Generalized and efficient algorithm for computing multipole energies and gradients based on Cartesian tensors

Dejun Lin

Citation: The Journal of Chemical Physics **143**, 114115 (2015); doi: 10.1063/1.4930984
View online: http://dx.doi.org/10.1063/1.4930984
View Table of Contents: http://scitation.aip.org/content/aip/journal/jcp/143/11?ver=pdfcov
Published by the AIP Publishing

Generalized and efficient algorithm for computing multipole energies and gradients based on Cartesian tensors

Dejun Lin[a)]
Department of Biochemistry and Biophysics, University of Rochester Medical Center, Rochester, New York 14642, USA

(Received 30 July 2015; accepted 2 September 2015; published online 21 September 2015)

Accurate representation of intermolecular forces has been the central task of classical atomic simulations, known as molecular mechanics. Recent advancements in molecular mechanics models have put forward the explicit representation of permanent and/or induced electric multipole (EMP) moments. The formulas developed so far to calculate EMP interactions tend to have complicated expressions, especially in Cartesian coordinates, which can only be applied to a specific kernel potential function. For example, one needs to develop a new formula each time a new kernel function is encountered. The complication of these formalisms arises from an intriguing and yet obscured mathematical relation between the kernel functions and the gradient operators. Here, I uncover this relation via rigorous derivation and find that the formula to calculate EMP interactions is basically invariant to the potential kernel functions as long as they are of the form $f(r)$, i.e., any Green's function that depends on inter-particle distance. I provide an algorithm for efficient evaluation of EMP interaction energies, forces, and torques for any kernel $f(r)$ up to any arbitrary rank of EMP moments in Cartesian coordinates. The working equations of this algorithm are essentially the same for any kernel $f(r)$. Recently, a few recursive algorithms were proposed to calculate EMP interactions. Depending on the kernel functions, the algorithm here is about 4–16 times faster than these algorithms in terms of the required number of floating point operations and is much more memory efficient. I show that it is even faster than a theoretically ideal recursion scheme, i.e., one that requires 1 floating point multiplication and 1 addition per recursion step. This algorithm has a compact vector-based expression that is optimal for computer programming. The Cartesian nature of this algorithm makes it fit easily into modern molecular simulation packages as compared with spherical coordinate-based algorithms. A software library based on this algorithm has been implemented in C++11 and has been released. [http://dx.doi.org/10.1063/1.4930984]

I. INTRODUCTION

Classical molecular mechanics simulations use empirical potential energy functions, called *force fields*, to model molecular interactions at the atomic level. Force fields are typically divided into bonded and nonbonded terms, the latter of which are usually approximations of electrostatics, dispersion, and repulsion interactions. The most commonly used models for the nonbonded interactions are Coulomb and Lennard-Jones potentials. These potentials are, in general, functions of the nuclear positions and atomic properties such as the electron density, the latter of which are called parameters and are determined from quantum mechanical calculations.

Historically, the parameters for the Coulomb potential are the so-called atomic "partial charges," which are fractional atom-centered charges representing the continuous electron density. Although the partial-charge model simplifies the calculation and is easy to implement, its failure to reproduce the molecular electrostatic potential has been known for decades,[1–9] especially in the biomolecular simulation field. This issue mainly arises from the fact that molecular orbitals, in general, are not isotropic in space as implicitly assumed by the partial-charge model. We refer the reader to recent reviews[10,11] on this issue. Most notably, this issue is more pronounced in the case of a coarse-grained force field, where a group of atoms are modeled as a super-atom whose partial charges sum to neutrality. Under this circumstance, the complete loss of electrostatics between neutral super-atoms has been shown to cause simulation artifacts.[12,13] One intuitive way to improve the electrostatic model is to include higher order electric multipole (EMP) moments to better represent the electron cloud. Significant effort has been devoted to the development of force fields that explicitly represent permanent and polarizable point atomic EMP moments[14–20] as well as continuous Gaussian EMP moments.[21,22] EMP moments have also been used in recent coarse-grained force fields for proteins[23] and lipids.[24] Very recently, a polarizable Gaussian atomic EMP model was used in refining x-ray crystal structure,[25–28] where the authors found that an expansion up to rank-4 EMP (hexadecapole) moments are sometimes necessary to describe bond electron density in tetrahedral geometries.

Another concern for calculating molecular potential is long-range interactions. This is most prominent in evaluating

a)Author to whom correspondence should be addressed. Electronic mail: dejun_lin@gmail.com

the Coulomb potential, although recently there are also reports on the importance of long-range dispersion interaction in computing fluid-fluid interfacial properties.[29–32] One of the first and yet most popular ways to evaluate the long-range interactions is Ewald summation and the particle-mesh (PM) method based on it. The details of this method have been extensively reviewed.[10,33,34] While the formalism for the PM method in the case of partial-charge model as well as the implementation has been established for several decades, the equivalent for the case of higher-rank EMP moments was only developed recently.[15,35–37] There are also real-space methods based on the Wolf summation[38] that have been developed recently[39,40] which explicitly incorporate EMP moments.

While the importance of higher-rank EMP moments in improving force fields has become more appreciated, the additional equations beyond the partial-charge model have not been cast into a easily recognizable form. Also, the formalism to compute the EMP-EMP interaction energies and forces is not generalized to encompass the variety of potential energy functions used under different situations, especially when additional, not necessarily complicated, mathematical or numerical manipulation is needed in treatment of long-range interactions. For example, the formalism to compute Ewald summation up to the quadrupole level was originally proposed by Smith[41] but later recast by Aguado and Madden,[35] in a completely different form and in either case, but neither formalism is readily generalizable to support EMP moments of higher ranks. Moreover, these formalisms use sparse and obscure expressions that have to be written out explicitly in computer program, which makes them difficult to implement and debug.

The other aspect of the calculation involving EMP is efficiency. The canonical tensor-based method for calculating EMP interaction energies and forces has a vector-matrix-vector bilinear form and requires populating the central matrix. However, it is generally not the most efficient way to first populate the matrix and then carry out the bilinear contraction (or vector-matrix multiplication); I will show later that the algorithm developed here needs significantly fewer floating point operations to evaluate EMP interaction energies and forces than several algorithms developed recently where the central matrix is populated by recursion. Specifically, the algorithm reported here is about 4 times faster (in terms of the required number of floating-point operations) than the McMurchie-Davidson algorithm prosed by Sagui *et al.*[15] and about 16 times faster than another one proposed by Boateng and Todorov[37] for evaluating the damped Coulomb potential used in Ewald summation at the hexadecapole level. It is even more competitive than a theoretically ideal recursion scheme where only 1 multiplication and 1 addition are needed to construct each element of the central matrix.

In this study, I summarize the problems in electric multipole-multipole interaction and provide an efficient algorithm to evaluate the multipole interaction energies, forces, and torques in Cartesian coordinates for any kernel function $f(r)$. This study is based on the work of Applequist[42] and Burgos and Bonadeo[43] on Cartesian tensor applied in the solution to Poisson's equation, i.e., the Coulomb potential function. This algorithm is superior to the canonical tensor-based one in terms of computational efficiency and has a compact vector-based expression that makes the implementation in computer programs very easy. I compare its computational complexity to that of several recursive algorithms developed recently and show that the current method is, in general, more efficient. A C++11 template library based on this algorithm has been developed and released.

The rest of this paper is organized as follows: Section II presents the basics of multipole expansion and multipole interaction with a generalization to a wide range of kernel functions. Sections III and IV present a derivation to an efficient algorithm to calculate multipole interactions via any kernel function of the form $f(r)$ and show that its mathematical expression is invariant with the kernel function. Sections V and VI provide insight into how the algorithm works and its novelty, respectively. Section VII shows how one can apply the algorithm to Ewald summation. Section VIII provides complexity analysis of this algorithm in comparison to various recursive algorithms developed recently. In Section IX, I will describe how to implement the algorithm in computer programs.

II. THE BASIC PROBLEM

First, I refer the reader to the Appendix for an explanation of the notation I use throughout this paper. The Poisson's equation for the electrostatic potential $\phi(\mathbf{r})$ of a given source $\rho(\mathbf{r})$ is

$$-\Delta\phi(\mathbf{r}) = 4\pi\rho(\mathbf{r}), \tag{2.1}$$

where $\Delta \equiv \nabla^2$ is the Laplace operator. For convenience, I denote the pair of $\phi(\mathbf{r})$ and $\rho(\mathbf{r})$ as

$$\phi(\mathbf{r}) \overset{Possion}{\rightleftharpoons} \rho(\mathbf{r}). \tag{2.2}$$

The Green's function for Equation (2.1) is

$$G(\mathbf{r}) = \frac{1}{r}. \tag{2.3}$$

Now consider a cluster of N sources around a point (called the electric multipole or EMP site): $\mathbf{r_j} \in \mathbb{R}^3$,

$$\rho_j(\mathbf{r}) \equiv \sum_{k=1}^{N} q_k \delta(\mathbf{r}-\mathbf{r}_k) * \gamma_j(\mathbf{r}), \tag{2.4}$$

where $\delta(\mathbf{r})$ is the Dirac delta function, $\gamma_j(\mathbf{r})$ represents the "shape" of the source function, and $*$ denotes convolution. Since convolution commutes with Δ, we have for $\phi_j \overset{Possion}{\rightleftharpoons} \rho_j$,

$$\phi_j(\mathbf{r}) = \sum_{k=1}^{N} q_k \delta(\mathbf{r}-\mathbf{r}_k) * \gamma_j(\mathbf{r}) * G(\mathbf{r}). \tag{2.5}$$

Define $\mathbf{d}_{j_k} \equiv \mathbf{r}_k - \mathbf{r}_j \ \forall k = 1,2,\ldots,N$ so that at a point $\mathbf{r}$ outside the cluster, i.e., $|\mathbf{r}-\mathbf{r}_j| > max\{|\mathbf{d}_{j_k}|\}$, the Taylor series of Equation (2.5) about $(\mathbf{d}_{j_1},\ldots,\mathbf{d}_{j_N}) = (\mathbf{0},\ldots,\mathbf{0})$ converges,

$$\phi_j(\mathbf{r}) = \sum_{n=0}^{+\infty} \boldsymbol{\mu}_j^{(n)} \cdot n \cdot \boldsymbol{\nabla}_j^{(n)} \delta(\mathbf{r}-\mathbf{r}_j) * \gamma_j(\mathbf{r}) * G(\mathbf{r}), \tag{2.6}$$

where $\cdot n \cdot$ denotes n-fold contraction (see the Appendix for a description of the notation used) and $\boldsymbol{\mu}_j^{(n)}, n = 0,1,\ldots,N$ are

the Cartesian multipoles at site j,

$$\boldsymbol{\mu}_j^{(\mathrm{n})} \equiv \frac{1}{n!}\sum_{k=1}^{N} q_{j_k}\mathbf{d}_{j_k}^{(n)}, \tag{2.7}$$

with $\mathbf{d}_{j_k}^{(n)}$ being the n-fold tensor product of $\mathbf{d}_{j_k}$:

$$\mathbf{d}_{j_k}^{(n)} \equiv \overbrace{\mathbf{d}_{j_k}\otimes\mathbf{d}_{j_k}\otimes\cdots\otimes\mathbf{d}_{j_k}}^{\text{a total of } n \text{ terms}} \tag{2.8}$$

and $\boldsymbol{\nabla}_j^{(n)}$ is the order-n gradient (with respect to the coordinates of site j) operator

$$\boldsymbol{\nabla}^{(n)} \equiv \boldsymbol{\nabla}^{(n)}_{\alpha_1\alpha_2\ldots\alpha n} \equiv \frac{\partial^n}{\partial\alpha_1\partial\alpha_2\ldots\partial\alpha_n}, \quad \text{with } \alpha_k = x,\ y \text{ or } z\ \ \forall k\in\mathbb{Z}^+. \tag{2.9}$$

Similarly, the electrostatic energy between EMP site j and another site i is

$$U(\mathbf{r}_i-\mathbf{r}_j) \equiv \int_{\text{all space}}\rho_i\phi_j(\mathbf{r})\mathrm{d}\mathbf{r} = \int_{\text{all space}}\sum_{m=0}^{+\infty}\boldsymbol{\mu}_i^{(\mathrm{m})}\cdot m\cdot\boldsymbol{\nabla}_i^{(m)}\delta(\mathbf{r}-\mathbf{r}_i)*\gamma_i(\mathbf{r})\phi_j(\mathbf{r})\mathrm{d}\mathbf{r}. \tag{2.10}$$

