MAT1750 Project

A Fast Multipole Method for Boundary Integral Solutions of the Helmholtz equation in 3D

Shashwat Sharma

December 5, 2018

Abstract

A fast multipole algorithm (FMA) is implemented for accelerating the boundary integral solution to the 3D Helmholtz equation, in the context of electromagnetic problems. Interactions between well-separated points are approximated by expressing the governing Green's function as a finite series, and explicit matrix entries corresponding to these interactions are avoided, leading to iterative solutions with sparse matrix-vector products. It is found that despite being computationally efficient, the performance of FMA is adversely affected by poor conditioning of the formulation, and remedies to improve the formulation are discussed.

1 Background

Boundary integral equation methods have garnered significant interest in the field of electromagnetics []. Such methods involve expressing solutions to differential equations in the form of integrals over boundaries of a given problem. This leads to a reduction in the dimensionality of the problem, and precludes the need to discretize space in between boundaries. On the other hand, integral equation methods require the storage and factorization of dense matrices. In realistically large problems with intricate boundaries, the computational cost of dense matrix manipulation is prohibitive. It becomes necessary to develop approximations that avoid explicitly dealing with dense matrices. The fast multipole algorithm (FMA) [1] is one such approach. The goal of this project is to implement and assess FMA in the context of the inhomogeneous 3D Helmholtz equation for electromagnetic problems.

2 Previous Work

The FMA was originally developed by Greengard and Rokhlin [1] for simulating particles that interact with each other through static forces, such as gravity and electrostatic fields. It was extended to two-dimensional dynamic problems for electromagnetics by Engheta *et. al* [2], and has since become immensely popular, spawning several modifications and improvements [3, 4]. In particular, an excellent description of the algorithm is provided by Gibson [5], which is used as the primary reference in this project.

3 Formulation

The three-dimensional inhomogeneous Helmholtz equation can be written in frequency domain as

$$\nabla^2 \vec{A}(\vec{r}) + k^2 \vec{A}(\vec{r}) = \vec{f}(\vec{r}), \tag{1}$$

where $\vec{A}(\vec{r})$ is some field quantity that is to be solved for, and $\vec{f}(\vec{r})$ is some forcing function, which is usually derived from the underlying physics. Assuming that solutions of this equation can be expressed in time harmonic form, the scalar quantity k is the wave number for a particular harmonic.

In general, obtaining the exact solution to (1) requires some boundary conditions. To be consistent with the physics of the problem, it is typically assumed that $\vec{A}(\vec{r})$ decays to 0 as $|(\vec{r})| \to \infty$, which is the pertinent boundary condition. Furthermore, in some problems we can assume that the forcing function $\vec{f}(\vec{r})$ exists on some closed surface $\mathcal{S} \in \mathbb{R}^3$. An example of such a forcing function is an electric current flowing on the surface of a perfect electric conductor (PEC) sphere, such that it produces an electric field only on the outside, and not on the inside.

Given the above, the general solution to (1) can be derived in terms of an auxiliary dyadic "impulse response" function, known as the Green's function, $\overline{\overline{\mathbf{G}}}(\vec{r}, \vec{r}')$, and can be written in integral form as

$$\vec{A}(\vec{r}) = \int_{\mathcal{S}} dS \, \overline{\overline{\mathbf{G}}}(\vec{r}, \vec{r}') \cdot \vec{f}(\vec{r}') \tag{2}$$

where primed coordinates represent source points, and unprimed coordinates represent target points.

In electromagnetic problems involving closed PEC objects, the unknown field quantity of interest is typically the electric surface current density $\vec{J}(\vec{r})$ on the surface of a PEC, and the forcing function is typically some incident electric field $\vec{E}_{\text{inc}}(\vec{r})$. Furthermore, assuming a homogeneous surrounding medium, the Green's function is a scalar. With this in mind, the relevant integral equation to be solved is

$$-j\omega\mu \int_{\mathcal{S}} dS' \left(1 + \frac{\nabla\nabla\cdot}{k^2}\right) G(\vec{r}, \vec{r}') \vec{J}(\vec{r}') = \vec{E}_{\text{inc}}(\vec{r}), \tag{3}$$

where

$$G(\vec{r}, \vec{r}') = \frac{\mathrm{e}^{-jk|\vec{r}-\vec{r}'|}}{|\vec{r}-\vec{r}'|}, \tag{4}$$

ω is the given angular frequency, and μ is a constant scalar material parameter (magnetic permeability).

