Fast Large-Scale Algorithm for Electromagnetic Wave Propagation in 3D Media

Mitchell Tong Harris*, M. Harper Langston*, Pierre-David Létourneau*, George Papanicolaou†, James Ezick*, Richard Lethin*

*Reservoir Labs, New York, USA. Email: lastname@reservoir.com

†Dept. of Mathematics, Stanford University, Stanford, USA. Email: lastname@stanford.edu

***Abstract*—We present a fast, large-scale algorithm for the simulation of electromagnetic waves (Maxwell's equations) in three-dimensional inhomogeneous media. The algorithm has a complexity of $O(N\log(N))$ and runs in parallel. Numerical simulations show the rapid treatment of problems with tens of millions of unknowns on a small shared-memory cluster (≤ 16 cores).**

***Index Terms*—Fast Algorithms, Fast Fourier Transform, Maxwell's equations, Electromagnetic Waves, Inhomogeneous Media, multiprocess physical system computing, high performance parallel applications, imaging data analysis**

I. Introduction

Maxwell's equations (4) form the backbone of classical electromagnetic theory. Their solutions provide the value of the electrical field and magnetic field as functions of space and are of prime importance in multiple applications ranging from medical imaging [1], radar scattering [2], and geophysics [3] to modern applications involving metamaterials [4].

A. Related Material

In an integral equation framework [5], the numerical solution of the system of eqn. (4) can be formulated through a self-consistent linear equation of the form

$$u(x) - \int \overleftrightarrow{G}(x,y)\, u(y)\, \mathrm{d}y = f(x), \tag{1}$$

where $\overleftrightarrow{G}(x,y)$ is called the kernel or Green's function (see Section II). This kernel has two characteristics that present challenges when solving the equation numerically: it is long-range, and it possesses a strong singularity at the origin ($O\left(\frac{1}{r^3}\right)$). These characteristics lead to *large* and *dense* linear systems, especially when high sccuracy is desired. To solve the latter numerically, fast and scalable algorithms are necessary.

In this regard, there exists two broad families of methods to tackle the problem: Fast Multipole Method (FMM)-based techniques [6]–[9] and Fast Fourier Transform (FFT)-based techniques [10].

This material is based upon work supported by the Defense Advanced Research Projects Agency ITA3 (Imaging Through Almost Anything Anywhere) Disruption Opportunity under Contract No. HR001118C0048. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement A (Approved for Public Release, Distribution Unlimited).

FMM-based algorithms rely on a complex adaptive/hierarchichal discretization of the computational domain that depends on the local regularity of the function $f(x)$ and the kernel. These algorithms are asymptotically fast but often slower than FFT-based algorithms due to a larger computational constant. Further, they can exhibit numerical instabilities in certain frequency regimes [11] and do not completely address issues associated with the singularity at the origin.

FFT-based techniques are well-adapted to problems in which a uniform mesh is appropriate given the regularity and smoothness of the function $f(x)$. In this case, there exists two common approaches to the discretization problem. The first approach is referred to as *locally-corrected* quadratures where the discretized version of the integral in eqn. (1) takes the form

$$\frac{1}{N^3}\sum_{n'} G(n-n')f(n') + \sum_{n'} w_{n,n'} f(n'), \tag{2}$$

where $\{w_{n,n'}\}$ are appropriately-chosen weights. In particular, it can be shown [12]–[18] that by choosing $\{w_{n,n'}\}$ and other parameters appropriately, one can obtain a high-order, numerically stable quadrature. Then, since the discretization is carried over a uniform grid, the resulting matrix-vector product can then be computed rapidly using the FFT.

We refer to the second approach as a *truncated kernel* technique [10]. This technique possesses two significant advantages over the aforementioned technique: First, it does not require the computation of weights $\{w_{n,n'}\}$, which can be quite involved. Secondly, and more importantly, when the function $f(x)$ is both smooth and compactly supported, the truncated kernel method offers *spectral* accuracy. Our work uses a technique closely related to the truncated kernel method. It is described in detail in Section II.

B. Our contributions

In summary, the benefits of the FFT-based methods include high accuracy and low computational costs. Our contributions in these regards include:

1) A generalization of the truncated kernel method to Maxwell's equations.
2) A parallel, large-scale shared-memory implementation of the resulting Maxwell's equations solver.

II. Notation & Framework

In this section, we introduce the problem and the notation we shall use throughout this paper. First, we designate the three-dimensional *electric field* and *magnetic field* by $\mathbf{E}(\mathbf{x})$ and $\mathbf{H}(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^3$. We also denote the spatially-dependent *permittivity, conductivity* and *permeability tensors* by $\epsilon(\mathbf{x})$, $\sigma(\mathbf{x})$ and $\mu(\mathbf{x})$, respectively, and let the constants ϵ_0 and μ_0 be the permittivity and permeability of a vacuum.[1] These properties characterize the medium through which electromagnetic waves propagate (Maxwell's equations (4)). We further restrict our attention to *diagonal* tensors, i.e.,

$$\epsilon(\mathbf{x}) = \begin{bmatrix} \epsilon^x(\mathbf{x}) & 0 & 0 \\ 0 & \epsilon^y(\mathbf{x}) & 0 \\ 0 & 0 & \epsilon^z(\mathbf{x}) \end{bmatrix}, \tag{3}$$

The special case where all three diagonal components are equal, i.e., $\epsilon^x(\mathbf{x}) = \epsilon^y(\mathbf{x}) = \epsilon^z(\mathbf{x})$ for all $x \in \mathbb{R}^3$, is also referred to as *isotropic*. Such restrictions do simplify the analysis and the presentation of the results but do not reduce the scope of the technique, which can deal with general 2-tensor fields.

