A kernel-independent fast multipole algorithm Technical Report TR2003-839 [1]

Lexing Ying∗, George Biros∗, and Denis Zorin∗
∗ *Courant Institute of Mathematical Sciences, New York University, New York 10012*
Email: {lexing,biros,dzorin}@cs.nyu.edu

Version: April 2003

We present a new fast multipole method for particle simulations. The main feature of our algorithm is that is kernel independent, in the sense that no analytic expansions are used to represent the far field. Instead we use equivalent densities, which we compute by solving small Dirichlet-type boundary value problems. The translations from the sources to the induced potentials are accelerated by singular value decomposition in 2D and fast Fourier transforms in 3D. We have tested the new method on the single and double layer operators for the Laplacian, the modified Laplacian, the Stokes, the modified Stokes, the Navier, and the modified Navier operators in two and three dimensions. Our numerical results indicate that our method compares very well with the best known implementations of the analytic FMM method for both the Laplacian and modified Laplacian kernels. Its advantage is the (relative) simplicity of the implementation and its immediate extension to more general kernels.

Key Words: Fast multipole methods, fast solvers, integral equations, single-layer potential, double-layer potential, particle methods, N-body problems

1. INTRODUCTION

Many methods in computational physics (e.g., vortex methods, molecular dynamics) are based on the evolution of particle systems with pairwise interactions corresponding to potentials related to the fundamental solution of elliptic partial differential equations (PDEs). The most important among these kernels is the single-layer Laplacian. Other kernels include the the kernels of Stokes and the Navier operators their modified versions arising in unsteady problems, and their derivatives (double-layer and hypersingular kernels).

Particle formulations result in dense linear algebraic systems because all pairwise interactions have to be computed. This is a significant bottleneck since for N particles this results in a $\mathcal{O}(N^2)$ computation. In order to make large scale problems tractable it is essential to efficiently compute these interactions. A number of algorithms were proposed for this purpose. The fast multipole method (FMM) has been one of the most successful, especially for nonuniform particle distributions.

In this paper we present a new kernel-independent FMM-like algorithm. Our algorithm has the structure of the adaptive FMM [11], but requires kernel evaluations only, and does not sacrifice the efficiency of the original algorithm. The crucial element of our approach is to replace the analytic expansions and translations with *equivalent density* representations. These representations are computed by solving small integral equation exterior and

[1] This work is supported by the National Science Foundation's Knowledge and Distributed Intelligence (KDI) program through grant DMS-9980069.

interior problems on discs (2D), spheres or cubes (3D). We demonstrate the efficiency of our method in both 2D and 3D for many kernels: the single and double layer potentials of the Laplacian, the modified Laplacian, the Navier, the Stokes, and their modified variants. Our method has $\mathcal{O}(N)$ asymptotic complexity, and, like analytic FMM, works well for nonuniform particle distributions.

Synopsis of the new method. The basic structure of our method follows [13], the original fast multipole method, which we briefly review in Section 2. FMM consists of the following steps:

1. generation of a hierarchical tree partitioning of the computational domain;
2. accumulation of the multipole expansions for the far field by a postorder traversal of the tree;
3. translation of the multipole moments to the local expansions;
4. construction of local expansions by a preorder traversal of the tree;
5. evaluation of the far field action on the particles using local expansions;
6. evaluation of the near field interactions.

The same steps are used in our algorithm. However the multipole expansion construction is replaced by solving local exterior inverse problems. To represent the potential generated by particles inside a box, we use a continuous distribution of an equivalent density on a surface enclosing the box, building on the idea introduced in [1]. To find this equivalent density on the surface, we match its potential to the potential of the original sources at a number of points in the far field. The translations are done by direct evaluation on the far field, sparsified with SVD or FFT. During the preorder traversal of the tree, we evaluate the far field on a surface enclosing a target box, and solve an interior Dirichlet-type integral equation to compute an equivalent density. Then we use this density to represent the potential inside a target box.

Our method does not require implementation of analytic expansions for the kernel, it only requires their existence, and uses exclusively kernel evaluations. Like FMM, our algorithm is recursive and has an $\mathcal{O}(N)$ complexity. Additional properties like scale invariance and rotational symmetries of kernels can be used to further accelerate the translation step, as in the case of the standard FMM.

Related work. There are four basic classes of fast summation algorithms: (1) tree codes like Barnes-Hut [2], (2) fast multipole methods and (3) regular grid fast convolution methods like FFT [2]. Our algorithm belongs to the second category. The description of the original fast multipole algorithm can be found in [13], and [22]. Although the method is highly successful in two dimensions, the three-dimensional version of the original method was inefficient. Efficient extensions in three dimensions were realized only recently [7]. For these reasons many researchers tried to devise algorithms which were hybrids of tree codes and FMM, in order to combine the high accuracy of FMM methods with the speed and simplicity of tree codes. In addition extension of the FMM to more general kernels like the modified Laplacian [12], the Stokes [9] , and the Navier [8, 15] operators can be

[2]This method is somewhat related to particle-particle (near field interaction) with particle-mesh algorithms. Particle-mesh methods use fast PDE solvers on regular grids (multigrid) to evaluate the far field contributions. In this paper we are not reviewing these methods since they are mostly useful for uniform particle distributions.

quite cumbersome. Here we review only the algorithms that, in our view, could be used to develop kernel independent methods.

The idea of using a set of equivalent sources was first introduced in [1]. In this paper, the far field is represented as the solution to an exterior Dirichlet problem on a ball surrounding the particles using the exact Green's function for Laplacian. The method is somewhat easier than FMM to implement, but requires knowledge of the Green's function for each kernel, which is may not be available in the general case.

In [3] instead of using the exact Green's function a number of equivalent densities are placed on a Cartesian grid in each source box; these densities are computed analytically by matching a number of multipole moments in the multipole expansion series of the original source densities. Another important feature of this method is the fact that the Cartesian grid allows the use of FFT to accelerate the multipole to local-expansions translations. However the method is not kernel-independent since for different kernels different expansions have to be constructed. The same idea is used in [19], but like in Anderson's method the densities are distributed over a ball containing the source box.

The idea of equivalent densities is also used in the precorrected FFT method, [20]. The equivalent densities are distributed over a regular grid, so that the far field convolutions can be computed with FFT instead of FMM. The term "precorrected" is related to the computation of the local interactions: the subtraction of the local influence of the equivalent densities and the addition of the near field interactions. The regular grid sources are computed by matching the field at selected checking points, usually located on a ball enclosing the original sources. In [6], a precorrected FFT method is applied to the Helmholtz kernel, but the equivalent sources are distributed along the faces of an enclosing cube, and three FFTs along the coordinate system planes are used to compute the far interaction. FFT-based methods are very efficient, often faster than FMM due to much smaller constants. For uniform distributions of particles FFT is likely to be preferable and it is kernel-independent. However, in the case of highly irregular particle distributions FMM is more efficient.

A hybrid method for kernel independent matrix-vector multiplication algorithm was proposed in [16] and [17]. Based on the fact that large blocks of the particle interaction matrix are low rank, this method uses singular value decomposition to sample and sparsify this blocks. It can be applied recursively and attains a $\mathcal{O}(N \log N)$ complexity. We have applied this method on the Stokes and Navier operators [4, 5] with very satisfactory results in both accuracy and speed. One serious shortcoming of this method is the high setup cost. For problems with static particle distributions this is not a concern, but it becomes a bottleneck for problems with time evolving particles. The SVD approach was been further explored in series of papers, [23], [24], and [10] to obtain a kernel-independent method that has smaller setup costs. However, as the authors of these papers assert, the method does not achieve the efficiency of FMM.

Another method for fast matrix multiplication is based on higher-order Taylor expansions in Cartesian coordinates. This approach is not suitable for high accuracy computations because is computationally expensive (for p_{th}-order accuracy it requires $\mathcal{O}(p^d)$ expansion terms) . However it is kernel-independent method (the higher-order expansions can be easily obtained by differentiation). For example it has been successfully used to accelerate problems with the Stokes kernel [21].

Yet another kernel-independent approach is based on wavelet decompositions, combined with a Galerkin scheme. This approach is quite promising, since it has the same complexity with FMM, and has built-in preconditioning. However, it is hard to compare directly to FMM, as different trade-offs are made: FMM is a "bottom-up" approach, and is relatively insensitive to the distribution of samples. Adaptive wavelet methods are "top-down" but require samples to be located on a surface satisfying certain assumptions.

Organization of the paper. In Section 2 we briefly review the classical FMM algorithm for the two dimensional Laplacian. In Section 3 we present the new algorithm and its implementation; in Section 4 we present a justification on the correctness of the algorithm and in Section 5 we present numerical results for several different scalar and vector kernels in two and three dimensions.

2. REVIEW OF THE FAST MULTIPOLE METHOD

Given N source densities $\{\phi_i\}$ located at N points $\{\mathbf{y}_i\}$ in $\mathbf{R}^d$ $(d = 2, 3)$, we want to compute the potential $\{u_i\}$ at N points $\{\mathbf{x}_i\}$ induced by a kernel G (single layer, double layer or other kernels of a elliptic PDE) using the following relation [3]:

$$u_i = u(\mathbf{x}_i) = \sum_{j=1}^{N} G(\mathbf{x}_i, \mathbf{y}_j)\phi(\mathbf{y}_j) = \sum_{j=1}^{N} G_{ij}\phi_j,\ i = 1, \ldots, N.$$

Direct implementation of this summation gives an $\mathcal{O}(N^2)$ algorithm. For a large class of kernels, FMM computes the same interactions in $\mathcal{O}(N)$ time. FMM is an approximate algorithm, in the sense that the summation is not computed exactly. The constant in the complexity estimate is related to the accuracy of the approximation.

We will use the single layer Laplacian kernel to describe FMM. In two dimensions we have $G(\mathbf{x}, \mathbf{y}) = -\frac{1}{2\pi}\log\rho$, with $\mathbf{r} = \mathbf{x} - \mathbf{y}$, and $\rho = |\mathbf{r}|$. In the FMM context it is convenient to use $G(\mathbf{x}, \mathbf{y}) = Re(\log(z_x - z_y))$ where z_x and z_y are complex numbers corresponding to $\mathbf{x}$ (target) and $\mathbf{y}$ (source) points on the plane. The idea of FMM is to encode the potentials of a set of source densities using the *multipole expansion* and *local expansion* at places far away from these sources. Suppose the source densities are supported in a disk centered at z_c with radius r, then for all z outside the disk with radius R $(R > r)$, we can represent the potential at z from the source densities using a set of coefficients $\{a_k, 0 \le k \le p\}$ where

$$u(z) = a_0 \log(z - z_c) + \sum_{k=1}^{p} \frac{a_k}{(z - z_c)^k} + \mathcal{O}(\frac{r^p}{R^p}) \quad \text{(Multipole expansion).} \tag{1}$$

On the other hand, if the source densities are outside the disk with radius R, the potential at a point z inside the disk with radius r can be represented using a set of coefficients $\{c_k, 0 \le k \le p\}$ where

$$u(z) = \sum_{k=0}^{p} c_k (z - z_c)^k + \mathcal{O}(\frac{r^p}{R^p}) \quad \text{(Local expansion).} \tag{2}$$

In both expansions, p is usually a small constant determining from the desired accuracy of the result. The definitions of the coefficients are given in Appendix B.

FMM employs the above representations in a recursive way. The computational domain, a box large enough to contain all source and target points, is hierarchically partitioned into a tree structure (a quadtree in 2D or an octtree in 3D). Each node of the tree corresponds to geometric box (square or cube). The tree is constructed so that the leafs contain no more than a prespecified number of points. For each box, the potential induced by the its source densities is represented using multipole expansion, while the potential induced by the sources from non-adjacent boxes is encoded in local expansion. The number

[3] We use $\mathbf{x}$ to refer to target locations and $\mathbf{y}$ to refer to source locations, but in general $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$ can be the same set of points.

of expansion terms p is chosen so that, both expansions give an error which is less than a prescribed threshold.

Not only these expansions (multipole and local) are efficient, efficient translations between these expansions are also available which make a $\mathcal{O}(N)$ complexity algorithm possible. In particular, the following types of translations are used:

M2M: The *multipole to multipole* translation transforms the multipole expansions of a box's children to its own multipole expansion.

M2L: The *multipole to local* translation transforms the multipole expansion of a box to the local expansion of another non-adjacent box.

L2L: Finally, *the local to local* translation of the local expansion of a box's parent to its own local expansion. See Appendix B for the details of these translations.

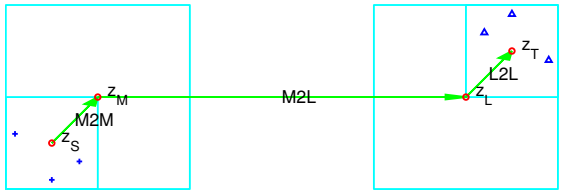

FIG. 1 The multipole expansion at z_S encodes the influence from the source densities (marked with "+") to the far field. The local expansion at z_T encodes the influence from the far field to the target points (marked with "$\triangle$"). M2M translation transforms between the multipole expansions of the boxes in adjacent levels (z_S to z_M), M2L translation transforms multipole expansion of a box to the local expansion of non-adjacent boxes (z_M to z_L), and finally L2L translation transforms between local expansions between adjacent levels (z_L to z_T).

