefore the computation time spent with ELC is proportional to $l_x l_y/(l_z - h)^2$. The total computation time is therefore minimized by a constant gap size independent of the box dimensions.

[1]In fact, already the extreme case of $h = 0$ immediately shows that this cannot be true, as a conventional method for three dimensional periodicity will not deliver exact results for a purely planar system.

5.2 Neutralization

Similar to P^{3}M it is possible to consider non-neutral systems. To this aim, one first neutralizes the system using a homogeneous background for the calculation using the 3D method, e.g. P^{3}M . This homogeneous background gives rise to an unwanted force that drives the particles towards the boundaries of the system. However, one can calculate this force analytically, and subtract it again. Note that the background also contributes to the dipole term for slabwise summation. It should now read:

$$\frac{2\pi}{V}\left[\sum_{i=1}^{N} q_i \left(z_i - \frac{l_z}{2}\right)\right]^2, \tag{5.10}$$

where the shift accounts for the neutralizing background, which in the slab system is a homogeneous charge in the range from 0 to l_z. For an Ewald-type method, the neutralizing background term U_N according to Eqn. (2.27) is required in addition. The interaction energy of the background with the slab system and itself is

$$\frac{2\pi}{V}\left[\sum_{i=1}^{N} q_i \sum_{j=1}^{N} q_j z_j \left(L_z - z_j\right) - \frac{1}{3}\left(\sum_{i=1}^{N} q_i\right)^2\right], \tag{5.11}$$

which we need to subtract together with the ELC correction. The first term rises from the interaction of the charges with the background, the last term from the self interaction of the background.

Literature

[1] I.C. Yeh and G. Hummer. Nucleic acid transport through carbon nanotube membranes. *Proc. Natl. Acad. Sci. U. S. A.*, 101(33):12177–12182, 2004.

[2] A. Arnold, J. de Joannis, and C. Holm. Electrostatics in Periodic Slab Geometries I. *J. Chem. Phys.*, 117:2496–2502, 2002.

[3] A. Arnold, J. de Joannis, and C. Holm. Electrostatics in Periodic Slab Geometries II. *J. Chem. Phys.*, 117:2503–2512, 2002.

6 Maxwell Equations Molecular Dynamics (MEMD)

by F. Rühle

6.1 Idea and introduction

The solution of electrostatic interactions via an electrostatic potential always suffers from three major drawbacks:

- The potential is not really needed, only the derivative.
- The solution of the Poisson equation is global and not easy to parallelize.
- For periodic boundaries, clever tricks are needed.

All these problems can be avoided when applying electrodynamics instead of electrostatics. After all, the latter is but the limit of its dynamic equivalent with infinite speed of light. And with electrodynamics come a couple of very neat properties:

- The Maxwell Equations are intrinsically local.
- Therefore we don't need a global solution, just local updates.
- We get directly the electric field, no potential.
- Periodic boundaries come naturally in a local algorithm.

This seems worth a closer look. Unfortunately, solving electrodynamics with a realistic speed of light is very costly and much slower than Ewald-based electrostatics algorithms. But for the system of nuclei and surrounding electrons, Car and Parinello proved ([1]) that, to get correct results, it is not necessary to simulate electrons at their actual speed (which is several orders of magnitude higher than the nuclei's), but it suffices to make sure that they are just a little faster than the nuclei if you keep the system on a special constraint surface. This became famous as Car-Parinello Molecular Dynamics (CPMD).

The idea is easy and totally applicable in the case of MD-particles and the photon-like propagation of electrostatic interaction, where the waves move a lot faster than the charged particles. MEMD simply means finding a suitable constraint for the given system and simulating electrodynamics with a CPMD background.