In their simplest form,[2] Maxwell's equations take the form

$$\begin{aligned} \nabla \times \mathbf{H}(\mathbf{x}) - (i\omega\epsilon(\mathbf{x}) + \sigma(\mathbf{x}))\mathbf{E}(\mathbf{x}) &= \mathbf{J}(\mathbf{x}) \\ \nabla \times \mathbf{E}(\mathbf{x}) + i\omega\mu(\mathbf{x})\mathbf{H}(x) &= \mathbf{M}(\mathbf{x}), \end{aligned} \tag{4}$$

where $\mathbf{E}(\mathbf{x})$ is the electrical field and $\mathbf{H}(\mathbf{x})$ is the magnetic field, and it is assumed that there is no free charge density, i.e., $\nabla \cdot \mathbf{J}(\mathbf{x}) = 0$ and $\nabla \cdot \mathbf{M}(\mathbf{x}) = 0$. Here, $\mathbf{J}(\mathbf{x})$ and $\mathbf{M}(\mathbf{x})$ are the *electric and magnetic source currents*, and ω is the *angular frequency*. These constitute the physical model for which we present a fast computational solution technique. Under the appropriate circumstances,[3] the PDE system in eqn.(4) has a unique solution characterized by the self-consistent linear system described in Theorem 1.

Theorem 1. Lippmann-Schwinger framework *Define the matrix-vector product* $A \begin{bmatrix} \mathbf{E}(\mathbf{x}) \\ \mathbf{H}(\mathbf{x}) \end{bmatrix}$ *as* $\begin{bmatrix} A_{11}E(x) + A_{12}H(x) \\ A_{21}E(x) + A_{22}H(x) \end{bmatrix}$ *where*

$$A_{11}(E(x)) = -i\omega\mu_0 \int \overleftrightarrow{G}(x,y)\, b(y)\, E(y) \mathrm{d}y, \tag{5}$$

$$A_{12}(H(x)) = \int \nabla \times \overleftrightarrow{G}(x,y)\, (-a(y))\, H(y) \mathrm{d}y \tag{6}$$

$$A_{21}(E(x)) = \int \nabla \times \overleftrightarrow{G}(x,y)\, b(y)\, E(y) \mathrm{d}y, \tag{7}$$

$$A_{22}(H(x)) = i\omega\epsilon_0 \int \overleftrightarrow{G}(x,y)\, (-a(y))\, H(y) \mathrm{d}y, \tag{8}$$

[1] $\epsilon_0 = 8.85418781 \cdot 10^{-12}$ F/m; $\mu_0 = 1.25663706144 \cdot 10^{-6}$ H/m

[2] Time-harmonic/single-frequency (ω), linear relationship between electrical displacement/magnetizing field and electrical/magnetic field, i.e., $E - \epsilon D$, $H = \frac{1}{\mu_0} B$.

[3] Sufficient regularity of the right-hand sides: $J(x)$, $M(x)$, and the coefficients: $\epsilon(x)$, $\sigma(x)$, $\mu(x)$. Sommerfeld radiation conditions at infinity in $\mathbb{R}^3$.

with $\tilde{a}(x) = i\omega\,(\mu(x) - \mu_0)$ *and* $\tilde{b}(x) = i\omega\,(\epsilon(x) - \epsilon_0) + \sigma(x)$. *Here,* $\overleftrightarrow{G}(x,y)$ *is the dyadic Green's function,*

$$\overleftrightarrow{G}(x,y) = \overleftrightarrow{I}\, G(x-y) + \frac{1}{k_0^2} \nabla \nabla G(x-y) \tag{9}$$

where $k_0 = \omega\sqrt{\epsilon_0\mu_0}$, $G(x) = \frac{e^{ik_0|x|}}{4\pi|x|}$, $\overleftrightarrow{I}$ *is the* 3×3 *identity matrix, and*

$$\nabla \nabla = \begin{bmatrix} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial x \partial y} & \frac{\partial^2}{\partial x \partial z} \\ \frac{\partial^2}{\partial x \partial x} & \frac{\partial^2}{\partial y^2} & \frac{\partial^2}{\partial y \partial z} \\ \frac{\partial^2}{\partial x \partial z} & \frac{\partial^2}{\partial y \partial z} & \frac{\partial^2}{\partial z^2} \end{bmatrix}. \tag{10}$$

If

$$\begin{aligned} E_0(x) &= -i\omega\mu_0 \int \overleftrightarrow{G}(x,y)\, J(y)\, \mathrm{d}y \\ &\quad + \int \nabla \times \overleftrightarrow{G}(x,y)\, M(y)\, \mathrm{d}y \\ H_0(x) &= \int \nabla \times \overleftrightarrow{G}(x,y)\, J(y)\, \mathrm{d}y \\ &\quad + i\omega\epsilon_0 \int \overleftrightarrow{G}(x,y)\, M(y)\, \mathrm{d}y, \end{aligned} \tag{11}$$

then the solution to eqn. (4) *is characterized by*

$$\begin{bmatrix} E(x) \\ H(x) \end{bmatrix} = \begin{bmatrix} E_0(x) \\ H_0(x) \end{bmatrix} + A \begin{bmatrix} E(x) \\ H(x) \end{bmatrix}. \tag{12}$$

See [19] for a proof.

Remark: Eqn. (11) corresponds to the incoming field in a homogeneous background medium.

Superscripts of the form $\cdot^x$, $\cdot^y$, $\cdot^z$ are used to denote the *Cartesian components* of the quantity under consideration. In particular, one should not confound the superscript and the argument in our expressions; for instance, $E^x(\mathbf{x})$ refers to the x-component of the electric field evaluated at $\mathbf{x} \in \mathbb{R}^3$. Also, given any vector $\nu \in \mathbb{R}^N$, we use $[\nu]_n$ for $n \in \{1, 2, ..., N\}$ to designate its n^{th} component.