Using the tree structure, FMM consists of two basic steps. During the first step, the *upward pass*, the tree is traversed in postorder[4] to compute the multipole expansion for each box. At the leafs, the multipole expansions are built following Equation (1) (this procedure is also called the *source to multipole* (S2M) translation). At each non-leaf node, the multipole expansion is shifted from its children using the M2M translation. In the second step, the *downwards pass*, the tree is traversed in a preorder[5] to compute the local expansion. For each box B, the local expansion is the sum of two parts: first, the local-to-local transformation collects the local expansion of B's parent (the result condenses the contribution from the sources in all the boxes which are not adjacent to B's *parent*), and secondly, the multipole-to-local transformation collects the multipole expansions of the boxes which are the children of the neighbors of B's parent but are not adjacent to B. The sum of these two parts encodes all the contribution from the sources in the boxes which are not adjacent to B *itself*. At the end, for each box, the far interaction, which is evaluated using the local expansion at this box (this step is called the *local to target* (L2T) translation), is combined with the near interaction evaluated by iterating over all the source points in the neighborhood of the target box to obtain the potential (see Figure 1).

In three dimensions instead of Laurent series, the far field is represented by spherical harmonics. There are several implementation details (mostly for the M2L transformation) that are required for efficient implementation (especially in 3D) and that we do not mention

[4]The children of a box are visited before the box itself.

[5]The children of a box are visited after the box itself.

here. Overall, however, the organization of the computation is the one we just described. For the derivation of the expansions and a detailed discussion on error bounds and implementation details see [7] and [13].

3. THE NEW ALGORITHM

Our algorithm is designed to generalize FMM to second-order constant coefficient non-oscillatory elliptic partial differential equations. Examples of such systems are given in the Appendix A, where we also list the corresponding fundamental solution kernels. Such kernels satisfy the underlying PDE everywhere but the singularity location (pole), and are smooth away from the singularity. All problems under consideration admit a unique solution for the interior/exterior Dirichlet problems. These are basic properties that we use in order to develop our FMM approximation.

Our algorithm has the same structure with original FMM method. The differences are how the densities are represented efficiently and how the M2M, M2L, and L2L transformations are computed. We first describe these representations and transformations, then state the complete algorithm and we conclude with a discussion on efficient implementation. Below we summarize the notation we use in the description of the method; most of the quantities are defined in Section 3.1.

B	a box in the computation tree
$\mathcal{N}^B$	the near range of the box B in $\mathbb{R}^d$
$\mathcal{F}^B$	the far range of the box B in $\mathbb{R}^d$
I_s^B	the set of of indices of source points or densities in B
I_t^B	the set of indices of target points or potentials in B
$\mathbf{y}^{B,u}$	the upward equivalent surface of B
$\phi^{B,u}$	the upward equivalent density of B
$\mathbf{x}^{B,u}$	the upward check surface of B
$u^{B,u}$	the upward check potential of B
$\mathbf{y}^{B,d}$	the downward equivalent surface of B
$\phi^{B,d}$	the downward equivalent density of B
$\mathbf{x}^{B,d}$	the downward check surface of B
$u^{B,d}$	the downward check potential of B
p	the degree of discretization for equivalent densities
s	the maximum number of source (or target) points allowed in a leaf box
N	the total number of source and target points
L	the depth of the computation tree
M	the total number of boxes in the computation tree

3.1. Density translations

Given a set of N points, we define the computational domain to a box large enough to contain all points. Then we construct the hierarchical tree (quadtree in 2D and octtree in 3D) so that each box contains no more than s points (s is a prescribed number). We assume that some points are labeled as source and others as targets. Given a box [6] B in the computation tree, we use I_s^B and I_t^B to denote the index sets of the source and target points in B respectively. Sometimes, we also use I_s^R and I_t^R to denote these index sets in a region

[6] A box is a square in 2D and a cube in 3D

R. The source densities $\{\phi_i, i \in I_s^B\}$ at the source locations $\{\mathbf{y}_i, i \in I_s^B\}$ are given, and we want to evaluate the potential $\{u_i, i \in I_t^B\}$ at the target locations $\{\mathbf{x}_i, i \in I_t^B\}$. If B is a box centered at $\mathbf{c}$ and has side length $2r$, then the box centered at $\mathbf{c}$ with side length $6r$ is called the *near range* of B and is denoted by $\mathcal{N}^B$. $\mathbb{R}^d \backslash \mathcal{N}^B$ is called the *far range* and is denoted by $\mathcal{F}^B$. Note that in our definition, B is a part of $\mathcal{N}^B$.

Equivalent densities. We represent the potential in $\mathcal{F}^B$ from the source densities $\{\phi_i, i \in I_s^B\}$ in B as the potential from a density distribution $\phi^{B,u}$ supported at prescribed locations $\mathbf{y}^{B,u}$ in $\mathcal{N}^B$ (Figure 2). We call $\phi^{B,u}$ the *upward equivalent density* and $\mathbf{y}^{B,u}$ the *upward equivalent surface*. Results from potential theory put two restrictions on the positions of $\mathbf{y}^{B,u}$ (see [18], chapter 6). Firstly, to guarantee the smoothness of the potential produced by $\phi^{B,u}$, its support $\mathbf{y}^{B,u}$ should not overlap with $\mathcal{F}^B$. Secondly, to guarantee that $\phi^{B,u}$ is "powerful" enough to represent the potential produced by any source distribution in B, $\mathbf{y}^{B,u}$ needs to enclose B. Therefore, in order to ensure the existence of $\phi^{B,u}$, $\mathbf{y}^{B,u}$ is required to lie between B and $\mathcal{F}^B$. We use a circle in 2D and a sphere or cube in 3D for reasons explained later.

Since the potentials induced by the source densities and the upward equivalent density both satisfy the underlying second order linear elliptic PDE, due to the uniqueness result for the exterior Dirichlet problems, the two potentials are guaranteed to be equal in $\mathcal{F}^B$ if we can match them at the boundary of $\mathcal{F}^B$ or any surface between $\mathcal{F}^B$ and $\mathbf{y}^{B,u}$. We call this surface the *upward check surface* and denoted it $\mathbf{x}^{B,u}$). We call the potential computed on this surface the *upward check potential* and denote it by $u^{B,u}$. These surfaces are also chosen to be circle in 2D, a spheres or cubes in 3D. The potential $\phi^{B,u}$ should satisfy the following equation for any $\mathbf{x} \in \mathbf{x}^{B,u}$:

$$\int_{\mathbf{y}^{B,u}} G(\mathbf{x}, \mathbf{y}) \phi^{B,u}(\mathbf{y}) \, \mathrm{d}\mathbf{y} = \sum_{i \in I_s^B} G(\mathbf{x}, \mathbf{y}_i) \phi_i. \tag{3}$$

Similarly, we represent the potential in B from the source densities in $\mathcal{F}^B$ as the potential induced by a density distribution $\phi^{B,d}$ defined at prescribed location $\mathbf{y}^{B,d}$ in $\mathcal{N}^B$ (Figure 2). We call $\phi^{B,d}$ the *downward equivalent density* and $\mathbf{y}^{B,d}$ the *downward equivalent surface*. To ensure the existence of $\phi^{B,d}$, $\mathbf{y}^{B,d}$ needs to be located between $\mathcal{F}^B$ and B. Since the potentials induced by both densities satisfy the underlying second order elliptic PDE, using the uniqueness result of the interior Dirichlet problem of this PDE, we only need to match the potentials on a surface (denoted by $\mathbf{x}^{B,d}$) between B and $\mathbf{y}^{B,d}$. We call this surface *downward check surface* (denoted by $u^{B,d}$) and the matched potential *downward check potential* usually choose $\mathbf{y}^{B,d}$ and $\mathbf{x}^{B,d}$ both to be circles in 2D and spheres or cubes in 3D. The potential $\mathbf{y}^{B,d}$ satisfies the following equation for any $\mathbf{x} \in \mathbf{x}^{B,d}$:

$$\int_{\mathbf{y}^{B,d}} G(\mathbf{x}, \mathbf{y}) \phi^{B,d}(\mathbf{y}) \, \mathrm{d}\mathbf{y} = \sum_{i \in I_s^{\mathcal{F}^B}} G(\mathbf{x}, \mathbf{y}_i) \phi_i. \tag{4}$$

In general inverting the integral equations (3) and (4) for a general right-hand side is an ill-conditioned problem since it is an ill-posed infinite dimensional problem. In Section 4 we explain why we still can solve these equations for this specific choice of right-hand sides.

M2M translation. For every leaf box B in the computation tree, the computation of the upward equivalent density $\phi^{B,u}$ from the source densities uses equation (3). The procedure of M2M translation is similar (Figure 3). To translate the upward equivalent

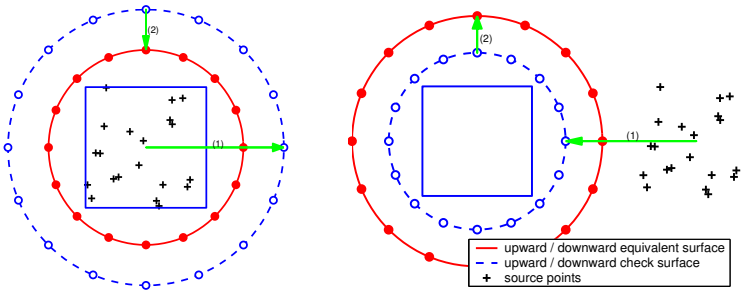

FIG. 2 The equivalent/check surfaces in 2D. *Left*: Given the potential generated by the source densities inside a box located at the points marked with "+", we represent it by using the upward equivalent density located at the upward equivalent surface. The equivalent surface is shown as the solid circle enclosing the box. The upward check potentials induced by the sources and the upward equivalent density are matched at the upward check surface (the dashed circle). *Right*: To represent the potential in the box generated by the source in the far range, we use the downward equivalent density located at the downward equivalent surface. The downward equivalent potentials induced by both sources are matched at the upward check surface. In both plots, the discretization points of the equivalent and check surfaces are equally spaced and marked with "•" and "◦" respectively. For both upward or downward steps, the computation of the equivalent density includes two steps shown by arrows in each plot: (1) the evaluation of the check potential using the original source, and (2) the inversion of the integral equation to obtain the equivalent density.

density from a box A to its parent box B, we solve the following equation for $\phi^{B,u}$ for all $\mathbf{x} \in \mathbf{x}^{B,u}$

$$\textbf{M2M:} \int_{\mathbf{y}^{B,u}} G(\mathbf{x},\mathbf{y})\phi^{B,u}(\mathbf{y})\,\mathrm{d}\mathbf{y} = \int_{\mathbf{y}^{A,u}} G(\mathbf{x},\mathbf{y})\phi^{A,u}(\mathbf{y})\,\mathrm{d}\mathbf{y} \quad \text{or} \quad \phi^{B,u} = U\phi^{A,u}, \tag{5}$$

where U denotes the M2M translation operator. To ensure the existence of $\phi^{B,u}$ for B, $\mathbf{y}^{B,u}$ must enclose $\mathbf{y}^{A,u}$ for any of its children A.

M2L translation. Once the upward equivalent density has been computed for each box, M2L translation computes the downward equivalent density (Figure 3). Suppose A is a box in $\mathcal{F}^B$. The M2L translation is similar to equation (4), and we solve the following equation to find $\phi^{B,d}$: for all $\mathbf{x} \in \mathbf{x}^{B,d}$

$$\textbf{M2L:} \int_{\mathbf{y}^{B,d}} G(\mathbf{x},\mathbf{y})\phi^{B,d}(\mathbf{y})\,\mathrm{d}\mathbf{y} = \int_{\mathbf{y}^{A,u}} G(\mathbf{x},\mathbf{y})\phi^{A,u}(\mathbf{y})\,\mathrm{d}\mathbf{y} \quad \text{or} \quad \phi^{B,d} = T\phi^{A,u}, \tag{6}$$

where T denotes the M2L translation operator. To ensure the existence of $\phi^{B,d}$, $\mathbf{y}^{B,d}$ must be disjoint from $\mathbf{y}^{A,u}$ for all A in $\mathcal{F}^B$.

L2L translation. The L2L translation computes the downward equivalent density of a box B from that of its parent A (Figure 3). The procedure is again similar to equation (4). For all $\mathbf{x} \in \mathbf{x}^{B,d}$, the potential $\phi^{B,d}$ satisfies

$$\textbf{L2L:} \int_{\mathbf{y}^{B,d}} G(\mathbf{x},\mathbf{y})\phi^{B,d}(\mathbf{y})\,\mathrm{d}\mathbf{y} = \int_{\mathbf{y}^{A,d}} G(\mathbf{x},\mathbf{y})\phi^{A,d}(\mathbf{y})\,\mathrm{d}\mathbf{y} \quad \text{or} \quad \phi^{B,d} = D\phi^{A,d}, \tag{7}$$

where D denotes the L2L translation operator. To ensure the existence of $\phi^{B,d}$, $\mathbf{y}^{B,d}$ must lie in $\mathbf{y}^{A,d}$.