6.2 The constraint

We start off with the most important of the Maxwell equations for our case, the Gauss law

$$\varepsilon_0 \nabla \cdot \mathbf{E}_0 = \rho \tag{6.1}$$

If we solve this equation once in the beginning and afterwards only apply updates that obey its time derivative, we should end up with the correct Gauss law at all times. When we use the charge conservation law $\dot{\rho} + \nabla \mathbf{j} = 0$, we get the constraint

$$\nabla \cdot \left(\varepsilon_0 \dot{\mathbf{E}} + \mathbf{j} \right) = 0$$

for the incremental update of our E-field. The simplistic solution for this constraint produces physically uncorrect results and we can see that the general solution of our incremental update field (when obeying $\nabla \mathbf{E}' = 0$) contains a transversal component we want to call $\mathbf{\Theta}$. This yields the most general form of constraint to obey the Gauss law:

$$\varepsilon_0 \dot{\mathbf{E}} + \mathbf{j} - \varepsilon_0 \nabla \times \dot{\mathbf{\Theta}} = 0 \tag{6.2}$$

6.3 Lagrangian treatment

Let us take a look at the phenomenological Lagrangian of our constrained system

$$\begin{aligned} L = & \sum_i \frac{m_i}{2} \mathbf{v}_i^2 - U \\ & + f_{\text{mass}} \frac{\varepsilon_0}{2} \int \dot{\mathbf{\Theta}}^2 d\mathbf{r} - \frac{\varepsilon_0}{2} \int \mathbf{E}^2 d\mathbf{r} \\ & + \int \mathbf{A} \left(\varepsilon_0 \dot{\mathbf{E}} - \varepsilon_0 \nabla \times \dot{\mathbf{\Theta}} + \mathbf{j} \right) d^3\mathbf{r} \end{aligned} \tag{6.3}$$

where we use the Lagrange multiplier $\mathbf{A}$ to impose the kinematic constraint. The prefactor f_{mass} simply denotes the mass of our photon-like particles, analog to the mass of electrons in CPMD. The equations of motion for this Lagrangian can be calculated as usual by variational calculus. Variation with respect to $\dot{\mathbf{r}}_i$ gives us

$$\begin{aligned} \frac{\partial L}{\partial \dot{r}_i^\alpha} &= m_i \dot{r}_i^\alpha + q_i A^\alpha(\mathbf{r}_i) \\ \frac{d}{dt} \frac{\partial L}{\partial \dot{r}_i^\alpha} &= m_i \ddot{r}_i^\alpha + q_i \dot{A}^\alpha(\mathbf{r}_i) + q_i \frac{\partial A^\alpha}{\partial r_i^\beta} \dot{r}_i^\beta \end{aligned}$$

where for the second line we just take the time derivative. Variation with respect to $\mathbf{r}_i$ yields

$$\frac{\partial L}{\partial r_i^\alpha} = -\frac{\partial U}{\partial r_i^\alpha} + q_i \dot{r}_i^\beta \frac{\partial A^\beta}{\partial r_i^\alpha}$$

Combining these two results and introducing the vector

$$\mathbf{B} = \nabla \times \mathbf{A} \tag{6.4}$$

we get the equations of motion for the particle

$$\begin{aligned} m_i \ddot{r}_i^\alpha &= -\frac{\partial U}{\partial r_i^\alpha} - q_i \dot{A}^\alpha + q_i \dot{r}_i^\beta \left(\frac{\partial A^\beta}{\partial r_i^\alpha} - \frac{\partial A^\alpha}{\partial r_i^\beta} \right) \\ m_i \ddot{\mathbf{r}}_i &= -\frac{\partial U}{\partial \mathbf{r}_i} - q_i \dot{\mathbf{A}} + q_i \mathbf{v}_i \times \mathbf{B} \end{aligned} \tag{6.5}$$

