Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 192 (2003) 244–261
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A fast algorithm for three-dimensional potential fields calculation: fast Fourier transform on multipoles

E.T. Ong [a,*], K.M. Lim [b], K.H. Lee [b], H.P. Lee [a]

[a] *Institute of High Performance Computing, 1 Science Park Road, #01-01 The Capricorn, Singapore Science Park II, Singapore 117528, Singapore*
[b] *Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore*

Received 30 January 2003; received in revised form 14 July 2003; accepted 16 July 2003

Abstract

In this paper, we present a fast algorithm for rapid calculation of the potential fields in three dimensions. This method arises from an observation that potential evaluation using the multipole to local expansion translation operator can be expressed as a series discrete convolutions of the multipole moments with their associated spherical harmonics functions. The high efficiency of the algorithm is primarily due to the use of FFT algorithms to evaluate the numerous discrete convolutions. We refer to it as the Fast Fourier Transform on Multipoles (FFTM) method. It is demonstrated that FFTM is an accurate method. It is significantly more accurate than FMM for a given order of expansion. It is also shown that the algorithm has computational complexity of $O(N^a)$, where a ranges from 1.0 to 1.3.

Keywords: Laplace equation; Multipole translation operators; Fast Fourier transform; Fast Fourier transform on multipoles

1. Introduction

The evaluation of Coulombic and gravitational interactions in large-scale ensembles of particles is generally expressed as [7,12–15]

$$\Phi(\boldsymbol{x}_j) = \sum_{i=1, i\neq j}^{N} \frac{q_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|} \quad \text{for } j = 1, \ldots, N, \tag{1}$$

where $\Phi(\boldsymbol{x}_j)$ corresponds to the potential at the field point $\boldsymbol{x}_j$ due to the effects from q_i point sources at position $\boldsymbol{x}_i$, and $\|\cdot\|$ denotes the Euclidean norm.

* Correspondent author. Tel.: +65-6419-1325; fax: +65-6778-0522.
E-mail address: onget@ihpc.a-star.edu.sg (E.T. Ong).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2003.07.004

Eq. (1) is known to be closely related to many physical problems, such as the evolution of large-scale gravitational systems in astrophysics [1,3,12–14], capacitance calculation of multi-conductors problems in electrical engineering [20–23], incompressible fluid dynamics [2], and molecular dynamics simulations in chemistry [4,8,18,19]. However, evaluating the expression directly requires $O(N^2)$ operations, which becomes prohibitively large when N exceeds several thousands.

To improve the computation, numerous fast algorithms have been developed, and the Fast Multipole Method (FMM) is definitely one of the most widely implemented algorithms. FMM was developed by Greengard and Rohklin [12–14] for N-body simulations. Subsequently, Nabor and White [20,21] implemented it in electrostatic analysis, mainly to calculate the capacitance of three-dimensional structures. The efficiency of FMM comes from the effective usage of the multipole and local expansions, which are employed repeatedly in a hierarchical manner through a series of translation operations. Greengard and Rokhlin [15] then developed a new version of FMM, by using the diagonal forms of translation operators with exponential expansions, which reduced the $O(p^4)$ scaling factor to $O(p^2)$, where p is the order of expansions. Further improvement on the FMM was made by Cheng et al. [7] with the "compressed" version of the translation operators. On the other hand, Elliott and Board [9] reduced the scaling factor to $O(p^2 \log p)$ by performing Fast Fourier Transforms (FFT) on the translation operators, and a special technique was proposed to deal with the numerical instability for large values of p.

Alternatively, using multipole expansion alone can give rise to a fast algorithm, which is generally known as the tree algorithm [1,3]. The basic idea is similar to the FMM algorithm, except that the local expansion is not used. Instead, the multipole expansion is evaluated directly at the potential point. Hence, to a certain extent, the FMM can be seen as an enhancement of the tree algorithm.

Another group of fast methods utilizes FFT to accelerate the potential evaluation. They include the particle-mesh-based approach [8,17–19], the precorrected-FFT method [23] and its variant [6]. Generally, these methods approximate a given distribution of charges by an equivalent system of smoothed charge distribution that falls on a regular grid. Subsequently, the potential at the grid points due to the smoothed charge distribution is derived by discrete convolution, which is done rapidly using FFT algorithms. However, local corrections are required for the "near" charge evaluations because these potential contributions are not accurately represented by the grid charges.

In this paper, we present an alternate fast algorithm that can also perform the potential evaluation rapidly, specifically for electrostatics analysis of charge particle systems. This method arises from an important observation that potential evaluation using multipole to local expansion translation operator can be expressed as series of discrete convolutions of the multipole moments with their associated spherical harmonics functions, where FFT algorithms can be employed to evaluate the discrete convolutions rapidly. We refer to this method as the Fast Fourier Transform on Multipoles (FFTM) method. This is an improvement on an older version of FFTM in [22], which only employed the discrete convolution on the multipole expansion. In the previous method, only the potentials at the cell centers were computed. The potentials at other desire locations (such as specific particle locations or nodal positions) were obtained by interpolating with the cell center potentials. However, the interpolation technique adopted by the method had imposed a limitation on the order of accuracy attainable by the previous method (to approximately 2–3 digit accuracy). This current version of FFTM resolves this accuracy problem, without loss in its high efficiency, by employing the discrete convolutions to the multipole-to-local expansion translation operator. In this case, the potentials gradients (up to the p order) are evaluated at the cells centers, and the potentials at other desired locations are obtained by using the local expansion.

It is remarked here that FFTM differs from the FMM as it forgoes the complicated hierarchical procedure in the FMM, which is the primary feature that provides the efficiency of the FMM algorithm. Instead, the speed up in FFTM comes from the use of FFT algorithms to evaluate the numerous discrete

convolutions. On the other hand, this approach differs from the other FFT-based methods [6,8,17–19,23] as the charges and potential distributions are approximated by multipole moments and local coefficients, respectively, instead of grid charges and grid potentials as in precorrected-FFT method [23]. This choice of charge sources and potential fields representations makes the "local correction" procedure extremely simple and does not incur extra computational cost, which may otherwise be quite expensive (see for example [23]). This point will be illustrated more clearly in Section 3.1C. The current method differs from the FFT approach proposed by Elliott and Board [9] in that the convolution variables in their method were the indices of the translation operators, that is j, k, n, m of (10), whereas in our method, they are the spatial coordinates of the source and field points, that is (ρ, α, β) in (10).

The present FFTM is similar to the approach implemented by Shimada et al. [24,25] for the biomolecular simulations. They called it the particle–particle and particle–mesh/multipole expansion (PPPM/MPE) method. In the paper [25], they compared the PPPM/MPE and the FMM, and suggested that the FMM was a more favorable approach then, despite that PPPM/MPE showed better efficiency than FMM in their numerical examples. One of the primary reasons was due the excessive memory requirements for storing the numerous FFT terms of the response functions, which were enlarged by eight times due to the periodic requirements by FFT. However in FFTM, we resolved this memory storage issue partially by exploiting the symmetry relations of the spherical harmonics functions. By doing so, we eliminated the need to increase the size of the response functions by eight times and hence improved the efficiency of the method in storage memory usage. Furthermore, significant improvement in accuracy gained by forgoing the hierarchical approach was not thoroughly investigated and this is addressed in this paper.

The paper is organized as follows. Section 2 summarizes the mathematical functions required for the implementation of FFTM. In Section 3, the FFTM algorithm is described in detail. Section 4 presents some numerical examples to demonstrate the accuracy and efficiency of the method, and finally, a conclusion is given in Section 5.

