Journal of Computational Physics 230 (2011) 5471–5487

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A fast directional algorithm for high-frequency electromagnetic scattering

Paul Tsuji [a,*], Lexing Ying [b]

[a] ICES, University of Texas at Austin, Austin, TX 78712, USA
[b] Department of Mathematics and ICES, University of Texas at Austin, TX 78712, USA

ARTICLE INFO

Article history:
Received 16 September 2010
Received in revised form 1 February 2011
Accepted 13 February 2011
Available online 27 March 2011

Keywords:
Electromagnetic scattering
Boundary integral equations
Fast algorithms
Fast multipole methods
Sparse Fourier transforms

ABSTRACT

This paper is concerned with the fast solution of high-frequency electromagnetic scattering problems using the boundary integral formulation. We extend the $O(N\log N)$ directional multilevel algorithm previously proposed for the acoustic scattering case to the vector electromagnetic case. We also detail how to incorporate the curl operator of the magnetic field integral equation into the algorithm. When combined with a standard iterative method, this results in an almost linear complexity solver for the combined field integral equations. In addition, the butterfly algorithm is utilized to compute the far field pattern and radar cross section with $O(N\log N)$ complexity.

1. Introduction

Electromagnetic scattering from perfectly conducting objects is a well-studied problem, having applications in antenna modeling, radar engineering, and remote sensing. The most popular computational technique for solving the time-harmonic exterior scattering problem is the boundary element method, or "method of moments" as it is known in the electrical engineering community. In this method, the electric and magnetic field integral equations can be derived using the boundary conditions of a perfectly conducting object, with the unknown quantity being the fictitious current induced on the surface. In the discretization process, an N-dimensional finite element approximation is used to represent the current, while the equations can be tested using the same basis (Galerkin) or by pointwise collocation. This results in an $N \times N$ linear system of equations. Once the coefficients of the basis functions are determined, relevant quantities such as the far-field pattern and scattered field can be computed by the appropriate radiation formulas.

In many of these applications, it is often found that the operating wavelength is much smaller in scale compared to the size of the target object; that is, the diameter of the scatterer is orders of magnitude larger than λ. For the boundary element method to be reasonably accurate, each element must be of size 0.2λ or smaller; thus, for high-frequency problems, the number of unknowns N can become very large, and solving the resulting $N \times N$ linear system in a reasonable amount of time becomes a difficult task. Standard $O(N^3)$ methods such as Gaussian elimination or Cholesky factorization do not suffice. In the past few decades, there has been a great deal of effort put forth in developing fast algorithms to remedy this problem.

There are two main types of fast solvers for the system matrix: direct methods and iterative methods. Direct methods aim at computing an approximate inverse to the method-of-moments matrix and applying it to the right hand side; the benefits here are that the computational time is usually unaffected by the condition number, and multiple right hand sides can be handled with ease once the approximate inverse is generated. In the realm of fast direct solvers for integral equations of

* Corresponding author.
E-mail addresses: ptsuji@gmail.com (P. Tsuji), lexing@math.utexas.edu (L. Ying).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.02.013

scattering theory, recent works presented include the matrix compression algorithms by Michielssen, Boag, and Chew [19], as well as Martinsson and Rokhlin [18]. For the two-dimensional case of elongated structures, these algorithms have computational complexities which scale as $O(N\log^2 N)$ and $O(N)$, respectively. Canning et al. [6] and Heldring et al. [16] have presented direct solvers for the three-dimensional case which scale as $O(N^2)$ and $O(N\log^3 N)$ in the low-frequency regime; these complexity estimates break down, however, in the high-frequency regime. To the author's knowledge, there currently are no $O(N)$ or $O(N\log N)$-type direct solvers for the integral equations of high-frequency scattering in three dimensions.

On the other hand, iterative-type fast solvers combine a basic iterative method with a fast algorithm to accelerate the matrix–vector multiplication necessary at each iteration. Standard iterative methods alone have an $O(N^2)$ memory requirement, with a complexity of $O(n_{it}N^2)$; here, n_{it} is the number of iterations necessary for convergence to a desirable error tolerance. This is easily seen, as there are N^2 matrix entries needed to be stored, and direct matrix–vector multiplications take $O(N^2)$ operations. The goal of the fast algorithm is to reduce the complexity of the matrix–vector operation down to $O(N\log N)$ or $O(N)$, thus resulting in a total complexity down to $O(n_{it}N\log N)$ or $O(n_{it}N)$, respectively. In the computational electromagnetics community, a plethora of algorithms have been utilized in this setting; some of these popular methods are based on the Fast Fourier Transform (FFT). The first method of this type was the adaptive integral method [3] proposed by Bleszynski et al., which reduces the complexity of surface problems down to $O(N^{1.5}\log N)$. An improvement by Bruno and Kunyansky in [4] uses non-uniform Cartesian grids to reduce the computational cost, but still has superlinear dependence on N. The precorrected-FFT method [21], presented by Phillips and White, can reduce the complexities of electrostatic and low-frequency problems down to $O(N\log N)$, but for higher frequency problems, this estimate returns to $O(N^{1.5}\log N)$; Lee's IE-FFT method [25] has also been shown to work on PEC surface problems with similar complexity.

The more commonly used algorithms are the variations of the fast multipole method (FMM) and other low-rank factorization methods. Many of these methods start by rewriting the matrix–vector multiplication as an N-body potential problem; due to properties of the Green's function in the far field, the interactions between well-separated sets can be computed with low-rank approximations. For low-frequency problems, a wide range of low-rank factorization algorithms have been introduced, most notably Bebendorf's adaptive cross approximation [2], Lee's IE-QR algorithm [24], Kapur and Long's *IES*^{3} method [17], and Canning's simply sparse method [32]. For high-frequency problems, however, the fast multipole method has become the norm. The original FMM for the Laplace equation was adapted by Rokhlin to solve the Helmholtz equation for high frequencies in [23]. In the paper, he states that the two-level method reduces the complexity to $O(N^{3/2})$, the three-level method reduces it further to $O(N^{4/3})$, and so on. With a multilevel approach, he argues that the complexity can be reduced to $O(N\log N)$. This multilevel approach was realized in the well-celebrated multilevel fast multipole algorithm (MLFMA) by Song, Lu, and Chew [27].

The focus of this paper is to adapt two recently developed algorithms to address high-frequency electromagnetic scattering problems: the directional multilevel algorithm [10,11] for the Helmholtz kernel, and the butterfly algorithm [30] for fast radar cross section calculations. In the high-frequency case, the fast directional algorithm constructs low-rank approximations between well-separated groups; unlike other low-rank factorization methods, however, these approximations are based on direction. This allows for the solution of high-frequency problems in $O(N\log N)$ time. The advantages of the directional multilevel algorithm in the electromagnetic case are the following: first, it does not use complicated analytical expressions for the far-field approximations; it only uses the directional low-rank property of the kernel function and kernel evaluations for the translation operators. This results in a simpler implementation and ease of use. Secondly, the directional multilevel algorithm can handle both the Green's function of the Helmholtz equation and its derivatives, as well as linear operations such as the curl operator in the magnetic field integral equation, with minor modifications. In this sense, it is more user-friendly than other descendants of the FMM.

The rest of the paper is as follows: in Section 2, we formulate the boundary element method for electromagnetics, including the relevant radiation formulas needed in the post-processing stage. In Section 3, we discuss how to apply the directional multilevel algorithm and the butterfly algorithm to the vector Maxwell equations. In Section 4, we present numerical results with canonical examples, and show that the complexity of our method scales as $O(N\log N)$. Finally, we conclude with some remarks in Section 5.

2. Boundary integral formulation and discretization

2.1. Surface integral equations

We begin with a brief review of surface integral equations for scattering from perfectly conducting objects. The discussion here mostly follows the method-of-moments formulation in [20]. Before we proceed, however, we must comment that their formulation assumes the continuity equation with the opposite sign, i.e. $\nabla\cdot\vec{J}=\imath\omega\rho$. This results in a sign change within the electric field integral equation. For consistency, we have assumed the time convention $\vec{\mathscr{J}}(\vec{r},t)=\vec{J}(\vec{r})e^{\imath\omega t}$, resulting in the continuity equation $\nabla\cdot J=-\imath\omega\rho$. All other fields follow the same convention, i.e. $\vec{\mathscr{E}}(\vec{r},t)=\vec{E}(\vec{r})e^{\imath\omega t}$, $\vec{\mathscr{H}}(\vec{r},t)=\vec{H}(\vec{r})e^{\imath\omega t}$, and so on.

Consider the scattering domain D with boundary $\Gamma=\partial D$; note that D does not necessarily have to be connected, so there can be more than one scattering object. Let us denote the incident fields $(\vec{E}_i,\vec{H}_i)$, the scattered fields $(\vec{E}_s,\vec{H}_s)$, and the total fields $(\vec{E},\vec{H})=(\vec{E}_i+\vec{E}_s,\vec{H}_i+\vec{H}_s)$. The time-harmonic Maxwell equations for the scattered fields in the exterior of the scatterers are

$$\begin{aligned}
&\nabla \times \vec{E}_s = -\imath\omega\mu \vec{H}_s \\
&\nabla \times \vec{H}_s = \imath\omega\varepsilon \vec{E}_s + \vec{J} \\
&\nabla \cdot \vec{E}_s = \frac{\rho}{\varepsilon} \\
&\nabla \cdot \vec{H}_s = 0 \\
&\lim_{|\vec{r}|\to\infty} (\vec{H}_s \times \vec{r} - |\vec{r}|\,\vec{E}_s) = 0 \\
&\lim_{|\vec{r}|\to\infty} (\vec{E}_s \times \vec{r} + |\vec{r}|\,\vec{H}_s) = 0,
\end{aligned} \tag{2.1}$$

where $\imath = \sqrt{-1}$, ε is the permittivity of the surrounding medium, μ is the permeability, ω is the angular frequency, and $\vec{J}$ is the fictitious induced current on the surface of the scatterer which satisfies the continuity equation $\nabla \cdot \vec{J} = -\imath\omega\rho$. In this paper, we take the surrounding medium to be free space, i.e. $\mu = 4\pi \cdot 10^{-7}$ and $\varepsilon \approx 8.854 \times 10^{-12}$. The last two conditions are known as the Silver–Müller radiation conditions; these equations guarantee uniqueness of the solution to the exterior problem. The boundary conditions enforced on Γ for the total fields are

$$\begin{aligned}
&\hat{n}(\vec{r}) \times \vec{E}(\vec{r}) = 0, \\
&\hat{n}(\vec{r}) \times \vec{H}(\vec{r}) = \vec{J}(\vec{r}) \quad \vec{r} \in \Gamma,
\end{aligned} \tag{2.2}$$

where $\hat{n}$ is the outward unit normal vector on Γ. The wavenumber k is equal to $\omega\sqrt{\mu\varepsilon}$ and the wavelength λ is $2\pi/k$. In this paper, we scale the geometry so that the wavelength $\lambda = 1$ (and equivalently $k = 2\pi$). The diameter of the scaled object is then denoted by K; that is, the object is centered at the origin and contained inside the box $\left[-\frac{K}{2}, \frac{K}{2}\right]^3$. For a problem in the high-frequency regime, K is typically quite large.

