A super cool scientific Title ...

Gautier VAILLANT

August 18, 2015

Acknoledgements

I would first like to gratefully thank Helmut Grubmüller and Bert De Groot for welcoming me in their lab. I also thank Carsten Kutzner who supervised my Internship. I also would like to Thank Bartosz Kohnke and Thomas Ullmann for providing me advice on my project.

Finally I would also like to thank Ivo Kabadschow and Andreas Beckmann who welcomed me in the Jülich Forschungszentrum to teach me the details of the FMM method.

Abstract

Simulating large pairwise interactions is a very important issue for Scientific research. It plays an important role in Astrophysics to know the dynamics of galaxies, in plasma physics or in our case in biophysics. This kind of simulations is typically with a complexity of $\mathcal{O}(N^2)$ which scales badly with the size of the system.

Some other techniques, such as the PME (Particle Mesh Ewald) and the FMM (Fast Multipole Method) are able to obtain a complexity of respectively $\mathcal{O}(N \log(N))$ qnd $\mathcal{O}(N)$. These techniques also allows a greter scalability of the system for parallel computations.

La simulation de larges systemes de particules en interaction est tres importante pour le calcul scientifique. Elles jouent un role important en Astrophysique pour connaitre la dynamique des galaxies, en physique des plasmas ainsi que, dans notre cas, en Biophysique pour l'etude de machines moleculaires complexes.

Contents

Contents

Chapter 1

Presentation of the Lab

Max Planck Institute for Biophysical Chemistry

The Max-Planck Institute for Biophysical Chemistry is part of the Max-Planck Institues, which are an ensemble of research center throughout Germany (As the CNRS in France could be).

The Max-Planck Institute for Biophysical Chemistry, located in Göttingen, is one of the biggest research centers in the Biophysics field in Europe. Its research fields includes Molecular Dynamics Simulation, but also NMR or Biochemistry.
The departments are : (from http://www.mpibpc.mpg.de/groups)

- Patrick Cramer - Molecular Biology
- Gregor Eichele – Genes and Behavior
- Dirk Görlich – Cellular Logistics
- Christian Griesinger – NMR-based Structural Biology
- **Helmut Grubmüller – Theoretical and Computational Biophysics**
- Stefan W. Hell – NanoBiophotonics
- Herbert Jäckle – Molecular Developmental Biology
- Reinhard Jahn – Neurobiology
- Reinhard Lührmann – Cellular Biochemistry

- Marina V. Rodnina – Physical Biochemistry
- Melina Schuh – Meiosis
- Alec M. Wodtke – Dynamics at Surfaces

Department of Theoretical and Computational Biophysics

This deparments led by both Helmut grubmüller and Bert de Groot aims at understanding of the physics and function of proteins, protein complexes, and other biomolecular structures at the atomic level. For this purpose, complex computer simulations of the atomistic dynamics are carried out.

There is basically two main parts in the lab, one part is on really studying systems (Proteins, membranes) and molecular machines in order to know their dynamics (for example some people are studying the molecular mechanisms of the ribosome), and another part which is more focussing on the methods, regardless of the biological system:

Concerning the part of the department dedicated to methods and theory, new statistical mechanics concepts, quantum hybrid methods, and efficient parallel simulation algorithms and codes are the methodological focus of the department.

Both lines of questions phrased above are closely interlinked. Progress in the understanding of the physics of proteins, on the one hand, enables improved and more realistic simulation techniques, which allow to study a growing number of biochemical processes in great detail. Through analysis of well-understood mechanisms, on the other hand, one can learn to separate relevant aspects in protein dynamics from irrelevant ones — which is prerequisite for the construction of effective protein models. In short, we find a close interplay between theory development, algorithmic progress, and application.

Chapter 2

Context of the Internship

The Context of this Internship is driven by the "SPPEXA (Software for exascale computing) / GromEx" project funded by the DFG (Deutsche Forschungsgeimeinschaft). The Idea of this project is to create a flexible and fast solver for computing forces and potentials, which is a preliminary for molecular simulations.

A poster[1] of the project can be found below :

[1] from http://www.mpibpc.mpg.de/grubmueller/sppexa

Figure 2.1: Poster for the SPEXXA project

Currently the method used for computing electrostatic forces is called the PME (Particle Mesh Ewald). It works nicely but one of its problems is that the algorithms cannot be efficiently parallelised tasks as there is a lot of communication between the CPU or GPU cores. The idea would be to replace this method with a new method called the Fast-Multipole Method which is based on a tree Structure and may allow an greater parallelization of the system as wanted.

So the Idea of the Internship is first to know how to tune those systems to know how they relate to each other. Which parameters do I need to take for the FMM in order to have the same accurarcy as the PME ?

Chapter 3

Methods for computing electrostatic forces

The electrostatic potential of a group of N of charge q_i at position r_i is defined as :

$$V_C = \frac{1}{4\pi\epsilon_0} \sum_{i=1}^{N} \sum_{j>i} \frac{q_i q_j}{r_{ij}} \tag{3.1}$$

Computing this potential is very important in different areas of physics, such as plasma physics, or in our case molecular dynamics where the forces between charged particles need to be computed in an efficient and scalable way, the being computed using

$$\vec{F_i} = -\vec{\nabla}_i V(\vec{r}) \tag{3.2}$$

3.1 Direct summation method

In this section, we will explain the most basic method to compute pairwise interactions and explain why the method leads to long computation times and sometimes .

3.1.1 Naive $\mathcal{O}(N^2)$ Method

The sum of all electrostatic forces $\vec{F_i}$ felt by once particle can be written the following way:

$$\vec{F_i} = \sum_{i=1}^{N} \sum_{j>i} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij^2}} \tag{3.3}$$

where q is the charge of one particle, and R_{ij} is the distance between particle i and particle j.

In the thesis we will simplify the equation (3.3) by just writing:

$$\vec{F_i} = f \cdot \sum_{i=1}^{N} \sum_{j>i} \frac{q_i q_j}{r_{ij^2}} \tag{3.4}$$

where f is called the **electric conversion factor** and is equal to 138.935485(9) kJ.mol^{-}1.nm.e^{-}2

The first, naive way to compute electrostatic forces is just to follow equation (3.4):

So if we consider a set of N charged particles, $N - 1$ interactions are needed to compute the force acting on one specific particle. So in order to know the forces of the set of particles, $N \cdot (N - 1)$ operations are needed, hence an algorithmic complexity of $\mathcal{O}(N^2)$.

This gives the following algorithm:

```
input  : A set of N charged particles
output: A list of the forces for each particle

For each particle i;
for i ← 1 to N − 1 do
    add interaction between particle i and particle j ;
    for j ← i + 1 to N do
        force [i] ← force [i] + computeForce(i, j) ;
    end
end
```

Algorithm 1: Naive method

The complexity of such a computation limits its use to rather small systems and is not really usable for bigger systems such as proteins or astrophysical systems. Moreover it is also important to say that if we want to add periodic boundary conditions the $\mathcal{O}(N^2)$ method is not sufficient as the system is infinite.

3.1.2 Possible improvements

A possible method to overcome this limitation for periodic systems is to limit the interaction to a certain radius : if the distance between two particles if greater than R_0, then the force is set to 0.

So we have the following system :

$$\overrightarrow{F}_{A\to B} = \begin{cases} \frac{q_A q_B \hat{r}_{AB}}{|R_{AB}|^2} & \text{if } R_{AB} < R_0 \\ \overrightarrow{0} & \text{otherwise} \end{cases} \tag{3.5}$$

This technique is for example used for Lennard-Jones potentials ($V_{LJ} = 4\epsilon[(\frac{\sigma}{r})^1 2 - (\frac{\sigma}{r})^6]$), where the intensity of the force is quickly decreasing. It allows to limit the number of interactions to only the close neighbors.

However, one of the problems of this cutoff technique, especially for long-range interactions such as coulombic interactions using a cut-off can lead to artefacts resulting from the sudden drop of the force to 0 at the cutoff. It was shown that this can lead to unphysical assemblies of particles at the cutoff distance as shown for instance in figure (3.1) .