2. Mathematical preliminaries

In this section, we present the mathematical formulas that are required in the implementation of the FFTM algorithm. These include the multipole and local expansions, and the multipole to local expansion translation operator. Detailed discussion on these formulas can be found in [10,13,15].

2.1. Multipole expansion

Theorem 2.1. *Suppose there are N_q charges of strength q_i that are located at positions $\boldsymbol{x}_i = (\rho_i, \alpha_i, \beta_i)$ for $i = 1, \ldots, N_q$, and are bounded within a sphere S_a of radius a centred at the origin, that is $|\rho_i| < a$. Then, for any point $\boldsymbol{y} = (r, \theta, \phi) \in \boldsymbol{R}^3$ with $r > a$, the potential generated by these charges is given by*

$$\Phi(\boldsymbol{y}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} M_n^m \frac{Y_n^m(\theta, \phi)}{r^{n+1}}, \tag{2}$$

where M_n^m is the multipole moment, which is defined as

$$M_n^m = \sum_{i=1}^{N_q} q_i \rho_i^n Y_n^{-m}(\alpha_i, \beta_i), \tag{3}$$

and $Y_n^m(\theta, \phi)$ is the spherical harmonics of degree n and order m, and it is given by

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|}(\cos\theta)\,\mathrm{e}^{\mathrm{i}m\phi}, \tag{4}$$

where $P_n^m(\cos\theta)$ is the associated Legendre function of the first kind with degree n and order m, which is defined only when n is a non-negative integer, and for $-n \leqslant m \leqslant n$.

Eq. (3) is the linear operator that converts a system of charge particles that is arbitrarily distributed within a sphere centred at the origin into its multipole moments M_n^m defined at the origin. The error incurred by truncating the multipole expansion in (2) to an order of p is bounded by

$$\left| \Phi(\boldsymbol{y}) - \sum_{n=0}^{p} \sum_{m=-n}^{n} M_n^m \frac{Y_n^m(\theta,\phi)}{r^{n+1}} \right| \leqslant \left(\frac{\sum_{i=1}^{N_q} |q_i|}{r-a} \right) \left(\frac{a}{r} \right)^{p+1}. \tag{5}$$

2.2. Local expansion

Theorem 2.2. *Suppose there are N_q charges of strength q_i that are located at positions $\boldsymbol{x}_i = (\rho_i, \alpha_i, \beta_i)$ for $i = 1, \ldots, N_q$, and fall outside a sphere S_a of radius a centred at the origin, that is $|\rho_i| > a$. Then for any point $\boldsymbol{y} = (r, \theta, \phi) \in S_a$, the potential generated by these charges is given by*

$$\Phi(\boldsymbol{y}) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta,\phi) r^j, \tag{6}$$

where L_j^k is the local expansion coefficient, which is defined as

$$L_j^k = \sum_{i=1}^{N_q} q_i \frac{Y_j^{-k}(\alpha_i,\beta_i)}{\rho_i^{j+1}}. \tag{7}$$

The physical interpretations of local expansion coefficients correspond to the potential and its gradients evaluated at the origin, which are generated by the N_q charge particles outside S_a. Eq. (6) is the linear operator that converts the local expansion coefficients defined at the origin into the potential at any arbitrary point $\boldsymbol{y} = (r, \theta, \phi)$ within S_a. Truncating the local expansion in (6) to an order of p incurs an error that is bounded by

$$\left| \Phi(\boldsymbol{y}) - \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta,\phi) r^j \right| \leqslant \left(\frac{\sum_{i=1}^{N_q} |q_i|}{a-r} \right) \left(\frac{r}{a} \right)^{p+1}. \tag{8}$$

2.3. Conversion of multipole expansion into local expansion

Theorem 2.3. *Suppose there are k charges of strength q_i that fall within a sphere S_X of radius a centred at the position $\mathbf{X} = (\rho, \alpha, \beta)$, and that $\rho > (c+1)a$ with $c > 1$. Then, the corresponding multipole expansion converges inside a sphere S_O of radius a centred at the origin. For any point $\boldsymbol{y} = (r, \theta, \phi) \in S_O$, the potential generated by these charges is given by the local expansion*

$$\Phi(\boldsymbol{y}) = \sum_{j=0}^{\infty} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta,\phi) r^j, \tag{9}$$

where

$$L_j^k = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{\mathrm{i}^{|k-m|-|k|-|m|} A_n^m A_j^k Y_{j+n}^{m-k}(\alpha,\beta)}{(-1)^n A_{j+n}^{m-k} \rho^{j+n+1}} M_n^m, \tag{10}$$

with

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}} \quad \textit{and} \quad \mathrm{i} = \sqrt{-1}.$$

Eq. (10) is the linear operator that converts the multipole moments M_n^m defined at the point $\mathbf{X} = (\rho, \alpha, \beta)$ into the local expansion coefficients L_j^k defined at the origin. However, in this present approach, we are required to evaluate the potential and its potential gradients at some distant point $\mathbf{X} = (\rho, \alpha, \beta)$ due to multipole sources at the origin. This is required by the FFT algorithms which are used to compute the discrete convolutions. Now, by substituting $\alpha = \pi - \alpha'$ and $\beta = \pi + \beta'$ into $Y_{j+n}^{m-k}(\alpha, \beta)$ of (10), the following formula

$$L_j^k = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \frac{\mathrm{i}^{|k-m|-|k|-|m|} A_n^m A_j^k Y_{j+n}^{m-k}(\alpha',\beta')}{(-1)^j A_{j+n}^{m-k} \rho^{j+n+1}} M_n^m, \tag{11}$$

where the only difference between (10) and (11) is the exponent of the (-1) term. The reason for using this definition for the translation operator will be more apparent in Section 3.1. The error bound for truncation of (10) or (11) to order p is given by

$$\left| \Phi(\boldsymbol{y}) - \sum_{j=0}^{p} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta,\phi) r^j \right| \leqslant \left(\frac{\sum_{i=1}^{k} |q_i|}{ca - a} \right) \left(\frac{1}{c} \right)^{p+1}. \tag{12}$$

Remark 2.1. It is observed from the error bounds given in (5), (8) and (12) that the accuracy of the truncated multipole and local expansions improves when: (i) one uses higher-order expansion p and/or (ii) the separation distance ratio, a/r for multipole expansion, r/a for local expansion and $1/c$ for multipole to local expansion translation operator, decreases.

3. Fast Fourier transform on multipole

As mentioned in Section 1, the main difference between the present method and the FMM is that it replaces the hierarchical procedure of translating multipole to local expansions in FMM, by discrete convolutions that are evaluated rapidly using FFT algorithms. Nevertheless, some resemblance between the FFTM and the FMM can be observed.

3.1. FFTM algorithm

This algorithm requires a number of translation operators that are denoted by three-letter abbreviations. The letter notations have the following meanings: ***M***, multipole moments; ***Q***, charge; ***P***, potential; and ***2***, To. Basically, this algorithm comprises of the following four steps:

A. Discretizing the spatial domain into many smaller cells.
B. Converting the cluster of distributed charges within each cell to multipole moments.

C. Evaluating the local expansion coefficients at cell centers due to the multipole moments. This process is regarded as evaluating a series of discrete convolutions that are accelerated by FFT algorithms.
D. Computing the potentials at the charge particle locations using the local expansions, which only account for the "distant" charge contributions. The potential contributions from the "near" charges are added directly to the charge particle locations.

This process is summarized in Fig. 1. The following sections elaborate on each of the steps.