III. Algorithm

In this section, we provide a description of our algorithm which corresponds to a generalization of the truncated kernel algorithm introduced by Vico et al. [10] to electromagnetic waves (see also Section I-A). To begin, consider the (scalar) free-space convolution,

$$u(x) = \int G(x-y)\, f(y)\, \mathrm{d}y, \tag{13}$$

where it is assumed that the function f is both *smooth* ($f \in C^\infty(\mathbb{R}^3)$) and *compactly supported* in the unit cube $\mathcal{C} = [-0.5, 0.5]^3$.

Let $\mathbb{I}_{B_0(1)}(x)$ be the indicator function of the ball $B_0(1)$ of radius 1 centered at the origin, i.e., $\mathbb{I}_{B_0(1)}(x) = 1$ if $x \in B_0(1)$ and 0 otherwise. Then for all $x \in \mathcal{C}$, due to the properties of f one can write,

$$\begin{aligned} u(x) &= \int_{\mathbb{R}^3} G(x-y)\, \mathbb{I}_{B_0(1)}(x-y)\, f(y)\, \mathrm{d}y \qquad (14) \\ &= \int_{\mathbb{R}^3} e^{ix\cdot\xi}\, \mathcal{F}\left[G(y)\, \mathbb{I}_{B_0(1)}(y)\right](\xi)\, \mathcal{F}\left[f(y)\right](\xi)\, \mathrm{d}\xi, \qquad (15) \end{aligned}$$

where $\mathcal{F}[\cdot]$ and $\mathcal{F}^{-1}[\cdot]$ represent the Fourier transform and its inverse, respectively. Also, note the second equality follows from the properties of the Fourier transform; convolutions correspond to point-wise multiplication in Fourier space.

Importantly, we note that the function $\mathcal{F}\left[G(y)\,\mathbb{I}_{B_0(1)}(y)\right](\xi)$ is entire,[4] and that $\mathcal{F}[f(y)](\xi)$ is smooth and decays exponentially fast as a function of $|\xi|$.[5] In particular, given $(n_x, n_y, n_z) \in \mathbb{N}^3$ one can analytically compute,

$$\mathcal{F}\left[G(y)\,\mathbb{I}_{B_0(1)}(y)\right](\xi) = \frac{-1 + e^{ik}\left(\cos(|\xi|) - i\frac{k}{|\xi|}\,\sin(|\xi|)\right)}{(k-|\xi|)\,(k+|\xi|)} \tag{16}$$

$$\mathcal{F}\left[\frac{\partial^{n_x+n_y+n_z}}{\partial x^{n_x}\,\partial y^{n_y}\,\partial z^{n_z}} G(y)\,\mathbb{I}_{B_0(1)}(y)\right](\xi) = \left(\xi_x^{n_x}\,\xi_y^{n_y}\,\xi_z^{n_z}\right)\;\hat{G}(\xi). \tag{17}$$

Combining everything, the integral in eqn. (15) can be discretized using the trapezoid rule as

$$\sum_{m\in\{-2N,\ldots,2N-1\}^3} \frac{e^{i\frac{n\cdot m}{4N}}}{(4N)^3}\,\hat{\mathcal{G}}(\xi_m)\left[\sum_{k\in\{-2N,\ldots,2N-1\}^3} e^{-i\frac{n\cdot m}{4N}}\,f\left(\frac{k}{4N}\right)\right] \tag{18}$$

which we recognize as a *Discrete Fourier Transform (DFT)*. A similar discretization can be obtained for expressions of the form,

$$\int \overleftrightarrow{G}(x,y)\,f(y)\,\mathrm{d}y,\ \int \nabla\times\overleftrightarrow{G}(x,y)\,f(y)\,\mathrm{d}y \tag{19}$$

using eqn. (17). In addition, it is possible to reduce the size of the DFTs from $(4N)^3$ to $(2N)^3$ through a pre-computation step (Algorithm 1). Finally, by leveraging the speed of the FFT, this scheme leads to a fast ($O(N\log(N))$ algorithm for the application of the matrix A in Theorem 1. Pseudo-code for that is provided in Algorithms 2-3. We refer the reader to [10] for a more detailed discussion on the discretization of the integral and the choice of N.

We describe the algorithms below. For this purpose, we introduce a few definitions. First, we let

$$I_M := \{-M, -M+1, ..., M-1\}^3 \tag{20}$$

Then, we introduce the "index" sets,

$$\mathcal{D}^{(1)} = \{n_x, n_y, n_z\} \tag{21}$$

$$\mathcal{D}^{(2)} = \{(n_x,n_x),(n_y,n_y),(n_z,n_z), \tag{22}$$

$$(n_x,n_y),(n_x,n_z),(n_y,n_z)\} \tag{23}$$

These corresponds to the *derivative indices*; the elements of $\mathcal{D}^{(1)}$ will be used to denote first-order derivatives $\partial^{(\alpha)}$ in x, y and z, respectively, whereas $\mathcal{D}^{(2)}$ will be used to denote second-order derivatives $\partial^{(\beta_1,\beta_2)}$. The truncation of an array X to the indices in an index set I is written $X\big|_I$. Finally F and F^{-1} represent the Discrete Fourier Transform (DFT) and its inverse, respectively.