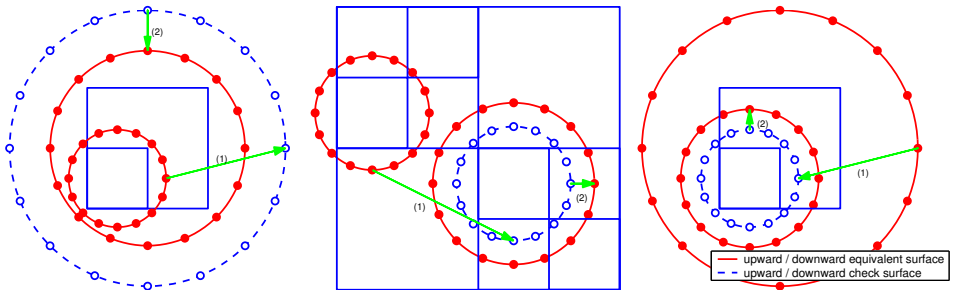

FIG. 3 Three translations in 2D. *Left*: M2M translation. To compute the upward equivalent density of the large square, we evaluate the (upward check) potential at the dashed circle using its child box's upward equivalent density at the small solid circle (this operation is marked with arrow (1)), and invert the integral equation to get its upward equivalent density at the large solid circle (marked with arrow (2)). *Middle*: M2L translation transforms the upward equivalent density of the left box (surrounded by one circle) to the downward equivalent density of the right box (surrounded by two circles). We first evaluate the downward check potential at the dashed circle using the upward equivalent density (located at the small solid circle) (marked with (1)), and then invert the equation to obtain the downward equivalent density at the downward equivalent surface — the large solid circle (marked with (2)). *Right*: L2L translation transforms the downward equivalent density of the large box to its child — the the small box.
In all three figures, the discretization points for the equivalent surface are marked with "•" and the ones for check surface are marked with "○".

Equations (5), (6) and (7) corresponding to M2M, M2L and L2L translations are all ill-conditioned for a general right-hand side. However, in the special case of right-hand sides that we use the equations happen to be well-conditioned. (see Section 4).

Summary. We have described two density representations and three translations used to convert between these equivalent densities. The two equivalent densities correspond to the multipole and local expansions in FMM, while the three translations replace the three transformations in FMM.

In order to guarantee the existence of the equivalent densities the equivalent and check surfaces have to satisfy certain restrictions. We summarize them as follows: for each box B

- $\mathbf{y}^{B,u}$ and $\mathbf{x}^{B,u}$ lie between B and $\mathcal{F}^B$; $\mathbf{x}^{B,u}$ encloses $\mathbf{y}^{B,u}$;
- $\mathbf{y}^{B,d}$ and $\mathbf{x}^{B,d}$ lie between B and $\mathcal{F}^B$; $\mathbf{y}^{B,d}$ encloses $\mathbf{x}^{B,d}$;
- $\mathbf{y}^{B,u}$ encloses $\mathbf{y}^{A,u}$ for any descendant box A,
- $\mathbf{y}^{B,u}$ is disjoint from $\mathbf{y}^{A,d}$ for all A in $\mathcal{F}^B$,
- $\mathbf{y}^{B,d}$ lies inside $\mathbf{y}^{A,d}$, where A is B's parent.

3.2. Discretization

Equations (3), (5), (6), and (7) need to be discretized. In our implementation we use the Nyström's discretization. Alternatively, Galerkin or collocation methods could be used. In 2D we choose circular equivalent and check surfaces. We use the trapezoidal rule to discretize the integral equations; in this manner we obtain super-algebraically convergent quadratures. In 3D this is no longer possible: to the best of our knowledge, there is no

simple quadrature rules for functions defined on spheres that converge super-algebraically. Instead, we use cubes as the equivalent and check surfaces (Figure 4 and 5), and construct quadratures of fixed order on the faces of the cubes. In Section 3.4 we explain how this approach facilitates fast M2L translations, and in Section 5 we show that the accuracy in 3D is not too different from the 2D case.

Surface discretization, 2D case. For a box B centered at $\mathbf{c}$ with side length $2r$, all the related surfaces are circles centered at $\mathbf{c}$. The upward equivalent surface $\mathbf{y}^{B,u}$ has radius $\sqrt{2}r + \epsilon$ for some small number ϵ (The correction ϵ is introduced so that $\mathbf{y}^{B,u}$ do not overlap with the sources in B). The upward check surface $\mathbf{x}^{B,u}$ has radius $(4-\sqrt{2})r - \epsilon$. The downward equivalent surface $\mathbf{y}^{B,d}$ has radius $(4-\sqrt{2})r - \epsilon$. Finally, the downward check surface $\mathbf{x}^{B,d}$ has radius $\sqrt{2}r + \epsilon$ (Figure 2 and 3). All circles are discretized with p equally spaced quadrature points with equal quadrature weights. This simple rule is known to have super-algebraic convergence. The accuracy of our method is determined by the choice of p. Note that our choice of the surfaces satisfies all the restrictions at the end of Section 3.1. We could have chosen the upward/downward check surface to be identical with the upward/downward equivalent surface. However in this case we would need more complex quadrature rules that can be used to integrate singular kernels.

Surface discretization, 3D case. For a box B centered at $\mathbf{c}$ with side length $2r$, all the related surfaces are the boundaries of cubes centered at $\mathbf{c}$. The upward equivalent surface $\mathbf{y}^{B,u}$ is the boundary of a box with side length $2r + \epsilon$. The upward check surface $\mathbf{x}^{B,u}$ is the boundary of a box with side length $6r - \epsilon$. The downward equivalent surface $\mathbf{y}^{B,d}$ is the same as $\mathbf{x}^{B,u}$. Finally, the downward check surface $\mathbf{x}^{B,d}$ is the same as $\mathbf{y}^{B,u}$ (Figure 4 and 5). For every surface, the quadrature points are distributed evenly on six faces, and on every face, the points are distributed on an evenly spaced 2D Cartesian grid. Under this distribution, the quadrature points at the corner of the box are shared by three faces, and those at the edge of the box are shared by two faces. We can also view these quadrature points as the boundary nodes of a 3D regular Cartesian grid. Same as 2D case, we use p to denote the total number of quadrature points on the surface of the box [7]. We choose the quadrature weights in such a way that, on every face, the quadrature rule integrates low order 2D polynomials exactly.

In our experiments, good quadrature results are observed since all the kernels are smooth away from the singularity. Note that, by choosing these surfaces in this way, we meet all the restrictions at the end of Section 3.1.

Discrete Formulation. We use $\{\mathbf{y}_i^{B,u}, 1 \leq i \leq p\}$ to represent the discretization points for $\mathbf{y}^{B,u}$ and $\{\phi_i^{B,u}, 1 \leq i \leq p\}$ the potential at these discretization points. We also use the same notation convention for the discretization of $\mathbf{x}^{B,u}$, $u^{B,u}$, $\mathbf{y}^{B,d}$, $\phi^{B,d}$, $\mathbf{x}^{B,d}$ and $u^{B,d}$. Suppose further that $w_i^{B,u}$ and $w_i^{B,d}$ for $i = 1, \cdots, p$ are the quadrature weights for $\mathbf{y}^{B,u}$ and $\mathbf{y}^{B,d}$. The discretized versions of equations (3),(5), (6) and (7) can be written as: For every $j = 1, \cdots, p$

[7] Note that, in 3D FMM, p is the order of the multipole/local expansion, therefore, p^2 is the actual number of coefficients used in the expansion

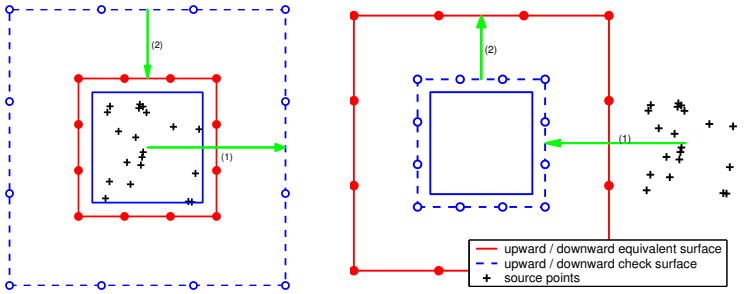

FIG. 4 The equivalent/check surfaces in 3D. We show the cross-sections in one direction. *Left*: the upward equivalent density. *Right*: the downward equivalent density. In both plots, the innermost square stands for the box. The equivalent and check surfaces are both discretized using the boundary nodes of a regular Cartesian grid. The nodes for the equivalent surfaces are marked with "•" and those for the check surfaces with "○".

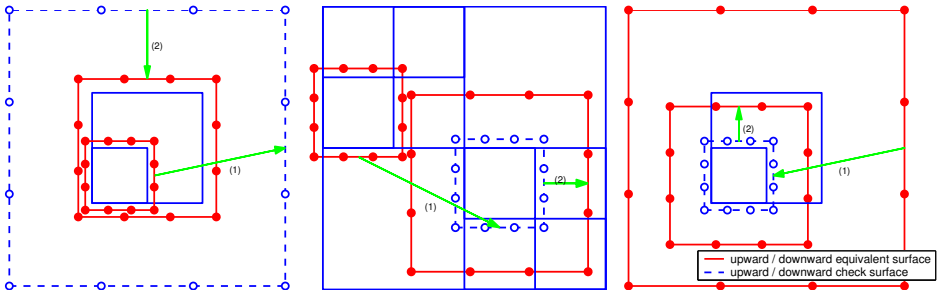

FIG. 5 Three translations in 3D. We only show the cross-sections. *Left*: M2M translation. *Middle*: M2L translation. *Right*: L2L translation. 3D translations are similar to 2D. There are two differences (1) equivalent/check surfaces are now cubes and (2) discretization points are the boundary nodes of a regular Cartesian grid. Note that for M2L translation the discretization points of upward equivalent surface and downward check surface are from the same Cartesian grid, therefore it can be speed up with FFT (interior nodes are padded with zero density).

$$\textbf{S2M:}\quad \sum_{i=1}^{p} G(\mathbf{x}_j^{B,u}, \mathbf{y}_i^{B,u}) w_i^{B,u} \phi_i^{B,u} = \sum_{i \in I_s^B} G(\mathbf{x}_j^{B,u}, \mathbf{y}_i) \phi_i, \tag{8}$$

$$\textbf{M2M:}\quad \sum_{i=1}^{p} G(\mathbf{x}_j^{B,u}, \mathbf{y}_i^{B,u}) w_i^{B,u} \phi_i^{B,u} = \sum_{i=1}^{p} G(\mathbf{x}_j^{B,u}, \mathbf{y}_i^{A,u}) w_i^{A,u} \phi_i^{A,u}, \tag{9}$$

$$\textbf{M2L:}\quad \sum_{i=1}^{p} G(\mathbf{x}_j^{B,d}, \mathbf{y}_i^{B,d}) w_i^{B,d} \phi_i^{B,d} = \sum_{i=1}^{p} G(\mathbf{x}_j^{B,d}, \mathbf{y}_i^{A,u}) w_i^{A,u} \phi_i^{A,u}, \tag{10}$$

$$\textbf{L2L:}\quad \sum_{i=1}^{p} G(\mathbf{x}_j^{B,d}, \mathbf{y}_i^{B,d}) w_i^{B,d} \phi_i^{B,d} = \sum_{i=1}^{p} G(\mathbf{x}_j^{B,d}, \mathbf{y}_i^{A,d}) w_i^{A,d} \phi_i^{A,d}. \tag{11}$$

We have mentioned in Section 3.1 that equations (3),(5), (6) and (7) are ill-conditioned. Regularization schemes are required when we solve the equivalent densities in equations (8) to (11). Our approach to solving this system is described in detail in Section 4.1.

3.3. The complete algorithm

In this section we describe the algorithm in detail. First we give some definitions related to the algorithm. Our definitions closely follow Greengard [11].

Definitions. Neighbors are adjacent boxes in the same level. For uniform distributions of particles, a uniformly refined grid is used. In this case the *neighbor list* L_N^B of a box B is the set of all neighbors of B and B itself. For a box away from the boundaries, the neighbor list contains 9 boxes in 2D or 27 boxes in 3D. These boxes are all contained in $\mathcal{N}^B$.

The *interaction list* L_I^B is the set of children of the neighbors of B's parent which are not B's neighbors. Again, ignoring the boundary effects, this list contains 27 boxes in 2D and 189 boxes in 3D. These boxes are all contained in $\mathcal{F}^B$.

If the particle distribution is uniform a regular grid can be used; however we are primarily interested in non-uniform particle distributions. In this case an adaptively refined grid is needed. The grid is recursively refined until the number of points in each leaf box is less than a fixed number s. Following the adaptive FMM algorithm, we give the following definitions (Figure 6).

For a leaf box B, the U *list* L_U^B contains B itself and the leaf boxes which are adjacent to B. For a non-leaf box, the U list is empty.

The V *list* L_V^B is the set of the children of the neighbors of the parent of B which are not adjacent to B.

If B is a leaf box, the W *list* L_W^B consists of all the descendants of B's neighbors whose parents are adjacent to B, but who are not adjacent to B themselves. For a non-leaf box, the W list is empty.

The X *list* L_X^B consists of all boxes A such that $B \in L_W^C$.

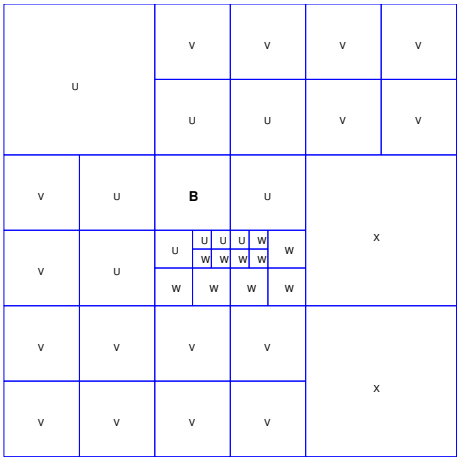

FIG. 6 Lists L_U^B, L_V^B, L_W^B and L_X^B of box B.

