# FAST UPDATING MULTIPOLE COULOMBIC POTENTIAL CALCULATION*

THOMAS A. HÖFT† AND BRADLEY K. ALPERT‡

**Abstract.** We present a numerical method to efficiently and accurately re-compute the Coulomb potential of a large ensemble of charged particles after a subset of the particles undergoes a change of position. Errors are bounded even after a large number of such shifts, making it practical for use in Monte Carlo Markov chain methods in molecular dynamics, computational astrophysics, computational chemistry, and other applications. The method uses truncated multipole expansions of the potential energy functional and a tree decomposition of the computational domain to reduce the computational complexity. Computational costs scale logarithmically in the size of the problem. Scaling, accuracy, and efficiency are confirmed with numerical experiments. The new method outperforms a direct calculation for moderate problem sizes.

**Key words.** fast multipole method, molecular dynamics, Laplace equation, potential theory, N-body problem

**AMS subject classifications.** 35J05, 35Q60, 65E05, 68W25, 70F10, 78M16, 92C40, 92E10

**1. Introduction.** In this paper we introduce a new multipole-based method for computing the potential energy functional of a large ensemble of particles when one or more of the particles undergoes a change of position. We focus on the Coulomb potential as a model problem in electrostatics. Given a collection of particles with real-valued charges $\{q_1, \ldots, q_N\}$ at locations $\{x_1, \ldots, x_N\}$ in a domain $D \subset \mathbb{R}^3$, the Coulomb potential at $x_i$ is

$$\Phi_i(\mathbf{x}) = \sum_{j=1, j\neq i}^{N} \frac{q_j}{|x_i - x_j|} \tag{1}$$

where $\mathbf{x} = \langle x_1, \ldots, x_N \rangle$ and $|\cdot|$ denotes Euclidean distance. The potential energy functional of the ensemble is then

$$V(\mathbf{x}, \mathbf{q}) = \sum_{i=1}^{N} q_i \Phi_i(\mathbf{x}) = \sum_{i=1}^{N} q_i \sum_{j=1, j\neq i}^{N} \frac{q_j}{|x_i - x_j|} \tag{2}$$

where $\mathbf{q} = \langle q_1, \ldots, q_N \rangle$ and we note that the double summation has $N(N-1)/2$ unique terms. Direct evaluation of (2) requires $\mathcal{O}(N^2)$ operations.

If a subset of $k$ particles undergoes a change in position then the potential of the ensemble changes. Letting $\tilde{\mathbf{x}}$ denote the vector containing all positions including changes, we seek to compute $V(\tilde{\mathbf{x}}, \mathbf{q})$ efficiently. By expressing the potential in terms of a multipole expansion, decomposing the domain $D$ using an octree structure, and re-using information from the multipole-based computation of $V(\mathbf{x}, \mathbf{q})$, the method described below evaluates the new potential energy functional in $\mathcal{O}(k \log N)$ time. Evaluating the new potential energy functional using a direct calculation requires $\mathcal{O}(kN)$ computation time.

*Contribution of NIST, an agency of the U.S. Government, not subject to copyright in the United States.

†Department of Mathematics, University of St. Thomas, Saint Paul, MN 55105 (hoft@stthomas.edu).

‡National Institute of Standards and Technology, Boulder, Colorado 80305 (alpert@boulder.nist.gov).

Note that the gravitational potential is of the same form as the Coulomb potential, with charges $q$ replaced by particle masses $m$; it may be treated in identical fashion as the work presented here.

The method described here has application to problems in applied mathematics, biology, chemistry, and physics. Evaluation of an updated potential energy functional is of use in varied applications including molecular dynamics simulations [8, 21] for protein folding [1, 19], chemical dynamics [35], biomolecules [20, 33], and biological cell membrane dynamics [30]; gravitational interactions in large-scale astrophysics simulations [9]; and computational drug discovery [18, 34].

The computational challenge that motivated this work is quantification of the properties of two-phase fluid systems by chemical Monte Carlo simulation [10, 31]. The Metropolis-Hastings algorithm (see, for example, [28]), used for this purpose, involves frequent updates of molecular configurations structured so that each new configuration involves position changes for relatively few charges. For large systems, update of the potential energy functional at a complexity of $\mathcal{O}(kN)$ for the naive method dominates the computational cost of the update: it is this cost we reduce to $\mathcal{O}(k \log N)$.

For molecular or astrophysical dynamics, accurate temporal integrations are governed by time steps limited by the smallest spatial separations among sources (charges or masses) which, correspondingly, may have highly localized coupling and evolution. Under standard methods, however, the trajectories of well-separated sources are also updated at this unnecessarily high rate, potentially entailing prohibitive computational cost. The algorithm of this paper, with further elaboration, could enable a time evolution scheme in which sources would have their trajectories updated at rates that depend on their separations.

The fast multipole method (FMM) was developed beginning in the 1980s [4, 7, 15, 17, 32]; it can be used to evaluate sums of the form (2) in $\mathcal{O}(N)$ time. Independently, tree codes [2] were developed which evaluate (2) in $\mathcal{O}(N \log N)$ time. We do not attempt to give a complete description of the FMM here, though we introduce elements of the method in section 2. See [3, 26] for further details on the FMM. Among recent developments in fast multipole methods are kernel-independent methods [25, 36]; parallel codes using GPUs [37], multicore processors [5], and heterogeneous architectures [22]; and volume integral methods [24].

The FMM framework can be used to construct solvers for elliptic PDEs and integral equations; a different approach to the construction of fast algorithms for integral equations involves skeletonization [6, 23], or interpolative decomposition, for the operator or, especially, a factorization of its inverse [12, 27], which offers some advantages over particle methods for these problems. In particular, [27] discretizes an integral equation and updates a hierarchical factorization of the resulting linear system with $N$ degrees of freedom in sub-linear time. The update corresponds to a local $k$-point perturbation of the problem geometry or parameters which results in changes to localized blocks of the system matrix. With a hierarchical skeletonization, the asymptotic complexity of an update is $\mathcal{O}(k \log^4 N)$. A notable benefit of the method is that it results in an updated factorization of the system matrix, not just a method for solving the system (a solve requires $\mathcal{O}(N)$ time). The method presented in this paper uses a uniform hierarchical decomposition of the problem domain which is inefficient for highly non-uniform particle distributions, while [27] uses the more efficient adaptive tree; however, [27] is limited to two-dimensional domains where this paper considers three-dimensional domains. The work we present here can be considered a special function method in contrast to the linear algebra based method

of [27]. The methods mentioned in this paragraph are limited to problems which originate with integral operators, rather than particles, and currently are implemented in two dimensions. This paper provides a fundamental advance for particle methods, implemented in three dimensions.

In [11] a tree code updating scheme is presented for a Metropolis-Hastings Monte Carlo simulation of a three-dimensional charged colloidal system containing macro-ions and small ions. The method uses truncated multivariate Taylor polynomials to represent the long-range electrostatic interactions between ions, and requires $\mathcal{O}(\log N)$ computation time per single-particle Monte Carlo move. Simulation results demonstrate sub-linear scaling of computation time and decreasing error with increasing order of polynomial. In this work we allow simultaneous shift of multiple particles in either individual or group-wise fashion and use spherical harmonics for the far-field interactions, enabling full control of expansion error.

The paper is organized as follows. In section 2 we provide an overview of the multipole expansion and related formulas. Section 3 describes the new multipole-based potential energy functional updating algorithm. We present numerical results in section 4. Finally, in section 5 we summarize the present work and discuss directions for future work.

**2. Mathematical preliminaries.** In subsection 2.1 we derive an expression for the potential energy functional following a shift of $k$ particles. Subsections 2.2, 2.3, 2.4 and 2.5 provide background from the fast multipole method that will be used in section 3. For more details on the background material presented in subsections 2.2 to 2.5, the interested reader is referred to [17] and the references therein.

**2.1. Problem formulation.** We consider an ensemble of $N$ particles, indexed $\{1,\dots,N\}$. The particle indexed by $i$ has charge $q_i \in \mathbb{R}$ and is located at $x_i \in D \subset \mathbb{R}^3$. The potential at $x_i$ due to the other $N-1$ charges is given by $\Phi_i(\mathbf{x})$ (1), and the potential energy functional of the ensemble is given by $V(\mathbf{x},\mathbf{q})$ (2). If the $i$-th particle shifts we let $\tilde{x}_i$ denote its changed position. If $k$ particles undergo a shift we let $\mathcal{S}$ denote the indices of particles that shift, and $\mathcal{U}$ denote the indices of particles that remain unchanged. Then the full position vector, as changed, is

$$\tilde{\mathbf{x}} = \langle \xi_1,\dots,\xi_n\rangle, \quad \text{where } \xi_i = \begin{cases} \tilde{x}_i & i \in \mathcal{S} \\ x_i & i \in \mathcal{U}. \end{cases} \tag{3}$$

We also let $\tilde{\mathbf{x}}_{\mathcal{S}}$ denote the vector containing just the $k$ shifted positions, $\mathbf{x}_{\mathcal{S}}$ the vector of their $k$ original positions, and $\mathbf{q}_{\mathcal{S}}$ the vector of the $k$ charge values. Now, if particles in a subset move, as in an MCMC simulation, we wish to compute the potential energy functional $V_{\text{new}} = V(\tilde{\mathbf{x}},\mathbf{q})$ of the new configuration.

The new functional is

$$\begin{aligned} V_{\text{new}} &= \sum_{i\in\mathcal{S}\cup U} q_i \Phi_i(\tilde{\mathbf{x}}) \\ &= \sum_{i\in\mathcal{S}} q_i \left( \sum_{\substack{j\in\mathcal{S}\\ j\neq i}} \frac{q_j}{|\tilde{x}_i - \tilde{x}_j|} + \sum_{j\in\mathcal{U}} \frac{q_j}{|\tilde{x}_i - x_j|} \right) + \sum_{i\in\mathcal{U}} q_i \left( \sum_{j\in\mathcal{S}} \frac{q_j}{|x_i - \tilde{x}_j|} + \sum_{\substack{j\in\mathcal{U}\\ j\neq i}} \frac{q_j}{|x_i - x_j|} \right). \end{aligned} \tag{4}$$

Many of the terms in (4) are redundant with (2). Now we adopt the perspective that to compute $V_{\text{new}}$ from the pre-shift functional $V_{\text{old}}$ we remove $k$ particles from the

ensemble, and subtract their contribution to the functional $V$. Then we introduce $k$ particles (at the new locations $\tilde{\mathbf{x}}_{\mathcal{S}}$) and add their contribution to the functional.

