An $\mathcal{H}^2$-Matrix-Based Integral-Equation Solver of Reduced Complexity and Controlled Accuracy for Solving Electrodynamic Problems

Wenwen Chai and Dan Jiao, *Senior Member, IEEE*

***Abstract*—Using an $\mathcal{H}^2$ matrix as the mathematical framework, we compactly represent a dense system matrix by a reduced set of parameters, thus enabling a significant reduction in computational complexity. The error bound of the $\mathcal{H}^2$-matrix-based representation of an electrodynamic problem was derived. We show that exponential convergence with respect to the number of interpolation points can be achieved irrespective of the electric size. In addition, we show that a direct application of $\mathcal{H}^2$-matrix-based techniques to electrodynamic problems would result in a complexity greater than $O(N)$, with N being the matrix size, due to the need of increasing the rank when ascending an inverted tree in order to keep a constant order of accuracy. A rank function was hence developed to maintain the same order of accuracy in a wide range of electric sizes without compromising computational complexity. With this rank function, we demonstrate that given a range of electric sizes which lead to a range of N, the dense system of $O(N^2)$ parameters can be compactly stored in $O(N)$ units, and the dense matrix-vector multiplication can be performed in $O(N)$ operations. Moreover, the same order of accuracy can be kept across this range. The method is kernel independent, and hence is suitable for any integral-equation-based formulation. In addition, it is applicable to arbitrary structures. Numerical experiments from small electric sizes to 64 wavelengths have demonstrated the performance of the proposed method.**

***Index Terms*—$\mathcal{H}^2$ matrix, electromagnetic analysis, fast solvers, integral-equation-based methods, low complexity.**

I. Introduction

THE design of advanced engineering systems generally results in numerical problems of very large scale, requiring billions of parameters to describe them accurately. For example, the design of a global power delivery system that involves voltage regulator module, motherboard, package, and chip. As another example, the design of electric machines co-optimizing the geometries of stators and rotors, winding patterns, and thermal dissipation. In order to make a real impact in today's and tomorrow's design of advanced engineering systems, computational electromagnetic methods have to scale favorably with the problem size. Therefore, there exists a continued demand of reducing the complexity of computational electromagnetic methods.

Since the advent of computational electromagnetics (CEM) in the 1960s, numerous fast algorithms have been developed. They can be categorized into two classes: integral equation (IE) based solvers and partial differential equation (PDE) based ones. The PDE-based techniques are particularly suited for handling the complicated inhomogeneities and irregular geometries. The IE-based methods, generally, are more efficient for open-region problems involving impenetrable or homogeneous objects since they reduce the solution domain by one dimension, and they satisfy the radiation condition through the Green's function. For open-region problems involving inhomogeneous objects, the combination of IE- and PDE-based solvers is often more efficient.

IE-based methods generally lead to dense systems of linear equations. When a direct method is used, the operation count is proportional to $O(N^3)$ and the memory requirement is proportional to $O(N^2)$, where N is the dimension of the matrix. When an iterative solver is used, the memory requirement remains the same, and the computing time is proportional to $O(N_{\rm it}N^2)$, where $N_{\rm it}$ denotes the total number of iterations required to reach convergence. In recent years, fast multipole based methods (FMM) [1], FFT-based methods [2]–[5], and fast low-rank compression methods [6]–[9] have been developed that dramatically reduce the memory requirement of iterative solvers to $O(N\log N)$, and the CPU time to $O(N\log N)$ for electrodynamic problems. Fast direct solvers have also been developed. Most recent work can be seen in [10], [11]. All these methods represent impressive improvements as compared with conventional $O(N^3)$ or $O(N^2)$ techniques.

In this paper, we study the feasibility of further reducing the complexity of the IE-based computation for electrodynamic problems in both memory and CPU consumption. Our solution hinges on the observation that the matrices resulting from an IE-based method, although dense, can be thought of as "data-sparse," i.e., they can be specified by few parameters. This can be accomplished by remodeling the problem subject to underlying hierarchical dependencies such that all interactions can be constructed from a reduced set of parameters. There exists a general mathematical framework called the "Hierarchical ($\mathcal{H}$) Matrix" framework [12]–[14], which enables a highly compact representation and efficient numerical computation of the dense matrices. Storage requirements and matrix-vector multiplications using $\mathcal{H}$ matrices have been shown to be of complexity $O(N\log N)$. The hierarchical matrix structure has

Manuscript received November 20, 2008; revised February 19, 2009. First published July 28, 2009; current version published October 07, 2009. This work was supported by the National Science Foundation (NSF) under Awards 0747578 and 0702567.

