Implementing Particle Mesh Ewald Method for Simulating Charged Particles

Project Goal

Implement a Particle Mesh Ewald method with neighbor list to calculate electrostatic forces in Molecular Dynamics simulations

Theoretical Background

Molecular simulation of materials and biopolymers is a powerful method to understand and explain physical properties and make design predictions. Usually such simulations are done with periodic box conditions (to capture the bulk nature of a real system) and involve long and short range interactions of particles. Typically, the main 2 components of these interactions are Van der Waals (1st parenthesis in Eq 1) and electrostatic term (given by coulomb's law in the 2nd parenthesis)

$$E(r) = \left(\frac{A_{rep}}{r^{12}} - \frac{B_{disp}}{r^{6}}\right) + \left(\frac{q_1 q_2}{r}\right) \tag{1a}$$

$$F(\vec{r}) = -\vec{\nabla} E(r) \tag{1b}$$

One of the main challenges in calculating these energies/forces in periodic systems is their slow convergence with distance. Given the pairwise nature of the potential, the obvious way forward is to implement a pairwise double loop to calculate all the pair energies. The computational cost scales with system size as $O(N^2)$ where N is the number of particles and this can become very slow with standard systems of size 10^3-10^4 particles[1]. The usual practice is truncating them at some cutoff distance and ignoring contributions beyond the cutoff distance, making the calculation scale as O(N). From an accuracy standpoint, this can be problematic for a potential with a $1/r^n$ term because the contribution beyond a cutoff distance r_c for a system with N particles of uniform density ρ is given by

$$U_{tail} = 2\pi N \rho \int_{r_c}^{\infty} \left[\frac{1}{r^n}\right] r^2 \, dr \tag{2}$$

For potentials with n > 3, like the Van der Waals potential in (1), the truncation is reasonable as the tail decays to zero. However, the electrostatic term in (1) has n=1 that means it is not convergent. Thus it is important to consider long-range interactions for such a potential.

A sophisticated way to overcome the problem of $O(N^2)$ scaling is to use the method of Particle Mesh Ewald proposed by Darden et. al[3]. In this treatment, the long-range

contribution is solved using Poisson's equation that converges rapidly in Fourier space when solved on a grid. Let's consider a system of N particles with charges q_i at positions r_i in an overall neutral cubic simulation box of length L. Since periodic boundary conditions are used, the electrostatic energy of the box is given by

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{n\in Z^3}\frac{q_i q_j}{\left|r_{ij}+nL\right|} \tag{3}$$

The sum over n takes into account the periodic images of the charges. Now the trick here is to decompose the above sum into two separate sums using a convergence function f(r) as seen in (4).

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{n\in Z^3}\frac{q_i q_j f(r)}{\left|r_{ij}+nL\right|} + E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{n\in Z^3}\frac{q_i q_j \left[1-f(r)\right]}{\left|r_{ij}+nL\right|} \tag{4}$$

The idea here is to distribute the two main complications of the coulomb potential – rapid variation at small values of r and slow decay at large value of r using a clever choice of f. This means

(1) The first sum should be negligible beyond a cutoff distance. This represents what is called the real space component and can be evaluated directly. The value of f(r) should be smooth for small r.
(2) The second sum represents the long range tail that can be calculated using a Fourier transform. At high r, the function f should decay quickly to zero so that this term is accurately represented by a small number of wave vectors k. This represents what is called the reciprocal space component of the potential.

The traditional selection for f(r) is the complementary error function erfc(r) defined by

$$f(r) = erfc(r) = \frac{2}{\sqrt{\pi}}\int_{r}^{\infty} e^{-t^2}\,dt \tag{5}$$

It can be derived that the final energy is given by

$$E = E^{(r)} + E^{(k)} - E^{(s)} \tag{6}$$

where the first term $E^{(r)}$ gives the real space electrostatics sum (corresponding to the 1st sum in (4))

$$E^{(r)} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{n\in Z^3}\frac{q_i q_j erfc(\alpha\left|r_{ij}+nL\right|)}{\left|r_{ij}+nL\right|} \tag{7}$$

