The problem of an FMM for axisymmetric density distributions

May 4, 2016

1 Introduction

This project examines the problem of creating a fast multipole method (FMM) algorithm for an important category of non-uniform density distributions: axisymmetric surfaces of revolution.

In this project, we will work in three-dimensional cylindrical coordinates (r, θ, z) such that a point in Cartesian coordinates (x, y, z) is represented by:

$$r = \sqrt{x^2 + y^2} \tag{1}$$

$$\theta = \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0 \\ \arcsin(\frac{y}{r}) & \text{if } x \geq 0 \\ \arctan(\frac{y}{x}) & \text{if } x > 0 \\ -\arcsin(\frac{y}{r}) + \pi & \text{if } x < 0 \end{cases} \tag{2}$$

$$z = z \tag{3}$$

2 Statement of problem

An axisymmetric, or rotationally symmetric, surface is one that has symmetry along the θ-axis. An axisymmetric surface Γ is obtained by rotating a two-dimensional curve γ about the z axis. γ is called the generating curve. In particular, $\Gamma = \gamma \times \mathbb{T}$ where $\mathbb{T}$ is the one-dimensional torus (circle) parametrized by $\theta \in (-\pi, \pi]$.

Consider P source points $\mathbf{y}_j$ distributed on this surface Γ, with each assigned a charge σ_j of ± 1. We want to compute the potential at some target point (in practice the sources and targets are simply the same points) represented by, in the continuous case, the integral equation

$$f(\mathbf{x}) = \int_\Gamma K(\mathbf{x}, \mathbf{y}) \sigma(\mathbf{y}) d\mathbf{y} \qquad \mathbf{y} \in \Gamma \tag{4}$$

that we discretize

$$f(\mathbf{x}_i) = \sum_{j=1}^{P} K(\mathbf{x}_i, \mathbf{y}_j)\sigma_j \tag{5}$$

for $\mathbf{x}_i = (r, \theta, z)$ target points and $\mathbf{y}_j = (r', \theta', z')$ source points, where $K(\mathbf{x}, \mathbf{y})$ is the kernel, typically the Green's function of the partial differential equation that governs the relationship between sources and targets. In particular, we are interested in the free-space Green's function for Laplace's equation:

$$K(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi|\mathbf{x} - \mathbf{y}|} \tag{6}$$

Our goal is to construct a fast summation method for equation (5). In the next sections we look at the ways a variety of fast multipole methods attack the fast computation of potentials in this scenario, and discuss why they aren't optimal.

Density distributions on axisymmetric surfaces such as this have many applications. For example, the problem of electromagnetic scattering by a surface of revolution has radar, geophysical exploration, and acoustics applications. This type of problem relies specifically on creating a fast algorithm for a set of densities (charges, weights, etc.) distributed on an axisymmetric surface.

3 Modal Green's Function

A common strategy to construct a fast method for a density distribution on an axisymmetric surface is to reduce the problem from one in three dimensions to a series of problems in two dimensions using Fourier analysis. This strategy is grounded in the fact that it is easier to solve boundary integral equations defined on curves in $\mathbb{R}^2$ than those defined on surfaces in $\mathbb{R}^3$.

We are able to reduce the problem because the rotationally symmetric nature of the surface yields $K(\mathbf{x}, \mathbf{y})$ symmetric along the θ-axis such that we the kernel is only a function of the difference between θ and θ':

$$K(\mathbf{x}, \mathbf{y}) = K(\theta - \theta', r, z, r', z') \tag{7}$$

We call such a kernel rotationally invariant.

Recall that if for two any points (r, θ, z) and (r', θ', z') in the computational domain, a kernel can be written

$$K(\mathbf{x}, \mathbf{y}) = K(r, \theta, z, r', \theta', z') = K(\theta - \theta', r - r', z - z') \tag{8}$$

then it is called translation invariant. This is not the case for all kernels, however, and that fact is crucial to the difficulty of creating an FMM in this scenario.

3.1 Fourier representation of 3D integral equations

Consider the Fredholm integral equation of the first kind defined on the axisymmetric surface Γ in three dimensions, similar to §1:

$$f(\mathbf{x}) = \int_\Gamma K(\mathbf{x},\mathbf{y})\sigma(\mathbf{y})d\mathbf{y} \qquad \mathbf{y} \in \Gamma \tag{9}$$

where K is a kernel function, σ are the charges, and f is the potential.

