Chengfei Wang
Orlando Acevedo
December 11, 2017

Before Long-Distance Force: 3D Fast Poisson Solver And Its GPU Implementation

Simulation of the molecules need to calculate the force between the the charged particles [6]. We formulate the total electrostatic potential energy for a system of N point charges as follow:

$$U = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{Z_i Z_j}{r_{ij}} \quad (1)$$

The equation (1) can be re-written by subtracting and adding a smooth spherical charge with density $\hat{\rho}_i(r)$ centered on each particle position:

$$\begin{aligned} U = &\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\left[\frac{Z_i Z_j}{r_{ij}} - \int\int \frac{\hat{\rho}_i(r)\hat{\rho}_j(r')}{|r-r'|}drdr'\right] \\ &+\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\int\int \frac{\hat{\rho}_i(r)\hat{\rho}_j(r')}{|r-r'|}drdr' - \frac{1}{2}\sum_{i=1}^{N}\int\int \frac{\hat{\rho}_i(r)\hat{\rho}_i(r')}{|r-r'|}drdr' \end{aligned} \quad (2)$$

By this method, we split the whole formula as short range part and long range part. The first term represent short range interaction because it is zero beyond overlap of $\hat{\rho}_i, \hat{\rho}_j$, because the integral equals the left. The second term add back the total sum of smooth spherical charge. The last term is constant self-energy. Normally, we use Gaussian for charge distribution of $\hat{\rho}_i$:

$$\hat{\rho}_i(r) = Z_i\left(G^2/\pi\right)^{3/2}\exp[-G^2(r-r_i)^2] \quad (3)$$

Use this distribution, we actually split the interaction 1/r into (1-f(r))/r + f(r)/r two terms and the f(r) is the error function. So we get the well-known the Ewald formula for the energy of main cell

$$U = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{Z_i Z_j}{r_{ij}}\mathrm{erfc}\left(Gr_{ij}/\sqrt{2}\right) + \frac{1}{2}\int\int\frac{\hat{\rho}_i(r)\hat{\rho}_i(r')}{|r-r'|}drdr' - \frac{G}{\sqrt{2\pi}}\sum_{i=1}^{N}Z_i^2 \quad (4)$$

The P3M and original Ewald some treat the second term differently. Ewald transform them into reciprocal space and calculate it analytically, which is an implicit Fourier transform, however, in P3M, this term is calculate in particle mesh and explicitly in fast Fourier transform

$$U^{(k)} = \frac{1}{2}\int\int\frac{\hat{\rho}_i(r)\hat{\rho}_i(r')}{|r-r'|}drdr' \quad (5)$$

The second term, the reciprocal space contribution, calculated numerically by solving the equation (6) (Warning: the red part in this report need further work to finish the gap between the calculation of an actual long distance value and the implemented fast Poisson solver in CUDA program. This report also did not derive the formula strictly in math, need double check.)

$$\nabla^2 U^{(k)} = -\frac{\rho}{\epsilon} \quad (6)$$

The numerical calculation of the equation (6) on GPU can be further re-written as a problem calculation of (7), a classic numerically question of solving Poisson equation:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f(x,y,z) \quad (7)$$

The u is the value of U represent in the 3D grid, which is a function of coordinates of x, y, z.

$$u = g(x,y,z) \quad (8)$$

In finite difference method, we know that (9):

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} &= \frac{\delta_x^2}{h^2(1+\frac{1}{12}\delta_x^2)} + O(h^4) \\
\frac{\partial^2 u}{\partial y^2} &= \frac{\delta_y^2}{h^2(1+\frac{1}{12}\delta_y^2)} + O(h^4) \quad (9) \\
\frac{\partial^2 u}{\partial z^2} &= \frac{\delta_z^2}{h^2(1+\frac{1}{12}\delta_z^2)} + O(h^4)
\end{aligned}$$

The delta is the center difference operator. Put the (9) back into (7) we have the (10) as following[2]:

$$h^2\left(1+\frac{1}{12}\left(\delta_x^2+\delta_y^2+\delta_z^2\right)+\frac{1}{144}\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)+\frac{1}{1728}\delta_x^2\delta_y^2\delta_z^2\right)f_{m,n,p}$$
$$=\left(\left(\delta_x^2+\delta_y^2+\delta_z^2\right)+\frac{1}{6}\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)+\frac{1}{48}\delta_x^2\delta_y^2\delta_z^2\right)u_{m,n,p} \quad (10)$$