To arrive at an expression of the form $V_{\text{new}} = V_{\text{old}} - \{\text{potential due to charges at old position}\} + \{\text{potential due to charges at new position}\}$ we add and subtract terms to (4) and re-arrange to match this form. We begin with

$$\begin{aligned} V_{\text{new}} = V_{\text{old}} &- \sum_{i\in\mathcal{S}} q_i\Phi_i(\mathbf{x}) - \sum_{i\in\mathcal{U}} q_i\Phi_i(\mathbf{x}) \\ &+ \sum_{i\in\mathcal{S}} q_i\Phi_i(\tilde{\mathbf{x}}) + \sum_{i\in\mathcal{U}} q_i\Phi_i(\tilde{\mathbf{x}}) \\ &+ \sum_{i\in\mathcal{S}} q_i\Phi_i(\tilde{\mathbf{x}}) - \sum_{i\in\mathcal{S}} q_i\Phi_i(\tilde{\mathbf{x}}) - \sum_{i\in\mathcal{S}} q_i\Phi_i(\mathbf{x}) + \sum_{i\in\mathcal{S}} q_i\Phi_i(\mathbf{x}) \end{aligned} \tag{5}$$

where the first line is zero, the second line is the new potential, and the third line is zero as well. Combining the first sum on line one with the third sum on line three, and the first sum on line two with the first sum on line three, we obtain

$$\begin{aligned} V_{\text{new}} = V_{\text{old}} &- 2\sum_{i\in\mathcal{S}} q_i\Phi_i(\mathbf{x}) + 2\sum_{i\in\mathcal{S}} q_i\Phi_i(\tilde{\mathbf{x}}) \\ &+ \sum_{i\in\mathcal{S}} q_i\Phi_i(\mathbf{x}) + \sum_{i\in\mathcal{U}} q_i\Phi_i(\tilde{\mathbf{x}}) - \sum_{i\in\mathcal{S}} q_i\Phi_i(\tilde{\mathbf{x}}) - \sum_{i\in\mathcal{U}} q_i\Phi_i(\mathbf{x}) \end{aligned} \tag{6}$$

where the last line can be simplified further. We expand the $\Phi$ terms in the last line according to (1) to write them as

$$\begin{aligned} &\sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{S}\cup\mathcal{U}, j\neq i} \frac{q_iq_j}{|x_i - x_j|} + \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{S}} \frac{q_iq_j}{|x_i - \tilde{x}_j|} + \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{U}, j\neq i} \frac{q_iq_j}{|x_i - x_j|} \\ &\quad - \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{S}, j\neq i} \frac{q_iq_j}{|\tilde{x}_i - \tilde{x}_j|} - \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{U}} \frac{q_iq_j}{|\tilde{x}_i - x_j|} - \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{S}\cup\mathcal{U}, j\neq i} \frac{q_iq_j}{|x_i - x_j|} \end{aligned} \tag{7}$$

and we see that the middle term from the first line and the middle term from the second line cancel. Expanding the first and last terms of this expression

$$\begin{aligned} &\sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{S}\cup\mathcal{U}, j\neq i} \frac{q_iq_j}{|x_i - x_j|} - \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{S}\cup\mathcal{U}, j\neq i} \frac{q_iq_j}{|x_i - x_j|} = \\ &\sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{S}, j\neq i} \frac{q_iq_j}{|x_i - x_j|} + \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{U}} \frac{q_iq_j}{|x_i - x_j|} - \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{S}} \frac{q_iq_j}{|x_i - x_j|} - \sum_{i\in\mathcal{U}} \sum_{j\in\mathcal{U}, j\neq i} \frac{q_iq_j}{|x_i - x_j|}, \end{aligned} \tag{8}$$

the only surviving terms are the fourth term of (7) and the first term of the expanded (8). The simplified expression for the last line of (6) is then

$$\sum_{i\in\mathcal{S}} q_i\Phi_i(\mathbf{x}_{\mathcal{S}}) - \sum_{i\in\mathcal{S}} q_i\Phi_i(\tilde{\mathbf{x}}_{\mathcal{S}}) \tag{9}$$

where we have returned to using $\Phi$ for compactness. A major benefit of this expression is that these terms involve only the $k$ particles being shifted, not the full ensemble of $N$ particles. This, along with a multipole expansion and tree structure, will allow our updating method to be of complexity less than $\mathcal{O}(N)$.

Now we can write the new potential energy functional of the ensemble in the desired form:

$$V_{\text{new}} = V_{\text{old}} - 2\sum_{i\in\mathcal{S}} q_i\Phi_i(\mathbf{x}) + \sum_{i\in\mathcal{S}} q_i\Phi_i(\mathbf{x}_{\mathcal{S}}) + 2\sum_{i\in\mathcal{S}} q_i\Phi_i(\tilde{\mathbf{x}}) - \sum_{i\in\mathcal{S}} q_i\Phi_i(\tilde{\mathbf{x}}_{\mathcal{S}}). \tag{10}$$

The form of (10) has a natural interpretation: the change in potential is due to simultaneous removal of particles, inclusion of particles at the new locations, and "correction" terms of the potential energy due to interactions among the particles being shifted.

If only one particle, with index $\ell$, is moved then (10) simplifies to

$$V_{\text{new}} = V_{\text{old}} - 2q_\ell \Phi_\ell(\mathbf{x}) + 2q_\ell \Phi_\ell(\tilde{\mathbf{x}}_\ell) \tag{11}$$

where we have written $\tilde{\mathbf{x}}_\ell$ to indicate $\tilde{x}_\ell$ replaces $x_\ell$ in $\mathbf{x}$. Successive application of (11) is equivalent to (10).

In the remainder of this section, we review formulae, error bounds, and algorithms from the FMM that will be needed in defining and analyzing our updating multipole method.

**2.2. Multipole expansion.** Any harmonic function $\Phi$ can be expanded as

$$\Phi(r,\theta,\phi) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left( L_n^m r^n + \frac{M_n^m}{r^{n+1}} \right) Y_n^m(\theta,\phi) \tag{12}$$

where $Y_n^m$ are the spherical harmonics obtained by solving Laplace's equation via separation of variables in spherical coordinates $(r,\theta,\phi)$. The terms $Y_n^m(\theta,\phi)/r^{n+1}$ are called the *multipoles* and the coefficients $M_n^m$ the *moments* of the expansion. In this work we will only be concerned with convergence *outside* a small region about $r = 0$ and thus $L_n^m \equiv 0$ to ensure decay to zero as $r \to \infty$. In the FMM, the $r^n Y_n^m(\theta,\phi)$ terms (the *local* expansion) are used for fast evaluation at each location; conversion from multipole to local expansions is at the heart of the FMM.

We define the spherical harmonics of degree $n$ and order $m$ as

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}}\ P_n^{|m|}(\cos\theta)\ e^{im\phi} \tag{13}$$

where $i^2 = -1$ and $P_n^m$ are associated Legendre functions. The latter may be defined using Rodrigues' formula

$$P_n^m(x) = (-1)^m \left(1-x^2\right)^{m/2} \frac{d^m}{dx^m} P_n(x) \tag{14}$$

where $P_n(x)$ is the Legendre polynomial of degree $n$. Notice that we have omitted from the spherical harmonics a normalization factor $\sqrt{(2n+1)/4\pi}$. Since the potential (1) is harmonic, we express, and compute, potentials using multipole expansions.

**2.3. Truncated multipole representation.** We will divide the domain $D$ into subdomains each containing $s \ll N$ charges. For a charge indexed $i$, sufficiently far away from the subdomain, we will evaluate the potential at $x_i$ due to all charges in the subdomain. We wish to aggregate the potential due to these charges into a single multipole representation and truncate the series to a finite number of terms. The following theorem (see [7, 14, 16, 17]), derived from an addition theorem for Legendre polynomials, provides a formula for the coefficients $M_n^m$ for the desired aggregate potential and provides an error bound for the truncation.

THEOREM 1 ([7, Theorem 2.1]). *Suppose $N$ charges $q_1,\ldots,q_N$ at locations $x_1,\ldots,x_N$ with spherical coordinates $(\rho_1,\alpha_1,\beta_1),\ldots,(\rho_N,\alpha_N,\beta_N)$ have $\rho_i < a$, for*

*$i=1,\ldots,N$. That is, $N$ charges lie inside a sphere of radius $a$ centered at the origin. Then, for $x=(r,\theta,\phi)$ with $r>a$, the potential $\Phi(x)=\sum_{i=1}^{N}\frac{q_i}{|x-x_i|}$ is*

$$\Phi(x)=\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{M_n^m}{r^{n+1}}\,Y_n^m(\theta,\phi) \tag{15}$$

*where*

$$M_n^m=\sum_{i=1}^{N}q_i\rho_i^n\,\overline{Y}_n^m(\alpha_i,\beta_i). \tag{16}$$

*Furthermore, for $p\geq 1$,*

$$\left|\Phi(x)-\sum_{n=0}^{p}\sum_{m=-n}^{n}\frac{M_n^m}{r^{n+1}}\,Y_n^m(\theta,\phi)\right|\leq\left(\frac{\sum_{i=1}^{N}|q_i|}{r-a}\right)\left(\frac{a}{r}\right)^{p+1}. \tag{17}$$

That is, we may evaluate the potential at a point outside the sphere of radius $a$ using a single multipole expansion for all $N$ charges inside the sphere with aggregated coefficients (the aggregation represents superposition of the potential fields). Truncation of the expansion to order $p$ (i.e. retaining $(p+1)^2$ terms) results in an error proportional to the total charge $Q=\sum_{i=1}^{N}|q_i|$ and, for fixed distance from the sphere, decreasing with increasing order $p$. Notice that the error becomes large as $r\to a$; we will use this representation to evaluate the potential in subdomains well-separated from the domain containing the charges.

**2.4. Translation of a multipole expansion.** Fast multipole methods rely on a hierarchical decomposition of the domain $D$ into an octree structure. At each level $\ell$ of the tree, the domain is composed of $8^\ell$ subdomains. It will be useful to store multipole expansions in each subdomain at each level of the tree. We will want to translate the center of expansions from the center of a subdomain at level $\ell$ to the center of a subdomain at level $\ell-1$. Indeed, translation of multipole expansions is a core part of fast multipole methods.

THEOREM 2 ([7, Theorem 2.3]). *Suppose we have a multipole expansion with coefficients $O_n^m$ for the potential due to charges $q_1,\ldots,q_N$ at locations $x_1,\ldots,x_N$ inside the sphere of radius $a$ with center $x_0$ having spherical coordinates $(\rho_0,\alpha_0,\beta_0)$. Let $x$ be a point outside the sphere of radius $a+\rho_0$ centered at the origin[1], and let $(r',\theta',\phi')$ be the spherical coordinates of the vector $x-x_0$. Then by (15),*

$$\Phi(x)=\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{O_n^m}{r'^{n+1}}\,Y_n^m(\theta',\phi') \tag{18}$$

*is the potential at $x$ with multipole expansion centered at $x_0$.*

*The multipole expansion centered at the origin for $x=(r,\theta,\phi)$ with $r>a+\rho_0$ is*

$$\Phi(x)=\sum_{j=0}^{\infty}\sum_{k=-j}^{j}\frac{M_j^k}{r^{j+1}}\,Y_j^k(\theta,\phi) \tag{19}$$

[1]For (18) to hold is only necessary that $x$ be outside the sphere of radius $a$ centered at $x_0$ but this condition is more convenient for (19) to hold.

*where*

$$M_j^k = \sum_{n=0}^{j} \sum_{m=-n}^{n} \frac{O_{j-n}^{k-m}\, i^{|k|-|m|-|k-m|}\, A_n^m\, A_{j-n}^{k-m}\, \rho_0^n\, Y_n^{-m}(\alpha_0, \beta_0)}{A_j^k} \tag{20}$$

*and*

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!\,(n+m)!}}. \tag{21}$$

*Furthermore, for* $p \geq 1$*,*

$$\left| \Phi(x) - \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}}\, Y_n^m(\theta, \phi) \right| \leq \left( \frac{\sum_{i=1}^{N} |q_i|}{r - (a + \rho_0)} \right) \left( \frac{a + \rho_0}{r} \right)^{p+1}. \tag{22}$$

The multipole translation theorem allows us to move a multipole expansion's center from $x_0$ to the origin while maintaining a bound on the approximation error for a truncated expansion. Indeed, we may now move an expansion from center $x_0$ to any other point.