We can restate this as a sequence of integral equations defined on the generating curve γ of Γ by performing a Fourier transform. If σ_n, f_n, and k_n are the Fourier modes of σ, f, and K, then we have

$$\sqrt{2\pi}\int_\gamma k_n(r,z,r',z')\sigma_n(r',z')r'dl(r',z') = f_n(r,z) \qquad (r',z') \in \gamma \tag{10}$$

for each $n \in \mathbb{Z}$, such that

$$f_n(r,z) = \int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}} f(r,z,\theta)d\theta \qquad f(\mathbf{x}) = \sum_{n\in\mathbb{Z}} \frac{e^{in\theta}}{\sqrt{2\pi}} f_n(r,z) \tag{11}$$

$$\sigma_n(r,z) = \int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}} \sigma(r,z,\theta)d\theta \qquad \sigma(\mathbf{x}) = \sum_{n\in\mathbb{Z}} \frac{e^{in\theta}}{\sqrt{2\pi}} \sigma_n(r,z) \tag{12}$$

$$k_n(r,z,r',z') = \int_{\mathbb{T}} \frac{e^{-in\theta}}{\sqrt{2\pi}} K(r,z,r',z',\theta)d\theta \quad K(\mathbf{x},\mathbf{y}) = \sum_{n\in\mathbb{Z}} \frac{e^{in(\theta-\theta')}}{\sqrt{2\pi}} k_n(r,z,r',z') \tag{13}$$

As stated before we are using the Green's function of a PDE as our kernel, so k_n is called the modal Green's function. In particular, we are interested in the Fourier expansion of the free-space Green's function for Laplace's equation:

$$\frac{1}{4\pi|\mathbf{x}-\mathbf{y}|} = \frac{1}{4\pi\sqrt{r^2+r'^2-2rr'\cos(\theta-\theta')+(z-z')^2}} \tag{14}$$

$$= \sum_{n\in\mathbb{Z}} \frac{e^{in(\theta-\theta')}}{\sqrt{2\pi}} s_n(r,z,r',z') \tag{15}$$

where the modal Green's function is

$$s_n(r,z,r',z') = \frac{1}{\sqrt{8\pi^3 rr'}} \mathbf{Q}_{n-\frac{1}{2}}\left(\frac{r^2+(r')^2+(z-z')^2}{2rr'}\right) \tag{16}$$

with $Q_{n-\frac{1}{2}}$ is the half-integer order Legendre function of the second kind. The efficient evaluation of modal Green's function for Laplace's equation, and the Q-function in particular, has been explored by Abdelmageed (see equation 39), and others. These should

be implemented to quickly evaluate the kernel at the different Chebyshev nodes. We are concerned not only with the efficient evaluation of these analytic expansions, however, but also the fast evaluation of their sums.

Now rather than constructing a fast summation for a three-dimensional density distribution using $K(\mathbf{x}, \mathbf{y})$, we can construct a series of fast summations for a two-dimensional density distribution using $s_n(r, z, r', z')$. However, notice s_n is only rotationally invariant - not translation invariant. This has implications for the computation of translation operators which we discuss later.

We should note that for now we are only considering the Green's function for Laplace's equation because its Fourier modes can be solved for analytically. For many other kernels, it is not possible to find the same type of explicit formula, and we would need to approximate the modes using a discretization.

We should also note that for densities distributed on the axisymmetric surface Γ the computational domain was some subset of $\mathbb{R}^3$. For the densities distributed on the generating curve γ, however, we only consider the right half-plane of $\mathbb{R}^2$ as our possible computational domain.

In the next few sections, we discuss several approaches to an FMM for axisymmetric surfaces.

4 Analytic FMM

The original FMM first developed by Greengard and Rokhlin relies on analytic expansions for the potential at some target in the far field due to sources in a box (multipole) and the potential at a target in a box due to all sources in the far field (local). Originally this was done using Legendre polynomials and spherical harmonics to represent $\frac{1}{|\mathbf{x}-\mathbf{y}|}$.

The full three-dimensional problem is currently not possible as stated with an analytic FMM since we do not yet have multipole or local expansions for the single-layer Laplace kernel in three-dimensional cylindrical coordinates.

We can consider an analytic FMM for the series of modal Green's functions, but again the analytic expansion has not been determined. In addition, this method will still require the evaluation of special functions. Why is this bad/expensive?

In addition, a problem arises when we have many sources distributed in close proximity. (why is this the case again??)

5 Kernel-Independent Methods

Ruling out an analytic FMM to accelerate the sum in question, we're left to explore so-called kernel-independent methods. These methods can be separated into two categories. First, the not so kernel-independent methods which require that the kernel is the Green's function for some elliptic partial differential equation. The most well-known method of

this type is the kernel-independent FMM (KIFMM), which was proposed by Ying, Biros, and Zorin. Second are methods which work with any smooth kernel and require only its evaluation at some points. The most well-known method of this type is the black-box FMM (bbFMM) by Fong and Darve.

In the first type of method, we want to use the properties of the PDE to construct translation operators. Consider an exterior Dirichlet problem. There are three approaches we can use to represent the potential in the far-field: 1) use Green's third identity – differential equations, 2) use integral equations, or 3) create equivalent densities and potentials. 3) is the choice of YBZ, and further detail of the construction of these operators is explored below. Essentially, an algorithm of this type depends on the source to multipole translation, and choice of a discretization scheme.

6 Translation Operators

Recall that in the analytic FMM, a translation operator translates the multipole or local expansion of one box to that of another in the computational domain. In kernel-independent methods, translation operators translate equivalent weights or densities in the same way. Translation operators are the backbone of any fast multipole method because these operations reduce the number of direct computations required to compute the sum.