which is what we expected. The equations of motion for the electromagnetic fields can be found by variating the Lagrangian density $\mathcal{L}$, which by definition satisfies $L = \int \mathcal{L} d^3\mathbf{r}$. Variation in $\dot{\mathbf{\Theta}}$ and in time gives us the equation of motion:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{\Theta}}} &= f_{\text{mass}}\varepsilon_0 \dot{\mathbf{\Theta}} - \varepsilon_0 \nabla \times \mathbf{A} = f_{\text{mass}}\varepsilon_0 \dot{\mathbf{\Theta}} - \varepsilon_0 \mathbf{B} \\
\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{\Theta}}} &= f_{\text{mass}}\varepsilon_0 \ddot{\mathbf{\Theta}} - \varepsilon_0 \dot{\mathbf{B}} = 0 \\
f_{\text{mass}}\ddot{\mathbf{\Theta}} &= \dot{\mathbf{B}} \qquad (6.6) \\
\frac{1}{c^2}\dot{\mathbf{\Theta}} &= \mathbf{B} \qquad (6.7)
\end{aligned}$$

where we used the natural initial condition $\dot{\mathbf{\Theta}}(t = 0) = 0$ in the last step and replaced f_{mass} with $1/c^2$, which turns out to be the speed of light. The next variation in $\mathbf{E}$ gives us

$$\dot{\mathbf{A}} = -\mathbf{E} \qquad (6.8)$$

which leads to the more commonly know expression for equation (6.5). With these two last results (6.7) and (6.8), we can obtain two more Maxwell equations by inserting them into the constraint equation (6.2), namely Ampère's and Faraday's law:

$$\begin{aligned}
\dot{\mathbf{E}} &= c^2 \nabla \times \mathbf{B} - \frac{1}{\varepsilon_0}\mathbf{j} \qquad (6.9) \\
\dot{\mathbf{B}} &= \nabla \times \dot{\mathbf{A}} = -\nabla \times \mathbf{E} \qquad (6.10)
\end{aligned}$$

This means that only by applying the constraint (6.2), we naturally reproduce the complete electromagnetic formalism. It should be noted that the equations (6.4) and (6.8) represent nothing but the so-called temporal or Weyl gauge in electromagnetism, in which the scalar potential ϕ is identically zero, and which turns out to be the most appropriate gauge for our purposes.

Since the Lagrangian we put up is constrained, we can not simply construct a Hamiltonian from it. This would be nice to simplify further proofs for the conservation of phase-space volume, energy and momentum. However, we can construct a Lagrangian that is not constrained and produces the exact same equations of motion. The following proofs and details will not be discussed here, but we want to note that the new Lagrangian looks like this:

$$\begin{aligned}
L = &\sum_i \frac{m_i}{2}\mathbf{v}_i^2 - U + \frac{\varepsilon}{2}\int \dot{\mathbf{A}}^2 d^3\mathbf{r} \\
&- \frac{\varepsilon_0 c^2}{2}\int (\nabla \times \mathbf{A})^2 d^3\mathbf{r} + \int \mathbf{A} \cdot \mathbf{j} d^3\mathbf{r}
\end{aligned} \qquad (6.11)$$

6.4 Discretization

Obviously, we can not do simulations with fields in continous spacetime. So a discretization in time and space is necessary. The first of these does not pose any big problems except for some proofs getting more nasty. For the latter discretization, we have to apply a lattice on our space that carries the fields. To get these fields, we also need to have electric current on the lattice and therefore need to interpolate our charges on the grid.

The definition of our operators (derivative and curl) is done canonically with finite differences (see Figure 6.1).

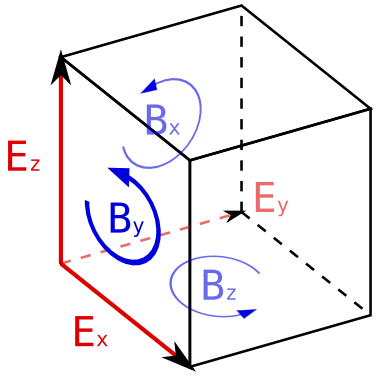


Figure 6.1: Positions of the variables and operators on the lattice. The E-field is placed on the links and its derivative in direction x is simply the finite difference to the next link. The B-field is placed on the plaquettes and is calculated by the rotation (finite differences) of the E-field.

6.5 How it actually works

So now, after all this math, how does the algorithm actually work? We already know that we need an initial solution for our E-field. Let's assume for now that we magically have that. After that, all we need are incremental updates that satisfy our constraint (i.e. the time derivative of the Gauss law, equation (6.2)).