The multipole translation formula (20) defines a linear operator $T_{MM}$ mapping the coefficients $\{O_n^m\}$ to the coefficients $\{M_n^m\}$. The cost of applying $T_{\mathrm{MM}}$ is $\mathcal{O}(p^4)$ since it maps $(p+1)^2$ coefficients via a dense $(p+1)^2 \times (p+1)^2$ matrix. The cost of translation can be reduced by rotating the coordinate system with a rotation $\Omega$ to align the direction of translation with the $z$-axis, at a cost of $\mathcal{O}(p^3)$. Translation along the $z$-axis is then diagonal, or $\mathcal{O}(p^2)$. Rotating back to the original coordinate system requires the rotation $\Omega^{-1}$ and an additional $\mathcal{O}(p^3)$. Denoting rotation by $R(\,\cdot\,)$ and translation along the $z$-axis by $T_{\mathrm{MM}}^z$ we have $T_{\mathrm{MM}} = R(\Omega^{-1}) \circ T_{\mathrm{MM}}^z \circ R(\Omega)$ which in this form is applied at a cost of $\mathcal{O}(p^3)$. For details and formulae we refer the interested reader to [7]. Below, in subsection 3.2.1, we note that to update the potential energy functional, it may be more efficient to re-compute the coefficients for the multipole expansion using (16), which requires $\mathcal{O}(p^2)$ operations for each particle, than to use a translation operator.

**2.5. The fast multipole method.** Here we briefly describe the classical FMM for evaluating the potential energy functional of a collection of charges. For a more complete description of the FMM, including asymptotic analysis of operation costs, see [7].

The FMM relies on hierarchical decomposition of the computational domain into an octree (either fixed or adaptive), multipole representation of the potential along with translation operators for multipole coefficients, conversion to expansions which converge locally rather than in the far-field, and translation of local expansion coefficients.

Some data structures are required to keep track of interactions between subdomains of the full computational domain. The desired number of particles per box at the leaf level of the octree is chosen by the user. This determines the number of boxes at the leaf level and therefore the number of levels $L$. The computational domain is recursively subdivided into eight *children* until the leaf level is reached. The whole domain is labeled level 0 and each successive subdivision increments the level number. The child of a box refers to that box as its *parent*, and the sequence of parents containing a leaf box is that box's *lineage*.

Each leaf box forms a list of the particles residing in that box. Every box in the tree (except at levels 0 and 1) forms two additional lists: its *neighbors* and its

*interaction list*. Two boxes are neighbors (or nearest neighbors) if they are at the same level of the tree and share a boundary point. The neighbor list of a box $b$ is denoted $Nbhd(b)$. In two dimensions (i.e. in a *quadtree*) a typical box has nine neighbors including itself. In an octree a typical box has $3^3 = 27$ neighbors including itself. Boxes at a boundary of the domain have fewer than 27 neighbors. Two boxes are *well-separated* if they are at the same level of the tree and are not nearest neighbors. The interaction list of a box $b$, $Intlist(b)$, is the set of well-separated boxes which are children of $b$'s parent's neighbors. In an octree the interaction list for a typical box contains $(2 \cdot 3)^3 - 3^3 = 189$ boxes. The interaction lists at levels 0 and 1 are empty.

The fast multipole algorithm consists of an *upward pass* through the octree followed by a *downward pass*. The final step evaluates the potential at each particle.

In the upward pass, multipole expansions are computed and stored at each box in the tree. First for each leaf-level box, at level $\ell = L$, one forms the truncated multipole expansion (15) for the collection of charges in that box. At level $l - 1$ each box $b$ translates the expansion of its eight children via $T_{MM}$ to the center of $b$ and adds these expansion coefficients. The result is a multipole expansion representing all charges below box $b$ in the tree. This process repeats recursively up to $\ell = 2$, completing the upward pass.

The downward pass converts multipole (far-field) expansions to local expansions and passes these down the tree. Three theorems similar to Theorem 1 and Theorem 2 establish (1) a truncated expansion of the form (12) with $M_n^m \equiv 0$ which converges *inside* a box well-separated from the collection of charges, (2) a linear operator $T_{ML}$ which converts the multipole expansion coefficients $M_n^m$ in a box $b_1$ to the local expansion coefficients $L_n^m$ for a box $b_2$ provided the boxes $b_1$ and $b_2$ are well-separated, and (3) a linear operator $T_{LL}$ to translate the center of a local expansion. First the downward pass converts all multipole expansions to local expansions. Each box in the tree uses $T_{ML}$ to convert the multipole expansion from each box in its interaction list to a local expansion, summing the converted expansions. Then, beginning at level $l = 2$ the downward pass uses $T_{LL}$ to translate the local expansion from each box at level $l$ to each box's eight children, where the parent coefficients are added to the coefficients from multipole-to-local conversion. This proceeds recursively down to the leaf level. At the end of the downward pass, the leaf boxes have a locally-converging expansion for the potential due to well-separated boxes.

Finally, the potential is evaluated at each particle. The local expansion is evaluated at the particle location, and the direct expression (1) is evaluated for all particles $x_j$ in the neighbor list of the box in which the particle resides. Summed over all particles, this is the value of the potential energy functional of the ensemble.

Note that following application of $T_{ML}$, all multipole coefficients may be discarded as the remainder of the method no longer requires them. In the updating method we will retain these throughout as we will use the multipole expansion to evaluate the potential instead of the local expansion.

**3. Updating multipole method.** In this section we describe in detail our $\mathcal{O}(\log N)$ algorithm for computing the potential energy functional following a shift via (10).

**3.1. Data structures.** Several modifications and additions to the data structures for the FMM are necessary to adapt it to the updating multipole method: (i) the list of particles in each leaf-level box must be dynamically updated as shifts may cause particles to cross box boundaries; (ii) the moments (multipole expansion coefficients) for each particle are stored, not only the coefficients for each box.

While (ii) is not necessary, storing each particle's coefficients avoids re-computing them when removing the particle from the data structure in a delete-insert shift (see below for details). This amounts to a storage-computation tradeoff. In the FMM, once each particle's moments have been added to its box's aggregate coefficients the particle moments are no longer needed and thus may be discarded.

**3.2. Update.** To shift $k$ particles we mirror the form of (10): We delete the $k$ particles from the data structures, subtract the potential due to these particles from the total, insert $k$ particles at the new locations into the data structures, and add the potential due to these particles to the potential energy functional. The two sets of pre- and post-shift terms in (10) differ only by the locations $\tilde{\mathbf{x}}_{\mathcal{S}}$ and sign in front of the summations. Thus, removal of particles is equivalent to introduction of particles with the opposite charge $-\mathbf{q}_{\mathcal{S}}$. We describe introduction of particles in detail.

For convenience we re-write (10) as

$$V_{\text{new}} = V_{\text{old}} \; - \; \Delta V_{\text{del}} \; + \; \Delta V_{\text{ins}} \tag{23}$$

where

$$\Delta V_{\text{ins}} = \; 2\sum_{i\in\mathcal{S}} q_i \Phi_i(\tilde{\mathbf{x}}) - \sum_{i\in\mathcal{S}} q_i \Phi_i(\tilde{\mathbf{x}}_{\mathcal{S}}) \tag{24}$$

and

$$\Delta V_{\text{del}} = \; 2\sum_{i\in\mathcal{S}} (-q_i) \Phi_i(\mathbf{x}) - \sum_{i\in\mathcal{S}} (-q_i) \Phi_(\mathbf{x}_{\mathcal{S}}). \tag{25}$$

The second term in $\Delta V_{\text{ins}}$ (and in $\Delta V_{\text{del}}$) involves only the shifted particles, and in most cases these will all be in each others' neighborhood. Therefore we compute this term via the direct method, with a cost of $k(k-1)$ operations. The first term requires both multipole-interaction evaluation and direct-interaction evaluation. The main difference in this term between $\Delta V_{\text{ins}}$ and $\Delta V_{\text{del}}$ is the location at which the multipoles are evaluated, $\mathbf{x}_{\mathcal{S}}$ versus $\tilde{\mathbf{x}}_{\mathcal{S}}$.

**3.2.1. Insert.** The following steps are required for introducing a group of particles: insert the particles' information into the data structures; evaluate the change in the potential energy functional due to these particles, $\Delta V_{\text{ins}}$, and add this change to the potential energy functional of the entire configuration; and update the multipole coefficients of every box in each particles' lineage, from leaf-level up to level 2.

To insert particles to the data structures, we determine the boxes in which they will reside, and append the particles' indices to the list of indices for that box. For each particle, the box in which it resides can be stored to reduce on-the-fly calculations. Since there is no restriction that a group of particles must respect box boundaries, it is possible that particles from a group reside in several different boxes, and therefore our method must account for the most general case.

To evaluate the change in the potential energy functional $\Delta V_{\text{ins}}$ we compute, for each new particle indexed $i$, the potential at its location $x_i$ due to all particles in boxes in the interaction lists of the particle's lineage via the corresponding box-aggregated multipole representations, and the potential at $x_i$ due to all particles in its neighborhood via the direct formula. The 'correction' term we then evaluate using the direct formula.

To update the multipole coefficients in the leaf-level box, we use the property of superposition of potentials or equivalently the addition property of multipole expansions. For simplicity, consider the case where all new particles reside in a single

leaf-level box, labeled $b_L$. We will consider exceptions to this scenario in subsection 3.2.2. To update the multipole coefficients $\{M_n^m(b_L)\}$ of the leaf-level box, we compute the multipole coefficients $\{M_n^m(i)\}$ of each newly-inserted particle $i$, taking the center of the leaf-level box as the center of the expansion. Then each particle's multipole coefficients may be added to the coefficients of the box $b_L$ so that the box's coefficients represent the potential due to all particles now in the box, both those previously in the box and those newly added. That is, $M_n^m(b_L) = M_n^m(b_L) + \sum_{i\in\mathcal{S}} M_n^m(i)$ for $n = 1, \ldots, p$ and $m = -n, n$ for each $n$, where the $M_n^m(i)$ are computed via (16). Since it is convenient to store the coefficients $\{M_n^m(i)\}$ of each particle, we compute them for each particle separately using (16) with $N = 1$ rather than for all $k$ new particles at once with $N = k$. We also temporarily store the aggregate coefficients for the new particles $\{M_n^m(b_L)\}_{\mathcal{S}} = \{\sum_{i\in\mathcal{S}} M_n^m(i)\}$ to use in updating the remainder of the tree.

Now that the leaf-level box $b_L$ has been updated we traverse the octree up from level $\ell = L$ to $\ell = 2$ and update the multipole coefficients in each box of $b$'s lineage. That is, given a box $b$ at level $\ell$ and its parent $\mathcal{P}(b)$ at level $\ell - 1$, we re-express the multipole coefficients due to the shifted particles, $\{M_n^m(b)\}_{\mathcal{S}}$, centered about the midpoint of box $b$, so that they are centered about the midpoint of box $\mathcal{P}(b)$. Then we may add them to $\mathcal{P}(b)$'s stored coefficients. Following this step, $\mathcal{P}(b)$'s multipole coefficients will have been updated to include the effect of the new particles. Performing this translation of multipole coefficients traversing up $b_L$'s lineage completes the update of multipole coefficients due to the introduction of new particles. Since the octree has $L$ levels this step consists of $L - 2$ multipole-to-multipole translations. We choose $L \sim \log N$ so this step consists of $\mathcal{O}(\log N)$ translations, and is what drives the asymptotic complexity of the multipole updating method.