For a leaf box B, L_U^B is similar to L_N^B in the uniform case, and L_V^B is similar to L_I^B. There is also a conjugate relation on these four lists. Suppose that A and B are two boxes.

- If A is in L_U^B, then B is in L_U^A.
- If A is in L_V^B, then B is in L_V^A.

- If A is in L_W^B, then B is in L_X^A.
- If A is in L_X^B, then B is in L_W^A.

For a box B, the U,V,W and X lists contain all boxes whose contribution needs to be processed by B itself. The contribution from more distant boxes are considered by B's ancestors. For a box U in L_U^B, a direct computation of the interaction of U's source points with B's target points is necessary since U and B are adjacent. For a box V in L_V^B, we compute the interaction from V to B using M2L translation since two boxes are well-separated. For a box W in L_W^B, we can evaluate the potential directly at B's target points using the upwards equivalent density of W, as B is in the far range of W. Finally, for a box X in L_X^B, since B is still in the near range of X, we represent the potential from X to B by first evaluating the potential at the downwards check surface at B and then invert it to the downwards equivalent density $\phi^{B,d}$. The pseudocode is given in Algorithm 1.

Algorithm 1 adaptive case

ASSUME
  N is the total number of points
  s is the maximum number of points allowed in leaf box

STEP 1 TREE CONSTRUCTION
  for each box B in *preorder* traversal of the tree **do**
    subdivide B if B has more than s points in it
  end for
  for each box B in *preorder* traversal of the tree **do**
    construct L_U^B, L_V^B, L_W^B and L_X^B for B
  end for

STEP 2 UPWARDS PASS
  for each leaf box B in *postorder* traversal of the tree **do**
    evaluate $u^{B,u}$ at $\mathbf{x}^{B,u}$ using $\{\phi_i, i \in I_s^B\}$
    solve for $\phi^{B,u}$ at $\mathbf{y}^{B,u}$ that matches $u^{B,u}$ at $\mathbf{x}^{B,u}$ (Equation (3))
  end for
  for each non-leaf box B in *postorder* traversal of the tree **do**
    add to $u^{B,u}$ at $\mathbf{x}^{B,u}$ the contribution from $\phi^{C,u}$ for each child C of B
    solve for $\phi^{B,u}$ at $\mathbf{y}^{B,u}$ that matches $u^{B,u}$ at $\mathbf{x}^{B,u}$ (Equation (5))
  end for

STEP 3 DOWNWARDS PASS
  for each non-root box B in *preorder* traversal of the tree **do**
    add to $u^{B,d}$ at $\mathbf{x}^{B,d}$ the contribution from $\phi^{V,u}$ for each box V in L_V^B
    add to $u^{B,d}$ at $\mathbf{x}^{B,d}$ the contribution from $\{\phi_i, i \in I_s^X\}$ for each box X in L_X^B
    add to $u^{B,d}$ at $\mathbf{x}^{B,d}$ the contribution from $\phi^{P,d}$, where P is the parent of B
    solve for $\phi^{B,d}$ at $\mathbf{y}^{B,d}$ that matches $u^{B,d}$ at $\mathbf{x}^{B,d}$ (Equation (6) and (7))
  end for
  for each leaf box B in *preorder* traversal of the tree **do**
    add to $\{u_i, i \in I_t^B\}$ the contribution from $\phi^{B,d}$
    add to $\{u_i, i \in I_t^B\}$ the contribution from $\{\phi_i, i \in I_s^U\}$ for each box U in L_U^B
    add to $\{u_i, i \in I_t^B\}$ the contribution from $\phi^{W,u}$ for each box W in L_W^B
  end for

3.4. Implementation Issues

In the previous section we described the overall structure of the algorithms. For clarity some implementation details were omitted. These details, however, are very important for an efficient implementation of any FMM method. The most important issues are the efficient acceleration of the M2L computation, and the overall memory management.

Another aspect of our discussion is the distinction between the setup phase and the fast summation phase. Many times the particle distributions come from discretization of integral equations; then, given a fixed spatial particle distribution, the summation is carried many times (i.e. the matrix vector multiplication within an iterative solver such as GMRES). Many issues that we discuss here are related to efficient multiple evaluations.

Acceleration techniques. In our analysis, we consider only the uniform particle distribution and uniform grids. While analysis of adaptive refinement is possible it requires assumptions on particle distribution. The most expensive part of our algorithm is M2L translations: the evaluation of the contribution to $u^{B,d}$ of a target box B from $\phi^{A,u}$ where A is a source box in the interaction list of B.

We denote the size of the interaction list by I. For a single box, the complexity of the M2L translation is $\mathcal{O}(I \cdot p^2)$. The M2M and L2L translations are applied only once for each box and their contribution to the overall algorithm is not as important. Thus, the M2L part needs to be efficiently implemented since it is one of the two most expensive parts of the algorithm. (The other bottleneck is the computation of particle-to-particle and dense interactions).

SVD-based acceleration (2D). In 2D, we use an SVD-based acceleration technique. We first assemble the matrix $\mathbf{M}$ of the interaction from $\mathbf{y}^{A,u}$ to $\mathbf{x}^{B,d}$. We observe that $\mathbf{M}$ is numerically low rank. The number of the significant singular values of M is small compared to the dimension of M, and all the rest singular values are less than the accuracy required by the pairwise interaction evaluation. Suppose $\mathbf{USV}^T = M$ is the SVD of $\mathbf{M}$. We can store only the columns of $\mathbf{U}$ and $\mathbf{V}$ which correspond to the dominant singular values of $\mathbf{S}$ and discard the rest. This approach gives us an efficient representation of $\mathbf{M}$. In 3D this approach does not yield satisfactory results. Although M2L operators are low rank, in practice the cutoff number of equivalent density points in which the compression is effective, is very large. For this reason an FFT-based approach is preferable.

FFT-based acceleration (3D). Suppose box A is in the interaction list of box B. As mentioned in Section 3.2, $\mathbf{y}^{A,u}$ is chosen to be the boundary of A, and the integration points are the nodes of a Cartesian grid which are on the boundary of of A. The same is true for $\mathbf{x}^{B,d}$. Therefore, by assigning zero density to the grid points in the interior of B we can view the evaluation of the potential $u^{B,d}$ from the density $\phi^{A,u}$ as a 3D convolution. This convolution can be evaluated efficiently by FFT. Since we use 3D convolutions, there are $\mathcal{O}(p^{3/2})$ instead of p densities and targets in each M2L translation. For each box, we carry out FFT and inverse FFT only once, to obtain a $\mathcal{O}(p^{3/2}\log(p))$ complexity. The convolution (pointwise vector multiplication) is applied I times for each box, with $\mathcal{O}(I \cdot p^{3/2})$ complexity.

In [14] and [7] *exponential representation*, an intermediate representation between multipole and local expansions is introduced. Based on this new representation, a *diagonal transformation* is used to transfer between exponential expansions efficiently. This technique cuts down the complexity to $\mathcal{O}(I \cdot p)$, which is asymptotically superior to the $\mathcal{O}(I \cdot p^{3/2})$ complexity of our FFT based acceleration technique. On the other hand,

the translation to exponential representation involves the computation of some nontrivial kernel-dependent quadrature weights, while our FFT based technique only involves potential evaluations and thus is kernel independent.

Storage compression. Since the M2M, M2L and L2L translations are used repeatedly, we precompute and store the matrices of these operators. Three storage compression techniques are used to reduce the memory usage.

Homogeneity. Many common kernels are homogeneous: if we scale the distance between the source point and the target point by a factor α, the potential at the target is amplified by a factor α^k, where k is a constant. For example, the 3D Laplace single layer kernel, $\mathbf{S}(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi}\frac{1}{r}$, has this property. Since the integration points of the equivalent densities of a box are fixed relative to the box, the translation operators between different levels of the computation tree only differ by a constant, usually a power of 2. Hence, instead of storing the matrices for each level, we store only the matrices for a single level. Modified kernels, like modified Laplace, modified Stokes and modified Navier equations, do not have this property.

Symmetry. In 2D the integration points are equally spaced on a circle; in 3D the integration points of the equivalent densities are chosen to be the nodes of a regular Cartesian grid. In both cases they are symmetric with respect to the x, y and z axes. For example, if we flip the positive x direction to be the negative x direction, the positions of the set of the integration points do not change, even though two integration points might swap their positions. Consider the M2M translation. Suppose B is the parent box of two different boxes C_1 and C_2 and we need to evaluate the potential $u^{B,u}$ at $\mathbf{x}^{B,u}$, the contribution from $\phi^{C_1,u}$ at $\mathbf{y}^{C_1,u}$ and from $\phi^{C_2,u}$ at $\mathbf{y}^{C_2,u}$. Suppose we already have the matrix of the operator from $\mathbf{y}^{C_1,u}$ to $\mathbf{x}^{B,u}$. In order to evaluate the contribution from $\phi^{C_2,u}$ at $\mathbf{x}^{B,u}$, we first perform a change of coordinates to move $\mathbf{y}^{C_2,u}$ to $\mathbf{y}^{C_1,u}$, and then evaluate the contribution using the operator from $\mathbf{y}^{C_1,u}$ to $\mathbf{x}^{B,u}$. The we perform another change of coordinates to move $\mathbf{y}^{C_1,u}$ back to $\mathbf{y}^{C_2,u}$. The same techniques can be carried out for M2L and L2L translations.

The above procedure is only correct in the case of a scalar density and a scalar potential. In the cases with vector or tensor densities and potentials, the change of coordinates not only affects the support of the density or potential, but it also modifies their values. Therefore, a rescaling step is necessary after each change of coordinates. A general translation using symmetry involves five steps: (a) forward change of coordinates, (b) rescaling of density, (c) translation using stored matrix, (d) rescaling of potential, and (e) backward change of coordinates. This technique works for all the kernels considered in this paper, and gives us a compression factor of eight in 3D and four in 2D.

Lazy computation. In the case of nonuniform density distribution, the depth of the computation tree can be quite large. However, not all the M2L translations are actually needed in the computation. Therefore, in our algorithm, the matrix representation of a M2L translation is only computed where it is actually needed by some box. This *lazy computation* strategy results in significant savings on memory usage in nonuniform density distributions, and modified kernels.

Complexity. For simplicity, we give the complexities of our method and FMM in [7] for 3D uniform particle distribution. The analysis of the adaptive algorithm is essentially

the same, but more involved and requires assumptions about the particle distribtion. The number of boxes M is approximately N/s. We use p to denote the number of coefficients.

Step	Our method	FMM
S2M translation	$\mathcal{O}(Np + Mp^2)$	$\mathcal{O}(Np)$
M2M translation	$\mathcal{O}(Mp^2)$	$\mathcal{O}(Mp^{3/2})$
M2L translation	$\mathcal{O}(Mp^{3/2}\log p + 189Mp^{3/2})$	$\mathcal{O}(20Mp^{3/2} + 189Mp)$
L2L translation	$\mathcal{O}(Mp^2)$	$\mathcal{O}(Mp^{3/2})$
L2T translation	$\mathcal{O}(Np)$	$\mathcal{O}(Np)$
Near Interaction	$\mathcal{O}(27Ns)$	$\mathcal{O}(27Ns)$

The constants in the complexity estimates are approximately the same for all translations; 189 is the number of the M2L boxes and 27 is the number of boxes in the near interaction. In practice, s is of the same order as p. Therefore, the S2M and L2T steps of both methods are of the same order $\mathcal{O}(Np)$. Our M2L translation is also of the same order as that of [7]. The M2M and L2L steps have higher complexity in our method, due to the fact that no acceleration techniques are applied in these two steps. However, in all experiments in Section 5, we observe that this does not slow down our method significantly since these steps are applied once for each box.

4. ERROR ANALYSIS

Given the direct interaction operator G between a source box B with a well-separated target box A, both at the refinement level l, we examine the error related to the FMM approximation G_h. For uniform distributions $l \approx \log_4(N/s)$. We describe the factorization of G that corresponds to computing the interaction using our algorithm under the assumption that all integrations are carried out exactly. We also show why the local inverse problems are well posed.

FMM factorization. FMM can be viewed as a factorization of the operator G:

$$u_A = G\phi_B = D_l D_{l-1} \dots D_{m+1}\, T\, U_{m+1} \dots U_{l-1} U_l = (\Pi_{i=l}^{m+1} D_i) T (\Pi_{i=m+1}^{l} U_i)\phi_B. \tag{12}$$

Here $u_A = \int_{\mathbf{y}^u(B)} G\phi^{B,u}\, d\mathbf{y}$ is the potential at a target box A, ϕ_B is the density at the source box B and m is the level at which M2L translations is applied. s U_i represents the M2M (upward pass) transformation for a box at level i, D_i represents the L2L transformation (downward pass), and T is M2L (the far field) translation.

It is straightforward to check the validity of (12). The validity of every M2M transformation operator U_i, given by (5), is based on three facts. First, we can solve a well-posed inverse problem to construct the equivalent density $\phi^{B,u}$ that matches the exact potential on the check surface $\mathbf{x}^c$. We discuss the well-posedness of computing $\phi^{B,u}$ further below in this section. Second, $\int_{\mathbf{y}^u(B)} G\phi^{B,u}\, d\mathbf{y}$ satisfies the underlying PDE everywhere in $\mathcal{F}^B$. Third, we have uniqueness of the Dirichlet problem on the exterior of $\mathbf{x}^c$. For example, for the Laplacian this problem is given by $-\Delta u = 0$ in $exterior(\mathbf{x}^c)$, $u = p_c$ on $\mathbf{x}^c$, and has analogous form for the other kernels we consider in this paper. Therefore U_i can be used to compute $\phi^{B,u}$ which results in an exact representation of u_A. Operators T and D_i are constructed similarly to U_i; the only difference is that they require the uniqueness of the interior Dirichlet problem.