So, the exact steps are:

- calculate the current $\mathbf{j}$ on the lattice.
- rotate $\mathbf{E}$ to get $\dot{\mathbf{B}}$ (eq. (6.10))
- propagate $\mathbf{B}$
- from the resulting $\mathbf{B}$, we get $\dot{\mathbf{\Theta}}$ via eq. (6.7)
- with $\mathbf{j}$ and $\dot{\mathbf{\Theta}}$, we can increment $\mathbf{E}$ (eq. (6.2))
- backinterpolate $\mathbf{E}$ on the particles and calculate the force

Most of these steps are obvious and straight forward. The third step (propagation of the B-field) needs to be thought through some more. The above Lagrangian (easiest to see via the equations (6.9), (6.4) and (6.8)) also delivers the equations of motion for our field degree of freedom $\mathbf{A}$:

$$\ddot{\mathbf{A}} = -c^2 \nabla \times \nabla \times \mathbf{A} + \frac{1}{\varepsilon_0}\mathbf{j} \tag{6.12}$$

This is the usual wave equation. It also leads the way how to update the vector field $\mathbf{A}$ on neighboring sites. The prefactor (light speed) is simply taken care of by the amount of local neighbor updates needed until we get our final distribution of $\mathbf{A}$, and therefore $\mathbf{B}$, and therefore $\mathbf{E}$.

It should be mentioned that there is another way besides the wave-like propagation we do here to propagate the B-field within the system. The original algorithm by Maggs diffuses the B-field through the system with Monte Carlo moves that are constructed in such a way that they obey the given constraint. However, the wave-like propagation is better suited for parallelization and is also capable

to work with locally varying dielectric constants, for which the Monte Carlo moves would need to be dynamically adapted.

In the ESPResSo simulation package, the second part of the Lorentz force $\mathbf{v} \times \mathbf{B}$ is ommited, since this greatly increases the speed of the algorithm. This means however, that the Hamiltonian structure of the system is destroyed and the momentum conservation does not hold anymore. Graphically, one can imagine that the momentum of the photons that are still travelling at the end of the propagation, is neglected. This is not so bad, since the contribution is of very small value ($\mathcal{O}(c^{-2})$ and the gain in speed outweighs the benefit of momentum conservation, especially when a thermostat is applied anyways. Energy is still conserved.

6.6 self-energy artefacts

The calculation of current on the lattice is also unproblematically derived from the movement of the particle and the interpolation on the lattice. However, there is a problem with the algorithm: Even in the continuum, the solution of the Maxwell equations for point charges is singular at the point where the charge is. The point charge carries along with it the electrostatic energy

$$\frac{1}{2}\int_{|\mathbf{r}-\mathbf{r}_i(t)|\leq R} \mathbf{E}(\mathbf{r},t)^2 d^3\mathbf{r} \propto \int_0^R r^2 (r^{-2})^2 dr = \int_0^R r^{-2} dr = \infty$$

This would mean that the particle has an infinite mass and can not respond to forces. With our lattice spacing, we introduce a "cut-off" for this self-interaction, but still the particle is driven to the center of the cell by the field created from its own (interpolated) charge. It is, from an energy point of view, energetic the most favorable for the particle to distribute its charge evenly on all surrounding lattice points, since it then produces the smallest possible curl in the cell and therefore no B-field.

This self-influence can be corrected either by calculating the potential energy of the interpolated charges and it's influence on the particle, or by directly substracting the exact Greens function for the problem. However, both methods are only applicable if the dielectric properties are constant within the lattice cell, otherwise we need a local, and therefore not potential-based, approach.

6.7 Initial solution

As mentioned in the beginning, we still need a way to solve the distribution of the E-field on the lattice to obey the Gauss law exactly in the beginning. Only then can we apply the incremental updates and stay on the constraint surface.

Of course, it is possible to use an Ewald based method for this, but the possibility of dealing with different dielectric properties within the system is then lost. So we apply a very simple and slow recursive scheme that is graphically described in Figure 6.2.