We define a rank-n *point* EMP operator m_j and the corresponding *shaped* operator m_j^γ of site j as

$$\mathsf{m}_j = \mathsf{m}_j(\mathbf{r}) \equiv \sum_{n=0}^{+\infty}\boldsymbol{\mu}_j^{(\mathrm{n})}\cdot n\cdot\boldsymbol{\nabla}_j^{(n)}\delta(\mathbf{r}-\mathbf{r}_j), \tag{2.11}$$

$$\mathsf{m}_j^\gamma \equiv \mathsf{m}_j * \gamma_j, \tag{2.12}$$

so that $\phi_j(\mathbf{r}) \overset{Possion}{\rightleftharpoons} \mathsf{m}_j^\gamma$ as in Equation (2.6) and

$$U(\mathbf{r}_i-\mathbf{r}_j) = \int_{\text{all space}}\mathsf{m}_i^\gamma\phi_j(\mathbf{r})\mathrm{d}\mathbf{r} \tag{2.13}$$

as in Equation (2.10), which means that m_j (m_j^γ) acts as a point (shaped) EMP density distribution of site j and the electrostatic energies, forces, and torques between a pair EMP sites can be obtained from that between a pair of charge density distributions with the charge density replaced with the EMP operator defined in Equation (2.11). For example, the electrostatic energy between 2 point-EMP sites i and j is (by inserting Equation (2.6) into Equation (2.10))

$$U(\mathbf{r}_i-\mathbf{r}_j) = \sum_{m=0}^{+\infty}\boldsymbol{\mu}_i^{(\mathrm{m})}\cdot m\cdot\boldsymbol{\nabla}_i^{(m)}\sum_{n=0}^{+\infty}\boldsymbol{\mu}_j^{(\mathrm{n})}\cdot n\cdot\boldsymbol{\nabla}_j^{(n)}I_{ij} = \sum_{m=0}^{+\infty}\sum_{n=0}^{+\infty}(-1)^m\boldsymbol{\mu}_i^{(\mathrm{m})}\cdot m\cdot\boldsymbol{\nabla}_j^{(m+n)}I_{ij}\cdot n\cdot\boldsymbol{\mu}_j^{(\mathrm{n})}, \tag{2.14}$$

where

$$I_{ij} = I(\mathbf{r}_i-\mathbf{r}_j) \equiv \int_{\text{all space}}\gamma_i(\mathbf{r}-\mathbf{r}_i)\gamma_j(\mathbf{r}-\mathbf{r}_j)*G(\mathbf{r})\mathrm{d}\mathbf{r} = \int_{\text{all space}}(G*\gamma_j)(\mathbf{u}+\mathbf{r}_i-\mathbf{r}_j)\gamma_i(\mathbf{u})\mathrm{d}\mathbf{u} \tag{2.15}$$

and the force on site i is

$$\mathbf{F}(\mathbf{r}_i-\mathbf{r}_j) = -\boldsymbol{\nabla}_i^{(1)}U(\mathbf{r}_i-\mathbf{r}_j) = \sum_{m=0}^{+\infty}\sum_{n=0}^{+\infty}(-1)^m\boldsymbol{\mu}_i^{(\mathrm{m})}\cdot m\cdot\boldsymbol{\nabla}_j^{(m+n+1)}I_{ij}\cdot n\cdot\boldsymbol{\mu}_j^{(\mathrm{n})} \tag{2.16}$$

and the torque on site i is (see Section S1 in the supplementary material for the derivation[57])

$$\mathbf{T}(\mathbf{r}_i-\mathbf{r}_j) = \sum_{m=0}^{+\infty}\sum_{n=0}^{+\infty}\boldsymbol{\epsilon}^{(3)}\cdot 2\cdot\left[(-1)^{m+1}(m+1)\boldsymbol{\mu}_i^{(\mathrm{m+1})}\cdot m\cdot\boldsymbol{\nabla}_j^{(m+n+1)}I_{ij}\cdot n\cdot\boldsymbol{\mu}_j^{(\mathrm{n})}\right], \tag{2.17}$$

where $\boldsymbol{\epsilon}^{(3)} \equiv \boldsymbol{\epsilon}^{(3)}_{ijk} = (i-j)(k-i)(j-k)/2$ is the Levi-Civita symbol. In general, we need to evaluate the bi-contraction of the form

$$\mathbf{A}^{(n)}\cdot n\cdot\boldsymbol{\nabla}^{(n+m)}f(\mathbf{r})\cdot m\cdot\mathbf{B}^{(m)}, \tag{2.18}$$

where the n-fold gradient of $f(\mathbf{r})$ exists.

For most applications of EMP moments, such as Ewald sum or the Fast Multipole Method[44,45] (FMM), f is a function of inter-particle distance only. If we restrict f to $f \equiv f(\mathbf{r}^2)$, we can rewrite Equation (2.18) as

$$\mathbf{A}^{(n)}\cdot n\cdot\boldsymbol{\nabla}^{(n+m)}f(\mathbf{r}^2)\cdot m\cdot\mathbf{B}^{(m)}. \tag{2.19}$$

In Sec. III, I will develop an expansion of $\boldsymbol{\nabla}^{(n)}f(\mathbf{r}^2)$ in terms of a function of $\mathbf{r}^{(n)} \equiv \overbrace{\mathbf{r}\otimes\mathbf{r}\otimes\cdots\otimes\mathbf{r}}^{\text{a total of } n \text{ terms}}$ so that expression (2.18) can be evaluated in the same expansion in terms of contractions of the form $\mathbf{A}^{(n)}\cdot k\cdot\mathbf{r}^{(k)}$ and $\mathbf{B}^{(m)}\cdot k\cdot\mathbf{r}^{(k)}$ without the need to populate the central tensor $\boldsymbol{\nabla}^{(n+m)}f(\mathbf{r}^2)$. However, I want to remind the reader here that the definitions of Equations (2.11) and (2.12) are independent of the linear operator Δ and its Green's function and are valid as long as the Taylor series in

Equation (2.6) converges. That said, I do assume the linear operator is translation invariant, which is true for simulations of particle systems. This means we can apply it to other linear equations (with appropriate shaping function $\gamma(\mathbf{r})$ to guarantee the convergence of Equation (2.15)) such as Yukawa's equation,

$$-(\Delta - \kappa^2)\phi(\mathbf{r}) = 4\pi\rho(\mathbf{r}), \quad \kappa \in \mathbb{R}^+ \tag{2.20}$$

or the Helmholtz equation,

$$-(\Delta + \kappa^2)\phi(\mathbf{r}) = 4\pi\rho(\mathbf{r}), \quad \kappa \in \mathbb{R}^+. \tag{2.21}$$

III. THE *n*-FOLD GRADIENT OF $f(\mathbf{r}^2)$

The direct approach to evaluating expression (2.19) is to first populate the tensor $\boldsymbol{\nabla}^{(n+m)}f(\mathbf{r}^2)$ as a matrix and then perform a vector-matrix-vector multiplication. However, the complexity of this approach could be very high depending on the complexity of calculating $\boldsymbol{\nabla}^{(n+n)}f(\mathbf{r}^2)$. For example, the complexity of populating $\boldsymbol{\nabla}^{(n+m)}\frac{1}{r}$ in $\mathbf{A}^{(n)} \cdot n \cdot \boldsymbol{\nabla}^{(n+n)}\frac{1}{r} \cdot n \cdot \mathbf{B}^{(n)}$ is on the order of n^6 based on the algorithm published previously.[42]

On the other hand, if we can decompose $\boldsymbol{\nabla}^{(n+n)}f(\mathbf{r}^2)$ into $\overbrace{\mathbf{t}\otimes\mathbf{t}\otimes\cdots\otimes\mathbf{t}}^{\text{a total of } 2n \text{ terms}}$, where $\mathbf{t} \equiv \mathbf{t}(\mathbf{r})$ is a simple function of $\mathbf{r}$, the order-$2n$ bi-contraction can be split into two order-n contractions and the complexity is reduced to n^3. Although such a decomposition does not exist, in general, a similar "divide-and-conquer" approach has been proposed previously for $f(\mathbf{r}^2) = \frac{1}{r}$ using a diagrammatic method to evaluate EMP interactions.[43] Here, we take one step forward and generalize the approach to any function $f \equiv f(\mathbf{r}^2)$ where f is a function of the squared norm of $\mathbf{r}$.

First, we need the help of the following definition:

Definition 3.1. Define $\mathbf{r}^{(n)}\boldsymbol{\delta}^{(s)}$ *as a tensor of rank* $n + 2s$,

$$\mathbf{r}^{(n)}\boldsymbol{\delta}^{(s)} \equiv \sum_{\mathrm{P}^s_{n+2s}\{\alpha_1\ldots\alpha_{n+2s}\}} r^{(n)}_{\alpha_1\ldots\alpha_n}\delta_{\alpha_{n+1}\alpha_{n+2}}\delta_{\alpha_{n+3}\alpha_{n+4}}\cdots\delta_{\alpha_{n+2s-1}\alpha_{n+2s}}, \tag{3.1}$$

where δ_{ij} *is the Kronecker delta and the sum is over all the permutations of the set of index* $\{\alpha_1\ldots\alpha_{n+2s}\}$ *that give distinct terms and there are a total of* $\mathrm{P}^s_{n+2s} \equiv (n+2s)!/(2^s s! n!)$ *terms. The permutations in the sum guarantee that* $\mathbf{r}^{(n)}\boldsymbol{\delta}^{(s)}$ *is totally symmetric.*

The following lemmas are direct applications of Definition 3.1:

Lemma 3.1. Given a tensor $\mathbf{r}^{(n)}\boldsymbol{\delta}^{(1)}$ *of rank* $n + 2$, *as in Definition 3.1 and one of its index* $i \in [0, n+2]$, *we have the following equality:*

$$\mathbf{r}^{(n)}\boldsymbol{\delta}^{(1)} = r_i\mathbf{r}^{(n-1)}\boldsymbol{\delta}^{(1)} + \delta_i\mathbf{r}^{(n)}, \tag{3.2}$$

where $\delta_i\mathbf{r}^{(n)}$ *is* $\mathbf{r}^{(r)}\boldsymbol{\delta}^{(1)}$ *with a specific index i fixed on* $\boldsymbol{\delta}$.