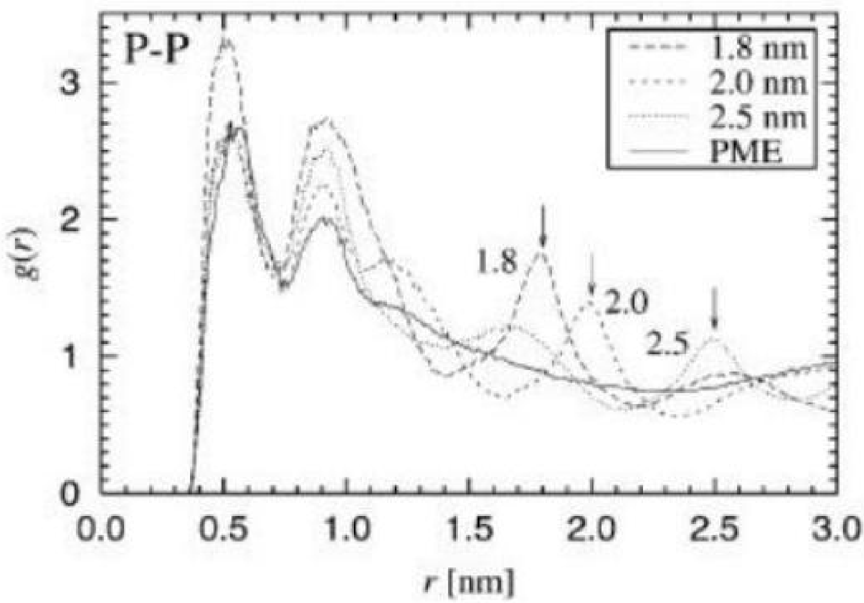


Figure 3.1: Radial distribution function (RDF) $g(r)$ between the two central atoms in the head-group of lipids: Cutoff distances are indicated by arrows. from [ADD REFERENCE]

As we can see in figure 3.1, the radial distribution of the distance between two atom shows a peak, corresponding to the cutoff of the system. This shows that by using a cutoff technique we might see some artifacts.

So we can see at least three reasons not to use the direct summation method : The first and main reason is that the method doesn't work with periodic boundary conditions. Then, is cutoff methods are used to solve the problem, artifacts might arise at cutoff distances. Finally the method is computationally inefficient.

3.2 Fourier Transform-Based methods

To compute the potentials and the forces of particles, Fourier-transform based techniques have been developed in order to overcome the problems stated above .

3.2.1 Ewald Summation

This subset of techniques comes from a theoretical physics technique called the Ewald summation.

in periodic boundary conditions, the Coulomb potential V_C is:

$$V = \sum_{n_x,n_y,n_z} \sum_{i}^{N} \sum_{j>i} \frac{q_i q_j}{r_{ij}} \tag{3.6}$$

where n_x, n_y, n_z are the box index vector.

The equation (3.6) is conditionally convergent and slow to converge. One technique discovered by Ewald is to split the potential in two absolutely convergent terms and one constant term:

If the system is neutral, ie. $\sum_{i=1}^{N} q_i = 0$, the $\frac{1}{r}$ can be written in the following way :

$$\frac{1}{r} = \frac{f(x)}{r} + \frac{1 - f(x)}{r} \tag{3.7}$$

The idea is to choose f, so that $\frac{f(x)}{r}$ is quickly decaying and can be computed in real space with a cutoff at high accuracy. At the same time, we want $\frac{1-f(x)}{r}$ is as smooth as possible and therefore be represented in reciprocal sapce with just a few $\vec{k}$ vectors This gives a quick and accurate computation of the real and the reciprocal space where $V = V_{\text{direct}} + V_{\text{reciprocal}}$

A good choice is often $f(x) = \text{erfc}(\alpha x)$,the complementary error function, where α is called the splitting parameter, and $\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{+\infty} e^{-t^2} \text{dt}$

so we obtain the following potentials :

For the direct space :

$$V_{\text{direct}} = \frac{1}{2} \sum_{\vec{n}} \sum_{i,j}^{N} q_i q_j \frac{\operatorname{erfc}(\alpha r)}{r} \tag{3.8}$$

and for the reciprocal space :

$$V_{\text{reciprocal}} = \frac{1}{2\pi L} \sum_{i,j}^{N} q_i q_j \sum_{\vec{m}} \frac{\exp -(\pi \overrightarrow{m}/\alpha)^2 + 2\pi i \overrightarrow{m} \cdot (r_i - r_j)}{\overrightarrow{m}^2} \tag{3.9}$$

where $\vec{m}$ are the vectors in the reciprocal space, and L is the volume of the cell.

$$\begin{cases} \vec{m}_x = & 2\pi \frac{\vec{n_y} \times \vec{n_z}}{\vec{n_x} \cdot (\vec{n_y} \times \vec{n_z})} \\ \vec{m}_y = & 2\pi \frac{\vec{n_z} \times \vec{n_x}}{\vec{n_y} \cdot (\vec{n_z} \times \vec{n_x})} \\ \vec{m}_z = & 2\pi \frac{\vec{n_x} \times \vec{n_y}}{\vec{n_x} \cdot (\vec{n_z} \times \vec{n_y})} \end{cases}$$

Once the electrostatic potentials are obtained, it is possible to differentiate the potentials with respect to the position $p = (x, y, z)$ to obtain the force on each particle.

$$\overrightarrow{F}_i^{\text{direct}} = \overrightarrow{\nabla}_i V_{\text{direct}} \tag{3.10}$$

so we have:

$$\overrightarrow{F}_p^{\text{direct}} = q_i \sum_{i=1, i \neq j}^{N} \sum_{\vec{n}} q_j \frac{(r_{ij,\vec{n}})_p}{r_{ij,\vec{n}}^3} \{\operatorname{erfc}(\alpha(r_{ij,\vec{n}}) + \frac{2\alpha}{\sqrt{\pi}} r_{ij,\vec{n}} \exp(-(\alpha r_{ij,\vec{n}})^2)\} \tag{3.11}$$

$$\overrightarrow{F}_p^{\text{reciprocal}} = \frac{2q_i}{L} \sum_{i=1, i \neq j}^{N} \sum_{\vec{m} \neq 0} \frac{\vec{m}_p^*}{\vec{m}^{*2}} \exp\left(-\left(\frac{\pi \vec{m}}{\alpha L}\right)^2\right) \sin \frac{2\pi}{L} m \cdot r_{ij} \tag{3.12}$$

Equations 3.11 and 3.12 can then be used to compute the force on each particle by adding the direct space and the reciprocal space contribution.

3.2.2 PME

One way to improve this method is to compute the Fourier sum using the FFT (Fast Fourier Transform). This allows the algorithm to get a complexity of $\mathcal{O}(N \log N)$. There is different methods that are based on the Ewald summation, namely the **P3M** (Particle-Particle Particle-Mesh Method) or the **FFP** (Fast Fourier Poisson method).

In our case, we will focus on the **PME** as it is the method currently used in GROMACS.

We remind that the potential for the reciprocal space $V_{\text{reciprocal}}$ is written in equation (3.9):

$$V_{\text{reciprocal}} = \frac{1}{2\pi V} \sum_{i,j}^{N} q_i q_j \sum_{\mathbf{n}^*} \frac{\exp\left(-(\pi \overrightarrow{m}/\alpha)^2 + 2\pi i \overrightarrow{m} \cdot (r_i - r_j)\right)}{m^2}$$

This equation can be rewritten as :

$$V_{\text{reciprocal}} = \frac{1}{2\pi V} \sum_{\mathbf{n}^* \neq 0}^{N} \frac{\exp -(\pi \overrightarrow{m}/\alpha)^2}{\overrightarrow{m}^2} S(-\overrightarrow{m}) S(\overrightarrow{m})$$

where $S(\mathbf{n}^*)$ is defined as the Structure factor:

$$S(\mathbf{n}^*) = \sum_{k=1}^{N} q_k \exp(2\pi i \vec{m} \cdot \mathbf{r}) \tag{3.13}$$

The idea of the PME is to approximate the structure factor $S(\vec{m})$ by the 3D Fourier Transform of the charge matrix,which is obtained by interpolating the charges to a discrete grid of size $p \times p$. Let also define $\mathcal{F}(Q)$ the 3D FFT of Q. The charges q_i are mapped to the grid using interpolation.Originally, Lagrange interpolation was used, but now a b-spline interpolation is used. The order of interpolation is called the *PME Order*

then the Structure factor can be approximated as its FFT :

$$S(\overrightarrow{m}) \approx \widetilde{S}(\overrightarrow{m}) = \mathcal{F}(Q)(\overrightarrow{m}) \tag{3.14}$$

so the reciprocal energy can also be approximated by:

$$V_{\text{reciprocal}} \approx \widetilde{V}_{\text{reciprocal}} = \frac{1}{2\pi V} \sum_{\mathbf{n}^* \neq 0}^{N} \frac{\exp\left(-(\pi \overrightarrow{m}/\alpha)^2\right)}{m^2} \mathcal{F}(Q)(\mathbf{m}) \mathcal{F}(Q)(-\mathbf{m}) \tag{3.15}$$

It can be shown that the complexity of such a system is $\mathcal{O}(N \log(N))$ which is a much bigger improvement compared to the $\mathcal{O}(N^2)$ complexity of the direct algorithm. Then it also allows to handle infinite systems with periodic boundary conditions.
However, the algorithm doesn't scale well for large scale parallelism[REFERENCE ?], hence the need for a more scalable algorithm for the computations of electrostatic forces.

