A Free-Space Adaptive FMM-Based PDE Solver in Three Dimensions

M.H. Langston[†] L. Greengard D. Zorin

July 2011

Abstract

We present a kernel-independent, adaptive fast multipole method (FMM) of arbitrary order accuracy for solving elliptic PDEs in three dimensions with radiation and periodic boundary conditions. The algorithm requires only the ability to evaluate the Green's function for the governing equation and a representation of the source distribution (the right-hand side) that can be evaluated at arbitrary points. The performance is accelerated in three ways. First, we construct a piecewise polynomial approximation of the right-hand side and compute far-field expansions in the FMM from the coefficients of this approximation. Second, we precompute tables of quadratures to handle the near-field interactions on adaptive octree data structures, keeping the total storage requirements in check through the exploitation of symmetries. Third, we employ shared-memory parallelization methods and load-balancing techniques to accelerate the major algorithmic loops of the FMM. We present numerical examples for the Laplace, modified Helmholtz and Stokes equations.

1 Introduction

Many problems in scientific computing call for the efficient solution to linear partial differential equations with constant coefficients. On regular grids with separable Dirichlet, Neumann or periodic boundary conditions, such equations can be solved using fast, direct methods. For free-space boundary conditions and highly nonuniform source distributions defined on adaptive and/or unstructured grids, alternative approaches are necessary. We describe a *direct* high-order adaptive solver for inhomogeneous linear constant-coefficient PDEs in three dimensions with decay conditions at infinity. A typical case is the Poisson equation

$$-\Delta u = g, \;\; supp(g) \subset \Omega, \tag{1}$$

where Ω is a bounded domain in $\mathbf{R}^3$, and $u(\mathbf{x}) = O(1/|\mathbf{x}|)$ as $|\mathbf{x}|$ goes to infinity. Our solver uses a kernel-independent fast multipole method (FMM) [60, 61] which can be applied to any PDE, for which a free-space Green's function evaluation routine is provided. It handles highly nonuniform sources in an efficient manner, using an adaptive approximation of the right-hand side in (1). The structure of the solver allows for natural integration with FMM-based boundary integral equation techniques, leading to the construction of an adaptive kernel-independent solver for inhomogeneous PDEs in complex geometries, which will be described in a companion paper.

Related work. For regular grids in separable coordinate systems (rectangles, disks, spheres, etc.), fast methods for constant-coefficient second order PDEs are well-established [14, 15]. These methods generally rely on cyclic reduction and/or fast Fourier transforms (FFTs) to achieve nearly linear scaling. For many problems, however, adaptive meshes resulting from adaptive mesh refinement (AMR) strategies are essential [2, 7, 48], and existing solvers typically rely on domain decomposition strategies [22] or multigrid acceleration [17, 37, 39, 43]. For complex geometries, unstructured grid generation techniques are often used (e.g., [44]). In such cases, both the grid generation process and the solution

[†]Courant Institute, New York University, New York 10012. Email: {harper,greengard,dzorin}@cims.nyu.edu

The work of M.H.L was supported by the U.S. Department of Energy CPES contract; the work of L.G. was supported in part by the U.S. Department of Energy under contract DEFG0288ER25053; the work of D.Z. was supported by the U.S. Department of Energy CPES contract and the National Science Foundation contract DMS-0612624.

of the resulting linear systems can be computationally expensive. The lack of regularity in the data structures adds complexities in parallelization as well [1, 17].

A more recent class of methods combines ideas from potential theory with finite difference methods. In [26], fast direct solvers were used on a sequence of refined grids with boundary conditions inherited from the coarser levels. This results in discontinuities at coarse-fine interfaces, which are corrected using a second pass through the grid hierarchy. In [4], the method of local corrections (MLC) [3] was combined with multigrid methods to solve the Poisson equation on a hierarchy of nested grids. The fastest free-space Poisson solver for three-dimensional problems of which we are aware is described in [46]. It first solves local Poisson problems on fine grids using FFT-based techniques and then couples together the solutions on coarser grids using MLC. This approach was shown to be very effective in parallel, with good scaling up to 1024 processors (A similar two-dimensional scheme is described in [29]). For unstructured meshes, the preceding methods do not apply without significant modification and most fast solvers are based on iterative methods using multigrid or domain decomposition acceleration [12, 13, 16].

In this paper, we concentrate on the integral equation (or, more precisely, the integral transform) viewpoint. Rather than solving (1), for example, we simply compute

$$u(\mathbf{x}) = \frac{1}{4\pi} \int_{\mathbf{R}^3} \frac{1}{|\mathbf{x}-\mathbf{y}|} g(\mathbf{y}) d\mathbf{y}. \tag{2}$$

Among the advantages of this approach is the increase in precision in computing derivatives. In PDE-based methods, if first or second derivatives of the solution are needed, accuracy tends to degrade due to the need for numerical differentiation. Instead, we can differentiate the kernel in (2) and compute derivatives from their integral representation as well. Other advantages are that free-space radiation conditions are automatically satisfied, we can obtain simple *a priori* error estimates, and high order accuracy is straightforward to achieve. However, the computational complexity of a naïve implementation is high: computing the solution u at N points $\mathbf{x}$ given N discretization points $\mathbf{y}$ requires $O(N^2)$ work. There have been a number of methods proposed to overcome this barrier. These include panel-clustering techniques [10, 36], hierarchical matrices ($\mathcal{H}$,$\mathcal{H}^2$-matrices) [9, 34, 35], the Barnes-Hut method [5], and the Fast Multipole Method (FMM) [20, 32, 24, 50] originally designed for gravitational/Coulomb interactions. These schemes all achieve linear $O(N)$ or nearly linear $O(N \log N)$ scaling. Most of these methods fall into the class of what are often called "tree codes" because they separate near and far-field interactions on a hierarchy of spatial scales using quadtree (2D) or octree (3D) data structures. Because it can achieve arbitrary precision at modest cost with straightforward error estimates, we concentrate on the FMM in the present setting. The classical FMM is kernel-specific and relies on detailed separation of variables solutions of the governing PDE. While the FMM references above considered the Laplace equation, the Helmholtz equation was subsequently treated in [51]. A three-dimensional version effective for all frequencies (and additional references) can be found in [19]. The modified Helmholtz equation was discussed in [11, 27], and the biharmonic equation in [28, 33, 56]. The Stokes equations are somewhat exceptional, since they can be handled by a sequence of calls to the original (Coulomb) FMM [54, 58]. An attractive alternative that avoids much of the detailed analytic work of these methods is the kernel-independent approach of [60, 61]. In this approach, expansions in special functions are replaced with equivalent source densities. The result is that the same numerical apparatus can be used for a variety of PDEs, and the user need only supply a subroutine for the evaluation of the relevant Green's function.

While the bulk of the work on FMMs over the last two decades has concentrated on particle interactions or the acceleration of boundary integral equation methods, there has been some work on solving inhomogeneous PDEs. One option is to couple the FMM with finite difference methods to allow for fast solvers in complex geometries [45, 47, 59]. While this is a significant improvement in terms of range of applicability over classical fast solvers, these methods require a regular volume mesh on which is superimposed an irregular boundary. Adaptive FMMs for volume source distributions in two dimensions were described in [21, 23, 29]. The present paper extends these two-dimensional schemes to three dimensions, incorporates them into kernel-independent FMMs, and introduces several new performance optimizations. The result is an efficient, adaptive method that is capable of computing volume integrals in three dimensions for a broad variety of PDE kernels.

Before turning to the method itself, we should also note that there has been a significant body of work in the quantum chemistry community on accelerating volume integral calculations using the FMM, where collections of Gaussians are typically used to describe the charge distribution [52, 57]. These are Poisson problems in free-space but with a different approach to defining the right-hand side.

2 Equations and Kernels

Given a linear, constant-coefficient PDE

$$\mathcal{L}(u)(\mathbf{x}) = g(\mathbf{x}), \tag{3}$$

classical mathematical methods can be used to compute the corresponding Green's function $K(\mathbf{x}, \mathbf{y})$ in free space such that

$$u(\mathbf{x}) = \int_\Omega K(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) d\mathbf{y}\,, \tag{4}$$

where Ω is the support of g. $K(\mathbf{x}, \mathbf{y})$ is in general weakly singular; assuming $g(\mathbf{x})$ is given at N points and $u(\mathbf{x})$ is desired at N points, the non-local character of the integral representation, as indicated above, would lead to an $O(N^2)$ solution procedure. Thus, we need both a suitable quadrature approach and a fast algorithm for (4) to yield a useful numerical technique. Assuming this is achieved, a number of advantages follow. First, no linear system needs to be solved. Second, adaptivity is achieved through the approximation of the right-hand side. Third, as mentioned in the previous section, derivatives can be computed without loss of precision (There is some loss in accuracy for derivatives of order greater than two, since at that point the integral operator becomes hypersingular and some catastrophic cancellation cannot be avoided). Finally, we have simple *a priori* error estimates. To see this, let $\hat{g}(\mathbf{x})$ be the approximation to $g(\mathbf{x})$ and let $\hat{\mathbf{Q}}[f](\mathbf{x})$ denote the quadrature approximation of $\int_\Omega K(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) dy$. Assuming the near field is computed exactly, the quadrature error satisfies an estimate of the form

$$\left| \hat{\mathbf{Q}}[f](\mathbf{x}) - \int_\Omega K(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) dy \right| \leq \epsilon \|f\|_1,$$

where ϵ is the approximation error in the FMM. ϵ is, in turn, controlled by the parameter p that determines the number of discretization points used for equivalent densities, as described in Section 4.1 and reference [60].

To estimate the total error, let us assume $\hat{g}(\mathbf{x})$ is a k^{th}-order polynomial approximation of the right hand-side

$$\hat{g}(\mathbf{x}) - g(\mathbf{x}) \leq \delta = O(h^k),$$

and that

$$\hat{u}(\mathbf{x}) = \hat{\mathbf{Q}}[\hat{g}](\mathbf{x})\,. \tag{5}$$

Then

$$\begin{aligned} e(\mathbf{x}) = u(\mathbf{x}) - \hat{u}(\mathbf{x}) &= \int_\Omega K(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) d\mathbf{y} - \int_\Omega \hat{\mathbf{Q}}[\hat{g}](\mathbf{x}) \\ &\leq \int_\Omega K(\mathbf{x}, \mathbf{y}) [g(\mathbf{y}) - \hat{g}(\mathbf{y})] d\mathbf{y} + \left| \int_\Omega K(\mathbf{x}, \mathbf{y}) \hat{g}(\mathbf{y}) d\mathbf{y} - \hat{\mathbf{Q}}[\hat{g}](\mathbf{x}) \right| \\ &\leq C_1 \|g(\mathbf{y}) - \hat{g}(\mathbf{y})\|_\infty + \|\hat{g}(\mathbf{y})\|_1 \, \epsilon, \quad \text{where } C_1 = \max_x \int_\Omega |K(\mathbf{x}, \mathbf{y})| d\mathbf{y} \quad \leq C_1 \delta + \|\hat{g}(\mathbf{y})\|_1 \, \epsilon. \end{aligned} \tag{6}$$

The estimate above is much sharper than one typically obtained when discretizing the PDE itself, where the order of accuracy is determined by high derivatives of the solution. Here, it depends only on the quality of the approximation of the right-hand side ($\delta = O(h^k)$) and the FMM tolerance (ϵ). Note that the constant C_1 is a bounded quantity determined by the volume of Ω with no dependence on the data. If ϵ is chosen to be of the same order as δ, the scheme is formally k^{th}-order accurate. In practice, it is convenient to decouple the right-hand side approximation error from the FMM tolerance, as above, permitting the user to control them independently.

The principal drawback with the integral formulation is that, when implemented naïvely, the complexity of the approach is quadratic in the number of sample points. FMM algorithms overcome this computational barrier by making systematic use of the smoothness of distant interactions on a hierarchy of spatial scales [6, 23, 30]. The kernel-independent versions of the FMM [60, 61] are particularly useful because of their generality; they make it possible to compute solutions of the form (4) for any (non-oscillatory) elliptic PDE, provided only a module which evaluates the kernel.

After describing the details of the approach, we demonstrate its performance for the Poisson equation (7), the modified Helmholtz equation (8), and the Stokes equations (9):

$$-\Delta u(\mathbf{x}) = g(\mathbf{x}), \tag{7}$$

$$\alpha u(\mathbf{x}) - \Delta u(\mathbf{x}) = g(\mathbf{x}),\ \alpha > 0, \text{ and} \tag{8}$$

$$\nabla p(\mathbf{x}) - \mu \Delta \mathbf{u}(\mathbf{x}) = \mathbf{g}(\mathbf{x}),\ \nabla \cdot \mathbf{u}(\mathbf{x}) = 0,\ \mu > 0. \tag{9}$$

Defining $\mathbf{r} = \mathbf{x} - \mathbf{y}$ and $r = ||\mathbf{r}||$, the corresponding kernels in three dimensions are given by

$$K(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi r}, \tag{10}$$

$$K(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi r} e^{-\sqrt{\alpha} r}, \text{ and} \tag{11}$$

$$K(\mathbf{x}, \mathbf{y}) = \frac{1}{8\pi\mu} \left(\frac{1}{r} I + \frac{\mathbf{r} \otimes \mathbf{r}}{r^3} \right), \text{ respectively.} \tag{12}$$

The classical FMM is reviewed briefly in section 3, the kernel-independent method is described in section 4, and symmetries for optimizing storage are discussed in section 5. Numerical experiments are presented in section 6 as well as a brief discussion on extending our method to periodic boundary conditions or including a singular source component along with a smooth background force. We provide additional error analysis in section 8 as well as a brief summary of how the method is optimized using OpenMP and load-balancing techniques to achieve near-linear strong scaling.

3 Analytic Fast Multipole Method

We briefly review the structure of the original two-dimensional FMM for the case of particle interactions [30]. Given a set of N_{src} charges of strength $g(\mathbf{y}_i)$ at locations $(\mathbf{y})_i$, the FMM was designed to compute the induced potentials u_j at N_{trg} target locations, $\mathbf{x}_j$,

$$u_j = u(\mathbf{x}_j) = \sum_{i=1}^{N_{src}} K(\mathbf{x}_j, \mathbf{y}_i) g(\mathbf{y}_i)\ ,\ j = 1, \ldots, N_{trg}, \tag{13}$$

where $K(\mathbf{x}, \mathbf{y}) = -\log |\{\mathbf{x} - \mathbf{y}\}| / 2\pi$. For $N_{src} \approx N_{trg} = N$, the FMM decreases the computational cost from $O(N^2)$ to $O(N)$ for fixed user-prescribed accuracy by introducing a hierarchical partition (represented by a tree data structure, T) of a regular bounding domain D and two series expansions for each box at each level of the hierarchy. More precisely, the root of T is associated with the entire box D and defined to be at *level* $\ell = 0$. Level $\ell + 1$ is obtained from level ℓ recursively, dividing each sub-domain at level ℓ into four equal-sized children. For a regular box B of width H, B's *near field*, $\mathcal{N}^B$, is defined as the set of all boxes in D that lie within a box centered at B of width $3H$. The *neighbor list*, L_N^B, is defined as the set of boxes in $\mathcal{N}^B$ which share a vertex with B. In the non-adaptive case, $L_N^B = \mathcal{N}^B$. The *far field*, $\mathcal{F}^B$, is the complement of the near field: $\mathcal{F}^B = D \setminus \mathcal{N}^B$. Finally, the *interaction list*, L_I^B, is the set of children of B's parent's neighbors that are not neighbors themselves. Thus, $L_I^B \subseteq \mathcal{F}^B$. The depth of T is chosen so that the smallest boxes (leaves in T) contain no more than some fixed number of points, say s. We first consider uniformly refined trees, where all leaves T are at the same level. Note that the total number of boxes in a 2D quadtree is bounded by $4N/3s$ (and $8N/3s$ in a 3D octree). Thus, if the workload per box is constant, the net algorithm has $O(N)$ complexity.

A *local expansion* is used to represent within each box B the influence of all sources in the far field of B. A *multipole expansion* about the center of B is used to represent the influence of sources *inside* B on boxes in the far field $\mathcal{F}^B$ [30].

The FMM computes the total field at a target point in leaf box B as the sum of (a) the field due to the source points contained in the boxes of the neighbor list L_N^B and (b) the contribution from sources in the far field $\mathcal{F}^B$. The contributions from source points inside the boxes of L_N^B are computed directly using (13), while the contributions from $\mathcal{F}^B$ are obtained by evaluating the local expansion of box B at the target. The essential task of the FMM is the *construction* of the local expansions in a hierarchical manner. This takes place in two steps.