To perform each translation, we have three methods from which to choose: (i) apply the multipole-to-multipole linear operator $T_{\mathrm{MM}}$ requiring $\mathcal{O}(p^4)$ operations, (ii) apply the rotation-based $T_{\mathrm{MM}}$ operator requiring $\mathcal{O}(p^3)$ operations, or (iii) compute, for each of the $k$ new particles, the multipole coefficients using (16) with the locations of the new particles and center of expansion the midpoint of box $\mathcal{P}(b)$ requiring $\mathcal{O}(kp^2)$ operations. Our choice of which method to use depends on $k$, the constants in the complexity, and implementation-specific details. In our implementation, (i) is always slowest, and (iii) is fastest for small $k$ while (ii) is fastest for large $k$ and $N$.

**3.2.2. Special cases.** If the new particles reside in multiple leaf-level boxes then the above procedure for updating the multipole storage in the octree using $T_{\mathrm{MM}}$ requires modification: separate multipole translations are required for each box. Separate translations continue to be necessary at each level until one reaches a level where the multiple leaf-level boxes share a common ancestor. The worst-case scenario, where leaf-level boxes share no common ancestor until the entire computational domain, is possible even when the leaf-level boxes are adjacent. For example, consider eight particles with one in each of the eight leaf-level boxes with one corner at the center of the computational domain. That is, even if one takes care to only move tightly-clustered groups of $k$ particles, $\mathcal{O}(8 \log N)$ translations may be necessary.

It is possible to encounter the scenario where a new particle is introduced in each of the $8^L$ leaf-level boxes, but since $L \sim \log_8 N$ we reject this case as not a practical operating condition, and certainly one to be avoided if possible. In this case $64(1 - 8^L)/7$ translations are required; this is a tight upper bound though the upper bound is much lower in practice.

**3.2.3. Group-wise shift versus individual shifts.** The choice of method for propagating multipole coefficients up the octree amounts to a choice between updating the multipole representation of the collection of particles in a group-wise versus individual particle fashion.

The rotation-based $T_{\mathrm{MM}}$ can operate on the aggregate multipole coefficients for all newly-introduced particles together. For simplicity of discussion, let us assume that all $k$ new particles reside in a single leaf-level box. Each of the $(p+1)^2$ leaf-level coefficients $\{M_n^m(b_L)\}_{\mathcal{S}} = \{\sum_{i\in\mathcal{S}} M_n^m(i)\}$ require summing $k$ spherical harmonics, each evaluated at a point, for a total of $\mathcal{O}(kp^2)$ operations. Updating the multipole coefficients for all boxes in the leaf-level box's lineage requires $L-2$ translations, each of which costs $\mathcal{O}(p^3)$ operations. The total for updating the multipole storage of $k$ new particles is then on the order of $kp^2 + (L-2)p^3$.

Computing multipole coefficients for each particle about the center of each box in the leaf-level box's lineage requires computing (16) $L-1$ times, for a total of $\mathcal{O}((L-1)kp^2)$ operations.

We expect the constant for group-wise rotation-based $\mathcal{O}(kp^2 + (L-2)p^3)$ translation is smaller than the constant for individual $\mathcal{O}((L-1)kp^2)$ translation. We have implemented both methods and comment on the results of numerical experiments below in section 4, offering guidelines for circumstances for use of each method.

**3.3. Initialization.** Prior to shifting $k$ particles we must have computed the potential energy functional of the collection and stored multipole coefficients in the tree structure. This initialization step can be carried out either via an FMM modified to store otherwise discarded information or via repeated application of particle insertion beginning with an empty tree structure. If using an existing FMM code, one must store the multipole coefficients for each box; all local expansion coefficients may be discarded as they are not used in our updating method. At complexity $\mathcal{O}(N)$ this is the faster of the two methods but it requires additional code. A simpler method is to begin with an empty collection of particles: all box multipole coefficients are set to zero and total potential is zero. Then one introduces each of the $N$ particles with the insertion method described above, either one-at-a-time or in groups of $k$. The complexity is worse, at $\mathcal{O}(N \log N)$ and may be impractical for large $N$ but the additional code is minimal.

**3.4. Error bound.** We now show that the updating multipole method introduced above has an error bound which does not increase with repeated shifts.

The error in any multipole method is controlled by choice of $p$. For fast multipole methods the worst-case error in the truncated multipole expansion is a good indicator of the global error [26]. That is, the error in the multipole ensemble potential $V^p$, $|V^p - V|$, is similar to the the worst-case of (17). In three dimensions (17) decays as $(\sqrt{3}/3)^p$. Thus, given a desired accuracy $\epsilon$ a heuristic choice is $p = \log_{\sqrt{3}}(1/\epsilon)$ [3].

THEOREM 3. *Consider an ensemble of particles with real-valued charges* $\mathbf{q} = \langle q_1, \ldots, q_N\rangle$ *at locations* $\mathbf{x} = \langle x_1, \ldots, x_N\rangle$ *in a domain* $D \subset \mathbb{R}^3$. *Denote by* $\Phi_i^p(\mathbf{x}, \mathbf{q})$ *the potential* (1) *computed using a p-moment multipole expansion* (17). *Let* $V_{old} = V(\mathbf{x}, \mathbf{q})$ *be the potential energy functional* (2) *of the ensemble and let* $V_{old}^p$ *be the same quantity computed using p-moment expansions* $\Phi_i^p$. *Let a bound* $\epsilon > 0$ *on the error in the p-moment multipole approximation to the total potential of an arbitrary ensemble be given. That is,*

$$|V_{old}^p - V_{old}| < \epsilon. \tag{26}$$

*Following a k-particle shift,* $\mathbf{x} \to \tilde{\mathbf{x}}$, *denote by* $V_{new} = V(\tilde{\mathbf{x}}, \mathbf{q})$ *the new potential energy*

*functional* (4). *Let the new multipole potential energy functional* $V^p_{new}$ *be computed using* (10) *where the multipole potentials* $\Phi^p_i$ *are used throughout.*

*Then the bound on the error in the* $p$*-moment multipole approximation to the potential energy functional remains*

$$|V^p_{new} - V_{new}| < \epsilon. \tag{27}$$

*Proof.* The expressions (4) and (10) are numerically identical in the sense that computing $V^p_{\text{new}}$ via (10) using the updating algorithm is the same, up to machine precision, as computing $V^p_{\text{new}}$ via (4) using a full FMM from scratch. The multipole expansion coefficients stored in the tree structure will also be identical, up to machine precision, in the updating algorithm as in a full FMM from scratch. Updating the multipole representation and the potential energy functional does not introduce additional approximation error beyond that of the $p$-term FMM approximation. The post-shift value of the potential energy functional therefore has the same error bound as it had pre-shift.

If we have a bound $\epsilon$ on the error $|V^p - V|$ for an arbitrary ensemble, then it holds for both $(\mathbf{x}, \mathbf{q})$ and $(\tilde{\mathbf{x}}, \mathbf{q})$. That is, the bound on the error remains the same for $\tilde{\mathbf{x}}$ as it is for $\mathbf{x}$. ∎

*Remark* 4. By Theorem 3 we have an upper bound on the error in the total potential due to the updating multipole method which does not change following a $k$-particle shift. That is, *errors do not accumulate following shifts.*

*Remark* 5. Floating-point arithmetic errors could occur if a particle has been shifted only a very short distance. In this case the multipole coefficients being subtracted and added in the delete and insert steps will be nearly equal in floating point representation. Consider shifting particle $i$ in box $b$ containing particle indices $\mathcal{I}$ and write box $b$'s multipole coefficients as $M^m_n(\mathcal{I} \setminus \{i\}) + M^m_n(i)$. The post-shift box coefficients will be $M^m_n(\mathcal{I} \setminus \{i\}) + M^m_n(i) - M^m_n(i) + \tilde{M}^m_n(i)$ where $\tilde{M}^m_n(i)$ are the post-shift box coefficients for particle $i$. The floating-point arithmetic error in this case remains machine precision, and the multipole representation remains accurate.

**3.5. Algorithm.** Here we present formal descriptions of the insertion, potential evaluation, and update algorithms. The first, Algorithm 1, is a high-level routine for moving $k$ particles. It requires the data structures described in subsection 3.1 and the new particles' positions; it returns the change in potential due to the particle shift, and as a side effect updates the data structures. Algorithm 2 evaluates and returns the potential at particle $i$. Algorithm 3 inserts $k$ particles and performs the data structure updates of multipole expansions. It requires the new locations and charges of the particles along with the data structures.

**3.6. Complexity analysis.** Direct evaluation of the potentials in (10) requires $2(k(N-k) + k(k-1)/2) + 2k(k-1)/2 = 2k(N-1)$ unique terms; no data structure updates are needed. Thus the direct method's complexity is $\mathcal{O}(kN)$ for fixed $k$.

The cost of the updating multipole algorithm for a $k$-particle shift is the cost of evaluating the change in the potential energy functional plus the cost of updating the stored multipole coefficients. By (10), evaluating the change in the potential energy functional requires $2k$ times the cost of evaluating $\Phi_i$ (or equivalently $\tilde{\Phi}_i$) plus $2k$ times the cost of evaluating the potential among $k$ particles. The latter is computed using the direct summation expression (1) with cost $k(k-1)/2$. The former has two components: (i) multipole evaluations from the interaction list at all levels, which costs $189(L-1)(p+1)^2$ operations, and (ii) direct evaluation of interactions with

**Algorithm 1** Move $k$ particles from $\mathbf{x}_{\mathcal{S}}$ to $\tilde{\mathbf{x}}_{\mathcal{S}}$: `update`

$\Delta V_{\text{del}} = 0$
**for** $i \in \mathcal{S}$ **do** {Potential at $x_i$ due to all $N-1$ particles.}
  $\Delta V_{\text{del}} = \Delta V_{\text{del}} + 2\Phi_i(\mathbf{x})$ {Compute $\Phi_i$ using `evalpot` (Algorithm 2)}
**end for**
**for** $i \in \mathcal{S}$ **do** {Potential at $x_i$ due to particles $\{x_j\}_{j\in\mathcal{S}, j\neq i}$.}
  $\Delta V_{\text{del}} = \Delta V_{\text{del}} - \sum_{j\in\mathcal{S}, j\neq i} q_i q_j/|x_i - x_j|$ {Compute using direct method.}
**end for**
Call `insert` (Algorithm 3) with $\mathbf{x}_{\mathcal{S}}$ and $-\mathbf{q}_{\mathcal{S}}$
Delete particles $\mathbf{x}_{\mathcal{S}}$ from data structures
Insert particles $\tilde{\mathbf{x}}_{\mathcal{S}}$ into data structures
Call `insert` with $\tilde{\mathbf{x}}_{\mathcal{S}}$ and $\mathbf{q}_{\mathcal{S}}$
$\Delta V_{\text{ins}} = 0$
**for** $i \in \mathcal{S}$ **do** {Potential at $\tilde{x}_i$ due to all $N-1$ particles.}
  $\Delta V_{\text{ins}} = \Delta V_{\text{ins}} + 2\Phi_i(\tilde{\mathbf{x}})$ {Compute $\Phi_i$ using ***evalpot***.}
**end for**
**for** $i \in \mathcal{S}$ **do** {Potential at $\tilde{x}_i$ due to particles $\{\tilde{x}_j\}_{j\in\mathcal{S}, j\neq i}$.}
  $\Delta V_{\text{ins}} = \Delta V_{\text{ins}} - \sum_{j\in\mathcal{S}, j\neq i} q_i q_j/|\tilde{x}_i - \tilde{x}_j|$ {Compute using direct method.}
**end for**
**return** $\Delta V = \Delta V_{\text{ins}} - \Delta V_{\text{del}}$

**Algorithm 2** Evaluate $\Phi_i(\mathbf{x})$, the potential at $x_i$: `evalpot`

$\Phi = 0$
$b = b_L$, the leaf-level box in which $x_i$ resides
$\ell = L$
**while** $\ell \geq 2$ **do** {far-field potential due to particles in well-separated boxes}
  **for** each box $\beta \in Intlist(b)$ **do**
    $c = center(\beta)$ and $(r, \theta, \phi) = x_i - c$ {Convert to spherical coordinates}
    $\Phi = \Phi + \sum_{n=0}^{p} \sum_{m=-n}^{n} M_n^m(\beta)\, Y_n^m(\theta, \phi) \,/\, r^{n+1}$
  **end for**
  $b = Parent(b)$
  $\ell = \ell - 1$
**end while**
**for** each box $\beta \in Nbhd(b_L)$ **do** {direct potential due to particles in neighbor boxes}
  **for** each particle $j \in \beta$ **do**
    $\Phi = \Phi + q_i q_j / |x_i - x_j|$
  **end for**
**end for**
**return** $\Phi$

neighbor boxes, which costs $27s$ operations where $s$ is the number of particles per box at the leaf level. Updating the stored multipole coefficients costs $kp^2 + (L-2)p^3$ for group-wise update or $(L-1)kp^2$ for individual update.