A. Spatial discretization

This step divides the problem domain into many smaller cells, and allocates the particles among them. The aim is to identify closely packed particles that can be approximated by simpler representations, such as multipole moments. It also helps to separate the "near" particles and the "distant" ones. The dimensions of the initial volume that bound the problem domain are chosen to satisfy the ratio required by FFT, which is usually in powers of two. Otherwise, dummy layers of empty cells have to be added to meet the requirement. This process is commonly known as zero padding. Nowadays, it is also possible to perform FFT on any arbitrary size of data, like using the freeware FFTW (Fastest Fourier Transform in the West), provided by Frigo and Johnson [11]. This improves the efficiency FFTM by minimizing the number of zero padding, which often increases the FFT size rapidly and unnecessarily.

Remark 3.1. FFTW is a comprehensive collection of fast C routines for computing Discrete Fourier Transform (DFT) in one or more dimensions, of both real and complex data, and more importantly of an arbitrary input size. The FFTW works most efficiently for arrays whose size can be factored into small primes, i.e., $N = 2^a 3^b 5^c$, where a, b and c are integers.

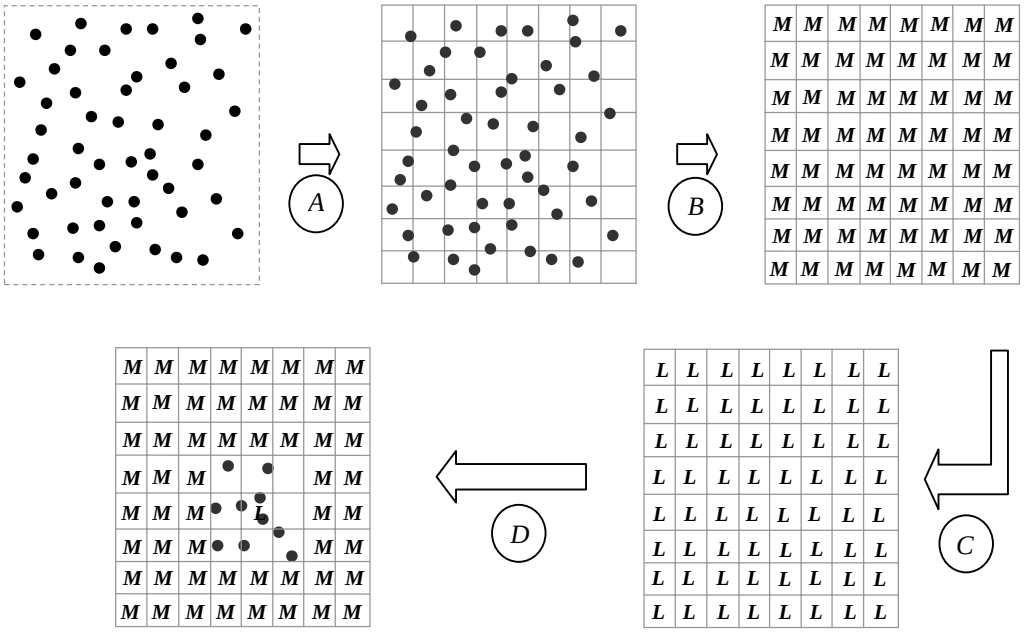

Fig. 1. Two-dimensional pictorial representation of FFTM algorithm. *Step A*: Division of problem domain into many smaller cells. *Step B*: Computation of multipole moments $\boldsymbol{M}$ for all cells. *Step C*: Evaluation of local expansion coefficients $\boldsymbol{L}$ at cell centers by discrete convolutions via FFT. *Step D*: For a given cell, compute the potentials at particles locations using $\boldsymbol{L}$, which accounts for effects of the charge particles that are considered "distant" from the cell concerned, and also adding the "near" charge contributions (particles within the neighboring cells of the cell concerned) directly onto particle locations.

B. Converting clusters of distributed charge particles into multipole moments

The operator that performs this task is given in (3), which is denoted by ***Q2M*** translation operator. It converts the clusters of distributed charge particles in the cells to an equivalent set of point sources defined at the centre of the cells. Applying ***Q2M*** to all the cells would transform the original system of charge particles to one that contains only regularly spaced point sources, as depicted in Fig. 1 after step *B*.

C. Evaluating of local expansion coefficients due to multipole moments

This step evaluates the local expansion coefficients at the field points, which coincide with the cell centers, due to the multipole moments in all the cells. The regular spacing of the cell centers enables the calculations to be done rapidly through discrete convolutions using FFT algorithms. This comes from the following discrete convolution theorem [26].

Theorem 3.1. *Suppose* $\{u_i\}$ *and* $\{v_i\}$ *are sequences of period N with discrete Fourier transforms given by* $\{U_k\}$ *and* $\{V_k\}$*, respectively. Then the discrete Fourier transform of the cyclic discrete convolution* $\{u \otimes v\}$ *is* $\{U_k V_k\}$*, where* $u \otimes v = \sum_{j=0}^{N-1} u_j v_{j-i}$ *for* $i = 0, \ldots, N-1$.

Mathematically, the multipole to local expansion translation formula, as given in (11), can be written as a series of three-dimensional discrete convolutions

$$L_j^k(x,y,z) \approx \sum_{n=0}^{p} \sum_{m=-n}^{n} \left[\sum_{x'} \sum_{y'} \sum_{z'} M_n^m(x',y',z') T_{j,n}^{m,k}(x-x', y-y', z-z') \right], \tag{13}$$

where

$$T_{j,n}^{m,k} = \frac{\mathrm{i}^{|k-m|-|k|-|m|} A_n^m A_j^k Y_{j+n}^{m-k}}{(-1)^j A_{j+n}^{m-k} \rho^{j+n+1}} \tag{14}$$

is the response function that relates the multipole moment M_n^m and the local expansion coefficients L_j^k, and the indices (x, y, z), and (x', y', z') denote the discrete locations of the field point and multipole moment, respectively. The discrete convolution in the square bracket of (13), in accordance to the discrete convolution theorem, can be obtained by taking the inverse Fourier transform of $\{\tilde{M}_n^m * \tilde{T}_{j,n}^{m,k}\}$, where $\tilde{M}_n^m$ and $\tilde{T}_{j,n}^{m,k}$ are the discrete Fourier transforms of M_n^m and $T_{j,n}^{m,k}$, respectively, and * denotes the element-wise complex multiplication operator.

Remark 3.2. Suppose the total number of cells used to discretize the spatial domain in step *A* is $N_x \times N_y \times N_z$. To include the effects of all the multipole moments M_n^m in the $N_x \times N_y \times N_z$ cells, the response functions $T_{j,n}^{m,k}$ have to be defined for the range $(-N_i, N_i)$ for $i = x, y, z$. In other words, the size of $T_{j,n}^{m,k}$ is $2N_x \times 2N_y \times 2N_z$, and the functions are evaluated in a wrap-around order. The size for M_n^m is also increased to $2N_x \times 2N_y \times 2N_z$ by zero padding [5]. Hence, the actual size of the discrete convolutions is $8 \times N_x \times N_y \times N_z$.

For a given order of expansion p, there are $O(p^2)$ local expansion coefficients, and each coefficient requires evaluating $O(p^2)$ discrete convolutions,. Hence, the total number of discrete convolutions is $O(p^4)$. It is well known that this $O(p^4)$ scaling factor has limited the practical implementation of the original FMM [12–14,20,21] for high order of accuracy applications. However, its impact is expected to be less severe on FFTM (at least up to 6-digits accuracy). This is because the FFTM requires a significantly lower order of expansion to achieve a desire order of accuracy, as compared to FMM. For example, FFTM needs only $p = 2$ to achieve 3-digits accuracy, while FMM would probably require $p = 9$ as given in [7,15]. The

superior accuracy of FFTM over FMM will be explained in Section 3.2. Furthermore, it will be shown in Section 3.3 that the number of FFTs required to evaluate the $O(p^4)$ discrete convolutions scales like $O(p^2)$ only.