3.3 Tree-based methods

Now we will look at an algorithm that scales linearly with the number of charges N and which does not have inherent parallel communication bottleneck as [ref]. Furthermore, it can be applied to systems with periodic as well to systems with open boundaries. Since it approximates the actual charge distribution by a set of multipoles, it is called the "Fast Multipole method" (FMM)

3.3.1 Mathematical preliminaries

Let's move back to the potential created by a charged particle as in (**??**),

$$V = \frac{1}{d}$$

Let two particles $A(a, \alpha, \beta)$ and $R(r, \theta, \phi)$, separated by a distance d: The distance between the particles is $d = |a - r|$. We would like to achieve is to "factorize the addition", having a product of one function depending only on a and one depending only on r.

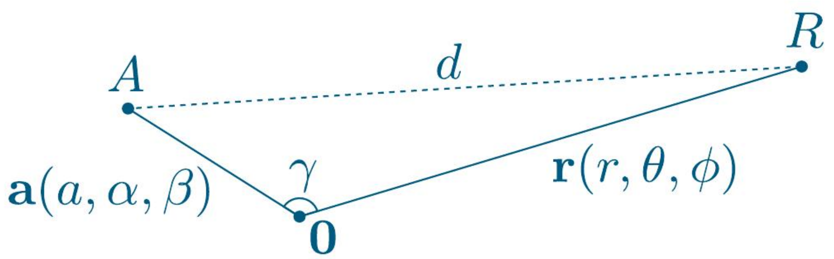


Figure 3.2: Two particles $A(a, \alpha, \beta)$ and $R(r, \theta, \phi)$, separated by a distance d

3.3. Tree-based methods

The inverse distance can be written as :

$$\frac{1}{d} = \frac{1}{|a-r|} = \frac{1}{\sqrt{a^2+r^2-2ar\cos(\gamma)}}$$

This distance can then written as a series of Legendre polynomials $P_l(u)$ of degree l if $r \gg a$:

$$\frac{1}{d} = \frac{1}{\sqrt{a^2+r^2-2ar\cos(\gamma)}} = \sum_{l=0}^{+\infty} P_l(u)\mu^l \tag{3.16}$$

where $\mu = \frac{a}{r}$ and $u = \cos\gamma$.
with some manipulation as explained [ADD REFERENCE], we obtain :

$$\frac{1}{|r-a|} = \sum_{l=0}^{+\infty}\sum_{m=-l}^{+l} \frac{(l-m)!}{(l+m)!}\frac{a^l}{r^{l+1}} P_{lm}\cos(\alpha) P_{lm}\cos(\theta) e^{-im(\beta-\phi)} \tag{3.17}$$

in order to approximate the scheme, we can truncate the infinite series to a certain order p, we call this order the **multipole order**

$$\frac{1}{|r-a|} \simeq \sum_{l=0}^{p}\sum_{m=-l}^{+l} \frac{(l-m)!}{(l+m)!}\frac{a^l}{r^{l+1}} P_{lm}\cos(\alpha) P_{lm}\cos(\theta) e^{-im(\beta-\phi)} \tag{3.18}$$

We can now rewrite the summation the following way :

$$\frac{1}{|r-a|} \simeq \sum_{l=0}^{p}\sum_{m=-l}^{+l} \underbrace{\frac{a^l}{(l+m)!} P_{lm}(\cos(\alpha)) e^{-im\beta}}_{O_{lm}(\mathbf{a})} \underbrace{\frac{(l-m)!}{r^{l+1}} P_{lm}(\cos(\theta)) e^{+im\phi}}_{M_{lm}(\mathbf{r})} \tag{3.19}$$

hence,

$$\frac{1}{|r-a|} \simeq \sum_{l=0}^{p}\sum_{m=-l}^{+l} O_{lm}(\mathbf{a}) M_{lm}(\mathbf{r}) \tag{3.20}$$

Thus it is possible to "factorize" the inverse of the the distance $\frac{1}{d}$ so we can obtain two independent terms $O_{lm}(\mathbf{a})$ and $M_{lm}(\mathbf{r})$ for two particles

We can then define the multipole moment $\omega_{lm}(q, \mathbf{a}) = qO_{lm}(\mathbf{a})$ and the Taylor-like moment $\mu_{lm}(q, \mathbf{r}) = qM_{lm}(\mathbf{r})$

One obtains the following "bipolar expansion" for two particles associated with two different origins :

$$\frac{1}{|\mathbf{a}_1 - \mathbf{a}_2 + \mathbf{R}|} = \sum_{l=0}^{+\infty} \sum_{j=0}^{+\infty} \sum_{m=-l}^{+l} \sum_{k=-j}^{+j} (-1)^j \cdot O_{lm}(\mathbf{a}_1) \cdot M_{l+j,m+k}(\mathbf{R}) \cdot O_{jk}(\mathbf{a}_2) \tag{3.21}$$

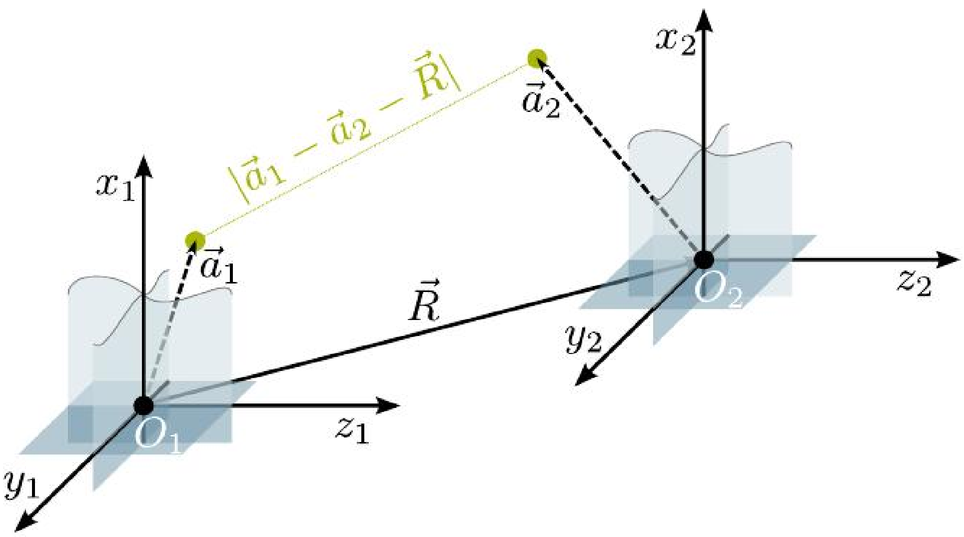


Figure 3.3: Scheme of two particles expanded according to two different origins [from ADD REFERENCE]

3.3.2 Workflow of the algorithm

Once some mathematical preliminaries are set, it is possible to explain the workflow of the algorithm.

Splitting the Space

One of the ideas of the FMM is to approximate far-away charges by multipoles expansions,whereas interactions between near charges are calculated directly without any approximation. So that's why we need to split the simulation box in sub-boxes, so that near field and far field contributions can be computed everywhere.

So first step of the scheme is to split the space in order to generate different groups. The idea is to recursively split the space in eight octants, created the so-called structure of an *octree*. So the number of boxes is 8^{D-1}, where D is the depth of the oct-tree.

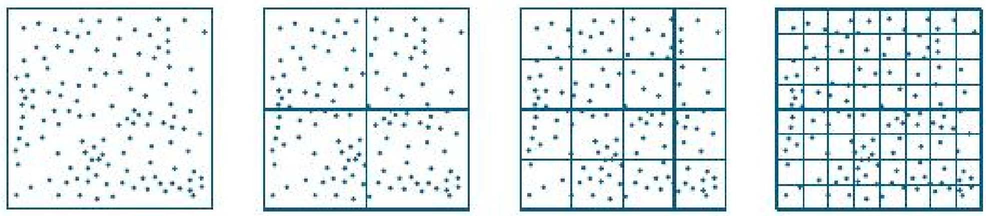


Figure 3.4: Example of a division of the space in boxes. The example is in 2D so the structure of the space is called a *quadtree*.

The number of subdivisions is called the *depth* D. For example, in figure 3.4, we can observe from left to right the depths $0, 1, 2$ and 3.

Then each particle is assigned to the box of lowest depth it is found in.

Defining a Separation Criterion

For the further computations it is needed to define a separation criterion ws: It is the ws^{th} next neighbors of a given depth. Two boxes A and B are called *next neighbors* is they are at the same level and box B is enclosed by a box of size $(2ws + 1)^3$ around the center of A

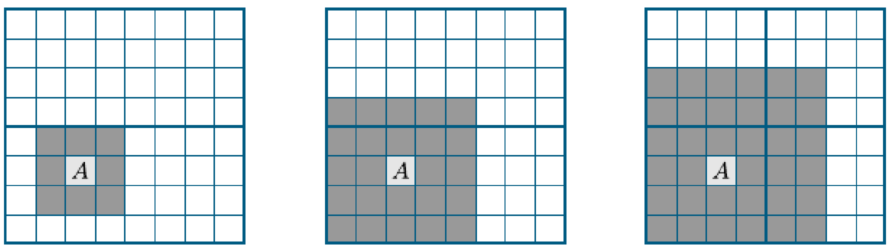


Figure 3.5: Example of the influence of the separation criterion ws for a given box A, from left to right, $ws = 1, 2, 3$.

