Journal of Computational Physics 371 (2018) 122–136

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Improve the efficiency of the Cartesian tensor based fast multipole method for Coulomb interaction using the traces

He Huang [a], Li-Shi Luo [a], Rui Li [b], Jie Chen [b], He Zhang [b,*]

[a] Old Dominion University, Norfolk, VA 23529, USA
[b] Jefferson Lab, Newport News, VA 23606, USA

ARTICLE INFO

Article history:
Received 21 June 2017
Received in revised form 10 May 2018
Accepted 16 May 2018
Available online 17 May 2018

Keywords:
Fast multipole method
Cartesian tensor
Coulomb interaction

ABSTRACT

To compute the non-oscillating mutual interaction for a system with N points, the fast multipole method (FMM) has an efficiency that scales linearly with the number of points. Specifically, for Coulomb interaction, FMM can be constructed using either the spherical harmonic functions or the totally symmetric Cartesian tensors. In this paper, we will present that the efficiency of the Cartesian tensor based FMM for the Coulomb interaction can be significantly improved by implementing the traces of the Cartesian tensors in calculation to reduce the independent elements of the n-th rank totally symmetric Cartesian tensor from $(n+1)(n+2)/2$ to $2n+1$. The computation complexity for the operations in FMM are analyzed and expressed as polynomials of the highest rank of the Cartesian tensors. For most operations, the complexity is reduced by one order. Numerical examples regarding the convergence and the efficiency of the new algorithm are demonstrated. A reduction of computation time up to 50% has been observed for moderate number of points and rank of tensors.

1. Introduction

The CPU cost of computing mutual interactions between N points distributed in $\mathbb{R}^3$ scales as $O(N^2)$ and it has engendered the need for computational methodologies that are efficient both in terms of memory and CPU time. Among those methodologies, the tree code [1–3] provides an efficiency of $O(N \log N)$ and the fast multipole method (FMM) [4,5] provides an efficiency of $O(N)$. Both algorithms are based on subdividing the computational domain into hierarchical subdomains, and computing the influence between subdomains that are sufficiently separated using multipole/local expansions. The FMM has revolutionized the analysis in fields ranging from astrophysics to biophysics, engineering sciences, etc. Originally, the FMM is invented to calculate the Coulomb interaction between charged particles, where the Coulomb potential of grouped particles is expressed as multipole expansions and local expansions using the spherical harmonic functions [4]. Soon the FMM is extended to treat other interactions and new methodologies are applied to construct the expansions [6,7,5,8–10]. Later on, using the mathematical techniques that allow to construct the multipole expansions and local expansions without using the explicit expression of the kernel function, which describe the interaction between the points, the FMM is extended to more general cases. The kernel-independent FMM [11] or the black box FMM [12] can automatically calculate a wide group of user defined kernel functions, including those have no explicit expressions and can only be numerically calculated.

* Corresponding author.
E-mail address: hezhang@jlab.org (H. Zhang).

https://doi.org/10.1016/j.jcp.2018.05.028
0021-9991/

Coulomb interaction is one of the most important interaction in many physical processes. Besides using spherical harmonic functions, the FMM for Coulomb interaction can also be constructed using Cartesian tensors. In [13] the authors provide the FMM for potentials in $R^{-\nu}$ form with any real number ν using Taylor expansions in the format of totally symmetric Cartesian tensors. The symmetry reduces the independent element number in an n-th rank tensor from 3^n to $(n+1)(n+2)/2$. They also derive the formulas specifically for R^{-1} interaction, where the detracer operator is used to reduce a totally symmetric Cartesian tensor into a traceless one with only $2n+1$ independent elements. However, the detracer operator is computationally expensive itself, and so far was we know, the algorithm using the detracer operator has not been implemented in codes. The work we will present is an extension of the work in [13]. We will show that it is not necessary to construct new formulas using the detracer operator. The traces of the totally symmetric Cartesian tensor of the Coulomb potential can be used directly in calculation to improve the efficiency.

2. Preliminaries

This section builds upon some of the earlier work [14–16,13]. We will describe the basic notations, theorems and lemmas, which will be used in the rest of this paper, as well as the necessary proofs.

2.1. Tensors and trace

A Cartesian tensor of rank n, denoted by $\mathbf{A}^{(n)}$ with its component $A^{(n)}_{\alpha_1\cdots\alpha_n}$ in the component notation with $\alpha_i \in \{1,2,3\}$, has 3^n components. The tensor is said to be totally symmetric if $A^{(n)}_{\alpha_1\cdots\alpha_n}$ is invariant under any permutation of sequence $\alpha_1\cdots\alpha_n$. A totally symmetric tensor contains only $(n+1)(n+2)/2$ independent components. In a compressed form, totally symmetric tensor is denoted by $A^{(n)}(n_1,n_2,n_3)$ where $n_1+n_2+n_3=n$, and n_i is the number of times the index i is repeated. In this paper, the direct product of the vectors $\underbrace{\mathbf{r}\mathbf{r}\cdots\mathbf{r}}_{n}$ is defined as $\mathbf{r}^n := \underbrace{\mathbf{r}\mathbf{r}\cdots\mathbf{r}}_{n}$ which forms a tensor $\mathbf{r}^n$ of rank n, whose component $r^{(n)}(n_1,n_2,n_3) = x^{n_1}y^{n_2}z^{n_3}$. If no specify, all tensors mentioned in this paper are totally symmetric and described in Cartesian coordinate.

The trace of $\mathbf{A}^{(n)}$ in one index pair is denoted by $A^{(n:1)}_{\alpha_3\cdots\alpha_n} = A^{(n)}_{\nu\nu\alpha_3\cdots\alpha_n}$ in Einstein notation of summation, which results in a tensor of rank $n-2$. If the trace is zero for any given index, the tensor is totally traceless. The trace of a totally symmetric tensor $\mathbf{A}^{(n)}$ can be written in the compressed tensor notation as

$$A^{(n:1)}(n_1,n_2,n_3) = A^{(n)}(n_1{+}2,n_2,n_3) + A^{(n)}(n_1,n_2{+}2,n_3) + A^{(n)}(n_1,n_2,n_3{+}2), \tag{1}$$

where $n_1+n_2+n_3=n-2$. If the trace of a totally symmetric tensor for one index vanishes, the trace for any index vanishes. Such a tensor is called traceless totally symmetric tensor, and it contains only $2n+1$ independent components. Obviously for a traceless totally symmetric tensor, we have

$$A^{(n)}(n_1{+}2,n_2,n_3) + A^{(n)}(n_1,n_2{+}2,n_3) + A^{(n)}(n_1,n_2,n_3{+}2) = 0. \tag{2}$$

Generally the m-fold trace of a totally symmetric tensor is also a totally symmetric tensor of rank $n-2m$ and is defined as $A^{(n:m)}_{\alpha_{2m+1}\cdots\alpha_n} = A^{(n)}_{\nu_1\nu_1\nu_2\nu_2\cdots\nu_m\nu_m\alpha_{2m+1}\cdots\alpha_n}$, or in the compressed notation as

$$A^{(n:m)}(n_1,n_2,n_3) = \sum_{k_1!k_2!k_3!} \frac{m!}{k_1!k_2!k_3!} A^{(n)}(n_1{+}2k_1,n_2{+}2k_2,n_3{+}2k_3), \tag{3}$$

where $n_1+n_2+n_3=n-2m$ and the sum satisfies $k_1+k_2+k_3=m$.