Using the standard machinery for Green's functions of the vector Helmholtz equation, we can formulate the exterior scattered fields in terms of the current distribution $\vec{J}(\vec{r})$ for $\vec{r} \in \Gamma$, i.e. the radiation formulas are

$$\vec{E}_s(\vec{r}) = -\imath\omega\mu \int_\Gamma G(\vec{r},\vec{r}') \left\{ \vec{J}(\vec{r}') + \frac{1}{k^2} \nabla'\nabla' \cdot \vec{J}(\vec{r}') \right\} d\vec{r}', \vec{H}_s(\vec{r}) = \nabla \times \int_\Gamma G(\vec{r},\vec{r}') \vec{J}(\vec{r}') d\vec{r}', \tag{2.3}$$

where $G(\vec{r},\vec{r}')$ is the Green's function for the scalar Helmholtz equation

$$G(\vec{r},\vec{r}') = \frac{e^{-\imath k|\vec{r}-\vec{r}'|}}{4\pi|\vec{r}-\vec{r}'|}. \tag{2.4}$$

Using the boundary condition $\vec{J} = \hat{n} \times \vec{H}$ and taking the limit as $\vec{r}$ approaches the surface, one can derive the electric field integral equation (EFIE) and magnetic field integral equation (MFIE), which respectively are

$$\hat{n}(\vec{r}) \times \vec{E}_i(\vec{r}) = \hat{n}(\vec{r}) \times \imath\omega\mu \left\{ \int_\Gamma G(\vec{r},\vec{r}') \vec{J}(\vec{r}') d\vec{r}' + \frac{1}{k^2} \nabla \int_\Gamma G(\vec{r},\vec{r}') \nabla' \cdot \vec{J}(\vec{r}') d\vec{r}' \right\}, \tag{2.5}$$

$$\hat{n}(\vec{r}) \times \vec{H}_i(\vec{r}) = \frac{\vec{J}(\vec{r})}{2} - \hat{n}(\vec{r}) \times \int_{\Gamma\setminus\{\vec{r}\}} J(\vec{r}') \times \nabla' G(\vec{r},\vec{r}') d\vec{r}'. \tag{2.6}$$

Here, the differential operators are redistributed for convenience in the discretization process. In order to get a better conditioned system of equations, one typically takes a convex linear combination of the EFIE and MFIE to get the combined field integral equation (CFIE). For a constant α between 0 and 1, the CFIE is

$$\mathrm{CFIE} = \alpha\mathrm{EFIE} + (1-\alpha)\eta\mathrm{MFIE}, \tag{2.7}$$

where $\eta = \sqrt{\frac{\mu}{\varepsilon}}$ is the intrinsic impedance.

2.2. Method-of-moments discretization

For any given scatterer, we first discretize its boundary surface Γ with a triangular mesh that has a constant number of triangles per wavelength λ; the typical edge length h of the triangles is on the order of λ but significantly smaller. Let us denote the triangles of the mesh by $\{T_i\}_{i=1}^{N_t}$ where N_t is the number of triangles and $\Gamma = \bigcup_{i=1}^{N_t} T_i$. The discretization of (2.7) is then performed using the standard RWG basis functions presented in [22]. Consider the RWG basis functions $\{\vec{f}_n\}_{n=1}^{N}$ defined on the triangular mesh of Γ, where $N = O(N_t)$ is the number of total edges of the mesh for a closed surface (for an open surface, basis functions are only defined on interior edges). Since the scatterer surface is discretized with a near-constant number of triangles per unit area and the surface area is of order $O(K^2)$, $N = O(K^2)$. The support Γ_n of each $\vec{f}_n$ is supported on a pair of triangles $(T_{n,1}, T_{n,2})$, i.e. $\Gamma_n = T_{n,1} \cup T_{n,2}$. The current $\vec{J}(\vec{r})$ is then expanded in terms of N basis functions,

$$\vec{J}(\vec{r}) = \sum_{n=1}^{N} j_n \vec{f}_n(\vec{r}), \tag{2.8}$$

where $\{j_n\}_{n=1}^{N} \in \mathbb{C}$ are the coefficients to be determined. Seeking a solution of this form, we plug expansion (2.8) into (2.5) and (2.6) with constants $\omega\mu = k\eta$ to get the discretized integral equations

$$\hat{n}(\vec{r}) \times \vec{E}_i(\vec{r}) = \hat{n}(\vec{r}) \times \sum_{n=1}^{N} j_n \left\{ \imath k\eta \int_{\Gamma} G(\vec{r},\vec{r}')\vec{f}_n(\vec{r}')d\vec{r}' - \frac{\eta}{\imath k}\nabla \int_{\Gamma} G(\vec{r},\vec{r}')\nabla' \cdot \vec{f}_n(\vec{r}')d\vec{r}' \right\},$$

$$\hat{n}(\vec{r}) \times \vec{H}_i(\vec{r}) = \frac{1}{2}\sum_{n=1}^{N} j_n \vec{f}_n(\vec{r}) - \hat{n}(\vec{r}) \times \sum_{n=1}^{N} j_n \int_{\Gamma\setminus\{\vec{r}\}} \vec{f}_n(\vec{r}') \times \nabla' G(\vec{r},\vec{r}')d\vec{r}'.$$

The next step is to test the integral equations by using the Galerkin method or another basis to obtain a linear system of equations. Since the emphasis of this paper is on the application of fast multilevel algorithms and not on discretization techniques, we choose the simplest testing functions for efficient implementation; that is, razor-blade testing functions for the EFIE and point-matching for the MFIE. For the details, please refer to Chapter 10 of [20]. With this formulation, the $N \times N$ linear system of equations obtained is

$$ZI = V, \tag{2.9}$$

where the entries of Z, I, and V are

$$\begin{aligned}
Z_{mn} &= \alpha A_{mn} + (1-\alpha)\eta\Delta t_m B_{mn},\\
A_{mn} &= \imath k\eta \int_{C_m}\int_{\Gamma_n} G(\vec{r},\vec{r}')\vec{f}_n(\vec{r}')d\vec{r}' \cdot d\vec{r} -\\
&\frac{\eta}{\imath k}\int_{C_m} \nabla\left[\int_{\Gamma_n} G(\vec{r},\vec{r}')\nabla' \cdot \vec{f}_n(\vec{r}')d\vec{r}'\right] \cdot d\vec{r},\\
B_{mn} &= \hat{e}_m \cdot \int_{\Gamma_n} \vec{f}_n(\vec{r}') \times \nabla G(\vec{r},\vec{r}')d\vec{r}'\Big|_{\vec{r}=\vec{r}_m}, \quad m \neq n\\
B_{mm} &= \frac{2\pi - \Omega_m}{2\pi}\\
I_n &= j_n,\\
V_m &= \alpha \int_{C_m} \vec{E}_i(\vec{r}) \cdot d\vec{r} + (1-\alpha)\eta\Delta t_m \hat{e}_m \cdot \vec{H}_i(\vec{r}_m).
\end{aligned} \tag{2.10}$$

Here, C_m is the path of the razor-blade function along the surface of the mth RWG, $\hat{e}_m$ is the unit vector in the direction of the adjoining edge of the mth RWG (oriented so that $\hat{n} \times \hat{e}_m$ points in the direction of the basis function, for either triangle), $\vec{r}_m$ is the center of the same adjoining edge, and Ω_m is the interior angle created by the two triangles $(T_{m,1}, T_{m,2})$ (see Fig. 1 for an illustration of these notations).

For the outer integrals in the matrix entries of A, we use the formulas

$$\int_{C_m} \nabla\phi(\vec{r}) \cdot d\vec{r} = \phi(\vec{c}_{m,2}) - \phi(\vec{c}_{m,1}), \quad \int_{C_m} \vec{F}(\vec{r}) \cdot d\vec{r} \approx \vec{F}(\vec{c}_{m,1}) \cdot \vec{t}_{m,1} + \vec{F}(\vec{c}_{m,2}) \cdot \vec{t}_{m,2}, \tag{2.11}$$

where $\vec{c}_{m,1}$ and $\vec{c}_{m,2}$ are the centroids of $T_{m,1}$ and $T_{m,2}$, respectively, and $\vec{t}_{m,1}, \vec{t}_{m,2}$ are the vectors prescribed by the razor-blade function at the centroids. If we set $\phi(\vec{r}) = \int_{\Gamma_n} G(\vec{r},\vec{r}')\nabla' \cdot \vec{f}_n(\vec{r}')d\vec{r}'$ and $\vec{F}(\vec{r}) = \int_{\Gamma_n} G(\vec{r},\vec{r}')\vec{f}_n(\vec{r}')d\vec{r}'$, the matrix entries of A now become

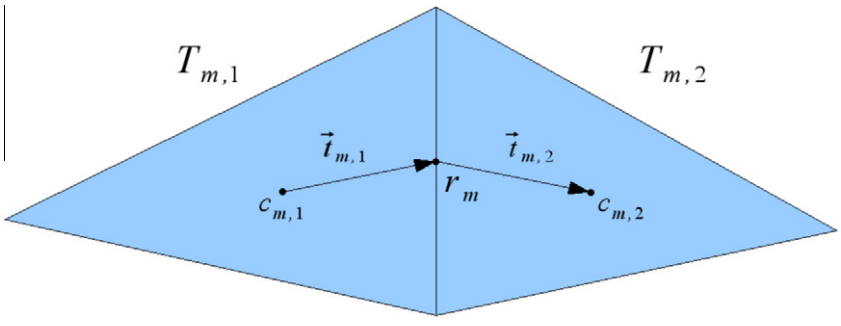


Fig. 1. Illustration of a razor-blade function defined on the mth RWG basis function.

$$A_{mn} = \imath k\eta\left(\vec{t}_{m,1}\cdot\int_{\Gamma_n} G(\vec{c}_{m,1},\vec{r}')\vec{f}_n(\vec{r}')d\vec{r}' + \vec{t}_{m,2}\cdot\int_{\Gamma_n} G(\vec{c}_{m,2},\vec{r}')\vec{f}_n(\vec{r}')d\vec{r}'\right) - \frac{\eta}{\imath k}\left(\int_{\Gamma_n} G(\vec{c}_{m,2},\vec{r}')\nabla'\cdot\vec{f}_n(\vec{r}')d\vec{r}' - \int_{\Gamma_n} G(\vec{c}_{m,1},\vec{r}')\nabla'\cdot\vec{f}_n(\vec{r}')d\vec{r}'\right).$$

It is clear that the computation of the matrix entries above first involve the evaluation of potential-type integrals at $\vec{c}_{m,1}, \vec{c}_{m,2}$, and $\vec{r}_m$ for $m = 1,2,\ldots,N$. This observation will become important in the very next section.