This separation criterion divides the space into two parts. If two particles are close so they are in the same grey space as defined figure (3.5), then, their interaction is computed directly, otherwise their interaction is approximated via multipoles.

3.3. Tree-based methods

It is most of the time not possible to set the separation criterion ws to 0, as the multipole expansion might overlap. Using a bigger ws means that the order of the multipole expansions need to be less important for the same precision, however this will increase the number of direct $\mathcal{O}(N^2)$ operation, which are more computationally expensive.

The algorithm by itself will be done in separate phases we call "passes" : there will be 4 different passes:

PASS 1 : Computing the Multipole moments

PASS 2 : Transforming distant expansions into multipoles

PASS 3 : Shift Taylor-Like expansions down the tree

PASS 4 : Computing forces and Energies

PASS 1 : Computing the Multipole moments

Once every particle is assigned to its box at the lowest level, we will compute the multipole moment ω_{lm}^{j} of each box.

$$\omega_{lm}^{j}(q, \mathbf{a}) = q^j a_j^l \widetilde{P}_{lm}(\cos(\alpha_j))e^{-im\beta_j} \tag{3.22}$$

To construct the multipoles at the higher levels from the existing multipoles at the lowest level, each multipole order is moved to the Center of the parent box using the so-called $M2M$ (Multipole to Multipole) operator:

$$\omega(\mathbf{a} + \mathbf{b}) = \sum_{j=0}^{k} \sum_{k=-j}^{j} \omega_{jk}(\mathbf{a}) O_{l-j,m-k}(\mathbf{b}) \tag{3.23}$$

$O_{l-j,m-k}(\mathbf{b})$ is called the $M2M$ operator.

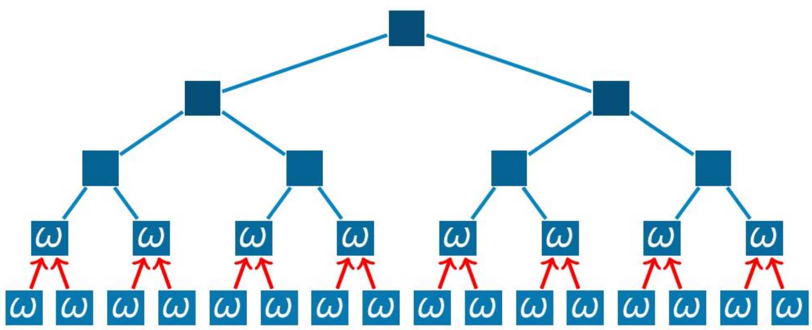


Figure 3.6: Example in 1-D showing the shifting of the multipole moments using the M2M operator.

PASS 2 : Transforming distant expansions into multipoles

PASS2 transforms distant expansions (beyond the separation criterion) ω_{lm} into a Taylor-like moment μ_{lm}. This transformation is done over at most 189 boxes. This transformation is applied for each depth of the tree.

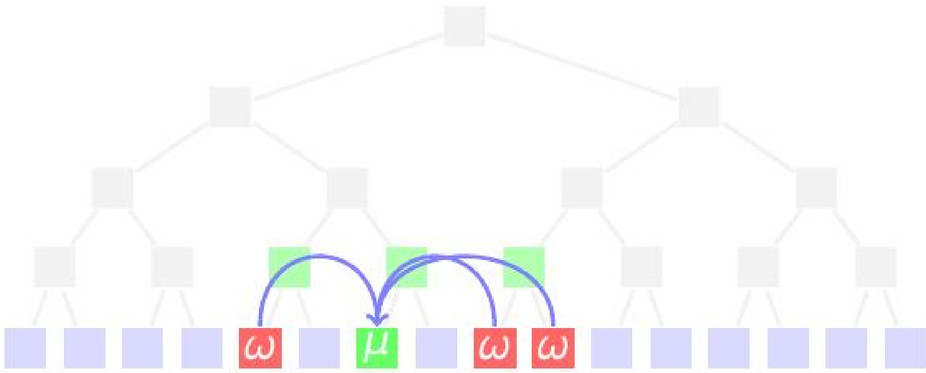
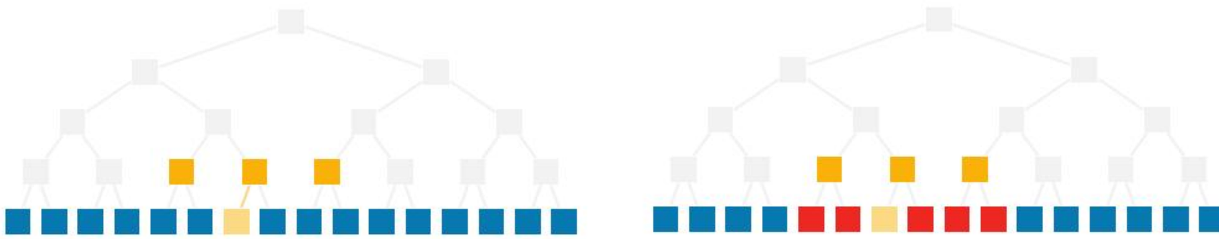


Figure 3.7: Example in 1-D showing the transformation of distant expansions into multipoles

In order to compute this transformation, two different rules have to be applied:

1. Take the non-separated parent boxes, as shown sub-figure (a) below.
2. Then the children of the parent box are selected, taking into account the separation criterion for the child boxes.



(a) First, the parents of the box are chosen

(b) Then, then children of the parents are chose
Then the separation criterion can apply

Figure 3.8: Interation Rules for the transformation of the multipole expansions

Mathematically, the transformation from multipole expansions to multipole moments is given using the M2L (Multipole to Local) Operator

$$\mu_{lm}(\mathbf{b}-\mathbf{a}) = \sum_{j=0}^{+\infty} \sum_{k=-j}^{j} M_{j+l,k+m}(\mathbf{b}) \cdot \omega_{jk}((\mathbf{a})) \tag{3.24}$$

where $M_{j+l,k+m}(\mathbf{b})$ is the M2L operator : It allows to exchange, as figure (**??**) shows, information between boxes of the same level.

PASS 3 : Shift Taylor-Like expansions down the tree

Once the local μ_{lm} expansions are computed, the expansion are shifted to the lowest level of the tree using the L2L (Local to Local) operator.

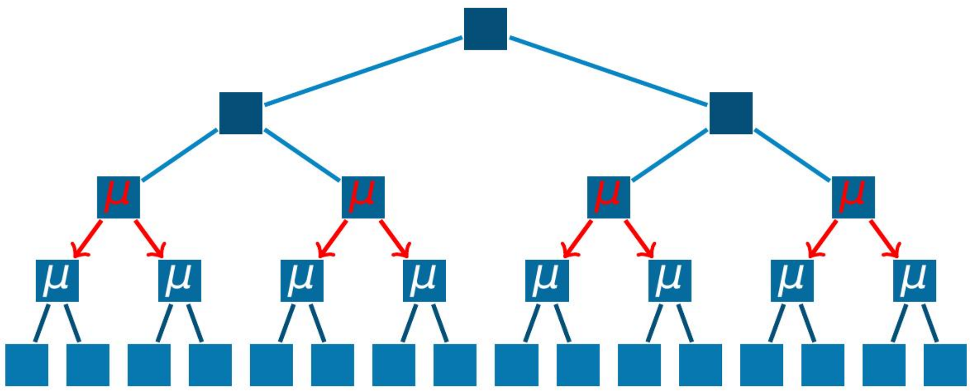


Figure 3.9: Example in 1-D showing the shifting of the local moments using the L2L operator.

This shift operation is done by the following formula :

$$\mu_{lm}(\mathbf{r}-\mathbf{b}) = \sum_{j=l}^{p} \sum_{k=-j}^{j} O_{j-l,k-m}(\mathbf{b}) \cdot \mu_{jk}((\mathbf{a})) \tag{3.25}$$

where $O_{j-l,k-m}(\mathbf{b})$ is the L2L operator : It allows to exchange, as figure (**??**) shows, information between levels downwards.

PASS 4 : Computing forces and Energies

After PASS 3, we have in every box at the lowest level both the multipole expansion as well as the local expansion, thus everything to compute the far-field contribution to the forces and the potential of the system.