2.2. Tensor contraction

Considering two tensors $\mathbf{A}^{(m+n)}$ and $\mathbf{B}^{(n)}$, the n-fold contraction between these two tensors is given by $\mathbf{A}^{(m+n)}_{\beta_1\cdots\beta_m\alpha_1\cdots\alpha_n}\mathbf{B}^{(n)}_{\alpha_n\cdots\alpha_1}$ and will be denoted as $\mathbf{C}^{(m)} = \mathbf{A}^{(m+n)} : \mathbf{B}^{(n)}$. If $\mathbf{A}^{(n+m)}$ and $\mathbf{B}^{(n)}$ are two totally symmetric tensors, then the n-fold contraction between them can be written in the compressed notation as

$$C^{(m)}(m_1,m_2,m_3) = \sum_{n_1,n_2,n_3} \frac{n!}{n_1!n_2!n_3!} A^{(n+m)}(n_1{+}m_1,n_2{+}m_2,n_3{+}m_3)\, B^{(n)}(n_1,n_2,n_3). \tag{4}$$

If $\mathbf{A}^{(n+m)}$ and $\mathbf{B}^{(n)}$ are totally symmetric and $\mathbf{A}^{(n+m)}$ is traceless, then $\mathbf{C}^{(m)}$ is also traceless. It can be proved as follows. Using Eq. (4), we have

$$\begin{aligned} C^{(m)}(m_1{+}2,m_2,m_3) + C^{(m)}(m_1,m_2{+}2,m_3) + C^{(m)}(m_1,m_2,m_3{+}2) = \sum_{n_1,n_2,n_3} \frac{n!}{n_1!n_2!n_3!} B^{(n)}(n_1,n_2,n_3)\cdot \\ [A^{(n+m)}(n_1{+}m_1{+}2,n_2{+}m_2,n_3{+}m_3) + A^{(n+m)}(n_1{+}m_1,n_2{+}m_2{+}2,n_3{+}m_3) + \\ A^{(n+m)}(n_1{+}m_1,n_2{+}m_2,n_3{+}m_3{+}2)], \end{aligned} \tag{5}$$

where $m_1+m_2+m_3=m-2$, and $n_1+n_2+n_3=n$. The part inside the square bracket is zero since $\mathbf{A}^{(m+n)}$ is traceless, according to Eq. (2). Then

$$C^{(m)}(m_1+2,m_2,m_3)+C^{(m)}(m_1,m_2+2,m_3)+C^{(m)}(m_1,m_2,m_3+2)=0, \tag{6}$$

which means $\mathbf{C}^{(m)}$ is traceless. Now we conclude that the n-fold contraction between a traceless totally symmetric tensor $\mathbf{A}^{(m+n)}$ and a totally symmetric tensor $\mathbf{B}^{(n)}$ yields an m-rank traceless totally symmetric tensor.

2.3. The detracer operator

The detracer operator $\mathcal{D}$ acts on a totally symmetric tensor $\mathbf{A}^{(n)}$ so that $\mathcal{D}\mathbf{A}^{(n)}$ is a traceless totally symmetric tensor. This operation is defined as

$$\mathcal{D}A^{(n)}_{\alpha_1\cdots\alpha_n}=\sum_{m=0}^{\lfloor\frac{n}{2}\rfloor}(-1)^m(2n-2m-1)!!\sum_{P\{\alpha\}}\delta_{\alpha_1\alpha_2}\cdots\delta_{\alpha_{2m-1}\alpha_{2m}}A^{(n:m)}_{\alpha_{2m+1}\cdots\alpha_n}, \tag{7}$$

where $P(\alpha)$ denotes all permutations of $\alpha_1\cdots\alpha_n$, and $n!!$ denotes the double factorial of n. If $\mathbf{A}^{(n)}$ is expressed in the compressed form, the same equation can be rewritten as

$$\mathcal{D}A^{(n)}(n_1,n_2,n_3)=\sum_{m_1=0}^{\lfloor\frac{n_1}{2}\rfloor}\sum_{m_2=0}^{\lfloor\frac{n_2}{2}\rfloor}\sum_{m_3=0}^{\lfloor\frac{n_3}{2}\rfloor}c(m,n)\begin{bmatrix}n_1\\m_1\end{bmatrix}\begin{bmatrix}n_2\\m_2\end{bmatrix}\begin{bmatrix}n_3\\m_3\end{bmatrix}A^{(n:m)}(n_1-2m_1,n_2-2m_2,n_3-2m_3) \tag{8}$$

where

$$c(m,n)=(-1)^m(2n-2m-1)!!\quad\text{and}\quad\begin{bmatrix}n\\m\end{bmatrix}=\frac{n!}{2^m m!(n-2m)!}$$

2.4. Gradients of $1/r$

Maxwell [17] derived a general expression for a solid spherical harmonic which are polynomials of negative degree $-(n+1)$ as the nth order gradient of $1/r$ with respect to a set of n arbitrary axes. When all of the axes coincide with Cartesian axes, his formula reduces to

$$\boldsymbol{\nabla}^n r^{-1}=(-1)^n r^{-2n-1}\mathcal{D}\mathbf{r}^n, \tag{9}$$

where the tensor operator $\boldsymbol{\nabla}^n:=\underbrace{\boldsymbol{\nabla}\cdots\boldsymbol{\nabla}}_{n}$.

$\mathbf{r}^n$ is obviously a totally symmetric tensor. From the definition of the detracer operation, it is easy to see that $\boldsymbol{\nabla}^n r^{-1}$ is a traceless and totally symmetric tensor. Thus the tensor $\boldsymbol{\nabla}^n r^{-1}$ has only $(2n+1)$ independent elements. Using Eqs. (7), (8), and (9), an element in tensor $\boldsymbol{\nabla}^n r^{-1}$ can be expressed explicitly as

$$\partial_i^{n_1}\partial_j^{n_2}\partial_k^{n_3}\left(\frac{1}{r}\right)=(-1)^n r^{-2n-1}\sum_{m_1=0}^{\lfloor\frac{n_1}{2}\rfloor}\sum_{m_2=0}^{\lfloor\frac{n_2}{2}\rfloor}\sum_{m_3=0}^{\lfloor\frac{n_3}{2}\rfloor}c(m,n)\begin{bmatrix}n_1\\m_1\end{bmatrix}\begin{bmatrix}n_2\\m_2\end{bmatrix}\begin{bmatrix}n_3\\m_3\end{bmatrix}r^{2m}x^{n_1-2m_1}y^{n_2-2m_2}z^{n_3-2m_3}, \tag{10}$$

where $n=n_1+n_2+n_3$, and $m=m_1+m_2+m_3$.

2.5. Taylor expansion of $f(\mathbf{r})$ in tensorial form

The Taylor expansion of a function $f(\mathbf{r}-\mathbf{r}')$ can be expressed in tensorial form as follows:

$$f(\mathbf{r}-\mathbf{r}')=\sum_{n=0}^{\infty}\frac{(-1)^n}{n!}\mathbf{r}'^n:\boldsymbol{\nabla}^n f(\mathbf{r}). \tag{11}$$

Furthermore the function $f(\mathbf{r}-\mathbf{r}')$ can be written as

$$f\left(\mathbf{r}-\mathbf{r}'\right)=\begin{cases}\sum_{n=0}^{\infty}\mathbf{M}^{(n)}:\boldsymbol{\nabla}^n f(\mathbf{r}) & \text{for } \mathbf{r}>\mathbf{r}'\\ \sum_{n=0}^{\infty}\mathbf{r}^n:\mathbf{L}^{(n)} & \text{for } \mathbf{r}'>\mathbf{r}\end{cases}, \tag{12}$$

with

$$\mathbf{M}^{(n)} = \frac{(-1)^n}{n!}\mathbf{r}'^n, \tag{13}$$

$$\mathbf{L}^{(n)} = \frac{(-1)^n}{n!}\nabla^n f(\mathbf{r}). \tag{14}$$

Equations (12)–(14) lead to the Cartesian tensor based expression for the multipole expansion and the local expansion, if $\mathbf{r}$ and $\mathbf{r}'$ denote the location of the observation and source points, respectively.