The total cost is then approximately

$$2k\left(189Lp^2 + 27s\right) + 2kk^2 + (kp^2 + Lp^3), \tag{28}$$

where we have assumed rotation-based group-wise update. Since $L \sim \log N$, we have

**Algorithm 3** Introduce $k$ particles: `insert`

$\mathcal{B} = \{\ \}$
**for** $i \in \mathcal{S}$ **do**
  $\mathcal{B} = \mathcal{B} \cup \{$leaf-level box in which $x_i$ resides$\}$
**end for**
**for** each box $b \in \mathcal{B}$ **do**
  $C_n^m(b) = 0$ for $n = 0, \ldots, p$ and $m = -n, \ldots, n$
  $c = center(b)$
  **for** each particle $j \in b$ **do**
    $(\rho_j, \alpha_j, \beta_j) = x_j - c$ {Convert to spherical coordinates}
    **for** $n = 0, \ldots, p$ and $m = -n, \ldots, n$ **do**
      $M_n^m(j) = q_j \rho_j^n \, Y_n^{-m}(\alpha_j, \beta_j)$ {Compute multipole coefficients}
      $M_n^m(b) = M_n^m(b) + M_n^m(j)$ {Add to leaf-level box}
      **if** group-wise point-and-shoot translation **then**
        $C_n^m(b) = C_n^m(b) + M_n^m(j)$ {Aggregate and store for translating}
      **end if**
    **end for**
  **end for**
**end for**
$\ell = L$ {Begin at leaf level}
**if** group-wise point-and-shoot shift **then**
  **while** $l > 2$ **do**
    $\mathcal{P} = \{\ \}$
    **for** each box $b \in \mathcal{B}$ **do**
      $P = Parent(b)$
      $C_n^m(P) = \mathcal{T}_{\mathrm{MM}} C_n^m(b)$ {Translate to parent}
      $M_n^m(P) = M_n^m(P) + C_n^m(P)$ {Add to parent box}
      $\mathcal{P} = \mathcal{P} \cup \{P\}$
    **end for**
    $\mathcal{B} = \mathcal{P}$
    $\ell = \ell - 1$ {Go up one level}
  **end while**
**else** {using individual shifts}
  **for** $i \in \mathcal{S}$ **do**
    $P(i) = Parent(i)$
  **end for**
  **while** $l > 2$ **do**
    **for** $i \in \mathcal{S}$ **do**
      $c = center(P(i))$
      $(\rho_i, \alpha_i, \beta_i) = x_j - c$
      **for** $n = 0, \ldots, p$ and $m = -n, \ldots, n$ **do**
        $C_n^m(i) = q_i \rho_i^n \, Y_n^{-m}(\alpha_i, \beta_i)$ {Expand about parent box's center}
        $M_n^m(P(i)) = M_n^m(P(i)) + C_n^m(i)$ {Add to parent box}
      **end for**
      $P(i) = Parent(P(i))$
    **end for**
    $\ell = \ell - 1$ {Go up one level}
  **end while**
**end if**

on the order of

$$(378kp^2 + p^3)\log N + 54ks + 2k^3 + kp^2 \tag{29}$$

operations per update. For individual update, the corresponding expression is

$$379kp^2 \log N + 54ks + 2k^3. \tag{30}$$

We use a heuristic choice of $s$ independent of $N$; see section 4 for further information regarding choice of $s$ in practice. Thus for fixed $p$, $k$, and $s$, the method scales logarithmically in the number of particles $N$.

For highly nonuniform distributions of particles, it can be advantageous to use an *adaptive* octree. Since the depth $L$ of an adaptive tree is not known *a priori*, the assumption $L \sim \log N$ no longer holds and we do not know the number $T$ of multipole translations in the updating method. The number of upward pass translations in an adaptive tree for the FMM is bounded by $2B$ where $B$ is the number of leaf-level boxes [29]. One aims to have approximately $s > p$ particles per box, so $B \approx N/s$ and we have at most $2N/s$ upward pass translations. We expect the multipole coefficient updating step to require much less than the full FMM up-pass step. Thus in non-pathological cases $T < N$ and the estimates (29) and (30) remain sub-linear in $N$.

**4. Numerical results.** In this section we present the results of numerical experiments using our updating multipole method.

**4.1. Numerical experiment set-up.** We implemented the updating multipole algorithm in Fortran 90. Numerical experiments were performed on a desktop computer with two 3.7GHz quad-core CPUs and 32GB of memory. We use subroutines from [13] to compute multipole coefficients, evaluate multipole expansions, and to compute and apply the rotation-based translation operator $T_{\mathrm{MM}}$.

For experiments with $k = 1$, that is with non-grouped individual particles, charges are distributed uniformly throughout the domain $D$, for which we use the unit cube $D = [0, 1]^3$. For $k > 1$, charges are distributed as follows. Each group is uniformly distributed in a small cube with side 1/100-th the width of a leaf-level box. The center of the group's cube is uniformly distributed over the domain. This distribution of groups mimics, for example, molecules in solution. Charge values $q$ are distributed uniformly in the interval $[0, 1)$. To compute the initial potential, we use the `insert` routine of Algorithm 3 rather than a full FMM. This imposes a speed cost but avoids the code complexity of a modifying a full FMM to store multipole coefficients. To choose the number of levels $L$ in the octree we use a heuristic on the average number of particles per leaf-level box. More particles per box provides greater compression into the $(p+1)^2$ coefficients of the multipole expansion but comes at the cost of increased direct interactions with particles in neighbor boxes. In practice this trade-off may be application- and implementation-dependent. We limit the number of particles per leaf-level box to $s = 350$, so $L = \lceil \log_8 (N/s) \rceil$ where $\lceil \cdot \rceil$ indicates the *ceiling* function which returns the smallest integer larger than the argument. Note that the choice of $s$ is independent of $N$, so the asymptotic scaling is still $\mathcal{O}(\log N)$ as discussed in subsection 3.6. In our implementation and numerical experiments we do not observe a dependence on $p$ or $k$ in this heuristic. Compared to the full FMM we observe a change in the balance of multipole and direct calculations in the direction of larger $s$, smaller $L$, and more direct calculations.

Once the initial potential energy functional has been computed, we conduct 1000 shifts as follows. A particle ($k = 1$) or group ($k > 1$) is chosen at random. Each

component of a shift vector $\Delta x$ is drawn from a uniform random distribution, scaled to be at most 1/500-th the width of a leaf-level box. Each particle in the group is moved by the vector $\Delta x$. The total number of shifted particles is $1000 \cdot k$. For simplicity, intra-group shifts are not included. These shifts mimic, for example, the types of shifts that would occur in an MCMC simulation of molecules in solution; the application setting determines the shift vectors. For these experiments, the maximal shift size and group cube size are chosen to reduce the number of shifts that move a group from residing entirely in one box to being distributed across two or more. Of course such shifts do occur and the algorithm handles this case correctly (see subsection 3.2.2). In the tables below we report the time to execute one shift of one particle, in seconds. That is, we report (total time spent shifting)/$(1000 \cdot k)$.

**4.2. Individual vs group-wise coefficient update.** Recall that in updating stored multipole coefficients to reflect the new position of shifted particles, we pass changes from the leaf level $\ell = L$ up the octree to level $\ell = 2$. To do so we have two options: update coefficients on an individual particle or group-wise basis (see subsection 3.2.3). Both updates require first computing the multipole coefficients for each particle in the group at the leaf-level box, at a cost proportional to $kp^2$ (for simplicity of analysis we assume for now that all particles reside in a single leaf-level box) followed by $L-2$ translations up the octree. The group-wise update aggregates the $k$ sets of coefficients by summing and uses the rotation-based $T_{\mathrm{MM}}$ to (recursively) translate the single summed set of coefficients to the parent box at a cost proportional to $p^3$ per shift for a total cost of $\mathcal{O}(kp^2 + (L-2)p^3)$. The individual update independently propagates each particle's multipole coefficients up the tree by re-computing the multipole coefficients (16) for the parent box at a cost proportional to $p^2$; repeating this up to level $\ell = 2$ and for each particle results in a total of $\mathcal{O}(kp^2 + k(L-2)p^2)$.

Clearly the size $k$ of a group of particles and the order $p$ of expansion impact whether to update individually or group-wise. For large $k$ we expect group-wise updates to be faster. The smaller complexity in $p$ implies that individual updates may be faster as $p$ increases. However, a complication in determining which method to use is that the constants of proportionality for the two methods are quite different: group-wise update requires application of $T_{\mathrm{MM}}$ while individual updates require evaluating Legendre polynomials. Overall for fixed $p$ we expect the group-wise shift to become faster as the number of particles in the group increases.

For groups whose particles are distributed across more than a single box, we expect the benefit of group-wise shift to be delayed. For a group covering $m$ leaf-level boxes, up to $m$ applications of $T_{\mathrm{MM}}$ are required at each level until a level is reached such that the $m$ boxes share a common ancestor. Thus we expect the number of particles in a group at which group-wise shift becomes faster than individual shift to be larger than in the case where we have only one leaf-level box involved.

**4.3. Timings.** Tables 1 to 5 compare the updating multipole method to the direct method for $p = 3, 6, 9, 12, 15$ and $k = 1$. The times reported are the time to shift one particle, in seconds. We report $\epsilon_{\max}$, the maximum relative error over all shifts of the updating multipole potential energy functional relative to the direct potential energy functional, where the relative error is $\epsilon = \left|V^p_{\mathrm{new}} - V^{\mathrm{D}}_{\mathrm{new}}\right| / \left|V^{\mathrm{D}}_{\mathrm{new}}\right|$ and $V^{\mathrm{D}}$ is the potential energy functional computed using the direct summation expression (1) for $\Phi$.