The upward pass. This pass begins at the finest level of the tree data structure, converting charge strengths at source points into multipole expansions for each leaf box; this computation is carried out by the *source-to-multipole* (S2M) operator. Multipole expansions for each non-leaf box B at each coarser level are obtained recursively. More precisely, the multipole expansions for the four children of B are merged into a single expansion about B's center using the *multipole-to-multipole* (M2M) operator.

The downward pass. For each box B, starting at the coarsest level, the local expansion of $\mathcal{F}^B$ is obtained by shifting the local expansion of B's parent to the center of B using the *local-to-local* (L2L) operator and by mapping multipole expansions centered at each box in L_I^B to B's local expansion using the *multipole-to-local* (M2L) operator. For leaf box B, local expansions are then evaluated at each target point using the *local to target* (L2T) operator.

Summary. FMM uses $S2M$, $M2M$, $M2L$, $L2L$ and $L2T$ linear operators: $M2M$ and $L2L$ operators are determined uniquely by the relative position of a box and its parent; each $M2L$ operator is determined by the relative position of a box in the interaction list; $S2M$ and $L2T$ operators depend on source and target point locations and can be different for each (finest level) box. Figure 1 illustrates the data flow involved in the $M2M$, $M2L$ and $L2L$ operators.

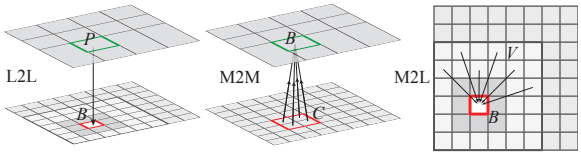


Figure 1: Boxes used by M2M, L2L and M2L operators. For box B at level ℓ, P in the $L2L$ operator represents the parent of B at level $\ell - 1$, and in the $M2M$ operator, C represents the children of B at level $\ell + 1$. Boxes labeled V in the $M2L$ operator reside in L_I^B.

For the Laplace kernel in three dimensions, far-field expansions are represented using a mixture of spherical harmonics [31] and plane-wave representations [32].

We turn now to the kernel-independent approach [60, 61] in order to design a volume integral FMM in three dimensions that can handle a broad class of PDEs.

4 3D Kernel-Independent FMM Volume Integral Solver

Given an octree T for our 3D bounding domain D, let $D = \sum\{B_i\}$, $i = 1 \ldots M$ be the set of leaf boxes resulting from hierarchical subdivision. For a single-layer kernel K, we compute the integral (4) at some point $\mathbf{x}$ as

$$u(\mathbf{x}) = \sum_{i=1}^{M} K[B_i, g^{B_i}](\mathbf{x}), \tag{14}$$

where $K[B, g^B](\mathbf{x}) = \int_B K(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) d\mathbf{y}$, and g^B represents the restriction of the source distribution to the box B.

The principal difference between the approach of this paper and the analytic FMM for point sources is that we use *sampled equivalent densities* instead of classical special functions and series expansions to account for far-field interactions, as in [60, 61]. This requires only a *black-box* kernel evaluation routine and allows for a kernel-independent implementation. A second difference between the approach of this paper and prior kernel-independent FMM schemes is that we are dealing with a continuous source distribution rather than a collection of point-like particles. To extend the method of [60, 61] to this setting, we use polynomial basis functions to approximate the source distribution g on each leaf box, following the two-dimensional approach of [21, 29, 23]. More precisely, we assume that the input source is given on each leaf box B by a polynomial g^B of degree $k + 1$ with coefficients γ^B,

$$g^B = \sum_{j=1}^{N_k} \gamma_j^B \beta_j \left(2^\ell (\mathbf{x} - \mathbf{c}_B)\right), \tag{15}$$

where β_j are polynomial basis functions, ℓ is the depth of the box B ($\ell = 0$ at the root of T), and $\mathbf{c}_B$ is its center. We use monomials for low-order accuracy and tensor-product Chebyshev polynomials for higher-order accuracy. The number of coefficients is $N_k = k(k+1)(k+2)/6$ for each scalar source function g. We describe an interpolation scheme to convert a set of source values defined on a grid of sample points to a polynomial representation in Section 4.6. As output, our algorithm can return either point values of the potential at each target point or a polynomial approximation of the potential on each leaf box (which can then be evaluated at arbitrary locations).

To simplify the exposition, we present our algorithm first for a uniformly refined octree of depth ℓ and then discuss the changes necessary for the adaptive octree case separately. The final algorithmic steps are outlined in section 4.8, and we briefly discuss how the major loops are optimized for shared-memory parallelization in section 8.4.

4.1 Equivalent Densities

The kernel-independent approach to translation operators is based on the following idea. For kernel K, suppose we have an arbitrary (smooth or non-smooth) source distribution g_s in a volume Ω_s with surface Γ_s. Let Γ_t denote an auxiliary surface in the exterior of Γ_s, and let Γ_{check} denote yet another auxiliary surface in the exterior of Γ_t. Finally, let E denote the exterior of Γ_{check}. We will compute a charge density ϕ_t on Γ_t such that the potentials $K[\Omega_s, g_s]$ and $K[\Gamma_t, \phi_t]$ coincide in E. This is always possible if the exterior Dirichlet problem on Γ_t has a unique solution and the exterior field can be represented in terms of a single layer potential[1]

Remark For some problems, such as the Helmholtz equation, a combination of single and double layer sources may be required because of non-physical resonances in the single layer representation, but it is generally sufficient for non-oscillatory kernels (cf. [40] for the Poisson equation, [41] for the Stokes equations).

Our goal is to use $K[\Gamma_t, \phi_t]$ to represent the far-field instead of a multipole expansion. For this, we let Γ_{check} approximate the outer boundary of the neighbor list L_N^B and solve a Fredholm integral equation of the first kind for ϕ_t,

$$K[\Gamma_t, \phi_t](\mathbf{x}) = K[\Omega_s, g_s](\mathbf{x}), \quad \text{for all } \mathbf{x} \in \Gamma_{check}. \tag{16}$$

Having matched the field on Γ_{check}, the fields will match in the exterior E (with precise estimates depending on the specific kernel). We refer to Γ_t as an *equivalent surface* with *equivalent density* ϕ_t, and Γ_{check} as a *check surface*. In the case when the original density is concentrated on the surface Γ_s, then (16) can be written as

$$K[\Gamma_t, \phi_t](\mathbf{x}) = K[\Gamma_s, \phi_s](\mathbf{x}), \quad \text{for all } \mathbf{x} \in \mathbf{x}_t. \tag{17}$$

We match the field created by charges *outside* the near neighbors of a box by a discretized layer potential defined on a surface enclosing the box, and a different equivalent density will be used to replace the local expansion. The number of samples used to represent the equivalent density is the analog of the number of expansion terms in a classical FMM. Γ_t and Γ_{check} are cubic surfaces, uniformly sampled at p locations. In discretized form, (17) can be written as

$$\mathbf{K}^{\Gamma_t, \mathbf{x}_t} \phi_t = \mathbf{K}^{\Gamma_s, \mathbf{x}_t} \phi_s, \tag{18}$$

where ϕ_s and ϕ_t are vectors of point-sampled densities, and $\mathbf{K}^{a,b}$ are matrices with entries given by $\mathbf{K}^{a,b}_{ij} = K(a_i, b_j)$ for sample points a_i and a_j on surfaces a and b. For known ϕ_s and solving for ϕ_t, (18) is a discretization of a Fredholm equation of the first kind. For large p, linear systems may be poorly conditioned; in such cases, we choose to utilize Tikhonov regularization methods [40] to invert $\mathbf{K}^{\Gamma_t, \mathbf{x}_t}$. We discuss this approach and its accuracy in sections 8.1 and 8.3.

Kernel invariance and matrix precomputation. For all equations we consider, the kernels are invariant with respect to a rigid transformation $\mathcal{T}$: for scalar kernels, $K(\mathcal{T}\mathbf{x}, \mathcal{T}\mathbf{y}) = \mathcal{T}K(\mathbf{x}, \mathbf{y})$, and for matrix kernels, $K(\mathcal{T}\mathbf{x}, \mathcal{T}\mathbf{y}) = \mathcal{T}K(\mathbf{x}, \mathbf{y})\mathcal{T}^T$. Hence, all matrices $\mathbf{K}$ need to be computed only once for each class of pairs of equivalent surfaces, closed with respect to a specific $\mathcal{T}$. Furthermore, many kernels are *homogeneous*: $\forall c > 0$, $\exists$ *scaling exponent* $r \neq 0$ such that $K(c\mathbf{x}, c\mathbf{y}) = c^r K(\mathbf{x}, \mathbf{y})$, further reducing the number of classes of surface pairs requiring separate matrices. We consider optimizations due to invariance for each translation operator in the next sections, assuming scalar kernels for simplicity, although our implementation can handle matrix kernels.

[1] For some kernels (Stokes), a low-dimensional nullspace may need to be eliminated.

4.2 Upward Pass

For the upward pass, recall now that for each leaf box, the *sources* are polynomials approximating the source distribution. For consistency with the FMM summary above, we use S, M, etc. in describing translation operator names.

Source to Multipole (S2M) translations. For each leaf box B, we choose $\mathbf{y}^{B,u}$, the *upward equivalent surface*, and $\mathbf{x}^{B,u}$, the *upward check surface*, as in [60]. Equation (16) for upward equivalent density $\phi^{B,u}$ in this case becomes

$$K[\mathbf{y}^{B,u}, \phi^{B,u}](\mathbf{x}) = K[B, g^B](\mathbf{x}), \quad \text{and} \tag{19}$$

$$K[B, g^B](\mathbf{x}) \approx \sum_{j=1}^{N_k} \gamma_j^B F_j^B(\mathbf{x}), \quad \text{where} \tag{20}$$

$$F_j^B(\mathbf{x}) = \int_B \beta_j \left(2^\ell(\mathbf{y} - \mathbf{c}_B)\right) K(\mathbf{x}, \mathbf{y}) d\mathbf{y}, \quad \text{for } \mathbf{x} \in \mathbf{x}^{B,u}. \tag{21}$$

By translation invariance, $F_j^B(\mathbf{x})$ depends only on the choice of β_j and level ℓ of B. To evaluate the integrals in (21), we use adaptive Gaussian quadrature [8]. In matrix form, the Nyström discretization of (19)-(21) at p sample points, $\boldsymbol{\phi}^{B,u}$, on $\mathbf{y}^{B,u}$ yields

$$\mathbf{K}_{S2M}^B \boldsymbol{\phi}^{B,u} = \mathbf{F}_{S2M}^B \gamma^B, \tag{22}$$

where $\mathbf{F}_{S2M}^B$ is the matrix of precomputed weights (21) and $\mathbf{K}_{S2M}^B$ is the matrix with entries $K(\mathbf{x}_i, \mathbf{y}_j)$, $i = 1 \dots p$, $j = 1 \dots p$. Solving for $\boldsymbol{\phi}^{B,u}$,

$$\boldsymbol{\phi}^{B,u} = (\mathbf{K}_{S2M}^B)^{-1} \mathbf{F}_{S2M}^B \gamma^B = \mathbf{T}_{S2M}^B \gamma^B. \tag{23}$$

For a uniformly-refined tree, $\mathbf{T}_{S2M}^B$ depends only on ℓ, so one matrix is computed. Figure 2(a) illustrates the computation of $\phi^{B,u}$ from γ^B.

Multipole to Multipole (M2M) translations. M2M translation operators translate $\phi^{C,u}$ at a child box C to $\phi^{B,u}$ for the parent box B, shown in figure 2(b): For all $\mathbf{x} \in \mathbf{x}^{B,u}$,

$$K[\mathbf{y}^{B,u}, \phi^{B,u}](\mathbf{x}) = \sum_C K[\mathbf{y}^{C,u}, \phi^{C,u}](\mathbf{x}), \text{ or in matrix form: } \mathbf{K}_{M2M}^{B,B} \boldsymbol{\phi}^{B,u} = \sum_C \mathbf{K}_{M2M}^{C,B} \phi^{C,u}. \tag{24}$$

Similar to the $S2M$ computations, these systems are solved as

$$\boldsymbol{\phi}^{B,u} = \sum_C (\mathbf{K}_{M2M}^{B,B})^{-1} \mathbf{K}_{M2M}^{C,B} \boldsymbol{\phi}^{C,u} = \sum_C \mathbf{T}_{M2M}^{C,B} \phi^{C,u}. \tag{25}$$

For any two children C_1 and C_2, rotation R maps C_1 to C_2; therefore, only *one* $\mathbf{T}_{M2M}^{C,B}$ is computed per level, with the contribution to $\boldsymbol{\phi}^{B,u}$ from any other child obtained by composing this matrix with an appropriate permutation of $\phi^{C,u}$. Further, for *homogeneous* kernels, only one matrix is stored at a single level ℓ and scaled as necessary.

4.3 Downward Pass

In the downward pass *downward equivalent densities* are computed through the $M2L$, $L2L$, and $L2T$ operators.

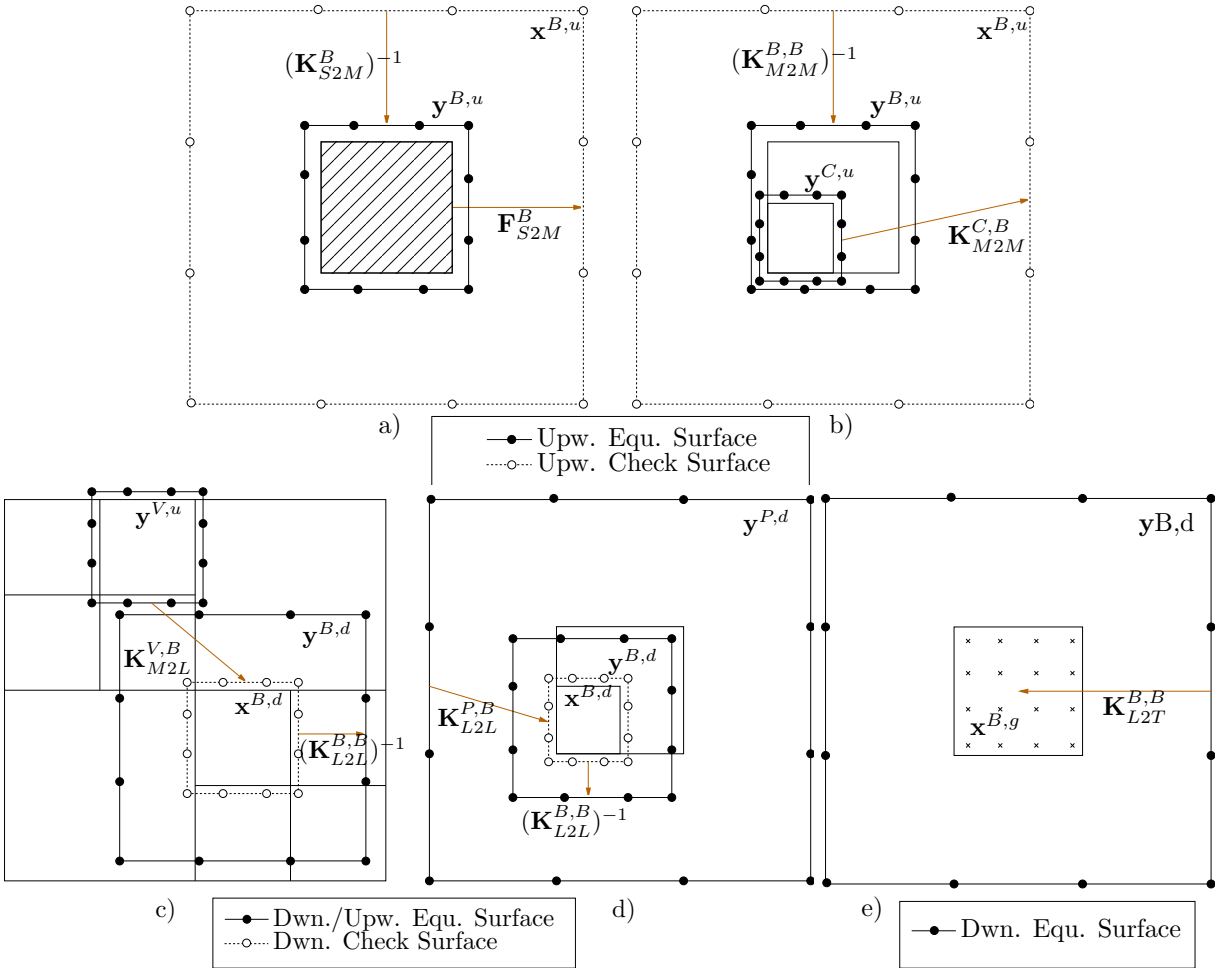


Figure 2: Kernel-independent FMM translation operators. From top-left: a) $S2M$, b) $M2M$, c) $M2L$, d) $L2L$, and e) $L2T$ translations.