3. Multilevel fast multipole algorithm and Cartesian tensor formulas for Coulomb interaction

3.1. Multilevel fast multipole algorithm

The well-known multiple level fast multipole algorithm (MLFMA) is described with details in [6]. The key idea is to represent the potential of groups of source particles that are far away from the observer in terms of expansions, taking advantage of the fact that the Coulomb potential decreases with the distance. The whole domain is divided into boxes of different sizes depending on the charge density, so that on average each box includes a similar number of particles. Thus the charges are grouped into different boxes. The interactions between groups far way are evaluated by expansions, while the interactions between the groups nearby are calculated directly by the pair-to-pair theoretical formula. Relations between boxes can be represented as a hierarchical tree. Small boxes generated by subdividing a large box are called child boxes of the large box, and the large box is called the parent box of the small ones. The boxes that have no child boxes are called childless boxes. If the distance from any observer to a source box is not less than the side length of the source box, the field on the observer can be represented by the multipole expansion located inside the source box. If the distance from any source particle to the observer box is not less than the side length of the observer box, its contribution to the field inside the observer box can be represented as a local expansion inside the observer box. Each box is treated as both a source box and a observer box, thus we will calculate both the multiple expansion and the local expansion for the box. For a childless box the multipole expansion is calculated from the charges inside it, while for a parent box the multipole expansion is calculated from the multipole expansions of its child boxes. Local expansion of a observer box can be calculated from the multipole expansion of a source box, if the distance between them is not less than the side length of either of them. If the distance is less than the side length of the observer box, but not less than the side length of the source box and the source box is childless, the local expansion can be calculated from the charges in the source box. Child boxes also inherit the local expansions from their parent boxes. The general strategy is to calculate the multipole expansions for all the childless boxes and then transfer it into multipole expansions of the parent boxes, going up the tree of boxes. Once this is done, the local expansions for all the boxes can be calculated from the coarsest level of the tree to the finest level. The potential or field on each particle in the childless boxes comprises two parts: the contribution of the nearby particles, which is calculated using the pairwise formula, and the field due to the particles far away, which is calculated from the multipole expansions and/or the local expansions.

As a summary, besides the construction of the hierarchical tree of boxes, the MLFMA includes the following operations: (i) Calculate the multipole expansion from the source particles (P2M), (ii) translate the multipole expansions to the parent boxes (M2M), (iii) convert the multipole expansions into the local expansions (M2L), (iv) translate the local expansions into the child boxes (L2L), (v) calculate the local expansions from the source particles (P2L), (vi) calculate the potential/field on the observer particles from the local expansions (L2P), (vii) calculate the potential/field on the observer particles from the multipole expansions (M2P), and (viii) calculate the potential/field on the observer particles due to the source particles nearby using the pairwise theoretical formula (P2P).

3.2. Cartesian tensor based formulas for Coulomb interaction

We use the same formulas based on the totally symmetric Cartesian tensor in [13] for the $R^{-\nu}$ kernel and set $\nu = 1$ for the Coulomb kernel. Only the formulas for the single level FMM are presented in [13], but the extra formulas for MLFMA are derived by us. For completion of the paper, all the formulas are presented as follows.

3.2.1. Particles to multipole expansion (P2M)

Consider a domain $\Omega_s \in \mathbb{R}^3$ that is populated with k sources. The ith source has charge q_i and locates at $\mathbf{r}_i \in \Omega_s$, where $i = 1, \cdots, k$. Using Eq. (13), the total potential due to these k sources at any point $\mathbf{r}$, which is far away enough, is given as

$$\begin{aligned}\phi(\mathbf{r}) &= \sum_{i=1}^{k} \frac{q_i}{|\mathbf{r} - \mathbf{r}_i|} = \sum_{n=0}^{\infty} \left(\sum_{i=1}^{k} (-1)^n \frac{q_i}{n!} \mathbf{r}_i^n \right) : \nabla^n \frac{1}{r} \\ &= \sum_{n=0}^{\infty} \mathbf{M}^{(n)} : \nabla^n \frac{1}{r}\end{aligned} \tag{15}$$

with

$$\mathbf{M}^{(n)} = \sum_{i=1}^{k} (-1)^n \frac{q_i}{n!} \mathbf{r}_i^n. \tag{16}$$

$\mathbf{M}^{(n)}$ is the multipole expansion around the origin. If the summation in Eq. (15) is truncated at rank P, the truncation error ϵ is generated as

$$\epsilon \leq C_1 \left(\frac{a}{r}\right)^{P+1} \cdot \frac{1}{r-a}, \tag{17}$$

where $C_1 = \sum_{i=1}^{k} q_i$ and $r > a \geq r_i$ for any source particle position $\mathbf{r}_i$.

3.2.2. Particles to local expansion (P2L)

The potential inside a observation domain Ω_o, centered at $\mathbf{r}_o$, due to k sources that are far away enough can be expressed as

$$\begin{aligned} \phi(\mathbf{r}) &= \sum_{i=1}^{k} \frac{q_i}{|\boldsymbol{\rho} - \mathbf{r}_i|} \\ &= \sum_{n=0}^{\infty} \boldsymbol{\rho}^n : \left(\sum_{i=1}^{k} (-1)^n \frac{q_i}{n!} \nabla^n \frac{1}{r_i} \right) \\ &= \sum_{n=0}^{\infty} \boldsymbol{\rho}^n : \mathbf{L}^{(n)}, \end{aligned} \tag{18}$$

with

$$\mathbf{L}^{(n)} = \sum_{i=1}^{k} (-1)^n \frac{q_i}{n!} \nabla^n \frac{1}{r_i}, \tag{19}$$

where $\mathbf{r}_i$ is the coordinate of the ith source with charge q_i with respect to $\mathbf{r}_o$, and $\boldsymbol{\rho} = \mathbf{r} - \mathbf{r}_o$. When the summation in Eq. (18) is truncated at rank P, the truncation error ϵ is

$$\epsilon \leq C_2 \left(\frac{\rho}{b}\right)^{P+1} \cdot \frac{1}{b-\rho}, \tag{20}$$

where $C_2 = \sum_{i=1}^{k} q_i$ and $\rho < b \leq r_i$ for any source particle position $\mathbf{r}_i$.

3.2.3. Multipole expansion to multipole expansion (M2M)

Given a multipole expansion of k sources around $\mathbf{r}_s$, the multipole expansion around the point $\mathbf{r}_{s'}$ can be expressed as

$$\mathbf{M}'^{(n)} = \sum_{m=0}^{n} \sum_{P(m,n)} \frac{m!}{n!} \mathbf{r}_{ss'}^{n-m} \mathbf{M}^{(m)} \tag{21}$$

where $\mathbf{r}_{ss'} = \mathbf{r}_{s'} - \mathbf{r}_s$ and $P(n,m)$ is the permutation of all partitions of n into sets $n-m$ and m. Taking advantage of the totally symmetric tensor, the permutation, in the compressed notation, reduces to

$$M'^{(n)}(n_1,n_2,n_3) = \sum_{m_1=0}^{n_1} \sum_{m_2=0}^{n_2} \sum_{m_3=0}^{n_3} \frac{(n-m)!}{n!} \begin{bmatrix} n_1 \\ m_1 \end{bmatrix} \begin{bmatrix} n_2 \\ m_2 \end{bmatrix} \begin{bmatrix} n_3 \\ m_3 \end{bmatrix} \mathbf{r}_{ss'}^{m}(m_1,m_2,m_3) M^{(n-m)}(n_1-m_1, n_2-m_2, n_3-m_3) \tag{22}$$

This translation is exact.

3.2.4. Multipole expansion to local expansion (M2L)

Assume that the domains Ω_s and Ω_o are sufficiently separated, and the distance between their centers $r_{os} = |\mathbf{r}_{os}| = |\mathbf{r}_o - \mathbf{r}_s|$ is greater than $D\{\Omega_s\}$ and $D\{\Omega_o\}$. If a multipole expansion $\mathbf{M}^{(n)}$ is located at $\mathbf{r}_s$, then another expansion $\mathbf{L}^{(n)}$ that locates at $\mathbf{r}_o$ and produces the same field $\forall \mathbf{r} \in \Omega_o$ is given by

$$\mathbf{L}^{(n)} = \sum_{m=n}^{\infty} \frac{1}{n!} \mathbf{M}^{\{m-n\}} : \widetilde{\nabla}^m \frac{1}{r_{so}}, \tag{23}$$

where $\widetilde{\nabla}$ is the derivative w.r.t. $\mathbf{r}_s$ and $\widetilde{\nabla} = -\nabla$. $\mathbf{L}^{(n)}$ is called the local expansion since it locates in the observer domain Ω_o, and the potential can be calculated as

$$\phi(\mathbf{r}) = \sum_{n=0}^{\infty} \boldsymbol{\rho}^n : \mathbf{L}^{(n)}, \tag{24}$$

where $\boldsymbol{\rho} = \mathbf{r} - \mathbf{r}_o$. If the calculations of the multipole expansion and the local expansion are both carried out up to rank P, the error ϵ is generated by the truncation of both expansions, which can be estimated as

$$\epsilon \leq C_1 \left(\frac{a}{r}\right)^{P+1} \cdot \frac{1}{r-a} + C_2 \left(\frac{\rho}{b}\right)^{P+1} \cdot \frac{1}{b-\rho} \tag{25}$$

where $\rho = |\boldsymbol{\rho}|$.