Note that the scaling with $N$, of $\mathcal{O}(\log N)$ for multipole updating versus $\mathcal{O}(N)$ for direct updating, appears evident in the data. For large $N$, doubling the number

TABLE 1

*Timings for direct and multipole updates for $k = 1$ and $p = 3$. The values $t_{mple}$ and $t_{dir}$ are time per shift of one particle, in seconds, for the multipole updating method and the direct method, respectively. The value $\epsilon_{max}$ is the maximum relative error occurring during the 1000 shifts.*

| $N$ | $L$ | $t_{\text{mple}}$ | $t_{\text{dir}}$ | $\epsilon_{\text{max}}$ |
|---|---|---|---|---|
| 3000 | 2 | 4.8E-05 | 4.4E-05 | 4.7E-05 |
| 4000 | 2 | 5.3E-05 | 6.0E-05 | 4.9E-05 |
| 5000 | 2 | 5.5E-05 | 7.3E-05 | 5.1E-05 |
| 10000 | 2 | 7.3E-05 | 1.5E-04 | 4.7E-05 |
| 20000 | 2 | 1.1E-04 | 2.9E-04 | 4.9E-05 |
| 40000 | 3 | 1.4E-04 | 5.8E-04 | 6.5E-05 |
| 80000 | 3 | 1.7E-04 | 1.2E-03 | 6.4E-05 |
| 160000 | 3 | 2.3E-04 | 2.3E-03 | 6.5E-05 |
| 320000 | 4 | 2.8E-04 | 4.7E-03 | 6.9E-05 |

TABLE 2

*Timings for direct and multipole updates for $k = 1$ and $p = 6$. The values $t_{mple}$ and $t_{dir}$ are time per shift of one particle, in seconds, for the multipole updating method and the direct method, respectively. The value $\epsilon_{max}$ is the maximum relative error occurring during the 1000 shifts.*

| $N$ | $L$ | $t_{\text{mple}}$ | $t_{\text{dir}}$ | $\epsilon_{\text{max}}$ |
|---|---|---|---|---|
| 5000 | 2 | 7.5E-05 | 7.4E-05 | 3.0E-08 |
| 10000 | 2 | 9.6E-05 | 1.5E-04 | 1.3E-08 |
| 20000 | 2 | 1.3E-04 | 2.9E-04 | 2.5E-08 |
| 40000 | 2 | 2.3E-04 | 5.8E-04 | 3.1E-08 |
| 80000 | 3 | 2.4E-04 | 1.2E-03 | 2.4E-08 |
| 160000 | 3 | 3.0E-04 | 2.3E-03 | 3.2E-08 |
| 320000 | 4 | 4.3E-04 | 4.7E-03 | 2.7E-08 |

of particles $N$ approximately doubles the direct shift time while the multipole shift time grows sub-linearly. The value of $N$ at which multipole updating becomes faster than direct updating can be found from the tables. As expected, this crossover value increases with $p$, ranging from $\sim 3,500$ for $p = 3$ to $\sim 20,000$ for $p = 15$.

In our experiments, the time to shift a single particle out of a group of $k$ does not appear to decrease meaningfully with increasing $k$, which might be expected as group-wise shift reduces the per-particle cost of updating. See Table 6 for timings with $N = 80,000$ and several values of $p$ and $k$. Thus the crossover values for $N$ at which the multipole update becomes advantageous over the direct update are unchanged for increasing $k$.

Tables 7 to 11 compare individual and group-wise updating of the multipole coefficients for $p = 3, 6, 9, 12, 15$. For $p = 3$ and $p = 6$ we do not observe a pattern as to which method is faster. The time difference between the two methods is typically less than two percent, and only occasionally 5-8%, so the impact of the choice is small for $p = 3$ and $p = 6$. For $p = 9, 12, 15$, individual updating is faster for group sizes $k = 1, 2, 5$ while group updating is generally faster for group sizes $k = 20, 50, 100$ and large $N$.

To demonstrate that errors do not accumulate over the course of repeated shifts, we report in Table 12 the relative error after $10^6$ shifts with $p = 3, 6, 9, 12, 15$ and $k = 1, 2, 5, 20, 50, 100$. We observe that errors in the multipole updating method do not accumulate even after many shifts.

TABLE 3

*Timings for direct and multipole updates for $k = 1$ and $p = 9$. The values $t_{mple}$ and $t_{dir}$ are time per shift of one particle, in seconds, for the multipole updating method and the direct method, respectively. The value $\epsilon_{max}$ is the maximum relative error occurring during the 1000 shifts.*

| $N$ | $L$ | $t_{\text{mple}}$ | $t_{\text{dir}}$ | $\epsilon_{\text{max}}$ |
|---|---|---|---|---|
| 5000 | 2 | 1.2E-04 | 7.4E-05 | 1.6E-10 |
| 10000 | 2 | 1.4E-04 | 1.5E-04 | 1.2E-10 |
| 20000 | 2 | 1.7E-04 | 2.9E-04 | 1.2E-09 |
| 40000 | 2 | 3.4E-04 | 5.9E-04 | 8.5E-10 |
| 80000 | 3 | 3.6E-04 | 1.2E-03 | 6.8E-10 |
| 160000 | 3 | 4.4E-04 | 2.3E-03 | 5.0E-10 |
| 320000 | 4 | 6.9E-04 | 4.7E-03 | 5.9E-10 |

TABLE 4

*Timings for direct and multipole updates for $k = 1$ and $p = 12$. The values $t_{mple}$ and $t_{dir}$ are time per shift of one particle, in seconds, for the multipole updating method and the direct method, respectively. The value $\epsilon_{max}$ is the maximum relative error occurring during the 1000 shifts.*

| $N$ | $L$ | $t_{\text{mple}}$ | $t_{\text{dir}}$ | $\epsilon_{\text{max}}$ |
|---|---|---|---|---|
| 10000 | 2 | 1.9E-04 | 1.5E-04 | 1.2E-10 |
| 20000 | 2 | 2.3E-04 | 2.9E-04 | 1.1E-11 |
| 40000 | 2 | 5.1E-04 | 5.9E-04 | 2.6E-11 |
| 80000 | 3 | 5.5E-04 | 1.2E-03 | 5.3E-12 |
| 160000 | 3 | 6.2E-04 | 2.3E-03 | 5.2E-12 |
| 320000 | 4 | 1.0E-03 | 4.7E-03 | 4.8E-12 |

TABLE 5

*Timings for direct and multipole updates for $k = 1$ and $p = 15$. The values $t_{mple}$ and $t_{dir}$ are time per shift of one particle, in seconds, for the multipole updating method and the direct method, respectively. The value $\epsilon_{max}$ is the maximum relative error occurring during the 1000 shifts.*

| $N$ | $L$ | $t_{\text{mple}}$ | $t_{\text{dir}}$ | $\epsilon_{\text{max}}$ |
|---|---|---|---|---|
| 10000 | 2 | 2.7E-04 | 1.4E-04 | 7.3E-12 |
| 20000 | 2 | 3.0E-04 | 2.9E-04 | 2.0E-12 |
| 40000 | 2 | 7.3E-04 | 5.9E-04 | 2.7E-12 |
| 80000 | 3 | 7.6E-04 | 1.2E-03 | 4.1E-12 |
| 160000 | 3 | 8.3E-04 | 2.3E-03 | 8.8E-13 |
| 320000 | 4 | 1.5E-03 | 4.7E-03 | 2.8E-12 |

TABLE 6

*Time in seconds to shift one particle for a range of values for k, the group size, and p, the multipole expansion size, with $N = 80,000$; rotation-based $T_{MM}$ is used for group-wise updating for $k > 1$ while individual (multipole re-expansion) updating is used for $k = 1$. There is no meaningful difference in time as k increases.*

| $p$ | $k = 1$ | 2 | 5 | 20 | 50 | 100 |
|---|---|---|---|---|---|---|
| 3 | 1.67E-04 | 1.58E-04 | 1.54E-04 | 1.52E-04 | 1.52E-04 | 1.54E-04 |
| 6 | 2.33E-04 | 2.45E-04 | 2.24E-04 | 2.21E-04 | 2.24E-04 | 2.22E-04 |
| 9 | 3.80E-04 | 3.67E-04 | 3.54E-04 | 3.56E-04 | 3.47E-04 | 3.52E-04 |
| 12 | 5.32E-04 | 5.68E-04 | 5.43E-04 | 5.25E-04 | 5.13E-04 | 5.16E-04 |
| 15 | 7.59E-04 | 7.75E-04 | 7.52E-04 | 7.55E-04 | 7.39E-04 | 7.44E-04 |

TABLE 7

*Comparison of timings for individual and group-wise shift for $p = 3$. Times are time per shift of one particle, in seconds; the median time of three runs of 1000 shifts is reported. Choice of $s$ to determine the number of levels in the octree is as in the text.*

| N | k=1 | | k=2 | | k=5 | |
|---|---|---|---|---|---|---|
| | indiv | group | indiv | group | indiv | group |
| 4000 | 5.13E-05 | 5.16E-05 | 4.92E-05 | 5.00E-05 | 4.83E-05 | 4.87E-05 |
| 5000 | 5.57E-05 | 5.45E-05 | 5.31E-05 | 5.33E-05 | 5.24E-05 | 5.20E-05 |
| 10000 | 7.35E-05 | 7.35E-05 | 7.20E-05 | 7.16E-05 | 7.07E-05 | 7.14E-05 |
| 20000 | 1.09E-04 | 1.11E-04 | 1.06E-04 | 1.08E-04 | 1.07E-04 | 1.06E-04 |
| 40000 | 1.39E-04 | 1.40E-04 | 1.34E-04 | 1.34E-04 | 1.30E-04 | 1.29E-04 |
| 80000 | 1.66E-04 | 1.65E-04 | 1.57E-04 | 1.61E-04 | 1.56E-04 | 1.57E-04 |
| 160000 | 2.29E-04 | 2.28E-04 | 2.05E-04 | 2.10E-04 | 2.03E-04 | 2.06E-04 |
| 320000 | 2.73E-04 | 2.76E-04 | 2.42E-04 | 2.48E-04 | 2.38E-04 | 2.37E-04 |
| N | k=20 | | k=50 | | k=100 | |
| | indiv | group | indiv | group | indiv | group |
| 4000 | 4.91E-05 | 4.87E-05 | 5.01E-05 | 4.97E-05 | 5.14E-05 | 5.92E-05 |
| 5000 | 5.31E-05 | 5.27E-05 | 5.44E-05 | 5.19E-05 | 5.52E-05 | 6.20E-05 |
| 10000 | 6.93E-05 | 6.95E-05 | 7.14E-05 | 7.61E-05 | 6.99E-05 | 7.85E-05 |
| 20000 | 1.06E-04 | 1.09E-04 | 1.06E-04 | 1.07E-04 | 1.09E-04 | 1.20E-04 |
| 40000 | 1.30E-04 | 1.30E-04 | 1.35E-04 | 1.32E-04 | 1.32E-04 | 1.29E-04 |
| 80000 | 1.55E-04 | 1.55E-04 | 1.59E-04 | 1.58E-04 | 1.62E-04 | 1.54E-04 |
| 160000 | 2.04E-04 | 2.05E-04 | 2.07E-04 | 2.07E-04 | 2.02E-04 | 2.03E-04 |
| 320000 | 2.34E-04 | 2.36E-04 | 2.56E-04 | 2.33E-04 | 2.38E-04 | 2.34E-04 |

**5. Conclusion.** In this paper we have introduced a new method for updating the Coulomb potential energy functional of a collection of point charges when a subset of those charges undergoes a shift in position. Compared to the direct method, our updating multipole method improves the asymptotic complexity as well as runtime for problems of practical size. Errors in the new method are bounded and do not accumulate as the number of shifts grows. This method may be extended to include periodic boundary conditions, which were not considered here. A natural application of the updating method is as a component of a design optimization. In such cases it would be useful to simultaneously compute the gradient of the potential energy functional with respect to the $k$ positions that are changing. The appendix outlines how to include this gradient calculation in our updating framework. Non-uniform particle distributions would benefit from an adaptive octree; this may be added to the method through modification of the data structures for the octree and associated bookkeeping. We expect the method presented here to find application in diverse fields of computational science.