Let us check the Gauss law for the resulting field of this scheme. Per construction, at each vertex of the lattice we have:

$$\begin{aligned} E_z^2 - E_z^1 &= \frac{q_{\text{plane}}}{\varepsilon a^2} \\ E_y^2 - E_y^1 &= \frac{q_{\text{line}}}{\varepsilon a^2} \\ E_x^2 - E_x^1 &= \frac{q_{\text{vertex}}}{\varepsilon a^2} \end{aligned}$$

Since the total charge on the vertex is given by

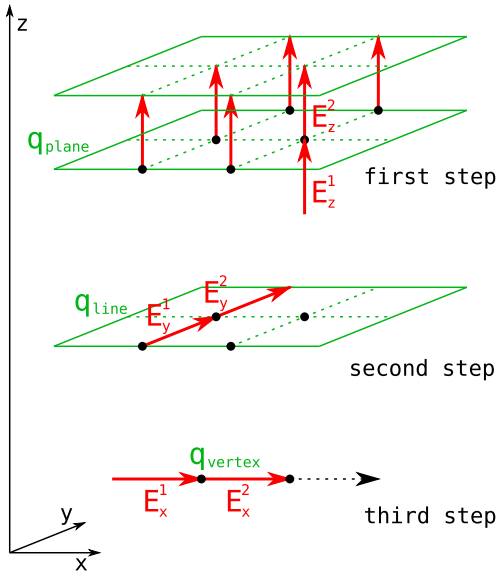


Figure 6.2: Recursive scheme for the initial solution of the E-field. The average charge in z-plane is added to the charge and field at each node, following $E_z^{n+1} = E_z^n + \frac{q_{\text{plane}}}{\varepsilon a^2}$. Then we again substract the charge q_{plane} from each charge in the z-plane and the average field of the system in z-direction from each node. Proceed exactly the same with y-lines and in x-direction. At the end, place the original charge on each vertex.

$$\begin{aligned} q_{\text{final}} &= q_{\text{plane}} + q_{\text{line}} + q_{\text{vertex}} \\ &\dots \\ &= \nabla \varepsilon_{(1\to 2)} E^{(1\to 2)} \end{aligned}$$

these equations yield the Gauss law directly.

Literature

[1] R. Car and M. Parrinello. Unified Approach For Molecular Dynamics and Density Functional Theory *Phys. Rev. Lett.*, 55:2471–2474, 1985.

[2] Igor Pasichnyk and Burkhard Dünweg. Coulomb interactions via local dynamics: A molecular-dynamics algorithm *J. Phys.: Condens. Matter*, 16(38): 3999–4020, 2004

7 Tree methods and the fast multipole method

The key to the improved scaling of the Ewald and MMM methods are Fourier transforms, followed by a product decomposition. Multipole methods are based on a product decomposition in real space. The advantage of this is, of course, that one can avoid the time-consuming Fourier transforms. The disadvantage is, that the real space decomposition can be done only approximately, via multipole expansions, and is much more difficult than the k-space decomposition. Moreover, the transformation into k-space inherently include periodic boundary conditions. Including periodic boundary conditions in multipole methods is possible, but not trivial, and will not be discussed here.

Let $r_i = (|r_i|, \theta_i, \phi_i)$ and $r_j = (|r_j|, \theta_j, \phi_j)$ be the spherical coordinates of two points, where r_j is closer to the origin than r_i, that is, $|r_j| < |r_i|$. Let $\gamma = \angle(r_i, r_j)$ be the angle between the origin and r_i and r_j. Then

$$|r_i - r_j|^2 = |r_i|^2 + |r_j|^2 - 2|r_i||r_j| \cos\gamma, \tag{7.1}$$

where

$$\cos\gamma = \cos\theta_i \cos\theta_j + \sin\theta_i \sin\theta_j \cos(\phi_i - \phi_j). \tag{7.2}$$