Simplify equation (10) we can have the equation (11) as following:

$$\begin{aligned}
&h^2\left(144+12\left(\delta_x^2+\delta_y^2+\delta_z^2\right)+\left(\delta_x^2\delta_y^2+\delta_x^2\delta_z^2+\delta_y^2\delta_z^2\right)+\frac{1}{12}\delta_x^2\delta_y^2\delta_z^2\right)f_{m,n,p}\\
&=-600u_{m,n,p}+60\left(u_{m,n,p-1}+u_{m,n,p+1}+u_{m-1,n,p}+u_{m+1,n,p}\right.\\
&\left.+u_{m,n-1,p}+u_{m,n+1,p}\right)+18\left(u_{m-1,n-1,p}+u_{m-1,n+1,p}\right.\\
&\left.+u_{m+1,n-1,p}+u_{m+1,n+1,p}+u_{m-1,n,p-1}+u_{m+1,n,p-1}\right.\\
&\left.+u_{m-1,n,p+1}+u_{m+1,n,p+1}+u_{m,n-1,p-1}+u_{m,n+1,p-1}\right.\\
&\left.+u_{m,n-1,p+1}+u_{m,n+1,p+1}\right)+3\left(u_{m+1,n-1,p-1}+u_{m-1,n-1,p-1}\right.\\
&\left.+u_{m+1,n+1,p-1}+u_{m+1,n+1,p+1}+u_{m-1,n-1,p+1}+u_{m+1,n-1,p+1}\right.\\
&\left.+u_{m-1,n+1,p-1}+u_{m-1,n+1,p+1}\right)
\end{aligned} \quad (11)$$

This formula is the 27-points stencil to calculate $\nabla^2 u$ numerically, which is solving the matrix form of linear system (12):

$$AU=B \quad (12)$$

The matrix A (13) stores the stencil and the U is flatten grid point values stacking in columns corresponding to the value of U(k) potential energy, and the B similarly corresponds to the charge distribution. (Warning: in other words, to use the FFT method in P3M, we also need to assign the spherical charges into grids, this job have not been done yet. I still do not know how.)

$$A=\begin{bmatrix}
R & S & & & & & & & \\
S & R & S & & & & & & \\
 & S & R & S & & & & & \\
 & & & & \ddots & & & & \\
 & & & & & S & R & S & \\
 & & & & & & S & R &
\end{bmatrix} \quad (13)$$

In details, the A is a blocked tridiagonal matrix, and the R block is:

$$R = \begin{bmatrix} R_1 & R_2 & & & & & & \\ R_2 & R_1 & R_2 & & & & & \\ & R_2 & R_1 & R_2 & & & & \\ & & & & \ddots & & & \\ & & & & & R_2 & R_1 & R_2 \\ & & & & & & R_2 & R_1 \end{bmatrix} \quad (14)$$

also the S block is:

$$S = \begin{bmatrix} S_1 & S_2 & & & & & & \\ S_2 & S_1 & S_2 & & & & & \\ & S_2 & S_1 & S_2 & & & & \\ & & & & \ddots & & & \\ & & & & & S_2 & S_1 & S_2 \\ & & & & & & S_2 & S_1 \end{bmatrix} \quad (15)$$

According to (11), the value of 27-points stencil, we can have the values of each M x M size matrix from (16) - (18):

$$R_1 = \begin{bmatrix} -600 & 60 & & & & & & \\ 60 & -600 & 60 & & & & & \\ & 60 & -600 & 60 & & & & \\ & & & & \ddots & & & \\ & & & & & 60 & -600 & 60 \\ & & & & & & 60 & -600 \end{bmatrix} \quad (16)$$

$$R_2 = S_1 = \begin{bmatrix} 60 & 18 & & & & & & \\ 18 & 60 & 18 & & & & & \\ & 18 & 60 & 18 & & & & \\ & & & & \ddots & & & \\ & & & & & 18 & 60 & 18 \\ & & & & & & 18 & 60 \end{bmatrix} \quad (17)$$

$$S_2 = \begin{bmatrix} 18 & 3 & & & & & & \\ 3 & 18 & 3 & & & & & \\ & 3 & 18 & 3 & & & & \\ & & & & \ddots & & & \\ & & & & & 3 & 18 & 3 \\ & & & & & & 3 & 18 \end{bmatrix} \quad (18)$$

Note that although we have all the values of equation (12), direct solving is not do-able for large system. Because the U and B vector are the flatten 3D grid, with size in x, y, z directions M

x N x P, the size of the matrix of A is $M^2N^2P^2 \approx O(n^6)$, which is extremely costly. As mentioned in[3], the efficiency of P3M method relies on the efficiently solving the numerical Poisson equation with FFT. Here we follow the scheme by[2] which extend the Hockney methods[1] from 2D grid into 3D grid.