Multipole to Local (M2L) translations. For any box B, $M2L$ operators (Figure 2(c)) translate $\phi^{V,u}$, approximating the field of sources inside of $V \in L^B_I$, to a *downward equivalent density* $\phi^{B,d}$. In this case, we seek to induce identical potentials *inside* of B, effectively, swapping upward equivalent and check surfaces to obtain downward equivalent and check surfaces: $\mathbf{y}^{B,d} = \mathbf{x}^{B,u}$ and $\mathbf{x}^{B,d} = \mathbf{y}^{B,u}$. Equation (17) takes the form

$$K[\mathbf{y}^{B,d}, \phi^{B,d}](\mathbf{x}) = \sum_V K[\mathbf{y}^{V,u}, \phi^{V,u}](\mathbf{x}), \quad \text{for all } \mathbf{x} \in \mathbf{x}^{B,d}, \tag{26}$$

where $\phi^{B,d}$ is discretized at p uniformly spaced samples on $\mathbf{y}^{B,d}$. The right-hand side of (26) is computed and stored as a *downward check potential*, $u^{B,d}$ at $\mathbf{x}^{B,d}$, and $\boldsymbol{\phi}^{B,d}$ is recovered after the $L2L$ contribution is added.

$$u^{B,d}_{M2L} = \sum_V \mathbf{K}^{V,B}_{M2L} \boldsymbol{\phi}^{V,u}. \tag{27}$$

We efficiently evaluate $u^{B,d}$ with FFTs by treating densities as being defined on extensions of $\mathbf{y}^{V,u}$ and $\mathbf{x}^{B,d}$ to 3D Cartesian grids with zero values in the interior. This results in $O(p^3/2)$ sample locations, and the computational cost of $O(p^3/2\log(p))$ for evaluation.

Further, there are at most 189 possible locations for $V \in L^B_I$ relative to any particular B; however, using translation and rotation invariance of the kernel as discussed in section 5, we store at most 16 total $\mathbf{K}^{V,B}_{M2L}$ matrices for a

homogeneous kernel.

Local to Local (L2L) translations. Contributions from $\mathcal{F}^B \setminus L_I^B$ are captured through the local field computed for B's parent box, P, using $L2L$ operators (Figure 2(d)). We translate $\phi^{P,d}$ at $\mathbf{y}^{P,d}$ to $\phi^{B,d}$ at $\mathbf{y}^{B,d}$ using the equation

$$K[\mathbf{y}^{B,d}, \phi^{B,d}](\mathbf{x}) = K[\mathbf{y}^{P,d}, \phi^{P,d}](\mathbf{x}), \quad \text{for all } \mathbf{x} \in \mathbf{x}^{B,d}. \tag{28}$$

The right-hand side is computed as a contribution to $u^{B,d}$, so (28) for B at depth ℓ becomes

$$u_{L2L}^{B,d} = \mathbf{K}_{L2L}^{P,B} \boldsymbol{\phi}^{P,d} \text{ such that} \tag{29}$$

$$\boldsymbol{\phi}^{B,d} = (\mathbf{K}_{L2L}^{B,B})^{-1} \left(u_{M2L}^{B,d} + u_{L2L}^{B,d} \right). \tag{30}$$

The precomputation of matrix $\mathbf{K}_{L2L}^{P,B}$ is completely analogous to $\mathbf{K}_{M2M}^{C,B}$, with parent and child swapped.

Local to Grid Target (L2T) translations. For each leaf box B, we evaluate $u^{B,g}$ at grid locations, $\mathbf{x}^{B,g}$. At depth ℓ, $\phi^{B,d}$ accounts for all contributions from $\mathcal{F}^B$ while direct near-field calculations (discussed in detail in section 4.4) account for the contributions from $\mathcal{N}^B$. The far-field potential is computed using $L2T$ operators (Figure 2(e)).

$$u(\mathbf{x}) = K[\mathbf{y}^{B,d}, \boldsymbol{\phi}^{B,d}](\mathbf{x}), \mathbf{x} \in \mathbf{x}^{B,g}, \text{ or in matrix form: } u^{B,g} = \mathbf{K}_{L2T}^{B,B} \boldsymbol{\phi}^{B,d}. \tag{31}$$

For a uniformly-refined tree, all leaves are at the same level, so we precompute and store one $\mathbf{K}_{L2T}^{B,B}$ matrix.

4.4 Near-Field Interactions

After the far-field contributions are computed, the final step is to compute near-field interactions for leaf boxes. This is the most expensive step in the computation, if carried out naïvely, and it is essential to optimize this part of the algorithm. For each leaf box B, we need to compute the influence of the volume density g^U for every box $U \in L_B^N$ (the near field boxes). Given a polynomial approximation γ^U to g^U, we evaluate the potential on an $n \times n \times n$ grid of samples $\mathbf{x}^{B,g}$ on B, which we then add to the far field contribution computed in (31).

The principal mechanism to accelerate this step is based on the observation that we may use a regular grid pattern of points in B, permitting the use of precomputation. More precisely,

$$u^{B,g}(\mathbf{x}) = \sum_U K[U, g](\mathbf{x}) = \sum_U \sum_{j=1}^{N_k} \gamma_j^U F_j^{U,B}(\mathbf{x}), \tag{32}$$

$$F_j^{U,B}(\mathbf{x}) = \int_U \beta_j \left(2^\ell (\mathbf{y} - \mathbf{c}_U) \right) K(\mathbf{x}, \mathbf{y}) d\mathbf{y}, \quad \text{for } \mathbf{x} \in \mathbf{x}^{B,g}, \tag{33}$$

where $\mathbf{c}_U$ is the center of box U. We evaluate $u^{B,g}$ on a uniform grid $\mathbf{x}_i^{B,g}, i = 1 \ldots n^3$ for $n < 6$ and on a tensor product Chebyshev grid for $n > 6$ to avoid condition problems, as discussed in section 8.2. In matrix form (32) becomes

$$u^{B,g} = \sum_U \mathbf{F}^{U,B} \gamma^U. \tag{34}$$

For a uniform octree, there are at most 27 possible locations for $U \in L_N^B$ with respect to B itself; using symmetries, however, only 4 are unique up to translation and rotation (section 5). As in the $S2M$ computations, adaptive Gaussian quadrature [8] is used to precompute and store the weights for these matrices. This can be done to machine precision for the function u and its first or second derivatives.

As each leaf box, B is not dependent on the near-field computations of any leaf box in T, it can quickly be seen how even the simplest approaches can take advantage of parallel architectures in the near-field computations. In section 8.4, we discuss how we use OpenMP [18] and load-balancing approaches to parallelize the near-field and other computational steps of the FMM for shared-memory, multiprocessor architectures.

4.5 Polynomial approximation of the solution

In order to compute the value of u^B at an arbitrary point in the box, it is convenient to approximate it as a polynomial v^B using a least-squares fit: minimizing

$$\sum_{i=1}^{n^3} \|u^B(\mathbf{x}_i) - \sum_{j=1}^{N_n} v_j^B \beta_j(\mathbf{x}_i - \mathbf{c}_B)\|^2 \quad \text{for } \mathbf{x}_i \in \mathbf{x}^B, \tag{35}$$

where $\beta_j \in \{P_a(x)P_b(y)P_c(z), 0 \le a+b+c \le n-1\}$ using either a monomial or Chebyshev polynomial basis, depending on the desired order n. For $n \le 6$ it is more convenient to use regular grids, while for $n > 6$ Chebyshev grid points provide greater stability. In section 8.2 we demonstrate the accuracy of equispaced points and Chebyshev points for $n = 4, 6, 8$. For box B at depth ℓ, if Γ is the matrix with entries $\Gamma_{ij} = \beta_j(2^\ell(\mathbf{x} - \mathbf{c}_B))$, (35) leads to the equation $v^B = \Gamma^{(+)} u^{B,g}$, where the pseudoinverse $\Gamma^{(+)}$ needs to be precomputed only once as it does not depend on the kernel and is scale-invariant in *all* cases; that is, $\Gamma_{ij} = \beta_j(\mathbf{x}_i^*)$ where $\mathbf{x}_i^*$ are grid points in $B^* = [-1, 1]^3$. Once the v_j^B are known, we can evaluate the solution at an arbitrary point $\mathbf{x}_t \in B$ as

$$u(\mathbf{x}_t) = \sum_{j=1}^{N_n} v_j^B \beta_j(\mathbf{x}_t - \mathbf{c}_B). \tag{36}$$

In general, we assume that k, the order of the approximation γ^B of the force g^B, is equal to n, the order of approximation of v^B; however, as source and target locations need not be the same, k and n can be different.

4.6 Polynomial force approximation from grid samples

We have assumed the right-hand side is already given as a polynomial. However, if the force is available in another form (e.g., as samples on an AMR grid or polynomials on an unstructured finite element grid), we need simply to build a k^{th}-order approximation of the right-hand side to the desired tolerance at regular grid points on each leaf node, followed by conversion to a polynomial representation, as in the preceding section. We view this step as outside the scope of the present paper.

4.7 Non-Uniform Source Distributions and Adaptive FMM

For nonuniform source distributions, leaf boxes may appear at different levels, leading to several additional types of interactions between boxes that need to be taken into account. For adaptive octrees, the number of relative positions of boxes one needs to consider can become very large. To avoid storing large number of precomputed matrices, we consider *level-restricted refinement*: we require adjacent leaf boxes be within one level of each other, a common restriction in tree codes and structured grids. Many fast approaches exist to convert arbitrary octrees to ones satisfying this constraint [53]; we currently use a straightforward sequential algorithm similar to [23].

Lists for adaptive FMM. Our definitions and notation follow [25, 30, 31]. For leaf box B, we define U and W lists:

- The *U-list*, L_U^B, consists of leaves adjacent to B, including itself; $L_U^B = L_N^B$ for uniform trees.
- The *W-list*, L_W^B, is the set of descendants of B's neighbors, not adjacent to B, but whose parents are adjacent to B. For any $W \in L_W^B$, W is at a finer level than B and $W \in \mathcal{N}^B$ (conversely, $B \in \mathcal{F}^W$).

For leaf and non-leaf boxes B, we define V and X lists.

- The *V-list*, L_V^B, is the set of B's parent's neighbor's children, not-adjacent to B. $L_V^B = L_I^B$ for uniform trees.
- The *X-list*, L_X^B, is the set of boxes A such that $B \in L_W^A$.

We note the following: $B \in L_U^A \Leftrightarrow A \in L_U^B$, $B \in L_V^A \Leftrightarrow A \in L_V^B$, and $B \in L_W^A \Leftrightarrow A \in L_X^B$. [60] shows an example domain with labeled lists; possible positions of boxes in the L_U^B, L_V^B, L_W^B and L_X^B are shown in Figure 3.

By the following lemma, for level-restricted trees, boxes in W and X lists have finite possible positions.

Lemma 4.1 *For a level-restricted tree T in which all neighboring leaf boxes are within one level of each other in the octree, for a box, B, all boxes in L_W^B and L_X^B must also be within one level of B.*

Proof For box B assume $\exists\, W \in L_W^B$ where $\ell_W - \ell_B \geq 2$. Then, for W's parent, P_W, $\ell_{P_W} - \ell_B \geq 1$, so $\exists$ descendants D of P_W, $D \in L_U^B$ and $\ell_D - \ell_B \geq 2$, violating our tree-level restriction. Thus, $\ell_W - \ell_B \leq 1$. Since $W \in L_W^B \Rightarrow B \in L_X^W, \forall\, X \in L_X^B, \ell_B - \ell_X \leq 1$. ∎

Boxes in L_U^B and L_V^B are handled as boxes in L_N^B and L_I^B, respectively, are in the uniform case. For leaf box B, if $W \in L_W^B$, then $W \notin \mathcal{F}^B$; therefore, W's contribution to B is not accounted for through its parent, P_B, but since $B \in \mathcal{F}^W$, we can evaluate $\phi^{W,u}$ at $\mathbf{x}^{B,g}$. Hence, using notation analogous to other operators, $M2T$ operators need to be defined. Further, for $X \in L_X^B$, $B \in \mathcal{N}^X$, but $X \in \mathcal{F}^B$. Thus, we need to evaluate contributions from X directly but can apply them to $\phi^{B,d}$; that is, we need to define an $S2L$ operator.

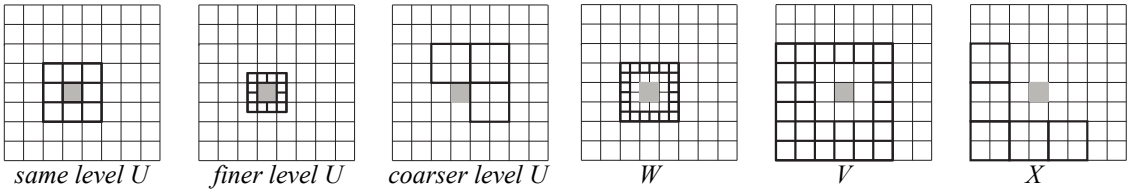


Figure 3: Possible box positions for different lists in a level-restricted tree in 2d. The configurations in 3d are analogous.

To summarize, for adaptive FMM, in addition to $M2M$, $M2L$, $L2L$ and $L2T$, two additional operators, $M2T$ and $S2L$ need to be defined, and $S2M$ and near-field ($S2T$) operators need to handle leaf boxes at arbitrary levels. We begin by describing changes to $S2M$ and $S2T$ operators and follow with a discussion of $M2T$ and $S2L$ operators.

S2M operators for the adaptive case. For *homogeneous* kernels, we store a *single* matrix $\mathbf{T}_{S2M}^{B^*}$, scaling for level ℓ as was done for the $M2M$ and $L2L$ operators. Let $B^* = [-1,1]^3$ at $\ell = 0$. Then, for $\mathbf{x} \in B$ at level ℓ, let $\mathbf{x}^* = 2^\ell(\mathbf{x} - \mathbf{c}^B)$ for $\mathbf{x}^* \in B^*$. For scaling exponent, r, $K(\mathbf{x}_i, \mathbf{y}_j) = \left(\frac{1}{2}\right)^{r\ell} K(\mathbf{x}_i^*, \mathbf{y}_j^*)$, and (21) becomes

$$F_j^B(\mathbf{x}) = \left(\frac{1}{2}\right)^{(r+2)\ell} K[B^*, \beta_j](\mathbf{x}^*) = \left(\frac{1}{2}\right)^{(r+2)\ell} F_j^{B^*}(\mathbf{x}^*) \quad \text{for all } \mathbf{x} \in \mathbf{x}^{B,g},\, \mathbf{x}^* \in \mathbf{x}^{B^*,g}.$$

In matrix form, $\mathbf{F}_{S2M}^B = 2^{-(r+2)\ell}\mathbf{F}_{S2M}^{B^*}$ and $\mathbf{K}_{S2M}^B = 2^{-r\ell}\mathbf{K}_{S2M}^{B^*}$. Solving for $\boldsymbol{\phi}^{B,u}$, (23) becomes

$$\boldsymbol{\phi}^{B,u} = \mathbf{T}_{S2M}^{B^*}\gamma^B, \tag{37}$$

where $\mathbf{T}_{S2M}^{B^*}$ is precomputed and stored. (For inhomogeneous kernels, we store one matrix per leaf level).