The 2nd term $E^{(k)}$ is the contribution from reciprocal space

$$E^{(k)} = \frac{1}{2L^3}\sum_{k\neq 0}\frac{4\pi}{k^2}\left|\tilde{\rho}(k)\right|^2 \exp\left(-k^2/4\alpha\right) \tag{8}$$

To avoid overcounting in (8), a self energy term has to be subtracted from the sum. This accounts for $E^{(s)}$. This term being a constant does not contribute to the force of the particle.

$$E^{(s)} = \frac{\alpha}{\sqrt{\pi}}\sum_{i=1}^{N} q_i^2 \tag{9}$$

The inverse length parameter α is referred to as the Ewald parameter and tunes the relative weight between the real space and the reciprocal space contribution. It is inversely proportional to the real space cutoff distance r_c. The final result of the calculation is independent of α by construction. The Fourier transformed charge density $\tilde{\rho}(k)$ is defined by

$$\tilde{\rho}(k) = \sum_{j=1}^{N} q_j e^{-ik.r_j} \tag{10}$$

Brief Description of Implementation

We now briefly describe the actual implementation of the method in our code. All the classes and function have been documented using Doxygen and can be found in documentation.pdf. For illustration purposes, everything has been described below for a 2D case. **However, in our code, they are all done for a 3D system**.

A. Setting up the system

We are simulating a system of Na and Cl ions in a lattice. At the beginning, depending on the input of number of ions and simulation box size, the code sets up a symmetrically distributed system with alternating Na and Cl. More on the input file can be found in section D. The simulation requires a set of parameters (example charges, mass etc) for each molecule that we have taken from the Amber force field. More details about the parameters can be found in the reference. Each atom is also assigned a random initial velocity taken from a Gaussian distribution.

B. Running the simulation

Since we are doing Molecular Dynamics simulation, we need to follow the trajectory of each particle evolving under Newton's law of motion. The simulation is carried out using discrete time steps Δt. This time step depends on the atoms being simulated. For example, we used a time step of 1 femtosecond ($\sim 10^{-15}$ s) .The position and

velocity of each particle at a later point t is calculated by using the following discretized equations of motions (formally referred to as velocity verlet algorithm)

$$\begin{aligned}
\vec{r}(t+\Delta t) &= \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{2}\vec{a}(t)\Delta t^2 \\
\vec{v}(t+\frac{\Delta t}{2}) &= \vec{v}(t) + \frac{1}{2}\vec{a}(t)\Delta t \\
\vec{a}(t+\Delta t) &= -\frac{1}{m}\vec{F}[r(t+\Delta t)] \\
\vec{v}(t+\Delta t) &= \vec{v}(t+\frac{\Delta t}{2}) + \frac{1}{2}\vec{a}(t+\Delta t)\Delta t
\end{aligned} \tag{11}$$

Briefly, the position at each time step is calculated using the velocities and acceleration of the previous time step and the half time step velocity is updated using the acceleration of the previous time step. The new acceleration is determined from Newton's 1st law using the forces calculated for the updated system. Finally, the updated velocity evaluated using the half step velocity and the updated acceleration. More details about the force calculation can be found in section C.
In this manner, we can simulate the system for any number of time steps. For this project, we tested up to 5,000 timesteps (~ 5 picoseconds).

C. Force Calculation

The most important aspect of Molecular dynamics simulations is the evaluation of forces in the system as it determines how the molecules evolve with time and in turn, physically relevant properties. As mentioned in the introduction, we have simulated the system with only 2 forces – Van der Waals and Coulombic interaction (Eq. 1). Since we only have a finite box, a serious issue when trying to compare with real world experimental data is finite size effects. This can be alleviated to a good extent by using periodic box condition.

C.1. Periodic Box Condition

To simulate bulk like condition, a common trick is to use periodic box conditons for the system. As illustrated in Figure 1, when the force is being calculated between atom i and atom j, we take the minimum image distance between them. If the distance between 2 particles, r_1 is less than L/2, we take the value of $r = r_1$. However, if we have $r_1 > L/2$, we take $r = r_2$ i.e the distance of the image from the green particle.