Proof. This can be obtained trivially by realizing that the sum in Definition 3.1 can be split into a sum over index i attached to $\mathbf{r}$ plus another sum where i is attached to δ. □

Lemma 3.2.

$$\nabla_{\alpha_{n+1}}\mathbf{r}^{(n)} = \delta_{\alpha_{n+1}}\mathbf{r}^{(n-1)}. \tag{3.3}$$

Proof. Since $\mathbf{r}^{(n)} = r_{\alpha_1}r_{\alpha_2}\ldots r_{\alpha_n}$, the chain rules give us

$$\nabla_{\alpha_{n+1}}\mathbf{r}^{(n)} = \sum_{i=\alpha_1}^{\alpha_n}\delta_{i\alpha_{n+1}}\mathbf{r}^{(n-1)} = \delta_{\alpha_{n+1}}\mathbf{r}^{(n-1)}. \tag{3.4}$$

□

With the help of Definition 3.1 and Lemmas 3.1 and 3.2, we arrive at the following theorem:

Theorem 3.1. *Given a real-valued function* f *on* $\mathbb{R}_{\geq 0}$, *whose n-fold gradient exists, its n-fold gradient can be expanded into*

$$\boldsymbol{\nabla}^{(n)}f(\mathbf{r}^2) = \sum_{k=0}^{\lfloor\frac{n}{2}\rfloor} 2^{n-k}F_{n-k}(\mathbf{r}^2)\mathbf{r}^{(n-2k)}\boldsymbol{\delta}^{(k)}, \tag{3.5}$$

where $\lfloor\frac{n}{2}\rfloor$ *is the biggest integer no larger than* $n/2$ *and*

$$F_k \equiv \frac{\mathrm{d}^k f}{\mathrm{d}(\mathbf{r}^2)^k}. \tag{3.6}$$

Proof. We will proceed by induction. Equation (3.5) is obviously true for $n = 0$ and we have

$$\begin{aligned}
\boldsymbol{\nabla}^{(n+1)}f(\mathbf{r}^2) &= \nabla_{\alpha_{n+1}}\boldsymbol{\nabla}^{(n)}f(\mathbf{r}^2)\\
&= \sum_{k=0}^{\lfloor\frac{n}{2}\rfloor} 2^{n-k}(\nabla_{\alpha_{n+1}}F_{n-k}\mathbf{r}^{(n-2k)}\boldsymbol{\delta}^{(k)} + F_{n-k}\nabla_{\alpha_{n+1}}\mathbf{r}^{(n-2k)}\boldsymbol{\delta}^{(k)})\\
&= \sum_{k=0}^{\lfloor\frac{n}{2}\rfloor}(2^{n+1-k}F_{n+1-k}r_{\alpha_{n+1}}\mathbf{r}^{(n-2k)}\boldsymbol{\delta}^{(k)} + 2^{n-k}F_{n-k}\delta_{\alpha_{n+1}}\mathbf{r}^{(n-2k-1)}\boldsymbol{\delta}^{(k)}),
\end{aligned}$$

where the 2nd equality follows directly from Lemma 3.2. Note that for $k = 0$ in the sum, the 1st term on the right-hand side is just $2^{n+1}F_{n+1}\mathbf{r}^{(n+1)}$; for $k = \lfloor\frac{n}{2}\rfloor$, the 2nd term on the right-hand side is just $2^{n-k}F_{n-k}\boldsymbol{\delta}^{(k+1)}$ if n is odd and it vanishes if n is even;

for $0 < k < \lfloor \frac{n}{2} \rfloor$, the kth and the $k+1$th term can be merged as

$$2^{n+1-k}F_{n+1-k}r_{\alpha_{n+1}}\mathbf{r}^{(n-2k)}\boldsymbol{\delta}^{(k)} + (2^{n-k}F_{n-k}\delta_{\alpha_{n+1}}\mathbf{r}^{(n-2k-1)}\boldsymbol{\delta}^{(k)} + 2^{n-k}F_{n-k}r_{\alpha_{n+1}}\mathbf{r}^{(n-2k-2)}\boldsymbol{\delta}^{(k+1)}) + 2^{n-k-1}F_{n-k-1}\delta_{\alpha_{n+1}}\mathbf{r}^{(n-2k-3)}\boldsymbol{\delta}^{(k+1)}$$
$$= 2^{n+1-k}F_{n+1-k}r_{\alpha_{n+1}}\mathbf{r}^{(n-2k)}\boldsymbol{\delta}^{(k)} + 2^{n+1-(k+1)}F_{(n+1)-(k+1)}\mathbf{r}^{(n+1-2(k+1))}\boldsymbol{\delta}^{(k+1)} + 2^{n+1-(k+2)}F_{n+1-(k+2)}\delta_{\alpha_{n+1}}\mathbf{r}^{(n+1-2(k+2))}\boldsymbol{\delta}^{(k+1)}$$

which follows from Lemma 3.1. Thus, we have for

$$n = 2g, \quad g \in \mathbb{N}^0, \ \boldsymbol{\nabla}^{(n+1)}f(\mathbf{r}^2) = \sum_{k=0}^{g} 2^{n+1-k}F_{n+1-k}\mathbf{r}^{(n+1-2k)}\boldsymbol{\delta}^{(k)}$$

and for

$$n = 2g+1, \quad g \in \mathbb{N}^0,$$
$$\boldsymbol{\nabla}^{(n+1)}f(\mathbf{r}^2) = \sum_{k=0}^{g} 2^{n+1-k}F_{n+1-k}\mathbf{r}^{(n+1-2k)}\boldsymbol{\delta}^{(k)} + 2^{n-g}F_{n-g}\boldsymbol{\delta}^{(g+1)}.$$

In general, we have

$$\boldsymbol{\nabla}^{(n+1)}f(\mathbf{r}^2) = \sum_{k=0}^{\lfloor \frac{n+1}{2} \rfloor} 2^{n+1-k}F_{n+1-k}\mathbf{r}^{(n+1-2k)}\boldsymbol{\delta}^{(k)}$$

which is Equation (3.5) with n replaced by $n+1$. □

Because $\boldsymbol{\nabla}^{(n)}f(\mathbf{r}^2)$ is totally symmetric, it is more convenient to rewrite Equation (3.5) in its compressed tensor form. Notice that the non-vanishing terms in $\mathbf{r}^{(n-2k)}\boldsymbol{\delta}^{(k)}$ are those of the form $r^{(n-2k)}(n_1-2k_1, n_2-2k_2, n_3-2k_3) \equiv r_x^{n_1-2k_1}r_y^{n_2-2k_2}r_z^{n_3-2k_3}$, where $\sum_{i=1}^3 n_i = n$, $\sum_{i=1}^3 k_i = k$, and $2k_i \le n_i$. The redundancy of this term in the sum $\mathrm{P}^s_{n+2s}\{\alpha_1 \dots \alpha_{n+2s}\}$ is the number of ways k_1 distinct pairs of index $\alpha_i\alpha_j$ can be selected from those in $\alpha_1 \dots \alpha_n$ which are assigned the value of 1, which is $\mathrm{P}^{k_1}_{n_1} \equiv n_1!/(2^{k_1}k_1!(n_1-2k_1)!)$, times that number for k_2 and k_3. Thus,

$$r^{(n-2k)}\delta^{(k)}(n_1,n_2,n_3) = \sum_{\{k;k_1,k_2,k_3\}} P^{k_1}_{n_1}P^{k_2}_{n_2}P^{k_3}_{n_3}r^{(n-2k)}(n_1-2k_1, n_2-2k_2, n_3-2k_3), \tag{3.7}$$

where the sum $\sum_{\{k;k_1,k_2,k_3\}}$ is over all the combination of k_1, k_2, k_3 whose sum is k. Insert this into Equation (3.5) and we get

$$\boldsymbol{\nabla}^{(n)}f(n_1,n_2,n_3) = \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} 2^{n-k}F_{n-k}r^{(n-2k)}\delta^{(k)}(n_1,n_2,n_3) = \sum_{k_1=0}^{\lfloor \frac{n_1}{2} \rfloor}\sum_{k_2=0}^{\lfloor \frac{n_2}{2} \rfloor}\sum_{k_3=0}^{\lfloor \frac{n_3}{2} \rfloor} 2^{n-k}F_{n-k}P^{k_1}_{n_1}P^{k_2}_{n_2}P^{k_3}_{n_3}r^{(n-2k)}(n_1-2k_1, n_2-2k_2, n_3-2k_3). \tag{3.8}$$

One can verify that when $f(\mathbf{r}^2) = \sqrt{\frac{1}{\mathbf{r}^2}}$, Equations (3.5) and (3.8) reduce to the solid spherical harmonics of degree n in Cartesian coordinates, which is central to EMP interactions and has been given before.[42,43] Also, if $f(\mathbf{r}^2) = \int_0^1 \exp(-\mathbf{r}^2t^2)\mathrm{d}t$, which is the Boys function of order 0, Equations (3.5) and (3.8) reduce to the Hermite Coulomb integral, which plays an important role in evaluation of quantum molecular integrals in Gaussian type orbitals theory.[46,47] Notice that Definition 3.1 and Theorem 3.1 are independent of the dimension N of $\mathbf{r}$, which means they can be applied to cases $N \ge 3$. For example, if $f(\mathbf{r}^2) = 1/(2\pi)^{N/2}\exp(-\mathbf{r}^2/2)$, Equation (3.5) reduces to the N-dimensional Hermite polynomials.[48]

The factorization of the derivatives of f in Theorem 3.1 provides a way to evaluate expression (2.18) without the need to populate the central tensor since $\mathbf{r}^{(n-2k)}\boldsymbol{\delta}^{(k)}$ is a simple function of $\mathbf{r}$ only. This is given in Section IV.

IV. EVALUATION OF THE BI-CONTRACTION FORM

Given Equation (3.5), evaluating expression (2.18) is equivalent to evaluating

$$\mathbf{A}^{(n)} \cdot n \cdot \mathbf{r}^{(m+n-2k)}\boldsymbol{\delta}^{(k)} \cdot m \cdot \mathbf{B}^{(m)} \tag{4.1}$$

for $\forall k \in [0, \lfloor \frac{m+n}{2} \rfloor]$, multiplying by the coefficients and summing over all k. The terms in expression (4.1) can be split into the following categories:

1. terms arising from $\mathbf{r}$ contracted with $\mathbf{A}^{(n)}$ or $\mathbf{B}^{(m)}$;
2. terms arising from δ_{ij} contracted with both $\mathbf{A}^{(n)}$ and $\mathbf{B}^{(m)}$ where i comes from $\mathbf{A}^{(n)}$ and j comes from $\mathbf{B}^{(m)}$;
3. terms arising from δ_{ij} contracted with $\mathbf{A}^{(n)}$, i.e., taking $\mathbf{A}^{(n)}$'s trace;
4. terms arising from δ_{ij} contracted with $\mathbf{B}^{(m)}$, i.e., taking $\mathbf{B}^{(m)}$'s trace.

If we let l_c, l_n, and l_m be the number of terms in categories 2–4, we arrive at the following expression:

$$\mathbf{A}^{(n)}\cdot n\cdot\mathbf{r}^{(m+n-2k)}\boldsymbol{\delta}^{(k)}\cdot m\cdot\mathbf{B}^{(m)}=\sum_{\{k;l_c,l_n,l_m\}}C\begin{pmatrix}m\ ,\ n\\ l_c,l_m,l_n\end{pmatrix}\left(\mathbf{A}^{(m:l_m)}\cdot O_m\cdot\mathbf{r}^{(O_m)}\right)\cdot l_c\cdot\left(\mathbf{B}^{(n:l_n)}\cdot O_n\cdot\mathbf{r}^{(O_n)}\right),\tag{4.2}$$

where $O_m\equiv m-2l_m-l_c$, $O_n\equiv n-2l_n-l_c$, and the sum is over all $l_c,l_n,l_m\in\mathbb{N}^0$ so that $l_c+l_n+l_m=k$. The coefficient $C\begin{pmatrix}m\ ,\ n\\ l_c,l_m,l_n\end{pmatrix}\equiv \mathrm{P}_n^{l_n}\mathrm{P}_m^{l_m}\binom{n-2l_n}{l_c}\binom{m-2l_m}{l_c}l_c!$ is the degeneracy of each $\left(\mathbf{A}^{(m:l_m)}\cdot O_m\cdot\mathbf{r}^{(O_m)}\right)\cdot l_c\cdot\left(\mathbf{B}^{(n:l_n)}\cdot O_n\cdot\mathbf{r}^{(O_n)}\right)$ term in the sum. We can insert Equation (4.2) into (3.5) and (2.19) to get

$$\begin{aligned}&\mathbf{A}^{(n)}\cdot n\cdot\boldsymbol{\nabla}^{(n+m)}f(\mathbf{r}^2)\cdot m\cdot\mathbf{B}^{(m)}\\&\quad=\sum_{k=0}^{\lfloor\frac{n+m}{2}\rfloor}2^{m+n-k}F_{m+n-k}\sum_{\{k;l_c,l_n,l_m\}}C\begin{pmatrix}m\ ,\ n\\ l_c,l_m,l_n\end{pmatrix}\left(\mathbf{A}^{(m:l_m)}\cdot O_m\cdot\mathbf{r}^{(O_m)}\right)\cdot l_c\cdot\left(\mathbf{B}^{(n:l_n)}\cdot O_n\cdot\mathbf{r}^{(O_n)}\right).\end{aligned}\tag{4.3}$$