To discretize (3), the geometry is meshed with planar triangular elements, and linear basis functions $\vec{f}$ are defined at each edge between two adjacent triangles [6]. This allows one to transfer the derivative operators in (3) on to the basis functions rather than the Green's function, thus avoiding hypersingular $(1/r^3)$ behaviour. Discretization leads to an $N_e \times N_e$ linear system of equations

$$\mathbf{L}\mathbf{J} = \mathbf{E}_{\text{inc}}, \tag{5}$$

where N_e is the number of edge basis functions in the geometry. The vector $\mathbf{J}$ contains the unknown basis function coefficients for $\vec{J}(\vec{r})$, and $\mathbf{E}_{\text{inc}}$ likewise represents the incident field. The matrix $\mathbf{L}$ results from the discretization of the integrals in (3); details on the generation of this matrix can be found in [5].

4 Acceleration with the Fast Multipole Algorithm

The guiding principle behind FMA (and most acceleration algorithms for integral equation solvers) is to express the system matrix as the sum of two matrices,

$$\mathbf{L} = \mathbf{L}_{\text{near}} + \mathbf{L}_{\text{far}}, \tag{6}$$

where $\mathbf{L}_{\text{near}}$ is a sparse matrix that only contains source-target interactions for points that are physically close to each other, and thus have strong contributions towards the overall accuracy of the problem. Matrix $\mathbf{L}_{\text{far}}$ instead contains weaker interactions between points farther apart, which can thus be approximated without overly hurting the overall accuracy. In solving (5) iteratively, the matrix-vector product at each iteration would thus become

$$\mathbf{L}\mathbf{J} = \mathbf{L}_{\text{near}}\mathbf{J} + \mathbf{L}_{\text{far}}\mathbf{J}. \tag{7}$$

Computing $\mathbf{L}_{\text{near}}\mathbf{J}$ at each iteration is $\mathcal{O}(N_e)$ since $\mathbf{L}_{\text{near}}$ is sparse. Computing $\mathbf{L}_{\text{far}}\mathbf{J}$ would be expensive if done naively, since $\mathbf{L}_{\text{far}}$ is dense. However, instead of explicitly constructing $\mathbf{L}_{\text{far}}$, FMA allows us to directly compute the matrix-vector product $\mathbf{L}_{\text{far}}\mathbf{J}$ "on the go", during the iterative solve.

The main idea behind FMA relies on the fact that the homogeneous Green's function can be expressed as an integral over the unit sphere [5],

$$\frac{\mathrm{e}^{-jk|\vec{r}-\vec{r}\,'|}}{|\vec{r}-\vec{r}\,'|} = \int_1 T\left(\vec{r}\,', \hat{n}_r\right) T_L\left(k, \hat{n}_r, \vec{r}_{ab}\right) R\left(\vec{r}, \hat{n}_r\right) dS. \tag{8}$$

The meaning of each term in the above is described with physical intuition below, followed by mathematical details.

- The radiation function $T\left(\vec{r}\,', \hat{n}_r\right)$ depends only the position of source points and the sample points on the unit sphere. It thus serves as a way of "aggregating" the contribution of a group of source points into a "packet" of information.