Neighbor list interactions for adaptive trees. For adaptive level-restricted trees, leaves may exist at any level and $U \in L_U^B$ may exist at one level finer or coarser than B. As above for the $S2M$ operators, for homogeneous kernels with scaling exponent r, we only compute matrices for pairs (B, U) with B scaled to B^* (U is appropriately scaled as well to U^*) such that (33) and (34) become

$$F_j^{U,B}(\mathbf{x}) = \left(\frac{1}{2}\right)^{(r+2)\ell} K[U^*, \beta_j](\mathbf{x}^*) = \left(\frac{1}{2}\right)^{(r+2)\ell} F_j^{(U,B)^*}(\mathbf{x}^*), \quad \text{for all } \mathbf{x} \in \mathbf{x}^{B,g},\, \mathbf{x}^* \in \mathbf{x}^{B^*,g}, \text{ and}$$

$$u^{B,g} = \sum_U \mathbf{F}_{S2T}^{U,B}\gamma^U = \left(\frac{1}{2}\right)^{(r+2)\ell} \sum_U \mathbf{F}_{S2T}^{(U,B)^*}\gamma^U. \tag{38}$$

Along with 27 possible same-level neighbors, there are 56 fine-level neighbors (one level deeper) and 7 coarse-level neighbors (one level higher), all constituting the 90 possible locations for boxes in L_U^B in a level-restricted octree. Using symmetries (section 5), we only precompute and store 10 matrices. For inhomogeneous kernels, this set of matrices is precomputed for each level for which leaf boxes exist.

M2T and S2L operators. For leaf box B and $W \in L_W^B$, we need an operator that evaluates $\phi^{W,u}$ at $\mathbf{x}^{B,g}$:

$$u^{B,g}(\mathbf{x}) = \sum_W K[\mathbf{y}^{W,u}, \boldsymbol{\phi}^{W,u}](\mathbf{x}) \quad \text{for } \mathbf{x} \in \mathbf{x}^{B,g}, \text{ or in matrix form: } u^{B,g} = \sum_W \mathbf{K}_{M2T}^{W,B} \phi^{W,u}, \tag{39}$$

for precomputed $\mathbf{K}_{M2T}^{W,B}$. For all boxes B, L_X^B contains leaves X, for which contributions to B are computed at $\mathbf{x}^{B,d}$.

$$u^{B,d}(\mathbf{x}) = \sum_X K[X, g^X](\mathbf{x}) \approx \sum_X \sum_{j=1}^{N_k} \gamma_j^X F_j(\mathbf{x}) \quad \text{for } \mathbf{x} \in \mathbf{x}^{B,d}, \text{ or in matrix form: } u^{B,d} = \sum_X \mathbf{F}_{S2L}^{X,B} \gamma^X. \tag{40}$$

There are 152 possible locations for $W \in L_W^B$; however, only six locations are distinct up to translation and rotation; also, due to the inverse relationship between L_X^B and L_W^B, the number of symmetry classes is the same (section 5). For homogeneous kernels, only six $\mathbf{K}_{M2T}^{W,B}$ and six $\mathbf{F}_{S2L}^{X,B}$ matrices are precomputed for level $\ell = 0$ and scaled as necessary. For inhomogeneous kernels we compute and store these sets for each leaf level.

Remark In cases where the order of γ is low compared to the order of $\phi^{B,u}$ and $\phi^{B,d}$, the size of $M2T$ and $S2L$ operators may actually be *larger* than those needed for direct computation of contributions from $W \in L_W^B$ or $X \in L_X^B$ to $\mathbf{x}^{B,g}$. Assuming we have a homogeneous kernel, if $W \in L_W^B$ is a leaf box, we can replace $\mathbf{K}_{M2T}^{W,B}$ with $\mathbf{F}_{S2T}^{W,B}$, constructed exactly in the same way as for boxes in the neighbor list L_U^B. Similarly, for leaf box B and for a box $X \in L_X^B$, we can replace $\mathbf{F}_{S2L}^{X,B}$ with $\mathbf{F}_{S2T}^{X,B}$. For homogeneous kernels, these operators are computed for B^* only and scaled as necessary as in section 4.4.

4.8 Pseudocode and Complexity for Kernel-Independent FMM Volume Solver

Pseudocode. We assume that a tree-level restricted octree, T, already exists [23] and that for each box, B, we are given the approximation, γ^B to the force g^B (we discuss how to construct γ from g in section 4.6). For clarity, we do not include the optimization of replacing $M2T$ and $S2L$ with $S2T$ operators when more efficient as discussed above.

Computational complexity and storage requirements. We analyze the complexity for a uniformly-refined octree. The analysis for the adaptive FMM is similar but slightly more complicated. We assume a homogeneous scalar kernel such as the Laplace kernel in equation (10) for analyzing the storage and computational complexities. Further, we assume that there are ℓ levels in the octree T. For a uniform tree, this implies we have $M_\ell = 8^\ell$ leaves and $M_t = (8^{\ell+1} - 1)/7$ total boxes in T. If we are using a k^{th}-order polynomial approximation to the force at each leaf, we further assume there are approximately $N = M_\ell n^3$ total target points and $C = M_\ell N_k$ total coefficients. Let p be the number of coefficients sought in the multipole expansion, affecting the size of the equivalent densities and surfaces; for a desired level of precision, $\epsilon_{fmm} = 10^{-n_p}$ in the expansion, $p = n_p^3 - (n_p - 2)^3$. In table 1, we indicate the computational complexity of each step of the non-adaptive FMM algorithm as well as the amount of precomputation and storage used for operators at each step. For non-uniform source distributions, we store additional operators for the near-field interactions in the *U*, *W*, and *X* operators; the complexity of these operators are based on the degree of adaptivity.

Finally, we note that the computational and storage complexities will scale linearly for matrix or inhomogeneous kernels. For example, for the Stokes kernel in equation (12), the number of coefficients, p, scales as a results of the matrix kernel size to $p = 9(n_p^3 - (n_p - 2)^3)$. For the Modified Helmholtz kernel in equation (11), the inhomogeneous nature of the kernel results in an increased storage complexity, which varies depending on the number of different levels in the tree.

5 Symmetries for precomputed interaction operators

For a box B and all boxes in L_U^B,L_V^B,L_W^B and L_X^B, the number of different relative positions can be large, so precomputing all possible interaction matrices may require significant time and storage. Performance can also be affected by the need for random access of large amounts of precomputed data. The number of precomputed matrices can be

Algorithm 1 Kernel-Independent Volume FMM

STEP 1 - BUILD LISTS
for each box B in *preorder* traversal of T **do**
  build L_U^B, L_W^B, L_X^B, and L_V^B (section 4.7)
end for
STEP 2 - UPWARD PASS (section 4.2)
for each box B in *postorder* traversal of T **do**
  if B is a leaf box **then**
    Convert local force approximations to upward densities:
    $\phi^{B,u} := \mathbf{T}_{S2M}^B \gamma^B$ (23)
  else
    Translate children's upward densities to parent's upward density:
    $\phi^{B,u} := \sum_C \mathbf{T}_{M2M}^{C,B} \phi^{C_i,u}$ (25)
  end if
end for
STEP 3 - DOWNWARD PASS (section 4.3)
for each non-root box B in *preorder* traversal of T **do**
  Add potentials due to parent downward density, U and X boxes to get the downward check potential

$$u^{B,d} := \mathbf{K}_{L2L}^{P,B} \phi^{P,d} + \sum_{V \in L_V^B} \mathbf{K}_{M2L}^{V,B} \phi^{V,u} + \sum_{X \in L_X^B} \mathbf{F}_{S2L}^{X,B} \gamma^X \text{ (29), (27), (40)}$$

  Translate the check potential to the downward density:
  $\phi^{B,d} := (\mathbf{K}_{L2L}^{B,B})^{-1} u^{B,d}$ (30)
  if B is a leaf box **then**
    Compute potentials from adjacent and W boxes to the potential at grid locations:

$$u^{B,g} := \sum_{U \in L_U^B} \mathbf{F}_{S2T}^{U,B} \gamma^U + \sum_{W \in L_W^B} \mathbf{K}_{M2T}^{W,B} \phi^{W,u} \text{ (34), (39)}$$

    Add the potential from the far field:
    $u^{B,g} := u^{B,g} + \mathbf{F}_{L2T} \phi^{B,d}$ (31)
  end if
end for

Operator	Complexity	Storage	Operator	Complexity	Storage
$S2M$: $\mathbf{T}_{S2M}^B$	$O(Cp)$	pN_k	Near Interaction: $\mathbf{F}_{S2T}^{U,B}$	$O(27NN_k)$	$4N_k n^3$
$M2M$: $\mathbf{T}_{M2M}^{C,B}$	$O((M_t - M_\ell)p^2)$	p^2	U-list (*adaptive*): $\mathbf{F}_{S2T}^{U,B}$		$10N_k n^3$
$M2L$: $\mathbf{K}_{M2L}^{V,B}$	$O(M_t p^{3/2} \log(p) + 189 M_t p^{3/2})$	$16p^{3/2}$	W-list: $\mathbf{F}_{S2T}^{W,B}$, $\mathbf{K}_{M2T}^{W,B}$		$6n^3(N_k + p)$
$L2L$: $\mathbf{K}_{L2L}^{P,B}$, $(\mathbf{K}_{L2L}^{B,B})^{-1}$	$O(M_t p^2)$	$2p^2$	X-list: $\mathbf{F}_{S2T}^{X,B}$, $\mathbf{F}_{S2L}^{X,B}$		$6N_k(n^3 + p)$
$L2T$: $\mathbf{K}_{L2T}^{B,B}$	$O(Np)$	pn^3			

Table 1: Computational complexity and storage requirements for a scalar homogeneous kernel. These values scale linearly for matrix and inhomogeneous kernels.

substantially reduced via symmetries; that is, many box positions are equivalent in the sense that there is a rigid transformation $\mathcal{T}$, mapping box Z_1 to Z_2 and box B to itself. We store a single matrix for a representative box for each symmetry class, obtaining matrices for all elements of the class by applying $\mathcal{T}$ to the matrix for the representative box.

For every list type $Z \in \{U, V, W, X\}$, we define a set of possible positions $Pos(Z)$ and a set of symmetry classes which form a partition of $Pos(Z)$. For each class, we define a *reference box*, and for each box position in $Pos(Z)$, we need an efficient way to determine its class and a transformation $\mathcal{T}(B) : \mathbf{R}^3 \to \mathbf{R}^3$ mapping it to the reference box.

For all lists, the symmetries are related to the transformations of space which map a grid of cubes to itself. We consider N^3 grids of sizes 1^3 to 7^3 (we discuss which lists correspond to which cubes in more detail below) and begin by classifying all symmetries of such grids.

Grid symmetries. The cubes on the N^3 grid are indexed by (i, j, k) with values $-M \ldots -1, 0, 1 \ldots M$ for odd $N = 2M + 1$ and $-M \ldots -1, 1 \ldots M$ for even $N = 2M$. We skip index 0 for even grids to ensure cube centers and indices are transformed by symmetries in the same way. If the cube size is 1, cube centers are exactly the indices (i, j, k) for odd N and differ by $\pm 1/2$ for even N, depending on the index sign. Each N^3 grid can be partitioned into M (for even N) or $M + 1$ (for odd N) layers. Layer 0 consists of one cube and exists for odd N, and layer M consists of cubes on the surface of the N^3 grid. For odd N, layer l has size $(2l + 1)^3$ and for even N, layer l has size $(2l)^3$.

The group of symmetries G_{cube} of a cube has order 48. For a cube centered at zero, transformations in G_{cube} are compositions of rotations and reflections, mapping each axis direction to another, possibly with orientation reversed. Any permutation of directions is possible, so we identify the group with $S_3 \times J^3$, where S_3 is the group of permutations of length 3, and J is the two-element group of reflections. The rotational part of any element of G_{cube} can be specified as a permutation of length 3 on the set of axes $\{x, y, z\}$, with an orientation 1 or -1 specified for each axis. Transformations from G_{cube} encoded in this way can be applied to points very efficiently: for a point $\mathbf{x} \in \mathbf{R}^3$, the permutation is applied to its coordinates, which are then scaled by 1 or -1.

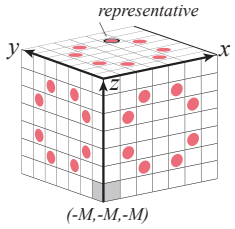


Figure 4: A 3d view of the $(1, 2, 3)$ class in the 7^3 grid for $M = 3$, showing the representative box.

For the N^3 grid, the equivalence classes under the action of G_{cube} can be enumerated combinatorially. If two indices (i, j, k) and (i', j', k') differ only by signs of components, corresponding cubes are in the same class, mapped by reflections. To enumerate all classes, we consider cubes with nonnegative indices. Two cubes with nonnegative indices (i, j, k) and (i', j', k') are in the same class if and only if there is a permutation mapping (i, j, k) to (i', j', k'). For $i \neq j \neq k$ and $i, j, k \in [1, M]$, seven series of equivalence classes are easily enumerated, corresponding to signatures (i, j, k), (i, i, j), (i, i, i), $(0, i, i)$, $(0, 0, i)$ and $(0, 0, 0)$. A reference box in every class is uniquely defined by requiring that its three indices are all nonnegative and are in nondecreasing order (Figure 4). The properties of classes in each series are summarized in Table 2. For a box Z, with grid index (i, j, k) relative to B, the reference box is obtained by taking absolute values and sorting the indices; sign changes and a permutation mapping (i, j, k) to the reference box index also encode the transformation.

Symmetries of L_U^B. Due to the tree-level restriction, boxes $U \in L_U^B$ are either neighbors of B, neighbors of B's parent and adjacent to B, or adjacent children of neighbors of B. We denote these three sublists of L_U^B by $L_{U,n}^B$, $L_{U,p}^B$ and $L_{U,c}^B$ respectively. Note that $A \in L_{U,p}^B$ is equivalent to $B \in L_{U,c}^A$; hence, it is sufficient to consider $L_{U,n}^B$ and $L_{U,c}^B$. The neighbors of B on the same level as B form a 3^3 grid centered at B, so from table 2, the number of classes is 4: $(1, 1, 1)$, $(0, 1, 1)$, $(0, 0, 1)$, and $(0, 0, 0)$. Locations of $U \in L_{U,c}^B$ can be thought of as the outer layer of a 4^3 grid, with $M = 2$ and B as the 2^3 subgrid in the center, so we obtain 3 classes: $(1, 1, 2)$, $(1, 2, 2)$, $(2, 2, 2)$, giving 10 classes for L_U^B.

Symmetries of L_V^B. Boxes in L_V^B are children of neighbors of the parent of B, so they can all be represented by cubes of a 6^3 grid; however, the group of rigid transformations of the grid mapping to itself do not necessarily preserve B. Hence, instead regard L_V^B as a subset of a 7^3 grid centered at B with $M = 3$. All $V \in L_V^B$ are in layers 2 and 3, and there are 10 classes: for layer 3, classes $(i, j, 3)$, $i, j = 1 \ldots 3$, $i \leq j$ and for layer 2, classes $(i, j, 2)$, $i, j = 0, 1, 2$, $i \leq j$. Because we consider only a subset of the the full 7^3 grid, the class sizes are smaller, but it can easily be seen that no class becomes empty, so the number is optimal.

Symmetries of L_W^B, L_X^B. For a level-restricted tree, boxes $W \in L_W^B$ are children of neighbors of B not adjacent to B, that is, they reside in the surface layer of a 6^3 grid with B as the central 2^3 grid. For $M = 3$, we have 6 classes

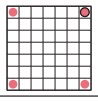
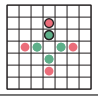
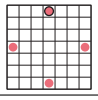
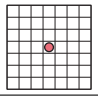
Series signature	7^3 classes	Reference cube	# of classes/grid	# classes/layer	class size
(i,j,k)		$(\|i\|,\|j\|,\|k\|)$, $\|i\|<\|j\|<\|k\|$	$\binom{M}{3}$	$\binom{M-1}{2}$	48
(i,i,j)		$(\|i\|,\|i\|,\|j\|)$, $\|i\|<\|j\|$ or $(\|i\|,\|j\|,\|j\|)$	$M(M-1)$	$2(M-1)$	24
(i,i,i)		$(\|i\|,\|i\|,\|i\|)$	M	1	8
$(0,i,j)$		$(0,\|i\|,\|j\|)$	$\binom{M}{2}$	$M-1$	24
$(0,i,i)$		$(0,\|i\|,\|i\|)$	M	1	12
$(0,i,i)$		$(0,0,\|i\|)$	M	1	6
$(0,0,0)$	—	$(0,0,0)$	1	—	1

Table 2: Series of equivalence classes of cubes in an N^3 grid. For even N, $N=2M$, and only the first 3 series of classes may be nonempty. For odd N, $N=2M+1$, and all classes are present. For $M\le 2$, (i,j,k) classes are empty, and for $M=1$, (i,i,j) and $(0,i,j)$ classes are also empty. Class (0,0,0) corresponding to the center of the grid, exists only in layer 0. In the figures, boxes in different classes in one series are marked with circles of different colors, representative boxes are marked with circles with black border. The view is from the top, with first index direction to the right, second direction up and third towards the viewer, as in Figure 4. The total number of classes for $(2M)^3$ layers is $(M+1)M/2$ (classes (i,j,M) with $i,j=1\dots M$, $i\le j$), and for $(2M+1)^3$ layers, it is $(M+2)(M+1)/2$ (classes (i,j,M) with $i,j=0\dots M$, $i\le j$).

from table 2: $(i,j,3)$, $i,j=1\dots 3$, $i\le j$. Due to duality, the number of classes for L_X^B is the same, but the class sizes may *not* be the same.