The key step to simplify the matrix A is introducing the "modal matrix" which has the following properties:

$$Q = Q^{-1} = Q^T \quad (19)$$

and the value of each elements:

$$q_{k,l} = \sqrt{\frac{2}{M+1}} Sin\left(\frac{\pi kl}{M+1}\right) \quad (20)$$

Multiply modal matrix to small blocks we can diagonalize the original into:

$$\begin{aligned} Q^T R_1 Q &= \mathrm{diag}(\eta_1, \eta_2, \eta_3, \ldots, \eta_M) \\ Q^T R_2 Q &= \mathrm{diag}(\tau_1, \tau_2, \tau_3, \ldots, \tau_M) \\ Q^T S_1 Q &= \mathrm{diag}(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_M) \\ Q^T S_2 Q &= \mathrm{diag}(\beta_1, \beta_2, \beta_3, \ldots, \beta_M) \end{aligned} \quad (21)$$

According to (20), we easily verified that the diagonalized matrix each element values:

$$\begin{aligned} \eta_m &= -600 + 120\cos\left(\frac{m\pi}{M+1}\right) \\ \tau_m = \alpha_m &= 60 + 36\cos\left(\frac{m\pi}{M+1}\right) \\ \beta_m &= 18 + 6\cos\left(\frac{m\pi}{M+1}\right) \end{aligned} \quad (22)$$

Here the m value ranges from 1, 2, 3,… to the last grid point number M in one direction. We combine the M x M size (19) into a larger matrix with MN x MN size:

$$\mathbb{Q} = \mathrm{diag}(Q, Q, Q, \ldots, Q) \quad (23)$$

If the large modal matrix multiply the R block of the whole MNP x MNP size matrix A, we have:

$$\mathbb{Q}^T R \mathbb{Q} = \mathrm{diag}(\psi_1, \psi_2, \psi_3, \ldots, \psi_N) \quad (24)$$

and for each value:

$$\psi_n = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_M) \quad (25)$$

also,

$$\mathbb{Q}^T S \mathbb{Q} = \mathrm{diag}(\phi_1, \phi_2, \phi_3, \ldots, \phi_N) \quad (26)$$

and for each value:

$$\phi_n = \mathrm{diag}(\mu_1, \mu_2, \mu_3, \ldots, \mu_M) \text{ (27)}$$

It is easy to verify (28) - (29) by using the (20) - (23) equation concluded in small block previously:

$$\lambda_m = \eta_m + 2\tau_m \cos\left(\frac{m\pi}{M+1}\right) \text{ (28)}$$

$$\mu_m = \alpha_m + 2\beta_m \cos\left(\frac{m\pi}{M+1}\right) \text{ (29)}$$

Also, due to (19) and (23), we have the property:

$$\mathbb{Q}\mathbb{Q}^T = \mathbb{Q}^T\mathbb{Q} = I \text{ (30)}$$

Multiply the (30) into the (12), we have the new form:

$$\mathbb{Q}^T A \mathbb{Q}\mathbb{Q}^T U = \mathbb{Q}^T B \text{ (31)}$$

and each matrix now has a new form(32) - (34):

$$\mathbb{Q}^T A \mathbb{Q} = T \text{ (32)}$$

$$\mathbb{Q}^T U = \overline{U} \text{ (33)}$$

$$\mathbb{Q}^T B = \overline{B} \text{ (34)}$$

The reason we doing this is because after this transformation, the relationship of each element can be represent as following(35):

$$\mu_m \overline{u}_{mn(p-1)} + \lambda_m \overline{u}_{mnp} + \mu_m \overline{u}_{mn(p+1)} = \overline{b}_{mnp} \text{ (35)}$$