[4]Following the Paley-Wiener theorem [20]

[5]Because of the smoothness of f; see, e.g., [21]

Algorithm 1 Kernel pre-computation (PRECOMP)

Input: N
Let $\mathcal{D}^{(1)}$, $\mathcal{D}^{(2)}$, I_{2N} and I_{4N} be as in eqn. (20)-(22).
for $n \in I_{4N}$, **do**
  $\hat{G}_n \leftarrow \hat{G}(n)$ (eqn. (16))
  $G_{4N} \leftarrow F^{-1}\,\hat{G}$
end for
for $\alpha \in \mathcal{D}^{(1)}$ **do**
  for $n \in I_{4N}$ **do**
    $\left[\hat{G}_{4N}^{(\alpha)}\right]_n \leftarrow \alpha\cdot\hat{G}_n$ (eqn. (17))
  end for
  $G_{4N}^{(\alpha)} \leftarrow F^{-1}\,\hat{G}_{4N}^{(\alpha)}$
  $G_{2N}^{(\alpha)} \leftarrow G_{4N}^{(\alpha)}\big|_{I_{2N}}$
  $\hat{G}_{2N}^{(\alpha)} \leftarrow F\,G_{2N}^{(\alpha)}$
end for
for $(\beta_1,\beta_2) \in \mathcal{D}^{(2)}$ **do**
  for $n \in I_{4N}$ **do**
    $\left[\hat{G}_{4N}^{(\beta_1,\beta_2)}\right]_n \leftarrow \beta_1\beta_2\cdot\hat{G}_n$ (eqn. (17))
  end for
  $G_{4N}^{(\beta_1,\beta_2)} \leftarrow F^{-1}\,\hat{G}^{(\beta_1,\beta_2)}$
  $G_{2N}^{(\beta_1,\beta_2)} \leftarrow G_{4N}^{(\beta_1,\beta_2)}\big|_{I_{2N}}$
  $\hat{G}_{2N}^{(\beta_1,\beta_2)} \leftarrow F\,G_{2N}^{(\beta_1,\beta_2)}$
end for
Output: $\hat{G}_{2N}$, $\left\{\hat{G}_{2N}^{(\alpha)}\right\}_{\alpha\in\mathcal{D}^{(1)}}$, $\left\{\hat{G}_{2N}^{(\beta_1,\beta_2)}\right\}_{(\beta_1,\beta_2)\in\mathcal{D}^{(2)}}$

Algorithm 2 FFT-based kernel application (KER)

Input: f, $\hat{G}_{2N}$, $\left\{\hat{G}_{2N}^{(\alpha)}\right\}_{\alpha\in\mathcal{D}^{(1)}}$, $\left\{\hat{G}_{2N}^{(\beta_1,\beta_2)}\right\}_{(\beta_1,\beta_2)\in\mathcal{D}^{(2)}}$
Let $\mathcal{D}^{(1)}$, $\mathcal{D}^{(2)}$, I_{2N} and I_{4N} be as in eqns. (20)-(22).
Pad with zeros: $[f_{2N}]_n = \begin{cases} f_n & \text{if } n \in I_N \\ 0 & \text{otherwise} \end{cases}$
$\hat{f}_{2N} \leftarrow F\,f_{2N}$
for $\alpha \in \mathcal{D}^{(1)}$, $(\beta_1,\beta_2) \in \mathcal{D}^{(2)}$ **do**
  $\hat{f}_{2N} \leftarrow \hat{G}_{2N}\cdot\hat{f}_{2N}$
  $\partial^{(\alpha)}\hat{f}_{2N} \leftarrow \hat{G}_{2N}^{(\alpha)}\cdot\hat{f}_{2N}$
  $\partial^{(\beta_1,\beta_2)}\hat{f}_{2N} \leftarrow \hat{G}_{2N}^{(\beta_1,\beta_2)}\cdot\hat{f}_{2N}$
  $f_N \leftarrow F^{-1}\;\hat{f}_{2N}\big|_{I_N}$
  $\partial^{(\alpha)} f_N \leftarrow F^{-1}\,\partial^{(\alpha)}\;\hat{f}_{2N}\big|_{I_N}$
  $\partial^{(\beta_1,\beta_2)} f_N \leftarrow F^{-1}\,\partial^{(\beta_1,\beta_2)}\;\hat{f}_{2N}\big|_{I_N}$
  Output: f_N, $\{\partial^{(\alpha)} f_N\}$, $\{\partial^{(\beta_1,\beta_2)} f_N\}$
end for

Algorithm 3 Fast Matrix-Vector Product ($*$ indicates pointwise multiplication)