If a kernel is translation invariant, then translation operators will only differ based on relative position and level in the hierarchical tree. This is because the kernels used there only depended the relative difference between coordinate values, e.g. $(r - r', \theta - \theta', z - z')$.

Translation operators can be computed without any knowledge of the sources or targets, so they are a true pre-computation. The only thing we'll need is a computational domain. We choose arbitrarily the box $[-1, 1] \times [-1, 1]$, although operators can be calculated for any rectangle.

There are five types of translation operators. The source-to-multipole (S2M) operator creates a multipole expansion for a given box. The multipole-to-multipole (M2M) operator translates densities in a box to densities in its parent box. The multipole-to-local (M2L) operator translates densities in a box to another box on the same level in the hierarchical tree that divides the computational domain. The local-to-local (L2L) operator translates densities in a box to densities in a child box. Lastly, the local-to-target (L2T) operator computes the far field interaction for the target points using the local expansion.

7 The kernel-independent FMM

The fast multipole method (FMM) has been applied to many different kernels. It has also been applied to many different density distributions. There are different flavors of the FMM to deal with difficult kernels for which there aren't workable analytic expansions. These so-called kernel-independent FMMs get around this issue. The original KIFMM

uses a continuous distribution of an equivalent density on a surface enclosing a box in the hierarchical tree to represent the potential generated by sources in that box, rather than using analytic multipole expansions as in the original FMM. This allows us to construct an efficient FMM that only requires kernel evaluations. The KIFMM is also relatively easy to implement, since in general it applies to an arbitrary kernel that is the fundamental solution of some elliptic PDE. To change the kernel in the original FMM, one would need to develop analytic multipole expansions for that kernel that may be difficult to produce.

In the KIFMM, all translation operators involve a kernel evaluation.

Rather than relying on analytic multipole and local expansions, the KIFMM by Ying, Biros, and Zorin, substitutes upward and downward equivalent densities that lie on surfaces surrounding each box. The idea is that to a target box in the far field, the potential due to the source charges is the same as the potential due to these equivalent densities.

The original KIFMM paper explored three data sets for the 3D case: densities distributed on the unit sphere, densities distributed uniformly on the unit cube, and densities distributed at the eight corners of the unit cube.

While the 3D KIFMM may be able to handle densities distributed on surfaces of revolution, [5] explains that the implementation is difficult and rather than using circular equivalent surfaces as in the 2D case, we need to use cubes. Due to this difficulty, the technique described earlier and covered at length in [6] is used – replacing the 3D integral equations required by the 3D KIFMM with their Fourier representations, sequences of 2D integral equations. In this way, the 3D KIFMM implementation is avoided in favor of repeatedly applying the 2D KIFMM. As mentioned earlier, this strategy is currently only applicable to the single-layer Laplace kernel because we can analytically determine its Fourier modes.

Martinsson, Hao, and Young explored this application to modal green's functions. However, the pre-computation is not optimal because of the translation operators. Its application to our problem is difficult because the pre-computation will be expensive. There isn't any constant proportionality between the operators and so each operator will have to be computed individually which would be O().

7.1 Application

KIFMM with modal green's functions (Martinsson, Young, Hao)

Its application to our problem is difficult because the pre-computation will be expensive. There isn't any constant proportionality between the translation operators and so each operator will have to be computed individually for every box on every level. This is expensive because every operator involves the evaluation of the kernel.

It's important to note that if the kernel is translation invariant, only depending on $(r - r', \theta - \theta', z - z')$, then translation operators only differ based on relative position and level in the hierarchical tree. Many of them, therefore, are the same, and need not be computed repeatedly. For example, on the same level, an M2L operator between a

box and the box one box length above it is the same regardless of where these boxes are located in the computational domain. As mentioned earlier, for the Green's function for Laplace's equation in cylindrical coordinates, however, we are not so lucky. The only way to reduce this computational cost would be to find constant relationships between these operators. However, attempts to find constant relationships between the translation operators to minimize the pre-computation failed, and these wouldn't be expected anyway due to the nature of the translation variant kernel. So in this method, we need to compute every possible translation operator in the entire computational domain.

The upward and downward formulation of the aforementioned equivalent densities and their translation are explained in detail in §3.2 and shown in Figures 3, 4, 5, and 6.

In this section we use the Fourier representation of surface integral equations described in §2.2 together with the 2D KIFMM to create a fast algorithm for densities distributed on axisymmetric surfaces of revolution. Notation in this section for the surfaces follows [5].

Notice that k_n does not depend only on the difference between each coordinate, e.g. $(r - r')$. This has implications for the translation operators we construct for the KIFMM which is discussed in §3.2.4.