Then

$$\frac{1}{|r_i - r_j|} = \frac{1}{|r_i|\sqrt{1 - 2u\cos\gamma + u^2}}, \tag{7.3}$$

where $u = |r_j|/|r_i| < 1$. We expand the square root in powers of u and obtain

$$\frac{1}{\sqrt{1 - 2u\cos\gamma + u^2}} = \sum_{n=0}^{\infty} P_n(\cos\gamma) u^n, \tag{7.4}$$

where the coefficients are the Legendre polynomials, given for example by

$$P_n(x) = \frac{1}{2^n\, n!} \frac{d^n}{dx^n} \left[(x^2 - 1)^n\right]. \tag{7.5}$$

Since $u^n = |r_j|^n |r_i|^{-n}$, this is already a product decomposition in the distances. However, we need to find a product decomposition also of the angular part. For this, we use the generalized addition theorem

$$P_n(\cos\gamma) = \sum_{m=-n}^{n} Y_n^{-m}(\theta_j, \phi_j) Y_n^m(\theta_i, \phi_i),$$

where Y_n^m are the surface harmonics of the first kind, given for example by

$$Y_n^m(\theta, \phi) = \sqrt{\frac{(n - |m|)!}{(n + |m|)!}}\, P_n^{|m|}(\cos\theta) e^{\mathrm{i}m\phi}, \tag{7.6}$$

where

$$P_n^m(x) = (-1)^m (1 - x^2)^{m/2} \frac{d^m}{dx^m} P_n(x). \tag{7.7}$$

Note that

$$P_n^m(\cos\theta) = (-1)^m \sin^m(\theta) \left(\frac{d}{d\cos\theta}\right)^m P_n(\cos\theta) \tag{7.8}$$

is a polynomial of degree m in $\sin\theta$ and degree $n - m$ in $\cos\theta$; only odd or even powers of $\cos\theta$ appear. Therefore, once the polynomial coefficients are determined, the spherical harmonics are rather easy to calculate numerically.

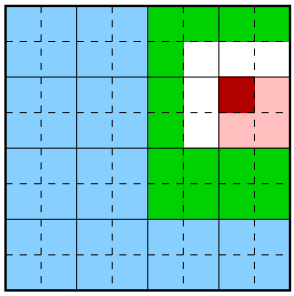


Figure 7.1: The tree method. Multipole expansions are first created for each coarsest level subbox, and used in coarsest level subboxes that are at least one subbox distant. For the light red box, this are for example the light blue boxes. Then, expansions on the next finer level are calculated, but only used for the subboxes that have not been taken into account on the coarser level. For the dark red subbox that are for example the green subboxes.

The *multipole expansion* of $1/|r_i - r_j|$ is then

$$\frac{1}{|r_i - r_j|} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left(Y_n^{-m}(\theta_j, \phi_j) |r_j|^n \right) \left(Y_n^m(\theta_i, \phi_i) |r_i|^{-n-1} \right). \tag{7.9}$$

Clearly, this expansion converges well, as long as $|r_j| \ll |r_i|$. Moreover, it is easy to give an upper bound for the pairwise error:

$$\left| \frac{1}{|r_i - r_j|} - \frac{1}{|r_i|} \sum_{n=0}^{p} P_n(\cos\gamma) \frac{|r_j|^n}{|r_i|} \right| \leq \frac{1}{|r_i| - |r_j|} \left(\frac{|r_j|}{|r_i|} \right)^{p+1}. \tag{7.10}$$

For a group of charges, we can now combine their contributions to the multipole expansion. Assume charges $|r_i| < a$ for $i = 1, \ldots N$. Then the potential at position $r = (|r|, \theta, \phi)$ with $|r| > a$ is

$$\sum_{i=1}^{N} \frac{q_i}{|r_i - r|} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} M_n^m Y_n^m(\theta, \phi) |r|^{-n-1}, \tag{7.11}$$

where the moments of the charge group are given by

$$M_n^m = \sum_{i=1}^{N} q_i Y_n^{-m}(\theta_i, \phi_i) |r_i|^n. \tag{7.12}$$

If one could choose the charge group as the set of all charges, one could therefore calculate the total energy in linear time, by first calculating the moments, and then evaluating the potential at each particles position. However, this is prohibited by the requirement $|r_j| \ll |r_i|$, which imposes two requirements: (i) the origin needs to be as close as possible to the charge group generating the moments, and (ii) we should not use the multipole expansion for particles that are close to each other. This problem can be solved similar to the MMM family of algorithms — by dividing the space into cells. For adjacent cells, one uses direct summation, for cells that are further away, multipole expansions.