3.2.5. Local expansion to local expansion (L2L)

A local expansion $\mathbf{L}^{(n)}$ that exists in the domain Ω_o centered around $\mathbf{r}_o$ can be shifted to a subdomain centered at $\mathbf{r}_{o'}$ using

$$\mathbf{L}'^{(n)} = \sum_{m=n}^{\infty} \begin{pmatrix} m \\ m-n \end{pmatrix} \mathbf{L}^{(m)} : \mathbf{r}_{oo'}^{m-n} \tag{26}$$

where

$$\mathbf{r}_{oo'} = \mathbf{r}_{o'} - \mathbf{r}_o \text{ and } \begin{pmatrix} m \\ m-n \end{pmatrix} = \frac{m!}{(m-n)!n!}$$

This translation does not generate errors.

3.2.6. Potential calculation using multipole expansion (M2P) and local expansion (L2P)

The potential at any point $\mathbf{r}_i$ far away from domain Ω_o centered around $\mathbf{r}_o$ can be obtained using

$$\phi(\mathbf{r}_i) = \sum_{n=0}^{\infty} \mathbf{M}^{(n)} : \nabla^n \frac{1}{\rho_i}, \tag{27}$$

or the potential at any point $\mathbf{r}_i$ inside domain Ω_o centered around $\mathbf{r}_o$ can be obtained using

$$\phi(\mathbf{r}_i) = \sum_{n=0}^{\infty} \mathbf{L}^{(n)} : \boldsymbol{\rho}_i^n, \tag{28}$$

where $\boldsymbol{\rho}_i = \mathbf{r}_i - \mathbf{r}_o$.

3.2.7. Field calculation

Once the explicit expressions for the potential found, it is straightforward to calculate the field by taking derivatives with respect to the coordinates. So we have

$$\mathbf{E}(\mathbf{r}) = -\nabla\phi(\mathbf{r}) = \begin{cases} -\sum_{n=0}^{\infty} \mathbf{M}^{(n)} : \nabla^{n+1} \frac{1}{r} & \text{for } \mathbf{r} > \mathbf{r}' \\ -\sum_{n=0}^{\infty} \nabla \mathbf{r}^n : \mathbf{L}^{(n)} & \text{for } \mathbf{r}' > \mathbf{r} \end{cases}. \tag{29}$$

3.3. Computational complexity

In this section, we will discuss the computational cost for the formulas in section 3.2. All the tensors related are totally symmetric and all the $(n+1)(n+2)/2$ independent elements are saved. For an nth rank totally symmetric Cartesian tensor, the elements can be expressed in the format of $x^{n_1} y^{n_2} z^{n_3}$ with $n_1 + n_2 + n_3 = n$. The tensors are sorted in ascending order of the rank n. In one tensor, the elements are sorted in ascending order of n_3, and the elements with the same n_3 are sorted in ascending order of n_2. For example, then elements of a rank 3 tensor are sorted as in Table 1. If a tensor is also traceless, only the $2n+1$ independent elements are calculated using the formulas in section 3.2. Any other element, who is also saved for the convenience of the contraction operation, is calculated using Eq. (2) as the summation of two other elements with an opposite sign. These two elements have already been calculated, if the elements are sorted in the aforesaid way. Comparing with the first $2n+1$ elements, the computation cost of the remaining element is usually negligible, and this is why the efficiency are improved with the traceless tensors. In the following, we estimate the computation cost of all the formulas in MLFMA. Since there are no special functions or any other concealed calculation involved, we simply count the number of the multiplications of double flouting point numbers in each formula.

Table 1
Sorting of elements in a tensor of rank 3.

Sorting	1	2	3	4	5	6	7	8	9	10
n_1	3	2	1	0	2	1	0	1	0	0
n_2	0	1	2	3	0	1	2	0	1	0
n_3	0	0	0	0	1	1	1	2	2	3

To calculate the multipole expansion for a childless box using Eq. (16), one needs to calculate the tensor $\mathbf{r}_i^{(n)}$ from zero to the desired order for each source particle i and take the summation over all the particles. The tensor $\mathbf{r}^{(n)}$, subscript i omitted for convenience here, is totally symmetric but not traceless. However, it has the following property of its trace:

$$x^{n_1+2}y^{n_2}z^{n_3} + x^{n_1}y^{n_2+2}z^{n_3} + x^{n_1}y^{n_2}z^{n_3+2} = x^{n_1}y^{n_2}z^{n_3}\cdot r^2, \tag{30}$$

where $n_1+n_2+n_3=n$ and $r^2=x^2+y^2+z^2$. For any source particle, the multipole expansion is calculated from the lower rank to the higher rank. Thus when calculating the $(n+2)$th rank tensor, $x^{n_1}y^{n_2}z^{n_3}$, as an element of the nth rank tensor, is known. Using Eq. (30), the number of independent elements for $\mathbf{r}^{(n)}$ is reduced to $2n+1$, the same as of a traceless totally symmetric tensor. To calculate the multipole expansions for all the childless boxes, Eq. (16) needs to be applied once on each particle. Hence the total cost of this operation is

$$\begin{aligned} C_{P2M} &\propto \text{total no. of particles} \times \text{cost for each particle} \\ &\propto \text{total no. of particles} \times \text{cost per tensor} \times \text{no. of tensors} \\ &\propto N \times \sum_{n=0}^{P}(2n+1) \\ &\propto N(1+2P+P^2) \end{aligned}$$

Unfortunately, Eq. (30) is only valid for the multipole expansion of an individual particle, not for that of a group of particles. When translating a multipole expansion to its parent box, for any nth rank tensor in the multipole expansion, one has to calculate all the $(n+1)(n+2)/2$ independent elements using Eq. (22). The cost for each element is $(n_1+1)(n_2+1)(n_3+1)$, where $n_1+n_2+n_3=n$. It is easy to see the cost is less than $(n/3+1)^3$ due to the symmetry. The total cost of calculating the multipole expansions for all the parent boxes is

$$\begin{aligned} C_{M2M} &\propto \text{total no. of boxes} \times \text{cost for each } \mathbf{M}^{(n)} \\ &\propto \frac{N}{s} \times \sum_{n=0}^{P} \frac{(n+1)(n+2)}{2} \cdot \left(\frac{n}{3}+1\right)^3 \\ &\propto \frac{N}{s}(2.63P + 2.67P^2 + 1.37P^3 + 0.38P^4 + 0.053P^5 + 0.0031P^6) \end{aligned}$$

One thing we want to note here is the transition tensor $\mathbf{r}_{ss'}^{(n)}$ in Eq. (22) has only eight possible values for the same level of boxes, which correspond to the eight child boxes. One only needs to calculate $\mathbf{r}_{ss'}^{(n)}$ once for the finest level, and then update it by multiplying 2^n to each element when move upward to the coarser level. The details are presented in Appendix A.2. The calculation cost of $\mathbf{r}_{ss'}^{(n)}$ is ignored.

A multipole expansion can be converted into a local expansion using Eq. (23). In Eq. (23), $\mathbf{M}^{\{m-n\}}$ is a totally symmetric tensor and $\widetilde{\nabla}^m \frac{1}{r_{so}}$ is a traceless totally symmetric tensor with higher order. As the contraction of them, $\mathbf{L}^{(n)}$ is a traceless totally symmetric tensor, as proved in section 2.2. So we only need to calculate the $2n+1$ independent elements in $\mathbf{L}^{(n)}$ using Eq. (23). All the other elements are calculated using Eq. (2) to save time. Converting a multipole expansion into a local expansion is the most complicated operation. The local expansion is composed of a group of traceless tensors up to rank P. For each nth rank traceless tensor, there are $2n+1$ independent elements. For each element, one needs to calculate the contraction of two tensors $P-n$ time. For each contraction, there are $(i+1)(i+2)/2$ multiplications of two respective elements, where i is the lower rank of the two tensors. There are at most 189 multipole expansions for each objective box, thus the total cost is

$$\begin{aligned} C_{M2L} &\propto 189 \times \text{total no. of boxes} \times \text{cost for each } \mathbf{L}^{(n)} \\ &\propto 189 \times \frac{N}{s} \times \sum_{n=0}^{P}(2n+1)\sum_{i=0}^{P-n}\frac{(i+1)(i+2)}{2} \\ &\propto 189 \times \frac{N}{s}(1 + 2.41P + 2.29P^2 + P^3 + 0.21P^4 + 0.016P^5) \end{aligned}$$

Similarly with the M2M transition tensor, the M2L transition tensor $\widetilde{\nabla}^m \frac{1}{r_{so}}$ in Eq. (23) has only a few possible values for a fixed level. Once it is calculated for a level l, it can be easily updated for level $l+1$ by multiplying a proper integer. The details are presented in Appendix A.3. The calculation cost of $\widetilde{\nabla}^m \frac{1}{r_{so}}$ is ignored.

