Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

# A small box Fast Fourier Transformation method for fast Poisson solutions in large systems

Xiang-Wei Jiang[a], Shu-Shen Li[a], Lin-Wang Wang[b,*]

[a] State Key Laboratory of Superlattices and Microstructures, Institute of Semiconductors, Chinese Academy of Sciences, P.O. Box 912, Beijing 100083, China

[b] Material Science Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States

ARTICLE INFO

*Article history:*
Received 6 December 2012
Received in revised form
10 July 2013
Accepted 11 July 2013
Available online xxxx

*Keywords:*
Small box
FFT
Coulomb
Large scale
First-principle
Poisson equation
Parallel

ABSTRACT

We present a new divide-and-conquer algorithm to efficiently evaluate the Coulomb interaction in a large system, which is an essential part of self-consistent first-principle calculations. The total Coulomb potential $\phi(\mathrm{r}) = 1/|\mathbf{r}|$ is divided into a short range part $\phi_S(\mathbf{r})$ and a smooth long range part $\phi_L(\mathbf{r})$. The system is divided into many cuboids, with a small box defined for each cuboid plus a buffer region. For the short range part, the interaction convolution integral is calculated directly using a Fast Fourier Transformation (FFT) in the small box. For the smooth long range part, the convolution integral is evaluated by a global FFT but on a coarse grid. The conversion between the dense grid and coarse grid values is done using small box FFTs with proper mask functions. Using this small box FFT method, the total Coulomb potentials of test quantum dot systems on $480^3$ grid and $2400^3$ grid are calculated. For the $2400^3$ grid case, the calculation is carried out by tens of thousands of processors with a computational speed up close to 10 times when compared with direct global FFT calculations using the FFTW package with the maximumly allowed number of processors. The maximum relative error is $4 \times 10^{-5}$ while the absolute error is less than 0.1 meV.

## 1. Introduction

The electrostatic Coulomb potential caused by an electron charge density is one of the most ubiquitous quantities in nearly any field of physics and chemistry. It is used in Kohn–Sham orbital based density functional theory (KSDFT) calculations [1], or in orbital-free DFT (OFDFT) [2] calculations. It can also be used to study the docking of protein molecules [3,4], or can be used to evaluate the Ewald energy in a classical molecular dynamics (MD) simulation in the particle mesh Ewald method [5]. The Coulomb potential $\phi(\mathbf{r})$ can be calculated from the electron charge density $\rho(\mathbf{r})$ based on the solution of a Poisson equation with the use of different boundary conditions:

$$\nabla^2\phi(\mathbf{r}) = -4\pi\rho(\mathbf{r}). \tag{1}$$

Here and in the following, bold letters such as $\mathbf{r}$ and $\mathbf{G}$ are for three dimensional vectors while $r$ and $G$ are used to denote radial variables of any spherical one-dimensional functions. The $\phi(\mathbf{r})$ can also be calculated from the Coulomb integral:

$$\phi(\mathbf{r}) = \int V(\mathbf{r}' - \mathbf{r})\rho(\mathbf{r}')d^3\mathbf{r}' \tag{2}$$

with $V(\mathbf{r}' - \mathbf{r}) = 1/|\mathbf{r}' - \mathbf{r}|$. For periodic boundary condition, the $d^3\mathbf{r}'$ integration in Eq. (2) has to be understood as an integration over the infinite space (periodically repeated from the finite box), and the total charge of the system must be zero (or a compensating uniform background charge must be used). In the current work, we will focus on the periodic system. In some calculations, e.g., the KSDFT, the Coulomb potential calculation takes only a small fraction of the total computational time. But for others, e.g., in OFDFT, it is one of the main computational tasks. How to speed up this part of the calculation is thus essential. There are many ways to calculate the electrostatic Coulomb potential $\phi(\mathbf{r})$, e.g., generalized fast multipole method (FMM) for large systems and free boundary conditions [6–8], reciprocal space based method for treating long range interactions [9,10], as well as the recently proposed real-space interpolating scaling function method arising from wavelet theory [11–13]. Most of these efforts achieved a nearly linear $O(N \log N)$ scaling (where $N = N_1N_2N_3$ is the grid number of the three-dimensional system), which is essential for the linear scaling electronic structure methods [14,15]. In the current study, we are not interested in competing with the nearly ideal $O(N \log N)$ scaling, but rather in scaling up the parallelization of large scale calculations with millions of atoms [16].