Summary. For a given pair (B,Z), if $Z\in\{L_{U,n}^B, L_{U,c}^B, L_V^B, L_W^B\}$, determine the translation and scaling which map B to the central box or 2^3 subgrid of a larger grid. Then, apply the same transformation to the center of Z; resulting coordinates yield the index (i,j,k), which is translated into the reference box and rotation as described above.

6 Numerical Results

The above algorithm has been implemented in C++, and we have tested several kernels and source and target point distributions. Our tests were run on an Intel Zeon-based X7560 (2.27GHz 64 bit) system with 16 CPUs and 128GB of RAM; the major computation loops are accelerated with OpenMP [18] as discussed in section 8.4.

We first test the free-space Poisson solver on three different types of problems designed to show how our algorithm handles increasing levels of complexity in the force distribution.

We use an adaptive-refinement strategy similar to [23]. For this, we compute a k^{th}-order polynomial approximation, γ^B, to the force $g^B(\mathbf{x})$ sampled on a $k\times k\times k$ grid. We let $\tilde{g}^B$ be the force evaluated on a refined $2k\times 2k\times 2k$ grid. If $\left|\left|g^B(\mathbf{x})-\tilde{g}^B(\mathbf{x})\right|\right|_2>\epsilon_{rhs}$, B is subdivided, and the octree is balanced as needed. Three force distributions, used in Examples (1-3) below are shown in Figure 5.

Example 1. The first experiment tests the accuracy of our method for solving the Poisson equation (equation (7) with kernel (10)) with a fast-decaying smooth right-hand side.

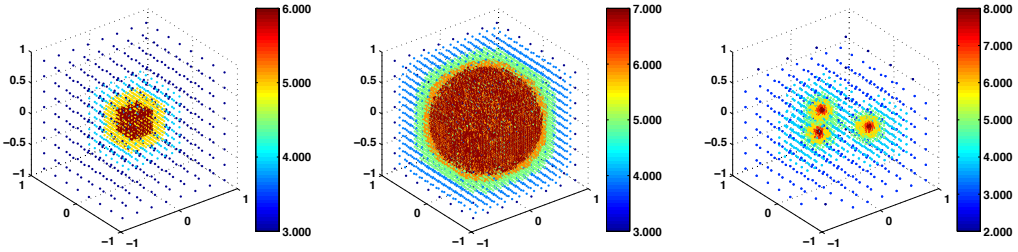


Figure 5: Sample force distributions based on adaptive refinement. Each point, colored by its tree level, ℓ, indicates the center of a leaf box, B. *Left*: A single sharply-peaked Gaussian function; *Middle:* A discontinuous force distribution, equal to one inside a sphere and zero outside; *Right:* A discontinuous force distribution involving oscillatory functions restricted to the interiors of a set of three spheres.

$$-\Delta u(\mathbf{x}) = \sum_{i=0}^{8} -e^{-L||\mathbf{x}-\mathbf{x}_i||^2} \left(4L\,||\mathbf{x}-\mathbf{x}_i||^2 - 6L\right), L = 250$$

with solution

$$u(\mathbf{x}) = \sum_{i=0}^{8} -e^{-L||\mathbf{x}-\mathbf{x}_i||^2},$$

where $\mathbf{x}_i = (\pm\frac{3}{40}, \pm\frac{3}{40}, \pm\frac{3}{40})$ inside of the $[-1,1]^3$ box. This test requires a high degree of adaptivity to achieve good accuracy with a limited number of points.

In Table 3, ϵ_{fmm} is the precision of the translation operators, ϵ_{rhs} is the refinement criterion for the adaptive refinement of the source distribution, and M_ℓ is the number of leaves in the tree T with L_T levels. The number of points N_{pts} is computed as $M_\ell k^3$ where k is the order of the polynomial. This number of points per leaf is chosen to be sufficiently large to build the polynomial approximation of order k. The computation time T_{FMM} is given in seconds, and the "rate" is in points per second. E_2 and E_∞ are the relative L^2 and L^∞ errors, respectively. Timings include FMM evaluation times only; when the precision ϵ_{fmm} remains constant, the rate of work per source and target point remains close to constant, as we would expect since the FMM algorithm scales linearly.

Example 2. In this example, we consider a discontinuous right-hand side, with $g(\mathbf{x}) = 1$ inside a sphere of radius $R = 0.75$, and $g(\mathbf{x}) = 0$ outside the sphere. Letting $r = ||\mathbf{x}||$, the problem becomes

$$-\Delta u(\mathbf{x}) = \left\{ \begin{array}{cc} 1 & \text{if } r \leq R \\ 0 & \text{else} \end{array} \right\}$$

with solution

$$-\Delta u(\mathbf{x}) = \left\{ \begin{array}{cc} \left(R^2 - r^2\right)/6 + R^2/3 & \text{if } r \leq R \\ R^3/3r^2 & \text{else} \end{array} \right\}.$$

While this problem can be handled analytically, it serves as a useful test of performance on adaptive data structures that are refined in the neighborhood of a surface. The number of points indicates the total number of points both inside and outside the sphere. Since the coefficient representation of the force for a leaf node entirely outside of the sphere is zero, these boxes are ignored in all evaluation phases; this increases the computed rate somewhat. A greater speedup is achieved from the observation that leaf nodes entirely in the interior have a constant source distribution, so that only one polynomial coefficient is non-zero. This significantly accelerates both the near-field and S2M calculation stages. Results are shown in Table 4.

ϵ_{fmm}	ϵ_{rhs}	M_ℓ	N_{pts}	L_T	E_2	E_∞	T_{FMM}	Rate
Fourth-Order Force Approximation								
10^{-2}	10^{-2}	736	47104	6	$2.3E-02$	$2.4E-02$	$9.1453E-03$	$5.15E+06$
10^{-4}	10^{-2}	736	47104	6	$1.1E-03$	$6.8E-04$	$2.4328E-02$	$1.94E+06$
10^{-4}	10^{-4}	3088	197632	7	$1.1E-04$	$2.7E-04$	$9.6590E-02$	$2.05E+06$
10^{-6}	10^{-4}	3088	197632	7	$3.8E-05$	$2.5E-05$	$3.7906E-01$	$5.21E+05$
10^{-6}	10^{-6}	19328	1236992	8	$3.8E-06$	$3.6E-06$	$2.3889E+00$	$5.18E+05$
10^{-8}	10^{-6}	19328	1236992	8	$3.7E-06$	$1.3E-06$	$6.2057E+00$	$1.99E+05$
10^{-8}	10^{-8}	143088	9157632	9	$1.6E-07$	$8.810E-08$	$4.5280E+01$	$2.02E+05$
Sixth-Order Force Approximation								
10^{-4}	10^{-4}	1408	304128	6	$1.1E-04$	$2.3E-04$	$1.1016E-01$	$2.76E+06$
10^{-6}	10^{-4}	1408	304128	6	$1.4E-05$	$3.5E-05$	$1.8717E-01$	$1.62E+06$
10^{-6}	10^{-6}	4936	1066176	7	$9.0E-07$	$2.2E-06$	$6.6737E-01$	$1.60E+06$
10^{-8}	10^{-6}	4936	1066176	7	$3.3E-07$	$1.6E-07$	$1.6155E+00$	$6.60E+05$
10^{-8}	10^{-8}	20112	4344192	8	$2.4E-08$	$6.2E-08$	$6.8652E+00$	$6.33E+05$
10^{-10}	10^{-8}	20112	4344192	8	$1.8E-08$	$1.0E-08$	$1.5771E+01$	$2.76E+05$
10^{-10}	10^{-10}	92072	19887552	9	$6.3E-09$	$9.7E-09$	$7.5103E+01$	$2.65E+05$
Eighth-Order Force Approximation								
10^{-6}	10^{-6}	2024	1036288	7	$9.2E-07$	$3.5E-06$	$4.6688E-01$	$2.22E+06$
10^{-8}	10^{-6}	2024	1036288	7	$3.7E-07$	$8.2E-07$	$7.4189E-01$	$1.40E+06$
10^{-8}	10^{-8}	5440	2785280	7	$1.9E-08$	$6.6E-08$	$2.1128E+00$	$1.32E+06$
10^{-10}	10^{-8}	5440	2785280	7	$7.7E-09$	$7.6E-09$	$4.3588E+00$	$6.40E+05$
10^{-10}	10^{-10}	22800	11673600	8	$4.1E-09$	$5.6E-09$	$1.9449E+01$	$6.00E+05$
10^{-12}	10^{-10}	22800	11673600	8	$2.6E-09$	$4.7E-09$	$4.1602E+01$	$2.81E+05$
10^{-12}	10^{-12}	50352	25780224	9	$2.1E-09$	$4.6E-09$	$9.2139E+01$	$2.80E+05$

Table 3: Free-Space Poisson Equation, Example 1: Gaussian bump at the origin

ϵ_{fmm}	ϵ_{rhs}	k	M_ℓ	N_{pts}	L_T	L^2	L^∞	T_{FMM}	Rate
10^{-2}	10^{-2}	4	232	14848	4	$1.1E-02$	$1.6E-02$	$1.8346E-03$	$8.09E+06$
10^{-3}	10^{-3}	4	1184	75776	5	$4.3E-03$	$4.6E-03$	$1.1007E-02$	$6.88E+06$
10^{-4}	10^{-4}	4	5888	376832	6	$1.4E-04$	$2.5E-04$	$1.1327E-01$	$3.33E+06$
10^{-5}	10^{-5}	6	11432	2469312	7	$1.6E-05$	$3.3E-05$	$6.2147E-01$	$3.97E+06$
10^{-6}	10^{-6}	6	80088	17299008	8	$7.2E-06$	$1.8E-05$	$6.0647E+00$	$2.85E+06$
10^{-7}	10^{-7}	8	127856	65462272	8	$9.4E-07$	$3.3E-06$	$1.8852E+01$	$3.47E+06$
10^{-8}	10^{-8}	8	528984	270839808	10	$3.7E-07$	$9.8E-07$	$1.2335E+02$	$2.20E+06$
10^{-9}	10^{-9}	8	2074360	1062072320	10	$3.2E-08$	$2.6E-07$	$6.4900E+02$	$1.64E+06$

Table 4: Free-Space Poisson Equation, Example 2: Discontinuous Force

Example 3. For our third example, we replicate an experiment from [46] for a highly-oscillatory force with discontinuities along multiple surfaces.

$$f_m(r) = \left\{ \begin{array}{ll} ((r-r^2)\sin(2m\pi r))^2 & \text{if } r < 1 \\ 0 & \text{if } r \geq 1 \end{array} \right\}$$

$$-\Delta u(\mathbf{x}) = \frac{1}{R^3} \sum_{i=0}^{2} f_m(|\mathbf{x}-\mathbf{c}_i|/R), \tag{41}$$

where $\mathbf{c}_0 = (3/16, 7/16, 13/16)$, $\mathbf{c}_1 = (7/16, 13/16, 3/16)$, $\mathbf{c}_2 = (13/16, 3/16, 7/16)$, $R = 0.05$, and the wavelength of f_m is $\lambda_m = R/(2m) = (1/40m)$. Defining

$$
\begin{aligned}
\phi_m(r) &= \frac{(10r^6 - 28r^5 + 21r^4 - 7)}{840} + \frac{(60r - 120)}{r\lambda_m^6} - \frac{9}{\lambda_m^4} \\
&+ \left[\frac{(300r - 120)}{r\lambda_m^6} - + \frac{(30r^2 - 36r + 9)}{\lambda_m^4} + \frac{(r^4 - 2r^3 + r^2)}{2\lambda_m^2}\right] \cos(r\lambda_m) \\
&+ \left[-\frac{360}{r\lambda_m^7} + \frac{(120r^2 - 96r + 12)}{r\lambda_m^5} + \frac{(5r^3 + 8r^2 - 3r)}{\lambda_m^3}\right] \sin(r\lambda_m), \qquad \text{and} \\
\theta_m(r) &= \left(\frac{360}{\lambda_m^6} - \frac{12}{\lambda_m^4} - \frac{1}{120}\right)/r,
\end{aligned}
$$

we write the solution to equation (41) as

$$
u^{exact}(\mathbf{x}) = \left\{ \begin{array}{ll}
\phi(||\mathbf{x} - \mathbf{c}_0||/R) + \sum_{i=1,2} \theta(||\mathbf{x} - \mathbf{c}_i||/R) & \text{if } ||\mathbf{x} - \mathbf{c}_0|| < R \\
\phi(||\mathbf{x} - \mathbf{c}_1||/R) + \sum_{i=0,2} \theta(||\mathbf{x} - \mathbf{c}_i||/R) & \text{if } ||\mathbf{x} - \mathbf{c}_1|| < R \\
\phi(||\mathbf{x} - \mathbf{c}_2||/R) + \sum_{i=0,1} \theta(||\mathbf{x} - \mathbf{c}_i||/R) & \text{if } ||\mathbf{x} - \mathbf{c}_2|| < R \\
\sum_{i=0}^{2} \theta(||\mathbf{x} - \mathbf{c}_i||/R) & \text{else}
\end{array} \right\}
$$

In order to compare our results to [46], we use the error metric introduced there. Let ϵ^B be the vector of errors calculated as the difference between the calculated and exact solutions on B and calculate the following norm over all leaf boxes.

$$
\left|\left|\epsilon_{all}^B\right|\right|_2 = \sum_B \left(\int \frac{\epsilon^B}{||u^{exact}||_\infty}\right)^{\frac{1}{2}}.
$$

As indicated in [46],

$$
\left|\left|u^{exact}\right|\right|_\infty = \left|\left(-\frac{6}{\lambda_m^4} - \frac{1}{120}\right)/R + \left(\frac{720}{\lambda_m^6} - \frac{24}{\lambda_m^4} - \frac{1}{120}\right)/||\mathbf{c}_i - \mathbf{c}_j||\right| \quad \forall i, j = 0, 1, 2, i \neq j
$$

Our automatic refinement strategy refines within or near the sphere surfaces, with refinement taking place in the exterior of the spheres only for the purpose of tree-balancing. We build coefficients only on leaf boxes which contain non-zero source distributions: either interior to or intersecting one of the three spheres. Results are shown in Table 5.

m	ϵ_{fmm}	ϵ_{rhs}	M_ℓ	N_{pts}	L_T	k	$\left\|\left\|\epsilon_{all}^B\right\|\right\|_2$	$\left\|\left\|\epsilon_{all}^B\right\|\right\|_\infty$
1	10^{-6}	10^{-6}	3984	254976	8	4	$2.3E-07$	$2.2E-05$
1	10^{-8}	10^{-8}	7296	1575936	9	6	$1.1E-08$	$7.1E-07$
1	10^{-10}	10^{-10}	24144	12361728	9	8	$1.6E-10$	$1.8E-08$
7	10^{-6}	10^{-6}	93816	6004224	10	4	$2.2E-06$	$1.8E-04$
7	10^{-8}	10^{-8}	195984	42332544	10	6	$9.3E-09$	$1.1E-06$
7	10^{-10}	10^{-10}	228312	116895744	10	8	$1.1E-10$	$4.3E-08$
15	10^{-6}	10^{-6}	140568	8996352	10	4	$6.4E-07$	$8.6E-05$
15	10^{-8}	10^{-8}	1092456	235970496	11	6	$7.2E-08$	$4.2E-06$
15	10^{-10}	10^{-10}	1596672	817496064	11	8	$4.8E-10$	$6.1E-08$
30	10^{-6}	10^{-6}	148272	9489408	10	4	$5.8E-07$	$6.2E-05$
30	10^{-8}	10^{-8}	1491216	322102656	11	6	$2.1E-08$	$3.8E-06$
30	10^{-10}	10^{-10}	1720152	880717824	12	8	$4.9E-09$	$2.0E-06$
60	10^{-6}	10^{-6}	150288	9618432	11	4	$6.0E-07$	$9.6E-05$
60	10^{-8}	10^{-8}	1502592	324559872	11	6	$9.2E-08$	$1.4E-05$
60	10^{-10}	10^{-9}	1659312	849567744	11	8	$7.7E-08$	$1.7E-05$

Table 5: Free-Space Poisson Equation, Example 3: Discontinuities along several spherical surfaces containing oscillating source distributions

While the performance of our code cannot be compared easily to the optimized and parallelized scheme presented in [46], we have implemented much higher order accurate schemes. Thus, as expected, we are able to reach comparable accuracies with significantly fewer points. To compare the number of points required, we consider the number of

points in the finest level solve of their three-level examples. For $m = 7$, we achieve accuracy on par with their most accurate tests with approximately $1/100$ as many points. For $m = 15$, we require approximately $1/5$ as many points, and with $1/4$ as many points, we achieve about two orders of magnitude greater accuracy. For $m = 30$, we achieve equivalent results with approximately $1/4$ as many points. Additionally, we extended the examples for an even higher wavenumber component ($m = 60$), decreasing the wavelength to 4.1710^{-4}, and achieving good results with fewer than 10^9 points.