Quadrature error. The error associated an approximate integral evaluation:

$$u - \tilde{u} = \int G(\mathbf{x}, \mathbf{y})\phi - \sum_i w_i G(\mathbf{x}, \mathbf{y}_i)\phi_i,$$

is the quadrature error. In 2D we use the trapezoidal rule on the circle which is super-algebraically convergent. To our knowledge, in 3D there is no simple integration rule on the sphere that will result in similar high order accuracy; standard polynomial accuracy algorithms must be used. This is an important difference with the analytic FMM, which guarantees exponential convergence (on the number of multipole terms) for the far field approximation. Nonetheless, in our numerical experiments we did not observe noticeable differences between the 2D and 3D version.

4.1. M2M and L2L operators

In general, one can show that (5) does have a solution if the right hand side is in $L^2(\mathbf{x}^c)$ (see [18], chapter 18). However, due to ill-posedness the problem becomes increasingly ill-conditioned, and spectral cutoff or regularization are necessary in order to obtain a solution.

In our case, however, the problem is well-posed because the right-hand side is in $C^\infty(\mathbf{x}^c)$ and has Fourier coefficients that decay faster than the corresponding singular values of the integral operator. To illustrate this idea we analyze the single layer potential for the Laplacian operator. [8]

Logarithmic kernel for the 2D Laplacian. Suppose the source surface, the equivalent density surface and the check surface, on which we match the source and equivalent density potentials, are concentric circles with radii $0 \leq \rho_s \leq \rho_e \leq \rho_c$.

We solve $\int_{\mathbf{y}^u(B)} G(\mathbf{x}, \mathbf{y})\phi^u(\mathbf{y})\, d\mathbf{y}(\mathbf{x}) = u^c(x)$. Standard logarithm expansion and simple algebraic manipulations yield

$$\log|\mathbf{x} - \mathbf{y}| = \log|\mathbf{x}| + \sum_{k=-\infty, k\neq 0}^{\infty} \frac{(-1)^k}{|k|} \left(\frac{|\mathbf{y}|}{|\mathbf{x}|}\right)^{|k|} e^{ik\theta_x} e^{-ik\theta_y},$$

where θ_x and θ_y are the polar coordinate angles of the position vectors $\mathbf{x}$ and $\mathbf{y}$ respectively. We assume that the source density has zero mean and we drop the $\log|x|$ term. The orthogonality of the trigonometric functions on $L^2(0, 2\pi)$ indicates that the above expression is a diagonalization of the single layer, and by positive definiteness we can identify its singular values with the eigenvalues.

Since $|\mathbf{x}| = \rho_c \geq |\mathbf{y}| = \rho_e$, the singular values decay exponentially: the problem of determining ϕ^u from u^c is not continuous on u^c. In fact, based on the SVD decomposition we can write the inverse as

$$\phi^u = \frac{1}{2\pi\rho_e} \sum_k \frac{|k|}{(-1)^k} \left(\frac{\rho_c}{\rho_e}\right)^{|k|} e^{ik\theta_y} \int_\gamma e^{-ik\theta_x} u^c(\theta_x)\rho_c\, d\gamma,$$

and small perturbations on the high frequency components of u^c get exponentially amplified. In our problem however, u^c is not arbitrary. Here u^c is $\int_{\mathbf{y}(B)} G(\mathbf{x}, \mathbf{y})\phi(\mathbf{y})\, d\mathbf{y}(\mathbf{x})$. In

[8] We could have taken the location of the check points to coincide with the location the equivalent densities. In this case (5) is a well-posed boundary integral equation. But this approach is kernel-dependent since the limit boundary values depend on the specific kernel (e.g., double layer has jumps, single layer is continuous).

this case ϕ^u is given by

$$\phi^u = \frac{1}{2\pi\rho_e} \sum_k \left(\frac{\rho_s}{\rho_e}\right)^{|k|} e^{ik\theta_y} \int_\gamma e^{-ik\theta_y} \rho_s \phi \, d\gamma.$$

If ϕ is sufficiently smooth, the modes of ϕ^u have exponentially decaying coefficients, and therefore the problem is well-posed.

For general kernels we can write the above computations using the SVD decomposition of an integral operator

$$G_e \phi^u(\gamma_c) = G_s \phi(\gamma_c), \quad U_e S_e V_e^T \phi^u = U_s S_s V_s^T \phi$$
$$\phi^u = V_e S_e^{-1} U_e^T U_s S_s V_s^T \phi, \quad \phi^u = V_e S_e^{-1} S_s V_s^T \phi.$$

For analytic kernels the Fourier modes again decay exponentially since $\rho_s < \rho_e$ \$. Here we have used $U_e^T U_s = I$. This only true for concentric balls; this assumption causes no loss of generality since we can replace the original source density on a enclosing ball by solving a well-posed boundary integral equation. We conclude that the mapping from ϕ to ϕ^u is bounded.

Discretization and regularization. Although the infinite dimensional problem is well-conditioned, we can not directly invert compact kernels which are discretized by Nyström's method [18]. One alternative is to use Nyström's method with Tikhonov regularization (i.e. ppenalize the L^2 norm of ϕ^u). We have combined this approach with spectral cutoff. For ill-posed problems the penalty parameter is chosen to filter out the noise. In our case "noise" is floating point error and high frequency aliasing. We have chosen value of the penalty parameter to be the desired accuracy of the overall computation. This approach has worked robustly for all the kernels we have considered in this paper.

5. NUMERICAL RESULTS

In this section we present numerical results for our method. First, we examine the accuracy of the equivalent density approximation. Second, we present results on the overall accuracy of the method.

5.1. Accuracy on the equivalent density approximation

In this section we present results that indicate that our equivalent density approximations give good accuracy in both two and three dimensions.

For two and three dimensions we show the results of three kernels: the Laplace single layer kernel, the modified Stokes double layer kernel and the Navier single layer kernel (Figure 7 and 8). For each kernel, the left plot is the accuracy of the upward equivalent density approximation, and the right one is the accuracy of the downward equivalent density approximation. For the upward equivalent density, we give the error for points in the exterior of the source box in the region corresponding to the interaction list of the box. For the downward equivalent density we give the error in the interior of the box. In all plots, the side length of the box is 2; we calculate the error by taking the maximum norm over a sphere centered at the center of the box. The abscissa of a plot is the radius of the sphere, and the ordinate is the logarithm of the error.

2D case. Figure 7 shows the error of the equivalent density approximation for the 2D Laplace single layer kernel, the 2D modified Stokes double layer kernel and the 2D Navier single layer kernel. In all three cases, the source density is located close to one of the box corners.

Although not reported here, we have generated similar plots for all kernels given in Section A. All results exhibited similar accuracy. [9]

We do not have an analytic error bound like the FMM algorithm for the Laplace equation. However, Figure 7 shows that our scheme gives comparable accuracy. As discussed in Section 4 this is due to the fast decay of the far field. For a box B, in the far range $\mathcal{F}^B$, the main contribution to the potential comes from the low frequency part of the densities in B, while the contribution from the high frequency part decays extremely fast. By using equally spaced points on circles, we can approximate the low frequency part of the densities with very high accuracy.

3D case. Figure 8 shows the equivalent density approximation errors for the 3D Laplace single layer kernel, the 3D modified Stokes double layer kernel and the 3D Navier single layer kernel. In each case, the source density is placed close to one of the box corners.

5.2. Overall approximation error

In this section we give wall-clock time and memory requirements for several kernels. All experiments were performed on a Sun Ultra 80 workstation with a 450 MHz CPU. In 3D case, the FFTW package is used for FFT computation. Our code has been implemented in C++.

In our experiments we assume that the source points and the target points coincide. We use three sets of density distributions in the cube with range $[-1, 1]$ in each dimension. The first set is a distribution on a sphere, which is typically nonuniform. The second set is a uniform distribution of density in a cube. The last set has densities only at one of the box corners. The objective of this set of points is to check the stability of multiple M2M and L2L transformations of our method. For all density distributions the densities are chosen randomly from $[0, 1]$. The three data sets for the 3D case are shown in Figure 9.

We organize the table in a way similar to [7].

The columns of every table represent the following quantities.

N : the number of points used in computation (we use the same number of source and target points).

L : the number of levels of the computation tree.

M : the number of boxes in the computation tree.

p : the number of discretization points used in the equivalent density approximations. In 2D examples, we use 16, 24, and 32 points. In 3D examples, we choose the discretization points to be the boundary nodes of volume Cartesian grids of size $4\times4\times4$, $6\times6\times6$, $8\times8\times8$. These numbers correspond to 56, 152 and 296 points respectively.

s : the maximum number of points allowed in a leaf box of the computation tree.

[9] In some plots for 2D case, the 32-point error curve has larger error than the 24-point error curve. This is related to the spectral cutoff: we use a value of about 10e-12 when solving the inverse problem and this complicates direct comparisons as we increase p.

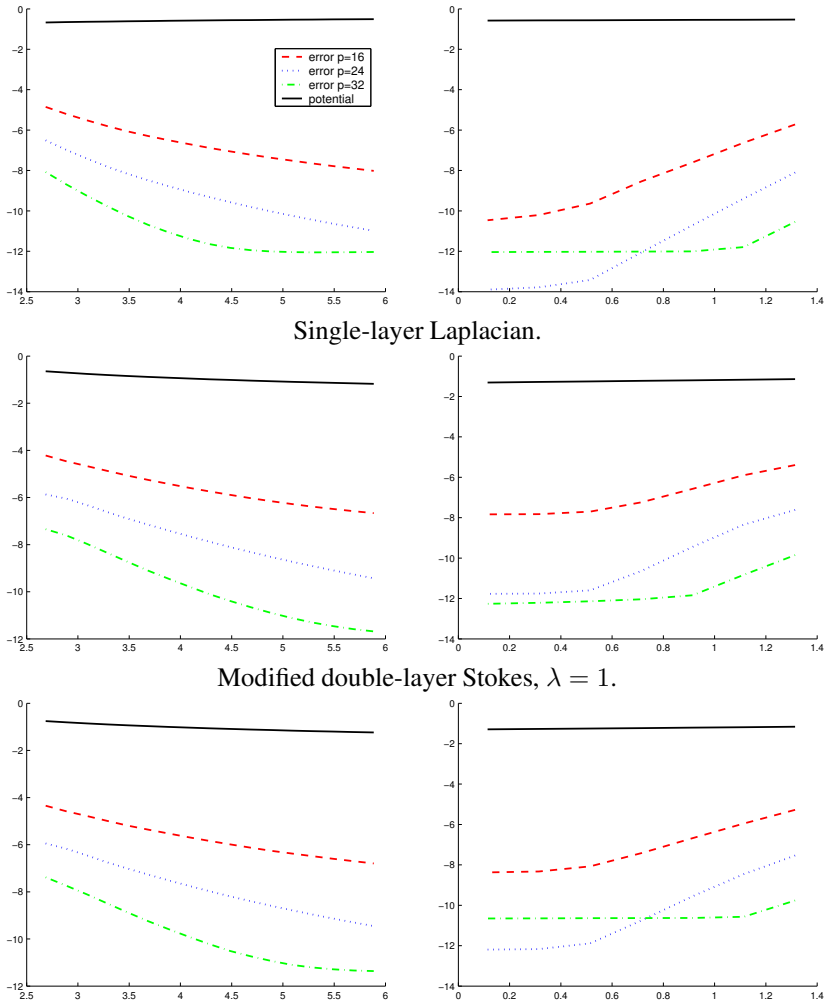

FIG. 7 Results of the equivalent density approximation in 2D. *Left*: the error of the upward equivalent density approximation. *Right*: the error of the downward equivalent density approximation. The abscissa of the plots is the radius of the sphere, and the ordinate is the logarithm of the error. The solid curve is the maximum norm of the potential. The remaining three curves are the maximum norm error for 16-, 24- and 32-point approximation of the equivalent densities. For modified Stokes, we tested λ from 1e-3 to 1e+3 and obtained similar error plots. For λ greater than 1e+3, far field interaction is negligible.

Storage: the memory used to store M2M, M2L, and L2L translations.

T_{fmm} : the running time of our algorithm.

