School of Computer Science

College of Engineering and
Computer Science

The Fast Multipole Algorithm vs the Particle Mesh Ewald method

Joshua Nelson - u4850020

COMP3006 - Computer Science Research Project

Supervisor: Dr Eric McCreath

October 11, 2012

Abstract

The N body problem is common across the fields of physics, biology and chemistry. The classic solution to this problem has an inhibitive complexity in the class $O(n^2)$. Two alternative methods were examined: The Fast Multipole Algorithm, and the Particle Mesh Ewald Method, with better complexities of $O(n)$ and $O(n\log(n))$, respectively. These algorithms were implemented in Java, and their efficiencies were discussed and compared. The algorithms were run over typical molecular dynamics simulations to determine the most efficient algorithm for the N-body problem.

Contents

Chapter 1

Introduction

1.1 The N body problem

Suppose we have a collection of n bodies in some space, that interact with each other. Each body interacts with every other body in the system in a pairwise way. Often this pairwise interaction is a function of the distance between the bodies, and their properties, such as mass or electric charge. The task is to calculate the total effect on each body from every other body.

The N body problem is key to the simulation of many different scientific environments. The bodies may be astrophysical objects, such as planets or galaxies, interacting based on distance and body mass [3], or atoms in a molecular dynamics simulation, based on distance and particle charge.[6]. For the remainder of the report, we will discuss the N body problem in regards to Molecular dynamics, however the approaches can be generalised.

Chapter 2

Algorithms for the n body problem

2.1 The $O(n^2)$ solution

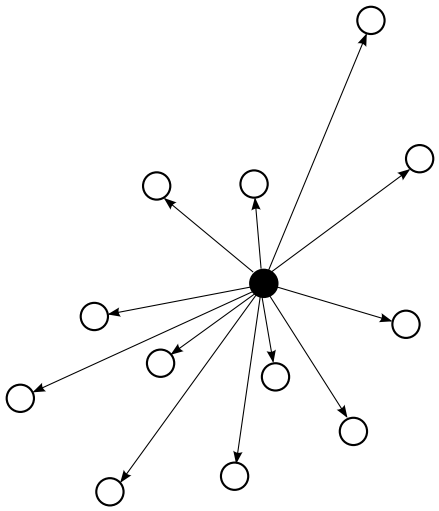

Figure 2.1: The naïve approach to the n body problem - calculate each interaction for each particle

The simplest solution to the N body problem is the basic $O(n^2)$ approach of calculating each interaction directly. Pseudocode for the algorithm is given below.

Data: r_i: particle positions, q_i: particle charges, N: number of particles, Q: output array of charges
for *i=0 to N* **do**
  for *j=i to N* **do**
    if $i \neq j$ **then**
      $d := |r_i - r_j|$;
      $Q[i] := q_i * q_j/d$;
    end
  end
end

Algorithm 1: The basic approach to the N body problem

The advantages to the $O(n^2)$ approach are it's simplicity, ease of implementation, and it's low overhead. However, it's primary disadvantage is that it is limited to small numbers of particles by

it's $O(n^2)$ complexity. These advantages and disadvantages are further discussed in Chapter 3

2.2 The particle mesh ewald method

2.2.1 Background

Ewald summation

The key concept behind the Particle Mesh Ewald methodis that of *Ewald Summation*. Ewald Summation splits the potential between two particles into two components, the long range force and the short range force. [5]

$$\phi(r) = \phi_{\text{sr}}(r) + \phi_{\text{lr}}(r)$$

The advantage of doing this is that ϕ_{sr}, the short range term, can be converges quickly in real space, while ϕ_{lr} converges quickly in reciprocal space. The Particle Mesh Ewald methodtakes advantage of this by calculating the ϕ_{sr} term by calculating potential directly for nearby particles, and uses a grid based direct fourier transformation to calculate the long range component.