One common approach to calculate the convolution integral of Eq. (2) is to use Fast Fourier Transformation (FFT) and work in both real ($\mathbf{r}$) and reciprocal ($\mathbf{G}$) space. Here and in the following vector $\mathbf{G}$ is defined in reciprocal space as $\mathbf{G} = g_1\mathbf{b_1} + g_2\mathbf{b_2} + g_3\mathbf{b_3}$, where

* Corresponding author. Tel.: +1 5104865571.
*E-mail address:* lwwang@lbl.gov (L.-W. Wang).

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.07.010

the reciprocal space basis set ($\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}$) is the counterpart of the real-space basis set ($\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}$) with definition of $\mathbf{a_i} \cdot \mathbf{b_j} = 2\pi\delta_{ij}$. In reciprocal space, we have $\phi(\mathbf{G}) = V(\mathbf{G})\rho(\mathbf{G})$, and $V(\mathbf{G}) = 4\pi/|\mathbf{G}|^2$. Thus, Eq. (2) can be carried out by two FFTs: one forward FFT to get $\rho(\mathbf{G})$ from $\rho(\mathbf{r})$, then a backward FFT to get $\phi(\mathbf{r})$ from $\phi(\mathbf{G})$. The $\rho(\mathbf{r})$ is described by a $N_1 \times N_2 \times N_3$ regular numerical grid. We are interested in cases where the $N_1, N_2, N_3$ is run into the order of thousands. The FFT is then carried out by parallel codes. The issue we are concerned with is how to scale such a calculation to hundreds of thousands of computer cores, as they become more available with the emergence of the new class of supercomputers. The parallel scalability of the FFT depends sensitively on its implementation and data decomposition. For a 1D decomposed FFT code, for example, the fast and popular *Fastest Fourier Transform in the West* (FFTW) [17], the maximum number of processors one can use is $N_3$ for Fortran implementation (or $N_1$ for C implementation). There are other implementations of parallel FFT with 2D data decomposition, with their scalability beyond the order of $N_1$, $N_2$ or $N_3$ [18]. Nevertheless, the overall scalability is limited by the heavy global communication needs in the FFT [18].

In the current paper, we present an alternative approach to solve Eq. (2). We like to avoid the use of global communication, instead, we will use local communication as much as possible. This will be in line with the architectures of modern petaflops supercomputers [19]. Guided by this principle, we will use a divide-and-conquer algorithm, separating the global convolution integral of Eq. (2) into many small spatially local parts. More specifically, we will decompose the long range convolution kernel $V(\mathbf{r}' - \mathbf{r})$ into a short range part $V_S(\mathbf{r}' - \mathbf{r})$, and a long range (but smooth) part $V_L(\mathbf{r}' - \mathbf{r})$. The integral $\phi_S(\mathbf{r}) = \int V_S(\mathbf{r}' - \mathbf{r})\rho(\mathbf{r}')d^3\mathbf{r}'$ can be carried out by many local small box FFTs (SBFFTs), each SBFFT will be executed by a small group of processors with only in-group communications. For the long range integral $\phi_L(\mathbf{r}) = \int V_L(\mathbf{r}' - \mathbf{r})\rho(\mathbf{r}')d^3\mathbf{r}'$, we will take advantage of the fact that $V_L(\mathbf{r}' - \mathbf{r})$ is smooth, thus short range in reciprocal space ($V_L(\mathbf{G})$ is zero for large $|\mathbf{G}|$). As a result, a rather coarse grid can be used for the $V_L$ integral. In a way, this is rather like the algorithm of calculating the nuclear Ewald energy [5]. The original real space long range Coulomb sum has been separated into two parts: one short range real space sum, and one short range reciprocal space sum. In the following we will show how this idea can be carried out in detail, especially how accurate results can be obtained compared with the original results of Eq. (2). Although the idea is simple, obtaining an accurate result is not trivial. It requires innovative use of various types of broadening functions and mask functions. The high accuracy is particularly important as the Coulomb energy is often the dominant energy in the calculation, which does not tolerate large errors. We will also demonstrate that using this approach, one can calculate $\phi(\mathbf{r})$ 10 times faster than using the global FFT directly, provided that 10 times more computer cores are used beyond what FFTW allows. This is important because for the global FFT, it ceases to scale after a certain number of processors are used, which makes it impossible to use the big supercomputers for large scale calculations (e.g., million atom OFDFT calculations [16]).