T_{dir} : the running time of the direct evaluation. For each table, only the number in the first line is actually tested, all other numbers are obtained by extrapolation. The error is computed in relative 2-norm. We randomly select k points $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$, evaluate the potential u_i using our algorithm and the potential $\tilde{u}_i$ using direct evaluation at

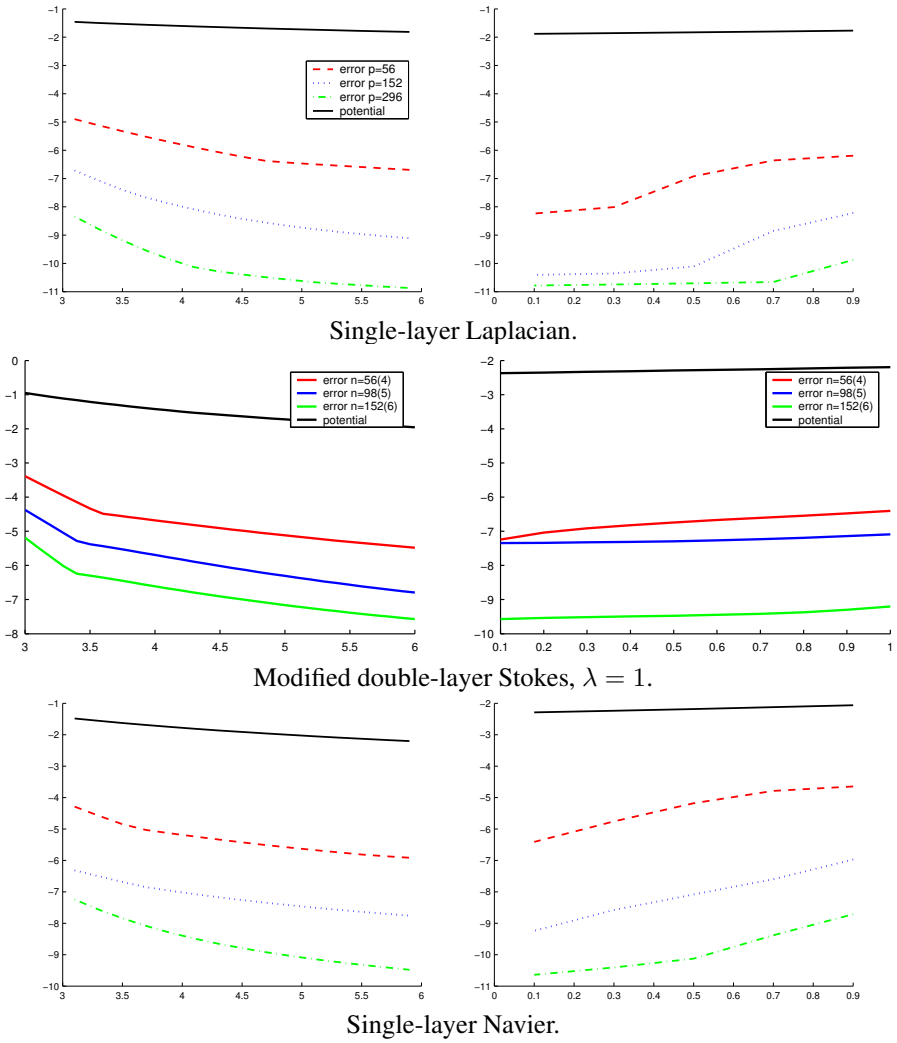

FIG. 8 Results of the equivalent density approximation in 3D. For each plot, the solid curve gives the maximum norm of the potential. The rest three curves give the maximum norm error where the equivalent density is approximated with 56, 152 and 296 points. These numbers correspond to discretization points that are the boundary nodes of volume Cartesian grids of size $4\times4\times4$, $6\times6\times6$, $8\times8\times8$ (per box).

these k points. The error is estimated using the formula from [7]:

$$E = \left(\frac{\sum_{i=1}^{k} |u_i - \tilde{u}_i|^2}{\sum_{i=1}^{k} |\tilde{u}_i|^2} \right)^{1/2},$$

where k is chosen to be 40 in all experiments.

Below, we report the results on the first two data sets (nonuniform nd uniform distribution) for five different kernels:

- 2D Laplace single layer kernel (Table 1),

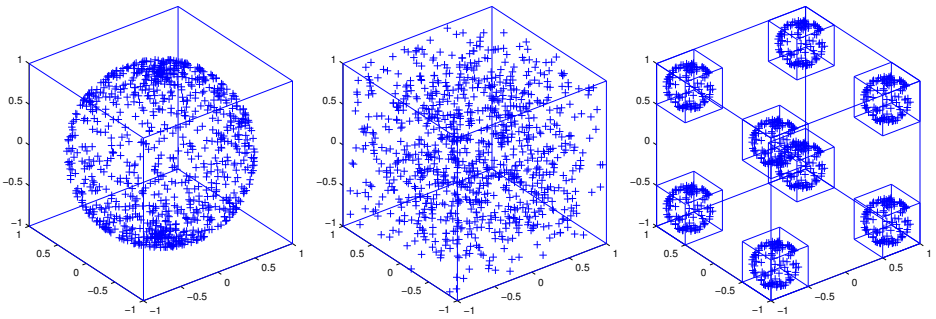

FIG. 9 Three data sets in 3D: *Left*: densities distributed on the unit sphere, *Middle*: densities distributed uniform in the unit cube, *Right*: densities distributed at the eight corners of the unit cube.

- 3D Laplace single layer kernel (Table 2),
- 3D Modified Laplace single layer kernel (Table 3),
- 3D Modified Stokes double layer kernel (Table 4),
- 3D Navier single layer kernel (Table 5).

Our results from 2D are quite satisfactory since we can compute interactions between 1.5 million particles in 4 digits of accuracy on less than 7 minutes, as we can see in Table 1. We discuss relative performance of our method in greater detail in in the 3D case since this is more difficult to implement efficiently. We compare with results from two papers: the single-layer 3D Laplacian results of Cheng, Greengard, and Rokhlin [7] and modified single-layer 3D Laplacian results of Greengard and Huang [12].

In the first paper the authors use a 167 MHz Sun workstation and in the second a 440 MHz Sun workstation. As mentioned before we are using a 450 Sun. For the purposes of comparison, we use the total number of CPU cycles in millions per grid point. We compute this number as

$$\eta = \frac{T_{fmm} \times \mathrm{CPU}}{N}.$$

η_a and η are the numbers of cycles per particle for the analytic FMM and and for our algorithm respectively. This is a only rough estimate that does not take into account the difference in chip architecture (e.g., memory bus clock), different floating point precision of the calculations (most calculations in the first paper were performed in single precision, all our results are in double precision), and different input densities.

First, we compare Table 2 with Tables IV, V, and VI of [7]. For the three digit accuracy (Table IV) the average η_a is 0.07 for single precision. Our method achieves an η equal to 0.11 (in double digit accuracy), approximately a factor of 1.5 slower. Similar conclusions hold for the 6-digit accuracy results(Table V), for which the analytic FMM achieves $\eta_a = 0.15$ in single precision, whereas our method achieves $\eta = 0.23$ in double precision. For the modified single layer Laplacian we compare the 6-digit accuracy entries (Table I, [12]), with Table 3 (uniform distribution in a cube). In this case $\eta_a = 0.3$ and $\eta = 0.4$, which is slightly better than 1.5; the actual difference in performance is even less, since we achieving about one additional digit of accuracy (average error 7×10^7 for the analytic FMM compared an average of 7×10^8 in our case).

Another reason our method is slower might be related to the dense interactions. In order to save storage we do not precompute them, and we have found that this slows down our

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
32768	10	2989	16	40	1.52e+00	1.53e+00	1.71e+02	2.80e-06
131072	12	11857	16	40	1.91e+00	5.85e+00	2.74e+03	1.24e-06
524288	14	47241	16	40	2.30e+00	2.36e+01	4.39e+04	1.51e-06
2097152	16	190601	16	40	2.69e+00	9.32e+01	7.02e+05	2.80e-06
32768	9	1597	24	60	2.97e+00	1.92e+00	1.71e+02	2.68e-08
131072	12	6505	24	60	3.94e+00	7.47e+00	2.74e+03	2.84e-08
524288	14	26073	24	60	5.10e+00	2.97e+01	4.39e+04	3.36e-08
2097152	16	104129	24	60	5.98e+00	1.24e+02	7.02e+05	2.24e-08
32768	9	1493	32	80	5.28e+00	2.23e+00	1.71e+02	1.89e-10
131072	11	5953	32	80	6.84e+00	1.03e+01	2.74e+03	1.77e-10
524288	13	23825	32	80	8.41e+00	4.04e+01	4.39e+04	7.05e-10
2097152	15	95425	32	80	9.97e+00	1.49e+02	7.02e+05	6.03e-10

The particles are uniformly distributed on the perimeter of a circle.

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
32768	8	2837	16	40	1.14e+00	1.45e+00	1.71e+02	5.72e-07
131072	10	12245	16	40	1.53e+00	5.26e+00	2.74e+03	3.71e-07
524288	12	47829	16	40	1.92e+00	2.16e+01	4.39e+04	4.46e-07
2097152	14	189717	16	40	2.31e+00	8.89e+01	7.02e+05	5.24e-07
32768	7	1557	24	60	2.13e+00	1.78e+00	1.71e+02	2.05e-09
131072	9	5909	24	60	3.01e+00	7.21e+00	2.74e+03	2.50e-09
524288	11	25557	24	60	3.88e+00	2.75e+01	4.39e+04	1.64e-09
2097152	14	104085	24	60	4.85e+00	1.07e+02	7.02e+05	1.48e-09
32768	7	1557	32	80	3.78e+00	2.12e+00	1.71e+02	2.83e-11
131072	9	5269	32	80	5.34e+00	8.81e+00	2.74e+03	2.87e-11
524288	11	23893	32	80	6.91e+00	3.54e+01	4.39e+04	2.17e-11
2097152	13	95253	32	80	8.47e+00	1.34e+02	7.02e+05	6.50e-11

The particles are uniformly distributed inside a cube.

TABLE 1
Performance for particles interacting via the single-layer Laplacian in 2D.

method by a factor of 2 to 4. The most time consuming part is computing the $1/\sqrt{(\mathbf{r}\cdot\mathbf{r})}$ term, which we have found impossible to optimize either with lookup tables or with special vector routines available from most vendors. For large problems that require several summations for the same particle partitions further running time improvements can be achieved by precomputing and storing all dense interactions. The memory requirements in this case can be substantial.

In conclusion, it appears that our method compares reasonably well with the analytic FMM by being a factor of 1.5 or less slower. On the other hand extending our code from Laplacian to the modified Laplacian was very easy, we just had to implement a different kernel evaluation. Inspecting the results for the other kernels we can confirm the $\mathcal{O}(N)$ complexity of our method and the convergence to the exact sum as we increase the number of quadrature points.

In all experiments, we store only the linear operators for M2M, M2L and L2L translations, since these operators are applied repetitively in a single pairwise interaction evaluation. The dense interactions between adjacent boxes are not stored. The storage number reported in all tables considers only the memory used by M2M, M2L and L2L operators, while the storage used to store the densities and potentials (which scales linearly with re-

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
24576	6	1377	56	60	1.72e+00	5.72e+00	9.74e+01	2.12e-05
98304	7	5049	56	60	1.72e+00	2.38e+01	1.56e+03	3.21e-05
393216	8	19065	56	60	1.72e+00	9.51e+01	2.49e+04	6.08e-05
1572864	9	76185	56	60	1.72e+00	3.82e+02	3.99e+05	6.03e-05
24576	5	585	152	150	5.90e+00	1.16e+01	9.74e+01	3.34e-07
98304	6	2289	152	150	5.90e+00	4.76e+01	1.56e+03	5.86e-08
393216	7	11193	152	150	5.90e+00	2.18e+02	2.49e+04	2.45e-07
1572864	9	44145	152	150	5.90e+00	8.35e+02	3.99e+05	3.08e-07
24576	4	273	296	250	1.47e+01	1.81e+01	9.74e+01	1.59e-09
98304	6	1449	296	250	1.47e+01	8.15e+01	1.56e+03	1.40e-09
393216	7	5073	296	250	1.47e+01	3.41e+02	2.49e+04	1.10e-09
1572864	8	19161	296	250	1.47e+01	1.38e+03	3.99e+05	2.81e-09

The particles are distributed on the surface of a sphere.

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
24576	4	585	56	60	1.72e+00	6.40e+00	9.74e+01	6.64e-06
98304	5	3657	56	60	1.72e+00	3.11e+01	1.56e+03	1.27e-05
393216	7	28233	56	60	1.72e+00	1.30e+02	2.49e+04	5.00e-05
1572864	8	88137	56	60	1.72e+00	4.08e+02	3.99e+05	5.84e-05
24576	4	585	152	150	5.90e+00	1.60e+01	9.74e+01	1.54e-08
98304	5	3657	152	150	5.90e+00	9.28e+01	1.56e+03	4.70e-08
393216	6	14409	152	150	5.90e+00	3.18e+02	2.49e+04	1.10e-07
1572864	7	37449	152	150	5.90e+00	8.47e+02	3.99e+05	2.13e-07
24576	4	585	296	250	1.47e+01	3.65e+01	9.74e+01	5.25e-10
98304	4	585	296	250	1.47e+01	1.11e+02	1.56e+03	4.57e-10
393216	5	3657	296	250	1.47e+01	4.31e+02	2.49e+04	6.85e-10
1572864	6	17481	296	250	1.47e+01	1.46e+03	3.99e+05	1.46e-09

The particles are uniformly distributed inside a cube.

TABLE 2
Performance for particles interacting via the single layer Laplacian in 3D.

spect to the number of points and boxes) is not included. This explains why for the results of homogeneous kernels (Tables 2 and 5), the storage numbers remain small and do not scale with the number of points and the number of levels.

Stability of M2M and L2L. Here we test the stability of the M2M and L2L translations of our algorithm using the last data set which only has density distribution at the corners of the cube. Table 6 shows the result on this data set with 2D Laplace single layer kernel. Table 7 reports the errors with 3D Laplace single layer kernel.

6. CONCLUSIONS AND FUTURE WORK

We have presented a new kernel-independent fast multipole method, which generalizes FMM to a broad class elliptic kernels while attaining an algorithmic complexity (including constants) which is on par with the analytic FMM. Here we summarize the main features of our algorithm.