Real space computation

The short range potential at a point can be computed by considering only particles within a small radius of the point. We can keep the radius small as ϕ_{sr} converges quickly in real space. This is also important, as we need to keep the number of particles considered less than N, in order to reduce the algorithm from $O(n^2)$ complexity. This is the case though, as the radius we consider is constant and less than the size of the simulation cell width, and we assume the particles are distributed randomly within the simulation cell.

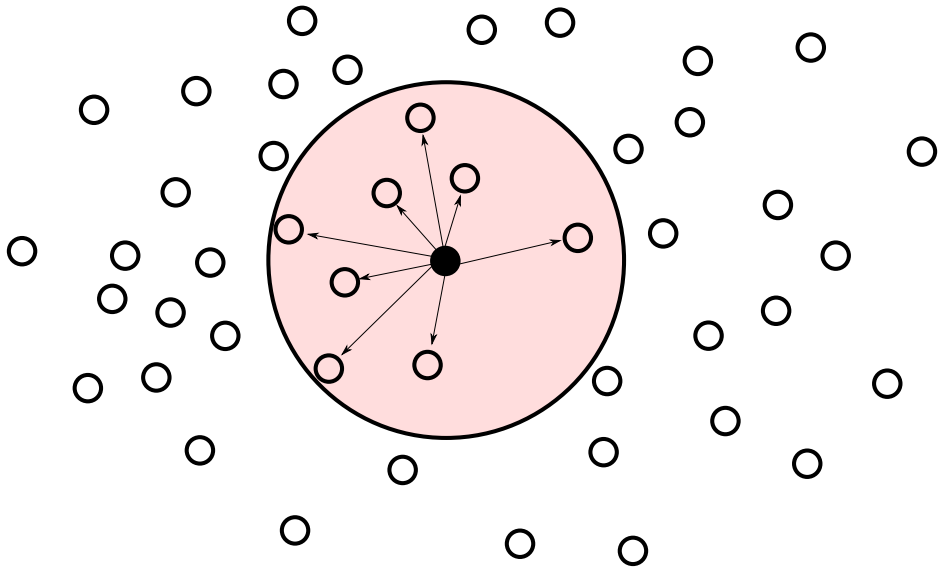

Figure 2.2: A particle and the particles that we consider it's interactions with, based on the cutoff distance

Reciprocal space computation

The reciprocal space is the long range part of the potential computation, and which converges slowly in real space. However, in reciprocal space, it converges quickly, so we use discrete fourier transformations to calculate this part of the sum.

Discrete fourier transformations require a discrete space to transform, however, our real space is continuous. So for this, we need to discretise the charges onto a grid. The approach taken in the original paper was to use Lagrangian interpolation to achieve this. Close mesh cells receive most of the charge from each particle, and this amount decreases to zero at some point, depending on the interpolation order. This is described in detail in Section 2.2.2.