Example 4. For the Modified Helmholtz equation (equation (8) with kernel (11)), we use a right-hand side similar to that of Example 1, setting the Helmholtz parameter (inverse Debye length) to $\alpha = \pi$:

$$\alpha\, u(\mathbf{x}) - \Delta u(\mathbf{x}) = \sum_{i=0}^{8} -e^{-L||\mathbf{x}-\mathbf{x}_i||^2} \left(4L\,||\mathbf{x}-\mathbf{x}_i||^2 - 6L - \alpha\right), L = 250$$

with solution

$$u(\mathbf{x}) = \sum_{i=0}^{8} -e^{-L||\mathbf{x}-\mathbf{x}_i||^2},$$

for $\mathbf{x}_i = (\pm\frac{3}{40}, \pm\frac{3}{40}, \pm\frac{3}{40})$ inside of the $[-1, 1]^3$ box. All translation matrices are computed to a precision of $\epsilon_{fmm}/10$. These matrices can be computed at run-time in a lazy manner; if α is known before run-time, these tables can be precomputed, stored, and loaded as necessary. Additionally, since the right-hand side is the same as in Example 1, we use the same point distribution; hence, the timings are essentially the same as Experiment 1 and are omitted here. Results are shown in Table 6.

ϵ_{fmm}	ϵ_{rhs}	M_ℓ	N_{pts}	L_T	E_2	E_∞
Fourth-Order Force Approximation						
10^{-2}	10^{-2}	736	47104	6	$2.3E-02$	$2.4E-02$
10^{-4}	10^{-2}	736	47104	6	$6.0E-04$	$5.6E-04$
10^{-4}	10^{-4}	3088	197632	7	$1.1E-04$	$1.8E-04$
10^{-6}	10^{-4}	3088	197632	7	$2.4E-05$	$2.1E-05$
10^{-6}	10^{-6}	19328	1236992	8	$2.3E-06$	$3.4E-06$
Sixth-Order Force Approximation						
10^{-4}	10^{-4}	1408	304128	6	$1.1E-04$	$2.4E-04$
10^{-6}	10^{-4}	1408	304128	6	$1.4E-05$	$3.5E-05$
10^{-6}	10^{-6}	4936	1066176	7	$8.5E-07$	$2.5E-06$
10^{-8}	10^{-6}	4936	1066176	7	$1.4E-07$	$1.3E-07$
10^{-8}	10^{-8}	20112	4344192	8	$1.5E-08$	$7.5E-08$
10^{-10}	10^{-8}	20112	4344192	8	$8.9E-09$	$7.4E-09$
10^{-10}	10^{-10}	92072	19887552	9	$2.2E-09$	$5.7E-09$
Eighth-Order Force Approximation						
10^{-6}	10^{-6}	2024	1036288	7	$1.9E-06$	$4.7E-06$
10^{-8}	10^{-6}	2024	1036288	7	$1.8E-07$	$4.6E-07$
10^{-8}	10^{-8}	5440	2785280	7	$1.2E-08$	$1.4E-08$
10^{-10}	10^{-8}	5440	2785280	7	$7.7E-09$	$7.6E-09$
10^{-10}	10^{-10}	22800	11673600	8	$3.4E-09$	$5.6E-09$
10^{-12}	10^{-10}	22800	11673600	8	$1.3E-09$	$2.0E-09$
10^{-12}	10^{-12}	50352	25780224	9	$2.0E-09$	$2.6E-09$

Table 6: Free-Space Modified Helmholtz Equation, Example 4: Gaussian bump at the origin

Example 5. We test the ability of our code to handle matrix kernels by solving the Stokes equations (equation (9) with kernel (12)) with the following divergence-free fast-decaying force.

$$-\mu\Delta u(\mathbf{x}) + \nabla p(\mathbf{x}) = \sum_{i=0}^{8} \left(8L^3\,||\mathbf{x}-\mathbf{x}_i||^2 - 20L^2\right) e^{-L||\mathbf{x}-\mathbf{x}_i||^2} \left[\nabla \times (\mathbf{x}-\mathbf{x}_i)\right]$$

with solution

$$u(\mathbf{x}) = \frac{2L}{\mu} \sum_{i=0}^{8} e^{-L||\mathbf{x}-\mathbf{x}_i||^2} \left[\nabla \times (\mathbf{x} - \mathbf{x}_i)\right]$$

for $\mathbf{x}_i = (\pm\frac{3}{40}, \pm\frac{3}{40}, \pm\frac{3}{40})$, $\mu = 1$, $L = 125$ inside of the $[-1, 1]^3$ box. Errors are again similar to those seen in the fast-decaying experiments from examples 1 and 4; timings are worse, as expected, since we are dealing with nine times as many degrees of freedom per point. Results are shown in in Table 7.

ϵ_{fmm}	ϵ_{rhs}	M_ℓ	N_{pts}	L_T	E_2	E_∞	T_{FMM}	Rate
Fourth-Order Force Approximation								
10^{-2}	10^{-2}	2038	130432	6	$1.3E-01$	$1.5E-01$	$1.3485E-01$	$9.67E+05$
10^{-4}	10^{-2}	2038	130432	6	$1.0E-03$	$8.4E-04$	$9.4899E-01$	$1.37E+05$
10^{-4}	10^{-4}	10606	678784	7	$9.1E-04$	$9.4E-04$	$5.1145E+00$	$1.33E+05$
10^{-6}	10^{-4}	10606	678784	7	$8.4E-06$	$1.1E-05$	$2.2359E+01$	$3.04E+04$
10^{-6}	10^{-6}	69140	4424960	8	$7.5E-06$	$1.4E-05$	$1.4655E+02$	$3.02E+04$
10^{-8}	10^{-6}	69140	4424960	8	$2.2E-07$	$4.4E-07$	$4.8311E+02$	$9.16E+03$
10^{-8}	10^{-8}	484408	31002112	9	$1.4E-07$	$4.4E-07$	$3.2253E+03$	$9.61E+03$
Sixth-Order Force Approximation								
10^{-4}	10^{-4}	2696	582336	7	$4.9E-04$	$8.6E-04$	$1.5720E+00$	$3.70E+05$
10^{-6}	10^{-4}	2696	582336	7	$4.3E-06$	$9.9E-06$	$5.9948E+00$	$9.71E+04$
10^{-6}	10^{-6}	10396	2245536	7	$8.1E-06$	$1.3E-06$	$2.3602E+01$	$9.51E+04$
10^{-8}	10^{-6}	10396	2245536	7	$1.6E-07$	$4.1E-07$	$7.7862E+01$	$2.88E+04$
10^{-8}	10^{-8}	59830	12923280	8	$1.4E-07$	$4.3E-07$	$4.1067E+02$	$3.15E+04$
10^{-10}	10^{-8}	59830	12923280	8	$5.4E-09$	$1.3E-08$	$1.0326E+03$	$1.25E+04$
10^{-10}	10^{-10}	295100	63741600	9	$5.2E-09$	$1.1E-08$	$5.3102E+03$	$1.20E+04$
Eighth-Order Force Approximation								
10^{-6}	10^{-6}	4894	2505728	7	$4.8E-06$	$1.5E-05$	$1.4287E+01$	$1.75E+05$
10^{-8}	10^{-6}	4894	2505728	7	$8.0E-08$	$4.3E-07$	$4.1362E+01$	$6.06E+04$
10^{-8}	10^{-8}	12860	6584320	7	$1.5E-07$	$4.5E-07$	$1.0268E+02$	$6.41E+04$
10^{-10}	10^{-8}	12860	6584320	7	$6.0E-09$	$1.5E-08$	$2.3717E+02$	$2.78E+04$
10^{-10}	10^{-10}	55854	28597248	8	$4.7E-09$	$1.6E-08$	$1.0489E+03$	$2.73E+04$
10^{-12}	10^{-10}	55854	28597248	8	$6.3E-09$	$8.8E-09$	$1.9641E+03$	$1.46E+04$
10^{-12}	10^{-12}	132490	67834880	9	$5.6E-09$	$7.0E-09$	$4.4599E+03$	$1.52E+04$

Table 7: Free-Space Stokes Equation, Example 5: Gaussian bump at the origin

Example 6. It is straightforward to extend the solver infrastructure described above to the case of periodic boundary conditions, using the the classical method of images of Lord Rayleigh [49], following the discussion of [23]. The influence of all separated image boxes can be incorporated using either a recursive approach [38] or a scheme based on lattice sums [30]. In either case, the additional work depends only on ϵ_{fmm} and not on the number of degrees of freedom. The main difference is that the unit cell B now has near neighbors, whose influence must be accounted for. This, too, has relatively little impact on performance. A small number of additional boxes are added to both the interaction and near neighbor lists, but no additional data structures are created; instead, everything is handled via careful book-keeping to minimize additional memory consumption.

As an example, we consider the periodic source function

$$f(\mathbf{x}) = CM^2\pi^2 \sin(\pi Mx) \sin(\pi My) \sin(\pi Mz),$$

for which the solution is

$$u(\mathbf{x}) = C \sin(\pi Mx) \sin(\pi My) \sin(\pi Mz).$$

We conduct our experiments for a non-trivial oscillatory force, choosing $C = 5$ AND $M = 7$ on the domain $[-1, 1]^3$ with varying degrees of depth and precision. We check the relative L_2 and L_∞ with results shown in Table 8.

It is straightforward to extend this approach to a variety of homogeneous Dirichlet, Neumann or mixed boundary conditions by the method of images as well with very little additional effort.

ϵ_{fmm}	ϵ_{rhs}	M_ℓ	N_{pts}	L_T	E_2	E_∞	T_{FMM}	Rate
Fourth-Order Force Approximation								
10^{-2}	10^{-2}	32768	2097152	6	$2.6E-02$	$3.3E-02$	$3.3250E-01$	$6.31E+06$
10^{-4}	10^{-4}	262144	16777216	7	$2.9E-04$	$7.6E-04$	$1.6296E+01$	$1.03E+06$
10^{-6}	10^{-6}	2097152	134217728	8	$5.6E-06$	$1.9E-05$	$2.9403E+02$	$4.56E+05$
Sixth-Order Force Approximation								
10^{-2}	10^{-2}	32768	7077888	6	$2.7E-02$	$3.4E-02$	$6.0730E-01$	$1.17E+07$
10^{-4}	10^{-4}	37248	8045568	7	$2.7E-04$	$8.2E-04$	$1.7895E+00$	$4.50E+06$
10^{-6}	10^{-6}	262144	56623104	7	$5.5E-06$	$1.8E-05$	$3.6250E+01$	$1.56E+06$
10^{-8}	10^{-8}	2097152	452984832	8	$1.0E-07$	$3.0E-07$	$7.2101E+02$	$6.28E+05$
Eighth-Order Force Approximation								
10^{-2}	10^{-2}	4096	2097152	5	$1.8E-02$	$2.8E-02$	$2.5445E-01$	$8.24E+06$
10^{-4}	10^{-4}	32768	16777216	6	$3.7E-04$	$9.5E-04$	$2.9459E+00$	$5.70E+06$
10^{-6}	10^{-6}	242432	124125184	7	$6.5E-06$	$2.0E-05$	$5.4881E+01$	$2.26E+06$
10^{-8}	10^{-8}	262144	134217728	7	$1.4E-07$	$5.3E-07$	$1.3578E+02$	$9.88E+05$
10^{-10}	10^{-10}	2097152	1073741824	8	$9.0E-09$	$4.3E-08$	$1.8685E+03$	$5.75E+05$

Table 8: Periodic Boundary Conditions Example, Example 6

Example 7. A number of applications require the modeling of source distributions that contain both a smooth component and a singular component. In electrostatics, for example, positively charged ions are often approximated as point charges and the neutralizing electrons as an inhomogeneous continuous background. We consider such a case here. The relevant Poisson equation takes the form

$$\Delta u(\mathbf{x}) = f_{smooth}(\mathbf{x}) + \sum_{i=1}^{N} q_i \delta(\mathbf{x} - \mathbf{x_i}),$$

where the q_i are positive and the neutralizing background takes the form of a sum of Gaussian distributions $f^i_{smooth}(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(\mathbf{x}-\sigma)^2/2\sigma^2}$ centered on each δ-function.

The smooth portion can be handled as above, while the particle sources can be handled with the corresponding particle-based kernel-independent FMM [61]. However, it is trivial to modify our solver to incorporate the particle sources into the *S2M* operator 4.2 by modifying equation (22).

$$\mathbf{K}^B_{S2M} \boldsymbol{\phi}^{B,u} = \mathbf{F}^B_{S2M} \gamma^B + \sum_{i=1}^{N} q_i G(\mathbf{x}, \mathbf{x_i}), \tag{42}$$

where G is the kernel used for evaluating the singular component, consisting of the point charges. Once the point charges are incorporated into $\boldsymbol{\phi}^{B,u}$, the rest of the components of the algorithm (i.e., M2M, M2L, and L2L) take care of the far-field interactions. We need only calculate the influence of near-field particle interactions directly, and evaluate both the local expansions (L2T) and the smooth contributions at particle locations. The latter is done by interpolation, as discussed in section 4.5.

The performance of our scheme is shown in Table 9.

The upward pass timings are minimally larger than for the particle-only case, and the downward pass timings are agnostic about the nature of the sources, dependent only on the tree-structure itself. The $Near$ computation timings are simply the sum of the particle and volume-based cases, since there is no amortization of cost in this step.

7 Conclusions

We have presented a kernel-independent FMM for solving a variety of constant-coefficient elliptic PDEs in free space, allowing for arbitrary levels of adaptivity, highly non-homogeneous forces and arbitrarily distributed target locations. Results for the Poisson, modified Helmholtz, and Stokes equations show that the performance is similar for each. Applying the method to other equations requires only a kernel evaluation routine.

Compared to the state-of-the-art technique [46], our method is higher order accurate and therefore solves similar problems with fewer degrees of freedom, and the work per point is approximately the same. Our current implementation uses OpenMP (but not MPI), although we expect the extension to be straightforward, as our solver is built

N_{pts}	ϵ_{fmm}	M_ℓ	L_T	$S2M/M2M$	$Near$	$M2L$	$L2L/L2T$	T_{FMM}
$1.024E+06$	10^{-2}	4096	5	$2.52E-02$	$6.42E-01$	$5.09E-02$	$4.84E-03$	$7.23E-01$
$1.024E+06$	10^{-4}	4096	5	$6.97E-02$	$6.49E-01$	$1.29E-01$	$2.35E-02$	$8.71E-01$
$1.024E+06$	10^{-6}	4096	5	$1.41E-01$	$6.50E-01$	$3.77E-01$	$6.66E-02$	$1.24E+00$
$1.024E+06$	10^{-8}	4096	5	$2.48E-01$	$6.41E-01$	$1.03E+00$	$1.58E-01$	$2.08E+00$
$4.096E+06$	10^{-2}	32768	6	$1.11E-01$	$1.58E+00$	$2.39E-01$	$1.85E-02$	$1.95E+00$
$4.096E+06$	10^{-4}	32768	6	$3.04E-01$	$1.61E+00$	$1.32E+00$	$1.01E-01$	$3.34E+00$
$4.096E+06$	10^{-6}	32768	6	$6.29E-01$	$1.61E+00$	$3.55E+00$	$2.92E-01$	$6.08E+00$
$4.096E+06$	10^{-8}	32768	6	$1.17E+00$	$1.60E+00$	$9.20E+00$	$6.48E-01$	$1.26E+01$
$1.638E+07$	10^{-2}	262144	7	$1.58E+00$	$4.31E+01$	$2.14E+00$	$2.58E-01$	$4.71E+01$
$1.638E+07$	10^{-4}	262144	7	$4.31E+00$	$4.34E+01$	$1.14E+01$	$1.48E+00$	$6.06E+01$
$1.638E+07$	10^{-6}	262144	7	$8.79E+00$	$4.38E+01$	$3.05E+01$	$4.20E+00$	$8.72E+01$
$1.638E+07$	10^{-8}	262144	7	$1.55E+01$	$4.32E+01$	$8.26E+01$	$9.96E+00$	$1.51E+02$
$6.554E+07$	10^{-2}	2097152	8	$7.00E+00$	$1.04E+02$	$1.80E+01$	$1.40E+00$	$1.31E+02$
$6.554E+07$	10^{-4}	2097152	8	$1.93E+01$	$1.05E+02$	$9.14E+01$	$6.76E+00$	$2.23E+02$
$6.554E+07$	10^{-6}	2097152	8	$4.01E+01$	$1.07E+02$	$2.61E+02$	$1.93E+01$	$4.27E+02$
$6.554E+07$	10^{-8}	2097152	8	$7.34E+01$	$1.06E+02$	$7.00E+02$	$4.19E+01$	$9.21E+02$

Table 9: Example 7: Poisson equation with a mixture of smooth and singular sources

on top of the MPI-based code of [61]. We discuss how the major loops are optimized for OpenMP shared-memory parallelization in section 8.4 for this current implementation.