Input: $\mathcal{E}, \mathcal{H}$, ϵ_0, μ_0, N
Let: $a_N = \{a\left(\frac{n}{2N}\right)\}_{n\in I_N}$, $b_N = \{b\left(\frac{n}{2N}\right)\}_{n\in I_N}$
for $\alpha \in \mathcal{D}^{(1)}$, $(\beta_1, \beta_2) \in \mathcal{D}^{(2)}$ **do**
$\hat{G}_{2N}, \left\{\hat{G}_{2N}^{(\alpha)}\right\}, \left\{\hat{G}_{2N}^{(\beta_1,\beta_2)}\right\} \leftarrow \texttt{PRECOMP}(N)$ // this is independent of inputs and should be pre-computed
$\mathcal{E}_N, \left\{\partial^{(\alpha)}\mathcal{E}_N\right\}, \left\{\partial^{(\beta_1,\beta_2)}\mathcal{E}_N\right\} \leftarrow \texttt{KER}\left(b_N * \mathcal{E}, \hat{G}_{2N}, \left\{\hat{G}_{2N}^{(\alpha)}\right\}, \left\{\hat{G}_{2N}^{(\beta_1,\beta_2)}\right\}\right)$
$\mathcal{H}_N, \left\{\partial^{(\alpha)}\mathcal{H}_N\right\}, \left\{\partial^{(\beta_1,\beta_2)}\mathcal{H}_N\right\} \leftarrow \texttt{KER}\left((-a_N) * \mathcal{H}, \hat{G}_{2N}, \left\{\hat{G}_{2N}^{(\alpha)}\right\}, \left\{\hat{G}_{2N}^{(\beta_1,\beta_2)}\right\}\right)$
end for
$E^x = -i\omega\mu_0\mathcal{E}_N^x - \frac{i\omega\mu_0}{k^2}\left(\partial^{(x,x)}\mathcal{E}_N^x + \partial^{(x,y)}\mathcal{E}_N^y + \partial^{(x,z)}\mathcal{E}_N^z\right) + \left(\partial^{(y)}\mathcal{H}_N^z - \partial^{(z)}\mathcal{H}_N^y\right)$
$E^y = -i\omega\mu_0\mathcal{E}_N^y - \frac{i\omega\mu_0}{k^2}\left(\partial^{(x,y)}\mathcal{E}_N^x + \partial^{(y,y)}\mathcal{E}_N^y + \partial^{(y,z)}\mathcal{E}_N^z\right) + \left(\partial^{(z)}\mathcal{H}_N^x - \partial^{(x)}\mathcal{H}_N^z\right)$
$E^z = -i\omega\mu_0\mathcal{E}_N^z - \frac{i\omega\mu_0}{k^2}\left(\partial^{(x,z)}\mathcal{E}_N^x + \partial^{(y,z)}\mathcal{E}_N^y + \partial^{(z,z)}\mathcal{E}_N^z\right) + \left(\partial^{(x)}\mathcal{H}_N^y - \partial^{(y)}\mathcal{H}_N^x\right)$
$H^x = i\omega\epsilon_0\mathcal{H}_N^x + \frac{i\omega\epsilon_0}{k^2}\left(\partial^{(x,x)}\mathcal{H}_N^x + \partial^{(x,y)}\mathcal{H}_N^y + \partial^{(x,z)}\mathcal{H}_N^z\right) + \left(\partial^{(y)}\mathcal{E}_N^z - \partial^{(z)}\mathcal{E}_N^y\right)$
$H^y = i\omega\epsilon_0\mathcal{H}_N^y + \frac{i\omega\epsilon_0}{k^2}\left(\partial^{(x,y)}\mathcal{H}_N^x + \partial^{(y,y)}\mathcal{H}_N^y + \partial^{(y,x)}\mathcal{H}_N^z\right) + \left(\partial^{(z)}\mathcal{E}_N^x - \partial^{(x)}\mathcal{E}_N^z\right)$
$H^z = i\omega\epsilon_0\mathcal{H}_N^z + \frac{i\omega\epsilon_0}{k^2}\left(\partial^{(x,z)}\mathcal{H}_N^x + \partial^{(y,z)}\mathcal{H}_N^y + \partial^{(z,z)}\mathcal{H}_N^z\right) + \left(\partial^{(x)}\mathcal{E}_N^y - \partial^{(y)}\mathcal{E}_N^x\right)$
Output: E, H

We have implemented Algorithms 1-3 in the C programming language. Performance results are provided in Section IV.

A. Parallelization

Our shared-memory parallelized implementation of Algorithm 2 relies on OpenMP [22] as well as a parallelized version of FFTW [23]. By re-arranging operations in Algorithm 3, we bring the total number of FFT per matrix-vector product down to 7, each of which is performed in parallel. Each of the loops is effectively a point-wise vector multiplication, which has been optimized for data locality and using temporary registers where necessary to ensure all OpenMP threads run with optimal concurrency. Additionally, we use *a priori* wisdom about the nature of the input data to construct the best FFTW plans; these plans are allocated once and re-used through pointer swapping in memory to increase efficiency and decrease any time wasted on memory allocation routines.

IV. Numerical Results

In this section, we present the results of five numerical experiments. The first is a showcase example with $6 \cdot 128^3 = 12,582,912$ unknowns. We then present results related to asymptotic, strong, and weak parallel scaling for Algorithm 3 – the core part of the solver. Finally, we show how refining the grid affects time and accuracy.

The showcase example has a source at $(0, 0, 2)$m with a randomly oriented magnetic dipole moment and a frequency of 1 GHz. The medium is supported within $[-0.5, 0.5]^3$m and consists in anisotropic permittivity and permeability random and smooth distributions located within a Gaussian lense (Figure 1). The discretization is uniform and uses $N = 128$ points per dimension. The maximum permittivity is $10\epsilon_0$, and the maximum permeability is $.99\mu_0$. Computed cross-sections of the total electric fields ($\mathcal{E}$) are plotted in Figure 2. The effects of the randomness and the Gaussian shape of the medium are clearly visible.

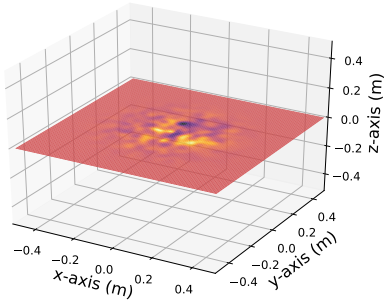


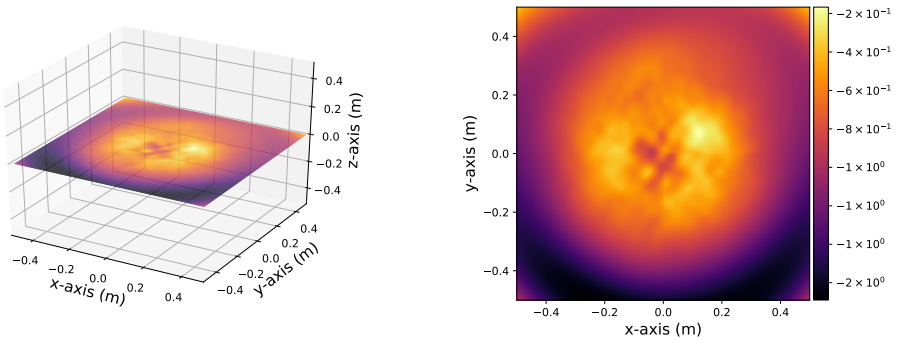
Fig. 1: x-component of permittivity (ϵ) of medium.