**Acknowledgements.** We thank the anonymous referees for their helpful additions to the text and references, as well as G. Beylkin, K. Coakley, Z. Gimbutas, and L. Greengard for their comments on a draft of this paper. TH thanks the Applied and Computational Mathematics Division of the Information Technology Laboratory at the National Institute for Standards and Technology in Boulder, CO, which hosted

TABLE 8

*Comparison of timings for individual and group-wise shift for $p = 6$. Times are time per shift of one particle, in seconds; the median time of three runs of 1000 shifts is reported. Choice of $s$ to determine the number of levels in the octree is as in the text.*

| N | k=1 | | k=2 | | k=5 | |
|---|---|---|---|---|---|---|
| | indiv | group | indiv | group | indiv | group |
| 10000 | 9.43E-05 | 9.68E-05 | 9.38E-05 | 9.39E-05 | 9.39E-05 | 9.30E-05 |
| 20000 | 1.31E-04 | 1.33E-04 | 1.29E-04 | 1.31E-04 | 1.29E-04 | 1.29E-04 |
| 40000 | 2.04E-04 | 2.16E-04 | 2.04E-04 | 2.13E-04 | 2.11E-04 | 1.99E-04 |
| 80000 | 2.37E-04 | 2.41E-04 | 2.52E-04 | 2.33E-04 | 2.48E-04 | 2.29E-04 |
| 160000 | 2.92E-04 | 3.08E-04 | 2.96E-04 | 2.83E-04 | 2.76E-04 | 2.77E-04 |
| 320000 | 4.23E-04 | 4.32E-04 | 3.82E-04 | 4.21E-04 | 3.96E-04 | 3.80E-04 |
| N | k=20 | | k=50 | | k=100 | |
| | indiv | group | indiv | group | indiv | group |
| 10000 | 9.61E-05 | 9.45E-05 | 9.25E-05 | 9.28E-05 | 9.39E-05 | 9.34E-05 |
| 20000 | 1.31E-04 | 1.30E-04 | 1.31E-04 | 1.29E-04 | 1.31E-04 | 1.32E-04 |
| 40000 | 2.07E-04 | 2.00E-04 | 2.08E-04 | 2.03E-04 | 2.01E-04 | 2.00E-04 |
| 80000 | 2.40E-04 | 2.26E-04 | 2.32E-04 | 2.26E-04 | 2.24E-04 | 2.23E-04 |
| 160000 | 2.90E-04 | 2.84E-04 | 2.95E-04 | 2.79E-04 | 2.75E-04 | 2.76E-04 |
| 320000 | 3.87E-04 | 3.79E-04 | 4.09E-04 | 3.76E-04 | 3.65E-04 | 3.63E-04 |

TABLE 9

*Comparison of timings for individual and group-wise shift for $p = 9$. Times are time per shift of one particle, in seconds; the median time of three runs of 1000 shifts is reported. Choice of $s$ to determine the number of levels in the octree is as in the text.*

| N | k=1 | | k=2 | | k=5 | |
|---|---|---|---|---|---|---|
| | indiv | group | indiv | group | indiv | group |
| 10000 | 1.36E-04 | 1.37E-04 | 1.33E-04 | 1.36E-04 | 1.33E-04 | 1.33E-04 |
| 20000 | 1.78E-04 | 1.74E-04 | 1.71E-04 | 1.78E-04 | 1.70E-04 | 1.72E-04 |
| 40000 | 3.42E-04 | 3.62E-04 | 3.32E-04 | 3.44E-04 | 3.28E-04 | 3.32E-04 |
| 80000 | 3.61E-04 | 3.81E-04 | 3.61E-04 | 3.74E-04 | 3.51E-04 | 3.61E-04 |
| 160000 | 4.21E-04 | 4.50E-04 | 4.02E-04 | 4.20E-04 | 4.08E-04 | 4.08E-04 |
| 320000 | 6.56E-04 | 7.21E-04 | 6.29E-04 | 6.60E-04 | 6.29E-04 | 6.33E-04 |
| N | k=20 | | k=50 | | k=100 | |
| | indiv | group | indiv | group | indiv | group |
| 10000 | 1.33E-04 | 1.34E-04 | 1.37E-04 | 1.35E-04 | 1.35E-04 | 1.35E-04 |
| 20000 | 1.68E-04 | 1.69E-04 | 1.71E-04 | 1.74E-04 | 1.73E-04 | 1.75E-04 |
| 40000 | 3.29E-04 | 3.29E-04 | 3.38E-04 | 3.28E-04 | 3.29E-04 | 3.32E-04 |
| 80000 | 3.53E-04 | 3.52E-04 | 3.56E-04 | 3.50E-04 | 3.53E-04 | 3.53E-04 |
| 160000 | 4.24E-04 | 3.99E-04 | 4.04E-04 | 4.04E-04 | 4.02E-04 | 3.97E-04 |
| 320000 | 6.16E-04 | 6.12E-04 | 6.32E-04 | 6.19E-04 | 6.14E-04 | 6.12E-04 |

him as a visitor during a portion of the work.

**Appendix A. Gradient of potential energy functional.** The potential energy functional $V(\mathbf{x}, \mathbf{q})$ defined in (2) and given by

$$V(\mathbf{x}, \mathbf{q}) = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \frac{q_i q_j}{|x_i - x_j|} \tag{31}$$

TABLE 10

*Comparison of timings for individual and group-wise shift for $p = 12$. Times are time per shift of one particle, in seconds; the median time of three runs of 1000 shifts is reported. Choice of $s$ to determine the number of levels in the octree is as in the text.*

| N | k=1 | | k=2 | | k=5 | |
|---|---|---|---|---|---|---|
| | indiv | group | indiv | group | indiv | group |
| 20000 | 2.29E-04 | 2.30E-04 | 2.34E-04 | 2.27E-04 | 2.26E-04 | 2.32E-04 |
| 40000 | 5.03E-04 | 5.66E-04 | 5.08E-04 | 5.32E-04 | 5.03E-04 | 5.20E-04 |
| 80000 | 5.27E-04 | 5.88E-04 | 5.41E-04 | 5.74E-04 | 5.41E-04 | 5.47E-04 |
| 160000 | 6.02E-04 | 6.53E-04 | 6.05E-04 | 6.32E-04 | 5.76E-04 | 5.97E-04 |
| 320000 | 1.00E-03 | 1.12E-03 | 9.96E-04 | 1.02E-03 | 9.82E-04 | 9.95E-04 |
| N | k=20 | | k=50 | | k=100 | |
| | indiv | group | indiv | group | indiv | group |
| 20000 | 2.25E-04 | 2.30E-04 | 2.27E-04 | 2.23E-04 | 2.27E-04 | 2.25E-04 |
| 40000 | 5.01E-04 | 5.22E-04 | 5.11E-04 | 4.99E-04 | 4.96E-04 | 5.00E-04 |
| 80000 | 5.38E-04 | 5.36E-04 | 5.35E-04 | 5.33E-04 | 5.36E-04 | 5.43E-04 |
| 160000 | 5.87E-04 | 5.80E-04 | 5.90E-04 | 5.79E-04 | 5.85E-04 | 5.84E-04 |
| 320000 | 9.74E-04 | 9.46E-04 | 9.46E-04 | 9.33E-04 | 9.59E-04 | 9.46E-04 |

TABLE 11

*Comparison of timings for individual and group-wise shift for $p = 15$. Times are time per shift of one particle, in seconds; the median time of three runs of 1000 shifts is reported. Choice of $s$ to determine the number of levels in the octree is as in the text.*

| N | k=1 | | k=2 | | k=5 | |
|---|---|---|---|---|---|---|
| | indiv | group | indiv | group | indiv | group |
| 20000 | 2.98E-04 | 3.01E-04 | 2.99E-04 | 2.95E-04 | 2.93E-04 | 2.98E-04 |
| 40000 | 7.11E-04 | 8.10E-04 | 7.30E-04 | 7.71E-04 | 7.23E-04 | 7.46E-04 |
| 80000 | 7.63E-04 | 8.46E-04 | 7.48E-04 | 7.91E-04 | 7.40E-04 | 7.68E-04 |
| 160000 | 8.30E-04 | 9.21E-04 | 8.08E-04 | 8.48E-04 | 7.97E-04 | 8.20E-04 |
| 320000 | 1.51E-03 | 1.61E-03 | 1.41E-03 | 1.49E-03 | 1.39E-03 | 1.41E-03 |
| N | k=20 | | k=50 | | k=100 | |
| | indiv | group | indiv | group | indiv | group |
| 20000 | 2.97E-04 | 2.97E-04 | 2.96E-04 | 2.98E-04 | 3.00E-04 | 2.97E-04 |
| 40000 | 7.27E-04 | 7.27E-04 | 7.28E-04 | 7.21E-04 | 7.30E-04 | 7.19E-04 |
| 80000 | 7.60E-04 | 7.52E-04 | 7.51E-04 | 7.55E-04 | 7.57E-04 | 7.53E-04 |
| 160000 | 8.00E-04 | 7.93E-04 | 8.01E-04 | 8.00E-04 | 7.79E-04 | 8.01E-04 |
| 320000 | 1.40E-03 | 1.38E-03 | 1.39E-03 | 1.39E-03 | 1.39E-03 | 1.38E-03 |

can be differentiated with respect to a particle position $x_i \in \mathbb{R}^3$ more simply than updated for a change of $x_i$ to $\tilde{x}_i$, since the former involves no movement of particles between boxes nor translation of multipole expansions. Instead it requires the gradient

$$\nabla_i \frac{1}{|x_i - x_j|} = \frac{x_j - x_i}{|x_i - x_j|^3} \tag{32}$$

of each term expressing a local interaction with particle $i$, and the gradient of each of $\mathcal{O}(\log N)$ multipole expansions

$$\Phi(r, \theta, \phi) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} M_n^m \frac{Y_n^m(\theta, \phi)}{r^{n+1}} \tag{33}$$

TABLE 12
*Relative errors at the end of 1,000,000 shifts with $N = 40,000$ and $L = 3$, for individual ("ind") and group ("grp") shifts.*

| | | $p = 3$ | 6 | 9 | 12 | 15 |
|---|---|---|---|---|---|---|
| $k = 1$ | ind | 6.6E-05 | 3.1E-08 | 7.8E-10 | 3.3E-11 | 2.4E-12 |
| | grp | 6.5E-05 | 3.0E-08 | 9.2E-10 | 1.5E-11 | 3.6E-12 |
| $k = 2$ | ind | 4.0E-05 | 1.2E-08 | 2.4E-10 | 3.0E-11 | 3.7E-12 |
| | grp | 3.9E-05 | 1.8E-08 | 5.6E-10 | 1.4E-11 | 3.1E-12 |
| $k = 5$ | ind | 1.8E-05 | 9.9E-10 | 2.7E-10 | 3.5E-11 | 1.2E-12 |
| | grp | 1.8E-05 | 7.8E-09 | 2.9E-10 | 8.1E-12 | 1.7E-12 |
| $k = 20$ | ind | 4.6E-06 | 1.9E-08 | 9.2E-11 | 2.7E-11 | 9.5E-12 |
| | grp | 4.8E-06 | 1.2E-08 | 1.7E-09 | 7.4E-11 | 3.0E-12 |
| $k = 50$ | ind | 2.2E-06 | 2.9E-09 | 1.4E-09 | 4.1E-11 | 9.0E-12 |
| | grp | 1.3E-06 | 1.5E-08 | 1.9E-09 | 7.2E-11 | 7.3E-12 |
| $k = 100$ | ind | 9.9E-07 | 4.8E-09 | 2.0E-10 | 3.1E-11 | 3.1E-12 |
| | grp | 6.4E-07 | 5.6E-09 | 3.7E-09 | 3.3E-10 | 2.8E-11 |

with respect to the point $x_i$ with spherical coordinates $(r, \theta, \phi)$ in the coordinate system of the multipole expansion.