2.3. Iterative solution

To solve the linear system (2.9) efficiently, we must use an iterative method such as GMRES. Thus, in order to construct the Krylov subspace, at each iteration a density vector I is given and a summation of the form

$$\sum_{n=1}^{N} Z_{mn} I_n \tag{2.12}$$

for $m = 1,2,\ldots,N$ must be computed. If the matrix entries are done explicitly and each summation is performed directly, this obviously yields a complexity of $O(N^2)$, with an $O(N^2)$ memory requirement. Instead, we will modify the approach by considering the evaluation of the potential integrals at the test points first, before applying the test vectors. That is, let us define the following integral operators

$$\begin{aligned} U[\vec{J}](\vec{r}) &= \sum_{i=1}^{N_t}\int_{T_i} G(\vec{r},\vec{r}')\vec{J}(\vec{r}')d\vec{r}', \\ V[\vec{J}](\vec{r}) &= \sum_{i=1}^{N_t}\int_{T_i} G(\vec{r},\vec{r}')\nabla'\cdot\vec{J}(\vec{r}')d\vec{r}', \\ W[\vec{J}](\vec{r}) &= \sum_{i=1}^{N_t}\int_{T_i} \vec{J}(\vec{r}')\times\nabla G(\vec{r},\vec{r}')d\vec{r}', \end{aligned} \tag{2.13}$$

where the integrals are now over each triangle instead of each RWG. Using these operators and (2.10), the summation in (2.12) is simply

$$\begin{aligned} \sum_{n=1}^{N} Z_{mn} I_n = \alpha\left\{\imath k\eta\left(\vec{t}_{m,1}\cdot U[\vec{J}](\vec{c}_{m,1}) + \vec{t}_{m,2}\cdot U[\vec{J}](\vec{c}_{m,2})\right) - \frac{\eta}{\imath k}\left(V[\vec{J}](\vec{c}_{m,2}) - V[\vec{J}](\vec{c}_{m,1})\right)\right\} \\ + (1-\alpha)\eta\Delta t_m\left\{\hat{e}_m\cdot W[\vec{J}](\vec{r}_m) + \frac{2\pi-\Omega_m}{2\pi} j_m\right\}. \end{aligned} \tag{2.14}$$

At each iteration, we can use expansion (2.8) to interpolate the current on every triangle given the input $\{j_n\}_{n=1}^{N}$. It can be observed from the matrix entries that our task is to evaluate $U[\vec{J}](\vec{r})$ and $V[\vec{J}](\vec{r})$ at points $\{\vec{c}_\ell\}_{\ell=1}^{N_t}$ and $W[\vec{J}](\vec{r})$ at points $\{\vec{r}_m\}_{m=1}^{N}$. Since these integrals cannot be analytically evaluated in most cases, it is necessary to introduce a numerical quadrature scheme. First, consider the evaluation of $U[\vec{J}](\vec{c}_\ell)$ and $V[\vec{J}](\vec{c}_\ell)$. If $\vec{c}_\ell \notin T_i$, then symmetric Gaussian quadrature in [28] can be used over T_i; if $\vec{c}_\ell \in T_i$, however, we must use a quadrature rule especially constructed to handle the $\frac{1}{r}$ singularity in the integral. To handle this, we subdivide the triangle T_i into three new triangles which share $\vec{c}_\ell$ as a vertex, and use Duffy quadrature proposed in [9] over each of the new triangles (see Fig. 2).

For a triangle T_i, let $\{\vec{p}_{i,q}\}_{q=1}^{Q}$ and $\{\alpha_{i,q}\}_{q=1}^{Q}$ be the nodes and weights of the Gaussian quadrature rule and let $\{\vec{t}_{i,s}\}_{s=1}^{S}$ and $\{\beta_{i,s}\}_{s=1}^{S}$ be the unions of the three sets of Duffy quadrature nodes and weights (one for each of the three new triangles). It is important to note here that the weights take into account the area when integrating over T_i, i.e. that the Jacobian is already built into each set of weights. We can then approximate the integrals for each $\vec{c}_\ell$ as

$$U[\vec{J}](\vec{c}_\ell) \approx \sum_{\substack{i=1\\ i\neq\ell}}^{N_t}\sum_{q=1}^{Q} G(\vec{c}_\ell,\vec{p}_{i,q})\vec{J}(\vec{p}_{i,q})\alpha_{i,q} + \sum_{s=1}^{S} G(\vec{c}_\ell,\vec{t}_{\ell,s})\vec{J}(\vec{t}_{\ell,s})\beta_{\ell,s},$$

$$V[\vec{J}](\vec{c}_\ell) \approx \sum_{\substack{i=1\\ i\neq\ell}}^{N_t}\sum_{q=1}^{Q} G(\vec{c}_\ell,\vec{p}_{i,q})\nabla\cdot\vec{J}(\vec{p}_{i,q})\alpha_{i,q} + \sum_{s=1}^{S} G(\vec{c}_\ell,\vec{t}_{\ell,s})\nabla\cdot\vec{J}(\vec{t}_{\ell,s})\beta_{\ell,s}.$$

The main computation is the first double sum in each formula, since for each $\vec{c}_\ell$ one needs to sum over $O(q\,N_t)$ terms. It is not easy to apply fast summation algorithms directly to this sum, since the summation is performed over a set that depends on $\vec{c}_\ell$, i.e., the constraint $i \neq \ell$. In order to facilitate the fast summation, we rewrite the formulas as

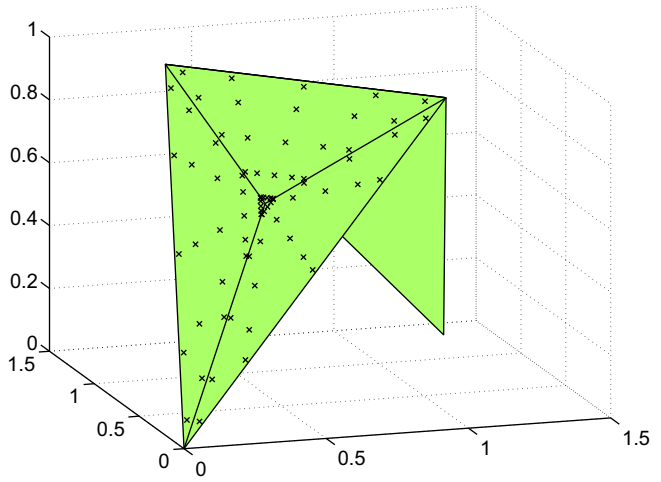


Fig. 2. Integrating over an RWG basis function with a singularity at the centroid of one triangle. The triangle is subdivided into 3 new triangles. The ×'s mark the location of the Duffy quadrature nodes.

$$U[\vec{J}](\vec{c}_\ell) \approx \sum_{i=1}^{N_t} \sum_{q:\vec{c}_\ell \neq \vec{p}_{i,q}} G(\vec{c}_\ell, \vec{p}_{i,q})\, \vec{J}(\vec{p}_{i,q})\alpha_{i,q} - \sum_{q:\vec{c}_\ell \neq \vec{p}_{\ell,q}} G(\vec{c}_\ell, \vec{p}_{\ell,q})\, \vec{J}(\vec{p}_{\ell,q})\alpha_{\ell,q} + \sum_{s=1}^{S} G(\vec{c}_\ell, \vec{t}_{\ell,s})\, \vec{J}(\vec{t}_{\ell,s})\beta_{\ell,s},$$

$$V[\vec{J}](\vec{c}_\ell) \approx \sum_{i=1}^{N_t} \sum_{q:\vec{c}_\ell \neq \vec{p}_{i,q}} G(\vec{c}_\ell, \vec{p}_{i,q}) \nabla \cdot \vec{J}(\vec{p}_{i,q})\alpha_{i,q} - \sum_{q:\vec{c}_\ell \neq \vec{p}_{\ell,q}} G(\vec{c}_\ell, \vec{p}_{\ell,q}) \nabla \cdot \vec{J}(\vec{p}_{\ell,q})\alpha_{\ell,q} \sum_{s=1}^{S} G(\vec{c}_\ell, \vec{t}_{\ell,s}) \nabla \cdot \vec{J}(\vec{t}_{\ell,s})\beta_{\ell,s}.$$

The advantage of this form is that now the summation is essentially over all possible pairs (i,q), except the case $\vec{c}_\ell = \vec{p}_{i,q}$, which can be handled easily in fast summation algorithms.

For the evaluation of $W[\vec{J}](\vec{r}_m)$, we again use Gaussian quadrature nodes when $\vec{r}_m \notin T_i$. For $\vec{r}_m \in T_i$, since $\nabla G(\vec{r}_m, \vec{r})$ lies in the plane of T_i, $\vec{f}_n(\vec{r}) \times \nabla G(\vec{r}_m, \vec{r})$ is perpendicular to T_i; thus, when dotted with $\hat{e}_m$, the contribution from T_i is zero. There is no need for singularity correction quadrature, and we simply have

$$W[\vec{J}](\vec{r}_m) \approx \sum_{i=1}^{N_t} \sum_{q} \alpha_{i,q}\, \vec{J}(\vec{p}_{i,q}) \times \nabla G(\vec{r}_m, \vec{p}_{i,q}).$$

The matrix–vector multiply can be summarized in the following process:

1. For $\ell = 1,2,\ldots,N_t$ and $m = 1,2,\ldots,N$, compute the sums

$$U_1[\vec{J}](\vec{c}_\ell) = \sum_{i=1}^{N_t} \sum_{q:\vec{c}_\ell \neq \vec{p}_{i,q}} G(\vec{c}_\ell, \vec{p}_{i,q})\, \vec{J}(\vec{p}_{i,q})\alpha_{i,q},$$

$$V_1[\vec{J}](\vec{c}_\ell) = \sum_{i=1}^{N_t} \sum_{q:\vec{c}_\ell \neq \vec{p}_{i,q}} G(\vec{c}_\ell, \vec{p}_{i,q}) \nabla \cdot \vec{J}(\vec{p}_{i,q})\alpha_{i,q}, \tag{2.15}$$

$$W[\vec{J}](\vec{r}_m) = \sum_{i=1}^{N_t} \sum_{q} \alpha_{i,q}\, \vec{J}(\vec{p}_{i,q}) \times \nabla G(\vec{r}_m, \vec{p}_{i,q}).$$

2. For $\ell = 1,2,\ldots,N_t$, compute the near-field terms of Gaussian quadrature, i.e.

$$U_2[\vec{J}](\vec{c}_\ell) = \sum_{q:\vec{c}_\ell \neq \vec{p}_{\ell,q}} G(\vec{c}_\ell, \vec{p}_{\ell,q})\, \vec{J}(\vec{p}_{\ell,q})\alpha_{\ell,q},$$

$$V_2[\vec{J}](\vec{c}_\ell) = \sum_{q:\vec{c}_\ell \neq \vec{p}_{\ell,q}} G(\vec{c}_\ell, \vec{p}_{\ell,q}) \nabla \cdot \vec{J}(\vec{p}_{\ell,q})\alpha_{\ell,q}.$$