Similarly, we can obtain the expansion for the bi-contraction for forces as in Equation (2.16),

$$\begin{aligned}&\mathbf{A}^{(n)}\cdot n\cdot\boldsymbol{\nabla}^{(n+m+1)}f(\mathbf{r}^2)\cdot m\cdot\mathbf{B}^{(m)}\\&\quad=\sum_{k=0}^{\lfloor\frac{n+m+1}{2}\rfloor}2^{m+n+1-k}F_{m+n+1-k}\left[\sum_{\substack{\{k;l_c,l_n,l_m\}\\ \{1;q_n,q_m,q_r\}}}C\begin{pmatrix}m\ ,\ n,\\ l_c,l_m,l_n,\\ q_m,q_n\end{pmatrix}\mathbf{r}^{(q_r)}\otimes\left(\mathbf{A}^{(m:l_m)}\cdot O'_m\cdot\mathbf{r}^{(O'_m)}\right)\cdot l_c\cdot\left(\mathbf{B}^{(n:l_n)}\cdot O'_n\cdot\mathbf{r}^{(O'_n)}\right)\right]\end{aligned}\tag{4.4}$$

and torques as in Equation (2.17),

$$\begin{aligned}&\mathbf{B}^{(m+1)}\cdot m\cdot\boldsymbol{\nabla}^{(m+n+1)}f(\mathbf{r}^2)\cdot n\cdot\mathbf{A}^{(n)}\\&\quad=\sum_{k=0}^{\lfloor\frac{n+m+1}{2}\rfloor}2^{n+m+1-k}F_{n+m+1-k}\sum_{\substack{\{k;l_c,l_n,l_m\}\\ \{1;q_n,q_m,q_r\}}}\left[C\begin{pmatrix}m+1\ ,\ n,\\ l_c,l_m,l_n,\\ q_m,q_n\end{pmatrix}\mathbf{r}^{(q_r)}\otimes\left(\mathbf{B}^{((m+1):l_m)}\cdot O'_m\cdot\mathbf{r}^{(O'_m)}\right)\cdot l_c\cdot\left(\mathbf{A}^{(n:l_n)}\cdot O'_n\cdot\mathbf{r}^{(O'_n)}\right)\right],\end{aligned}\tag{4.5}$$

where the inner sum is over all $l_c,l_n,l_m,q_r,q_n,q_m\in\mathbb{N}^0$ so that $l_c+l_n+l_m=k$ and $q_r+q_n+q_m=1$ and $O'_m\equiv m-2lm-l_c-q_m$, $O'_n\equiv n-2ln-l_c-q_n$. The coefficient $C\begin{pmatrix}m\ ,\ n,\\ l_c,l_m,l_n,\\ q_m,q_n\end{pmatrix}\equiv\binom{m}{q_m}\binom{n}{q_n}\mathrm{P}^{l_m}_{m-q_m}\mathrm{P}^{l_n}_{n-q_n}\binom{m-q_m-2l_m}{l_c}\binom{n-q_n-2l_n}{l_c}l_c!$ is again the degeneracy. One can verify that Equations (4.3) and (4.4) in the case of $f(\mathbf{r}^2)=\sqrt{\frac{1}{\mathbf{r}^2}}$ and $f(\mathbf{r}^2)=\int_1^\infty\exp(-\mathbf{r}^2t^2)\mathrm{d}t$ are equivalent to those given by Burgos and Bonadeo,[43] and Smith,[41] respectively, although the expression given here is not limited to those cases.

There are a few things I want to point out here before changing the topic. First, note that when $\mathbf{A}^{(n)}$ or $\mathbf{B}^{(m)}$ is traceless, the terms corresponding to $l_n\neq0$ or $l_m\neq0$ vanish and Equations (4.3)–(4.5) are then much simpler. When both $\mathbf{A}^{(n)}$ and $\mathbf{B}^{(m)}$ are *non-traceless*, one can easily prove that if and only if $\boldsymbol{\nabla}^{(n+m)}f$ is totally symmetric and traceless would the following equations be true:

$$\boldsymbol{\nabla}^{(n+m)}f\ \cdot n\cdot\mathcal{T}_n\mathbf{A}^{(n)}=(2n-1)!!\boldsymbol{\nabla}^{(n+m)}f\cdot n\cdot\mathbf{A}^{(n)}\tag{4.6}$$

and

$$\begin{aligned}&\mathcal{T}_n\mathbf{A}^{(n)}\cdot n\cdot\boldsymbol{\nabla}^{(n+m)}f\cdot m\cdot\mathcal{T}_m\mathbf{B}^{(m)}\\&\quad=(2n-1)!!(2m-1)!!\mathbf{A}^{(n)}\cdot n\cdot\boldsymbol{\nabla}^{(n+m)}f\cdot m\cdot\mathbf{B}^{(m)},\end{aligned}\tag{4.7}$$

where !! indicates the double factorial and $\mathcal{T}_n$ is the so called "detracer" operator defined by Applequist,[42] which acts on a totally symmetric tensor $\mathbf{A}^{(n)}$ to give a totally symmetric and traceless tensor $\mathcal{T}_n\mathbf{A}^{(n)}$. In practice, the traceless EMP moments $\mathcal{T}_n\boldsymbol{\mu}_j^{(n)}$ are often used in lieu of $\boldsymbol{\mu}_j^{(n)}$ as defined in Equation (2.7). However, we want to remind the reader here that this $\boldsymbol{\nabla}^{(n+m)}f$, in general, is not traceless and Equation (4.7) is not necessarily true. It can be proved that $\boldsymbol{\nabla}^{(n)}f(\mathbf{r}^2)$ is totally traceless if and only if

$$(2n-2k+1)F_{n-k}+2\mathbf{r}^2F_{n-k+1}=0\ \ \forall k\in[1,\left\lfloor\frac{n}{2}\right\rfloor].\tag{4.8}$$

The proof is given in Section S2[57] but the reader can easily verify that the Coulomb kernel $f(\mathbf{r}^2)=\sqrt{\frac{1}{\mathbf{r}^2}}$ satisfies this condition while $f(\mathbf{r}^2)=\int_1^\infty\exp(-\mathbf{r}^2t^2)\mathrm{d}t$ does not, the latter of which is the damped Coulomb potential as used in the direct-space sum in the Ewald summation method. Second, some spherical coordinate-based algorithms for calculating EMP interactions also take advantage of the traceless tensor when Equation (4.8) is satisfied.[36] With that, one would argue that a spherical coordinate-based algorithm is more efficient than a Cartesian coordinate one. However, I want to point out here that this very same trick in the spherical coordinate-based algorithm simply manifests itself in the current Cartesian coordinate-based approach by reducing the number of terms in Equations (4.3)–(4.5). Since Cartesian coordinate-based algorithms are generally more straightforward to implement, most molecular simulation packages use Cartesian rather than spherical coordinates. Therefore, coordinate transformation at

every time step is required to use such a spherical coordinate-based algorithm in most simulation packages. Note that this transformation involves evaluating a large number of trigonometric functions, which, in general, is at least an order of magnitude slower than simple multiplication or addition and incurs a big overhead. Thus, the current algorithm is more efficient than the spherical coordinate-based one in terms of implementation in modern simulation software.

V. INTERPRETATION OF THE ALGORITHM

The tensor algebra shown in Section IV might seem difficult to digest for non-specialist so I also supply an intuitive explanation. Let us take Equation (4.3), for example. After a close examination, it is easy to show that the right-hand side of Equation (4.3) is a series of tensor contractions of the EMP moments with $\mathbf{r}^{(m+n)}$ (O_m and O_n), with themselves (l_m and l_n) and each other (l_c). As dot products can be interpreted as projections between vectors, the tensor contraction here can be read as the projection of the EMP moments onto $\mathbf{r}^{(m+n)}$, themselves, and each other. These projections arise from the contraction of the EMP moments with the $\mathbf{r}^{(m+n)}\boldsymbol{\delta}^{(k)}$ tensor, which in turn arises from the application of gradient operator on $\mathbf{r}^{(m+n)}$ by the chain rule of derivative (see the derivation from Lemma 3.2 and Theorem 3.1). In fact, the multipole expansion (right-hand side of Equation (2.6)) is equivalent to an expansion on the basis of spherical harmonics in Cartesian coordinates when the kernel function is Coulomb kernel $\frac{1}{r}$, with each term in the expansion being a projection of the EMP moment on the spherical harmonic of the same degree. This has been discussed previously by Applequist.[42] Here, the author generalized such expansion to any kernel function of the form $f(r)$.

VI. NOVELTY OF THE ALGORITHM

There are three major novelties of the algorithm developed here. First of all, it shows that multipole interactions via all kernel functions of the form $f(r)$, i.e., any function that depends on inter-particle distance, have essentially the same mathematical expression except for a few coefficients. This means the same set of working equations can be used for a wide variety of kernel potentials, e.g., direct Coulomb potential, reaction field potential, or the damped Coulomb potential in the Ewald summation, making it fit easily into modern molecular dynamics simulation packages for a broad range of applications.

Second, the number of floating-point operations required to perform the calculation is minimal as compared to the recursive algorithms developed previously;[15,37] the algorithm developed here is even faster than the best possible recursion scheme. The comparison will be given in Section VIII along with the explanation for why the algorithm is fast. I want to point out here that this reduction in floating-point operation has nothing to do with the implementation of the algorithm but it stems from the fact that the algorithm only takes into account necessary operations.

Last but not the least, the mathematical expression of this algorithm is highly compact and modularized, making it very easy to implement. One can easily cast these formulas into a set of matrix-vector multiplications, which can be performed using various high performance linear algebra packages. Also, a close examination of working equations (4.3)–(4.5) reveals that the derivatives of the kernel functions are just coefficients (F_i) of the tensor contraction and one only needs up to the $(m+n)$th scalar derivatives. These derivatives can be implemented as modules independent of the contraction, making the codes highly reusable for a wide variety of kernel functions. The details of the implementation will be given in Section IX.

VII. THE EWALD SUMMATION

The idea of Ewald summation for point charges has been discussed extensively so I refer the reader to the excellent reviews for details.[33,34] Here, I will apply the same idea to a system of general EMPs. I again refer the reader to the Appendix for the notation I use in what follows. Consider a system of N point EMP m_j at position $\mathbf{r}_j, j = 1,2,\ldots,N$ in a unit cell with m_j defined as in Equation (2.11). The unit cell is replicated to form a lattice $\mathcal{L}$. The electrostatic energy of the unit cell is

$$U = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\mathbf{p}\in\mathcal{L}}{}'\int_{\text{all space}}\phi_{j,\mathbf{p}}(\mathbf{r})\mathsf{m}_i(\mathbf{r})\mathrm{d}\mathbf{r}, \tag{7.1}$$

where $\phi_{j,\mathbf{p}} \equiv \mathsf{m}_{j,\mathbf{p}}(\mathbf{r}) * \gamma_j(\mathbf{r}) * G(\mathbf{r})$ with $\mathsf{m}_{j,\mathbf{p}}(\mathbf{r}) \equiv \sum_{n=0}^{+\infty}\boldsymbol{\mu}_j^{(\mathrm{n})}\cdot n\cdot\boldsymbol{\nabla}_j^{(n)}\delta(\mathbf{r}-\mathbf{r}_\mathbf{j}-\mathbf{p})$ and the $'$ in the 3rd sum means exclusion of $i=j$ if $\mathbf{p}=\mathbf{0}$. This sum is conditionally convergent for the interacting EMPs where the sum of the ranks is less than 2, i.e., for charge-charge, charge-dipole, dipole-dipole, and charge-quadrupole interaction.[49] However, in practice, the appropriate cutoff for rank >2 might be needed to be very large in order to converge the sum. In fact, it has been shown that short cutoffs of dispersion interaction of the form r^{-6} could cannot correctly reproduce the fluid-fluid interfacial properties.[29–32] It is thus reasonable to consider EMP interaction of rank 5 as long-range interaction. The idea of Ewald summation is to decompose $\phi_{j,\mathbf{p}}$ into two functions, one of which decays rapidly in real space while the other decays rapidly in Fourier space, so that the sum can be evaluated with relatively small cutoffs in real and Fourier space. Because of the linearity of the Poisson's equation, decomposition of the potential is equivalent to decomposition of the source density. The canonical choice of such decomposition is to let $\gamma_j(\mathbf{r}) = \delta(\mathbf{r}) = \delta(\mathbf{r}) - g_\alpha(\mathbf{r}) + g_\alpha(\mathbf{r})$ as in Equation (2.6), where $g_\alpha(\mathbf{r}) \equiv (\alpha/\pi)^{3/2}e^{-\alpha|\mathbf{r}|^2}$, so that $\phi_{j,\mathbf{p}} = \mathsf{m}_{j,\mathbf{p}} * G * (\delta - g_\alpha) + \mathsf{m}_{j,\mathbf{p}} * G * g_\alpha$. Let $\phi^{\mathrm{d}}_{j,\mathbf{p}} \equiv \mathsf{m}_{j,\mathbf{p}} * G * (\delta - g_\alpha)$ and $\phi^{\mathrm{k}}_{j,\mathbf{p}} \equiv \mathsf{m}_{j,\mathbf{p}} * G * g_\alpha$ so that Equation (7.1) can be rewritten as $U = U_{\mathrm{d}} + U_{\mathrm{k}}$, where

$$U_{\mathrm{d}} \equiv \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\mathbf{p}\in\mathcal{L}}{}'\int_{\text{all space}}\phi^{\mathrm{d}}_{j,\mathbf{p}}\mathsf{m}_i(\mathbf{r})\mathrm{d}\mathbf{r}, \tag{7.2}$$

$$U_{\mathrm{k}} \equiv \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\mathbf{p}\in\mathcal{L}}{}'\int_{\text{all space}}\phi^{\mathrm{k}}_{j,\mathbf{p}}\mathsf{m}_i(\mathbf{r})\mathrm{d}\mathbf{r}. \tag{7.3}$$