Next, we studied the asymptotic, weak parallel and strong parallel scaling of the algorithm (Figure 3-5). For each simulation, we used N points per dimension, for a total of $6 \cdot N^3$ unknowns. Each test case was run 10 times, using random inputs $\mathcal{E}$ and $\mathcal{H}$, and the mean of those times is plotted along with error bars representing the standard deviation of the trials.

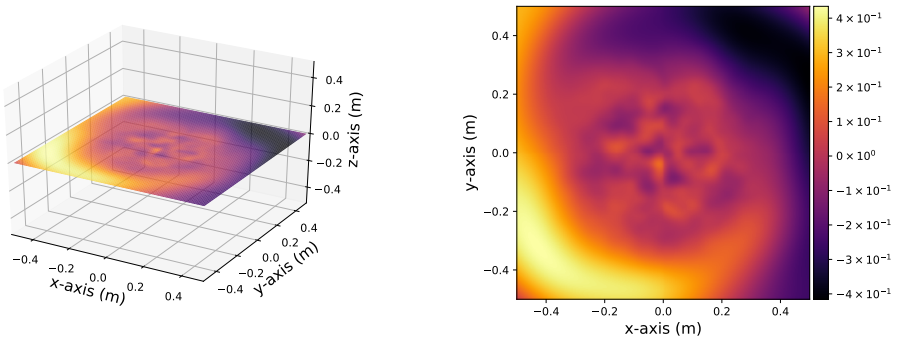
To study the asymptotic complexity, we used a single processor on various problem sizes. The results are shown in Figure 3, from which an asymptotic quasi-linear ($O(N\log(N))$) scaling can be observed.

Results pertaining to parallel scalings are shown in Figure 4 and Figure 5. Details about the number of unknowns per processor are presented in Table I and comparison with theoretical expectations are shown in the figures. The column "ratio of times" refers to the ratio of the mean time taken for the number of processors in that row to the mean time taken for the number of processors in the previous row. The mean times are plotted in Figure 5. Advantageous weak scaling holds until about 16 processors, and the strong scaling exhibits a behavior similar to the perfect theoretical scaling within the whole studied range.

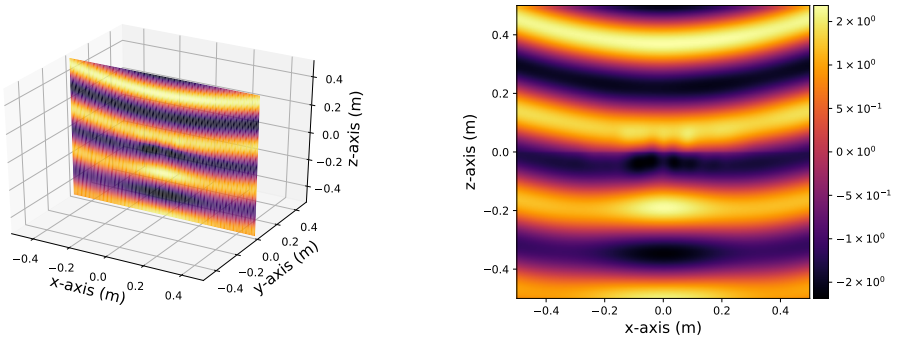
Fig. 2: Real part of the numerically computed electric field ($\mathcal{E}$) generated from a random magnetic dipole located at $(0, 0, 2)$m traveling through a random, anisotropic Gaussian lense.



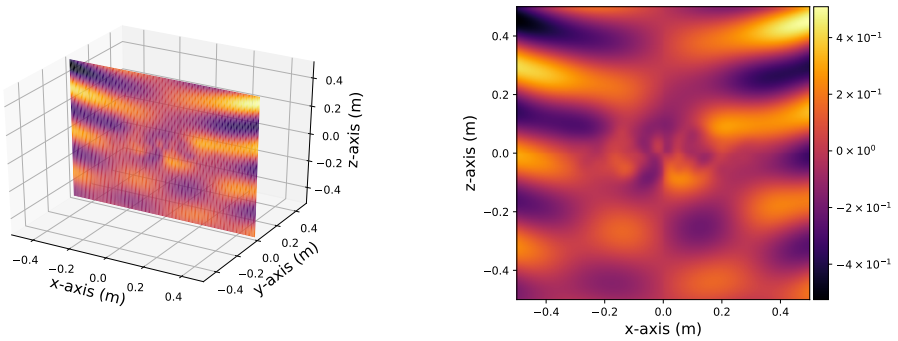
(a) x-component, xy-plane, $z = 0$.



(b) z-component, xy-plane, $z = 0$.



(c) x-component, xz-plane, $y = 0$.



(d) z-component, xz-plane, $y = 0$.