Definition 3.1. Two cells are considered as neighbour if they share at least one common vertex. Hence, a cell has at most 27 nearest neighbours, including itself. We called this the first layer stencil of "near" neighbours. The second layer stencil of "near" neighbours of a given cell includes all its nearest neighbours' nearest neighbours, which gives a total of 125 cells. In general, the D layer stencil of "near" neighbours would have at most $(2D+1)^3$ neighbouring cells.

For a given cell, the effects of the multipole moments from its neighbours are often inaccurate. This is evident from the multipole expansion error bound given in (5). To overcome this problem, one needs to perform local correction, also referred to as precorrecting in the precorrect-FFT method [23]. Generally, this correction process involves: (i) removing the inaccurate contributions from the "near" multipole moments that are included when computing the discrete convolutions and (ii) replacing these erroneous results by those computed exactly using (1). In FFTM, the first part can be easily achieved by setting the response functions at its neighbouring cells to zero, that is,

$$T^{m,k}_{j,n}(x-x',y-y',z-z') = 0, \quad \text{for } |x-x'|,\ |y-y'| \text{ and } |z-z'| \leqslant 1. \tag{15}$$

Using (15) naturally excludes the "near" multipole moment effects in the discrete convolution step. In this case, the removal process is no longer necessary, since no error is incurred in the first place. The direct interaction calculation in the second part of the process is included in the next step.

D. Evaluating the potentials at particles locations

With the local expansion coefficients for all the cells, the potentials at the particle locations can be easily computed using the ***L2P*** translation operator. However, this only accounts for the "distant" charge contributions, which include all the particle effects, except those that fall within the neighbouring cells. The effects of the "near" charges are added to the particle locations by the direct interaction calculation given by (1).

3.2. Accuracy of FFTM

3.2.1. Superior accuracy of FFTM over FMM

Suppose the number of cells used to discretize the spatial domain in FFTM is identical to that in FMM at its finest level L, where $L > 2$. Then, for a given order of expansion p, FFTM is always more accurate than FMM. This simple but important observation can be illustrated as follows.

The original problem as given in (1) can be rewritten as

$$\Phi(\boldsymbol{x}_j) = \sum_{i=1,\, i\neq j}^{N_n} \frac{q_i}{\|\boldsymbol{x}_j - \boldsymbol{x}_i\|} + \sum_{k=1}^{N_d} \frac{q_k}{\|\boldsymbol{x}_j - \boldsymbol{x}_k\|} \quad \text{for } j = 1,\ldots,N, \tag{16}$$

where the potential contributions to point $\boldsymbol{x}_j$ from all the N charge particles are separated into the "near" (N_n particles) and "distant" (N_d particles) components, such that the set of particles N_n and N_d are mutually exclusive, and satisfy $N_n + N_d = N$.

Both FMM and FFTM compute the "near" charge effects in exactly the same manner and hence this part of the potential contribution is identical. As for the "distant" charge contributions, both approaches use the multipole and local expansions to approximate their effects. However, the multipole to local

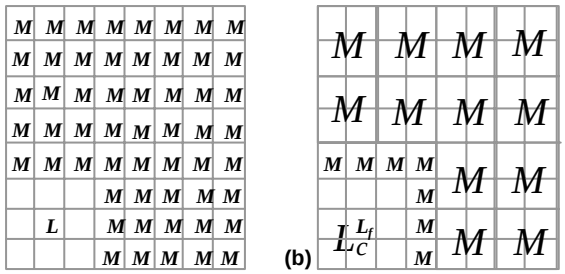

Fig. 2. Multipole moments representations for: (a) FFTM and (b) FMM.

expansion conversion is performed differently in the two methods and this significantly affects the accuracy in calculating the "distant" charge contributions.

For simplicity, consider the two-dimensional case as depicted in Fig. 2. Suppose we want to determine the local expansion coefficients of the cell L, due to all the multipole moments in other cells M, as depicted in Fig. 2(a). The "direct" approach is to compute the entire cell-to-cell interactions directly, but this is usually computationally too expensive.

FMM uses a number of translation operators to reduce this computational cost, which involves passing multipole and local expansions in a hierarchical manner. This process results in multipole moment representation that contains cells of different sizes from different levels, such as the one shown in Fig. 2(b). The local expansion coefficients L is also composed of contributions from different levels, namely the coarse level L_c and fine level L_f. One important consequence of this hierarchical process is that for a given order of expansion p, the accuracy for converting multipole to local coefficients at different levels remains approximately the same, which is primarily determined by p.

On the other hand, FFTM performs the multipole to local expansion translation by casting it as a series of discrete convolutions, as given in (13), which can be evaluated rapidly using FFT algorithms. In terms of accuracy, the FFTM is identical to the "direct" approach, since the FFT is exact (up to machine precision in computation). This means that the order of accuracy for different cells would vary, due to the differences in the relative distant between the interacting cells given by the parameter $1/c$ in (12). Hence, it is this part of the potential contribution in which the FFTM can produce significantly more accurate results than the FMM.

3.2.2. Approximation in FFTM

As described above, the accuracy of the multipole approximations improves as the relative distance between the interacting cells increases. This suggests that for a given desire accuracy ε and expansion order p^*, there exist a threshold distance c^*, such that the error incurred in the multipole approximation would be less than the desire accuracy. Using the error bound in (12), this means that

$$K\left(\frac{1}{c^*-1}\right)\left(\frac{1}{c^*}\right)^{p^*+1} < \varepsilon \tag{17}$$

where $K = (\sum_{i=1}^{k} |q_i|/a)$. In FFTM, we can only satisfy $(1/(c^*-1))(1/c^*)^{p^*+1} < \mathrm{O}(\varepsilon)$, since K is rather arbitrary. Now, we further categorize the "distant" cells into: (i) the "well-separated" cells for those cells whose relative distant are greater than c^* and (ii) the "moderately separated" which are the remaining "distant" cells. Using these new criteria of cells classification, the error E at a given evaluation potential point can be approximated as

$$E_{\text{total}} = \alpha E_{\text{near}} + \beta E_{\text{moderately separated}} + \gamma E_{\text{well separated}}$$
$$= \beta E_{\text{moderately separated}} + \gamma E_{\text{well separated}} \quad (\text{since } E_{\text{near}} = 0), \tag{18}$$

where the E's denote the average error of the different groups of cells, and α, β and γ are the corresponding weighs for the potential contribution, with $\alpha + \beta + \gamma = 1$.

In FFTM, we would want γ to be significantly larger than β, and this can be achieved by choosing the appropriate p^* value. We note that β and γ depend on two quantities, namely: (i) the strength of the multipole moments, which is strongly dependent on the relative distance r between the interacting cells and (ii) the number of cells that belong to the respective groups. Although each of the "moderately separated" cells generally has a larger potential contribution than the "well separated" ones, it is also important to note that there are usually significantly more cells in the "well separated" group than the former one. Unfortunately, it is difficult to quantify β and γ, since they depend on the p^* value, and are likely to be problem-dependent.

In contrast to the FMM, where p^* is often chosen such that $E_{\text{moderately separated}} < \mathrm{O}(\varepsilon)$, we allow the error in this component to be larger than $\mathrm{O}(\varepsilon)$ in the FFTM. To appreciate this, suppose p^* is chosen such that $\beta = \tau\gamma$ in (18), i.e.,

$$\gamma\left(\tau E_{\text{moderately separated}} + E_{\text{well separated}}\right) < \mathrm{O}(\varepsilon) \Rightarrow E_{\text{moderately separated}} < \frac{1}{\tau}\mathrm{O}(\varepsilon) \quad \text{since } E_{\text{well separated}} < \mathrm{O}(\varepsilon). \tag{19}$$

Eq. (19) suggests that we can allow an error of order $(1/\tau)\mathrm{O}(\varepsilon)$ in $E_{\text{moderately separated}}$, and yet maintain an overall accuracy of $\mathrm{O}(\varepsilon)$ in the computed potential. In Section 4, it is shown heuristically that the p^* value for FFTM can be significant lower that required by FMM.