The second sum U_{k} is done in Fourier space (see the Appendix for a description of the notation used),

$$U_{\mathrm{k}} = \frac{|\det\tilde{\mathbf{A}}^{(2)}|}{2} \sum_{\mathbf{k}\in\tilde{\mathcal{L}}}^{\mathbf{k}\neq\mathbf{0}} \hat{g}_\alpha(\mathbf{k})\hat{G}(\mathbf{k})|\hat{\mathsf{M}}(\mathbf{k})|^2 - U_{\mathrm{self}}, \tag{7.4}$$

where $\hat{g}_\alpha(\mathbf{k})$, $\hat{G}(\mathbf{k})$, and $\hat{\mathsf{M}}(\mathbf{k})$ are the Fourier transform of g_α, G, and $\mathsf{M}(\mathbf{r}) \equiv \sum_{i=1}^{N} \mathsf{m}_i(\mathbf{r})$, respectively. U_{self} is defined as

$$\begin{aligned} U_{\mathrm{self}} &\equiv \frac{1}{2}\sum_{i=1}^{N}\sum_{j=i}\int_{\mathrm{allspace}} \phi^{\mathrm{k}}_{j,\mathbf{p}}(\mathbf{r})\mathsf{m}_i(\mathbf{r})\mathrm{d}\mathbf{r} \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=i}\sum_{m=0}^{+\infty}\sum_{n=0}^{+\infty}(-1)^n \boldsymbol{\mu}_i^{(\mathrm{n})}\cdot n \cdot \boldsymbol{\nabla}_j^{(n+m)}\frac{\mathrm{erf}(\sqrt{\alpha}|\mathbf{r}_i-\mathbf{r}_j|)}{|\mathbf{r}_i-\mathbf{r}_j|}\cdot m \cdot \boldsymbol{\mu}_j^{(\mathrm{m})}, \end{aligned} \tag{7.5}$$

where $\mathrm{erf}(x) \equiv \frac{2}{\sqrt{\pi}}\int_0^x \exp(-u^2)\mathrm{d}u$ is the error function. U_{self} is the interaction energy between $\mathsf{m}_j * G * g_\alpha$ and m_j at r_j, which is a non-physical term included in the Fourier series of U_{d} and should be subtracted from the sum. The derivation will be given in Section S3 of the supplementary material.[57] Because $\hat{\mathsf{M}}(\mathbf{k})$ has a very simple form in Fourier space (see Section S3[57]), the evaluation of U_{k} does not involve complicated tensor-tensor contractions and can be done efficiently using the Fast Fourier Transform (FFT) technique.[15] On the other hand, U_{d} can be summed rapidly in real space with a relatively small cut-off distance r_{cut} due to rapid decay of $\phi^{\mathrm{d}}_{j,\mathbf{p}} \equiv \mathsf{m}_j * G * (\delta - g_\alpha) = \frac{\mathrm{erfc}(\sqrt{\alpha}|\mathbf{r}-\mathbf{r}_j|)}{|\mathbf{r}-\mathbf{r}_j|}$,

$$U_{\mathrm{d}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j\neq i \\ r_{ij}<r_{\mathrm{cut}}}}^{N}\sum_{m=0}^{+\infty}\sum_{n=0}^{+\infty}(-1)^n \boldsymbol{\mu}_i^{(\mathrm{n})}\cdot n \cdot \boldsymbol{\nabla}_j^{(n+m)}\frac{\mathrm{erfc}(\sqrt{\alpha}|\mathbf{r}_i-\mathbf{r}_j|)}{|\mathbf{r}_i-\mathbf{r}_j|}\cdot m \cdot \boldsymbol{\mu}_j^{(\mathrm{m})}, \tag{7.6}$$

where erfc $\equiv 1 -$ erf is the complementary error function and $r_{ij} \equiv |\mathbf{r}_i - \mathbf{r}_j|$. Thus, aside from the need to maintain physical fidelity as discussed earlier in this section, efficient calculation is another argument for using Ewald summation to handle long-range EMP interactions.

Here, I will apply the results obtained from Sections III and IV to the evaluation of U_{self} and U_{d}. First, from Theorem 3.1, it is clear that if $\mathbf{r} = \mathbf{0}$, as in Equation (7.5) with $i = j$, all the terms involving $\mathbf{r}^{(n-2k)}$ with $n - 2k \neq 0$ vanish, i.e., for $i = j$ and $|\mathbf{r}_i - \mathbf{r}_j| = \mathbf{0}$,

$$\begin{aligned} &\boldsymbol{\nabla}_j^{(n+m)}\frac{\mathrm{erf}(\sqrt{\alpha}|\mathbf{r}_i-\mathbf{r}_j|)}{|\mathbf{r}_i-\mathbf{r}_j|} \\ &\quad= \begin{cases} \mathbf{0} & \text{if } m+n = 2g+1,\ \ g\in\mathbb{N}^0 \\ 2^{n+m-g}F_{n+m-g}\boldsymbol{\delta}^{(g)} & \text{if } m+n = 2g,\ \ g\in\mathbb{N}^0 \end{cases} \end{aligned} \tag{7.7}$$

so that in case of $m + n = 2g$, $g \in \mathbb{N}^0$, U_{self} can be evaluated using Equation (4.3) with the constraint $k = g$, $O_n = 0$, and $O_m = 0$.

On the other hand, U_{d} can be evaluated using Equation (4.3) with $F_k = 2\sqrt{\frac{\alpha}{\pi}}(-\alpha)^k B_k(\alpha\mathbf{r}^2)$, where $B_k(x) \equiv \int_1^{+\infty}\exp(-xt^2)t^{2k}\mathrm{d}t$ is the complementary Boys function of order k. A very useful downward recursion can be used to evaluate all B_{n-k} and F_{n-k} if we know B_n: $B_k(x) = (2xB_{k+1}(x) - \exp(-x))/(2n+1)$.

VIII. COMPLEXITY ANALYSIS OF THE ALGORITHM

A. Comparison to recently developed recursion schemes

Previously, the McMurchie-Davidson formalism[46] has been exploited to populate the components of $\boldsymbol{\nabla}_j^{(n+m)}\frac{\mathrm{erfc}(\sqrt{\alpha}|\mathbf{r}_i-\mathbf{r}_j|)}{|\mathbf{r}_i-\mathbf{r}_j|}$ and make the evaluation of U_d efficient.[15] More recently, similar recursion schemes for the kernel functions $\frac{1}{r^\nu}$ and $\frac{\mathrm{erfc}(\sqrt{\alpha}|\mathbf{r}|)}{|\mathbf{r}|}$ were proposed by Boateng and Todorov.[37] Since the algorithm I develop here (Equation (4.3)) does not require population of the central tensor in Equation (7.6), I would like to compare the efficiency of the current approach with the aforementioned recursion ones. For simplicity, the discussion here is restricted to the interaction energy between two EMP moments of the same rank p, but it is trivial to generalize the conclusions here to forces or torques as well as to EMP moments of different ranks.

In general, there are two major steps in the aforementioned recursion schemes:

1. Construct a matrix representing the central tensor of $\boldsymbol{\nabla}^{(n)}f(\mathbf{r}^2)$ where $f(\mathbf{r}^2)$ is the kernel function.
2. Evaluate the vector-matrix-vector bilinear form, where the two interacting EMP moments are represented by the two vectors.

Without going into the details, I simply give the number of multiplication and addition as a function of p for the

algorithms under comparison here (the derivation is given in Section S4 of the supplementary material[57]). Note that I do not explicitly give the "big O" notation for complexity here because it only tells how the complexity scales as a function of the inputs and does not necessarily indicate the efficiency of the algorithm in solving a given problem. (However, one can easily fit a polynomial to the result I give later to obtain a "big O" estimate.) For example, the FMM, which scales as $O(N)$, is significantly slower than the Particle Mesh Ewald (PME), which scales as $O(N \log N)$, for most reasonable system sizes.[10,50] In molecular simulations with explicit representations of EMP, one usually cannot afford going higher than $p = 4$ even with state-of-the-art computation facilities. $p \leq 2$ is more commonly seen.[14–20,23,24] In fact, in the later comparison with the recursive algorithms developed previously (see below), one can verify that the algorithm here is faster than the recursive ones until $p \geq 100$. Therefore, the "big O" notation is irrelevant in the comparison among the methods under discussion.

The McMurchie-Davidson formalism costs

$$\binom{2p+4}{4} * 2 + \binom{p+2}{2}^2 + \binom{p+2}{2} \tag{8.1}$$

multiplications and

$$\binom{2p+4}{4} + \binom{p+2}{2}^2 - 1 \tag{8.2}$$

additions. The recursion scheme proposed by Boateng and Todorov for $\frac{1}{r^v}$ costs

$$\binom{2p+3}{3} * 17 + \binom{p+2}{2}^2 + \binom{p+2}{2} \tag{8.3}$$

multiplications and

$$\binom{2p+3}{3} * 9 + \binom{p+2}{2}^2 - 1 \tag{8.4}$$

additions, while that for $\frac{\mathrm{erfc}(\sqrt{\alpha}|\mathbf{r}|)}{|\mathbf{r}|}$ costs

$$\binom{2p+3}{3} * 31 + \binom{p+2}{2}^2 + \binom{p+2}{2} \tag{8.5}$$

multiplications and

$$\binom{2p+3}{3} * 16 + \binom{p+2}{2}^2 - 1 \tag{8.6}$$

additions.

On the other hand, Equation (4.3) costs

$$6\binom{p+2}{3} + \sum_{lc=0}^{p}(2 + (\left\lfloor \frac{p-lc}{2} \right\rfloor + 1)(3 + \binom{lc+2}{2} + (3 + \binom{l_c+2}{2})(\left\lfloor \frac{p-lc}{2} \right\rfloor + 1))) \tag{8.7}$$

multiplications and

$$4\binom{p+2}{3} + 2T_r + \sum_{lc=0}^{p}(\binom{lc+2}{2}(\left\lfloor \frac{p-lc}{2} \right\rfloor + 1)\left\lfloor \frac{p-lc}{2} \right\rfloor + \binom{lc+2}{2} - 1) \tag{8.8}$$

additions, where

$$T_r \equiv \begin{cases} \frac{4}{3}\binom{g+1}{2}(g+1)(g+2) & \text{if } p = 2g+1, \quad g \in \mathbb{N} \\ \frac{1}{3}\binom{g+1}{2}(p^2+2p-2) & \text{if } p = 2g, \quad g \in \mathbb{N} \end{cases}. \tag{8.9}$$

The ratios between the complexity of the 3 recursion schemes and Equation (4.3) are plotted as function of p in Figure 1. While the recursion schemes are intuitively easy to interpret, they are significantly slower as compared to the algorithm here. For example, to evaluate the energy between two hexadecapoles ($p = 4$) with the kernel $\frac{1}{r^v}$, the algorithm here is about 9 times faster than that prosed by Boateng and Todorov (Figures 1(c) and 1(d)); to evaluate the direct space energy between two hexadecapoles in the Ewald summation with the kernel function $\frac{\mathrm{erfc}(\sqrt{\alpha}|\mathbf{r}|)}{|\mathbf{r}|}$, the algorithm here is about 4 times faster than the McMurchie-Davidson formalism (Figures 1(a) and 1(b)) and about 16 times faster than the algorithm proposed by Boateng and Todorov (Figures 1(e) and 1(f)).