7.1 The tree method

The simplest way to exploit this expansion, is by hierarchically subdividing the simulation cell, see Fig. 7.1. At each level, we can calculate the interactions between subboxes that are at least one subbox separated, using multipole expansions of apropriate order. The minimal subdivision is 4 boxes per side, on coarser levels, all boxes are neighbors.

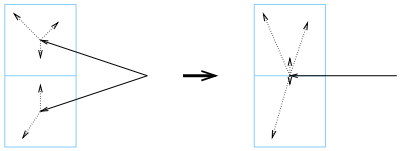
If we center the multipole expansion at the center of the subbox of edge length $a = L/4$, then for all particles in the subbox $|r_j| \leq \sqrt{3}/2a$. Particles in a non-neighbor subbox are $|r_i| \geq 3/2a$ away from from the center, therefore the error bound Eqn. (7.10) tells us that we have to use $p = -log_{\sqrt{3}}(\epsilon)$ terms in the multipole expansion to achieve precision p. Now, we refine our subboxes, by dividing each of them up into 8 smaller subboxes. Now, we use these expansions again only for boxes that are at least one subbox away. However, we also don't use them for subboxes that have already been taken into account on the next coarser level. After $\log_8 N$ subdivision steps, there is on average only one particle left per box, and further ubdivisions don't make sense. Instead, we calculate the remaining interactions explicitely by direct summation.

For each level, we need to once loop through all particles to calculate $\mathcal{O}(p^2)$ multipole moments in each of the subboxes. After that, we loop again over all particles to make use of the multipole expansion, in each of the up to 189 subboxes that are not yet taken into account. Since we chose the last level such that on average there is one particle per subbox, the computation for it is also linear. In total, we have a computing time of $\mathcal{O}(Np^2 \log N) = \mathcal{O}(N \log N \log \epsilon)$.

7.2 Fast multipole method

However, due to the rather large number of interacting subboxes at each subdivision, the prefactors in the tree method are large, which makes it rather slow. The crossover with a direct $\mathcal{O}(N^2)$-computation occurs only at about 100,000 particles! Therefore, a further improvement is required, the fast multipole method.

For this, we need three more ingredients. First, we need to be able to combine the multipole expansions of distant cells. That is not a problem, since multipole expansions are additive, however, we need to *shift* them to a common origin:



This can be accomplished by linear combinations of the moments. Assume a multipole expansion around a point $q = (|q|, \alpha, \beta)$:

$$\Phi(r) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} M_n^m Y_n^m(\theta', \phi') |r - q|^{-n-1}, \tag{7.13}$$

where $r - q = (|r - q|, \theta', \phi')$ in spherical coordinates, and all generating charges are located within a sphere of radius a around q. If $|r| > |q| + a$, then

$$\Phi(r) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} N_n^m Y_n^m(\theta, \phi) |r|^{-n-1}, \tag{7.14}$$

where

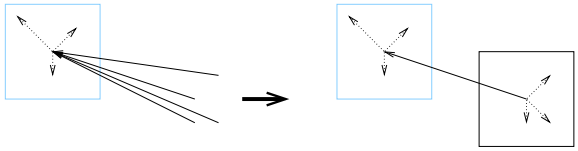
$$N_j^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} \mathrm{i}^{|k|-|m|-|k-m|} \frac{A_n^m A_{j-n}^{k-m}}{A_j^k} |q|^n \, Y_n^{-m}(\alpha, \beta) \, M_{j-n}^{k-m} \tag{7.15}$$

and

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!\,(n+m)!}} \tag{7.16}$$

In other words, we have shifted the multipole expansion from the point q to the origin.