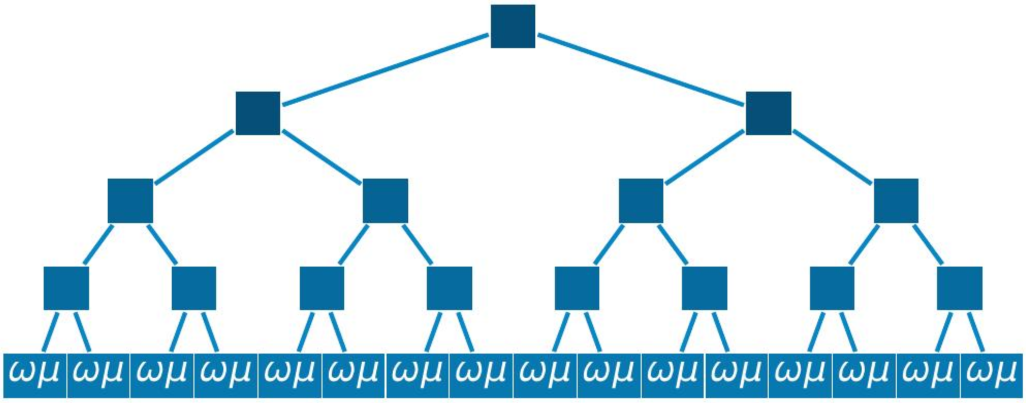


Figure 3.10: Situation after PASS 3

So we have for the electric field :

$$E_{FF} = \sum_{l=0}^{p} \sum_{m=-l}^{+l} \mu_{lm}(\mathbf{r}) \omega_{lm}(\mathbf{a}) \tag{3.26}$$

For the potential :

$$\Phi_{FF} = -\sum_{l=0}^{p} \sum_{m=-l}^{+l} \mu_{lm}(\mathbf{r}) \nabla_{\mathbf{a}_i} [a_i^l \widetilde{P}_{lm}(\cos \alpha_i) e^{-im\beta_i}] \tag{3.27}$$

and for the force :

$$\mathbf{F}_{FF} = -\sum_{l=0}^{p} \sum_{m=-l}^{+l} \mu_{lm}(\mathbf{r}) \nabla_{\mathbf{a}_i} [a_i^l \widetilde{P}_{lm}(\cos \alpha_i) e^{-im\beta_i}] \tag{3.28}$$

Then the near-field contribution can be easily computed :

Chapter 4

Comparing FMM and PME accuracy

In this chapter I will more precisely explain my work done in the Lab, which consisted in comparing the PME methods, which is for example used by gromacs, and the FMM method, used by a solver made between the Julich Forschungszentrum and the Max Planck Institute for Biophysics in Göttingen.

4.1 Presentation of GROMACS

GROMACS (GROningen MAchine for Chemical Simulations) is a simulation software used for Molecular dynamics. It was originally developed in the Biophysical Chemistry department of University of Groningen but it is now developed around the world. GROMACS is made to be as fast as possible using all possible techniques to improve its performance (MPI, GPU computing)

4.1.1 Structure of a File

The program is used by modifying text files that giving information on the structure of the system that has to be simulated and the parameters of the simulation. The files are the following

- The *.pdb file are the *Protein DataBank* file : In this file is denoted the position and the type of the particles of the system. It also gives the size of the simulation box

- The *.top files (namelt *topoloy* files) are the file where the properites of the atoms are defined. These properties are for example the charge, the Van-der-Waals parameters or the binding forces of the system. This configuration is often done with force field files such as AMBER or CHARMM
- The *.mdp files are defining the physical and computational parameters of the simulation. It is defining the methods used for the simulation. In our case we are mostly interested in the electrostatics part of the computation ; it is then possible to select the method (PME or Cut-Off methods) and the parameters for the PME

4.1.2 PME parameters

In this paragraph I will explain the parameters it is possible to play with:

CutOff The first parameter is the CutOff : It allows to set the Cut-Off radius for the Cut-Off method, but it also sets the difference between the direct part and the reciprocal part in the PME method

Fourier Spacing The other important parameter is the fourier spacing. I remind that in the PME method a 3D FFT is done in order to compute the energies and the forces of the system. The FFT is so computed on a grid : The dimension of the grid is given by the *Fourier Spacing* parameter.

PME Order The PME order gives the order of the interpolation. For instance, PMEorder= 4 corresponds to a cubic interpolation.

4.1.3 Command to launch a GROMACS simulation

The workflow to laumch a GROMACS Simulation is the following :

First generate a .tpr binary file conaining all the information about the simulation. The file is processed using the *grompp* (Gromacs Preprocessor). The command is :

```
grompp -f mdpfile
  -p topFile.top
  -c pdbFile.pdb
  -o tprFile.tpr
```

where the input files are the *.mdp , *.top and *.pdb (Respectively properties of the Simulation, of the atom, and the opsition of the At and contains arroundoms). The *.tpr file ris the output file.

It is then possible to run the simulation using the program mdrun, which as its name stands, runs the md simulation. A possible example is :

```
mdrun -s tprfile.tpr
      -ntomp = 4
      -ntmpi = 4
      -nsteps= 0
```

Where the .tpr File is now the input file, containing everything about the simulation, -nsteps is the number of time steps needed (In the case of this Intership, no time intergration is needed as just the forces are required). It is also possible to choose the number of CPU cores needed for the simulation (Using MPI and OpenMP).

4.2 Presentation of fmsolvr

The software used to compute the electrostatic forces with the FMM is not gromacs (at least not for the moment). It is develloped by both the Max-Planck Institut for Biophysical Chemistry and the Jülich Forschungszentrum . In the following chapters, we will call this code *fmsolvr*.

The codebase is mostly written in C++. There exists several git branches for the program, allowing different versions of the system. The First version is a sequential version, which is the "basic" version of the FMM code . There also exists a version which allows periodic boundary conditions : We will use this version a lot as we need to campare it to the PME, which, by construction, uses periodic boundary conditions.

The input file, is a *.hpp file containing 4 arrays of size N, where N is the number of atoms in the simulation : one charge array q , and three array x, y, z for the positions of the atoms.

4.2.1 FMM Parameters

The simulation can be launched using the following command :

```
DEPTH=$DEPTH MULTIPOLEORDER=$MULTIPOLEORDER WS=$WS
    OPENBOUNDARY=0 UNITBOX=$UNITBOX CENTER=c
    ./fmmtest $inputFile.qxyz $outputFile.dat
```

The environment variables are the following :

DEPTH The maximal subdivision of the space : if DEPTH=n the space will be divided in 8^n boxes.

MULTIPOLEORDER The Order of truncation in a the series, for instance :

$$\frac{1}{|r-a|} \simeq \sum_{l=0}^{p} \sum_{m=-l}^{+l} \frac{(l-m)!}{(l+m)!} \frac{a^l}{r^{l+1}} P_{lm} \cos(\alpha) P_{lm} \cos(\theta) e^{-im(\beta-\alpha)}$$

15:20 - 16:00 - Gautier VAILLANT

where the multipole order is the red p in the first summation.

WS This gives the separation criterion $ws \geq 1$, which separates the far-field computation from the near space.

OPENBOUDARY if OPENBOUNDARY = 0, periodic boundary conditions will be used, if OPENBOUNDARY=1, then openbondaries will be used.

UNITBOX Gives the size of the simulation box (in nm).

4.3 Making GROMACS and fmsolvr comparable

The first part, if we want to make the PME and the FMM comparable, is to use the same simulation for both programs, hence requiring coding some tools transforming a GROMACS-compatible file to fmsolvr-compatible one. Then, we saw that the results we obtained weren't good enough as one system was using some dipole correction, so we needed to implements the dipole correction to the FMM system.

4.3.1 File manipulation

In this section will be explained the file modifications needed in order to have the simulations comparable on both systems.

Making GROMACS only compute electrostatics forces

In this Internship we just want to compute the electrostatic forces on our system. However, what we obtain in the GROMACS output file is the sum of all forces on each atom. These forces may include forces such as for instance Lennard-Jones potentials. ($V_{LJ} = 4\epsilon[(\frac{\sigma}{r})^1 2 - (\frac{\sigma}{r})^6])$.

So the first thing to do is to modify the topology file (*.top), so there is no more Lennard Jones potential at all ; this snippet is added to the *.top file:

```
[ atomtypes ]
type atnum mass charge ptype sigma epsilon
CLA 17 35.450000 0.000 A 0.000000000000 0.00000
SOD 11 22.989770 0.000 A 0.000000000000 0.00000
```

In the expression of the potential, we set $\epsilon = 0$ and $\sigma = 0$, so it gives $V_{LJ} = 0$. So we have modified the files such that there is only the electrostatic interaction at play in the simulation.

Assure compatibility between *.qxyz and *.pdb files

The next thing to do is to have the same simluation system for both methods : So I needed to make a few scripts to transform a *.pdb file in a *.qxyz or a *.hpp file.