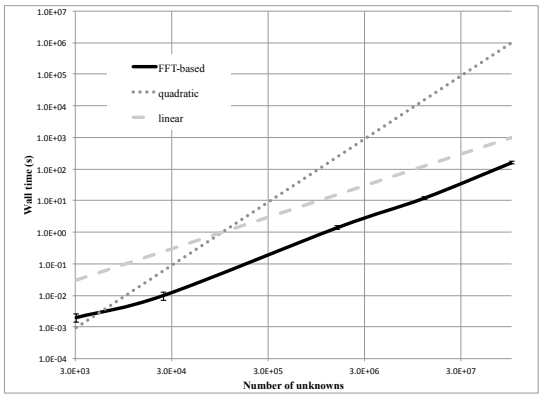


Fig. 3: Asymptotic scaling of Algorithm 3 on 1 processor. Solid line: mean wall time with error bars(one standard deviation). Dashed line: theoretical linear and quadratic scalings.

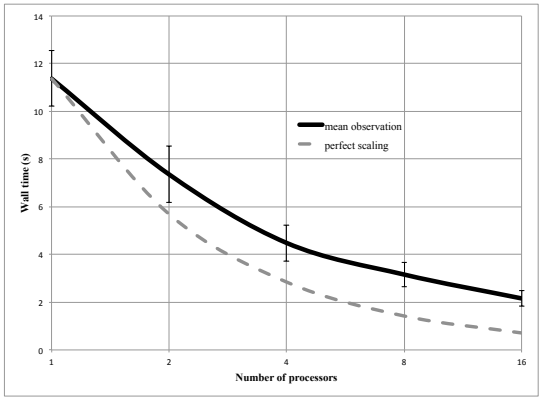


Fig. 4: Strong parallel scaling of Algorithm 3 with $N = 12,582,912$ total unknowns. Solid line: mean wall time with error bars (one standard deviation). Dashed line: perfect theoretical scaling ($\frac{t}{N}$).

The final experiment involved computing the solution to Maxwell's equations with different numbers of points per dimension and study the behavior of the error. To do so, we used a background medium consisting of isotropic permittivity and permeability distributed as Gaussians centered at the origin, with a standard deviation of 0.075 and a maximum values $10 \cdot \epsilon_0$ and $5 \cdot \mu_0$, respectively. A randomly oriented magnetic dipole located at $(0, 0, 5)$m was used as a source, and the field was computed in the unit cube $[-.5, .5]^3$ m for $N \in \{16, 32, 64, 128\}$ points per dimension. The results are presented in Table II. If F_N is the vector of electric or magnetic field measurements with N points per dimension and F_{128} is the vector of field measurements with $N = 128$, the relative error is calculated as $\frac{||F_N - F_{128}||_2}{||F_{128}||_2}$.[6] The precomputation time is the wall time in seconds for Algorithm 1. We use LGMRES to solve eqn. (12). Each iteration of LGMRES is an application of Algorithm 3. The seconds per iteration is the total solver time (with a tolerance of 10^{-14}) divided by the number of iterations. Sixteen processors were used for these simulations. As expected, the error decays exponentially fast.

[6]Only coinciding gridpoints are considered.

TABLE I: Weak scaling data

# processors	# unknowns	unknowns per processor	Ratio of times
1	1,572,864	1,572,864	
2	3,072,000	1,536,000	1.35
4	6,000,000	1,500,000	1.24
8	12,582,912	1,572,864	1.28
16	24,576,000	1,536,000	1.79

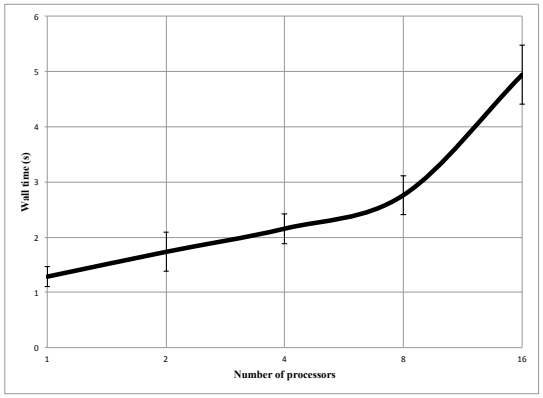


Fig. 5: Weak parallel scaling of Algorithm 3. Solid line: mean wall time with error bars (one standard deviation).

TABLE II: Grid refinement results

pts/dim	$\mathcal{E}$ rel. err.	$\mathcal{H}$ rel. err.	precompute time	# LGMRES iters	s/iter
16	1.3×10^{-3}	6.3×10^{-3}	0.067	8	0.225
32	2.6×10^{-4}	1.8×10^{-4}	0.15	10	1.64
64	3.8×10^{-7}	1.8×10^{-7}	2.31	12	143
128	–	–	25.79	12	1400

V. Conclusion

In conclusion, we have presented a generalization of a truncated kernel technique for the solution of Maxwell's equations in 3D. Our implementation of the algorithm efficiently employs FFTs, which run in parallel, enabling large scale and fast simulations. The shared-memory implementation permits us to reach problems with hundreds of millions of unknowns thanks to an advantageous scaling (Section IV). However, as the number of processes increases such advantages are diminished due to the overhead from the non-parallelized portions of the implementation, including memory management and communication between threads. To achieve exascale, one must be capable of scaling well into the 1000s of processors. To do so, a distributed processes approach (e.g. MPI [24]) is more appropriate. Unfortunately, communications associated with the FFT are well-known to inhibit such scaling [25]. To overcome this bottleneck, and as part of our future work, we intend to study the use of the *communication-avoiding FFT* [26] together with highly efficient matrix-vector and vector-vector routines on GPUs, optimized for optimal data locality and parallelization using Reservoir Labs' proprietary R-Stream compiler [27].