3. For $\ell = 1,2,\ldots,N_t$, compute the near-field terms of Duffy quadrature, i.e.

$$U_3[\vec{J}](\vec{c}_\ell) = \sum_s G(\vec{c}_\ell, \vec{t}_{\ell,s})\,\vec{J}(\vec{t}_{\ell,s})\beta_{\ell,s},$$

$$V_3[\vec{J}](\vec{c}_\ell) = \sum_s G(\vec{c}_\ell, \vec{t}_{\ell,s})\nabla\cdot\vec{J}(\vec{t}_{\ell,s})\beta_{\ell,s}.$$

4. For $\ell = 1,2,\ldots,N_t$, compute the corrected sums

$$U[\vec{J}](\vec{c}_\ell) = U_1[\vec{J}](\vec{c}_\ell) - U_2[\vec{J}](\vec{c}_\ell) + U_3[\vec{J}](\vec{c}_\ell),$$
$$V[\vec{J}](\vec{c}_\ell) = V_1[\vec{J}](\vec{c}_\ell) - V_2[\vec{J}](\vec{c}_\ell) + V_3[\vec{J}](\vec{c}_\ell).$$

5. For $m = 1,2,\ldots,N$, calculate the sums in (2.12) through (2.14).

Here, steps 2, 3, and 4 are of $O(N_t) = O(N)$ complexity, since the number of quadrature nodes over each triangle remains constant and the number of triangles is always less than the number of basis functions. It is clear that step 5 also takes $O(N)$ operations. The only step that requires fast summation is step 1, and in Section 3.1 we show how to reduce its cost down to $O(N\log N)$ by applying the directional multilevel algorithm.

2.4. Far-field pattern and RCS

Once the coefficients of the RWG basis functions are found, the scattered field in $\mathbb{R}^3$ can be computed by reinserting the expansion (2.8) into (2.3). If $\vec{r}$ is in the far field, i.e. $|\vec{r}| \gg \lambda$, then the integral can be approximated by the leading order as

$$\vec{E}_s(\vec{r}) \approx -\imath\omega\mu\frac{e^{-\imath k|\vec{r}|}}{4\pi|\vec{r}|}\int_\Gamma \vec{J}(\vec{r}')e^{\imath k\hat{r}\cdot\vec{r}'}\,d\vec{r}',$$

where $\hat{r} = \frac{\vec{r}}{|\vec{r}|}$. The radar cross section (RCS) in the direction $\hat{r}$ is defined by

$$\sigma(\hat{r}) = \lim_{\rho\to\infty} 4\pi\rho^2\frac{|\vec{E_s}(\rho\hat{r})|^2}{|\vec{E_i}(\rho\hat{r})|^2} = \frac{(\omega\mu)^2}{4\pi}\left|\int_\Gamma \vec{J}(\vec{r}')e^{\imath k\hat{r}\cdot\vec{r}'}\,d\vec{r}'\right|^2, \tag{2.16}$$

when $|\vec{E_i}(\vec{r})| = 1$ for the incoming plane wave. Since the integral in (2.16) is non-singular, we can use Gaussian quadrature over each triangle; the task of calculating the scattered far field is now doing the summation

$$\int_\Gamma \vec{J}(\vec{r}')e^{\imath k\hat{r}\cdot\vec{r}'}\,d\vec{r}' \approx \sum_{i=1}^{N_t}\sum_q \alpha_{i,q}\vec{J}(\vec{p}_{i,q})e^{\imath k\hat{r}\cdot\vec{p}_{i,q}}. \tag{2.17}$$

Typically, in order to resolve the radar cross section, one samples the unit sphere with a discrete set of directions $\{\hat{r}_s\}_{s=1}^{N_{\hat{r}}}$, where $N_{\hat{r}}$ is on the same order as $N = O(K^2)$. Therefore, calculation of the RCS for these directions is equivalent to evaluating

$$\vec{A}(\hat{r}_s) := \sum_{i=1}^{N_t}\sum_{q=1}^{Q}\alpha_{i,q}\vec{J}(\vec{p}_{i,q})e^{\imath k\hat{r}_s\cdot\vec{p}_{i,q}}. \tag{2.18}$$

Since both the number of directions $N_{\hat{r}}$ and the number of quadrature points QN_t is $O(N)$, the direct computation of (2.18) will be of $O(N^2)$ complexity. Once again, we will propose an application of the butterfly algorithm for sparse Fourier transforms in Section 3.2; this will reduce the complexity down to $O(N\log N)$.

3. Fast summation algorithms

In this section, we adapt two algorithms proposed recently in [10,30] to speed up the computation of (2.15) and (2.18).

3.1. Directional multilevel algorithm

The tool for speeding up the computation in (2.15) is based on the directional multilevel algorithm for the high-frequency Helmholtz kernel proposed in [10]. The general problem considered in [10] is the evaluation of the potentials $\{u_i\}_{i=1}^{N_z}$ defined by

$$u_i = \sum_{\substack{j=1\\ j\neq i}}^{N_z} G(\vec{z}_i, \vec{z}_j)f_j, \tag{3.1}$$

where $\{f_i\}_{i=1}^{N_z}$ are the point sources located at $\{\vec{z}_i\}_{i=1}^{N_z} \subset [-K/2, K/2]^3$ and $G(x,y)$ is the Green's function of the scalar Helmholtz equation in (2.4). The directional multilevel algorithm in [10] provides an efficient and accurate method for evaluating all $\{u_i\}_{i=1}^{N_z}$ in $O(N_z\log N_z)$ time. A brief outline of this algorithm is contained in Appendix A.

Let us now discuss how to adapt (2.15) to the form of (3.1). We first observe that in (3.1) there is only one set of points that serve both as "sources" and "targets," while in (2.15) the source points and target points are different. To address this difference, we can combine the source points and target points into a single set of points, assign zero weights to the target points, apply the fast summation algorithm to the combined set of points, and extract the potentials only at the target points. Clearly, this does not change the complexity of the algorithm; from now on, we can safely assume that the source and target points can be different.

If we denote $\vec{u}_\ell = U_1[\vec{J}](\vec{c}_\ell), j(i,q) = (i-1)Q + q, N_y = N_t Q, \vec{y}_j = \vec{p}_{i,q}$, and $\vec{f}_j = \alpha_{i,q}\vec{J}(\vec{p}_{i,q})$, then we can rewrite the first summation of (2.15) as

$$\vec{u}_\ell = \sum_{\substack{j=1 \\ \vec{c}_\ell \neq \vec{y}_j}}^{N_y} G(\vec{c}_\ell, \vec{y}_j)\vec{f}_j,$$

where $\vec{u}_\ell = (u_{\ell,1}, u_{\ell,2}, u_{\ell,3})$ and $\vec{f}_j = (f_{j,1}, f_{j,2}, f_{j,3})$. Similarly, if we define $v_\ell = V_1[\vec{J}](\vec{c}_\ell)$ and $g_j = \alpha_{i,q}\nabla \cdot \vec{J}(\vec{p}_{i,q})$, the second summation of (2.15) is in the form

$$v_\ell = \sum_{\substack{j=1 \\ \vec{c}_\ell \neq \vec{y}_j}}^{N_y} G(\vec{c}_\ell, \vec{y}_j) g_j.$$

Thus, we now have the forms in which we are able to apply the directional multilevel algorithm. The summation for v_ℓ is in the exact same form as (3.1), i.e. the product of an $N_t \times N_y$ matrix and an $N_y \times 1$ vector. The summation for $\vec{u}_\ell$ is also of the same form, only now there are three components to sum over; instead of a matrix–vector operation, it is a multiplication of an $N_t \times N_y$ matrix with an $N_y \times 3$ matrix. In practice, since the discretization points are the same for both sums, we can aggregate the computations by combining the three components from $\vec{f}_j$ with g_j. By defining $\vec{\alpha}_\ell = (u_{\ell,1}, u_{\ell,2}, u_{\ell,3}, v_\ell)$ and $\vec{\beta}_j = (f_{j,1}, f_{j,2}, f_{j,3}, g_j)$, the summations are of the form

$$\vec{\alpha}_\ell = \sum_{\substack{j=1 \\ \vec{c}_\ell \neq \vec{y}_j}}^{N_y} G(\vec{c}_\ell, \vec{y}_j)\vec{\beta}_j.$$

Here, instead of running the directional multilevel algorithm four times (once for each component), we run the algorithm once but vectorize all of the translation operators to handle multiple columns. This process is fairly simple, since the equivalent source and potential locations remain the same; the only difference is that the sources and potentials take on vector values. When computing the equivalent sources or any of the translation operators, matrix–matrix products are used instead of matrix–vector products. The resulting complexity of this computation is $O(N_y \log N_y)$ where we recall $N_y = N_t Q$. Since Q is constant and the number of triangles N_t is always less than the number of unknowns, this complexity is $O(N \log N)$.

Using the same notation for $j, \vec{y}_j, N_y$, and $\vec{f}_j$, the third summation of (2.15) is of the form

$$\vec{w}_m = \sum_{j=1}^{N_y} \nabla G(\vec{r}_m, \vec{y}_j) \times \vec{f}_j,$$

where $\vec{w}_m = W[\vec{J}](\vec{r}_m) = (w_{m,1}, w_{m,2}, w_{m,3})$ and the gradient operator $\nabla = (\partial_x, \partial_y, \partial_z)$ acts on the first argument of G. Rewriting the cross product as a matrix–vector operation, we see that

$$\begin{pmatrix} w_{m,1} \\ w_{m,2} \\ w_{m,3} \end{pmatrix} = \sum_{j=1}^{N_y} \begin{pmatrix} 0 & -G_z(\vec{r}_m, \vec{y}_j) & G_y(\vec{r}_m, \vec{y}_j) \\ G_z(\vec{r}_m, \vec{y}_j) & 0 & -G_x(\vec{r}_m, \vec{y}_j) \\ -G_y(\vec{r}_m, \vec{y}_j) & G_x(\vec{r}_m, \vec{y}_j) & 0 \end{pmatrix} \begin{pmatrix} f_{j,1} \\ f_{j,2} \\ f_{j,3} \end{pmatrix}. \tag{3.2}$$

At first, it may seem as if the directional multilevel algorithm cannot be applied to this situation. If we make a minor modification, however, it will become apparent that the algorithm can work for (3.2). More specifically, let us look at the first component,

$$w_{m,1} = \sum_{j=1}^{N_y} (-G_z(\vec{r}_m, \vec{y}_j) f_{j,2} + G_y(\vec{r}_m, \vec{y}_j) f_{j,3}). \tag{3.3}$$

The essential step of the directional multilevel algorithm is the construction of the equivalent sources. For a leaf level box B in the low-frequency regime, (A.4) states that the potential field produced by an arbitrary set of sources inside B can be well approximated by a small set of equivalent sources in B, when observing the far field of B. As G_z and G_y are derivatives of the Green's function, the sources in (3.3) are essentially dipoles in the z and y directions. Since the field generated by a dipole can be well approximated by a group or a distribution of monopoles (i.e., sources determined by the Green's function) in its vicinity, the field generated by points in B with the kernels of (3.3) can also be approximated by a small set of equivalent sources, when observing the far field of B. Moreover, notice that in (A.4) the construction of the equivalent sources for a

box B requires only the potentials at $\left\{x_p^B\right\}$; clearly, these potentials can be evaluated directly using the formulas of the kernels $G_z(\cdot,\vec{y}_j)$ and $G_y(\cdot,\vec{y}_j)$. We would like to emphasize two points here: first, the equivalent sources are always computed using the Green's function even though the kernels of (3.3) are the derivatives; second, once the true sources of (3.3) are transformed into equivalent sources at the leaf level, the computation in the high-frequency regime only involves the Green's function $G(x,y)$, and hence requires no modification at all. Compared with the algorithm for the Green's function $G(x,y)$ in Appendix A, the algorithm for the kernels of (3.3) requires only the following modifications:

- For the computation of the equivalent sources for the leaf boxes in the low-frequency regime, use the kernels of (3.3) to compute the potentials at $\left\{x_p^B\right\}$.
- Use the kernels of (3.3) for direct near-field calculations in the low-frequency regime.