3.3. Algorithmic complexity analysis

This section gives an estimate of the computational time complexity of FFTM. There are two major parts: (i) the cost to compute the "near" charge contributions directly via particle-to-particle interaction, C_{near} and (ii) the cost to compute the "distant" charge contribution, which is primarily dominated by the cost of evaluating the discrete convolutions, C_{distant}. For reasonably uniform distributions of charge particles, the cost can be approximated as

$$C_{\text{near}} = 27Ns \quad \text{and} \quad C_{\text{distant}} = 2Np^2 + \mathrm{O}(p^4)\left[3\left(8\frac{N}{s}\right)\log\left(8\frac{N}{s}\right) + \left(8\frac{N}{s}\right)\right], \tag{20}$$

where s is the average number of particles per cell. In C_{near}, the constant of 27 corresponds to the maximum possible number of neighbours for a given cell when the first layer stencil is used to define the "near" cells. The first term of C_{distant} is due to the formation of multipole moments (via ***Q2M***), and evaluation of the local expansions (via ***L2P***) for all the cells, and the second term is the cost of evaluating the numerous discrete convolutions, as given in (13). The terms in the square bracket is the cost for evaluating one discrete convolution, which includes the Fourier transforms of M_n^m and $T_{j,n}^{m,k}$, the element-wise complex multiplication of these transforms $\tilde{M}_n^m$ and $\tilde{T}_{j,n}^{m,k}$, and finally an inverse FFT of $\{\tilde{M}_n^m * \tilde{T}_{j,n}^{m,k}\}$. The constant factor of 8 is due to the zero-padding required to eliminate the aliasing effects (see Remark 3.2).

However, the actual implementation of FFTM requires evaluating significantly lesser number of FFTs. This is due to the sharing of the multipole moments M_n^m and response functions $T_{j,n}^{m,k}$. This point is more

$$
\underbrace{\begin{Bmatrix} 0,0 \\ 0,1 \\ 1,1 \\ 0,2 \\ 1,2 \\ 2,2 \end{Bmatrix}}_{\boldsymbol{L}(k,\,j)} = \overbrace{\begin{bmatrix} 0,0 & -1,1 & 0,1 & 1,1 & -2,2 & -1,2 & 0,2 & 1,2 & 2,2 \\ 0,1 & -1,2 & 0,2 & 1,2 & -2,3 & -1,3 & 0,3 & 1,3 & 2,3 \\ -1,1 & -2,2 & -1,2 & 0,2 & -3,3 & -2,3 & -1,3 & 0,3 & 1,3 \\ 0,2 & -1,3 & 0,3 & 1,3 & -2,4 & -1,4 & 0,4 & 1,4 & 2,4 \\ -1,2 & -2,3 & -1,3 & 0,3 & -3,4 & -2,4 & -1,4 & 0,4 & 1,4 \\ -2,2 & -3,3 & -2,3 & -1,3 & -4,4 & -3,4 & -2,4 & -1,4 & 0,4 \end{bmatrix}}^{\boldsymbol{T}(m-k,\,j+n)} \underbrace{\begin{Bmatrix} 0,0 \\ -1,1 \\ 0,1 \\ 1,1 \\ -2,2 \\ -1,2 \\ 0,2 \\ 1,2 \\ 2,2 \end{Bmatrix}}_{\boldsymbol{M}(m,\,n)}
$$

Fig. 3. Discrete convolution matrix for second-order expansion, $p = 2$.

clearly illustrated in Fig. 3, which shows the discrete convolution matrix for a second-order expansion (see (13)).

First, the local expansion coefficients are evaluated only for the non-negative values of the index k, because the coefficients for the negative indices are the complex conjugates of the corresponding positive ones, i.e., $L_j^{-k} = (L_j^k)^*$. This means that the total number of discrete convolutions is reduced by approximately half. Then, it is noted that there are only $O(p^2)$ and $O(4p^2)$ distinct terms for M_n^m and T_{j+n}^{m-k}, respectively. These functions can be evaluated once and used repeatedly in the various discrete convolutions as required. Finally, due to the linearity property of the Fourier transform, the number of inverse FFT required to derive the local expansion coefficients is also $O(p^2)$. Hence, a more reasonable estimate for the cost of evaluating the discrete convolutions would be

$$O(6p^2)\left[\left(8\frac{N}{s}\right)\log\left(8\frac{N}{s}\right)\right] + O(p^4)\left(8\frac{N}{s}\right), \tag{21}$$

where the first term corresponds to the cost of evaluating the FFT and the second term is that due to the complex multiplication.

Remark 3.3. The cost of evaluating the FFT of $T_{j,n}^{m,k}$ can be reduced significantly by exploiting the symmetric/anti-symmetric properties of the functions. The speedup comes from the use of fast sine/cosine transform with anti-symmetric/symmetric condition [26]. More importantly, the memory requirement storing these functions is significantly reduced, since we are required to store the response functions at the first quadrant only.

4. Numerical examples

In this section, some numerical examples are used to study the performance of FFTM, in terms of its accuracy and efficiency. Two examples are considered here, namely: (i) a system of charge particles that are distributed randomly but uniformly within a unit cube and (ii) a system of charge particles that are distributed randomly on the surface of a sphere of unit radius. The particles are randomly given charges that range from −0.5 to 0.5 (but not zero). These two examples are also used in [7]. For convenience, they are referred to as the cube and sphere examples, hereafter. Note that the cube problem represents a homogenous problem, while the sphere problem is a sparse one since a large part of the domain does not contain any particle. The numerical experiments are carried out using a SGI Octane workstation with a CPU clock rate of 175 MHz and memory storage of 512 MB of RAM.

4.1. Accuracy analysis of FFTM

Definition 4.1. In this study, the error in the FFTM approximation is defined in the L_2 norm as

$$\text{Error} = \left(\frac{\sum_{i=1}^{k} |\Phi_{\text{Dir}}(x_i) - \Phi_{\text{FFTM}}(x_i)|^2}{\sum_{i=1}^{k} |\Phi_{\text{Dir}}(x_i)|^2} \right)^{1/2}, \tag{22}$$

where $\Phi_{\text{Dir}}(x_i)$ and $\Phi_{\text{FFTM}}(x_i)$ denote the potentials computed using the direct calculation approach and the FFTM, respectively. Note that only the first 100 particles potentials are evaluated, i.e., $k = 100$ in (22), due to the excessive computational times required by the direct approach. The CPU times for the direct approach are linearly extrapolated from this set of computations. All the calculations in this accuracy analysis were performed in double precision.