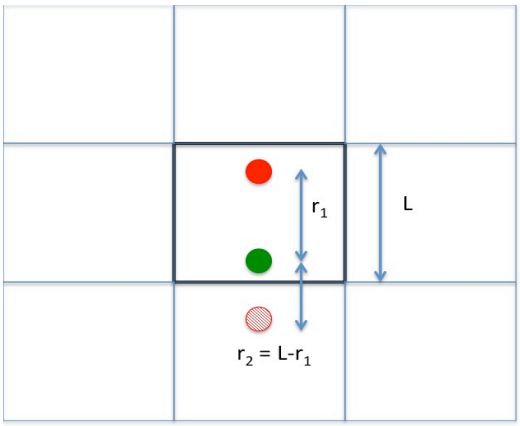


Fig 1: *Periodic Box condition*. This trick is employed to replicate bulk conditions in finite sized systems. When evaluating the force between 2 particles, the distance r is taken to be r_1 when $r < L/2$. Otherwise, the distance with the nearest periodic image of the red particle, r_2 is used.

C.2. Force Evaluation

The force evaluation is done using Eq. 1. As mentioned before, the pairwise way of doing electrostatic/Van der Waals calculations scale as $O(N^2)$ where N is the number of particles. We have implemented both the pairwise as well as the Particle Mesh Ewald method to speed up the calculation of forces. We describe the PME implementation now. The real space contribution to electrostatics (Eq. 7) can be calculated in a pairwise fashion. Since it decays fast, we only consider interactions between atoms within a certain cutoff distance r_c . As Van der Waals forces decay fast as well, we evaluate it along with the short range component. To further speed up this calculation, we split up the simulation box into cubes and assigned particles to each box. Fig 2 illustrates the scheme for a 2D case. The length of the cube is $r_c/2$ and only considering neighboring cubes during force evaluation. For example, for the particle colored green, we only considered particles in cubes colored red for force evaluation.

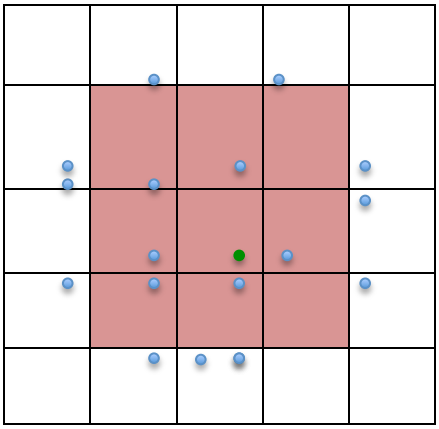


Fig 2: *Construction of neighbor list for short range forces*. Neighbor lists were constructed by dividing up the simulation box into cubes (in figure above in 2D) assigning each particles to a cube shown above. Force evaluation was only done between neighboring cubes. Thus for the green particle highlighted, only the particles in the red cubes were considered for short-range force evaluation. Cube length was $r_c/2$

The particles are reassigned cubes after a certain number of steps of simulation that depend on the average velocity of the system as well as cube length. We approximated this to be about 1000 simulation steps.

The long-range part (Eq. 8) is evaluated as follows:

(1) The entire simulation box is divided up into grids with spacing dictated by cutoff r_c

(2) The nearest grid point i^m for each particle m can be determined using the following equation

$$i^m = \left\lfloor \frac{\vec{r}^m}{h} \right\rfloor \tag{11}$$

Here h is the grid spacing. For example in Figure 3, the grid points closest to the green particle have been circled with grid point i labeled.

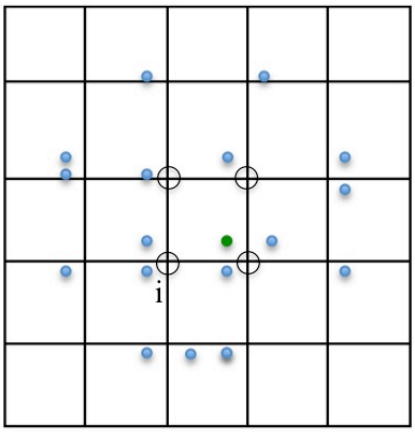


Fig 3. *Deposition of charges on the grid*. The simulation box is divided up into grids of spacing h. For each particle charges are deposited to the closest grid points($\{i\}$). For the 2D case illustrated here, they are the circled points for the green particle.