The scripts are written in python, the workflow is the following:
*.hpp $\xleftarrow{\texttt{qxyz2hpp.py}}$ *qxyz $\xrightarrow{\texttt{qxyz2gromacs.py}}$ *.pdb

So it is now possible to move the positions of the atoms one software to another software. (PME to FMM or FMM to PME)

Output positions and forces to a Text file for GROMACS

The last preliminary thing to do is to make the GROMACS Simulation print the positions and the forces in an easy and detailled way. We decided to modify the Gromacs code to add after the so-called `do_force(...)` function, which computes the forces for the system, a routine that prints the force as well as the positions in order to be easily used afterwards.

4.4 First Comparisions and Dipole Correction

4.4.1 Silicamelt Simulation

The first thing we did afterwards it to take a good precision for both methods and compare the forces.

The model we are using is called the "Silicamelt" system which is a system composed of SiO_2 atoms : It is a system compased of about 100000 atoms, the silicon atoms (Si) have a $+2.4e$ charge and the Oxygen has a charge of $-1.2e$, where e is the elementary charge $e \approx 1.6 \cdot 10^{-19}$C. The size of the simulation box is 124.120 nm.

4.4. First Comparisions and Dipole Correction

The parameters for the PME simulation are cutoff=1.2nm, PME order=12 and fourier spacing=0.015 nm.

The paramerters for the FMM are : Depth=4, Multipole Order=40, and WS(Separation Criterion) =4. The simulations is done with periodic boundaries

The relative error is computed the following way:

$$error_{\text{relative}} = \frac{F_{FMM} * F_{PME}}{F_{PME}} \tag{4.1}$$

Then the following histogram can be plotted:

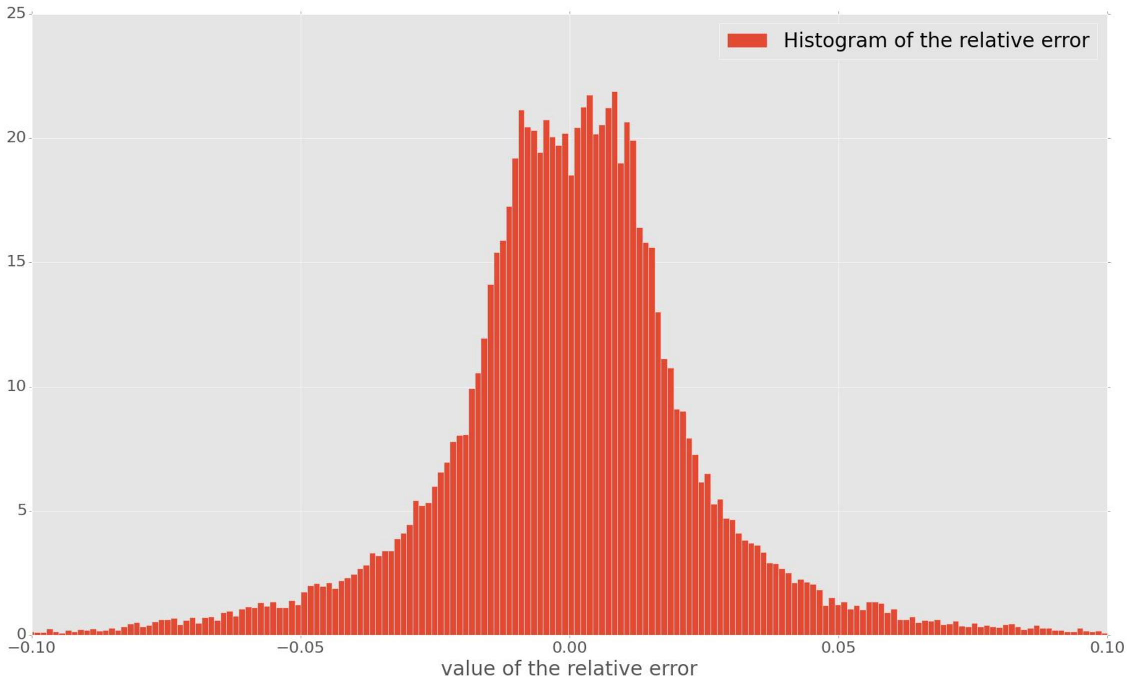


Figure 4.1: Histogram representing the relative error for a silicamelt simulation : The histogram is computed before any dipole correction.

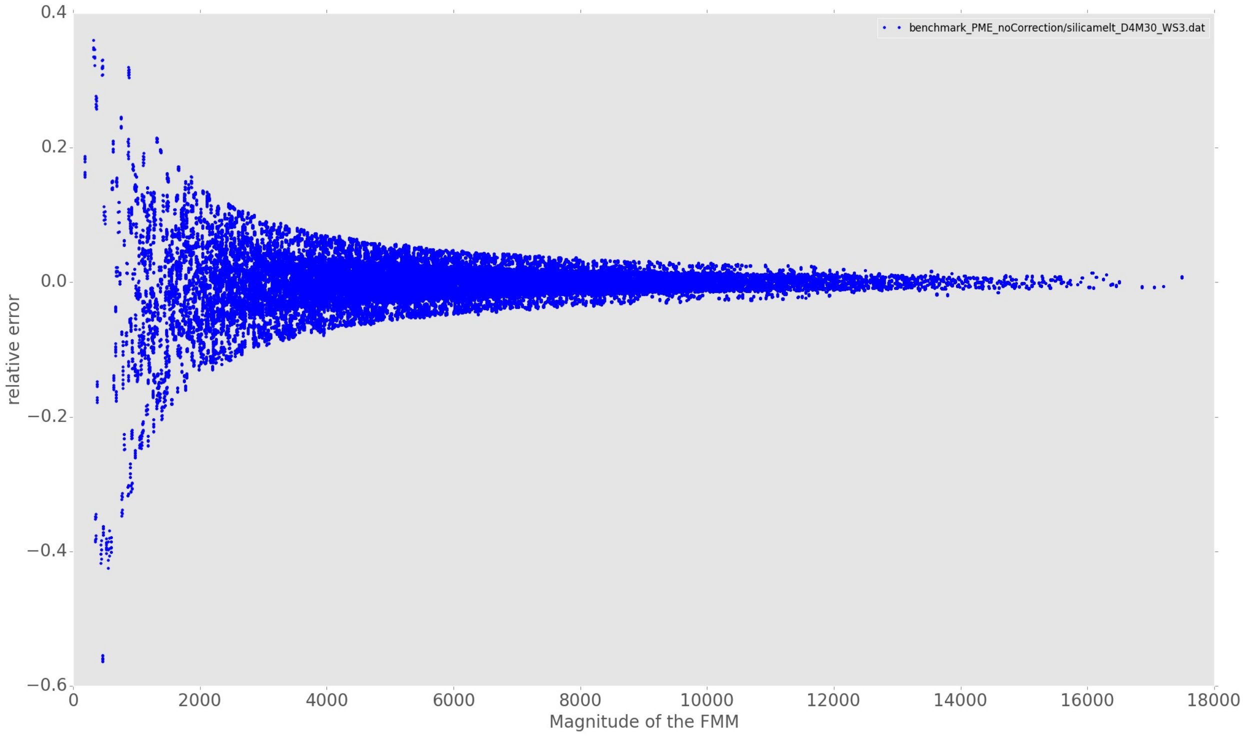


Figure 4.2: Figure representing the Magnitude of the force in the x-axis and the error in relative error in the y-axis. It is possible to see that the biggest errors are when the magnitude of the force is smaller.

Some statistical information about the distribution gives us a maximum error of about 56.35%, this maximum error is much bigger than expected and is therefore not acceptable.

As shown figure (4.2) we study the distribution of errors according to the magnitude of system we can see that the biggest errors are some smaller magnitudes.

4.4.2 Dipole correction

Study of a two-particle system

In order to understand these big errors, we first decided to study how 2 particles behave. Then if the errors are small enough, it is possible to move to bigger systems.

We also observed that if we set 4 particles in a square in a quadrupole, so there is no dipole here, the value of the FMM system and the PME system are matching

The hypothesis is that the dipole moment may be corrected in one system and not in another system. So the idea is to take two particles, one positively charged and the other negatively charged ; the two particles will form a dipole, namely :

$$\vec{p} = q\vec{d} \tag{4.2}$$

where $\vec{d}$ is the displacement vector pointing from the negative charge to the positive charge.

Then we can vary the distance between the two particles, compute the force using both the FMM method and the PME method and compare the force.