## 2. The algorithm

### 2.1. Separating $V$ into short range $V_S$ and long range $V_L$

As mentioned above, Eq. (2) will be broken into two parts, one is a short range integral, another a smooth long range integral. To do this, we need to break $V(\mathbf{r}) = 1/|\mathbf{r}|$ into $V_S(\mathbf{r})$ and $V_L(\mathbf{r})$. Due to the spherical symmetry of the Coulomb potential in both real and reciprocal space, we can omit the vector notation in the following and work on a radial one-dimensional separation of the total Coulomb potential. While it is simple to require that $V_S(r) = 0$ for $r > r_c$, it is more challenging to require $V_L(r)$ is smooth, which means $V_L(G) = 0$ for $G > G_c$. Since $V_S(G) + V_L(G) = V(G) = 4\pi/G^2$, thus we have $V_S(G) = 4\pi/G^2$ for $G > G_c$, and $V_S(G) = 4\pi/G^2 - V_L(G)$ for $G < G_c$. The functions $V_L$ and $V_S$ are thus obtained by solving the $V_L(G)$ within the range of $[0, G_c]$ while requiring $V_S(r)$ to be as small as possible for $r > r_c$.

More specifically, we have a radial 1D Fourier transform (FT) (converted from a spherical symmetric 3D FFT) as:

$$V(r) = \frac{2}{r(2\pi)^2}\int_0^{G_{\max}} \sin(Gr)GV(G)dG. \tag{3}$$

Numerically, if we use a discrete arrays $r_j = j\Delta, j = 1, M$ and $G_i = i\pi/L, i = 1, N$ to represent $r$ and $G$ respectively, then the above integral can be written in a matrix form as:

$$V(r_j) = \sum_{i=1}^{N} C_{ji}V(G_i) \tag{4}$$

where $N\pi/L = G_{\max}$ and $C_{ji} = \frac{iN}{2jL^3}\sin(ij\pi/N)$. Now, our requirement is to minimize $V_S(r)$ for $r > r_c$, or say to numerically minimize

$$O = \sum_{j=M_c}^{M} |V_S(r_j)|^2 r_j^2$$
$$= \sum_{j=M_c}^{M} j^2\left(\frac{L}{N}\right)^2\left[\sum_{i=1}^{N} C_{ji}\frac{4\pi}{G_i^2} - \sum_{i=1}^{N_c} C_{ji}V_L(G_i)\right]^2 \tag{5}$$

where $M_c = r_c/\Delta$. This quadratic minimization can be solved easily by a corresponding linear equation of dimension $N_c$. The results for $V_S$ and $V_L$ are shown in Fig. 1(a) and (b) for real and reciprocal space respectively using $r_c = 5.58, G_c = 2.79$. One can see that, $V_S(r)$ is less than $10^{-5}$ for $r > r_c$ while $V_L(G)$ is exactly zero for $G > G_c$ by construction. We also have $V_S(r) + V_L(r) = 1/r$. Note, because $\alpha V_S(\alpha r) + \alpha V_L(\alpha r) = 1/r$, one can change $r_c$ and $G_c$ easily by redefining $V_S'(r) = \alpha V_S(\alpha r)$ and $V_L'(r) = \alpha V_L(\alpha r)$, then $r_c' = r_c/\alpha$ and $G_c' = \alpha G_c$. Thus we can trade $r_c$ with $G_c$, with $r_cG_c$ fixed. In our case (Fig. 1), we have used $r_cG_c = 15.6$. A smaller value means a shorter range in both real and reciprocal space, but it will result in bigger errors in $V_S(r)$ for $r > r_c$. More choices of $(r_c, G_c)$ pairs and the corresponding calculation error analysis can be found in Appendix A.

### 2.2. SBFFT for short range integral

Now, to calculate

$$\phi_S(\mathbf{r}) = \int V_S(\mathbf{r}' - \mathbf{r})\rho(\mathbf{r}')d^3\mathbf{r}' \tag{6}$$

we will use small box FFT (SBFFT). The idea is shown in Fig. 2. In order to obtain the $\phi_S(r)$ within the domain $\Omega_{\mathrm{I}}$, we only need to integrate the $d^3\mathbf{r}'$ in Eq. (6) within the small box of $\Omega_{\mathrm{II}}$ ($= \Omega_{\mathrm{I}} + \Omega_B$) as long as the buffer size $L_c$ is larger than the short range length $r_c$. The convolution integral within $\Omega_{\mathrm{II}}$ can be performed by a pair of forward and backward FFTs on the original numerical grid points within $\Omega_{\mathrm{II}}$ (which will be called dense grid in the following). We will call this dense grid small box FFT (SBFFT) as DG-SBFFT. Clearly, the DG-SBFFT for different small box can be carried out in parallel. This provides the possibility for massive parallelization. The programming implementation of this part is straightforward.

### 2.3. Coarse global grid for long range convolution integral

The long range integral

$$\phi_L(\mathbf{r}) = \int V_L(\mathbf{r}' - \mathbf{r})\rho(\mathbf{r}')d^3\mathbf{r}' \tag{7}$$