As in [23], we have extended our solver to handle periodic, Dirichlet and Neumann boundary conditions for problems on cubic domains using the method of images. We are also coupling the present volume integral code with boundary integral methods to allow for the solution of linear, constant-coefficient, inhomogeneous elliptic PDEs in complex geometries, as in [59]. Additional current work involves incorporating this solver into the state-of-the-art in [42]. These extensions will be reported at a later date.

8 Appendix

In this appendix, we first verify numerically that the equivalent density representation yields the expected accuracy, followed by a discussion of how the choice of grids affects the order of convergence and an overall numerical justification for the use of Tikhonov regularization. We close with a discussion of how we accelerate the major computation loops of our algorithm using OpenMP shared-memory parallelization and load-balancing techniques.

8.1 Equivalent Density Accuracy

As discussed in section 4.1, we invert several matrices of discretized Fredholm equations of the first kind in order to build out far-field representations,

$$\mathbf{K}^{\mathbf{y}_d,\mathbf{x}_d}\phi_d = \mathbf{K}^{\mathbf{y}_s,\mathbf{x}_d}\phi_s.$$

As in [60], we choose to use Tikhonov regularization [40] when solving these ill-conditioned systems. This solves two problems: in this way, we eliminate the null space in the cases when it is present (Stokes kernel) and we significantly improve accuracy of the inversion for higher numbers of samples (Section 8.3). We verify the potential we get from $\phi^{B,u}$, computed using our regularized method in the $S2M$ operation, approximates well $u(\mathbf{x})$, computed directly from a force. We test using $g^B = \sum_{a,b,c}^{(a+b+c)\leq(k-1)} x^a y^b z^c$ for box B of width 2 and compute

$$u(\mathbf{x}) = \int_B K(\mathbf{x},\mathbf{y}) g^B(\mathbf{y}) d\mathbf{y},\ \mathbf{x} \in \mathbf{x}^{B,u},$$

to within 10^{-16} accuracy using adaptive Gaussian quadrature [8]. We then compute $\phi^{B,u}$ at $\mathbf{y}^{B,u}$ using (23) where $(\mathbf{K}_{S2M})^{-1}$ is replaced with $(\alpha I + (\mathbf{K}_{S2M})^*\mathbf{K}_{S2M})^{-1}\mathbf{K}^*_{S2M}$. For FMM precision, n_p, we choose $\alpha = 10^{-(n_p+1)}$. More details on the choice of α are available in [60]. Our algorithm relies on the fact that for surfaces outside the near field of B, $\phi^{B,u}$ is a sufficiently accurate representation of B's volume force. We compute

$$u(\mathbf{x})_{equiv} = \int_{\mathbf{y}^{B,u}} K(\mathbf{x},\mathbf{y})\phi^{B,u}(\mathbf{y})d\mathbf{y}$$

for $\mathbf{x} \in S$, some surface. To evaluate the accuracy of this approximation, we compute

$$u(\mathbf{x})_{exact} = \int_{B} K(\mathbf{x},\mathbf{y})g^{B}(\mathbf{y})d\mathbf{y}$$

up to an accuracy of 10^{-16} [8]. In figure 6, we compare the infinity-norm of the resulting error for three different kernels (Laplace, Modified Helmholtz, Stokes) and varying levels of the polynomial approximation and multiple degrees of FMM evaluation precision. For each of the kernels of interest, $\phi^{B,u}$, computed by inverting our ill-conditioned kernels, is recovered on each surface S to within the requested degree of precision. For evaluating the accuracy of the kernel inversion and regularization in the computation of $\phi^{B,d}$, we note that this computation is equivalent to the particle-based FMM, of which numerical analysis for the $M2L$ and $L2L$ operators is available in [60].

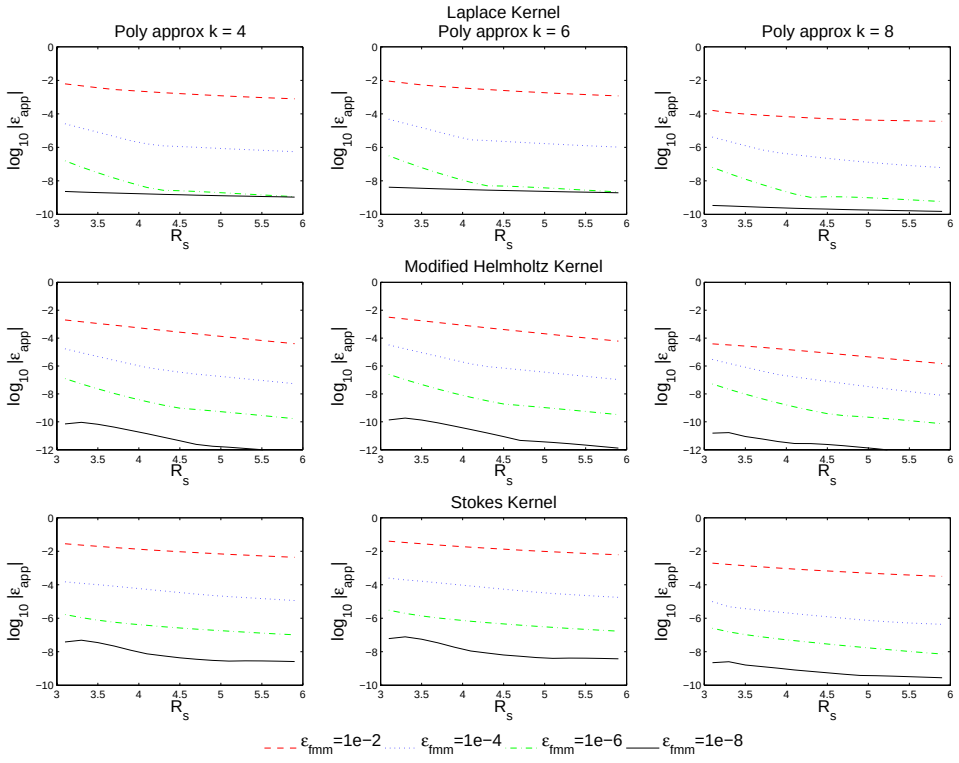


Figure 6: Error due to upward equivalent density approximation of the field. From left to right, three columns show the errors for the polynomial force approximations of degree 4, 6 and 8. Each plot shows four levels of FMM precision, $\epsilon_{fmm} = 10^{-n_p}$, $p = n_p^3 - (n_p - 2)^3$ points are used on the surfaces $\mathbf{y}^{B,u}$ and $\mathbf{x}^{B,u}$. For the evaluation surfaces S, we vary the radius R_S from 3.1 to 5.9, the region covering $L_I^B \in \mathcal{F}^B$. The y-axis of each plot is the infinity norm $||u_{equiv} - u_{exact}||_\infty$ computed over 488 samples on S.

8.2 Polynomial Basis and Grid Spacing

As discussed in section 4.4, we evaluate the solution at a leaf box B on a grid of points, $\mathbf{x}^{B,g}$ and construct an approximating polynomial from these points. Additionally, (section 4.6) we construct a k^{th}-order polynomial approximation to B's distributed force if g^B is given on a grid. For consistency with AMR codes and efficiency of implementation, it would have been desirable to use uniform grid samples. This approach works well for $n \leq 6$, but it is well-known for

large n that equispaced grids lead to instabilities [55]; as a result, for $n > 6$ we use Chebyshev grid points. To show that regularly-spaced grid points perform poorly for $n, k > 6$, we consider the following test case:

$$-\Delta u(\mathbf{x}) = e^{-L(||\mathbf{x}||_2)^2}\left(4L(||\mathbf{x}||_2)^2 - 6L\right), L = 250, \mathbf{x} \in [-1,1]^3$$

In figure 7, we compare the overall relative L^2 error, E_2, for solutions using equispaced and Chebyshev grid points in the evaluation of the solution and construction of the polynomial approximations of degree $4, 6$ and 8. Errors for discretizations using equispaced or Chebyshev grid points are similar for $k \leq$, but for $k = 8$, Chebyshev points are more accurate.

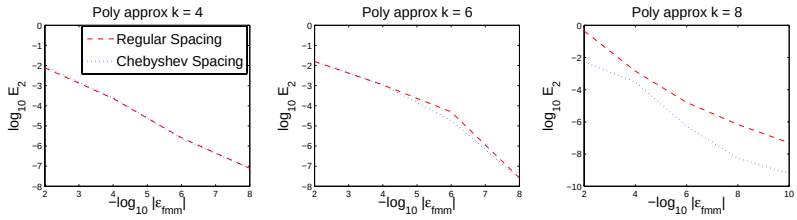


Figure 7: For each of the test examples, the x-axis indicates the negative log of the requested FMM accuracy, ϵ_{fmm}, and the y-axis indicates the log of E_2. The number of points chosen for each ϵ_{fmm} is equivalent to those in Example 1 of section 6 for $\epsilon_{rhs} = \epsilon_{fmm}$. *Left*: for polynomial approximation of degree 4 and $\mathbf{x}^{B,g}$ of size 4^3 on each leaf B, overall relative error is close for equispaced and Chebyshev points. *Middle:* For $n, k = 6$ differences are visible but insignificant. *Right:* For $n, k = 8$, solutions based on equispaced grid are less accurate.

8.3 Tikhonov Regularization

As discussed above in section 8.1, we use Tikhonov regularization [40] to invert Fredholm equations of the first kind, specifically the S2M, M2M, and L2L operators in section 4. Further, in section 8.1, we looked specifically at the accuracy resulting from this inversion process. To justify the overall use of Tikhonov regularization, we consider the following test cases for the Poisson and Stokes equations, respectively.

$$-\Delta u(\mathbf{x}) = e^{-L(||\mathbf{x}||_2)^2}\left(4L(||\mathbf{x}||_2)^2 - 6L\right), L = 250, \mathbf{x} \in [-1,1]^3$$

$$-\Delta u(\mathbf{x}) + \nabla p(\mathbf{x}) = \left(8L^3\,||\mathbf{x}-\mathbf{x}_i||^2 - 20L^2\right)e^{-L||\mathbf{x}-\mathbf{x}_i||^2} \quad \text{, and } [\nabla \times (\mathbf{x}-\mathbf{x}_i)], L = 125, \mathbf{x} \in [-1,1]^3.$$

In figure 8, we compare the overall relative L^2 error, E_2, solutions, resulting from Tikhonov regularization versus no regularization and the construction of polynomial approximations of degree $k = 6$ for the right-hand sides (errors for $k = 4, 8$ are similar). For decreasing levels of ϵ_{fmm}, we choose $\epsilon_{fmm} = \epsilon_{rhs}$.

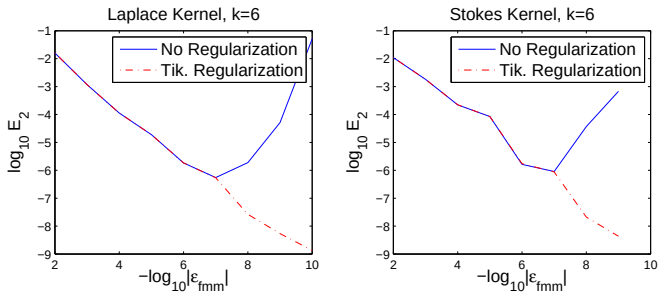


Figure 8: For each of the test examples, the x-axis indicates the negative log of the requested FMM accuracy, ϵ_{fmm}, and the y-axis indicates the log of E_2. The number of points chosen for each ϵ_{fmm} is equivalent to those in Examples 1 and 5 of section 6 for $\epsilon_{rhs} = \epsilon_{fmm}$. *Left*: for polynomial approximation of degree 6 for the Laplace kernel with and without regularization. *Right*: for polynomial approximation of degree 6 for the Stokes kernel with and without regularization.

We notice that for $\epsilon_{fmm} > 10^{-7}$, the effect of not employing regularization is equivalent to using regularization for both the Laplace and Stokes operators. However, as ϵ_{fmm} decreases, the number of sample points on the equivalent and check surfaces increases, resulting in larger linear systems, which as mentioned earlier, may be poorly conditioned. Indeed, for such larger systems resulting from $\epsilon_{fmm} \leq 10^{-7}$, it is necessary to regularize the systems to achieve desirable results.

8.4 Shared-Memory Parallelization and Load-Balancing

As indicated in section 7, we have designed the code to take advantage of shared-memory architectures through the use of OpenMP. In particular, we highlight the steps to accelerate the various major steps in Algorithm 1. For details on the nature of OpenMP and its usage, we refer the reader to [18].

S2M and M2M Computations In the upward pass (step 2 of Algorithm 1 and section 4.2), we begin by building a list of all leaf boxes, B in the octree T, which have sources. We then do a simple OpenMP parallelization step over these boxes for the $S2M$ step. As all components of equation (23) are of the same size for each leaf box, there is no need to rebalance the load among threads

In order to ensure proper order of computation, we proceed by sorting all non-leaves in reverse-order by depth. For each non-leaf level in T, beginning at the deepest level, we translate a box B's children's upward equivalent densities to its own through the $M2M$ computation in (25). Again, as each of the components is of the same size, there is no need to rebalance among threads. As we parallelize only among boxes at the same depth in T, level ℓ is not processed until $\ell + 1$ has completed. Further, once we have reached coarse level $\ell = 1$ (which only occurs for periodic or Dirichlet boundary conditions), we discontinue the parallelization.

M2L, L2L, and L2T Computations In the downward pass of Algorithm 1 (section 4.3), we perform a similar operation as above for the $M2M$ step. First, we sort all boxes B in T from the shallowest to deepest levels in the tree. For each level, ℓ, we parallelize among the boxes being processed at that level for the $M2L$ and $L2L$ computations. The $L2L$ components in equation (29) are of equivalent size for each box B; however, for each box B, the size of L_V^B vary greatly from other boxes (for example, this list is much smaller for boxes on the edge or corners of our domain). In order to ensure proper balancing among threads, we further sort all boxes for each level, ℓ by the size of L_V^B and then reorder the boxes such that the sum of all L_V^B for each thread is of roughly the same size.

For the $L2T$ computations in equation (31), we once again build a list of only leaf boxes, for which the target solution is desired, and we parallelize the computations in this list. The components of the discretized equation are all the same size, as with the $S2M$ computation, so there is no need to rebalance among threads for this step.

Near-Field Computations We focus our discussion here on the U-list computations. Parallelizing the near-field computations in equation (34) is the most straightforward in that no leaf box B is dependent on the completion of computations by any other box. That is, we can simply parallelize the computations among leaf boxes, for which the L_U^B exists. However, even more so than with the $M2L$ computations, the sizes of L_U^B can be very different among leaf boxes (especially in the most adaptively-refined octrees). Thus, we sort all leaf boxes B in T by the size of L_U^B and reorder the list of leaves such that the sum of the size of L_U^B among each thread is roughly equivalent, ensuring a relatively well-balanced load among threads. The size of the components and operators are the same for each box B, so balancing by list sizes is optimal. We note that this rebalancing is largely unnecessary for uniformly-refined trees.