What makes the recursion algorithms slow is the construction of the central matrix in step 1. For small p, the reader can easily verify that step 1 alone is already much more expensive than the complete computation of the energy via Equation (4.3). This can also be seen from the fact that for the same kernel $\frac{\mathrm{erfc}(\sqrt{\alpha}|\mathbf{r}|)}{|\mathbf{r}|}$, the McMurchie-Davidson formalism differs only in step 1 from that proposed by Boateng and Todorov and former is faster than the latter (compare Figures 1(a) and 1(b) with 1(e) and 1(f)) because of a much simpler recursion

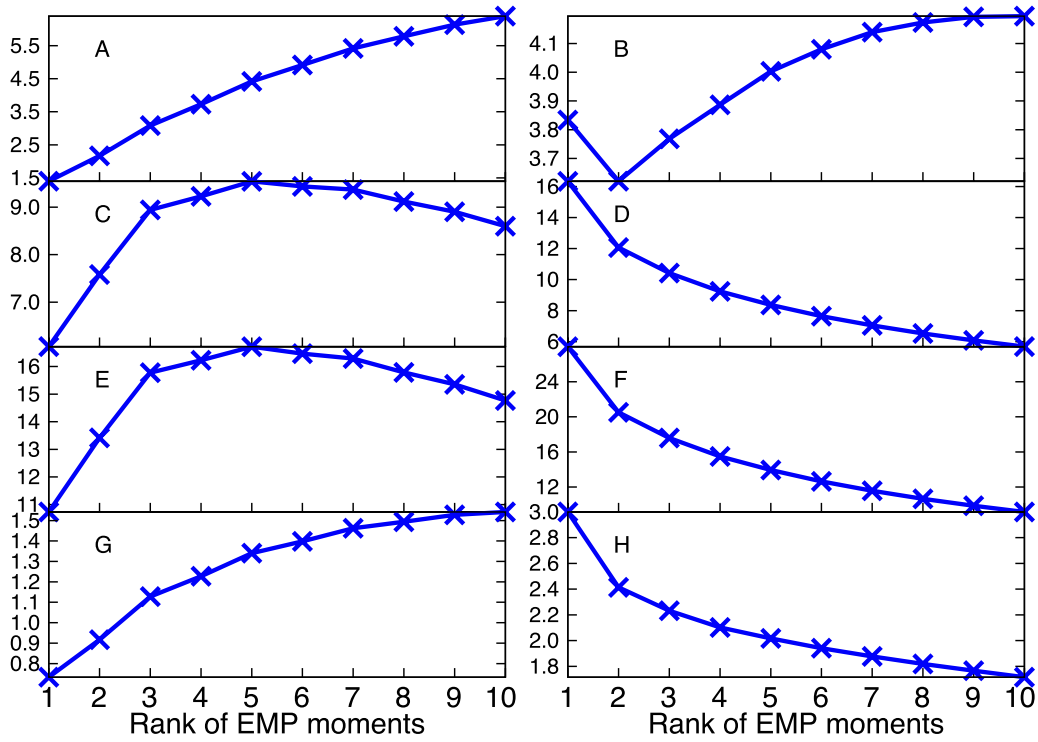

FIG. 1. The ratio of computational complexity (Y-axis) between the two recursion schemes and Equation (4.3) ((a) and (b): McMurchie-Davidson;[15] (c) and (d): recursion scheme for computing gradients of $\frac{1}{r^v}$ as in Ref. 37; (e) and (f): recursion scheme for computing gradients of $\frac{\mathrm{erfc}(\sqrt{\alpha}|\mathbf{r}|)}{|\mathbf{r}|}$ as in Ref. 37; (g) and (h): ideal recursion scheme where constructing the central tensor requires 1 operation per element) in terms of multiplication ((a), (c), (e), and (g)) or addition ((b), (d), (f), and (h)) as a function of EMP ranks (X-axis).

scheme[15,37] for constructing the central tensor. Also, from an optimization prospective, it is much easier to parallelize the computation in Equation (4.3) (see Section IX) than the recursive ones; the construction of the central tensor, which is the bottleneck step, is almost impossible to parallelize since its elements have to be computed in a specific order.

The need to construct the central tensor also requires a large amount of memory. In typical molecular simulations, the central tensor in Equation (7.6) has to be stored in memory for each pair of atoms, and this costs $nC\binom{p+2}{2}^2$, where n is the total number of atoms and C depends on the distance cut-off scheme in the simulation. Care has to be taken in order to optimize for memory access, which adds to the difficulty of implementation.[15] On the other hand, Equation (4.3) only requires storage of an array of $n(\binom{p+2}{3} + \frac{T_r}{2})$ elements regardless of what cut-off scheme is used. This means Equation (4.3) is not only faster and more memory efficient but also easier to implement and optimize than the recursive algorithms for computing EMP interactions. Figure 2 shows the ratio between the memory consumption of the recursion schemes and the current algorithm. It is obvious that the recursion schemes consume about 3 orders of magnitude more memory than the current algorithm.

B. Comparison to the best possible recursion scheme

As the current algorithm is faster than the aforementioned recursive ones, one interesting question to ask is whether we can improve the recursion to speed up the calculation. Let us assume that in an ideal recursion scheme, one would need only one multiplication and one addition to construct each element of the central tensor/matrix. To build the central tensor/matrix up to rank p, one would need to calculate at least $\binom{2p+3}{3}$ elements. This means we need at least $\binom{2p+3}{3}$ multiplications and additions. To perform the bilinear form, we need additional $\binom{p+2}{2}^2 + \binom{p+2}{2}$ multiplications and $\binom{p+2}{2}^2 - 1$ additions. This totals to

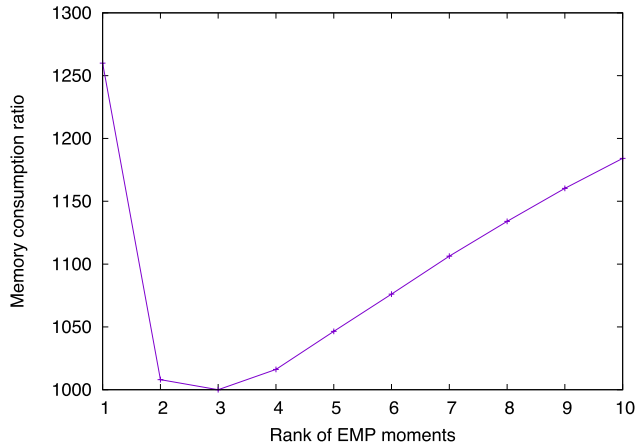

FIG. 2. The ratio of memory consumption between the recursion schemes and Equation (4.3) as a function of EMP ranks. Note that in the recursion schemes, one needs to store a matrix for each pair of atoms. The number of pairs for each atom is assumed to be 140, which amounts to the typical size of the pair list with a distance cutoff of 10 Å in a simulation system that has the same density as water.

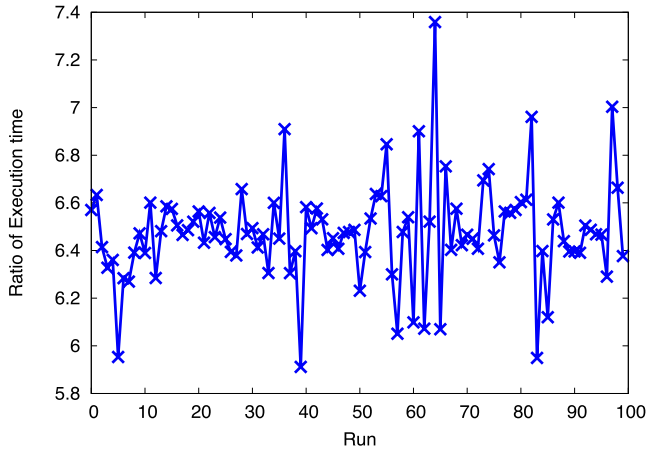

FIG. 3. The ratio of execution time between the recursive algorithm by Boateng and Todorov[37] and Equation (4.3) with the Coulomb kernel $\frac{1}{r}$ at the hexadecapole level ($p = 4$) for each of the 100 tests.

$$\binom{2p+3}{3} + \binom{p+2}{2}^2 + \binom{p+2}{2} \tag{8.10}$$

multiplications and

$$\binom{2p+3}{3} + \binom{p+2}{2}^2 - 1 \tag{8.11}$$

additions. Figures 1(g) and 1(h) show the complexity ratio between this ideal recursion scheme and the algorithm I develop here. It is obvious that the algorithm here is even more competitive than such an ideal (impossibly simple) recursion scheme. Also note that when the central tensor is traceless, the algorithm developed here can be even faster (see discussion in Section IV) while the recursion scheme cannot use this trick to speed up the calculation.

C. Numerical validation

To validate the comparison in Section VIII A, I implemented the algorithm developed here and the recursive one by Boateng and Todorov[37] for the Coulomb kernel ($\frac{1}{r}$) and did a performance test on both. The test was to evaluate the electrostatic potential of a system containing 5000 hexadecapoles randomly and evenly distributed across a $20 \times 20 \times 20$ box. The components of the hexadecapole were chosen randomly between 0 and 0.05. Units are chosen such that the Coulomb's constant is 1. The potential energies between all unique pairs of hexadecapoles are evaluated, totaling the number of pairwise energy evaluations to $\binom{5000}{2}$. The same test is repeated for 100 times and at each time, the ratio between the execution time of the two algorithms is plotted in Figure 3. The recursive algorithm costs about 6.5 times as much as the algorithm developed here when the two algorithms give the same electrostatic potential energy (with 15-digit accuracy in double precision). Note that this ratio might vary a bit with different implementations but it agrees qualitatively with the result (9 times) from complexity analysis in Section VIII A. Again, this shows that the algorithm developed here is faster than the recursive one.

D. Reasons for the algorithm's efficiency

The efficiency of the current algorithm stems from the gist of the multipole expansion, which is basically a series of projections in multilinear space (see Section V), and decomposing the bilinear form into the minimal set of these projection operations. Perhaps more importantly, each of these projection operations has very simple operands, i.e., the two interacting EMP moments and the distance-dependent $\mathbf{r}^{(m+n)}$, so that no operation is wasted on computing intermediate variables. I will simply conclude this section with the textbook example of dipole-dipole interaction. It is easy to recover the following well-known dipole-dipole interaction energy equation by plugging the Coulomb potential ($\frac{1}{r}$) into Equations (4.3) and (2.10):

$$U = \frac{-3(\boldsymbol{\mu}_i^{(1)} \cdot \hat{\mathbf{r}}_{ij})(\boldsymbol{\mu}_j^{(1)} \cdot \hat{\mathbf{r}}_{ij}) + \boldsymbol{\mu}_i^{(1)} \cdot \boldsymbol{\mu}_j^{(1)}}{r_{ij}^3}, \tag{8.12}$$

where $\hat{\mathbf{r}}_{ij}$ is the unit vector of $\mathbf{r}_{ij} \equiv \mathbf{r}_i - \mathbf{r}_j$. This only requires 3 dot products and a few scalar multiplications. On the other hand, if one was to use the recursive algorithm, e.g., the one proposed by Boateng and Todorov,[37] the working equation would be

$$U = \boldsymbol{\mu}_i^{(1)} \mathbf{M} \boldsymbol{\mu}_j^{(1)}, \tag{8.13}$$

where one would need to first fill out the upper triangle of the 3×3 symmetric matrix $\mathbf{M}$ via recursion, with the costs of each element being more than 20 floating-point operations, and then carry out the bilinear form with the equivalent of 4 dot products. Arguably, one could manually optimize out some multiplication-by-zero operations from the published equations;[37] however, even if we assume an ideal case where it needs only 1 multiplication and 1 addition, such a method spends a significant number of operations on computing unnecessary intermediate variables, making it suboptimal compared to the algorithm developed here.