Using Eq. (19) to calculate the local expansion from source particles, there are $2n+1$ independent elements for each nth rank tensor. Each element can be calculated by Eq. (10), whose cost is $(n_1/2+1)(n_2/2+1)(n_3/2+1)$, where $n_1+n_2+n_3=n$. Due to the symmetry, it is easy to see the cost is less than $(n/6+1)^3$. The total cost of P2L is

$$\begin{aligned} C_{P2L} &\propto \text{total no. of boxes} \times \text{cost for each } \mathbf{L}^{(n)} \\ &\propto \frac{N}{s} \times \sum_{n=0}^{P} (2n+1)\left(\frac{n}{6}+1\right)^3 \\ &\propto \frac{N}{s}(1+2.43P+1.83P^2+0.45P^3+0.047P^4+0.0019P^5) \end{aligned}$$

Comparing Eq. (26) with Eq. (23), it is obvious the calculation cost of translating a local expansion is the same with that of converting a multipole expansion into a local expansion. The calculation cost of the transition tensor, as in Appendix A.2, can be ignored for the same reason above. The total cost of L2L is

$$\begin{aligned} C_{L2L} &\propto \text{total no. of boxes} \times \text{cost for each } \mathbf{L}^{(n)} \\ &\propto \frac{N}{s}(1+2.41P+2.29P^2+P^3+0.21P^4+0.016P^5) \end{aligned}$$

Calculate the potential for each particle from either a multipole expansion using Eq. (27) or a local expansion using Eq. (28) has the same cost, which comprises the construction of the tensor, $\nabla^n \frac{1}{\rho}$ or $\boldsymbol{\rho}^n$, and the contraction of tensors up to rank P. The total cost of both M2P and L2P is

$$\begin{aligned} C_{M2C}+C_{L2C} &\propto \text{total no. of particles} \times \text{cost for each particle} \\ &\propto N \times \left(\sum_{n=0}^{P}(2n+1)+\sum_{n=0}^{P}\frac{(n+1)(n+2)}{2}\right) \\ &\propto N(2+4.83P+3.5P^2+0.67P^3) \end{aligned}$$

The computation cost of direct calculation between particles in their near region is obviously

$$\begin{aligned} C_{P2P} &\propto \text{total no. of particles} \times \text{no. of nearby particles} \\ &\propto N \times 27s \\ &\propto 27Ns \end{aligned}$$

The total computation cost is the summation of the cost of each operation above. Using the property of the trace, the computation costs of P2M, M2L, P2L, L2L, M2P and L2P have been reduced from $O(P^6)$ to $O(P^5)$. The computation cost of M2M remains as $O(P^6)$. But the M2L translation is generally the most complicated operation and dominates the computation cost. The M2M cost will not be significant without an extremely large P. But in practice, a moderate P is usually accurate enough. In the next section, we will present some numerical results, which demonstrate the improvement of efficiency.

4. Numerical results

To test the property of the algorithm, numerical experiments have been carried out in a personal computer with Intel i7 3630QM CPU running at 2.4 GHz. In all the calculations below, the particles have three dimensional Gaussian distributions with unit standard deviation.

4.1. Convergence of the algorithm

The error of the Cartesian tensor based FMM is determined by the rank of the tensors, as in Eq. (17), Eq. (20) and Eq. (25). The higher the rank, the smaller the error. To test the convergence of the algorithm, we calculate the Coulomb potential and the Coulomb field between 1,000,000 particles using the algorithm and changing the rank of the tensors from two to ten. The results are compared with those calculated using the theoretical formulas. The relative errors for the potential and the field are calculated respectively as

Table 2
Relative errors for different ranks.

Rank	2	4	6	8	10
σ_ϕ	1.825×10^{-3}	1.345×10^{-4}	2.523×10^{-5}	6.517×10^{-6}	2.720×10^{-6}
$\sigma_{\mathbf{E}}$	2.728×10^{-2}	4.645×10^{-3}	1.342×10^{-3}	4.724×10^{-4}	1.946×10^{-4}

Table 3
Computation time for different particle numbers and different ranks.

N	Rank 2		Rank 4		Rank 6	
	T_ϕ (s)	$T_{\mathbf{E}}$ (s)	T_ϕ (s)	$T_{\mathbf{E}}$(s)	T_ϕ (s)	$T_{\mathbf{E}}$ (s)
10,000	0.148	0.145	0.290	0.384	0.554	0.644
50,000	0.656	0.754	1.259	1.424	2.494	2.923
100,000	1.367	1.622	2.728	3.321	6.283	7.296
500,000	5.703	7.987	12.790	15.903	29.092	33.584
1,000,000	10.326	14.181	23.178	28.250	55.206	62.576
5,000,000	73.300	88.428	136.147	172.813	304.463	337.392
10,000,000	151.747	200.746	270.753	338.419	541.240	623.197

$$\sigma_\phi = \sqrt{\frac{\sum_{i=1}^{N}(\phi(\mathbf{r}_i) - \phi_0(\mathbf{r}_i))^2}{\sum_{i=1}^{N}\phi_0^2(\mathbf{r}_i)}} \quad \text{and} \quad \sigma_{\mathbf{E}} = \sqrt{\frac{\sum_{i=0}^{N}|\mathbf{E}(\mathbf{r_i}) - \mathbf{E_0}(\mathbf{r_i})|^2}{\sum_{i=0}^{N}|\mathbf{E}(\mathbf{r_i})|^2}}, \tag{31}$$

where ϕ and $\mathbf{E}$ are the potential and the field calculated by the algorithm, ϕ_0 and $\mathbf{E}_0$ are those calculated by the theoretical formulas, and N is the number of particles. The result of the test calculation is presented in Table 2. We can see that σ_ϕ is reduced by about 2.5 orders and $\sigma_{\mathbf{E}}$ by about two orders, when the rank increases from two to ten. The rapid decrease of the relative error agrees with the error analysis and demonstrates convergence of the algorithm. We also noticed that the relative error of the potential is much less than the relative error of the field.

4.2. Linear computational complexity

The linear scaling of the computation time with the particle number is well known for the algorithms in the FMM family. To confirm this property of our algorithm, we calculated the potential and the field respectively for different number of particles, ranging from 10,000 to 10,000,000, and recorded the computation time. The results of test runs with different ranks of two, four, and six are presented in Table 3. It is not surprising to see that higher rank results in longer time, since more elements have to be calculated and higher accuracy is achieved. The time for field calculation is a little longer than the potential calculation. This is because the field, as a vector, has three components, all of which need to be calculated, while the potential is a scalar. However, this difference only matters in the near region calculation and in the M2P and L2P operations in the far region calculation. The creation of the multipole expansion and local expansion, the locomotion of them, and the transformation between them are all the same for both the potential calculation and the field calculation. This explains why the time cost for field calculation is only slightly longer than the potential calculation. When plotted in the logarithmic frame, as shown in Fig. 1, the computation time for different particle numbers can be fitted by a straight line. The slope is marked for each line in the figure. All the slopes, for the different ranks and for both the potential and the field calculation, are very close to one, which means the efficiency of the algorithm scales linearly with the number of particles for all the cases.

4.3. Improved computational efficiency

Using the trace property of the totally symmetric Cartesian tensor, the independent element number in an nth rank tensor is reduced from $(n+1)(n+2)/2$ to $2n+1$. Thus the computation time is reduced. In Table 4, the number of the independent elements in a general totally symmetric Cartesian tensor is listed in column two, and that in a traceless totally symmetric Cartesian tensor is listed in column four. The total number of the independent elements in a multipole/local expansion, which includes all the tensors up to the specific rank, is listed in column three and five respectively for the two kinds of tensors. The last column lists the reduction of the total number in percentage (Δ), when the trace property is implemented in calculation. One can reasonably expect that the reduction of computation time is close to Δ.