The computation of the second and third components $w_{m,2}$ and $w_{m,3}$ can be handled in essentially the same way. In fact, since the algorithm for $w_{m,1}$, $w_{m,2}$, and $w_{m,3}$ above the leaf level is exactly the same, we again aggregate the computations by vectorizing the translation operations for all three components, allowing us to run the algorithm only once. The resulting algorithm for the MFIE kernel has $O(N_y \log N_y) = O(N \log N)$ complexity.

3.2. Butterfly algorithm for RCS calculations

The tool for accelerating the computation in (2.15) is the butterfly algorithm proposed in [30]. Recall that a sparse Fourier transform is a computation of potentials in the form

$$u_i = \sum_j e^{2\pi \imath \vec{x}_i \cdot \vec{k}_j / M} f_j, \tag{3.4}$$

where $\{\vec{k}_j\}$ is a set of $O(M^2)$ points sampled from a smooth surface in the Fourier domain $[-M/2, M/2]^3$, $\{\vec{x}_i\} \subset [-M/2, M/2]^3$ is a set of $O(M^2)$ points sampled from another smooth surface in the spatial domain $[-M/2, M/2]^3$, and $\{f_j\}$ are the sources at $\{\vec{k}_j\}$. The butterfly algorithm proposed in [30] first constructs adaptive octrees T_X and T_K for the sets $\{\vec{x}_i\}$ and $\{\vec{k}_j\}$ respectively. The octree T_X takes $[-M/2, M/2]^3$ as the top level box. Each box is partitioned recursively into eight identical child boxes until all leaf boxes are of unit size, and only the boxes that contain points in $\{\vec{x}_i\}$ are kept. The octree T_K is constructed in the same way with $[-M/2, M/2]^3$ as the top level box and $\{\vec{k}_j\}$ as the point set. Since the sets $\{\vec{x}_i\}$ and $\{\vec{k}_j\}$ are both of order $O(M^2)$, the overall cost of this butterfly algorithm is $O(M^2 \log M)$, which is essentially linear. We refer to [30] for the detailed complexity analysis, and [5] for the error analysis.

Let us discuss now how to turn the far-field integral in (2.18) into the form of (3.4). If we set as before $j(i,q) = (i-1)Q + q$, $\vec{y}_j = \vec{p}_{i,q}$, $\vec{f}_j = \alpha_{i,q} \vec{J}(\vec{p}_{i,q})$, and $N_y = N_t Q$, then the summation in (2.18) can be written as

$$\vec{A}(\hat{r}_s) = \sum_{j=1}^{N_y} e^{2\pi \imath \hat{r}_s \cdot \vec{y}_j} \vec{f}_j. \tag{3.5}$$

In order to get the above sum in the same form as the sparse Fourier transform, we introduce a simple trick. Setting $M = 2K$, we multiply the size of our scattering object by 2; since $\vec{y}_j \in \left[-\frac{K}{2}, \frac{K}{2}\right]^3$, we have that $2\vec{y}_j \in [-K, K]^3 = \left[-\frac{M}{2}, \frac{M}{2}\right]^3$. In addition, instead of evaluating the sum on the unit sphere (where $|\hat{r}| = 1$), we will evaluate on a sphere of radius K. Now, we have $K\hat{r}_s \in [-K, K]^3 = \left[-\frac{M}{2}, \frac{M}{2}\right]^3$; thus, both point sets fall into the octree structure of the butterfly algorithm. With these modifications, we can rewrite (3.5) as

$$\sum_{j=1}^{N_y} e^{2\pi \imath (K\hat{r}_s) \cdot (2\vec{y}_j)/M} \vec{f}_j,$$

which is in the form of the sparse Fourier transform (3.4). Since $N_{\hat{r}} = O(N) = O(M^2)$ and $N_y = O(N) = O(M^2)$, the computational cost of the Butterfly algorithm is $O(M^2 \log M) = O(N \log N)$. One minor difference is that $\vec{f}_j$ has three components instead of one. Instead of running the butterfly algorithm three times (once for each component), we run the algorithm once but with vectorized operations, much like we did with the directional multilevel algorithm.

4. Numerical results

The general setup of our numerical tests is as follows. The triangular meshes we use have a refinement of 5 elements per wavelength ($h = 0.2$) to 10 elements per wavelength ($h = 0.1$); since the goal of this paper is to develop fast algorithms more than to show the convergence of the boundary element method, we feel that this is sufficient enough. For numerical integration over each triangle, we use Gaussian quadrature of order 5 for non-singular integrals and Duffy quadrature of order 5 around singularities. All of the code has been implemented serially in C++. To show the complexity of our methods, we produced the run times on a computer with a 2.13 GHz CPU benchmarked at 1.38 GFLOPS using serial LINPACK.

Table 1
Computational times and relative error 2-norms for the directional algorithm with discretization points on the surface of a sphere.

(K,ε)	N_p	T_e(sec)	T_m(sec)	ε_e	ε_m
(12, 1e−4)	4.669e+5	1.080e+2	1.250e+2	3.771e−4	6.428e−4
(24, 1e−4)	1.867e+6	4.830e+2	5.470e+2	4.027e−4	6.594e−4
(48, 1e−4)	7.471e+6	2.150e+3	2.384e+3	3.760e−4	6.991e−4
(12, 1e−6)	4.669e+5	4.140e+2	3.690e+2	1.500e−6	3.639e−6
(24, 1e−6)	1.867e+6	1.790e+3	1.601e+3	1.646e−6	3.794e−6
(48, 1e−6)	7.471e+6	7.665e+3	6.773e+3	2.227e−6	3.472e−6
(12, 1e−8)	4.669e+5	1.217e+3	1.000e+3	8.648e−9	6.371e−8
(24, 1e−8)	1.867e+6	5.148e+3	4.198e+3	1.637e−8	6.625e−8
(48, 1e−8)	7.471e+6	2.165e+4	1.742e+4	1.349e−8	5.284e−8

Table 2
Computational times and relative error 2-norms for the directional algorithm with discretization points on the surface of the NASA almond.

(K,ε)	N_p	T_e(sec)	T_m(sec)	ε_e	ε_m
(32, 1e−4)	3.164e+5	8.400e+1	8.800e+1	4.083e−4	5.291e−4
(64, 1e−4)	1.265e+6	3.860e+2	3.860e+2	3.463e−4	5.042e−4
(128, 1e−4)	5.059e+6	1.785e+3	1.745e+3	4.350e−4	4.824e−4
(32, 1e−6)	3.164e+5	3.130e+2	2.700e+2	2.386e−6	2.860e−6
(64, 1e−6)	1.265e+6	1.372e+3	1.171e+3	1.363e−6	2.814e−6
(128, 1e−6)	5.059e+6	6.082e+3	5.177e+3	1.977e−6	2.579e−6
(32, 1e−8)	3.164e+5	9.310e+2	7.460e+2	1.396e−8	3.300e−8
(64, 1e−8)	1.265e+6	3.978e+3	3.226e+3	1.361e−8	3.139e−8
(128, 1e−8)	5.059e+6	1.719e+4	1.363e+4	1.573e−8	3.099e−8

4.1. Tests of the directional multilevel algorithm

Before solving electromagnetic scattering problems by the CFIE, we first test the directional multilevel algorithm on the summations (2.15) to show the scaling properties and attainable accuracy. The tests are performed on two commonly-used examples: the sphere and the NASA almond (see Fig. 4). Here, we choose to use meshes using 5 elements per wavelength. In these tables, K is the diameter of the object in terms of wavelength, ε is the prescribed order of accuracy, T_e is the time of the EFIE summation, T_m is the time of the MFIE summation, ε_e is the relative error for the EFIE summation, and ε_m is the relative error for the MFIE summation. N_p is the total number of discretization points in the directional multilevel algorithm, including the Gaussian quadrature nodes over each triangle, the centroid of each triangle, and the center of each edge. To estimate the relative error between direct calculation and the algorithm, we compute the true values of the summations at a subset R of 200 points sampled from the total number of points. The relative error is computed via

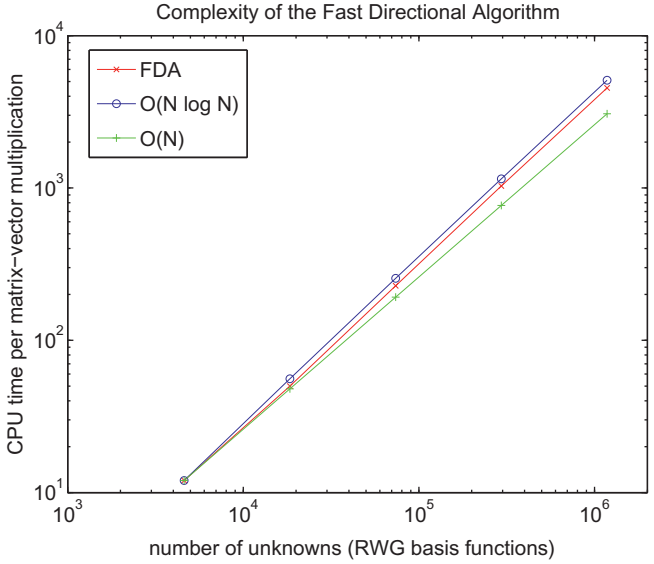
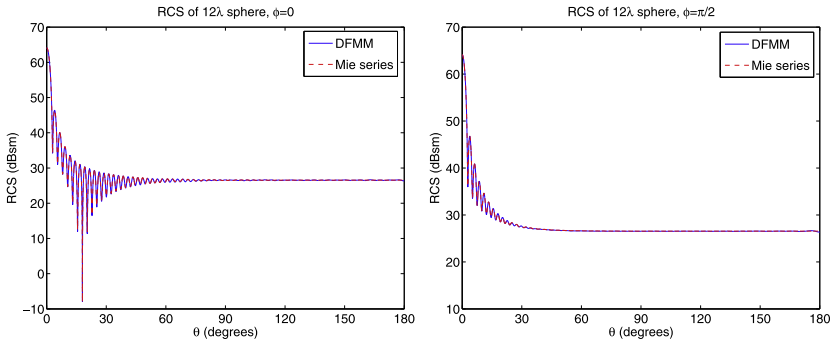