- Our algorithm has the same structure as the original adaptive FMM method.

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
6144	5	441	56	60	4.55e+00	1.97e+00	1.15e+01	3.55e-05
24576	6	1377	56	60	6.27e+00	8.24e+00	1.83e+02	7.71e-05
98304	7	5049	56	60	8.29e+00	3.33e+01	2.94e+03	3.11e-05
393216	8	19065	56	60	1.00e+01	1.28e+02	4.70e+04	8.22e-05
6144	4	225	152	150	1.08e+01	4.38e+00	1.15e+01	2.48e-07
24576	5	585	152	150	1.57e+01	1.99e+01	1.83e+02	9.55e-08
98304	6	2289	152	150	2.26e+01	7.58e+01	2.94e+03	3.18e-07
393216	7	11193	152	150	2.85e+01	3.39e+02	4.70e+04	3.63e-07
6144	3	57	296	250	1.18e+01	6.90e+00	1.15e+01	2.50e-09
24576	4	273	296	250	2.64e+01	3.00e+01	1.83e+02	1.88e-09
98304	6	1449	296	250	5.30e+01	1.23e+02	2.94e+03	1.96e-09
393216	7	5073	296	250	6.99e+01	5.35e+02	4.70e+04	3.71e-09

The particles are distributed on the surface of a sphere.

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
6144	4	585	56	60	3.35e+00	3.72e+00	1.15e+01	5.28e-06
24576	4	585	56	60	3.35e+00	1.06e+01	1.83e+02	2.29e-05
98304	5	3657	56	60	5.07e+00	4.25e+01	2.94e+03	3.98e-05
393216	7	28233	56	60	8.14e+00	1.64e+02	4.70e+04	4.88e-05
6144	3	73	152	150	5.38e+00	4.09e+00	1.15e+01	2.10e-08
24576	4	585	152	150	1.13e+01	2.11e+01	1.83e+02	9.86e-08
98304	5	3657	152	150	1.72e+01	1.08e+02	2.94e+03	7.23e-08
393216	6	14409	152	150	2.31e+01	4.14e+02	4.70e+04	4.57e-08
6144	3	73	296	250	1.29e+01	5.87e+00	1.15e+01	7.15e-10
24576	4	585	296	250	2.75e+01	4.39e+01	1.83e+02	6.02e-10
98304	4	585	296	250	2.75e+01	1.98e+02	2.94e+03	4.28e-10
393216	5	3657	296	250	4.22e+01	6.65e+02	4.70e+04	8.24e-10

The particles are uniformly distributed in a cube.

TABLE 3
Performance of our method for particles interacting via the modified single layer Laplacian in 3D.

- We have demonstrated that the method performs well for single and double layers, the Laplacian, the modified Laplacian, the Stokes, the modified Stokes, and the Navier kernels in two and three dimensions. By providing just a kernel evaluation routine our method is immediately applicable, as long as the kernel is associated with a non-oscillatory second-order elliptic PDEs.
- Comparisons of the running times between our method and the best known FMM implementations, and for same accuracy levels, indicate that our approach was successful in efficiently extending FMM to other kernels.
- To our knowledge, our results are the first fast summation computations for the modified Stokes and Navier operators.
- Our method is also directly applicable for derivatives of the kernels we have presented here. Indeed, we have tested our method on the hypersingular kernels resulting from taking the stresses of the double layer Stokes and Navier equations.
- The M2L translations in our method are suboptimal. In 3D, the analytic exponential translations require $\mathcal{O}(p)$, whereas our method requires $\mathcal{O}(p^{3/2})$, with p being the

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
6144	5	441	56	60	8.18e+01	2.65e+01	1.04e+02	9.56e-04
24576	6	1377	56	60	1.13e+02	1.02e+02	1.66e+03	1.45e-03
98304	7	5049	56	60	1.49e+02	3.91e+02	2.66e+04	1.47e-03
6144	4	225	152	150	2.00e+02	7.59e+01	1.04e+02	5.66e-06
24576	5	585	152	150	2.92e+02	2.39e+02	1.66e+03	6.90e-06
98304	6	2289	152	150	4.20e+02	1.01e+03	2.66e+04	1.06e-05
6144	3	57	296	250	2.16e+02	6.44e+01	1.04e+02	8.77e-08
24576	4	273	296	250	4.89e+02	3.59e+02	1.66e+03	1.67e-07
98304	6	1449	296	250	9.87e+02	1.69e+03	2.66e+04	1.88e-07

The particles are distributed on the surface of a sphere.

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
6144	4	585	56	60	6.03e+01	6.97e+01	1.04e+02	5.32e-04
24576	4	585	56	60	6.03e+01	1.23e+02	1.66e+03	5.01e-04
98304	5	3657	56	60	9.13e+01	6.09e+02	2.66e+04	7.00e-04
6144	3	73	152	150	9.87e+01	4.35e+01	1.04e+02	1.77e-06
24576	4	585	152	150	2.09e+02	3.57e+02	1.66e+03	2.96e-06
98304	5	3657	152	150	3.19e+02	2.04e+03	2.66e+04	9.32e-06
6144	3	73	296	250	2.36e+02	7.63e+01	1.04e+02	3.71e-08
24576	4	585	296	250	5.09e+02	8.28e+02	1.66e+03	8.02e-08
98304	4	585	296	250	5.09e+02	2.01e+03	2.66e+04	9.88e-08

The particles are uniformly distributed in a cube.

TABLE 4
Performance of our method for particles interacting via the modified double layer Stokes kernel in 3D.

number of moments in FMM, and the number of discretization points in our method.

- Our method does not have the precise error estimates that come with the original FMM; derivations in Section 4 can be extended to obtain such estimates, but this is non-trivial. Unlike FMM our technique introduces error associated with M2M and L2L transformations.

In this paper we have focused on second order constant coefficient PDEs with non-oscillatory solutions. However, our method is not restricted to such systems. It should be straightforward to generalize it to higher order systems like the biharmonic equation. In such cases the Dirichlet problem involves first and second derivatives of the underlying field. We can either differentiate the kernel to obtain the derivatives or use a set of two check-point surfaces. We plan to explore this approach in the future.

Another class of problems is related to second order PDEs with oscillatory solutions or Helmholtz-type problems. For low frequencies we have performed preliminary tests (on the M2M and L2L transformations) that indicate that our method works as is. An implementation for this class of problems, adding the kernels and support for complex numbers, is under way.

Finally let us mention that our method has been fully parallelized using MPI. Algorithmic details and numerical results will be presented in a future paper.

APPENDIX A: KERNELS

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
6144	5	441	56	60	1.55e+01	1.29e+01	5.91e+01	8.54e-05
24576	6	1377	56	60	1.55e+01	4.93e+01	9.46e+02	6.71e-05
98304	7	5049	56	60	1.55e+01	1.98e+02	1.51e+04	6.32e-05
6144	4	225	152	150	5.50e+01	3.29e+01	5.91e+01	1.07e-06
24576	5	585	152	150	5.50e+01	1.10e+02	9.46e+02	1.66e-06
98304	6	2289	152	150	5.50e+01	4.59e+02	1.51e+04	1.02e-06
6144	3	57	296	250	1.08e+02	3.28e+01	5.91e+01	7.30e-09
24576	4	273	296	250	1.36e+02	1.82e+02	9.46e+02	8.51e-09
98304	6	1449	296	250	1.36e+02	8.51e+02	1.51e+04	8.73e-09

The particles are distributed on the surface of a sphere.

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
6144	4	585	56	60	1.55e+01	3.41e+01	5.91e+01	3.70e-05
24576	4	585	56	60	1.55e+01	6.65e+01	9.46e+02	4.82e-05
98304	5	3657	56	60	1.55e+01	3.13e+02	1.51e+04	6.68e-05
6144	3	73	152	150	4.94e+01	2.19e+01	5.91e+01	1.81e-07
24576	4	585	152	150	5.50e+01	1.62e+02	9.46e+02	3.50e-07
98304	5	3657	152	150	5.50e+01	9.48e+02	1.51e+04	4.86e-07
6144	3	73	296	250	1.18e+02	3.78e+01	5.91e+01	2.56e-09
24576	4	585	296	250	1.36e+02	4.22e+02	9.46e+02	3.58e-09
98304	4	585	296	250	1.36e+02	1.00e+03	1.51e+04	4.39e-09

The particles are uniformly distributed in a cube.

TABLE 5
Performance of our method for particles interacting via the single layer Navier kernel in 3D.

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
524288	18	47449	16	40	2.17e+00	2.17e+01	4.39e+04	4.46e-06
524288	18	26041	24	60	4.54e+00	2.63e+01	4.39e+04	1.20e-08
524288	17	23833	32	80	7.91e+00	3.50e+01	4.39e+04	1.04e-10

TABLE 6
Performance of our method for a the 2D single layer Laplacian. In this experiment the particles are distributed over the boundaries of four circles. These circles are quite small compared the size of the (square) computational domain, and located near to the four corners of the domain. In this way the tree is "forced" to have several levels (up to 18). We use this experiment to test the numerical stability of our M2M and L2L translations.

In this section, we give a summary of the elliptic partial differential equations (PDE) studied in this paper and their relevant kernels. In the formulas below, $\mathbf{y}$ is the location of the singularity, $\mathbf{x}$ is the location the evaluation point, $\mathbf{n}$ a unit vector (usually the normal direction at $\mathbf{y}$), $\mathbf{r} = \mathbf{x} - \mathbf{y}$ and $r = |\mathbf{r}|$, denoting the length or $\mathbf{r}$. $\mathbf{S}$ stands for single layer and $\mathbf{D}$ for double layer.

Laplace Equation.

$$-\Delta u = 0,$$

N	L	M	p	s	Storage (Mb)	T_{fmm} (sec)	T_{dir} (sec)	Error
196608	12	11057	56	60	1.72e+00	4.58e+01	6.23e+03	1.75e-05
196608	11	4721	152	150	5.90e+00	1.04e+02	6.23e+03	1.20e-07
196608	10	2225	296	250	1.47e+01	1.50e+02	6.23e+03	1.53e-09

TABLE 7
Performance of our method for a the 3D single layer Laplacian. In this experiment the particles are distributed over the boundaries of eight spheres. These spheres are quite small compared the size of the (cubic) computational domain, and located near to the eight corners of the box.

$$\mathbf{S}(\mathbf{x},\mathbf{y}) = \begin{cases} \frac{1}{2\pi}\ln\frac{1}{r} & \text{(2D)} \\ \frac{1}{4\pi}\frac{1}{r} & \text{(3D)} \end{cases}, \qquad \mathbf{D}(\mathbf{x},\mathbf{y}) = \begin{cases} -\frac{1}{2\pi}\frac{1}{r^2}(\mathbf{r}\cdot\mathbf{n}) & \text{(2D)} \\ -\frac{1}{4\pi}\frac{1}{r^3}(\mathbf{r}\cdot\mathbf{n}) & \text{(3D)} \end{cases}.$$

Modified Laplace Equation.

$$\alpha u - \Delta u = 0,$$

$$\mathbf{S}(\mathbf{x},\mathbf{y}) = \begin{cases} \frac{1}{2\pi}k_0(\lambda r) & \text{(2D)} \\ \frac{1}{4\pi}\frac{1}{r}\mathrm{e}^{-\lambda r} & \text{(3D)} \end{cases}, \qquad \mathbf{D}(\mathbf{x},\mathbf{y}) = \begin{cases} -\frac{\lambda}{2\pi}\frac{k_1(\lambda r)}{r}(\mathbf{r}\cdot\mathbf{n}) & \text{(2D)} \\ -\frac{1}{4\pi}\left(\frac{1}{r^3}+\frac{\lambda}{r^2}\right)\mathrm{e}^{-\lambda r}(\mathbf{r}\cdot\mathbf{n}) & \text{(3D)} \end{cases},$$

where $\lambda = \sqrt{\alpha}$.