The $\overline{u}$ and $\overline{b}$ are the elements of new matrix in (33) and (34), the μ, λ are two kinds of elements in matrix T. Note that the complicated original matrix A now only contains two type elements, besides, if you observe the (35) and reshape the dimension and written in matrix, we can have (36):

$$\begin{bmatrix} \lambda & \mu & & & & & & \\ \mu & \lambda & \mu & & & & & \\ & \mu & \lambda & \mu & & & & \\ & & & \ddots & & & & \\ & & & & \mu & \lambda & \mu & \\ & & & & & \mu & \lambda & \end{bmatrix}_{mn} \begin{bmatrix} \overline{u}_1 \\ \overline{u}_2 \\ \overline{u}_3 \\ \vdots \\ \vdots \\ \overline{u}_P \end{bmatrix}_{mn} = \begin{bmatrix} \overline{b}_1 \\ \overline{b}_2 \\ \overline{b}_3 \\ \vdots \\ \vdots \\ \overline{b}_P \end{bmatrix}_{mn} \text{ (36)}$$

Note that the now the whole large linear system (12) becomes a serials (quantity MN) of small (size P) linear system (36). Unlike (13), in this small linear system, the tridiagonal matrix is a

single element not a block matrix, therefore, fast method can be used to solve the tridiagonal matrix system. In our CUDA program, we use cuSparseSgtsv() function[4] to solve it in $O(P)$ time, which is very fast.

To transform U and B from (12), we could use (33) and (34). However, again since the size of the U and B is very large, we do not multiply directly. In fact, we can found the from (20), (33) and (34), we have the following formula of the relationship between original element and new ones:

$$\overline{b}_{mnp} = \sqrt{\frac{2}{M+1}}\mathbf{Im}\left(\sum_{m=1}^{M} b_{mnp} e^{-\frac{mn\mathbf{i}}{M+1}}\right) \quad (37)$$

$$\overline{u}_{mnp} = \sqrt{\frac{2}{M+1}}\mathbf{Im}\left(\sum_{m=1}^{M} u_{mnp} e^{-\frac{mn\mathbf{i}}{M+1}}\right) \quad (38)$$

(37), (38) derived from Eula equation, and the sin function is the imaginary part of the e exponent. Besides, (37) and (38) is a special real version of Discrete Fourier Transformation(DFT) (39), called Discrete Sine Transformation Type I (DST-I).can be calculated from standard DFT (39)

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi kn}{N}} \quad (39)$$

In other words, the (37) and (38) can be calculated by (39), and the relation between them can be illustrated in (40)

DST-I(a, b, c) = DFT(0, a, b, c, 0, -c, -b, -a) (40)

The reason we use the DFT to calculate the DST-I is that the calculation can be efficiently done by Fast Fourier Transformation (FFT). In our CUDA program, we use the cuFFT library[5] to perform this work. In total, the efficiency can be $O(NlogN)$, here the N represent the number of molecules/atom, which roughly proportion to the O(MNP).

In conclusion, the whole fastPoissonSolver function does following jobs:

(1) Take a grid of M x N x P size grid as input, the value in each point is the charge of system assigned to the grid.

(2) Prepare the corresponding T matrix in (32) by calculating from (22), (28) and (29) (Warning: in our program for now, the set-up be done every time in the function fastPoissonSolver() function, however, it is not necessary. Once the user decided the size of the

grid, those data can be use over and over again. A condition should be added to see if preparing have already done, so no need to calculate repeatedly. This need to be improved in our program in future.)

(3) Use the FFT to transform (37) and get (34) matrix $\overline{B}$

(4) Solving simple linear system (36) in batch and get matrix $\overline{U}$

(5) Do the second time FFT to transform $\overline{U}$ back to U, and return the result back.

The source code of program "solveFastPoisson.cu" has been attached in folder, shows how to call the function in main(), and has been complied to "a.out" and tested. The README text file shows how to compile and run. Another improvement should be done in the future is that, the Scheme of the program from (1)~(5), especially before doing steps (4) and (5), the data in GPU device memory always send back to CPU and prepare the data for next steps. The transfer of data between GPU and CPU slows down the performance and can not release the full-potential of GPU parallel.

[1] A First Course in the Numerical Analysis of Differential Equations, Chapter 15

[2] An Efficient Direct Method to Solve the Three Dimensional Poisson's Equation

[3] Computer Simulation Using Particles, Chapter 8

[4] cuSPARSE library manual, function cuSparseSgtsv()

[5] cuFFT library manual

[6] Comments on P3M, FMM, and the Ewald Method for Large Periodic Coulombs Systems