IX. IMPLEMENTATION

Equations (4.3), (2.16), and (2.17) can be easily implemented as a series of matrix-vector products. Taking Equation (4.3) as an example, we can cast it into a triple sum over l_c, l_m, and l_n as follows:

$$\begin{aligned}
&\mathbf{A}^{(n)} \cdot n \cdot \boldsymbol{\nabla}^{(n+m)} f(\mathbf{r}^2) \cdot m \cdot \mathbf{B}^{(m)} \\
&= \sum_{l_c=0}^{\min(m,n)} \left\{ \sum_{l_m=0}^{\left\lfloor \frac{m-l_c}{2} \right\rfloor} \left\{ \left[\mathbf{A}^{(m:l_m)} \cdot O_m \cdot \mathbf{r}^{(O_m)} \right] \cdot l_c \cdot \left[\sum_{l_n=0}^{\left\lfloor \frac{n-l_c}{2} \right\rfloor} \left(\mathbf{B}^{(n:l_n)} \cdot O_n \cdot \mathbf{r}^{(O_n)} C'\binom{m, n}{l_c, l_m, l_n} \right) \right] \right\} \right\},
\end{aligned} \tag{9.1}$$

where

$$C'\binom{m\ ,\ n}{l_c,l_m,l_n} \equiv C\binom{m\ ,\ n}{l_c,l_m,l_n} 2^{m+n-l_c-l_m-l_n} \times F_{m+n-l_c-l_m-l_n}. \tag{9.2}$$

For a given triplet of m, n, and l_c, $C'\binom{m\ ,\ n}{l_c,l_m,l_n}$ can be reinterpreted as a $(\lfloor \frac{m-l_c}{2} \rfloor + 1) \times (\lfloor \frac{n-l_c}{2} \rfloor + 1)$ matrix $\mathbf{C}'^{\{m,n,l_c\}}$,

$$\mathbf{C}'^{\{m,n,l_c\}}_{l_m,l_n} \equiv C'\binom{m\ ,\ n}{l_c,l_m,l_n}, \tag{9.3}$$

where the superscript $\{m,n,l_c\}$ is just a reminder of the fact that $\mathbf{C}'$ is a function of m, n, and l_c. Similarly, $\mathbf{A}^{(m:l_m)} \cdot O_m \cdot \mathbf{r}^{(O_m)}$ and $\mathbf{B}^{(n:l_n)} \cdot O_n \cdot \mathbf{r}^{(O_n)}$ can be treated as the elements of the respective $l_m + 1$ and $l_n + 1$ array of l_c-rank tensors,

$$\mathbf{A}^{\{m,l_c\}}_{l_m} \equiv \mathbf{A}^{(m:l_m)} \cdot O_m \cdot \mathbf{r}^{(O_m)}, \tag{9.4}$$

$$\mathbf{B}^{\{n,l_c\}}_{l_n} \equiv \mathbf{B}^{(n:l_n)} \cdot O_n \cdot \mathbf{r}^{(O_n)}, \tag{9.5}$$

where the superscripts $\{m,l_c\}$ and $\{n,l_c\}$ again indicate that $\mathbf{A}$ and $\mathbf{B}$ are functions of m or n and l_c. Equation (9.1) can then be cast into a vector-matrix-vector bilinear form

$$\mathbf{A}^{(n)} \cdot n \cdot \boldsymbol{\nabla}^{(n+m)} f(\mathbf{r}^2) \cdot m \cdot \mathbf{B}^{(m)} = \sum_{l_c=0}^{\min(m,n)} \mathbf{A}\mathbf{C}'\mathbf{B} \tag{9.6}$$

if we define the vector-vector dot product between the two tensor arrays $\mathbf{A}$ and $\mathbf{C}'\mathbf{B}$ as element-wise l_c-fold tensor contraction.

There are a few advantages to using the bilinear form in Equation (9.6) to calculate the EMP energy in Equation (2.14). First of all, as discussed in Section VIII, the number of operations is much smaller than directly evaluating the bi-contraction via the canonical matrix formalism even in the case where the central tensor can be populated recursively. Second, in applications where the EMP sites are represented by EMP polytensor,[51,52] e.g., as in a molecular dynamics force field, one needs to evaluate Equation (9.6) for each $\mathbf{A}^{(n)}$ against a series of $\mathbf{B}^{(m)}$ at the same displacement $\mathbf{r}$ where m varies. One can just populate the array defined in Equation (9.4) for the largest possible l_c and use it for any m. This is in contrast with the canonical matrix formalism where the central tensor has to be populated for each individual pair of m and n, which costs extra operations. Last but not the least, unlike the formalism proposed by Smith,[41] where a large number of scalar arithmetic terms have to be written out manually, the simplicity of Equation (9.6) makes it possible to use compact data structures such as arrays in implementation with the ease of debugging and code maintenance. For example, for a list of EMP sites interacting with $\mathbf{A}^{(n)}$, one can build up a matrix $\mathbf{D}$ whose rows are the $\mathbf{B}$ arrays for each of the interacting sites so that the total energy can be computed using Equation (9.6) with the matrix-vector product $\mathbf{C}'\mathbf{B}$ simply replaced by the matrix-matrix product of $\mathbf{C}'\mathbf{D}$. This also makes it very easy to use high performance linear algebra libraries such as BLAS[53,54] or Blaze[55,56] to carry out the computation.

For force and torque as in Equations (4.4) and (4.5), one can also find an expression similar to Equation (9.6). Instead of elaborating the details, the author has implemented a C++11 template library for basic operations of symmetric Cartesian tensors with the focus on its application in computation of EMP energy, force, and torque. It is available for download at https://github.com/dejunlin/emp/releases. The implementation achieves consistency between the numerical and analytical forces and torques with 10-digit accuracy (see Section S5 and Figure S1 for details[57]).

X. CONCLUSIONS

In conclusion, I established the algorithm to calculate multipole-multipole interaction of a generalized potential, which is a function of inter-site distance only. The method presented here is much faster than the canonical tensor-based formalism and has a compact expression that is very easy to implement in computer programs. A comparison of the current method to various recursive algorithms developed recently showed that this method is generally faster. This formalism is applicable to various interaction potentials where the explicit representation of multipole moments is needed and its Cartesian basis makes it a natural fit in modern molecular simulation scheme.

ACKNOWLEDGMENTS

Thanks goes to Dr. Alan Grossfield at the University of Rochester for helpful comments on the paper. This work was supported by Grant No. GM095496 from the NIH and the Elon Huntington Hooker Graduate Fellowship from the University of Rochester.

APPENDIX: MATHEMATICAL NOTATION

1. Tensors

Here, I adopt the tensor notation used elsewhere.[42] We will denote a Cartesian tensor of rank n by the boldface symbol $\mathbf{A}^{(n)}$, or by $A^{(n)}_{\alpha_1\ldots\alpha_n}$, where the subscripts α_i $(i = 1,2,\ldots,n)$ denote Cartesian axes 1, 2, 3 (corresponding to x, y, z, respectively). $\mathbf{A}^{(n)}$ is called a *totally symmetric* tensor if $A^{(n)}_{\alpha_1\ldots\alpha_n}$ is invariant under any permutation of the sequence of subscripts. A totally symmetric tensor can be denoted by its compressed form $A^{(n)}(n_1,n_2,n_3)$, where n_i is the number of times the i occurs in the sequence of subscripts $\alpha_1\ldots\alpha_n$ and $n_1 + n_2 + n_3 = n$. For simplicity, I will abbreviate a rank-1 tensor $\mathbf{v}^{(1)}$, which is a vector, as $\mathbf{v}$ unless specified otherwise. In the context that $\mathbf{v}$ is defined as a vector, I will denote its kth component by the corresponding *non-bold* character with subscript k, i.e., v_k and the norm of $\mathbf{v}$ by the same character with no subscript, i.e., $v \equiv |\mathbf{v}|$ if $\mathbf{v}$ is a vector.

2. Tensor contractions and traces

We denote an n-fold contraction by $\cdot n \cdot$, e.g.,

$$\mathbf{A}^{(n)} \cdot n \cdot \mathbf{B}^{(n)} \equiv A^{(n)}_{\alpha_1\ldots\alpha_n} B^{(n)}_{\alpha_n\ldots\alpha_1}, \tag{A1}$$

where a summation is assumed over a index that appears twice in a product. Contraction can also be performed between

tensors of different ranks, e.g.,

$$T^{(m-n)}_{\beta_1\ldots\beta_{m-n}} = \mathbf{A}^{(n)} \cdot n \cdot \mathbf{B}^{(m)} \equiv A^{(n)}_{\alpha_1\ldots\alpha_n} B^{(m)}_{\alpha_n\ldots\alpha_1\beta_1\ldots\beta_{m-n}} \ (m \geq n) \quad \text{(A2)}$$

or

$$s^{(m-n)}_{\alpha_1\ldots\alpha_{m-n}} = \mathbf{B}^{(m)} \cdot n \cdot \mathbf{A}^{(n)} \equiv B^{(m)}_{\alpha_1\ldots\alpha_{m-n}\beta_1\ldots\beta_n} A^{(n)}_{\beta_n\ldots\beta_1} \ (m \geq n). \quad \text{(A3)}$$

We will abbreviate $\cdot 1 \cdot$ as $\cdot$ and thus a dot product between 2 vectors or a matrix-vector multiplication is abbreviated as

$$\mathbf{u} \cdot \mathbf{v} \equiv \mathbf{u}^{(1)} \cdot 1 \cdot \mathbf{v}^{(1)} \quad \text{(A4)}$$

or

$$\mathbf{A}^{(2)} \cdot \mathbf{v} \equiv \mathbf{A}^{(2)} \cdot 1 \cdot \mathbf{v}, \quad \text{(A5)}$$

respectively.

The *m-fold trace* of $\mathbf{A}^{(n)}$ is defined as

$$A^{(n:m)}_{\alpha_{2m+1}\ldots\alpha_n} \equiv A^{(n)}_{\mu_1\mu_1\ldots\mu_m\mu_m\alpha_{2m+1}\ldots\alpha_n}. \quad \text{(A6)}$$

3. Fourier transform

The Fourier transform of a function $f(\mathbf{r})$ is denoted as

$$\hat{f}(\mathbf{s}) \equiv \int_{\text{all space}} e^{-i2\pi\mathbf{s}\cdot\mathbf{r}} f(\mathbf{r}) d\mathbf{r}, \quad \text{(A7)}$$

where $d\mathbf{r} \equiv dr_1 dr_2 dr_3$. The corresponding inverse transform is denoted as

$$\check{f}(\mathbf{r}) = \int_{\text{all space}} e^{i2\pi\mathbf{r}\cdot\mathbf{s}} f(\mathbf{s}) d\mathbf{s} \quad \text{(A8)}$$

so that $f = \check{\hat{f}}$.

4. Lattice and Dirac comb

A lattice in $\mathbb{R}^3$ is defined by 3 linearly independent bases $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ so that any lattice point $\mathbf{p}$ is expressed by the *integer* linear combination of the bases,

$$\mathbf{p} = n_1\mathbf{u}_1 + n_2\mathbf{u}_2 + n_3\mathbf{u}_3, \quad n_1, n_2, n_3 = 0, \pm1, \pm2, \ldots. \quad \text{(A9)}$$

We will denote such a lattice by $\mathcal{L}$. Any $\mathcal{L}$ can be transformed from $\mathbb{Z}^3$ via an invertible linear transformation A, i.e.,

$$\mathcal{L} = A(\mathbb{Z}^3). \quad \text{(A10)}$$

A can be expressed as a rank-2 Cartesian tensor $\mathbf{A}^{(2)}$ so that

$$\mathbf{A}^{(2)} \cdot \mathbf{n} \in \mathcal{L}, \ \ \forall \mathbf{n} \in \mathbb{Z}^3. \quad \text{(A11)}$$

The *dual lattice* $\tilde{\mathcal{L}}$ of $\mathcal{L}$ is given by

$$\tilde{\mathcal{L}} = \tilde{A}(\mathbb{Z}^3), \quad \text{(A12)}$$

where $\tilde{A} \equiv (A^{-1})^{\mathrm{T}}$ is the transpose of the inverse of A, so that

$$\mathbf{p} \cdot \mathbf{q} \in \mathbb{Z}, \ \ \forall \mathbf{p} \in \mathcal{L}, \text{ and } \forall \mathbf{q} \in \tilde{\mathcal{L}}. \quad \text{(A13)}$$

We denote the *Dirac comb* function associated with $\mathcal{L}$ as

$$\text{Ш}_{\mathcal{L}}(\mathbf{r}) \equiv \sum_{\mathbf{n}\in\mathbf{Z}^3} \delta(\mathbf{r} - \mathbf{A}^{(2)} \cdot \mathbf{n}), \quad \text{(A14)}$$

where the sum is over all points in $\mathbb{Z}^3$. A function $f(\mathbf{r})$ defined in $\mathbb{V}$ ($\mathbb{V} \subset \mathbb{R}^3$) can be periodized by convolving it with $\text{Ш}_{\mathcal{L}}(\mathbf{r})$,

$$f_{\mathcal{L}}(\mathbf{r}) \equiv \sum_{\mathbf{p}\in\mathcal{L}} f(\mathbf{r} - \mathbf{p}) = f(\mathbf{r}) * \text{Ш}_{\mathcal{L}}(\mathbf{r}), \quad \text{(A15)}$$

where $*$ denotes convolution.