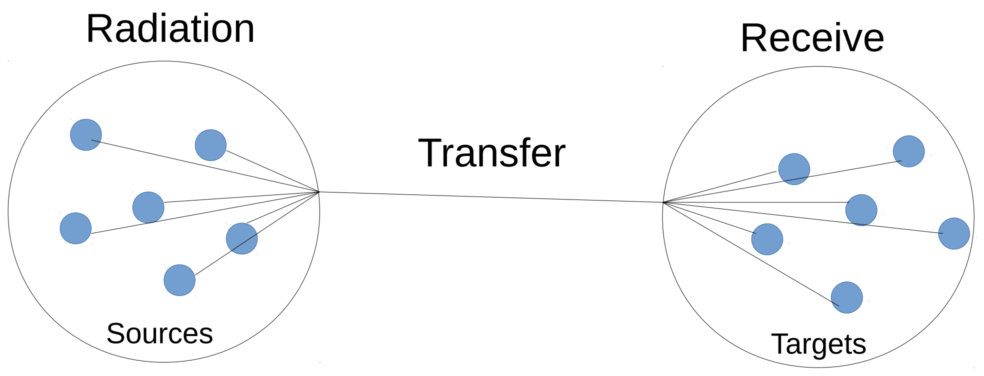


Figure 1 Visual depiction of the three phases in FMA.

- The receive function $R\left(\vec{r}, \hat{n}_r\right)$ conversely only depends on the position of target points, and is thus a way of "disseminating" packets of information among the members of a group of target points.
- The transfer function $T_L\left(k, \hat{n}_r, \vec{r}_{ab}\right)$ acts as an intermediary which transmits information from a source group of points to a target group of points. The vector $\vec{r}_{ab} = \vec{r}_a - \vec{r}_b$ for some fixed $\vec{r}_a$ in the source group, and $\vec{r}_b$ in the target group. $T_L\left(k, \hat{n}_r, \vec{r}_{ab}\right)$ thus encodes interactions between groups of points, rather than individual points.

The points above are visualized in Fig. 1. This decomposition of the Green's function allows one to replace point-to-point interactions with grou-to-group interactions, which immensely improves efficiency if the functions R, T and T_L can be precomputed for each basis function and group.

Mathematical expressions for R, T and T_L are derived in [5] as

$$T\left(\vec{f}_n, \hat{n}_r\right) = \left[1 - \hat{n}_r\hat{n}_r\cdot\right]\int_{\vec{f}_n} \vec{f}_n\left(\vec{r}'\right)\mathrm{e}^{jk\hat{n}_r\cdot\vec{r}'}d\vec{r}' \tag{9}$$

$$R\left(\vec{f}_m, \hat{n}_r\right) = j\omega\mu\int_{\vec{f}_m} \vec{f}_m\left(\vec{r}\right)\mathrm{e}^{-jk\hat{n}_r\cdot\vec{r}}d\vec{r} \tag{10}$$

$$T_L\left(k, \hat{n}_r, \vec{r}_{ab}\right) = \frac{k}{(4\pi)^2}\sum_{l=0}^{\infty}\left(-j\right)^{l+1}\left(2l+1\right)h_l^{(2)}\left(k\left|\vec{r}_{ab}\right|\right)P_l\left(\hat{n}_r\cdot\vec{r}_{ab}\right) \tag{11}$$

where $\vec{f}\left(\vec{r}\right)$ is the edge basis function on the mesh, $\hat{n}_r$ is a unit vector directly radially outwards from the unit sphere over which the group-by-group integration is performed, $h_l^{(2)}$ is the spherical Hankel function of second kind and order l, and P_l is the Legendre polynomial of degree l.

The most important feature of note is that the transfer function is expressed as an infinite sum. Practically speaker, this sum must be truncated at a finite number of terms L (which depends on the size of the problem and the parameter k). This is the core

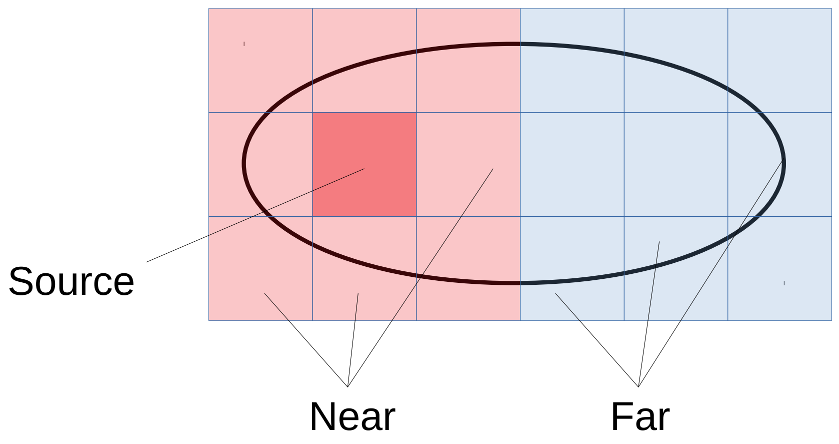


Figure 2 Grouping of basis functions into cubes.

approximation used in FMA, and the choice of number of terms allows the user to pick a desired level of accuracy.