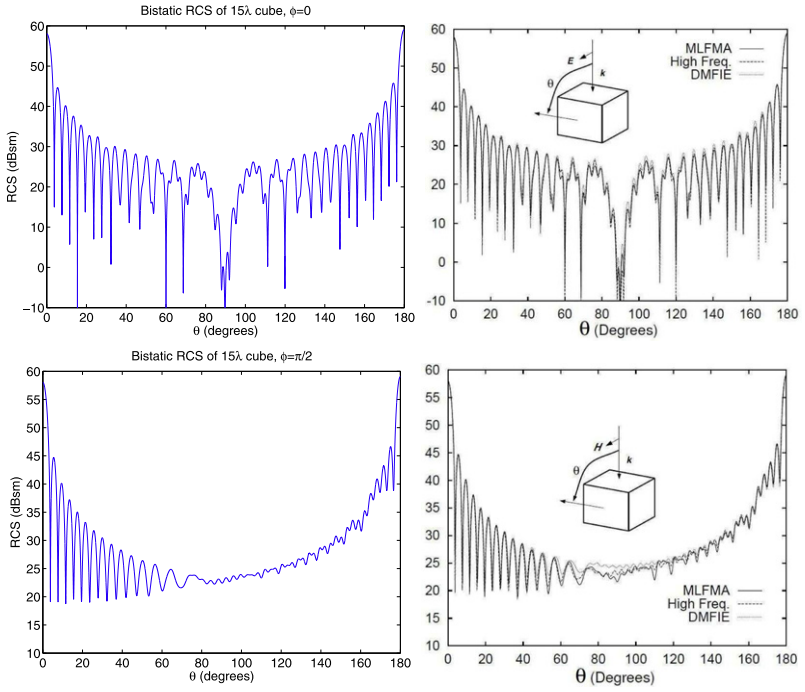
Fig. 3. CPU time per matrix–vector multiplication vs. the number of unknowns for the sphere.

Table 3
Top: Bistatic RCS of a 12λ radius sphere ($K = 24$). Bottom: Solver times and RCS relative error 2-norms for the boundary element code with the fast directional algorithm for the sphere.



(K, ε)	N	T_{solve}(sec)	n_{it}	RCS error
(6, 1e−4)	7.373e+4	3.439e+3	18	1.659e−02
(12, 1e−4)	2.949e+5	1.856e+4	23	1.136e−02
(24, 1e−4)	1.180e+6	9.602e+4	28	8.851e−03

Table 4
Top: Bistatic RCS of a 15λ cube. On the left are figures for our boundary element code using the fast directional algorithm and butterfly algorithm. On the right are figures done by Song and Chew in [26] using the MLFMA. Bottom: Solver times for the boundary element code with the fast directional algorithm for the cube.



(K, ε)	N	T_{solve}(sec)	n_{it}
(3.75, 1e−4)	1.843e+4	9.620e+2	14
(7.5, 1e−4)	7.373e+4	2.887e+3	16
(15, 1e−4)	2.949e+5	1.466e+4	18

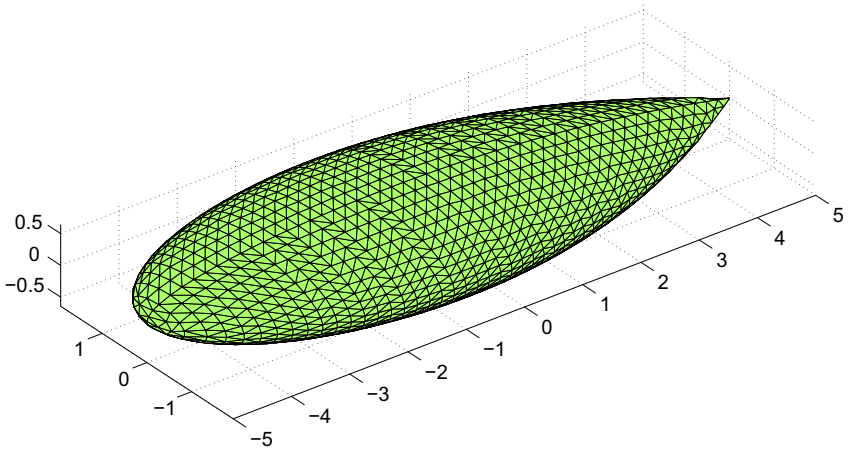


Fig. 4. The NASA almond geometry.

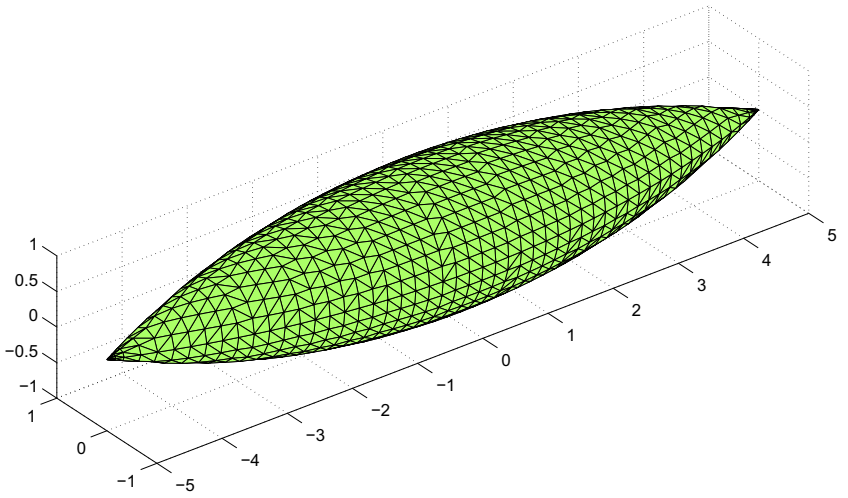


Fig. 5. The ogive geometry.

$$\sqrt{\frac{\sum_{i\in R}|\vec{u}_i-\vec{u}_i^c|^2}{\sum_{i\in R}|\vec{u}_i|^2}}, \tag{4.1}$$

where $\vec{u}_i$ is the value by direct computation and $\vec{u}_i^c$ is the value by the directional algorithm.

Table 1 shows data for the sphere, while Table 2 shows data for the NASA almond. It is observed that the computational time grows roughly by a factor of 3 or 4 when increasing the accuracy by two digits. We note that the algorithm for the MFIE summation is slightly less accurate; nevertheless, the same order of accuracy is retained. To illustrate the $O(N\log N)$ scaling of the algorithm, we plot computational time versus the number of unknowns for the sphere in Fig. 3. Here, the size of the sphere ranges from $K = 3$ to $K = 48$, and the error tolerance is 1e−4.

4.2. Electromagnetic scattering problems

For scattering examples, we choose a few test geometries that have been well established in the electromagnetics community. In these tests, we set the residual tolerance for GMRES iteration at 1e−3, while the error tolerance of the directional algorithm and butterfly algorithm are set at 1e−4. The constant α, which determines the balance between the EFIE and MFIE, is set at 0.3. Since the goal is to produce sufficiently accurate RCS plots, we choose to use refined meshes with 10 elements per wavelength. To show the accuracy of the butterfly algorithm, we compare our radar cross section computations to either analytical results or measured experiments. For analytical results, we measure the relative error of the radar cross section;

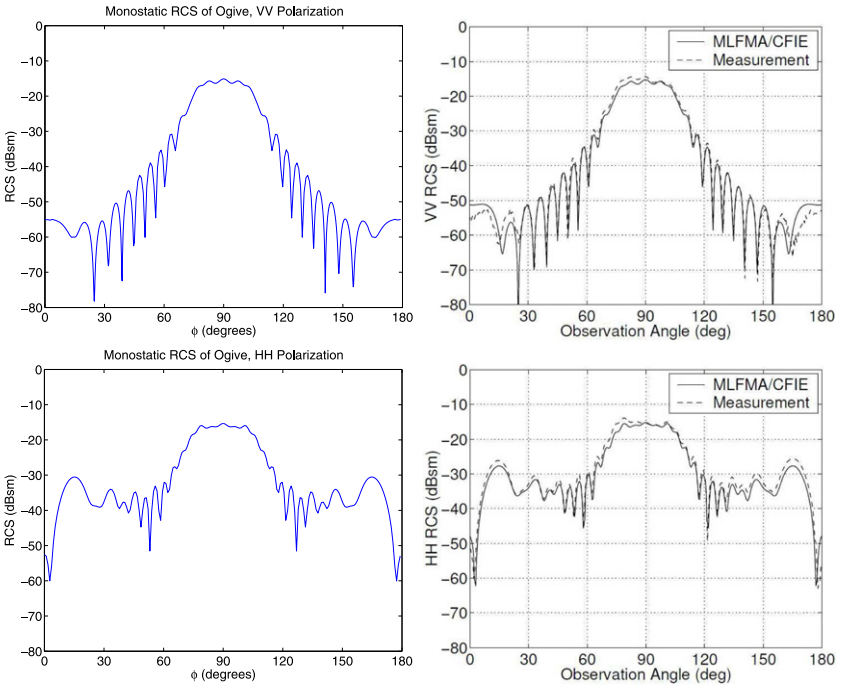
that is, if $\sigma(\hat{r})$ is the analytical solution of the RCS and $\sigma^c(\hat{r})$ is the numerical result, given the point set $\{\hat{r}_s\}_{s=1}^{N_{\hat{r}}}$ on the unit sphere we compute

$$\frac{\sqrt{\sum_{s=1}^{N_{\hat{r}}}|\sigma(\hat{r}_s)-\sigma^c(\hat{r}_s)|^2}}{\sqrt{\sum_{s=1}^{N_{\hat{r}}}|\sigma(\hat{r}_s)|^2}}. \tag{4.2}$$

We perform tests on three commonly-used surfaces. The first example is a conducting sphere, which has well-documented analytical solutions [15]. In our setup, the incident plane wave is propagating in the $-\hat{z}$ direction, with the E-field in the $+\hat{x}$ direction and H-field in the $-\hat{y}$ direction. For each simulation, the bistatic RCS was calculated along θ for $\phi = 0$ and $\phi = \frac{\pi}{2}$; here, θ and ϕ take on their usual definitions in spherical coordinates. Table 3(top) shows the bistatic RCS of a 12λ-radius sphere. The Mie series solution is compared to the boundary element method using the directional multilevel algorithm in Table 3(bottom) for diameters $K = 6$, 12, and 24. We list both the solver times and number of unknowns for each example to show the $O(N\log N)$ scaling, where N is the number of RWG basis functions. Here, T_{solve} is the total run-time of the GMRES iterative solver, and n_{it} is the number of iterations necessary for convergence. It is important to note that since we are not using any preconditioners, the condition number of our problem grows with respect to the number of unknowns, leading to an increased number of iterations.