In Table 5, the time cost to calculate the potential and the field between 100,000, 1,000,000, and 5,000,000 particles using both the totally symmetric Cartesian tensor based FMM and the traceless totally symmetric Cartesian tensor based FMM are presented. The calculations are repeated for different ranks of the tensors, from two to ten. As one can see in the table, the traceless FMM has better efficiency. For all the cases, the computation time of the traceless FMM is less than that of the control group. Take the computation time for the potential between 1,000,000 particles as an example. The computation time using the two kinds of FMM with different ranks of tensors is plotted in Fig. 2(a). As it shows, the higher the rank, the more reduction of computation time is achieved. The reduction in percentage is marked above the respective data points

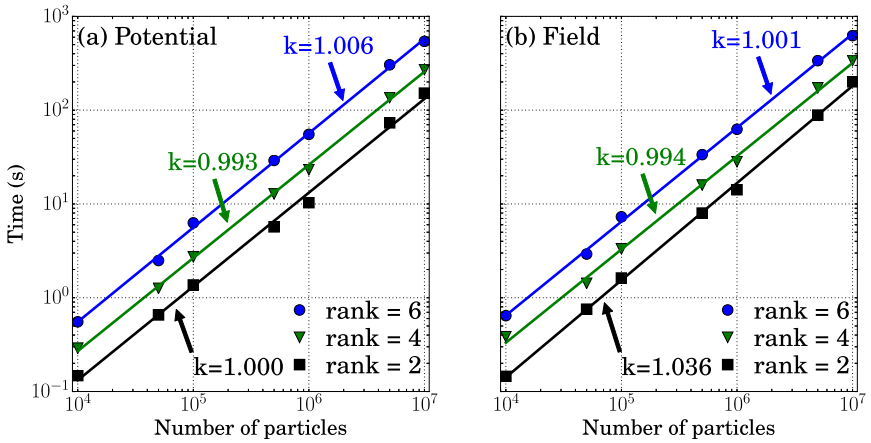

Fig. 1. Efficiency of the algorithm.

Table 4
Number of elements in Cartesian tensor based FMMs.

n	Totally symmetric		Traceless		Δ (%)
	in n-th order	total elements	in n-th order	total elements	
0	1	1	1	1	0.0
1	3	4	3	4	0.0
2	6	10	5	9	10.0
3	10	20	7	16	20.0
4	15	35	9	25	28.6
5	21	56	11	36	35.7
6	28	84	13	49	41.7
7	36	120	15	64	46.7
8	45	165	17	81	50.9
9	55	220	19	100	54.5
10	60	280	21	121	56.8

Table 5
Computation time of Cartesian tensor based FMMs.

N	Rank	Totally symmetric		Traceless	
		T_ϕ (s)	$T_{\mathbf{E}}$ (s)	T_ϕ (s)	$T_{\mathbf{E}}$ (s)
100,000	2	1.565	1.750	1.445	1.600
	4	3.454	4.146	2.756	3.295
	6	9.431	10.776	6.299	7.178
	8	24.020	26.635	13.448	14.943
	10	56.570	61.103	26.861	29.412
1,000,000	2	11.332	15.173	10.733	14.276
	4	29.052	34.631	23.244	28.260
	6	81.717	93.712	55.787	63.125
	8	221.130	238.088	128.257	137.291
	10	531.841	558.893	261.524	279.105
5,000,000	2	77.573	93.856	73.902	90.073
	4	165.034	187.902	134.948	160.234
	6	427.411	482.567	297.073	330.811
	8	1138.320	1216.080	637.377	693.958
	10	2708.530	2869.260	1302.180	1395.570

and it also increases as the rank goes up. For the rank equal to or greater than ten, the reduction is more than 50%. If we compare the reduction percentage in Fig. 2(a) with the respective number in the sixth column of Table 4, we found the reduction of the computation time is close to the reduction of the number of elements, which means the traceless FMM behaves as expected. We also noticed that the reduction of computation time is slight smaller than the reduction of the number of elements, which can be explained by two reasons. First, all the $(n+1)(n+2)/2$ elements in an nth rank tensor are calculated, and the computation time for non-independent ones is small but not zero. Second, the algorithm includes calculations that cannot be reduced using the trace, such as M2M, M2P, L2P, etc. In Fig. 2(b), the computation time for the

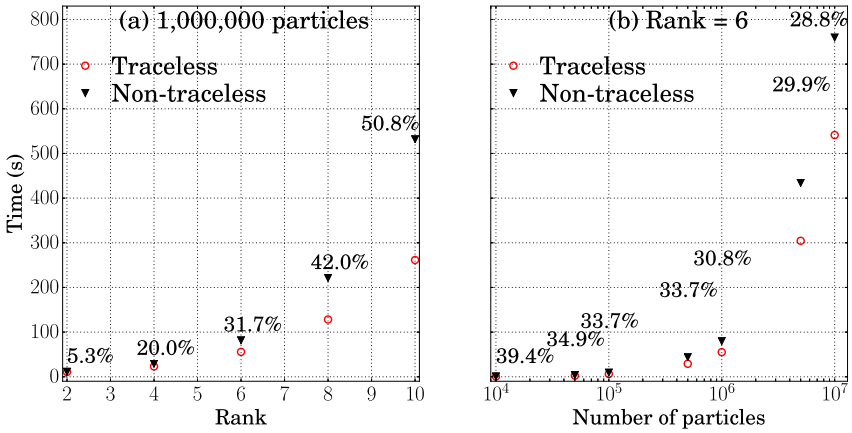

Fig. 2. (a) Computation time for potential between 1,000,000 particles using tensors up to different ranks; (b) Computation time for potentials between various amounts of particles using tensors up to rank 6.

Table 6
Potential calculation for 1,000,000 particles of normal distribution.

pyfmmlib			Traceless Cartesian tensor FMM			
iprec	σ_ϕ	T_ϕ (s)	rank	s	σ_ϕ	T_ϕ (s)
-2	7.78×10^{-4}	29.70	3	64	4.09×10^{-4}	18.71
-1	1.00×10^{-4}	37.11	4	64	1.10×10^{-4}	26.74
0	2.03×10^{-6}	60.88	10	256	2.27×10^{-6}	159.25

Table 7
Field calculation for 1,000,000 particles of normal distribution.

pyfmmlib			Traceless Cartesian tensor FMM			
iprec	$\sigma_{\mathbf{E}}$	$T_{\mathbf{E}}$ (s)	rank	s	$\sigma_{\mathbf{E}}$	$T_{\mathbf{E}}$ (s)
-2	1.68×10^{-2}	33.31	3	64	7.14×10^{-3}	24.37
-1	3.82×10^{-3}	42.06	4	64	2.12×10^{-3}	33.00
0	9.53×10^{-5}	70.69	9	128	1.04×10^{-4}	164.36

potential between various amounts of particles using the two kinds of FMMs are plotted. The reductions of the computation time in percentage are marked above the respective data points. The tensors are calculated up to the rank six. We observed that the percentage of reduction decreases as the number of particle N increases. When $N = 10{,}000$, the reduction is 39.4%, which is very close to 41.7%, the reduction of elements for rank six. But when $N = 10{,}000{,}000$, the reduction is only 28.8%. This is because the operations like M2M, M2P, L2P cost very little time, when N is small. When N increase, the time cost of M2P and L2P naturally increases. So does the time cost of M2M, since more boxes are generated. The calculation time of these operations are roughly the same for the both kinds of FMMs. When they take large portion of the total calculation time, the percentage of reduction due to the less elements in tensors decreases. But we also want to note here that the absolute time reduction is actually increases together with the particle number, although the percentage decreases. So more time will be saved when implementing the traceless FMM in computation on a larger amount of particles.