7.2 Full algorithm

Recall that in the 3D KIFMM, we needed to solve surface integral equations like:

$$\text{S2M: } \int_{\mathbf{y}^{B,u}} K(\mathbf{x},\mathbf{y})\phi^{B,u}(\mathbf{y})d\mathbf{y} = \sum_{i\in I_s^B} K(\mathbf{x},\mathbf{y}_i)\phi_i \text{ for all } \mathbf{x}\in\mathbf{x}^{B,u} \tag{17}$$

We can write (15) as a sequence of 2D equations, (16), by using the Fourier representation explained in §2.2. If this is unclear, equation (15) is to (16) as equation (1) is to (3). In calculations below, I will skip this derivation and just state the sequences of 2D equations.

Now that we have our kernel k_n, we proceed through the standard 2D KIFMM algorithm for the sources on the 2D generating curve γ **for each** $n \in [-N, -N+1, \ldots, N]$ where N is the truncation parameter chosen earlier in (10). The KIFMM algorithm is very similar to the FMM algorithm described in [1], apart from how the equivalent densities are represented, and how the translation operators are computed. As mentioned earlier, rather than by multipole expansions, the potential due to sources in a box is matched to an equivalent density at discretization points on a surface enclosing the box. In 2D, these surfaces are circles with radii prescribed in [5]. To compute these equivalent densities, we will need to discretize several integral operators on different surfaces, which is explained in §3.2.2.

In the following equations, $\mathbf{x} = (r, z)$ and $\mathbf{y} = (r', z')$. Also, please note the seemingly out-of-place r' term under each integral, and recall that this is actually part of the integral operator K_n as in (3).

7.2.1 Equivalent densities

After partitioning the hierarchical tree with no more than a prescribed number of sources in each box, compute the upward equivalent density for each leaf box. Similar to the **S2M** step in the FMM, solving the following equation for $\phi^{B,u}$ gives the upward equivalent density for a box B in the KIFMM:

$$\text{S2M: } \int_{\mathbf{y}^{B,u}} k_n(\mathbf{x},\mathbf{y})\phi_n^{B,u}(\mathbf{y})r'd\mathbf{y} = \sum_{i\in I_s^B} k_n(\mathbf{x},\mathbf{y}_i)\phi_i r_i' \text{ for all } \mathbf{x}\in\mathbf{x}^{B,u} \tag{18}$$

$$\text{Discretized S2M: } M_n\phi_n^{B,u} = q_n^{B,u} \tag{19}$$

where in (17), M_n is the matrix of kernel evaluations of pairs of discretization points on the upward check surface and upward equivalent surface of a box B, $q_n^{B,u}$ is the upward check potential, and the densities at the actual source points in the box are ϕ_i. This is similar to (7), and this process is illustrated in Figure 3. Once the upward equivalent density is computed for each leaf box, we use translation operators to find the upward and downward equivalent densities for every other box.

Figure 1: Left: The upward check (blue) and equivalent (red) surfaces of a box used to compute the upward equivalent density due to source densities (black) in the box. Right: The downward check (blue) and equivalent (red) surfaces of a box used to compute the downward equivalent density due to sources densities (black) in the far field on the box. In the algorithm, we never actually directly compute a downward equivalent density.

7.2.2 Translation Operators

Translation operators limit the number of equivalent densities we need to compute directly by translating upward equivalent densities of the leaf boxes to upward and downward equivalent densities of all other boxes. In the KIFMM in particular, the translation operators in their discretized form are matrices that translate an equivalent density from one box to that of another. They are a pre-computation, as they are constant regardless of the source distribution.

In the previous step, we computed the upward equivalent density for each leaf box. The **M2M** operator translates the upward equivalent density from a leaf box A to the upward equivalent density of its parent box B. Solving the following equation for $\phi_n^{B,u}$ gives the

M2M operator.

$$\text{M2M: } \int_{\mathbf{y}^{B,u}} k_n(\mathbf{x},\mathbf{y})\phi_n^{B,u}(\mathbf{y})r'd\mathbf{y} = \int_{\mathbf{y}^{A,u}} k_n(\mathbf{x},\mathbf{y})\phi_n^{A,u}(\mathbf{y})r'd\mathbf{y} \text{ for all } \mathbf{x} \in \mathbf{x}^{B,u} \tag{20}$$

$$\text{Discretized M2M: } M_n^B\phi_n^{B,u} = M_n^A\phi_n^{A,u} \tag{21}$$

where M_n^A is the matrix of kernel evaluations of pairs of discretization points on the upward check surface of box B and the upward equivalent surface of box A, and M_n^B is the matrix of kernel evaluations of pairs of discretization points on the upward check surface of box B and the upward equivalent surface of box B. Figure 4 gives a graphical representation of this step.