As shown in Section 3.2.1, for a given order of expansion p and with identical cell discretization, FFTM is more accurate than FMM. The following discussion aims to quantify these differences, in terms of their error convergence behaviours with increasing p. However, we do not have access to FMM algorithms, and hence its error convergence behaviours are approximated based on the error bound in (12). According to (12), the error incurred in using ***M2L*** scales like $O(1/c)^{p+1}$, and for FMM, $1/c$ is bounded by (0.2, 0.764). This bound is obtained using the formula, $c = (\rho/a) - 1$, for the nearest and furthest interacting cells. Now, consider the situation where all the 189 interactions cells have the average value of $1/c = 0.5 * (0.2 + 0.764) = 0.482$. Hence, the error of FMM can be realistically be approximated by $O(0.482^{p+1})$, which is denoted by FMM (estimate) in Figs. 4 and 5. Although this curve may not correspond to the actual convergence of the FMM, where its placement in the figures can be problem-dependent, it does predict the order of accuracy for the two examples quite accurately (see [7,15]). In Figs. 4 and 5, we

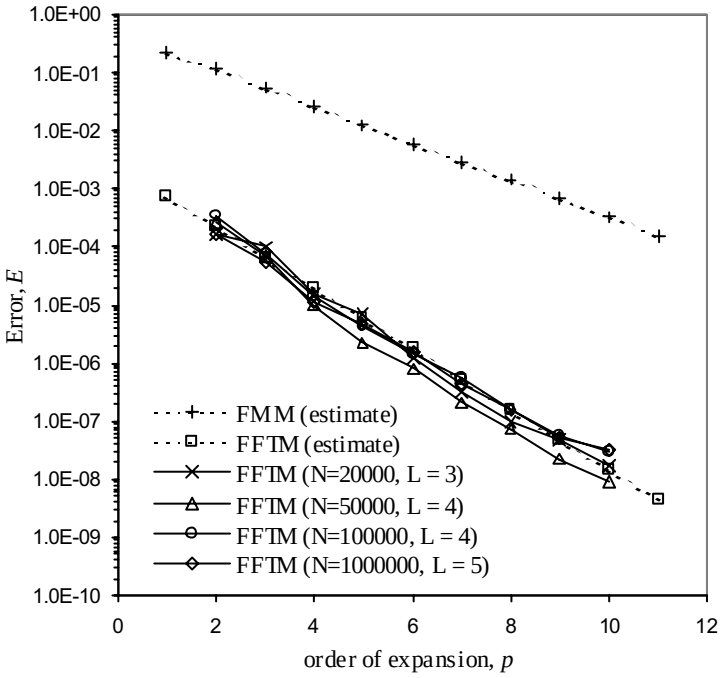

Fig. 4. Convergence behavior of FFTM for cube example.

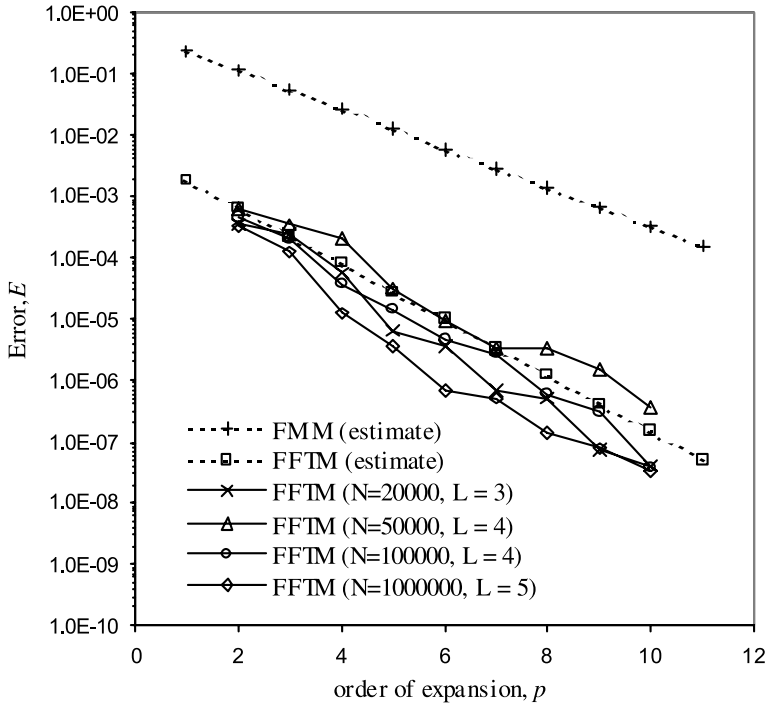

Fig. 5. Convergence behavior of FFTM for sphere example.

plot the errors of the FFTM for the cube and sphere examples, respectively, for different number of particles N and cell discretization level L.

From Fig. 4, it is noted that all the FFTM curves behave in a similar manners, regardless of the problem size N, and the level of discretization L. This indicates that the accuracy of FFTM for a generally homogenous problem, like FMM, is primarily determined by the order of expansion p. But, this is not entirely true for the sphere example, where the accuracy for some of the cases can differ by more than one order of magnitude for the same value of p. This behaviour is likely due to the sparse nature of the sphere example. However, a more important observation here is that the FFTM curves converge more rapidly than the FMM (estimate), thought less significant for the sphere example. Based on the gradients of the FFTM curves in Figs. 4 and 5, we estimate the error of FFTM for homogenous and sparse problems to scale like $O(0.3^{p+5})$ and $O(0.35^{p+5})$, respectively, which are denoted by FFTM (estimate) in the figures. However, we would like to emphasis that these error estimates do not necessarily represent all problems. The convergence rate of the FFTM is more likely to be problem's characteristics, such as the sparseness of the charge particles.

4.2. Efficiency analysis of FFTM

In this study, we aim to establish the computational complexity of FFTM for 3- and 6-digit accuracy. The calculations for the 3- and 6-digit accuracy were performed in single and double precision, respectively. The cube and sphere examples were solved for problem sizes ranging from 20,000 to 1,000,000. The CPU times (in *seconds*) for the cube example are tabulated in Tables 1 and 2, and that for the sphere example in Tables 3 and 4. These tables are organized as follows:

Table 1
CPU times for cube example for 3-digit accuracy

N	$N_x \times N_y \times N_z$	P	T_{FFTM}	T_{DIR}	Error
20,000	$16 \times 16 \times 16$	2	1.47	106	1.59E − 4
50,000	$20 \times 20 \times 20$	2	4.01	720	3.92E − 4
100,000	$24 \times 24 \times 24$	2	8.63	3110	4.18E − 4
200,000	$30 \times 30 \times 30$	2	18.0	12,460	9.85E − 4
500,000	$48 \times 48 \times 48$	2	47.5	77,900	4.68E − 4
1,000,000	$54 \times 54 \times 54$	2	99.2	311,500	2.20E − 4

Table 2
CPU times for cube example for 6-digit accuracy

N	$N_x \times N_y \times N_z$	P	T_{FFTM}	T_{DIR}	Error
20,000	8×8×8	8	8.45	112	9.01E − 08
50,000	12×12×12	8	22.1	885	1.99E − 07
100,000	16×16×16	8	48.8	3510	1.53E − 07
200,000	20×20×20	8	106	14,140	8.36E − 07
500,000	30×30×30	8	305	87,700	2.97E − 07
1,000,000	32×32×32	8	562	350,800	1.59E − 07

Table 3
CPU times for sphere example for 3-digit accuracy

N	$N_x \times N_y \times N_z$	p	T_{FFTM}	T_{DIR}	Error	Sparcity
20,000	$20 \times 20 \times 20$	2	3.88	106	7.30E − 04	10.2
50,000	$30 \times 30 \times 30$	2	12.3	720	1.50 − 04	7.16
100,000	$40 \times 40 \times 40$	2	31.7	3110	2.01E − 04	5.31
200,000	$54 \times 54 \times 54$	2	75.0	12,460	6.36E − 04	4.05
500,000	$80 \times 80 \times 80$	2	258	77,900	2.54E − 04	2.70

Table 4
CPU times for sphere example for 6-digit accuracy

N	$N_x \times N_y \times N_z$	p	T_{FFTM}	T_{DIR}	Error	Sparcity
20,000	$10 \times 10 \times 10$	8	15.4	112	4.49E − 07	24.2
50,000	$12 \times 12 \times 12$	8	52.7	885	3.35E − 07	18.5
100,000	$16 \times 16 \times 16$	8	132	3510	2.99E − 07	14.2
200,000	$24 \times 24 \times 24$	8	310	14,140	1.50E − 07	9.20
500,000	$32 \times 32 \times 32$	8	1080	87,700	5.86E − 07	7.11

Column 1: Number of particles in the simulations, N.
Column 2: Number of cells used to discretize the spatial domain, $N_x \times N_y \times N_z$.
Column 3: Order of expansion used in the example, p.
Column 4: CPU times for FFTM in seconds, T_{FFTM}.
Column 5: CPU times for Direct approach in seconds, T_{DIR}.
Column 6: Errors in the computed potentials for FFTM using (22), Error.
Column 7: Sparcity of the cell discretization, *Sparcity*. It measures the percentage of the non-empty cells to that defined in Column 2. This parameter only applies to the sphere example.