In the following we compare our codes with pyfmmlib [19], which is the Python wrapper for fmmlib2d [20] and fmmlib3d [21] implementations of the fast multipole method for Laplace and Helmholtz potentials. The computation is carried out in the same computer aforementioned. The fmmlib2d and fmmlib3d codes are well optimized for performance. Table 6 and Table 7 show the relative error and time cost in the second and the third column for the computation of potential and field for 1,000,000 particles with the normal distribution using pyfmmlib. The parameter, iprec, allows the users to define the error tolerance. The value of iprec ranges from -2 to 5. The larger the iprec is, the smaller the error is. Here we want to note that our code is developed for illustration of the idea. Although we have been careful to avoid any redundant computation, the codes can be further optimized for efficiency. For example, if an error tolerance is given, one can use adaptive rank of the tensors to improve the efficiency. The highest rank of the tensor is determined by the source box and the objective box with the shortest distance. For boxes further away from each other, the rank can be reduced to save time. Another trick is to rotate the frame so that the source box center and the objective box center are aligned in one axis. With the disappearance of the other two coordinates, the contraction operation is simplified and the efficiency can be improved [22]. After the computation, the frame is rotated back to the original. The two rotations are not free, so this method is only helpful when high rank tensors are needed. None of these have been implemented in our codes. The error and the time cost of our codes is listed in the last two columns in Table 6 and Table 7. We cannot predefine an error tolerance to run our codes as

pyfmmlib does. So we adjust the rank to make sure the error is comparable with the respective result by pyfmmlib. The parameter s in Table 6 and Table 7 defines the maximum number of particles inside each childless box. It mainly affects the efficiency with tiny effect on the error. We can see that for an error tolerance of 1×10^{-4} or larger, our codes have better efficiency. But when higher accuracy is desired and the rank goes up, pyfmmlib outperforms our codes. This is because the computation time of the Cartesian tensor FMM has a higher order dependency of the rank than the other versions of FMM. Although our work reduces the order dependency of the rank by one and our codes can be further optimized, it will not change the above conclusion in general. However, the Cartesian tensor FMM offers some different property than the other FMMs and it is implemented in some state-of-the-art FMM libraries, *e.g.* exaFMM [23]. As we observed, the Cartesian tensor FMM has better efficiency with lower rank, so it may be preferred for computations with moderate accuracy requirement. More important, the Cartesian tensor FMM provides higher flop intensity than the solid harmonics FMM [24]. In the current state-of-the-art implementations of FMMs, the operators are pre-calculated and called repeatedly during the calculation. Operators with less terms provide better efficiency. But the modern multi-core platforms show an tendency that the computation is becoming cheaper and the bandwidth is becoming more expensive. Potentially the FMM could become bandwidth-bound in future. In such a case, it will be more efficient to calculate the operators on-the-fly and the Cartesian tensor FMM will be preferred for its higher flop intensity [24,25].

5. Summary snd conclusion

In this paper we presented how the traceless property of the Cartesian tensor $\nabla^n r^{-1}$ can be implemented to improve the efficiency of the totally symmetric Cartesian tensor based FMM for the Coulomb interaction. We proved that the contraction of a high rank traceless totally symmetric Cartesian tensor with a low rank totally symmetric Cartesian results in a traceless totally symmetric Cartesian tensor, which allows us to reduce the independent element number from $(n+1)(n+2)/2$ to $2n+1$ directly in calculation, without using the detracer operator. The computation cost of M2L and L2L operations are reduced from $O(P^6)$ to $O(P^5)$. That of P2M operation is also reduced by one order of P, using the trace of the Cartesian tensor $\mathbf{r}^n$. Although the computation cost of M2M remains at $O(P^6)$, it is not the dominating operation of time cost. The M2L operation is usually dominating. In our numerical test, we have observed significant improvement of efficiency. The percentage of computation time reduction increases as the rank of the tensors goes up. Using tensors of the same rank, the absolute computation time reduction increases as the number of particles goes up. Depending on the rank and the number of particles, we have observed more than 50% reduction of time. This technique of independent element reduction will be especially useful for the simulations using the Cartesian tensor based FMM on the Coulomb interaction between a large amount of particles with high accuracy, which entails high rank of tensors.

Acknowledgements

This work is supported by the Department of Energy, Laboratory Directed Research and Development Funding, under Contract No. DE-AC05-06OR23177.

Appendix A

A.1. Error analysis

The locomotion of any expansion is exact. The error primarily comes from three sources; (i) Taylor expansion to create the multipole expansion, (ii) Taylor expansion to create the local expansion, and (iii) conversion of a multiple expansion into a local expansion. In this section, we will discuss the error in the above three process respectively.

Using Eq. (12), the function $f(\mathbf{r}-\mathbf{r}') = 1/|\mathbf{r}-\mathbf{r}'|$ can be expressed as

$$\frac{1}{|\mathbf{r}-\mathbf{r}'|} = \begin{cases} \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \mathbf{r}'^n : \nabla^n \frac{1}{r} & \text{for } \mathbf{r} > \mathbf{r}', \\ \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \mathbf{r}^n : \nabla^n \frac{1}{r'} & \text{for } \mathbf{r}' > \mathbf{r}. \end{cases} \tag{32}$$

On the other hand, using Legendre polynomials, we have [18,9]

$$\frac{1}{|\mathbf{r}-\mathbf{r}'|} = \begin{cases} \sum_{n=0}^{\infty} \frac{r'^n}{r^{n+1}} P_n(\cos\theta) & \text{for } \mathbf{r} > \mathbf{r}', \\ \sum_{n=0}^{\infty} \frac{r^n}{r'^{n+1}} P_n(\cos\theta) & \text{for } \mathbf{r}' > \mathbf{r}. \end{cases} \tag{33}$$

Comparing Eq. (32) and Eq. (33), one can see they have a term-by-term correspondence, which indicates

$$P_n(\cos\theta) = \frac{(-1)^n}{n!} \cdot r^{n+1} \cdot \left(\frac{\mathbf{r}'}{r'}\right)^n : \nabla^n \frac{1}{r}.$$

Notice that the Legendre polynomials $P_n(\cos\theta)$ only depend on the angle θ between $\mathbf{r}$ and $\mathbf{r}'$.

If the multipole expansion in Eq. (16) is truncated at rank P, the error can be estimated as

$$\begin{aligned}
\epsilon &= \left| \phi - \sum_{n=0}^{P} \mathbf{M}^{(n)} : \nabla^n \frac{1}{r} \right| \\
&= \left| \sum_{n=P+1}^{\infty} \mathbf{M}^{(n)} : \nabla^n \frac{1}{r} \right| \\
&= \left| \sum_{n=P+1}^{\infty} \sum_{i=1}^{k} (-1)^n \frac{q_i}{n!} \mathbf{r}_i^n : \nabla^n \frac{1}{r} \right| \\
&= \left| \sum_{n=P+1}^{\infty} \sum_{i=1}^{k} q_i \frac{(-1)^n}{n!} r^{n+1} \left(\frac{\mathbf{r}_i}{r_i} \right)^n : \nabla^n \frac{1}{r} \cdot \frac{r_i^n}{r^{n+1}} \right| \\
&= \left| \sum_{n=P+1}^{\infty} \sum_{i=1}^{k} q_i P_n(\cos\theta) \cdot \frac{r_i^n}{r^{n+1}} \right| \\
&\le \left| \sum_{n=P+1}^{\infty} C_1 \frac{a^n}{r^{n+1}} \right| \\
&= C_1 \cdot \frac{a^{P+1}}{r^{P+2}} \cdot \left| \sum_{n=0}^{\infty} \frac{a^n}{r^{n+1}} \right| \\
&= C_1 \cdot \frac{a^{P+1}}{r^{P+2}} \cdot \frac{1}{1 - \frac{a}{r}} \\
&= C_1 \left(\frac{a}{r} \right)^{P+1} \cdot \frac{1}{r-a},
\end{aligned}$$

where $C_1 = \sum_{i=1}^{k} q_i$, and $r > a \ge r_i$ for any source particle position $\mathbf{r}_i$. Similarly the error due to a local expansion in Eq. (19) truncated as rank P can be estimated as

$$\epsilon \le C_2 \left(\frac{\rho}{b} \right)^{P+1} \cdot \frac{1}{b-\rho},$$

where $C_2 = \sum_{i=1}^{k} q_i$ and $\rho < b \le r_i$ for any source particle position $\mathbf{r}_i$. When converting a multipole expansion into a local expansion using Eq. (23), the error is generated by the truncation of both expansions. Based on the above discussion, the error can be estimated as

$$\epsilon \le C_1 \left(\frac{a}{r} \right)^{P+1} \cdot \frac{1}{r-a} + C_2 \left(\frac{\rho}{b} \right)^{P+1} \cdot \frac{1}{b-\rho}.$$

Obviously the error decreases as the truncation rank P increases. The error also decreases, as decreases the ratio of the box size to the distance between two well separated boxes, which leads to the diminution of a/r and ρ/b.