(3) The fraction of charge assigned to each point is determined by its distance from the particle as shown in equation 12.

$$s^m = \frac{\vec{r}^m - i^m h}{h} \tag{12a}$$

$$\begin{aligned}
\rho_{i^m} &+= q_m (1 - s_0^m)(1 - s_1^m) \\
\rho_{i^m+(1,0)} &+= q_m (s_0^m)(1 - s_1^m) \\
\rho_{i^m+(0,1)} &+= q_m (1 - s_0^m)(s_1^m) \\
\rho_{i^m+(1,1)} &+= q_m (s_0^m)(s_1^m)
\end{aligned} \tag{12b}$$

ρ_i is the charge density on each grid point i.

(4) Once the charge deposition is done, we obtain the fourier transform of the charge density using FFTW given as ρ^M_i (k) where k represents the k^{th} vector. This is then multiplied by the greens function (first parenthesis in Eq 13) as well as a Gaussian smearing function to obtain the electrostatic potential ϕ^M(k) in Fourier space. The expression (Eq. 13) is actually a solution to the Poisson equation in Fourier space for a Gaussian smeared charge distribution

$$\phi_i^M(k) = \left(\frac{4\pi}{k^2}\right)\left(\exp\left(\frac{-k^2}{4\alpha}\right)\right)\rho_M(k) \tag{13}$$

As can be seen, the advantage of the Fourier space representation is that only a few k vectors are needed for convergence of the electrostatic potential.

(5) ϕ^M_i(k) can be transformed back to real space using the inverse FFTW. Once in real space, we take the gradient of the potential to obtain the field at each grid point (Eq 14a). The total field for each particle, U^m can then be calculated using Eq 14b

$$\vec{U}_i^M = -\vec{\nabla}\phi_i^M(r) \tag{14a}$$

$$\begin{aligned}
\vec{U}^m = &\vec{U}^M_{i^m}(1 - s_0^m)(1 - s_1^m) \\
&+ \vec{U}^M_{i^m+(1,0)}(s_0^m)(1 - s_1^m) \\
&+ \vec{U}^M_{i^m+(0,1)}(1 - s_0^m)(s_1^m) \\
&+ \vec{U}^M_{i^m+(1,1)}(s_0^m)(s_1^m)
\end{aligned} \tag{14b}$$

The field when multiplied by the charge of the particle q_m gives the long-range force on that particle which is added to the total force.

At this stage, a couple of important points deserve clarification. A key aspect of dividing the work between long range and short range electrostatics is the value of α that is inversely proportional to the cutoff distance r_c. We chose a default value of 10^4 $nm^{-1.}$ This was chosen based on the lengthscale (nanometer) and a preliminary attempt to have equal contribution from real and reciprocal space. This is not an optimized number and in fact will require considerable effort to optimize it for specific systems.
The 2nd point we wish to discuss is the number of k-vectors used to calculate the potential in Fourier space. Our software uses the FFTMD implementation given by Prof. Collela in homework 4. This implementation does not have anyway of choosing lesser number of k-vectors and we were unable to tune it to our specific need due to paucity of time. One of the benefits of using PME is using relatively few k-vectors (defined by green circle in Fig 4) to speed up the calculation but a very subtle point is that we need to use negative k-vectors in Eq 8. In the implementation we used all the positive k-vectors (which in the FFTMD implementation is equal to the number of particles). An unexpected advantage of doing this is we took into account negative k-vectors by virtue of periodicity of Fourier space (Illustrated in Fig 4). Thus, effectively we only considered N/2 k-vectors but both positive and negative k-vectors.