The grouping of basis functions is done by enclosing the domain of interest in a series of cubes, as depicted in Fig. 2. The vectors $\vec{r}_a$ and $\vec{r}_b$ are set to be the centres of the respective source and target cubes. For each source cube, its immediate neighbours are considered to be "near" cubes whose basis function interactions are computed directly, contributing to entries in $\mathbf{L}_{\text{near}}$. The remaining cubes are denoted as "far" cubes, whose interactions are computed using FMA. The size of cubes is generally dependent on the frequency of interest; in this implementation, a variety of sizes were explored. The computational complexity of the resulting algorithm is $\mathcal{O}(N_e^{3/2})$.

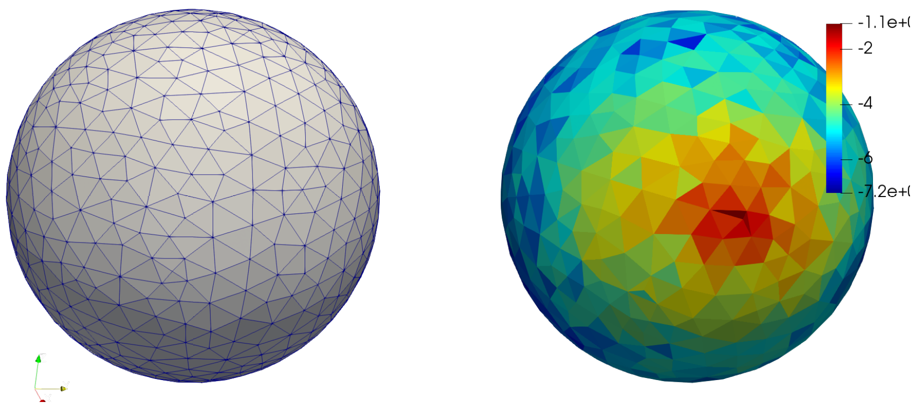
5 Implementation

The current implementation is in C++, and the open-source library PETSc [7] was used to iteratively solve sparse linear systems via the GMRES algorithm, and a diagonal left-preconditioner. Hankel functions were computed using the open-source library by Amos [8], while Legendre polynomials were computed using the Boost C++ library [9].

6 Results and Discussion

Fig. 3a shows the geometry used as a test case for the algorithm. Fig. 3b shows the current distribution $\vec{J}(\vec{r})$ obtained by naively constructing the entire system matrix, and using LU factorization to obtain the solutions. This will be used as the reference solution herein.

It was observed that although FMA greatly decreases the overall computation time, the poor convergence properties of the system matrix (with condition number estimated to be $\mathcal{O}(10^{11})$) prevents the iterative solution from converging. Thus, in order to validate



(a) **(b)**

Figure 3 (a) Test geometry and mesh. (b) Reference solution obtained from a direct solver.

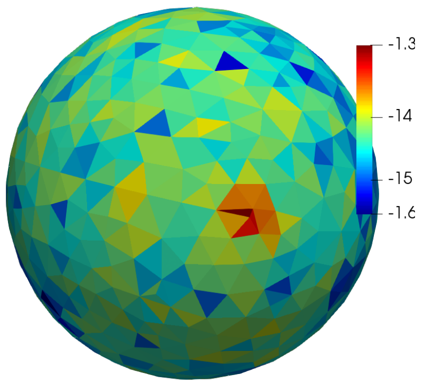


Figure 4 Solution obtained via FMA.