Derivatives of multipoles are most simply expressed in terms of the operators $\partial_+$, $\partial_-$, and $\partial_z$, defined by

$$\partial_\pm = \partial_x \pm i\partial_y, \quad \partial_z = \frac{\partial}{\partial z}, \quad \partial_x = \frac{\partial}{\partial x}, \quad \partial_y = \frac{\partial}{\partial y}. \tag{34}$$

With these definitions,

$$\partial_+\partial_-(\psi) = -\partial_z^2(\psi) \tag{35}$$

for any harmonic function $\psi$ and

$$\partial_+^m \partial_z^{n-m}\left(\frac{1}{r}\right) = \frac{Y_n^m(\theta,\phi)}{A_n^m \cdot r^{n+1}} \tag{36}$$

$$\partial_-^m \partial_z^{n-m}\left(\frac{1}{r}\right) = \frac{Y_n^{-m}(\theta,\phi)}{A_n^m \cdot r^{n+1}} \tag{37}$$

for $0 \le m \le n$, where $A_n^m$ is defined by (21) and $Y_n^m$ is given by (13) (after [14]). From these expressions we obtain

$$\partial_x \frac{Y_n^m(\theta,\phi)}{r^{n+1}} = \frac{B_{n+m}}{2}\frac{Y_{n+1}^{m+1}(\theta,\phi)}{r^{n+2}} - \frac{B_{n-m}}{2}\frac{Y_{n+1}^{m-1}(\theta,\phi)}{r^{n+2}} \tag{38}$$

$$\partial_y \frac{Y_n^m(\theta,\phi)}{r^{n+1}} = \frac{B_{n+m}}{2i}\frac{Y_{n+1}^{m+1}(\theta,\phi)}{r^{n+2}} + \frac{B_{n-m}}{2i}\frac{Y_{n+1}^{m-1}(\theta,\phi)}{r^{n+2}} \tag{39}$$

$$\partial_z \frac{Y_n^m(\theta,\phi)}{r^{n+1}} = -\sqrt{(n+1)^2 - m^2}\,\frac{Y_{n+1}^{m+1}(\theta,\phi)}{r^{n+2}}, \tag{40}$$

where $B_n = \sqrt{(n+1)(n+2)}$. Expressions (38)–(40) can be used to construct the multipole expansion of $\nabla_i \Phi$, and therefore the gradient of $V(\mathbf{x}, \mathbf{q})$.

## REFERENCES

[1] D. Baker and A. Sali, *Protein structure prediction and structural genomics*, Science, 294 (2001), pp. 93–96, doi:10.1126/science.1065659.

[2] J. Barnes and P. Hut, *A hierarchical $O(N \log N)$ force-calculation algorithm*, Nature, 323 (1986), pp. 446–449, doi:10.1038/324446a0.

[3] R. Beatson and L. Greengard, *A short course on fast multipole methods*, in Wavelets, Multilevel Methods and Elliptic PDEs, Oxford University Press, 1997, pp. 1–37.

[4] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM Journal on Scientific and Statistical Computing, 9 (1988), pp. 669–686, doi:10.1137/0909044.

[5] A. Chandramowlishwarany, K. Madduri, and R. Vuduc, *Diagnosis, tuning, and redesign for multicore performance: A case study of the fast multipole method*, in 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, Nov 2010, pp. 1–12, doi:10.1109/SC.2010.19.

[6] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM Journal on Scientific Computing, 26 (2005), pp. 1389–1404, doi:10.1137/030602678.

[7] H. Cheng, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm in three dimensions*, Journal of Computational Physics, 155 (1999), pp. 468 – 498, doi:10.1006/jcph.1999.6355.

[8] G. Cisneros, M. Karttunen, P. Ren, and C. Sagui, *Classical electrostatics for biomolecular simulations*, Chemical Reviews, 114 (2014), pp. 779–814, doi:10.1021/cr300461d.

[9] W. Dehnen, *A fast multipole method for stellar dynamics*, Computational Astrophysics and Cosmology, 1 (2014), pp. 1–23, doi:10.1186/s40668-014-0001-7.

[10] V. Diky, R. D. Chirico, C. D. Muzny, A. F. Kazakov, K. Kroenlein, J. W. Magee, I. Abdulagatov, J. W. Kang, R. Gani, and M. Frenkel, *ThermoData Engine (TDE): Software implementation of the dynamic data evaluation concept. 8. properties of material streams and solvent design*, Journal of Chemical Information and Modeling, 53 (2013), pp. 249–266, doi:10.1021/ci300470t.

[11] Z. Gan and Z. Xu, *Efficient implementation of the Barnes-Hut octree algorithm for Monte Carlo simulations of charged systems*, Science China Mathematics, 57 (2014), pp. 1331–1340, doi:10.1007/s11425-014-4783-5.

[12] A. Gillman and P. G. Martinsson, *A direct solver with $O(N)$ complexity for variable coefficient elliptic pdes discretized via a high-order composite spectral collocation method*, SIAM Journal on Scientific Computing, 36 (2014), pp. A2023–A2046, doi:10.1137/130918988.

[13] Z. Gimbutas and L. Greegard, *fmmlib3d*, 2012, https://github.com/zgimbutas/fmmlib3d (accessed 2015/01/29). Version 1.2.

[14] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.

[15] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, Journal of Computational Physics, 135 (1987), pp. 280–292, doi:10.1016/0021-9991(87)90140-9.

[16] L. Greengard and V. Rokhlin, *The rapid evaluation of potential fields in three dimensions*, in Vortex Methods: Proceedings of the U.C.L.A. Workshop held in Los Angeles, May 20–22, 1987, C. Anderson and C. Greengard, eds., Springer, Berlin, 1988, pp. 121–141, doi:10.1007/BFb0089775.

[17] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numerica, 6 (1997), pp. 229–269, doi:10.1017/S0962492900002725.

[18] A. Hospital, J. R. Goñi, M. Orozco, and J. L. Gelpí, *Molecular dynamics simulations: advances and applications*, Advances and Applications in Bioinformatics and Chemistry, 8 (2015), pp. 37–47, doi:10.2147/AABC.S70333.

[19] M. Karplus and J. Kuriyan, *Molecular dynamics and protein function*, Proceedings of the National Academy of Sciences, 102 (2005), pp. 6679–6685, doi:10.1073/pnas.0408930102.

[20] M. Karplus and J. A. McCammon, *Molecular dynamics simulations of biomolecules*, Nature Structural & Molecular Biology, 9 (2002), pp. 646–652, doi:10.1038/nsb0902-646.

[21] J. Kurzak and B. M. Pettitt, *Fast multipole methods for particle dynamics*, Molecular simulation, 32 (2006), pp. 775–790, doi:10.1080/08927020600991161.

[22] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin, and G. Biros, *A massively parallel adaptive fast multipole method on heterogeneous architectures*, Communications of the ACM, 55 (2012), pp. 101–109, doi:10.1145/2160718.2160740.

[23] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proceedings of the National Academy

of Sciences, 104 (2007), pp. 20167–20172, doi:10.1073/pnas.0709640104.

[24] D. Malhotra, A. Gholami, and G. Biros, *A volume integral equation Stokes solver for problems with variable coefficients*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '14, Piscataway, NJ, USA, 2014, IEEE Press, pp. 92–102, doi:10.1109/SC.2014.13.

[25] W. B. March, B. Xiao, S. Tharakan, C. D. Yu, and G. Biros, *A kernel-independent fmm in general dimensions*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, New York, NY, USA, 2015, ACM, pp. 24:1–24:12, doi:10.1145/2807591.2807647.

[26] P.-G. Martinsson, *Fast multipole methods*, in Encyclopedia of Applied and Computational Mathematics, B. Engquist, ed., Springer, Berlin, 2015, pp. 498–508, doi:10.1007/978-3-540-70529-1_448.

[27] V. Minden, A. Damle, K. L. Ho, and L. Ying, *A technique for updating hierarchical skeletonization-based factorizations of integral operators*, Multiscale Modeling and Simulation, 14 (2016), pp. 42–64, doi:10.1137/15M1024500.

[28] D. D. L. Minh and D. L. P. Minh, *Understanding the Hastings algorithm*, Communications in Statistics - Simulation and Computation, 44 (2015), pp. 332–349, doi:10.1080/03610918.2013.777455.

[29] K. Nabors, F. Korsmeyer, F. Leighton, and J. White, *Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory*, SIAM Journal on Scientific Computing, 15 (1994), pp. 713–735, doi:10.1137/0915046.

[30] M. Patra, M. Karttunen, M. T. Hyvönen, E. Falck, P. Lindqvist, and I. Vattulainen, *Molecular dynamics simulations of lipid bilayers: Major artifacts due to truncating electrostatic interactions*, Biophysical Journal, 84 (2003), pp. 3636–3645, doi:10.1016/S0006-3495(03)75094-2.

[31] E. Paulechka, K. Kroenlein, A. Kazakov, and M. Frenkel, *A systematic approach for development of an OPLS-like force field and its application to hydrofluorocarbons*, Journal of Physical Chemistry B, 116 (2012), pp. 14389–14397, doi:10.1021/jp309119h.

[32] V. Rokhlin, *Rapid solution of integral-equations of classical potential theory*, Journal of Computational Physics, 60 (1985), pp. 187–207, doi:10.1016/0021-9991(85)90002-6.

[33] C. Sagui and T. A. Darden, *Molecular dynamics simulations of biomolecules: Long-range electrostatic effects*, Annual Review of Biophysics and Biomolecular Structure, 28 (1999), pp. 155–179, doi:10.1146/annurev.biophys.28.1.155. PMID: 10410799.

[34] T. Schlick, R. Collepardo-Guevara, L. A. Halvorsen, S. Jung, and X. Xiao, *Biomolecular modeling and simulation: a field coming of age*, Quarterly Reviews of Biophysics, 44 (2011), pp. 191–228, doi:10.1017/S0033583510000284.

[35] M. Schwörer, K. Lorenzen, G. Mathias, and P. Tavan, *Utilizing fast multipole expansions for efficient and accurate quantum-classical molecular dynamics simulations*, Journal of Chemical Physics, 142 (2015), 104108, doi:10.1063/1.4914329.

[36] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, Journal of Computational Physics, 196 (2004), pp. 591–626, doi:10.1016/j.jcp.2003.11.021.

[37] R. Yokota, L. Barba, T. Narumi, and K. Yasuoka, *Petascale turbulence simulation using a highly parallel fast multipole method on GPUs*, Computer Physics Communications, 184 (2013), pp. 445 – 455, doi:10.1016/j.cpc.2012.09.011.