The following observations can be seen from these tables. First, it is noted that the number of cells used to discretize the spatial domain, i.e., $N_x \times N_y \times N_z$, increases with the problem sizes for a given order of accuracy. This observation can be explained as follows.

The estimated complexity of FFTM in (20) indicates that the two cost components, namely C_{near} and C_{distant}, have an inverse relationship via the parameter s, where s is the average number of particles per cell. Reducing s leads to lesser number of particles in the direct interaction lists, causing a decrease in C_{near}. But this can only be achieved by increasing the number of cells, which inevitably increases the discrete convolution size, and hence C_{distant}.

In general, as the problem size increases, it is natural to increase the number of cells in order to reduce C_{near} but at the same time ensure that C_{distant} is not significantly increased. Hence, there exists an optimum cell discretization in which FFTM is most efficient, but this is likely to be problem-dependent.

It is also noted that only C_{distant} is dependent on the order of expansion p. As shown in Section 4.1, the accuracy of FFTM is largely determined by the order of expansion p. This means that for higher-order accuracy, C_{distant} is likely to be the dominant component, which can only be kept low by using lesser number of cells. Hence, the number of cells used for the 6-digit accuracy is always lesser than those used for the 3-digit accuracy.

Fig. 6 gives a plot of the computation time against the problem size. Based on the gradients of the curves in Fig. 6, the estimated computational complexities for the cube and sphere examples are $O(N^{1.09})$ and $O(N^{1.31})$, respectively.

From these two examples, it is seen that FFTM is less efficient for the sphere example due to the sparseness of the problem. This is a common problem faced by the FFT-based approaches [6,8,17–19,23]. However in FFTM, there is a simple technique that can be used to further reduce its computational cost.

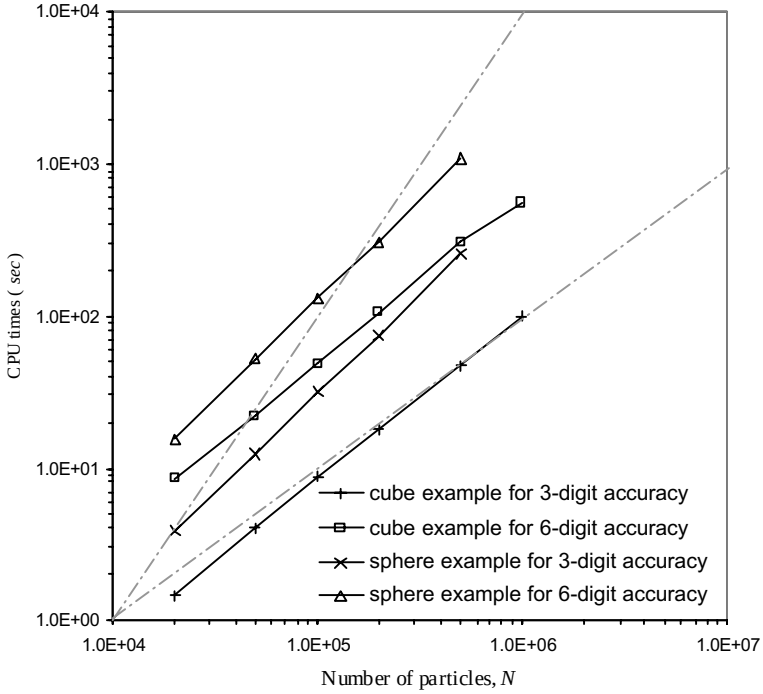

Fig. 6. Computational complexities of FFTM for cube and sphere examples.

4.3. Truncating the influence region of the response functions

This technique relies on the fact that the size of the discrete convolutions depends on the number of cells used to discretize the spatial domain $N_x \times N_y \times N_z$ and the extent of influence of the response functions T_{j+n}^{m-k}. Suppose the extent of influence of the response functions is truncated to $R_x \times R_y \times R_z$, where $R_i \leqslant N_i$ for $i = x, y$ and z, then the size of the discrete convolutions is $(R_x + N_x) \times (R_y + N_y) \times (R_z + N_z)$. Note that for all the examples done above, $R_i = N_i$, which assumes that all the response function influences are significant throughout the entire spatial domain. This results in an $8N_x \times N_y \times N_z$ size of discrete convolutions (see Remark 3.2). However, if we reduce R_i to $0.5N_i$, then the size of the discrete convolution becomes $3.375N_x \times N_y \times N_z$, which means a reduction of the computational cost for evaluating the discrete convolution by about half.

Now, truncating the influence region of the response functions is possible because the accuracy of the multipole approximation depends on the separation distant between the interacting cells, which scales like $O((1/c)^{p+1})$. For example, the FFTM requires a second-order expansion to achieve a 3-digit accuracy (Error $< 1.0E-3$). However, interacting cells that are separated with $c = 31.6$ can readily achieve that order of accuracy with only the first-order expansion. In other words, the effects of the second-order multipole moments from these "well separated" cells are expected to be lesser than the desire order of accuracy, and hence can reasonably be neglected.

Using this truncating technique, we recalculated some of the larger cube and sphere problems. The results for the cube and sphere examples are tabulated in Tables 5 and 6, respectively. Columns 1, 3 and 4 represent the same quantities as in Tables 1 and 4. Column 2 defines the size of the numerous discrete convolutions and the limits, p_1 and p_2, of the orders of expansions. Basically, it means that for $p \leqslant p_1$, the response functions were not truncated, i.e., $R_i = N_i$, whereas for $p_1 < p \leqslant p_2$ the response functions were truncated to $R_x \times R_y \times R_z$. Column 5 gives the speedup obtained by using this simple truncation technique over the naïve approach (no truncation), that is, Speedup $= T_{\text{original}}/T_{\text{truncated}}$, where $T_{\text{truncated}}$ is the CPU time using the truncated scheme and T_{original} is the corresponding CPU time in Tables 1–4. It is observed that this truncation technique can give significant speedup for the high accuracy problems. Here, we also solve the larger problems for 9-digits accuracy, and the CPU times are given in Table 7. In this case, the "near" cells

Table 5
CPU times for cube example using the truncating technique

N	$N_x \times N_y \times N_z(p_1),\ R_x \times R_y \times R_z(p_2)$	T_{FFTM}	Error	Speedup
500,000	$48 \times 48 \times 48$ (1), $16 \times 16 \times 16$ (2)	41.0	6.52E − 04	1.16
1,000,000	$60 \times 60 \times 60$ (1), $20 \times 20 \times 20$ (2)	84.6	3.90E − 04	1.17
500,000	$30 \times 30 \times 30$ (4), $10 \times 10 \times 10$ (8)	172	4.03E − 07	1.77
1,000,000	$32 \times 32 \times 32$ (4), $8 \times 8 \times 8$ (8)	357	2.58E − 07	1.57