Stokes Equation (Incompressible creeping flows).

$$-\mu\Delta u + \nabla p = 0,\ \mathrm{Div}\, u = 0$$

$$\mathbf{S}(\mathbf{x},\mathbf{y}) = \begin{cases} \frac{1}{4\pi\mu}(\ln\frac{1}{r}\mathbf{I} + \frac{\mathbf{r}\otimes\mathbf{r}}{r^2}) & \text{(2D)} \\ \frac{1}{8\pi\mu}(\frac{1}{r}\mathbf{I} + \frac{\mathbf{r}\otimes\mathbf{r}}{r^3}) & \text{(3D)} \end{cases}, \qquad \mathbf{D}(\mathbf{x},\mathbf{y}) = \begin{cases} -\frac{1}{\pi}\frac{\mathbf{r}\otimes\mathbf{r}}{r^4}(\mathbf{r}\cdot\mathbf{n}) & \text{(2D)} \\ -\frac{6}{8\pi}\frac{\mathbf{r}\otimes\mathbf{r}}{r^5}(\mathbf{r}\cdot\mathbf{n}) & \text{(3D)} \end{cases}.$$

Modified Stokes Equation (Unsteady incompressible creeping flows).

$$\alpha u - \mu\Delta u + \nabla p = 0,\ \mathrm{Div}\, u = 0$$

$$\mathbf{S}(\mathbf{x},\mathbf{y}) = \frac{1}{\mu}\left(G\mathbf{I} + H(\mathbf{r}\otimes\mathbf{r})\right),$$

$$\mathbf{D}(\mathbf{x},\mathbf{y}) = A\left((\mathbf{r}\cdot\mathbf{n})\mathbf{I} + \mathbf{n}\otimes\mathbf{r}\right) + B(\mathbf{r}\otimes\mathbf{n}) + C(\mathbf{r}\cdot\mathbf{n})(\mathbf{r}\otimes\mathbf{r}),$$

where

$$G = -f_{rr} - (d-2)\frac{f_r}{r},$$

$$H = \frac{f_{rr}}{r^2} - \frac{f_r}{r^3},$$

$$A = -\frac{f_{rrr}}{r} - (d-3)\frac{f_{rr}}{r^2} + (d-3)\frac{f_r}{r^3},$$

$$B = -p + 2\frac{f_{rr}}{r^2} - 2\frac{f_r}{r^3},$$

$$C = 2\frac{f_{rrr}}{r^3} - 6\frac{f_{rr}}{r^4} + 6\frac{f_r}{r^5},$$

and

$$f = \begin{cases} \frac{1}{2\pi\lambda^2}(\ln(\frac{1}{r}) - k_0(\lambda r)) & \text{(2D)} \\ \frac{1}{4\pi\lambda^2}(\frac{1}{r} - \frac{1}{r}\mathrm{e}^{-\lambda r}) & \text{(3D)} \end{cases}, \qquad p = \begin{cases} \frac{1}{2\pi}\frac{1}{r^2} & \text{(2D)} \\ \frac{1}{4\pi}\frac{1}{r^4} & \text{(3D)} \end{cases}, \qquad \lambda = \sqrt{\frac{\alpha}{\mu}}.$$

Navier Equation (Elastostatics).

$$-\mu\Delta u - \frac{\mu}{1-2\nu}\nabla\cdot \operatorname{Div} u = 0$$

$$\mathbf{S}(\mathbf{x},\mathbf{y}) = \begin{cases} \frac{1}{\mu}\left(\frac{3-4\nu}{8\pi(1-\nu)}\log(\frac{1}{r}) + \frac{1}{8\pi(1-\nu)}\frac{(\mathbf{r}\otimes\mathbf{r})}{r^2}\right) & \text{(2D)} \\ \frac{1}{\mu}\left(\frac{3-4\nu}{16\pi(1-\nu)}\frac{1}{r} + \frac{1}{16\pi(1-\nu)}\frac{(\mathbf{r}\otimes\mathbf{r})}{r^3}\right) & \text{(3D)} \end{cases},$$

$$\mathbf{D}(\mathbf{x},\mathbf{y}) = \begin{cases} \frac{1-2\nu}{4\pi(1-\nu)}\left(-\frac{((\mathbf{r}\cdot\mathbf{n})\mathbf{I}+\mathbf{n}\otimes\mathbf{r})}{r^2} + \frac{(\mathbf{r}\otimes\mathbf{n})}{r^2} - \frac{2}{1-2\nu}\frac{(\mathbf{r}\cdot\mathbf{n})(\mathbf{r}\otimes\mathbf{r})}{r^4}\right) & \text{(2D)} \\ \frac{1-2\nu}{8\pi(1-\nu)}\left(-\frac{((\mathbf{r}\cdot\mathbf{n})\mathbf{I}+\mathbf{n}\otimes\mathbf{r})}{r^3} + \frac{(\mathbf{r}\otimes\mathbf{n})}{r^3} - \frac{3}{1-2\nu}\frac{(\mathbf{r}\cdot\mathbf{n})(\mathbf{r}\otimes\mathbf{r})}{r^5}\right) & \text{(3D)} \end{cases}.$$

Modified Navier Equation (Elastodynamics).

$$\alpha u - \mu\Delta u - \frac{\mu}{1-2\nu}\nabla\cdot \operatorname{Div} u = 0$$

$$\mathbf{S}(\mathbf{x},\mathbf{y}) = \frac{1}{\mu}\left(G\mathbf{I} + H(\mathbf{r}\otimes\mathbf{r})\right),$$

$$\mathbf{D}(\mathbf{x},\mathbf{y}) = A\left((\mathbf{r}\cdot\mathbf{n})\mathbf{I} + \mathbf{n}\otimes\mathbf{r}\right) + B(\mathbf{r}\otimes\mathbf{n}) + C(\mathbf{r}\cdot\mathbf{n})(\mathbf{r}\otimes\mathbf{r}),$$

where

$$G = \eta^2 f - f_{rr} + (\beta+1-d)\frac{f_r}{r},$$

$$H = \beta\frac{f_{rr}}{r^2} - \beta\frac{f_r}{r^3},$$

$$A = -\frac{1}{r}f_{rrr} + \frac{2\beta+1-d}{r^2}f_{rr} + (\frac{\eta^2}{r} - \frac{2\beta+1-d}{r^3})f_r,$$

$$B = \frac{\gamma(\beta-1)}{r}f_{rrr} + \frac{2\beta+\gamma(\beta-1)(d-1)}{r^2}f_{rr} + (\frac{\gamma\eta^2}{r} - \frac{2\beta+\gamma(\beta-1)(d-1)}{r^3})f_r,$$

$$C = \frac{2\beta}{r^3}f_{rrr} - \frac{6\beta}{r^4}f_{rr} + \frac{6\beta}{r^5}f_r,$$

and

$$f = \begin{cases} \frac{1}{2\pi(\lambda^2-\eta^2)}(k_0(\eta r) - k_0(\lambda r)) & \text{(2D)} \\ \frac{1}{4\pi(\lambda^2-\eta^2)}(\frac{1}{r}\mathrm{e}^{-\eta r} - \frac{1}{r}\mathrm{e}^{-\lambda r}) & \text{(3D)} \end{cases},$$

$$\lambda = \sqrt{\frac{\alpha}{\mu}}, \qquad \eta = \sqrt{\frac{1-2\nu}{2(1-\nu)}\cdot\frac{\alpha}{\mu}}, \qquad \beta = \frac{1}{2(1-\nu)}, \qquad \gamma = \frac{2\nu}{1-2\nu}.$$

APPENDIX B: COEFFICIENTS OF FAST MULTIPOLE METHOD

We give the coefficients of the FMM for 2D single layer Laplacian. Figure 1 illustrates the relative positions of the symbols used in the following equations.

Multipole expansion. Suppose the m source densities $\{\phi_j\}$ located at $\{z_j\}$, with $|z_j - z_C| < r$, then for any $|z - z_C| > R$, the induced potential $u(z)$ can be approximated by:

$$u(z) = a_0 \log(z - z_C) + \sum_{k=1}^{p} \frac{a_k}{(z - z_C)^k} + \mathcal{O}(\frac{r^p}{R^p}) \tag{13}$$

where $\{a_k, 0 \le k \le p\}$ satisfies

$$a_0 = \sum_{j=1}^{m} \phi_j \quad \text{and} \quad a_k = \sum_{j=1}^{m} \frac{-\phi_i (z_i - z_C)^k}{k}.$$

Local expansion. Suppose the m source densities $\{\phi_j\}$ located at $\{z_j\}$, with $|z_j - z_C| > R$, then for any $|z - z_C| < r$, the induced potential $u(z)$ can be approximated by:

$$u(z) = \sum_{k=0}^{p} c_k (z - z_C)^k + \mathcal{O}(\frac{r^p}{R^p}) \tag{14}$$

where $\{c_k, 0 \le k \le p\}$ satisfies

$$c_0 = \sum_{j=1}^{m} \phi_j \log(z_C - z_j) \quad \text{and} \quad c_l = \sum_{j=1}^{m} \frac{-\phi_j}{l \cdot (z_j - z_C)^l}.$$

M2M translation. Suppose z_C is the center of a box and z_M is the center of its parent. Suppose further $\{a_k\}$ is the multipole expansion at z_C, then the multipole expansion at z_M can be written as:

$$u(z) = b_0 \log(z - z_M) + \sum_{l=1}^{p} \frac{b_l}{(z - z_M)^l} + \mathcal{O}(\epsilon), \tag{15}$$

where $\{b_k, 0 \le k \le p\}$ satisfies

$$b_0 = a_0 \quad \text{and} \quad b_l = -\frac{a_0 (z_C - z_M)^l}{l} + \sum_{k=1}^{l} a_k (z_C - z_M)^{l-k} \binom{l-1}{k-1}.$$

M2L translation. Suppose z_M and z_L are the centers of two non-adjacent boxes on the same level, $\{b_k\}$ is multipole expansion at z_M. Then the local exp-anion at z_L transformed from $\{b_k\}$ is:

$$u(z) = \sum_{l=0}^{p} c_l (z - z_L)^l + \mathcal{O}(\epsilon), \tag{16}$$

where $\{c_k, 0 \le k \le p\}$ satisfies

$$\begin{aligned} c_0 &= b_0 \log(z_L - z_M) + \sum_{k=1}^{p} \frac{b_k}{(z_M - z_L)^k} (-1)^k \\ c_l &= -\frac{b_0}{l \cdot (z_M - z_L)^l} + \frac{1}{(z_M - z_L)^l} \sum_{k=1}^{p} \frac{b_k}{(z_M - z_L)^k} \binom{l+k-1}{k-1} (-1)^k. \end{aligned}$$

L2L translation. Suppose z_T is the center of a box and z_L the center of its parent. Suppose further $\{\mathbf{c}_l\}$ is the local expansion at $\mathbf{z}_L$, then the local expansion at z_T can be written as

$$u(z) = \sum_{l=0}^{p} d_l (z - z_T)^l + \mathcal{O}(\epsilon), \tag{17}$$

where $\{d_k, 0 \le k \le p\}$ satisfies

$$d_l = \sum_{k=l}^{p} c_k \binom{k}{l} (z_T - z_L)^{(k-l)}.$$

REFERENCES

[1] Christopher R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM Journal on Scientific and Statistical Computing*, 13(4):923–947, 1992.

[2] Josh Barnes and Piet Hut. A hirerachical O(N logN) force-calculation algorithm. *Nature*, 324(4):446–449, December 1986.

[3] C. Leonard Berman. Grid-multipole calculations. *SIAM Journal on Scientific Computing*, 16(5):1082–1091, 1995.

[4] George Biros, Lexing Ying, and Denis Zorin. The embedded boundary integral method for the stokes equations. Technical Report TR2003-837, Courant Institute, New York University, 2002. http://www.cs.nyu.edu/csweb/Research/TechReports/TR2003-837/TR2003-837.pdf.

[5] George Biros, Lexing Ying, and Denis Zorin. The embedded boundary integral method for the unsteady incompressible navier-stokes equations. Technical Report TR2003-838, Courant Institute, New York University, 2002. http://www.cs.nyu.edu/csweb/Research/TechReports/TR2003-838/TR2003-838.pdf.

[6] Oscar P. Bruno and Leonid A. Kunyansky. A fast, high-order algorithm for the solution of surface scattering problems: Basic implementation, tests, and applications. *Journal of Computational Physics*, 169:80–110, 2001.

[7] H. Cheng, Leslie Greengard, and Vladimir Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155:468–498, 1999.

[8] Yuhong Fu et al. A fast solution for three-dimensional many-particle problems of linear elasticity. *International Journal for Numerical Methods in Engineering*, 42:1215–1229, 1998.

[9] Yuhong Fu and Gregory J. Rodin. Fast solution method for three-dimensional Stokesian many-particle problems. *Communications in Numerical Methods in Engineering*, 16:145–149, 2000.

[10] Zydrunas Gimbutas and Fladimir Rokhlin. A generalized fast mulipole method for nonoscillatory kernels. *SIAM Journal on Scientific Computing*, 24(3):796–817, 2002.

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

[12] Leslie Greengard and Jingfang Huang. A new version of the fast multipole method for screened Coulomb interactions in three dimensions. *Journal of Computational Physics*, 180:642–658, 2002.

[13] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[14] Leslie Greengard and Vladimir Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, pages 229–269, 1997.

[15] Ken ichi Yoshida, Naoshi Nishimura, and Shoichi Kobayashi. Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *International Journal for Numerical Methods in Engineering*, 50(3):525–547, 2001.

[16] Sharad Kapur and David E. Long. IES_3: Efficient electrostatic and electromagnetic simulation. *IEEE Computational Science and Engineering*, 5(4):60–67, 1998.

[17] Sharad Kapur and Jinsong Zhao. A fast method of moments solver for efficient parameter extraction of MCMs. In *Design Automation Conference*, pages 141–146, 1997.

[18] Rainer Kress. *Linear Integral Equations*. Applied Mathematical Sciences. Springer, 1999.

[19] Junichiro Makino. Yet another fast multipole method without multipoles–pseudoparticle multipole method. *Journal of Computational Physics*, 151:910–920, 1999.

[20] Joel R. Phillips and Jacob K. White. A precorrected-FFT method for electorstatic analysis of complicated 3-d structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(10):1059–1072, 1997.

[21] V. Popov and Henry Power. An O(N) taylor sereis multipole boundary element method for three-dimensional elasticity problems. *Engineering Analysis with Boundary Elements*, 25:7–18, 2001.

[22] Vladimir Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60:187–207, 1983.

[23] Norman Yarvin and Vladimir Rokhlin. Generalized gaussian quadratures and singular value decompositions of integral operators. *SIAM Journal on Scientific Computing*, 20(2):699–718, 1998.

[24] Norman Yarvin and Vladimir Rokhlin. An improved fast multipole algorithm for potential fields on the line. *SIAM Journal on Numerical Analysis*, pages 629–666, 1999.