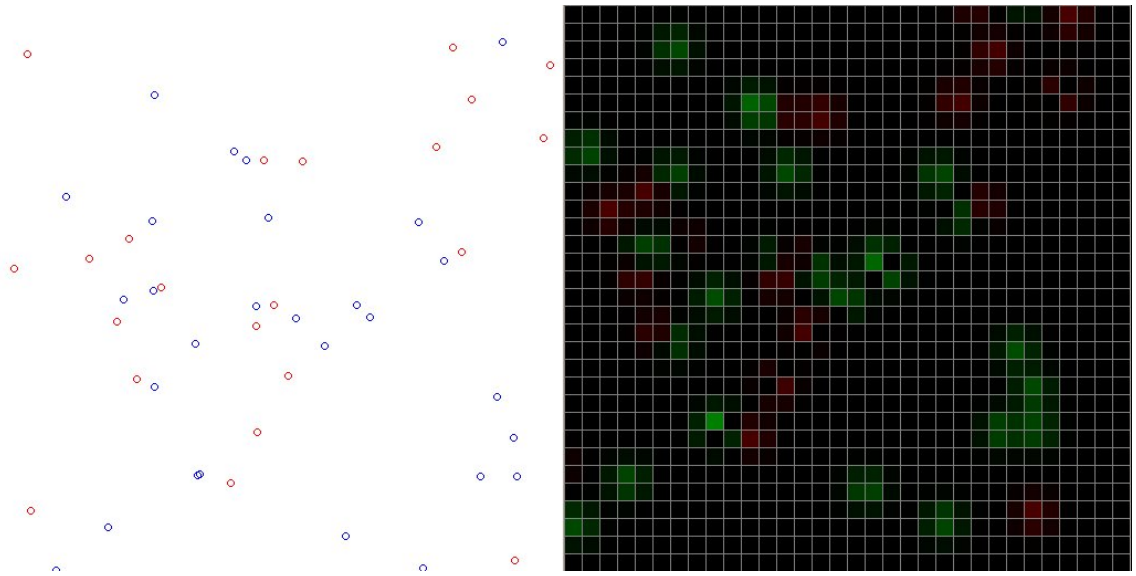

Figure 2.3: A distribution of continuous charges, and an interpretation of them as discrete charges

An alternative to using lagrange interpolation is to use cardinal B splines. This modification is known as the *Smooth Particle Mesh Ewald* method, and is advantageous in terms of accuracy, and is also easily differentiable, which is important if the forces are required as well as the potentials. [2] This is the method that was implemented in this paper.

2.2.2 Mathematical description

Interpolating the charges to the Q array

We first present the formulation of the Q array based on lagrangian interpolation.

$$Q(k_1, k_2) = \sum_{i=1}^{N} \sum_{n_x, n_y} q_i W_{2p}(u_{xi} - k_1 - n_x K) * W_{2p}(u_{yi} - k_y - n_y K) \tag{2.1}$$

Where N is the number of particles,
n_1, n_2 are integers $< N$,
W_{2p} is a Lagrangian polynomial of order p with a value in the range $[0, 1]$,
K is the number of cells we split the grid into,
u_{xi}, u_{yi} are scaled fractional coordinates of particle i (Scaled fractional coordinate coordinates meaning $u_{xi} = K * (r_{xi}/(\text{simulation width}))$, r_{xi} is the particle i's x coordiante, so $0 \leq u_{xi} \leq K$). More information can be found in [2]

We can replace W_{2p} in the above with M_n, a cardinal B spline of order n, and the formulation of the Q array is the same. From this point forward, we will use cardinal B spline interpolation.

Calculating electrostatic potential from the Q array

With this Q array we can long range contribution to the electrostatic potential in reciprocal space. The reciprocal space contribution to the electrostatic energy can be written as,

$$E_{\text{rec}} = \frac{1}{2 * \pi * V} \sum_{m \neq 0} \frac{\exp(-\pi^2 m^2 / \beta^2)}{m^2} B(m_1, m_2) S(m) S(-m) \tag{2.2}$$

Where V is the volume (or in the two dimensional case, area) of the simulation cell,
β is the ewald coefficient,
S is the structure factor.
B is the matrix of B spline inverse fourier transform moduli, $B(m_1, m_2) = |b_1(m_1)|^2 * |b_2(m_2)|^2$. More detail on this can be found in [2].
It is shown in [2] that $S(m) \approx F(Q(m))$, so we can rewrite this as a convolution

$$E_{\text{rec}} = \frac{1}{2} \sum_{m_1=0}^{K} \sum_{m_2=0}^{K} Q(m_1, m_2) * (\theta_{\text{rec}} \star Q)(m_1, m_2) \tag{2.3}$$

With $\theta_{\text{rec}} = F(B * C)$, and so $(\theta_{\text{rec}} \star Q)(m_1, m_2) = F(B * C * F^{-1}(Q))$ [2] [4]
Where C is the matrix for the original exponential term from Equation 2.2, that is,

$$C(m_1, m_2) = \frac{1}{\pi V} \frac{\exp(-\pi^2 m^2 / \beta^2)}{m^2} \text{ for } m \neq 0, C(0,0) = 0$$