Additionally, for matrix kernels (e.g. Stokes) and larger orders of polynomial approximation, constantly loading large matrices into memory results in little speedup as we increase the number of processes. In order to correct this, for each equivalence class as described in section 5, we perform all of the operations involving a single class first before performing all computations for other classes of operators. Hence, we only load each matrix operator at most once per processor.

We note that for the $M2L$ step, as we have to process level ℓ before moving to level $\ell - 1$, operators will constantly have to be reloaded. Performing all computations in order for each equivalence class at each level is done, but we have seen little time savings for this in practice as opposed to the near-field computations, where it is essential for good speedup.

Remark As with the near-field computations, for adaptively-refined trees, we rebalance the loads among threads for the X and W lists, which involve additional near-field $S2T$, $M2T$ and $S2L$ computations in equations (40) and (39),

based on the sizes of L_X^B and L_W^B, respectively. Additionally, we perform all computations in order of equivalence class, again loading each matrix operator at most once.

Timing Results Versus Number of Processors In order to see the effect of our use of OpenMP and load-balancing strategies, we investigate the strong scaling of two fixed problems. First, from Example 1 in section 6, we set the polynomial order, ϵ_{rhs}, and ϵ_{fmm} to 8. The reasoning behind this is to ensure that for a single processor, neither the near-field nor far-field computations fully dominate the timings. In table 10 we look at the timings for the different algorithmic steps (note that the near-field computation times include U,W,and X list computation times) as well as plot the decreasing times in figure 9a).

N_{procs}	$S2M/M2M$	$Near$	$M2L$	$L2L/L2T$	T_{FMM}	$Rate$
1	$1.1257E+00$	$8.5343E+00$	$1.4642E+01$	$9.3401E-01$	$2.5236E+01$	
2	$5.8800E-01$	$5.1123E+00$	$7.2851E+00$	$4.7791E-01$	$1.3463E+01$	$1.8744E+00$
4	$3.7504E-01$	$2.3771E+00$	$4.5594E+00$	$2.9275E-01$	$7.6042E+00$	$1.7705E+00$
8	$1.8386E-01$	$1.2319E+00$	$2.2799E+00$	$1.4595E-01$	$3.8416E+00$	$1.9794E+00$
16	$9.7009E-02$	$6.8944E-01$	$1.2409E+00$	$8.5467E-02$	$2.1128E+00$	$1.8183E+00$

Table 10: Timings (all in wall-time seconds) for the various components of the FMM volume solver for a fixed problem size for the Poisson equations. The polynomial-order, ϵ_{rhs}, and ϵ_{fmm} are set to 8. The number of leaves, $M_\ell = 5440$, the tree level, $L_T = 7$, and $N_{pts} = 2785280$ as seen in Example 1. N_{procs} indicates the number of processors, which we scale linearly. We separate the $S2M/M2M$, $Near$ (U,W,X-list computations), $M2L$ (V-list computations), $L2L/L2T$ timings with the total shown as T_{FMM}. The scaling $Rate$ is shown last.

For our second study of the effect of shared-memory parallelization, we look at the Stokes kernel tests from Example 5 in section 6. We fix the polynomial order to 8 and look at $\epsilon_{rhs} = \epsilon_{fmm} = 6$, again in an effort to not have one step fully dominate the computational time, allowing us to look at the effect of scaling the number of processors. Timing results can be seen in table 11 and figure 9b.

N_{procs}	$S2M/M2M$	$Near$	$M2L$	$L2L/L2T$	T_{FMM}	$Rate$
1	$8.7946E+00$	$5.4169E+01$	$9.4023E+01$	$6.8327E+00$	$1.6382E+02$	
2	$5.1822E+00$	$2.8574E+01$	$5.2139E+01$	$3.6178E+00$	$8.9513E+01$	$1.8301E+00$
4	$2.8669E+00$	$1.3076E+01$	$3.0299E+01$	$1.7337E+00$	$4.7976E+01$	$1.8658E+00$
8	$1.5697E+00$	$6.7810E+00$	$1.6137E+01$	$8.3674E-01$	$2.5325E+01$	$1.8944E+00$
16	$8.2480E-01$	$3.6117E+00$	$9.4520E+00$	$3.9180E-01$	$1.4287E+01$	$1.7726E+00$

Table 11: Timings for the various components of the FMM volume solver for a fixed problem size for the Stokes equations. The polynomial-order, ϵ_{rhs}, and ϵ_{fmm} are set to 6. The number of leaves, $M_\ell = 4894$, the tree level, $L_T = 7$, and $N_{pts} = 2505728$ as seen in Example 5. N_{procs} indicates the number of processors, which we scale linearly. We separate the $S2M/M2M$, $Near$ (U,W,X-list computations), $M2L$ (V-list computations), $L2L/L2T$ timings with the total shown as T_{FMM}. The scaling $Rate$ is shown last.

As can be seen in tables 10 and 11, our scheme exhibits the desirable result of nearly-linear speedup as we scale the number of processors. As indicated in the conclusion, current work is being done to incorporate this work with [42] in order to achieve parallelization on a significantly larger scale.

References

[1] M. F. Adams and J. Demmel. Parallel multigrid solver algorithms and implementations for 3D unstructured finite element problem. In *ACM/IEEE Proceedings of SC99: High Performance Networking and Computing*, Portland, Oregon, November 1999.

[2] M.J. Aftosmis, M.J. Berger, and J.E. Melton. Adaptive Cartesian mesh generation. In J.F. Thompson, editor, *The Handbook of Grid Generation*, pages 22–1–22–26. CRC Press, 1998.

[3] C.R. Anderson. A method of local corrections for computing the velocity field due to a distribution of vortex blobs. *J. Comput. Phys.*, 62(1):111–123, 1986.

[4] G.T. Balls and P. Colella. A finite difference domain decomposition method using local corrections for the solution of Poisson's equation. *J. Comput. Phys.*, 180(1):25–53, 2002.

[5] J. Barnes and P. Hut. A hierarchical O(N log N) force calculation algorithm. *Nature*, 324:446–449, December 1986. Technical report.

[6] R. Beatson and L. Greengard. A short course on fast multipole methods. In M. Ainsworth et al., editors, *Wavelets, multilevel methods and elliptic PDEs*, pages 1–37, Walton Street, Oxford OX2 6DP, UK, 1997. Oxford University Press.

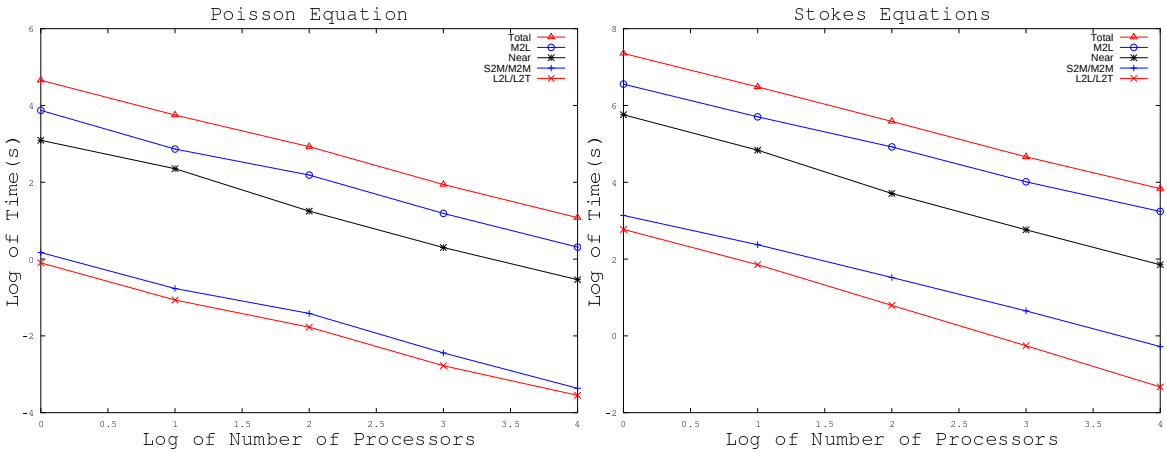


Figure 9: a) Log-log plot for timings from table 10, b) Log-log plot for timings from table 11.

[7] M.J. Berger, M. Aftosmis, and J. Melton. Accuracy, adaptive methods and complex geometry. In *1st AFOSR Conference on Dynamic Motion CFD*, 1996.

[8] J. Berntsen, T.O. Espelid, and A. Genz. Algorithm 698; dcuhre: an adaptive multidemensional integration routine for a vector of integrals. *ACM Trans. Math. Softw.*, 17(4):452–456, 1991.

[9] S. Börm. H2-matrix arithmetics in linear complexity. *Computing*, 77(1):1–28, 2006.

[10] S. Börm and W. Hackbusch. Hierarchical quadrature for singular integrals. *Computing*, 74(2):75–100, 2005.

[11] A.H. Boschitsch, M.O. Fenley, and W.K. Olson. A fast adaptive multipole algorithm for calculating screened coulomb (yukawa) interactions. *J. Comput. Phys.*, 151:212–241, 1999.

[12] A. Brandt. Multilevel adaptive solutions to boundary value problems. *Math. Comp.*, 31:333–390, 1977.

[13] L. Briggs, V. Emden Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, 2000.

[14] B.L. Buzbee, G.H. Golub, and C.W. Nielson. On direct methods for solving Poisson's equation. *SIAM J. Numer. Anal.*, 7:627–656, 1970.

[15] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics*. Springer–Verlag, New York, 1987.

[16] T. Chan, R. Glowinski, J. Périaux, , and O. Widlund. *Domain decomposition Methods*. SIAM, Philadelphia, 1989.

[17] T. Chan and B. Smith. Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes. *Electron. Trans. Numer. Anal.*, 1994:171–182, 1994.

[18] B. Chapman, G. Jost, and R. Pas. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007.

[19] H. Cheng, W.Y. Crutchfield, Z. Gimbutas, L. Greengard, J.F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao. A wideband fast multipole method for the Helmholtz equation in three dimensions. *J. Comput. Phys.*, 216:300–325, 2006.

[20] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *J. Comput. Phys.*, 155:468–498, 1999.

[21] H. Cheng, J. Huang, and T. J. Leiterman. An adaptive fast solver for the modified Helmholtz equation in two dimensions. *J. Comput. Phys.*, 211(2):616–637, 2006.

[22] G. Chesshire and W. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *J. Comput. Phys.*, 90:1–64, 1991.

[23] F. Ethridge and L. Greengard. A new fast-multipole accelerated Poisson solver in two dimensions. *SIAM J. Sci. Comput.*, 23(3):741–760, May 2001.

[24] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems: ACM Distringuished Dissertation*. MIT Press, 1988.

[25] L. Greengard. Fast algorithms for classical physics. *Science*, 265(5174):909–914, August 1994.

[26] L. Greengard and J. Huang. A fast direct solver for elliptic partial differential equations on adaptively refined meshes. *SIAM J. Sci. Comput.*, 21:1551–1566, 1999.

[27] L. Greengard and J. Huang. A new version of the fast multipole method for screened coulomb interactions in three dimensions. *J. Comput. Phys.*, 180:642–658, 2002.

[28] L. Greengard, M.C. Kropinski, and A. Mayo. Integral equation methods for Stokes flow and isotropic elasticity in the plane. *J. Comput. Phys.*, 125:403–414, 1996.

[29] L. Greengard and J.Y. Lee. A direct adaptive Poisson solver of arbitrary order accuracy. *J. Comput. Phys.*, 125:415–424, 1996.

[30] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325–348, August 1987.

[31] L. Greengard and V. Rokhlin. The rapid evaluation of potential fields in three dimensions. In C. Anderson and C. Greengard, editors, *Vortex Methods*, Lecture Notes in Mathematics. Springer Verlag, N.Y., 1988.

[32] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numer.*, 6:229–269, 1997.

[33] N.A. Gumerov and R. Duraiswami. Fast multipole method for the biharmonic equation in three dimensions. *J. Comput. Phys.*, 215:363–383, 2006.

[34] W. Hackbusch. A sparse matrix arithmetic based on h-matrices. part i: introduction to h-matrices. *Computing*, 62(2):89–108, 1999.

[35] W. Hackbusch and S. Börm. H2-matrix approximation of integral operators by interpolation. *Appl. Numer. Math.*, 43(1-2):129–143, 2002.

[36] W. Hackbusch and Z.P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numerische Mathematik*, 54(4):463–491, 1989.

[37] W. Hackbusch and U. Trottenberg. *Multigrid Methods, Lecture Notes in Mathematics Volume 960*. Springer-Verlag, 1st edition, 1982.

[38] J. Helsing. Fast and accurate calculations of structural parameters for suspensions. *Proc. Roy. Soc. Lond. A*, 445:127–140, 1994.

[39] H. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 147:60–85, 1998.

[40] R. Kress. *Linear Integral Equations: Applied Mathematical Sciences 82*. Springer-Verlag, 2nd edition, 1999.

[41] O.A. Ladyzhenskaya. *The Mathematical Theory of Viscous Incompressible Flow*. Gordon and Breach, 2nd edition, 1964.

[42] I. Lashuk, A. Chandramowlishwaran, M.H. Langston, T.A. Nguyen, R.S. Sampath, A. Shringarpure, R.W. Vuduc, L. Ying, D. Zorin, and G. Biros. A massively parallel adaptive fast-multipole method on heterogeneous architectures. In *SC'2009 Conference*, Portland, OR, nov 2009. IEEE/ACM SIGARCH.

[43] D. Martin and K. Cartwright. Solving Poisson's equations using adaptive mesh refinement. Technical Report M96/66, University of California, Berkeley Electronic Research Laboratory, 1996.

[44] D.J. Mavripilis. Unstructured grid techniques. *Annu. Rev. Fluid Mech.*, 29:473–514, 1997.

[45] A. Mayo. Fast high order accurate solution of Laplace's equation on irregular regions. *SIAM J. Sci. Comput.*, 6(1):144–157, January 1985.

[46] P. McCorquodale, P. Colella, G.T. Balls, and S.B. Baden. A local corrections algorithm for solving Poisson's equation in three dimensions. *Comm. Appl. Math. Comput.*, 2(1):57–81, 2007.

[47] A. McKenney, L. Greengard, and A. Mayo. A fast Poisson solver for complex geometries. *J. Comput. Phys.*, 118:348–355, 1995.

[48] M.L. Minion. A projection method for locally refined grids. *J. Comput. Phys.*, 148(1):81–124, 1999.

[49] Lord Rayleigh. On the influence of obstacles arranged in rectangular order upon the properties of a medium. *Phil. Mag.*, 34:481–502, 1892.

[50] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.*, 60:187–207, 1985.

[51] V. Rokhlin. Rapid solution of integral equations of scattering theory in two dimensions. *J. Comput. Phys.*, 86:414–439, 1990.

[52] M.C. Strain, G.E. Scuseria, and M.J. Frisch. Achieving linear scaling for the electronic quantum Coulomb problem. *Science*, 271:51–53, 1996.

[53] H. Sundar, R. Sampath, and G. Biros. Bottom-up construction and 2:1 balance refinement of linear octrees in parallel. *Submitted: SIAM J. Sci. Comput.*, 2007.

[54] A.K. Tornberg and L. Greengard. A fast multipole method for the three-dimensional Stokes equations. *J. Comput. Phys.*, 227(3):1613–1619, 2008.

[55] L.N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[56] H. Wang, T. Lei, J. Li, J. Huang, and Z. Yao. A parallel fast multipole accelerated integral equation scheme for the 3d stokes equations. *J. Numer. Methods Engrg.*, 70:812–839, 2007.

[57] C.A. White, B.G. Johnson, P.M.W. Gill, and M. Head-Gordon. The continuous fast multipole method. *Chem. Phys. Lett.*, 230:8–16, November 1994.

[58] G.J. Rodin Y. Fu. Fast solution method for three-dimensional Stokesian many-particle problems. *Comm. Numer. Methods Engrg.*, 16(2):145–149, 2000.

[59] L. Ying, G. Biros, and D. Zorin. A fast solver for the Stokes equations with distributed forces in complex geometries. *J. Comput. Phys.*, 194(1):317–348, 2004.

[60] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole method in two and three dimensions. *J. Comput. Phys.*, 196(2):596–626, 2004.

[61] L. Ying, G. Biros, D. Zorin, and M.H. Langston. A new parallel kernel-independent fast multipole method. In *SC'2003 Conference CD*, Phoenix, AZ, November 2003. IEEE/ACM SIGARCH.