So the M2M translation operator is:

$$T^{M2M} = (M_n^B)^{-1}M_n^A \tag{22}$$

$$= \begin{pmatrix} K(\mathbf{x_1},\mathbf{y_1}) & \cdots & K(\mathbf{x_1},\mathbf{y_m}) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x_m},\mathbf{y_1}) & \cdots & K(\mathbf{x_m},\mathbf{y_m}) \end{pmatrix}^{-1} \begin{pmatrix} K(\mathbf{x_1},\mathbf{y_1}) & \cdots & K(\mathbf{x_1},\mathbf{y_m}) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x_m},\mathbf{y_1}) & \cdots & K(\mathbf{x_m},\mathbf{y_m}) \end{pmatrix} \tag{23}$$

Figure 2: The upward equivalent density due to the source densities (black) in the child box computed from the upward equivalent (red) and check (blue) surfaces of the child box is translated to the upward equivalent density of the parent box, checking against its upward check (blue) and equivalent (red) surfaces.

After repeating that step, every box now has an upward equivalent density. The **M2L** operator translates the upward equivalent density from a non-leaf box A to the downward equivalent density of a box B on the same level.

$$\text{M2L: } \int_{\mathbf{y}^{B,d}} k_n(\mathbf{x},\mathbf{y})\phi_n^{B,d}(\mathbf{y})r'd\mathbf{y} = \int_{\mathbf{y}^{A,u}} k_n(\mathbf{x},\mathbf{y})\phi_n^{A,u}(\mathbf{y})r'd\mathbf{y} \text{ for all } \mathbf{x} \in \mathbf{x}^{B,d} \tag{24}$$

$$\text{Discretized M2L: } M_n^B\phi_n^{B,d} = M_n^A\phi_n^{A,u} \tag{25}$$

where M_n^A is the matrix of kernel evaluations of pairs of discretization points on the downward check surface of box B and the upward equivalent surface of box A, and M_n^B is the matrix of kernel evaluations of pairs of discretization points on the downward check surface of box B and the downward equivalent surface of box B.

Figure 3: The upward equivalent density due to the source densities (black) in a box computed from the upward equivalent (red) and check (blue) surfaces of the child box is translated to the downward equivalent density of another box on the same level, checking against its upward check (blue) and equivalent (red) surfaces.

After repeating that step, every non-leaf box has a downward equivalent density. The **L2L** operator translates the downward equivalent density of a non-leaf box A to the downward equivalent density of a child box B.

$$\text{L2L: } \int_{\mathbf{y}^{B,d}} k_n(\mathbf{x},\mathbf{y})\phi_n^{B,d}(\mathbf{y})r'd\mathbf{y} = \int_{\mathbf{y}^{A,d}} k_n(\mathbf{x},\mathbf{y})\phi_n^{A,d}(\mathbf{y})r'd\mathbf{y} \text{ for all } \mathbf{x} \in \mathbf{x}^{B,d} \tag{26}$$

$$\text{Discretized L2L: } M_n^B \phi_n^{B,d} = M_n^A \phi_n^{A,d} \tag{27}$$

where M_n^A is the matrix of kernel evaluations of pairs of discretization points on the downward check surface of box B and the downward equivalent surface of box A, and M_n^B is the matrix of kernel evaluations of pairs of discretization points on the downward check surface of box B and the downward equivalent surface of box B. After repeating this step, every box now has an upward and downward equivalent density.

Figure 4: The downward equivalent density due to the source densities (black) in a parent box computed from the upward equivalent (red) and check (blue) surfaces of the parent box is translated to the downward equivalent density of a child box, checking against its upward check (blue) and equivalent (red) surfaces.

Once we have these upward and downward equivalent densities for every $n \in [-N, \ldots, N]$, then we can reconstruct the 3D equivalent densities ϕ_{approx} by summing as in (10). This summing can be accelerated via the FFT. Having these equivalent densities in 3D, we can complete the last step of the KIFMM algorithm by evaluating the far- and then near-field interactions.

Note that all requirements for smoothness and uniqueness of the solution to the integral equations in the KIFMM listed in §3.1.5 of [5] (mostly pertaining to the sizes and positions of the surfaces) are satisfied here.

7.2.3 Discretization details

The equations in §3.2.1 are discretized using p points on the equivalent and check surfaces. p is constant for all boxes, and the choice of p determines the error level of the compu-

tations.This discretization requires two steps. First, the right hand side is the evaluation of a check potential. This step checks that the potential represented by the equivalent density and the actual source densities are the same to all boxes in the far field. Then on the left hand side, we invert a Dirichlet-type boundary integral equation to obtain the equivalent density. To stably solve this equation, as in [5], we use Tikhonov regularization with regularization parameter $\alpha = 10^{-12}$. Essentially, each translation is simply applying a series of matrices.