The ewald coefficient

The ewald coefficient, β, is a number describing the ratio between the real space and the reciprocal space contributions to the calculation of the total energy. In practice, it depends on the tolerance ϵ_{tol}, and our desired cutoff distance r_{cut} in the following way, [1] [2]

$$\frac{\text{erfc}(\beta r_{\text{cut}})}{r_{\text{cut}}} \leq \epsilon_{\text{tol}} \tag{2.4}$$

Where erfc is the complimentary error function, a function that tends quickly to zero, depending on it's argument. The relation means that for $r > r_{cut}$, we have $\frac{\text{erfc}(\beta r_{\text{cut}})}{r_{\text{cut}}} \leq \epsilon_{\text{tol}}$, and for all $r > r_{\text{cut}}$ we can ignore the direct energy contribution, given by

$$E_{\text{dir}} = \frac{1}{2} \sum_{n} \sum_{i,j=1}^{N} \frac{q_i q_j \text{erfc} \beta |r_j - r_i|}{|r_j - r_i|} \tag{2.5}$$

(Equation from [2])

2.2.3 The algorithm

Main particle mesh ewald flow

The basic flow of the algorithm is given below

Initialise the ewald coefficient (Equation 2.4)
Calculate the B spline coefficients (Details in [2])
Allocate particles to their cells (Section 2.2.3)
Initialise the Q matrix (Equation 2.1)
Calculate reciprocal energy (Equation 2.3)
Calculate direct energy (Equation 2.5)

Verlet list algorithm

The following algorithm utilises a list known as a *Verlet list* to keep track of which particles are within a cutoff distance of each other, with only $O(n)$ complexity. It is used in the direct energy calculation of the Particle Mesh Ewald method

```
for i=0 to N do
    cell_x := ⌈u_xi⌉; (With u_xi the scaled fractional x coordinate, as defined in Equation 2.1))
    cell_y := ⌈u_yi⌉; (With u_yi the scaled fractional y coordinate))
    direct range := ⌈r_cut/mesh cell width⌉
    for δ_x = −direct range to +direct range do
        for δ_y = −direct range to +direct range do
            if cell_x + δ_x and cell_y + δ_y are in the mesh then
                Add i to the array closeParticles[cell_x + δ_x][cell_y + δ_y]
            end
        end
    end
end
```

After this, the array closeParticles[x][y] contains all particles within r_cut distance from a particle contained in mesh cell x, y.

2.2.4 The implementation

2.2.5 Analysis of the implementation

2.3 The Fast multipole algorithm

2.3.1 Background

2.3.2 Mathematical description

2.3.3 The algorithm

2.3.4 The implementation

2.3.5 Analysis of the implementation

Chapter 3

Comparison of the algorithms

Chapter 4

Discussion

Bibliography

[1] Tom Darden, Darrin York, and Lee Pedersen. Particle mesh ewald: An n [center-dot] log(n) method for ewald sums in large systems. *The Journal of Chemical Physics*, 98(12):10089–10092, 1993.

[2] Ulrich Essmann, Lalith Perera, Max L. Berkowitz, Tom Darden, Hsing Lee, and Lee G. Pedersen. A smooth particle mesh ewald method. *The Journal of Chemical Physics*, 103(19):8577–8593, 1995.

[3] Max Planck institute of Astrophysics. The millennium simulation project. `http://www.mpa-garching.mpg.de/galform/virgo/millennium/`. Accessed: 10/10/2012.

[4] Sam Lee. An fpga implementation of the smooth particle mesh ewald reciprocal sum compute engine (rsce). 2005.

[5] Henrik G. Petersen. Accuracy and efficiency of the particle mesh ewald method. *The Journal of Chemical Physics*, 103(9):3668–3679, 1995.

[6] Theoretical and Computational Biophysics group. Namd scalable molecular dynamics. `http://www.ks.uiuc.edu/Research/namd/`. Accessed: 10/10/2012.