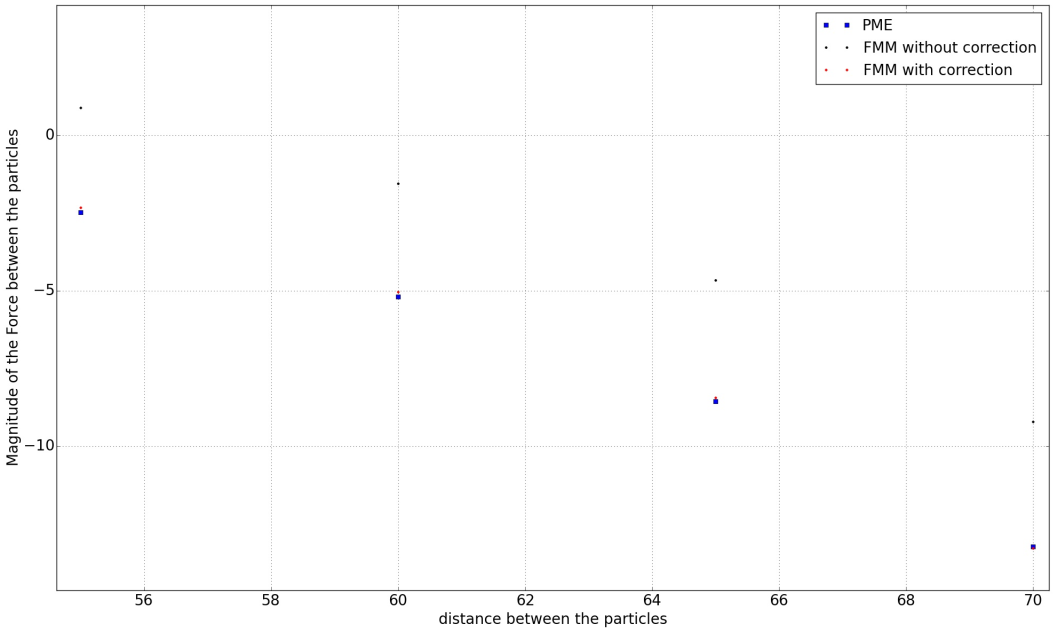


Figure 4.3: Figure representing the distance between two particles the x-axis (here 55,60,65,70 nm) and the Magnitude of the force for PME (Blue Square), FMM without correction (Black point) and with correction(red star). It is possible to see that the dipole correction makes the FMM magnitude much closer to the PME.

The dipole correction is done the following way :
It is possible to modify the potential in the FMM code using the following equation:

$$\nabla_i E_{\text{after correction}} = \nabla_i E_{textbeforecorrection} - \frac{4\pi}{3 \cdot V} q_i \mathbf{d} \tag{4.3}$$

where V is the volume of the simulation box, q_i the charge of the particle and $\mathbf{d}$ is the dipole moment.

The dipole moment can be computed from the multipole moments, which contains the dipole moment : The multipole moments are contained in a triangluar array (because of symmetry and memory space reasons) the following way :

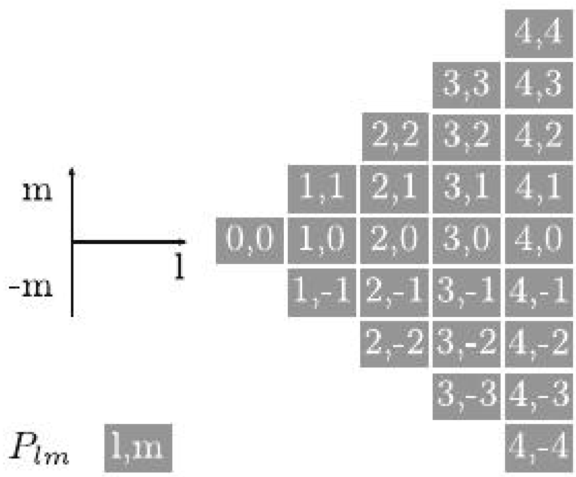


Figure 4.4: Structure of the array containing the multipoles $\omega_{lm}\mathbf{a})$: the 'm' values are on the y-axis and the 'l'-values are on the x-axis. The dipole moment is contained in the boxes (1,1), (1,0) and (1,-1)

As showed in figure (4.4), it is possible to obtain the dipole moment from the multipole ones, using the following formula :

$$\begin{cases} \vec{d_x} = & -2 * \Re(\omega_{1,1}) \\ \vec{d_y} = & 2 * \Im(\omega_{1,1}) \\ \vec{d_z} = & -\Re(\omega_{1,0}) \end{cases} \tag{4.4}$$

Then the correction can be applied using equation (4.3)

4.4. First Comparisions and Dipole Correction

The code used for this correction, in C++11, is the following :

```
for (size_t i = 0; i < n; ++i) {

    auto cell_volume = (abc.a).x * (abc.b).y * (abc.c).z; // Only
        works with square Cell Unit
   std::cout << "Periodic Vector = " << (abc.a).x << std::endl;
   std::cout << "Volume " << cell_volume << std::endl;


    std::cout << Real3(ordered_particles[i]) << "\t"
        <<reference_center << std::endl;

    auto dr = Real3(ordered_particles[i]) - reference_center;
    std::cout << "dr = " << dr << std::endl;

    const auto& omega = tree_omega[0][0];
     dipole_moment.x = -omega.get(1,1).real()*2;
    dipole_moment.y = omega.get(1,1).imag()*2;
    dipole_moment.z = -omega.get(1,0).real();

    // Potential Correction
   auto correction = (4.0*PI/(3.0 * cell_volume )) * (dr.x *
       dipole_moment.x) + (dr.y * dipole_moment.y) + (dr.z *
       dipole_moment.z);
   std::cout << "Potential Before = " << potential[i] << std::endl;
   std::cout << "Correction = " << correction << std::endl;
    potential[i] -= correction;
   std::cout << "Potential After = " << potential[i] << std::endl;
    /*
    efield[i].x -= 4*PI/3 * dipole_moment.x;
    efield[i].y -= 4*PI/3 * dipole_moment.y;
    efield[i].z -= 4*PI/3 * dipole_moment.z;
    */



            Real q = -(-ordered_particles[i].s); // negative gradient
            force[i] = efield[i] * q - dipole_moment *
                (4.*PI/(3.0*cell_volume)) * q ;
       Ec2 += q * potential[i];
        }
```

```
Ec += Ec2 * 0.5;
```

Back to the silicamelt simulation with dipole correction

Now we can relaunch the simulation with the same parameters as before, including the dipole correction. We obtain now a maximum error of 3.52%, which is a much bigger improvement compared to the 56.35% of the former simulation.

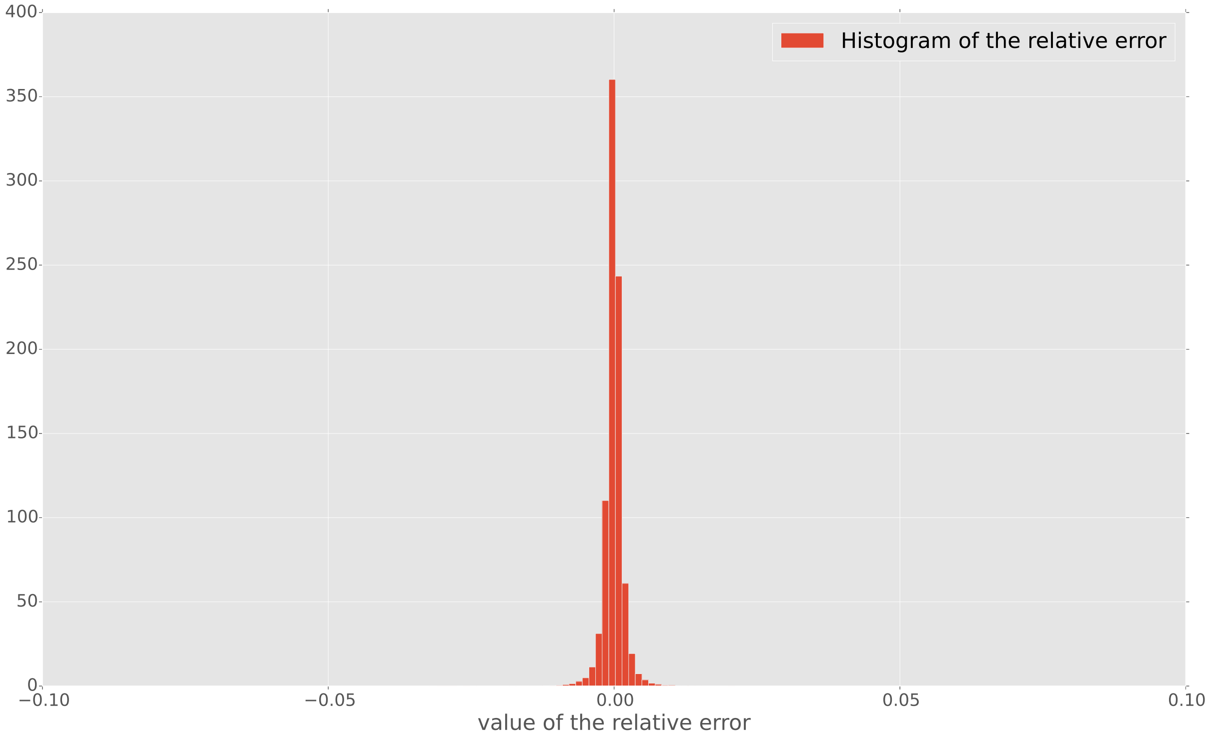


Figure 4.5: Histogram representing the relative error for a silicamelt simulation : The histogram is computed with the dipole correction. The scale in x-direction is the same as in figure (4.1)