REFERENCES

[1] N. K. Soni, K. D. Paulsen, H. Dehghani, and A. Hartov, "Finite element implementation of maxwell's equations for image reconstruction in electrical impedance tomography," *IEEE Transactions on Medical Imaging*, vol. 25, no. 1, pp. 55–61, 2005.

[2] A. Taflove and K. Umashankar, "Radar cross section of general three-dimensional scatterers," *IEEE Transactions on electromagnetic compatibility*, no. 4, pp. 433–440, 1983.

[3] S. H. Ward and G. W. Hohmann, "Electromagnetic theory for geophysical applications," in *Electromagnetic Methods in Applied Geophysics: Voume 1, Theory*. Society of Exploration Geophysicists, 1988, pp. 130–311.

[4] D. R. Smith and J. B. Pendry, "Homogenization of metamaterials by field averaging," *JOSA B*, vol. 23, no. 3, pp. 391–403, 2006.

[5] D. Colton and R. Kress, *Inverse acoustic and electromagnetic scattering theory*. Springer Science & Business Media, 2012, vol. 93.

[6] F. Ethridge and L. Greengard, "A new fast-multipole accelerated poisson solver in two dimensions," *SIAM Journal on Scientific Computing*, vol. 23, no. 3, pp. 741–760, 2001.

[7] H. Langston, L. Greengard, and D. Zorin, "A free-space adaptive fmm-based pde solver in three dimensions," *Communications in Applied Mathematics and Computational Science*, vol. 6, no. 1, pp. 79–122, 2011.

[8] D. Malhotra and G. Biros, "Pvfmm: A parallel kernel independent fmm for particle and volume potentials," *Communications in Computational Physics*, vol. 18, no. 3, pp. 808–830, 2015.

[9] P. McCorquodale, P. Colella, G. T. Balls, and S. B. Baden, "A scalable parallel poisson solver in three dimensions with infinite-domain boundary conditions," in *2005 International Conference on Parallel Processing Workshops (ICPPW'05)*. IEEE, 2005, pp. 163–172.

[10] F. Vico, L. Greengard, and M. Ferrando, "Fast convolution with free-space green's functions," *Journal of Computational Physics*, vol. 323, pp. 191–203, 2016.

[11] H. Langston, L. Greengard, and D. Zorin, "A free-space adaptive FMM-based PDE solver in three dimensions," *Commun. Appl. Math. Comput. Sci.*, vol. 6, no. 1, pp. 79–122, 2011. [Online]. Available: https://doi.org/10.2140/camcos.2011.6.79

[12] O. Marin, O. Runborg, and A.-K. Tornberg, "Corrected trapezoidal rules for a class of singular functions," *IMA Journal of Numerical Analysis*, vol. 34, no. 4, pp. 1509–1540, 2014.

[13] J. C. Aguilar and Y. Chen, "High-order corrected trapezoidal quadrature rules for functions with a logarithmic singularity in 2-d," *Computers & Mathematics with Applications*, vol. 44, no. 8-9, pp. 1031–1039, 2002.

[14] J. Aguilar and Y. Chen, "High-order corrected trapezoidal quadrature rules for the coulomb potential in three dimensions," *Computers & Mathematics with Applications*, vol. 49, no. 4, pp. 625–631, 2005.

[15] J. T. Beale and M.-C. Lai, "A method for computing nearly singular integrals," *SIAM Journal on Numerical Analysis*, vol. 38, no. 6, pp. 1902–1925, 2001.

[16] R. Duan and V. Rokhlin, "High-order quadratures for the solution of scattering problems in two dimensions," *Journal of Computational Physics*, vol. 228, no. 6, pp. 2152–2174, 2009.

[17] J. Goodman, T. Y. Hou, and J. Lowengrub, "Convergence of the point vortex method for the 2-d euler equations," *Communications on Pure and Applied Mathematics*, vol. 43, no. 3, pp. 415–430, 1990.

[18] J. S. Lowengrub, M. J. Shelley, and B. Merriman, "High-order and efficient methods for the vorticity formulation of the euler equations," *SIAM Journal on Scientific Computing*, vol. 14, no. 5, pp. 1107–1142, 1993.

[19] P.-D. Letourneau, M. Harris, M. H. Langston, G. Papanicolaou, J. Ezick, and R. Lethin, "Large-scale imaging algorithm for low-frequency electromagnetic wave propagation and imaging in inhomogeneous media," in preparation.

[20] L. Hörmander, *Linear Partial Differential Equations*. Springer, 1976.

[21] W. Rudin, *Real and complex analysis*. Tata McGraw-hill education, 2006.

[22] L. Dagum and R. Menon, "Openmp: An industry-standard api for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998. [Online]. Available: https://doi.org/10.1109/99.660313

[23] M. Frigo and S. G. Johnson, "The design and implementation of fftw3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[24] R. Thakur, P. Balaji, D. Buntinas, D. Goodell, W. Gropp, T. Hoefler, S. Kumar, E. Lusk, and J. L. Träff, "Mpi at exascale," *Procceedings of SciDAC*, vol. 2, pp. 14–35, 2010.

[25] P. T. P. Tang, J. Park, D. Kim, and V. Petrov, "A framework for low-communication 1-d fft," *Scientific Programming*, vol. 21, no. 3-4, pp. 181–195, 2013.

[26] M. H. Langston, M. Baskaran, B. Meister, N. Vasilache, and R. Lethin, "Re-introduction of communication-avoiding fmm-accelerated ffts with gpu acceleration," in *2013 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2013, pp. 1–6.

[27] B. Meister, N. Vasilache, D. Wohlford, M. M. Baskaran, A. Leung, and R. Lethin, "R-Stream compiler," in *Encyclopedia of Parallel Computing*, D. A. Padua, Ed. Springer, 2011, pp. 1756–1765.