The second example is a conducting cube with the same incoming field as the previous example. Table 4(top) shows the bistatic RCS of a cube with sides which are 15λ long; although we do not have an analytical solution for the RCS or measured data on the cube, we present two figures from the MLFMA paper by Song and Chew [26] as a comparison. The running time and iteration numbers are reported in Table 4(bottom). We note that the RCS for the $\phi = 0$ cut is not exactly symmetric, but by no fault of the butterfly algorithm; we believe this is a result of using a crude testing scheme and not employing accurate near-field techniques in the regions of corners and edges.

Table 5
Top: Monostatic RCS of the ogive at 9.0 GHz. On the left are figures for our boundary element code using the fast directional algorithm and butterfly algorithm. On the right are figures done by Gibson in [12] using the MLFMA. Bottom: Solver times for the boundary element code with the fast directional algorithm for the ogive.



(K,ε)	N	T_{solve}(sec)	n_{it}
(16, 1e−4)	3.246e+4	1.012e+3	14
(32, 1e−4)	1.297e+5	5.183e+3	16
(64, 1e−4)	5.188e+5	2.529e+4	18

Table 6
Computational times per iteration for the sphere, using the MLFMA; numbers are determined based on the tables in [12] and [8]. Here, K is the diameter in terms of wavelength and T_{iter} is time per iteration.

K	N	CPU speed	T_{iter} (sec)
5	3.072e+4	1.5 GFLOPS	85
10	1.229e+5	1.5 GFLOPS	365
120	9.633e+6	11 GFLOPS	1088

Table 7
Computational times per iteration for the sphere, using the fast directional multilevel algorithm. Here, K is the diameter in terms of wavelength and T_{iter} is time per iteration.

K	N	CPU speed	T_{iter} (sec)
5	3.175e+4	1.38 GFLOPS	80
10	1.210e+5	1.38 GFLOPS	325
120	9.618e+6	1.38 GFLOPS	8108

The third example, the ogive, is a popular benchmark target for testing electromagnetics codes [29] (see Fig. 5). Here, we take the example of the 10 inch ogive at a frequency of 9.0 GHz. Since we take our wavelength to be 1 at all times, we scale the geometry appropriately so that the object is still of the same diameter K in terms of wavelength. Table 5(top) shows the monostatic RCS about the observation angle ϕ. In these plots, VV polarization signifies the E-field oriented in the $+\hat{z}$ direction and the H-field directed in the $x-y$ plane; HH polarization signifies the E-field in the $x-y$ plane and the H-field oriented in the $-\hat{z}$ direction. In the comparison plots provided by Gibson in [12], the RCS curves are not normalized by wavelength; thus, in order to scale our RCS calculations down to their actual levels, we must compute $10\log_{10}(RCS \cdot \lambda^2)$, where λ is the free-space wavelength at the chosen frequency. Table 5(bottom) illustrates the complexity in solving larger problems; that is, for $K = 16$, 32, and 64.

4.3. Comparison to the MLFMA

To compare the performance of the fast directional algorithm with current methods, we refer to table 8.2 on page 252 in [12]. Here, the MLFMA is used to solve a variety of scattering problems; the simulations are done with Gibson's *Serenity* code on a comparable 2.0 GHz processor. Since Gibson uses an ILUT preconditioner to reduce the number of iterations, we observe only the approximate iteration time. For the 2-meter sphere at 0.75 and 1.5 GHz, the solution times (which include FMM setup and RCS calculations) are 11 min and 68 min, respectively. Given that convergence is achieved in 10 iterations or less, this corresponds to iteration times of approximately 85 s and 365 s. At these frequencies, the 2-meter sphere is 5λ and 10λ in diameter; we scale our geometry appropriately with a similar number of unknowns. For the 5λ and 10λ spheres, the fast directional algorithm completes one iteration in 80 s and 325 s, respectively. The results are presented in Tables 6 and 7.

To compare algorithms in a truly high-frequency setting, we take the example of a 120λ diameter sphere in [8]. For this example, we use one quadrature point per face, instead of the regular fifth order Gaussian quadrature. The machine used by the authors in [8] is an SGI Origin2000 utilizing 32 processors, which is benchmarked at 11 GFLOPS; therefore, the computational times presented by them should be about 8 times faster. At 13 h for 43 iterations, their computational time is 1088 s per iteration; our code completes one iteration in 8108 s. Thus, we can confidently say that our method is competitive with the MLFMA.

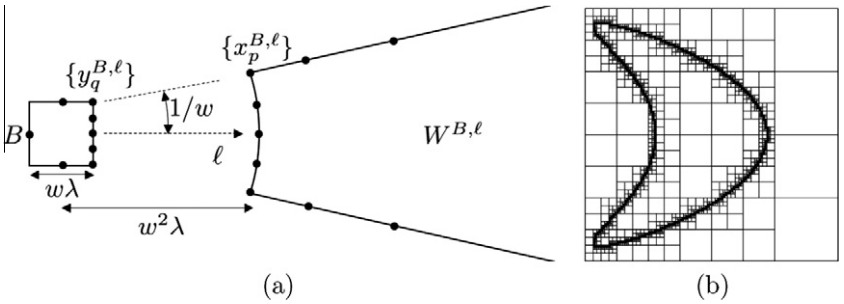


Fig. 6. 2D illustrations for components of the 3D algorithm. (a) B and $W^{B,\ell}$ follow the directional parabolic separation condition. (b) A cross-section of the octree of a kite-shaped scatterer.

5. Conclusion

In this paper, we adapted the fast directional multilevel algorithm proposed in [10] for electromagnetic scattering from perfectly conducting structures; most notably, we extended the algorithm to handle the curl operator in the magnetic field integral equation. This extension allows the use of combined field integral equations, which lead to better conditioned linear systems after discretization and a fewer number of GMRES iterations necessary for convergence. From an implementation standpoint, we have aggregated the computations such that at each iteration, only two matrix–vector products are performed (one for the EFIE and one for the MFIE). The result is an integral equation solver of $O(N\log N)$ complexity, where N is the number of unknowns; it is possible to solve 3D scattering problems which are hundreds of wavelengths long in a very reasonable time.

A few remarks can be made about the versatility of the directional algorithm here. First, it is important to note that the same directional algorithm can be used to speed up the solution of scattering problems with homogeneous dielectric bodies. Since the Green's function in an infinite homogeneous medium has the same form as the free space Green's function (the only difference is the wavenumber k), the only modifications necessary are to the integral equation and the scaling of the geometry; no changes to the directional multilevel code are needed. Secondly, although we only consider the standard CFIE formulation, the method can also be applied to a variety of other formulations, including the Calderon multiplicative preconditioner presented in [1]. Finally, we note that the directional algorithm could have be used to compute the radar cross section in the post-processing stage. That is, if $\vec{r}$ is chosen to be large enough, the scattered field can be computed using the radiation formulas (2.3) instead of the far-field approximation. However, this would require another setup phase for the extra evaluation points, and the scattered field would have to be computed at the source point locations as well; this is a result of treating the source and target points the same, as discussed in paragraph 2 of Section 3.1. For the butterfly algorithm, the source and target points are independent of each other; thus, it is more efficient to use the butterfly algorithm in our case.

Appendix A. Directional multilevel algorithm

The main idea behind the directional multilevel algorithm proposed in [10] is the directional low rank property of the Green's function. That is, consider a box B of width $w\lambda$ with $w \geqslant 1$ and a wedge $W^{B,\ell}$ as illustrated in Fig. 6 (a); if $W^{B,\ell}$ spans an angle of size $O(1/w)$ and the distance between B and $W^{B,\ell}$ is at least $O(w^2\lambda)$, we say that $W^{B,\ell}$ and B satisfy the *directional parabolic separation condition*. In [10], it was proven that for any arbitrary accuracy ε, there exists a t_ε-term separated approximation of $G(x,y)$ for any $y \in B$ and $x \in W^{B,\ell}$. In practice, we find sets $\left\{y_q^{B,\ell}\right\}_{q=1}^{t_\varepsilon}$, $\left\{x_p^{B,\ell}\right\}_{p=1}^{t_\varepsilon}$ and a matrix $D^{B,\ell} = \left(d_{qp}^{B,\ell}\right)_{1\leqslant p,q\leqslant t_\varepsilon}$ such that

$$\left| G(x,y) - \sum_{q=1}^{t_\varepsilon} G\left(x, y_q^{B,\ell}\right) \sum_{p=1}^{t_\varepsilon} d_{qp}^{B,\ell} G\left(x_p^{B,\ell}, y\right) \right| \leqslant \varepsilon \tag{A.1}$$

for $y \in B$ and $x \in W^{B,\ell}$. We call this the *directional separated approximation* of B in direction ℓ. The locations $\left\{y_q^{B,\ell}\right\}$, $\left\{x_p^{B,\ell}\right\}$, and the matrix D can be computed easily using the low-rank factorization scheme detailed in [11]; it is important to emphasize that the separation rank t_ε is independent of the size of B.

Consider the sources $\{f_i\}$ located at $\{z_i\}$ in B. The directional separated approximation allows us to compute the potentials in $W^{B,\ell}$ generated by $\{f_i\}$ with a smaller set of t_ε sources. More precisely, after applying (A.1) to each z_i and summing them up over all the sources, we have

$$\left| \sum_{z_i \in B} G(x,z_i) f_i - \sum_{q=1}^{t_\varepsilon} G\left(x, y_q^{B,\ell}\right) \left(\sum_{p=1}^{t_\varepsilon} d_{qp}^{B,\ell} \sum_{z_i \in B} G\left(x_p^{B,\ell}, z_i\right) f_i \right) \right| = O(\varepsilon). \tag{A.2}$$

This states that we can place a set of sources

$$\left\{ f_q^{B,\ell} := \sum_{p=1}^{t_\varepsilon} d_{qp}^{B,\ell} \sum_{z_i \in B} G\left(x_p^{B,\ell}, z_i\right) f_i \right\} \tag{A.3}$$

at points $\left\{y_q^{B,\ell}\right\}$ in order to reproduce the potential at $x \in W^{B,\ell}$ generated by the sources $\{f_i\}$ located at points $\{z_i\}$ in B. These sources are called the *directional equivalent sources* of B in direction ℓ and they play the role of the multipole expansions in the original FMM [13,14]. It is obvious from (A.2) that the computation of $\left\{f_q^{B,\ell}\right\}$ essentially requires only the potentials $\left\{\sum_{z_i \in B} G\left(x_p^{B,\ell}, z_i\right) f_i\right\}$ at $\left\{x_p^{B,\ell}\right\}$.