For example, the M2L operator is the matrix T_n^{M2L} for a mode n is obtained by solving (20):

$$\phi_n^{B,d} = \left[\left[\alpha I + (M_n^B)^T M_n^B\right](M_n^B)^T M_n^A\right]\phi_n^{A,u} \tag{28}$$

$$\implies T_n^{M2L} = \left[\left[\alpha I + (M_n^B)^T M_n^B\right](M_n^B)^T M_n^A\right] \tag{29}$$

8 The black-box FMM

The black-box FMM is a Chebyshev interpolation-based $\mathcal{O}(P)$ algorithm for non-oscillatory kernels. In this section, we consider using this scheme to quickly compute this sum for each modal Green's function

$$f_n(\mathbf{x}_i) = \sum_{j=1}^{P} s_n(\mathbf{x}_i, \mathbf{y}_j)\sigma_{n,j} \tag{30}$$

for targets $\mathbf{x}_i$ and sources $\mathbf{y}_j$. The bbFMM is ideal for complicated analytic kernels, perhaps without translation invariance, or if it has a very complicated analytic form that isn't easy to work with in the standard FMM or KIFMM. The bbFMM only requires the evaluation of the kernel at specific points. The other advantage for this method is that it has a small pre-computation even for large systems. The pre-computation for any kernel is $\mathcal{O}(P)$ where P is the number of source points. This is crucial for this problem since the pre-computation for the KIFMM was expensive because of the translation variant kernel.

The full method combines the methods of SVD compression from...

$$s_n(\mathbf{x}, \mathbf{y}) = \sum_i \alpha_i u_i(\mathbf{x}) v_i(\mathbf{y}) \tag{31}$$

and Chebyshev interpolation as in....

$$s_n(\mathbf{x}, \mathbf{y}) = \sum_i \sum_m s_n(\mathbf{x}_i, \mathbf{y}_m) w_i(\mathbf{x}) w_m(\mathbf{y}) \tag{32}$$

The Chebyshev polynomial is

$$T_k(x) = \cos(k \arccos(\frac{2}{b-a}(x - \frac{a+b}{2}))) \tag{33}$$

on $[a, b]$. The formula for Chebyshev nodes, which are the roots of the polynomial, on $[a, b]$ is

$$x = \frac{a+b}{2} + \frac{b-a}{2}\cos(\frac{2i-1}{2k}\pi) \text{ for } i = 1, \ldots, k. \tag{34}$$

This is just in one dimension, points on an interval. For two dimensions, consider two axes r and z, and the Chebyshev nodes on each interval. The nodes in the box these intervals create are all of the possible (r, z) pairs of the nodes in each respective interval.

From this basic function we derive the interpolation functions

$$R_k(\mathbf{x}, \mathbf{y}) = (\frac{1}{k} + \frac{2}{k}\sum_{i=1}^{k-1} T_k(x_1)T_k(y_1))(\frac{1}{k} + \frac{2}{k}\sum_{i=1}^{k-1} T_k(x_2)T_k(y_2)) \tag{35}$$

for $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$.

From there we use a low-rank approximation for the modal kernel given by

$$s_n(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{l}=1}^{k}\sum_{\mathbf{m}=1}^{k} s_n(\overline{\mathbf{x}}_\mathbf{l}, \overline{\mathbf{y}}_\mathbf{m})R_k(\overline{\mathbf{x}}_\mathbf{l}, \mathbf{x})R_k(\overline{\mathbf{y}}_\mathbf{m}, \mathbf{y}) \tag{36}$$

to approximate the sum

$$f_n(\mathbf{x}_i) = \sum_{\mathbf{l}=1}^{k} R_k(\overline{\mathbf{x}}_\mathbf{l}, \mathbf{x}_i)\sum_{\mathbf{m}=1}^{k} s_n(\overline{\mathbf{x}}_\mathbf{l}, \overline{\mathbf{y}}_\mathbf{m})\sum_{j=1}^{P}\sigma_{n,j}R_k(\overline{\mathbf{y}}_\mathbf{m}, \mathbf{y}_j) \tag{37}$$

where P is the number of source points, k^2 is the number of Chebyshev nodes in each box, $\mathbf{y}_j$ are the source points, $\mathbf{x}_i$ are the target points, $\overline{\mathbf{x}}_\mathbf{l}$ are the Chebyshev nodes in the target box, and $\overline{\mathbf{y}}_\mathbf{m}$ are the Chebyshev nodes in the source box.

This low-rank approximation yields the translation operators for this method. For a given mode n, we detail each operation.

8.1 S2M

The source-to-multipole operation computes weights $W^{n,B}_{m_1,m_2}$ at each Chebyshev node $\overline{\mathbf{y}}_{m_1,m_2}$ by anterpolation in a box B on the finest level.

$$W^{n,B}_{m_1,m_2} = \sum_{\mathbf{y}_j \in B}\sigma_{n,j}R_k(\overline{\mathbf{y}}^B_{m_1,m_2}, \overline{\mathbf{y}}_j) \tag{38}$$

for $m_1, m_2 = 1, \ldots, k$.

8.2 M2M

The multipole-to-multipole operation translates weights at the nodes of four child boxes B_i, $W^{n,B_i}_{m_1,m_2}$, to weights at the nodes of its parent box A, $W^{n,A}_{m_1,m_2}$.

$$W^{n,A}_{m_1,m_2} = \sum_{i=1}^{4} \sum_{m_1'=1}^{k} \sum_{m_2'=1}^{k} W^{n,B_i}_{m_1',m_2'} R_k(\overline{\mathbf{y}}^{A}_{m_1,m_2}, \overline{\mathbf{y}}^{B_i}_{m_1',m_2'}) \tag{39}$$

for $m_1, m_2 = 1, \ldots, k$.