Define

$$\mathbf{k} \equiv \tilde{\mathbf{A}}^{(2)} \cdot \mathbf{n} \ (\mathbf{n} \in \mathbb{Z}^3) \quad \text{(A16)}$$

as a vector in $\tilde{\mathcal{L}}$. The Fourier transform of $f_{\mathcal{L}}(\mathbf{r})$ is given by

$$\hat{f}_{\mathcal{L}}(\mathbf{s}) = \hat{f}(\mathbf{s}) \hat{\text{Ш}}_{\tilde{\mathcal{L}}}(\mathbf{s}) \quad \text{(A17)}$$

$$= \frac{1}{|\det\mathbf{A}^{(2)}|} \sum_{\mathbf{k}\in\tilde{\mathcal{L}}} \hat{f}(\mathbf{k}) \delta(\mathbf{s} - \mathbf{k}), \quad \text{(A18)}$$

where $|\det\mathbf{A}^{(2)}|$ is the determinant of $\mathbf{A}^{(2)}$. Taking the inverse transform of both sides of Equation (A18) will give

$$f_{\mathcal{L}}(\mathbf{r}) = \frac{1}{|\det\mathbf{A}^{(2)}|} \sum_{\mathbf{k}\in\tilde{\mathcal{L}}} e^{i2\pi\mathbf{r}\cdot\mathbf{k}} \hat{f}(\mathbf{k}). \quad \text{(A19)}$$

5. Poisson's equation on a lattice

The Poisson's equation for the electrostatic potential $\phi_{\mathcal{L}}(\mathbf{r})$ of a given source $\rho_{\mathcal{L}}(\mathbf{r})$ is

$$-\Delta\phi_{\mathcal{L}}(\mathbf{r}) = 4\pi\rho_{\mathcal{L}}(\mathbf{r}), \quad \text{(A20)}$$

where both $\phi_{\mathcal{L}}(\mathbf{r})$ and $\rho_{\mathcal{L}}(\mathbf{r})$ are periodic on lattice $\mathcal{L} \equiv A(\mathbb{Z}^3)$ and $\Delta \equiv \nabla^2$ is the Laplace operator. We will denote the pair of solution $\phi_{\mathcal{L}}(\mathbf{r})$ and the source function $\rho_{\mathcal{L}}(\mathbf{r})$ in Equation (A20) as

$$\phi_{\mathcal{L}}(\mathbf{r}) \overset{Possion}{\rightleftharpoons} \rho_{\mathcal{L}}(\mathbf{r}). \quad \text{(A21)}$$

By using Equation (A19), Equation (A20) has the solution as the Fourier series,

$$\phi_{\mathcal{L}}(\mathbf{r}) = \frac{1}{|\det\mathbf{A}^{(2)}|} \sum_{\mathbf{k}\in\tilde{\mathcal{L}}} e^{i2\pi\mathbf{r}\cdot\mathbf{k}} \hat{\phi}(\mathbf{k}) \quad \text{(A22)}$$

$$= \frac{1}{|\det\mathbf{A}^{(2)}|} \sum_{\mathbf{k}\in\tilde{\mathcal{L}}} e^{i2\pi\mathbf{r}\cdot\mathbf{k}} \frac{1}{\pi|\mathbf{k}|^2} \hat{\rho}(\mathbf{k}) \ \ (\mathbf{k} \neq \mathbf{0}), \quad \text{(A23)}$$

where $\hat{\phi}(\mathbf{k})$ and $\hat{\rho}(\mathbf{k})$ are the Fourier transform of $\phi(\mathbf{r})$ and $\rho(\mathbf{r})$, which are in turn one period of $\phi_{\mathcal{L}}(\mathbf{r})$ and $\rho_{\mathcal{L}}(\mathbf{r})$, respectively, and $\mathbf{k}$ and $\tilde{\mathcal{L}}$ are defined as in Equations (A16) and (A12), respectively.

[1]A. D. Buckingham and P. W. Fowler, Can. J. Chem. **63**, 2018–2025 (1985).
[2]G. J. B. Hurst, P. W. Fowler, A. J. Stone, and A. D. Buckingham, Int. J. Quantum Chem. **29**, 1223–1239 (1986).
[3]D. E. Williams, J. Comput. Chem. **9**, 745–763 (1988).
[4]M. A. Spackman, J. Chem. Phys. **85**, 6587 (1986).
[5]F. Colonna, E. Evleth, and J. G. Ángyán, J. Comput. Chem. **13**, 1234–1245 (1992).
[6]W. A. Sokalski, D. A. Keller, R. L. Ornstein, and R. Rein, J. Comput. Chem. **14**, 970–976 (1993).
[7]C. E. Dykstra, Chem. Rev. **93**, 2339–2353 (1993).
[8]S. L. Price, C. H. Faerman, and C. W. Murray, J. Comput. Chem. **12**, 1187–1197 (1991).
[9]Y. Kong, "Multipole electrostatic methods for protein modeling with reaction field treatment," Ph.D. thesis, Graduate School of Arts and Sciences of Washington University, 1997.

[10]G. A. Cisneros, M. Karttunen, P. Ren, and C. Sagui, Chem. Rev. **114**, 779–814 (2014).
[11]G. Marshall, J. Comput.-Aided Mol. Des. **27**, 107–114 (2013).
[12]I. Vorobyov, L. Li, and T. W. Allen, J. Phys. Chem. B **112**, 9588–9602 (2008).
[13]W. D. Bennett and D. P. Tieleman, J. Chem. Theory Comput. **7**, 2981–2988 (2011).
[14]A. Toukmaji, C. Sagui, J. Board, and T. Darden, J. Chem. Phys. **113**, 10913–10927 (2000).
[15]C. Sagui, L. G. Pedersen, and T. A. Darden, J. Chem. Phys. **120**, 73–87 (2004).
[16]W. Wang and R. D. Skeel, J. Chem. Phys. **123**, 164107 (2005).
[17]J. J. Cerda, V. Ballenegger, O. Lenz, and C. Holm, J. Chem. Phys. **129**, 234104 (2008).
[18]J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. DiStasio, M. Head-Gordon, G. N. I. Clark, M. E. Johnson, and T. Head-Gordon, J. Phys. Chem. B **114**, 2549–2564 (2010).
[19]Y. Shi, Z. Xia, J. Zhang, R. Best, C. Wu, J. W. Ponder, and P. Ren, J. Chem. Theory Comput. **9**, 4046–4063 (2013).
[20]M. L. Laury, L.-P. Wang, V. S. Pande, T. Head-Gordon, and J. W. Ponder, J. Phys. Chem. B **119**, 9423–9437 (2015).
[21]D. Elking, T. Darden, and R. J. Woods, J. Comput. Chem. **28**, 1261–1274 (2007).
[22]D. M. Elking, G. A. Cisneros, J.-P. Piquemal, T. A. Darden, and L. G. Pedersen, J. Chem. Theory Comput. **6**, 190–202 (2010).
[23]J. Wu, X. Zhen, H. Shen, G. Li, and P. Ren, J. Chem. Phys. **135**, 155104 (2011).
[24]M. Orsi and J. W. Essex, PLoS One **6**, e28637 (2011).
[25]M. J. Schnieders, T. D. Fenn, V. S. Pande, and A. T. Brunger, Acta Crystallogr., Sect. D: Biol. Crystallogr. **65**, 952–965 (2009).
[26]T. D. Fenn, M. J. Schnieders, A. T. Brunger, and V. S. Pande, Biophys. J. **98**, 2984–2992 (2010).
[27]M. J. Schnieders, T. D. Fenn, and V. S. Pande, J. Chem. Theory Comput. **7**, 1141–1156 (2011).
[28]T. D. Fenn and M. J. Schnieders, Acta Crystallogr., Sect. D: Biol. Crystallogr. **67**, 957–965 (2011).
[29]P. J. in't Veld, A. E. Ismail, and G. S. Grest, J. Chem. Phys. **127**, 144711 (2007).
[30]M. Mazars, J. Phys. A: Math. Theor. **43**, 425002 (2010).
[31]R. E. Isele-Holder, W. Mitchell, and A. E. Ismail, J. Chem. Phys. **137**, 174107 (2012).
[32]J. M. Míguez, M. M. Piñeiro, and F. J. Blas, J. Chem. Phys. **138**, 034707 (2013).
[33]M. Deserno and C. Holm, J. Chem. Phys. **109**, 7678–7693 (1998).
[34]C. Sagui and T. A. Darden, Annu. Rev. Biophys. Biomol. Struct. **28**, 155–179 (1999).
[35]A. Aguado and P. A. Madden, J. Chem. Phys. **119**, 7471–7483 (2003).
[36]A. C. Simmonett, F. C. Pickard IV, H. F. Schaefer III, and B. R. Brooks, J. Chem. Phys. **140**, 184101 (2014).
[37]H. A. Boateng and I. T. Todorov, J. Chem. Phys. **142**, 034117 (2015).
[38]D. Wolf, P. Keblinski, S. R. Phillpot, and J. Eggebrecht, J. Chem. Phys. **110**, 8254–8282 (1999).
[39]M. Lamichhane, J. D. Gezelter, and K. E. Newman, J. Chem. Phys. **141**, 134109 (2014).
[40]M. Lamichhane, K. E. Newman, and J. D. Gezelter, J. Chem. Phys. **141**, 134110 (2014).
[41]W. Smith, CCP5 Info. Quart. **46**, 18–30 (1998).
[42]J. Applequist, J. Phys. A: Math. Gen. **22**, 4303 (1989).
[43]E. Burgos and H. Bonadeo, Mol. Phys. **44**, 1–15 (1981).
[44]L. Greengard and V. Rokhlin, J. Comput. Phys. **73**, 325–348 (1987).
[45]L. Greengard and V. Rokhlin, Acta Numer. **6**, 229–269 (1997).
[46]L. E. McMurchie and E. R. Davidson, J. Comput. Phys. **26**, 218–231 (1978).
[47]T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic Structure Theory* (John Wiley & Sons, Ltd., Chichester, 2000).
[48]H. Grad, Commun. Pure Appl. Math. **2**, 325–330 (1949).
[49]S. W. de Leeuw, J. W. Perram, and E. R. Smith, Proc. R. Soc. London, Ser. A **373**, 27–56 (1980).
[50]E. Pollock and J. Glosli, Comput. Phys. Commun. **95**, 93–110 (1996).
[51]J. Applequist, J. Math. Phys. **24**, 736–741 (1983).
[52]J. Applequist, J. Chem. Phys. **83**, 809–826 (1985).
[53]J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff, ACM Trans. Math. Software **16**, 1–17 (1990).
[54]J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, ACM Trans. Math. Software **14**, 18–32 (1988).
[55]K. Iglberger, G. Hager, J. Treibig, and U. Rüde, in 2012 International Conference on High Performance Computing & Simulation (HPCS), 2012.
[56]K. Iglberger, G. Hager, J. Treibig, and U. Rüde, SIAM J. Sci. Comput. **34**, C42–C69 (2012).
[57]See supplementary material at http://dx.doi.org/10.1063/1.4930984 for the details of the derivation of some equations.