Table 6
CPU times for sphere example using the truncating technique

N	$N_x \times N_y \times N_z(p_1),\ R_x \times R_y \times R_z(p_2)$	T_{FFTM}	Error	Speedup
200,000	$54 \times 54 \times 54$ (1), $18 \times 18 \times 18$ (2)	66.9	7.72E − 04	1.12
500,000	$80 \times 80 \times 80$ (1), $20 \times 20 \times 20$ (2)	209	4.07E − 04	1.23
200,000	$30 \times 30 \times 30$ (4), $6 \times 6 \times 6$ (8)	176	4.27E − 07	1.76
500,000	$40 \times 40 \times 40$ (4), $8 \times 8 \times 8$ (8)	681	6.52E − 07	1.59

Table 7
CPU times for 9-digit accuracy using the truncating technique

Example (N)	$N_x \times N_y \times N_z(p_1),\ R_x \times R_y \times R_z(p_2)$	T_{FFTM}	Error
Cube (500,000)	$36 \times 36 \times 36$ (6), $12 \times 12 \times 12$ (9)	507	5.43E − 10
Cube (1,000,000)	$40 \times 40 \times 40$ (6), $8 \times 8 \times 8$ (9)	1040	2.89E − 10
Sphere (200,000)	$32 \times 32 \times 32$ (6), $8 \times 8 \times 8$ (9)	488	8.58E − 10
Sphere (500,000)	$40 \times 40 \times 40$ (6), $8 \times 8 \times 8$ (9)	1740	1.77E − 10

are defined by the second layer stencil (see Definition 3.1). The CPU times are found to be about three times those of 6-digit accuracy in Tables 5 and 6.

5. Conclusion

In this paper, we present an alternate fast algorithm for rapid evaluation of potential fields in three dimensions. We refer it to as the Fast Fourier Transform on Multipoles (FFTM) method. The speedup in the algorithm is achieved by recognizing the discrete convolution form of the multipole to local expansion translation operator, which can be rapidly evaluated using FFT algorithms.

It is demonstrated that FFTM is an accurate method. More importantly, it can be significantly more accurate than FMM for the same order of expansion. This algorithm is exceptionally efficient for low order of accuracy (3-digit accuracy) and for problems where the particle distribution is relatively homogenous. Based on the numerical simulations, the algorithm is found to have computational complexity of $O(N^a)$, where a ranges from 1.0 to 1.3. We would like to emphasis that FFTM is simpler to implement than the FMM, especially when it is compared to the new version of FMM [7,15].

One possible and useful extension of this algorithm is to apply it to solve the Helmholtz equation, commonly encountered in computational acoustics. The corresponding multipole and local expansions are well established, and the necessary translation operators are readily available. Furthermore, Gumerov and Duraiswami [16] have recently derived recursive formulae for the computations of various translation operators for the Helmholtz equation so that the use of complicated formula involving the Wigner-3j symbols can be avoided. However, direct application of this method may only be suited for solving such problems in the low to medium frequency regime. Very high-order expansions are needed for high frequency problems and this may defeat the effectiveness of the FFTM.

Finally, it is noted that FFTM involves evaluating numerous discrete convolutions that are essentially uncoupled. This gives a great potential for simple parallel implementation of the algorithm that can further reduce the computation time for large-scale problems.

References

[1] A.W. Appel, An efficient program for many-body simulations, SIAM J. Sci. Stat. Comput. 6 (1985) 85.
[2] C.R. Anderson, A method of local corrections for computing the velocity field due to a distribution of vortex blobs, J. Comput. Phys. 62 (1986) 111.
[3] J. Barnes, P. Hut, A hierarchical $O(N \log N)$ force calculation algorithm, Nature 324 (1986) 446.
[4] J.A. Board, J.W. Causey, J.F. Leathrum, A. Windermuth, K. Schulten, Accelerated molecular dynamics simulations with the parallel fast multipole method, Chem. Phys. Lett. 198 (1992) 89.
[5] E.O. Brigham, The Fast Fourier Transform and its Applications, Prentice-Hall, Englewood Cliffs, 1988.
[6] O.P. Bruno, L.A. Kunyansky, A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, tests and applications, J. Comput. Phys. 169 (2001) 80.

[7] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, J. Comput. Phys 155 (1999) 468.
[8] T. Darden, D. York, L. Pedersen, Particle mesh Ewald: An $N\log(N)$ method for Ewald sums in large systems, J. Chem. Phys. 98 (1993) 10089.
[9] W.D. Elliott, J.A. Board Jr., Fast Fourier transform accelerated fast multipole algorithm, SIAM J. Sci. Comput. 17 (1996) 398.
[10] M.A. Epton, B. Dembart, Multipole translation theory for the three-dimensional Laplace and Helmholtz equations, SIAM J. Sci. Comput. 16 (1995) 865.
[11] M. Frigo, S.G. Johnson, FFTW, C subroutines library for computing Discrete Fourier Transform (DFT). Freeware can be downloaded from http://www.fftw.org.
[12] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput Phys. 73 (1987) 325.
[13] L. Greengard, V. Rokhlin, The rapid evaluation of potential fields in three dimensions, in: C. Anderson, C. Greengard (Eds.), Vortex Methods, Lecture Notes in Mathematics, vol. 1360, Springer-Verlag, Berlin, 1988, p. 121.
[14] L. Greengard, V. Rokhlin, On the evaluation of electrostatics interactions in molecular modeling, Chem. Scripta 29A (1989) 139.
[15] L. Greengard, V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, Acta Numerica 6 (1997) 229.
[16] N.A. Gumerov, R. Duraiswami, Fast, exact, and stable computation of multipole translation and rotation coefficients for the 3D Helmholtz equation, Instititute for Advance Computer Studies, University of Maryland, Technical Report UMIACS-TR#2001-44, 2001.
[17] R.W. Hockey, J.W. Eastwood, Computer Simulation using Particles, McGraw-Hill, New York, 1981.
[18] B.A. Luty, M.E. Davis, I.G. Tironi, W.F. van Gunsteren, A comparsion of particle–particle–particle–mesh, and Ewald methods for calculating electrostatics interactions in periodic molecular systems, Mol. Simul. 14 (1994) 11.
[19] B.A. Luty, W.F. van Gunsteren, Calculating electrostatic interactions using particle–particle–particle–mesh method with nonperiodic long-range interactions, J. Phys. Chem. 100 (1996) 2581.
[20] K. Nabors, J. White, Fastcap: a multipole accelerated 3-D capacitance extraction program, IEEE Trans. CAD Integrated Circuits Systems 11 (1991) 1447.
[21] K. Nabors, F.T. Korsmeyer, F.T. Leighton, J. White, Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory, SIAM J. Sci. Stat. Comput. 15 (1994) 713.
[22] E.T. Ong, K.H. Lee, K.M. Lim, A Fast Algorithm for three-dimensional electrostatic analysis: fast Fourier transform on multipole (FFTM) (submitted for publication).
[23] J.R. Phillips, J. White, A precorrected-FFT method for electrostatic analysis of complicated 3-D structures, IEEE Trans. CAD Integrated Circuits Systems 16 (1997) 1059.
[24] J. Shimada, H. Kaneko, T. Takada, Efficient calculations of Coulomic interactions in biomolecular simulations with periodic boundary conditions, J. Comput. Chem. 14 (1993) 867.
[25] J. Shimada, H. Kaneko, T. Takada, Performance of fast multipole methods for calculating electrostatic interactions in biomacromolecular simulations, J. Comput. Chem. 15 (1994) 28.
[26] J.S. Walker, Fast Fourier Transforms, CRC Press, Boca Raton, 1991.