8.3 M2L

The multipole-to-local operation computes the far-field contribution at Chebyshev nodes $\overline{\mathbf{x}}^{A}_{l_1,l_2}$ in box A from all boxes B_i in the interaction list of A.

$$g^{n,A}_{l_1,l_2} = \sum_{B_i} \sum_{m_1=1}^{k} \sum_{m_2=1}^{k} W^{n,B_i}_{m_1,m_2} s_n(\overline{\mathbf{x}}^{A}_{l_1,l_2}, \overline{\mathbf{y}}^{B_i}_{m_1,m_2}) \tag{40}$$

for $l_1, l_2 = 1, \ldots, k$.

8.4 L2L

On the root level let $f^{n,A}_{l_1,l_2} = g^{n,A}_{l_1,l_2}$ and use the local-to-local operation to obtain the full local expansion by adding the far-field contribution from the parent box B

$$f^{n,A}_{l_1,l_2} = g^{n,A}_{l_1,l_2} + \sum_{l_1'=1}^{k} \sum_{l_2'=1}^{k} f^{n,B}_{l_1,l_2} R_k(\overline{\mathbf{x}}^{A}_{l_1,l_2}, \overline{\mathbf{x}}^{B}_{l_1',l_2'}) \tag{41}$$

for $l_1, l_2 = 1, \ldots, k$ where B is the parent box of box A.

8.5 L2T

The local-to-target operation computes $f_n(\mathbf{x}_i)$ for target point $\mathbf{x}_i$ in box B by interpolating the far-field approximation.

$$f_n(\mathbf{x}_i) = \sum_{l_1=1}^{k} \sum_{l_2=1}^{k} f^{n,B}_{l_1,l_2} R_k(\overline{\mathbf{x}}^{B}_{l_1,l_2}, \mathbf{x}_i) \tag{42}$$

Notice that the $S2M$, $M2M$, $L2L$, and $L2T$ translation operators in this scenario do not require kernel evaluations. In addition, the interpolation functions R_k do not depend on which modal Green's function we are using. This will amount to savings in the pre-computation because for each mode n, the weights $W^{n,B}_{m_1,m_2}$ can be represented by a matrix

with k^2 rows and N, the truncation parameter discussed earlier chosen for the Fourier representation, columns. Therefore these translation operations are just matrix-matrix multiplications which can be performed quickly in practice.

The full bbFMM also reduces the cost of the $M2L$ operation by using SVD compression. However, with a translation variant kernel some of these operations may not save anything since recyclable transfer vectors on each level was crucial to their argument.

9 Numerical Results

I feel like there should be some numerical result. One thing the bbFMM authors did was the pre-computation time for all necessary operators over multiple computational domains with different number of levels.

In particular, we fix the root level of the computational domain as the box $[0,1]\times[0,1]$ in two-dimensional (r,z)-space, and vary the number of levels in the computational domain, and the number of Chebyshev nodes in each box. Note that the number of Chebyshev nodes is the same for every box, regardless of size or level.

10 Future Work

Ultimately, the translation operators are the core of any kernel-independent fast multipole method. They will work with any density distribution, so once that is specified, we can add a hierarchical tree and complete the algorithm.

Appendices

A Legend

n is the index for each Fourier mode of the Green's function

N is the truncation parameter for the Fourier expansion of the Green's function

P is the number of source points in the computational domain

B Implementation

Explanation of the test code files.

Currently, we are able to produce and have tested all equivalent densities and translation operators with Python programs for the Chebyshev interpolation-based FMM. However, we are just using the logarithmic kernel instead of the modal Green's function for Laplace's

equation. These functions are very similar, so we expect these tests to succeed with the modal Green's function as well.

Much of the future work to do is in fully implementing the algorithm in Python. One key piece of this will be accelerating the $M2L$ operation using singular value decomposition (SVD) compression technique described in [4] and [2].

Now we describe the computations done by each of our test programs.

2D − S2M.py

1. Put sources in box A and assign charges of ± 1.
2. Put target in a box B in the far field.
3. For each Chebyshev node of A, assign a weight

$$W_{m_1,m_2} = \sum_{y_j \in A} \sigma_j R_k(\overline{\mathbf{y}}_{m_1,m_2}, \mathbf{y}_j)$$

2D − M2M

1. Put sources in box A and assign charges of ± 1.
2. For each Chebyshev node of A, assign a weight

$$W_{m_1,m_2} = \sum_{y_j \in A} \sigma_j R_k(\overline{\mathbf{y}}_{m_1,m_2}, \mathbf{y}_j)$$

3. Repeat step 2 for the four child boxes of A.
4. Use the $M2M$ operation to compute $W^A_{m_1,m_2}$ and compare the weight at each node with the computation in step 2.