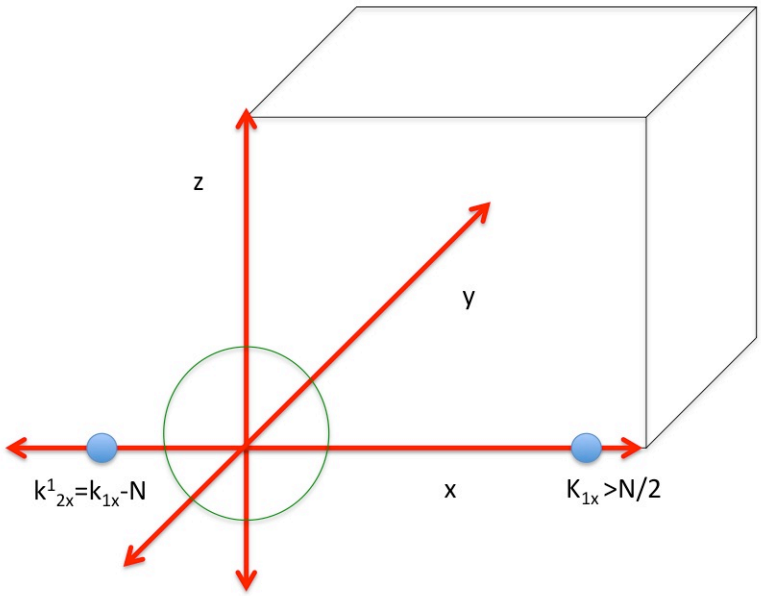


Fig 4. K-vectors used for PME calculation. The FFTMD implementation (Homework 4) was used for Fourier Space calculation. No attempt was made to include a cutoff to k-vectors (depicted by green sphere) that would have significantly increased the speed of convergence. However, since FFTMD calculated all positive k-vectors, by virtue of periodicity, contributions from the k-vectors in the negative octants are also included in our calculation.

D. Input Files

The software requires an input file (runtime.cfg) that has all the keywords for doing MD simulation followed by their values/entries. For this software to operate, all keywords are required. The default runtime.cfg file can be found in the submitted software.

TEMP: Sets the temperature of the system in Kelvin.
CELLX: Sets the Simulation box length in the x-dimension in nanometer.
CELLY: Sets the Simulation box length in the y-dimension in nanometer.
CELLZ: Sets the Simulation box length in the z-dimension in nanometer.
KINSTEP: Number of steps of Molecular dynamics to be run. For example, for this project we ran 5000 steps.
TIMESTEP: Timestep of simulation in femtosecond
NATOMEACHSIDE: Number of atoms in each direction. Thus the total number of atoms in the system is $(NATOMEACHSIDE)^3$
MOVIEFLAGON: If True, movie of the simulation is recorded in xyz format
MOVIEFNAME: File name of the xyz file to which movie of simulation needs to be saved
MOVIEFILEFREQ: Frequency of saving frames to the movie file
RESTARTVELFNAME: Filename to store final velocities of each particle. Necessary if we want to continue the same simulation later.
RESTARTPOSNAME: File name to store position of each particle in xyz format
RESTARTFREQ: Frequency of writing out the restart file
PARMFILE: The parameter file in prm format containing all the parameters of the atoms in the simulations
CUTOFF: Cutoff distance, r_c used in the simulation to separate long range and short range interactions
METHOD: Want to use Pairwise (keyword: PAIRWISE) or Neighborlist (keyword: NBLIST) or Particle Mesh Ewald with Neighborlist (keyword: PME)

A parameter file is required that has all information about the atoms in the system. The file provided with our implementation is SodiumChloride.prm and it has parameters for Na and Cl atoms.