A.2. Calculation of the M2M and the L2L transition tensor

Both the multipole expansion to multipole expansion transition tensor, in Eq. (26) and the local expansion to local expansion transition tensor, in Eq. (26) have the format of $\mathbf{r}^{(n)}$. $\mathbf{r}$ is the vector from the center of the child box to the center of the parent box for M2M operation, and it switches the direction for L2L operation. $\mathbf{r}$ has eight possible values, corresponding to the eight child boxes. Due to the symmetry, one only need to calculate one of them. The others can be constructed by changing the sign of some elements. For example, if the tensor for $\mathbf{r}(i, j, k)$ has been calculated, the tensor for $\mathbf{r}(-i, j, k)$ can be constructed by simply changing the sign of all the elements with odd order of coordinate x. If the tensors are known for level l, the tensors for a coarser level $l-1$ can be constructed by multiplying 2^n to all the elements, and the tensors for a finer level $l+1$ can be constructed by multiplying 0.5^n to all the elements.

A.3. Calculation of the M2L transition tensor

The multipole expansion to local expansion transition tensor in Eq. (23) has the format of $\nabla^n \frac{1}{r}$, where r is the distance from the source box, which holds the multipole expansion, to the objective box, which holds the local expansion, and it is calculated by Eq. (10). We separate the terms within the three layer summation in Eq. (10) into two parts:

$$\partial_i^{n_1}\partial_j^{n_2}\partial_k^{n_3}\left(\frac{1}{r}\right) = (-1)^n r^{-2n-1}\sum_{m_1=0}^{\left\lfloor\frac{n_1}{2}\right\rfloor}\sum_{m_2=0}^{\left\lfloor\frac{n_2}{2}\right\rfloor}\sum_{m_3=0}^{\left\lfloor\frac{n_3}{2}\right\rfloor} c(m,n)\begin{bmatrix} n_1 \\ m_1 \end{bmatrix}\begin{bmatrix} n_2 \\ m_2 \end{bmatrix}\begin{bmatrix} n_3 \\ m_3 \end{bmatrix} r^{2m} x^{n_1-2m_1} y^{n_2-2m_2} z^{n_3-2m_3}$$

$$= \sum_{m_1=0}^{\left\lfloor\frac{n_1}{2}\right\rfloor}\sum_{m_2=0}^{\left\lfloor\frac{n_2}{2}\right\rfloor}\sum_{m_3=0}^{\left\lfloor\frac{n_3}{2}\right\rfloor} \text{Part I}\cdot\text{Part II}$$

where $n = n_1+n_2+n_3$, $m = m_1+m_2+m_3$,

$$\text{Part I} = c(m,n)\begin{bmatrix} n_1 \\ m_1 \end{bmatrix}\begin{bmatrix} n_2 \\ m_2 \end{bmatrix}\begin{bmatrix} n_3 \\ m_3 \end{bmatrix},$$

and

$$\text{Part II} = (-1)^n r^{-2n-1} r^{2m} x^{n_1-2m_1} y^{n_2-2m_2} z^{n_3-2m_3}.$$

The tensor can be constructed by the multiplication of the respective elements in Part I and Part II in the sequence determined by the three layer summation in Eq. (10). Part I does not depend on r. One only needs to calculate it once and save it in the memory for later use. Part II has to be calculated multiple times for different values of r. In the FMM, if we translate the multipole expansion in box a into the local expansion in box b, then a and b are in the same size, and the parent box of a and the parent box of b attaches each other, but a and b do not attach each other. This relation limits the possible value of r. Without loss of generality, we assume the length of the edge of box a and b is one, the center of a is $(0,0,0)$, and the center of b is (i,j,k). The possible values of i, j, k, are 0, ±1, ±2, and ±3. At least one of them is equal to or greater than two. The symmetry of Part II can be used to reduce the calculations. If we switch the sign of any coordinates, we only need to switch the sign of respective elements in Part II. For example, if change (i,j,k) into $(-i,j,k)$, we need to change the sign of all the elements in which x is in odd order. If we swap the value of two coordinates, e.g. change (i,j,k) into (j,i,k), we just need to change the sequence of the elements. Excluding all the duplications due to the symmetry, the value of (i,j,k) are

$$i\in[2,3],\ \ j\in[0,\ldots,i],\ \ k\in[0,\ldots,j].$$

There are totally 16 possible values of (i,j,k), all of which are calculated and saved in memory for later use. For level l to a finer level $l+1$, the value of (i,j,k) shrinks by half. If we have already calculated the M2L transition tenor for level l, the one for level $l+1$ can be easily constructed by multiplying each element in the old tenor by 2^{n+1}, because the total order of x, y, z, r in each element is $-n-1$ according to Eq. (12).

References

[1] A.W. Appel, An efficient program for many-body simulation, SIAM J. Sci. Stat. Comput. 6 (1) (1985) 85–103.
[2] J. Barnes, P. Hut, A hierarchical $O(N\log N)$ force-calculation algorithm, Nature 324 (4) (1986).
[3] J.A. Board, Z.S. Hakura, W.D. Elliott, D.C. Gray, W.J. Blanke, J.F. Leathrum, Scalable implementations of multipole-accelerated algorithms for molecular dynamics, in: Proceedings of the Scalable High-Performance Computing Conference, IEEE, 1994, pp. 87–94.
[4] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (2) (1987) 325–348.
[5] L. Greengard, V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, Acta Numer. 6 (1) (1997) 229–269.
[6] J. Carrier, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, SIAM J. Sci. Stat. Comput. 9 (1988) 669–686.
[7] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, J. Comput. Phys. 155 (2) (1999) 468–498.
[8] L.F. Greengard, J. Huang, A new version of the fast multipole method for screened Coulomb interactions in three dimensions, J. Comput. Phys. 180 (2) (2002) 642–658.
[9] F. Zhao, An $O(N)$ Algorithm for Three-Dimensional N-Body Simulations, Technical report, MIT Artificial Intelligence Lab, 1987.
[10] C.R. Anderson, An implementation of the fast multipole method without multipoles, SIAM J. Sci. Stat. Comput. 13 (1992) 923.
[11] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys. 196 (2) (2004) 591–626.
[12] W. Fong, E. Darve, The black-box fast multipole method, J. Comput. Phys. 228 (23) (2009) 8712–8725.
[13] B. Shanker, H. Huang, Accelerated Cartesian expansions – a fast method for computing of potentials of the form $r^{-\nu}$ for all real ν, J. Comput. Phys. 226 (1) (2007) 732–753.
[14] J. Applequist, Cartesian polytensors, J. Math. Phys. 24 (4) (1983) 736–741.
[15] J. Applequist, Fundamental relationships in the theory of electric multipole moments and multipole polarizabilities in static fields, Chem. Phys. 85 (2) (1984) 279–290.
[16] J. Applequist, Traceless Cartesian tensor forms for spherical harmonic functions: new theorems and applications to electrostatics of dielectric media, J. Phys. A, Math. Gen. 22 (1989) 4303.
[17] E.W. Hobson, The Theory of Spherical and Ellipsoidal Harmonics, CUP Archive, 1931.
[18] L.C. Andrews, Special Functions for Engineers and Applied Mathematicians, Macmillan, 1985.
[19] A. Kloeckner, https://pypi.org/project/pyfmmlib/.

[20] L. Greengard, Z. Gimbutas, FMMLIB2D User's guide, version 1.2, 2012.
[21] L. Greengard, Z. Gimbutas, FMMLIB3D User's guide Version 1.2, 2012.
[22] S. Abeyratne, B. Erdelyi, Optimization of the multipole to local translation operator in the adaptive fast multipole method, in: Proceedings of PAC2013, Pasadena, CA, USA, 2013.
[23] R. Yokota, L.A. Barba, ExaFMM user's manual, 2011.
[24] L.A. Barba, R. Yokota, How will the fast multipole method fare in the exascale era?, SIAM News 46 (6) (2013).
[25] R. Yokota, Personal communication, 2018.