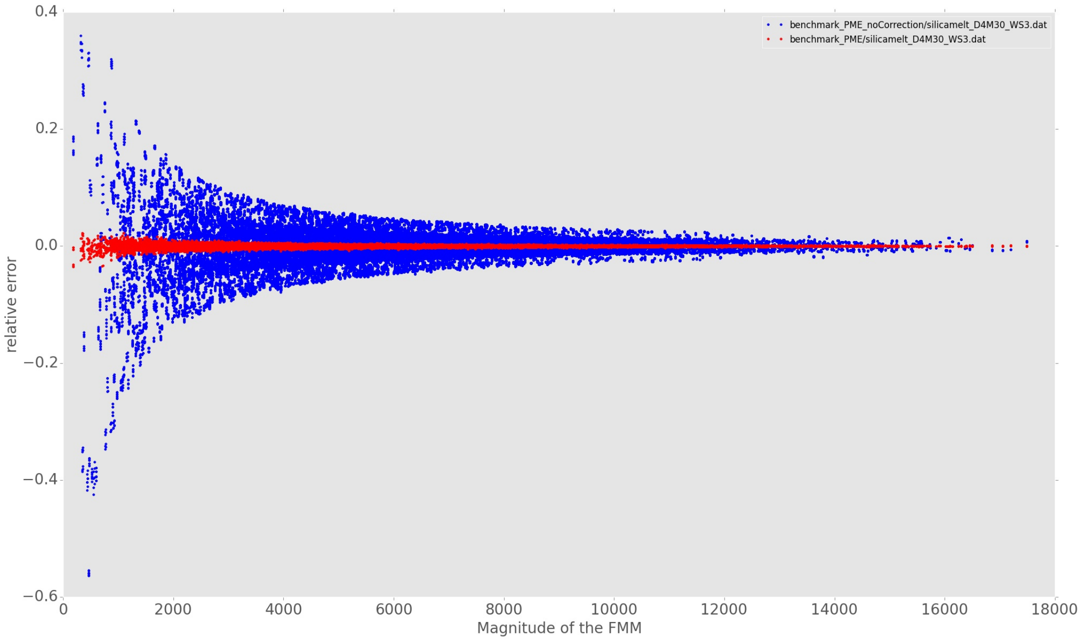


Figure 4.6: Figure representing the Magnitude of the force in the x-axis and the error in relative error in the y-axis. The blue points represent the force before the dipole correction,then red points after the dipole correction

4.5 Comparing with an analytical solution : the NaCl system

One other thing we want to try is to see the convergence of the scheme. The idea is to know which set of parameters for the FMM has the same accuracy than a given set of parameters for the PME method.

In order to study that, we want to have a system which is analytically solvable with periodic boundary conditions : That is why we decided to use a cristal as a system.

4.5.1 Generating the system

The system used is based on a NaCl cristal structure. As a cristal is a stable structure, the force acting on each atom has to be equal to 0. The potential felt by one atom can also be computed with a series ; this series converges to a value called the **Madelung Constant** which is equal for this cristal structure to approximnately 3.495 V.

The generation is done with a python script writing the charges and the positions of the atoms in a *.qxyz file, then translated into a pdb file to obtain cristal structure necessary for GROMACS.

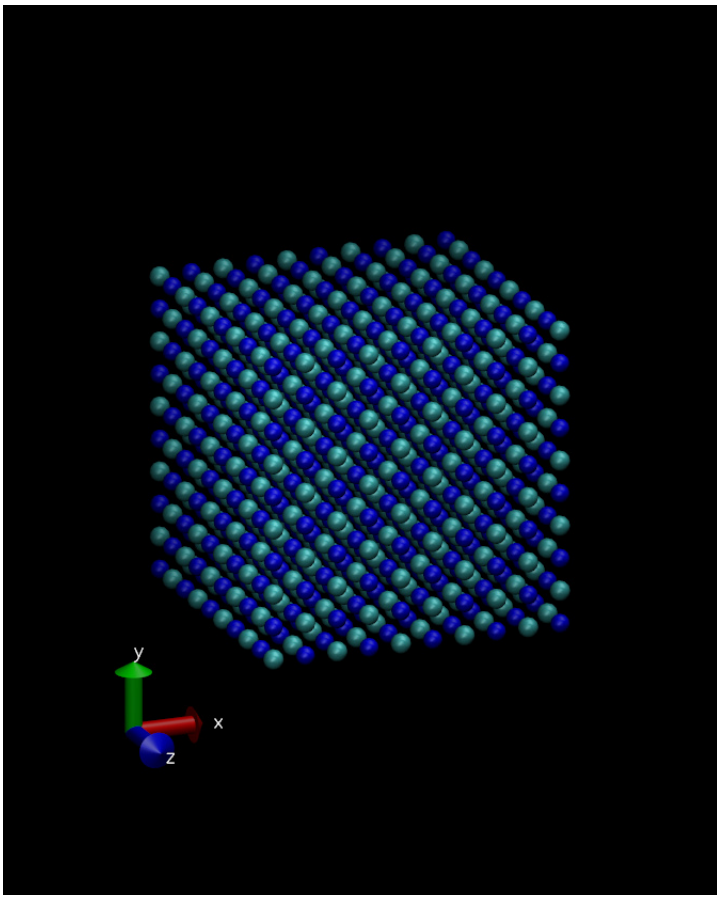


Figure 4.7: NaCl system generated with the script described above : The Blue atoms are Na atoms and the Green ones are Cl atoms. The Following image has been rendered using the visualization software VMD (Visual Molecular Dynamics)

4.5.2 Some explainaitions on the Owl system

Now I will quickly explain how to submit a certain program to the cluster of the lab, which is called *owl*.

Owl is the cluster of the lab used for the MD simulations :

The cluster is accessible via an ssh connection :

4.5.3 Error plots

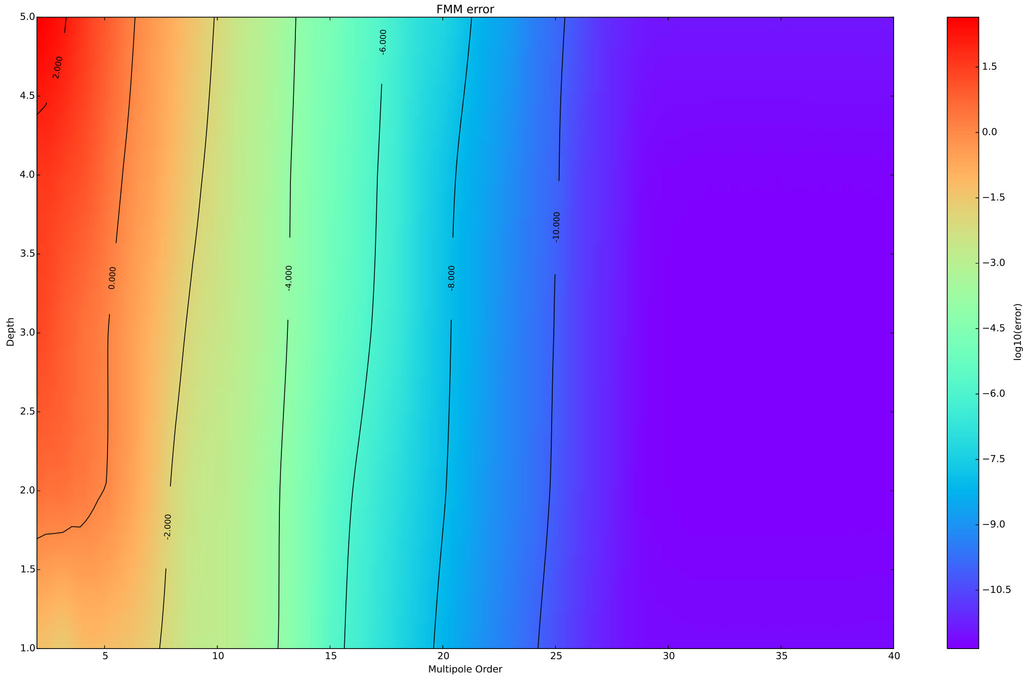


Figure 4.8: Figure representing the Magnitude of the force in the x-axis and the error in relative error in the y-axis. The blue points represent the force before the dipole correction,then red points after the dipole correction

Appendices

Appendix A

Glossary and Abbreviations

GROMACS GROningen MAchine for Chemical Simulations

PME Particle Mesh Ewald

FMM Fast Multipole Method

Owl MPI-BPC Cluster Used to make the Simulations

MPI Message Passing Interface

Omp OpenMP

Appendix B

Structure of Gromacs files

The contents...