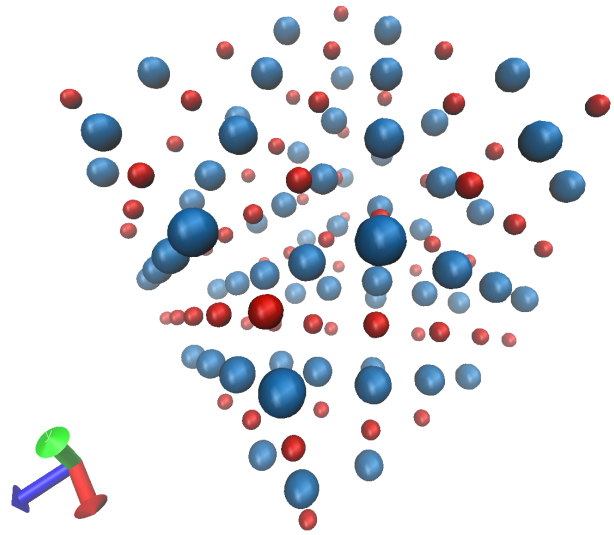
Running the Code

We have provided a Makefile (GNUMakefile) that can be used to compile the code on any machine, provided the correct compiler and FFTW libraries are specified in the Makefile. Default input files for running the simulation have been provided (runtime.cfg and SodiumChloride.prm). The executable that is created is MD.exe. Once the code is compiled, the user needs to type in ./MD.exe –p runtime.cfg. This should run the simulation.

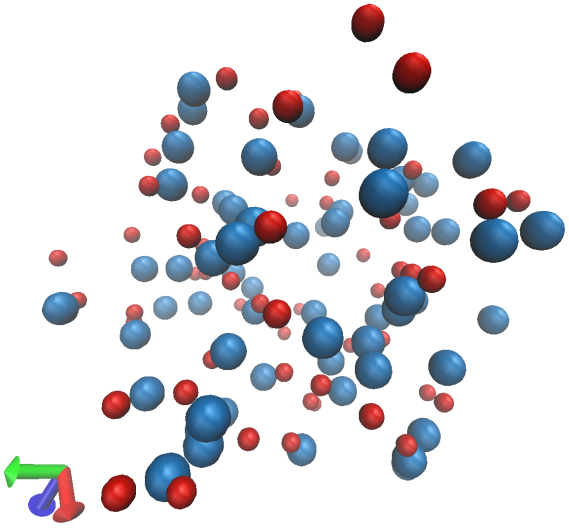
Results

We ran Molecular dynamics simulations for 5000 steps. Both Pairwise and PME methods were used to run dynamics and we can see particles displaced at the end of the simulation(Fig 5)

(a)



(b)



(c)

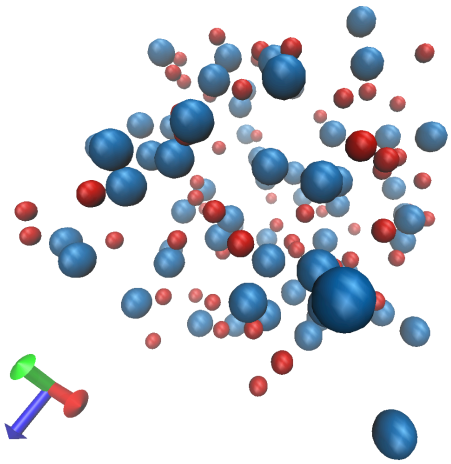


Fig 5: Snapshots from molecular dynamics simulations of Na Cl atoms. The simulation was done with the default input files provided. 125 atoms were simulated.(a) The initial configuration which is placed symmetrically with alternating Na (red) and Cl (blue). (b) Configuration after 5000 steps of dynamics using the Pairwise interaction for all forced (c) Configuration after 5000 steps of dynamics using the PME method.

Timings

We recorded whether the simulations we ran were speeded up by PME implementation. However, due to lack of optimization of the Ewald part (for example number of k-vectors) as well as small system size (~100 atoms), calculating pairwise interactions is faster. PME is known to be much faster when system size is of the order of 10^3-10^4.

Method	Time to run simulation(in sec)
Pairwise	191
PME + NeighbourList	259

References:

1. Frenkel D. and Smit B., *Understanding Molecular Simulations From Algorithms to Applications*, Academic Press, Second Edition, (2002)
2. Deserno M. and Holm C., How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines, J. Chem. Phys., 109, 18, (1998)
3. Darden T. et al, *Particle mesh Ewald: An N-log(N) method for Ewald sums in large systems*, J. Chem. Phys., 98, 12, (1993)
4. Petersen H. G., *Accuracy and efficiency of the particle mesh Ewald method*, J. Chem. Phys., 103, (1995)