Since $G(x,y) = G(y,x)$, we can reverse the role of the source and target to get a similar result for the analogous local expansions. Consider now the sources $\{f_i\}$ located at points $\{z_i\}$ in $W^{B,\ell}$. Applying (A.1) to each z_i and summing over all the sources gives

$$\left| \sum_{z_i \in W^{B,\ell}} G(y, z_i) f_i - \sum_{p=1}^{t_\varepsilon} G\left(y, x_p^{B,\ell}\right) \left(\sum_{q=1}^{t_\varepsilon} d_{qp}^{B,\ell} \sum_{z_i \in W^{B,\ell}} G\left(y_q^{B,\ell}, z_i\right) f_i \right) \right| = O(\varepsilon).$$

This means that from the potentials $\left\{u_q^{B,\ell} := \sum_{z_i \in W^{B,\ell}} G\left(y_q^{B,\ell}, z_i\right) f_i\right\}$, we can reproduce the potential at any $y \in B$ generated by $\{f_i\}$ at $\{z_i\} \subset W^{B,\ell}$. These potentials are called the *directional check potentials* of B in direction ℓ; they play the role of the local expansions in the original FMM.

For a box B of width $w\lambda$ with $w < 1$, the wedges are replaced with a single annulus W^B with an outer radius that extends to infinity. The equivalent sources, the check potentials, the point sets $\left\{x_p^B\right\}$ and $\left\{y_q^B\right\}$, and the transform matrices can be defined similarly, but they are non-directional now (see [31] for details). For example, the equivalent sources for B can be computed by

$$\left\{ f_q^B := \sum_{p=1}^{t_\varepsilon} d_{qp}^B \sum_{z_i \in B} G\left(x_p^B, z_i\right) f_i \right\} \tag{A.4}$$

and it is clear that the computation again requires only the potentials $\sum_{z_i \in B} G\left(x_p^B, z_i\right) f_i$ at x^B.

The directional multilevel algorithm begins by constructing an octree that contains the entire scatterer (see Fig. 6(b)). Similarly to [7], we split the tree into two regimes: the low-frequency regime and high-frequency regime. A box B of width $w\lambda$ is considered to be in the low frequency regime if $w < 1$ and in the high-frequency regime if $w \geqslant 1$. In the low-frequency regime, a box B is partitioned as long as the number of points in B is greater than a fixed constant P; in the high-frequency regime, the domain is partitioned uniformly without any adaptivity, with empty boxes being discarded. The algorithm now uses the translation operators introduced in [31] in the low-frequency regime. In order to use the directional separated approximation in (A.1), we define the *far field* F^B of a box B to be the region that is at least $w^2\lambda$ away in the high-frequency regime; conversely, the *near field* N^B is defined as the union of boxes which are less than $w^2\lambda$ away. A box A is said to be in the *interaction list* of B if A is in B's far field but not in the far field of B's parent. F^B is further partitioned into a group of directional wedges $\{W^{B,\ell}\}$, where each wedge is contained in a cone with spanning angle $O(1/w)$.

A crucial point is that the wedges of the parent box and the child box are *nested*, so that we are able to construct M2M, M2L, and L2L translations of $O(1)$ complexity as in the original FMM algorithm [14]. However, these translations in the high frequency regime are now *directional*. Combining these parts gives us the following *high-frequency directional multilevel algorithm*.

1. Construct the octree. In the high-frequency regime, the boxes are partitioned uniformly. In the low-frequency regime, the boxes are partitioned adaptively until each leaf level box contains at most P points.
2. Travel up the low-frequency regime. For each box B, compute the non-directional equivalent sources following [31].
3. Travel up the high-frequency regime. For each box B and each $W^{B,\ell}$, compute $\left\{f_q^{B,\ell}\right\}$ using the directional M2M translation. We skip the boxes with width greater than $\sqrt{K}\lambda$ since their interaction lists are empty.
4. Travel down the high-frequency regime. For each box B and each $W^{B,\ell}$
   (a) Transform $\left\{f_q^{A,\ell}\right\}$ of the boxes $\{A\}$ in B's interaction list and in direction ℓ using the directional M2L translation. Next, add the result to $\left\{u_q^{B,\ell}\right\}$.
   (b) Transform $\left\{u_q^{B,\ell}\right\}$ into the directional check potentials for B's children using the directional L2L translation.
5. Travel down the low-frequency regime. For each box B:
   (a) Transform the non-directional equivalent sources of the boxes $\{A\}$ in B's interaction list using the non-directional M2L translation. Next, add the result to the non-directional check potentials.
   (b) Perform the non-directional L2L translation. If B is a leaf box, add the result to the potentials at the original points inside B. If not, then add the result to the non-directional check potentials of B's children.
6. Compute the near-field interactions. For each leaf box B, add contributions from sources in N^B directly to the potentials at points in B.

For a point set $\{z_i\}_{i=1}^{N_z}$ obtained from discretizing the surface of a scatterer in $[-K/2, K/2]^3$, $N_z = O(K^2)$. It is shown in [10,11] that the overall complexity of this algorithm is $O(N_z \log N_z) = O(K^2 \log K)$.

The directional multilevel algorithm is also efficient memory-wise; the main memory requirement comes from storing the translation operators. Since the kernel function $G(x,y)$ is known explicitly, one only needs to store the $D^{B,\ell}$ matrices for each translation operator. Due to the translation invariance of the kernel, these matrices only depend on the width of B and the wedge direction ℓ; therefore, the actual number of matrices is very small. The Green's function terms in the translation operators and near-field interactions are all computed on the fly for each iteration.

References

[1] H. Bagci, F. Andriulli, K. Cools, F. Olyslager, E. Michielssen, A Calderon multiplicative preconditioner for the combined field integral equation, IEEE Trans. Antennas Propag. 57 (10) (2009) 3387–3392.

[2] M. Bebendorf, S. Kunis, Recompression techniques for adaptive cross approximation, J. Integral Equat. Appl. 21 (3) (2009) 331–357.
[3] E. Bleszynski, M. Bleszynski, T. Jaroszewicz, AIM: adaptive integral method for solving large-scale electromagnetic scattering and radiation problems, Radio Sci. 31 (5) (1996) 1225–1251.
[4] O.P. Bruno, L.A. Kunyansky, A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, tests, and applications, J. Comput. Phys. 169 (1) (2001) 80–110.
[5] E. Candes, L. Demanet, L. Ying, A fast butterfly algorithm for the computation of Fourier integral operators, Multiscale Model. Simul. 7 (4) (2009) 1678–1694.
[6] F.X. Canning, K. Rogovin, Fast direct solution of standard moment-method matrices, IEEE Trans. Antennas Propag. 40 (3) (1998) 15–26.
[7] H. Cheng, W.Y. Crutchfield, Z. Gimbutas, L.F. Greengard, J.F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, J. Zhao, A wideband fast multipole method for the Helmholtz equation in three dimensions, J. Comput. Phys. 216 (1) (2006) 300–325.
[8] W. Chew, J. Jin, E. Michielssen, J. Song, Fast and Efficient Algorithms in Computational Electromagnetics, Artech House, 2001.
[9] M. Duffy, Quadrature over a pyramid or cube of integrands with a singularity at a vertex, SIAM J. Numer. Anal. 19 (1982) 1260–1262.
[10] B. Engquist, L. Ying, Fast directional multilevel algorithms for oscillatory kernels, SIAM J. Sci. Comput. 29 (4) (2008) 1710–1737.
[11] B. Engquist, L. Ying, A fast directional algorithm for high frequency acoustic scattering in two dimensions, Commun. Math. Sci. 7 (2) (2009) 327–345.
[12] W. Gibson, The Method of Moments in Electromagnetics, Chapman and Hall, 2007.
[13] L. Greengard, The rapid evaluation of potential fields in particle systems, ACM Distinguished Dissertations, MIT Press, Cambridge, MA, 1988.
[14] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (2) (1987) 25–348.
[15] R. Harrington, Time-harmonic Electromagnetic Fields, McGraw-Hill, 1961.
[16] A. Heldring, J.M. Tamayo, J.M. Rius, Fast direct solution of the combined field integral equation, URSI Int. Symp. Electromagn. Theory (2010) 524–527.
[17] S. Kapur, D. Long, IES^3: efficient electrostatic and electromagnetic simulation, IEEE Comput. Sci. Eng. 5 (4) (1998) 60–67.
[18] P.G. Martinsson, V. Rokhlin, A fast direct solver for scattering problems involving elongated structures, J. Comput. Phys. 221 (1) (2007) 288–302.
[19] E. Michielssen, A. Boag, W.C. Chew, Scattering from elongated objects: direct solution in O(N log^2 N) operations, IEE Proc. Microw. Antennas Propag. 143 (4) (1996) 277–283.
[20] A. Peterson, S. Ray, R. Mittra, Computational Methods for Electromagnetics, Wiley-IEEE Press, 2001.
[21] J. Phillips, J. White, A precorrected-FFT method for electrostatic analysis of complicated 3-D structures, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 16 (10) (1997) 1059–1072.
[22] S. Rao, D. Wilton, A. Glisson, Electromagnetic scattering by surfaces of arbitrary shape, IEEE Trans. Antennas Propag. 30 (3) (1982) 409–418.
[23] V. Rokhlin, Rapid solution of integral equations of scattering theory in two dimensions, J. Comput. Phys. 86 (2) (1990) 414–439.
[24] S.M. Seo, J.F. Lee, A single-level low rank IE-QR algorithm for PEC scattering problems using EFIE formulation, IEEE Trans. Antennas Propag. 52 (8) (2004) 2141–2146.
[25] S.M. Seo, J.F. Lee, A fast IE-FFT algorithm for solving PEC scattering problems, IEEE Trans. Magn. 41 (5) (2005) 1476–1479.
[26] J. Song, W. Chew, Multilevel fast multipole algorithm for solving combined field integral equations of electromagnetic scattering, Microwave Optical Technol. Lett. 10 (1) (1995) 14–19.
[27] J. Song, C. Lu, W. Chew, Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects, IEEE Trans. Antennas Propag. 45 (10) (1997) 1488–1493.
[28] S. Wandzura, H. Xiao, Symmetric quadrature rules on a triangle, Comput. Math. Appl. 45 (12) (2003) 1829–1840.
[29] A. Woo, H. Wang, M. Schuh, M. Sanders, Benchmark radar targets for the validation of computational electromagnetics programs, IEEE Antennas Propag. Mag. 35 (1) (1993) 84–89.
[30] L. Ying, Sparse Fourier transform via butterfly algorithm, SIAM J. Sci. Comput. 31 (3) (2009) 1678–1694.
[31] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys. 196 (2) (2004) 591–626.
[32] A. Zhu, R. Adams, F. Canning, Modified simply sparse method for electromagnetic scattering by PEC, IEEE Antennas Propag. Soc. Int. Symp. 4A (2005) 427–430.