2D − M2L + L2L.py

This file compares the actual local expansion of a box with our estimate of its local expansion. In this example we consider the computational domain to be only these boxes, with empty space (no sources) outside it.

The computation of the actual local expansion is straightforward. This is the potential at a target point inside the target box due to all sources in the far field $\mathbf{F}$. For our example, this means a target point in the box T due to sources in $P_1, \ldots, P_7$ and $C_1, \ldots, C_{27}$. For the target point $\mathbf{x_0}$ and source points $\mathbf{x_j}$ with weight σ_j, the calculation is:

$$f(\mathbf{x_0}) = \sum_{\mathbf{x_j} \in \mathbf{F}} K(\mathbf{x_0}, \mathbf{x_j}) \sigma_j \tag{43}$$

Now we compute the estimated local expansion. In general in an FMM, the local expansion for a box is the sum of the multipole-to-local operations and the local-to-local operation for this box.

We start by placing sources points in each box $P_1, \ldots, P_7$ and $C_1, \ldots, C_{27}$, and assigning each source a charge of $+1$ or -1. The C boxes are in the interaction list of the target box. The P boxes are in the interaction list of the parent box of the target box. We need to develop an equivalent density for each box $P_1, \ldots, P_7$ and $C_1, \ldots, C_{27}$, which is a set of n^2 weights at the Chebyshev nodes in each box. For each box, take $P1$ for example, this equivalent density W at a node $\overline{y}_{m_1,m_2}$ is the sum over $\mathbf{y}_j$ in P_1 of the R function at each node and each source multiplied by the charge at the source:

$$W(m_1, m_2) = \sum_{y_j \in P_1} R_n(\overline{\mathbf{y}}_{m_1,m_2}, \mathbf{y}_j)\sigma_j \tag{44}$$

Once these weights have been computed for every box in the far field, we compute the multipole-to-local operations for each of the boxes in the interaction list of the parent box. The sum of these over each interaction list box will be the entire local expansion for the parent, since the contribution of the local-to-local operator for this parent is 0 as the grandparent in our case has no interaction list in the computational domain. The $M2L$ operation at each node in the parent box is such for parent box P and interaction boxes $P_1, \ldots, P_7$:

$$g^P_{l_1,l_2} = \sum_{i=1}^{7} \sum_{m_1=1}^{n} \sum_{m_2=1}^{n} W^{P_i}_{m_1,m_2} K(\overline{\mathbf{x}}^P_{l_1,l_2}, \overline{\mathbf{y}}^{P_i}_{m_1,m_2}) \tag{45}$$

Now we compute the same $M2L$ operations at each node in the target box T:

$$g^T_{l_1,l_2} = \sum_{i=1}^{27} \sum_{m_1=1}^{n} \sum_{m_2=1}^{n} W^{C_i}_{m_1,m_2} K(\overline{\mathbf{x}}^C_{l_1,l_2}, \overline{\mathbf{y}}^{C_i}_{m_1,m_2}) \tag{46}$$

Then we compute the estimated local expansion f at each node in the box T:

$$f^T_{l_1,l_2} = g^T_{l_1,l_2} + \sum_{l'_1=1}^{n} \sum_{l'_2=1}^{n} g^P_{l'_1,l'_2} R_n(\overline{\mathbf{x}}^T_{l_1,l_2}, \overline{\mathbf{x}}^P_{l'_1,l'_2}) \tag{47}$$

Finally we compute the estimated potential at the target point due to sources in the far field $\mathbf{F}$:

$$f(\mathbf{x}_0) = \sum_{l_1=1}^{n} \sum_{l_2=1}^{n} f^T_{l_1,l_2} R_n(\overline{\mathbf{x}}^T_{l_1,l_2}, \mathbf{x}_0) \tag{48}$$

All that's left to do is compare this with the above computation of the actual potential we computed naively.

References

[1] Cheng, H., Greengard, L., Rokhlin, V., *A Fast Adaptive Multipole Algorithm in Three Dimensions*. Journal of Computational Physics, 155, (1999), 468-498.

[2] Fong, W., Darve, E., *A black-box fast multipole method*. Journal of Computational Physics.

[3] Hao, S., Martinsson, P.G., Young, P., *An efficient and highly accurate solver for multi-body acoustic scattering problems involving rotationally symmetric scatterers*. Computers and Mathematics with Applications, 69, (2015), 304-318.

[4] Martinsson, P.G., Rokhlin, V., *An accelerated kernel-independent fast multipole method in one dimension*. SIAM Journal of Scientific Computing, Vol. 29, No. 3, (2007), 1160-1178.

[5] Ying, L., Biros, G., Zorin, D., *A kernel-independent adaptive fast multipole method algorithm in two and three dimensions*. Journal of Computational Physics, 196, (2004), 591-626.

[6] Young, P., Yao, S., Martinsson, P.G., *A high-order Nyström discretization scheme for boundary integral equations defined on rotationally symmetric surfaces*. Journal of Computational Physics, 231, (2012), 4142-4159.