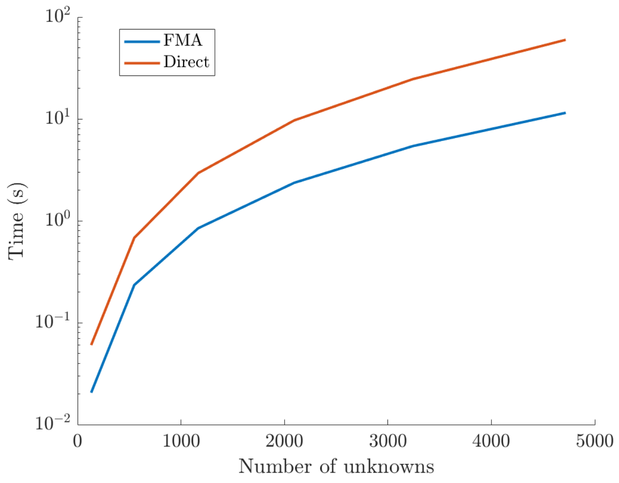


Figure 5 Near matrix and FMA function generation time, vs. direct matrix generation time.

the solution, the only appropriate way was to compare the FMA solution with the solution obtained by applying the iterative solver to the explicitly-generated full system matrix. The solutions thus obtained, although not nearly converged, did match. The resulting FMA current distribution is shown in Fig. 4, which clearly shows that the solution suffers from lack of convergence. At best, the relative residual norm was only reduced to 30 %.

Assessing the performance of the iterative solution is difficult without convergence. However, it is possible to confirm the acceleration obtained in the generation of the matrix using FMA. The mesh of the sphere was refined progressively to see the time taken for generation of the near matrix and the precomputation of the FMA functions. This total time was compared with the matrix generation time of the entire system directly, and is shown in Fig. 5. The improvement in efficiency is quite apparent, and the level of improvement increases with problem size, as expected. The size of FMA cubes was set to be 1/5 of the size of the bounding box of the sphere for all runs. This is an extremely conservative setting, and was used in order to achieve the 30 % relative residual norm. Given better convergence properties, one would likely use much smaller cubes. In this case, however, this caused an even worse residual norm.

Overall, the implementation was not very successful, and it was difficult to analyze the results in much detail. However, this underscores the importance of having a well conditioned formulation. It has been proposed in literature that separating out each term in $\left(1 + \frac{\nabla\nabla\cdot}{k^2}\right)$ and writing it as two separate matrices allows resolving the static problem (the second part) from the dynamic one (the first part). This would lead to an additional set of unknowns, for which additional equations can be generated from the physics of the problem. The increase in size of unknowns, evidently, is well worth the immense improvement in conditioning. This would be a valuable future step.

References

[1] L. Greengard and V. Rokhlin, “A Fast Algorithm for Particle Simulations,” *Journal of Computational Physics*, vol. 73, no. 2, pp. 325 – 348, 1987.

[2] N. Engheta, W. D. Murphy, V. Rokhlin, and M. S. Vassiliou, “The fast multipole method (fmm) for electromagnetic scattering problems,” *IEEE Transactions on Antennas and Propagation*, vol. 40, pp. 634–641, June 1992.

[3] W. Chew, E. Michielssen, J. M. Song, and J. M. Jin, eds., *Fast and Efficient Algorithms in Computational Electromagnetics.* Norwood, MA, USA: Artech House, Inc., 2001.

[4] J. Song, C.-C. Lu, and W. C. Chew, “Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects,” *IEEE Transactions on Antennas and Propagation*, vol. 45, pp. 1488–1493, Oct 1997.

[5] W. C. Gibson, *The Method of Moments in Electromagnetics.* CRC press, 2014.

[6] S. Rao, D. Wilton, and A. Glisson, “Electromagnetic Scattering by Surfaces of Arbitrary Shape,” vol. 30, pp. 409–418, May 1982.

[7] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, “PETSc Web Page,” 2018.

[8] D. Amos, “Subroutine package for bessel functions of a complex argument and nonnegative order,” 1985.

[9] “Boost Web Page,” 2018.