The second ingredient is the possibility to *exchange* the role of the generating charge group and the point where the expansion is evaluated:



If $|q| > 2a$, then we can obtain an expansion for $|r| < a$ as

$$\Phi(r) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} L_n^m Y_n^m(\theta, \phi) |r|^n, \tag{7.17}$$

where

$$L_j^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} \mathrm{i}^{|k|-|m|-|k-m|} \frac{A_n^m A_j^k}{(-1)^n A_{j+n}^{m-k}} |q|^{-j-n-1} Y_{j+n}^{m-k}(\alpha, \beta) M_n^m. \tag{7.18}$$

The error of this series is of the order of $\mathcal{O}[(|q| - a)^{-p-1}]$.

Finally, one needs to be able to *translate* local expansions. Let

$$\Phi(r) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} L_n^m Y_n^m(\theta', \phi') |r - q|^n \tag{7.19}$$

be a local expansion expansion around $q = (|q|, \alpha, \beta)$. Then

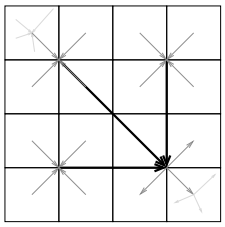
$$\Phi(r) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} K_n^m Y_n^m(\theta, \phi) |r|^n, \tag{7.20}$$

where

$$L_j^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} \mathrm{i}^{|m|-|m-k|-|k|} \frac{A_{n-j}^{m-k} A_j^k}{(-1)^{n+j} A_n^m} |q|^{n-j} Y_{n-j}^{m-k}(\alpha, \beta) L_n^m. \tag{7.21}$$

That is, we have shifted the local expansion from the point q to the origin.

Now let us assume that the number of cells per simulation box side is a power of two, $S = 2^L$, where L is the number of subdivision *levels*. Always 8 of the cells of a level l are combined to form a division of the simulation box into 2^{l-1} larger cells per side, and so on. We can combine the multipole expansions of eight neighbouring of level l cells through translations into one multipole expansions for the cell of level $l - 1$ formed by the eight original cells. This procedure can be continued again to obtain multipole expansions for all cells on each level. Once we have arrived at the bottom level $l = 1$ we convert the expansion to local expansions in all three other top level cells and distribute them up to the higher levels again until for all cells the multipole expansion of the full system is available. The multipole expansion for the top level are used to calculate the electrostatic interaction:



The dotted lines represent the calculation of the multipole expansion and the calculation of the energy rsp. the forces, the solid lines translations and conversions. Here only the data flow from one

cell to another is shown, in a real simulation this flow occurs for all pairs of cells, and of course the number of cells is much larger. In pseudo code the energy rsp. force calculation looks like this:

```
for each particle i
    add contribution of particle p to the
    local multipole expansion of its cell
end
for each l = L ... 2
    for each cell c of level l
        add multipole expansion to expansion of the
        level l − 1 supercell, translated to its center
end
for each cell c of level 1
    for each cell c' of level 1
        add converted contribution of cell c to the
        local expansion for cell c'
for each l = 2 ... L
    for each cell c of level l
        calculate local multipole expansion from the
        translated expansions of same level neighbors
        and of the level l − 1 supercell
end
```

The algorithm as presented here is called the fast multipole method. The number of terms needed in the multipole expansion only depends on the precession requirement and the number of particles in a cell. At constant density and constant number of particles per cell the number of operations therefore only depends on the loops shown above, which are all either of order N or $S^3(1 + 1/8 + 1/64 + \ldots) = \mathcal{O}(N)$, such that the overall computational order is $\mathcal{O}(N)$. One drawback of the method is that all the intermediate multipole expansions have to be stored, since they are needed in the last loop. This can use a considerable amount of memory, since the number of terms in the multipole expansion can be large.

Literature

[1] L. Greengard and V. Rhoklin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325, 1987.

[2] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997.