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: djiao@purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAP.2009.2028665

been used in [15], [16] to solve electro- and magneto-static problems. Authors in [17]–[19] later introduced $\mathcal{H}^2$ matrices, which are a specialized subclass of hierarchical matrices. It was shown that the storage requirements and matrix-vector products are of complexity $O(N)$. The nested structure is the key difference between general $\mathcal{H}$ matrices and $\mathcal{H}^2$ matrices, since it permits an efficient reuse of information across the entire cluster tree. As a general mathematical framework, the $\mathcal{H}$- and $\mathcal{H}^2$-matrix allows for the acceleration of both iterative and direct IE solvers, which cannot be easily achieved in the framework of other fast IE solvers. In addition, the methods are kernel independent, and hence suitable for any IE-based formulation.

The complexity analysis given in the literature of $\mathcal{H}$- and $\mathcal{H}^2$-matrices was all conducted based on kernel functions that do not change with frequency. It is not clear whether the same complexity can be obtained for electrodynamic problems. In [20], the original authors of $\mathcal{H}$ matrices applied the $\mathcal{H}$-matrix-based techniques to 2-D high-frequency Helmholtz problems. A dense Galerkin matrix arising from the IE-based analysis was represented by a sum of an $\mathcal{H}$ matrix and an $\mathcal{H}^2$ matrix. The error was discussed in this paper. However, the η-admissibility condition was not considered ([20, Eq. (2.11)]). In addition, the cost of a matrix-vector multiply was reported to be almost linear. Linear complexity has not been reported yet. In [21], the feasibility of using $\mathcal{H}$ matrices was studied to reduce the complexity of IE-based solutions of electrodynamic problems. In [22], an $\mathcal{H}^2$-matrix-based integral-equation solver was reported for large-scale full-wave modeling of 3-D circuits, in which an $\mathcal{H}^2$ matrix was directly constructed to represent the dense system matrix resulting from an IE-based analysis of 3-D circuit problems without any compression cost. The error of $\mathcal{H}^2$-matrix-based representation of an *electrodynamic* problem and its impact on computational complexity still needs to be addressed.

The main contribution of this paper is three fold. First, the error bound of the $\mathcal{H}^2$-matrix-based representation of electrodynamic problems was derived. It was shown that exponential convergence with respect to the number of interpolation points can be achieved irrespective of the electric size.

Second, we show that a direct application of $\mathcal{H}^2$-matrix-based techniques to electrodynamic problems would result in a complexity greater than $O(N)$ in both CPU time and memory consumption. This is because although the storage requirements and matrix-vector products of $\mathcal{H}^2$ matrices are of complexity $O(N)$, the solution accuracy cannot be kept to the same order for electrodynamic problems when problem size increases or frequency increases. To keep the accuracy to the same order, one approach is to increase the number of interpolation points when ascending an inverted tree. Although in this case, the complexity is not better than that offered by FMM-based techniques, $\mathcal{H}^2$-matrix based methods do have their unique merits as mentioned at the end of the fourth paragraph in this section.

Last and more important, we show that the cost of $\mathcal{H}^2$-matrix-based solutions of electrodynamic problems can be reduced to linear in a wide range of electric sizes by developing a rank function. This rank function can be used to systematically determine the rank based on tree level. Meanwhile, this rank function has constant coefficients that do not change with N. A detailed complexity analysis based on this rank function revealed linear cost in both memory consumption and matrix-vector multiplication. Numerical experiments from small electric sizes to tens of wavelengths have demonstrated a constant error together with linear cost.

The remainder of this paper is organized as follows. In Section II, an IE formulation is presented. In Section III, the proposed $\mathcal{H}^2$-matrix-based IE solver is detailed, which includes an $\mathcal{H}^2$-matrix representation and its error bound, the cluster tree and block cluster tree construction, rank function, and complexity analysis based on the rank function. In Section IV, numerical results are given to demonstrate the accuracy and efficiency of the proposed IE solver. Section V relates to our conclusions.

II. Integral-Equation Based Formulation

Consider a 3-D arbitrarily shaped conducting object immersed in a medium characterized by permittivity ϵ and permeability μ. The object is illuminated by an incident wave $\vec{E}_i$ that induces current $\vec{J}_s$ on the conducting surface. The current satisfies the following electric-field integral equation, as shown in (1) at the bottom of the page, in which Green's function $g(\overrightarrow{r}, \overrightarrow{r}') = e^{-j\kappa|\overrightarrow{r}-\overrightarrow{r}'|}/4\pi|\overrightarrow{r}-\overrightarrow{r}'|$, ω is angular frequency, and κ is the wave number which is $\omega\sqrt{\mu\varepsilon}$. The subscript "tan" denotes the component that is tangential to the conducting surface S. By expanding the unknown surface current density $\vec{J}_s$ using RWG basis functions [23], and applying Galerkin's method to (1), we obtain (2), as shown at the bottom of the page, in which $\vec{J}_m$ ($\vec{J}_n$) are basis functions,

$$\vec{E}_i|_{\tan} = \int\int_S \left[j\omega\mu\vec{J}_s(\overrightarrow{r}')g(\overrightarrow{r}, \overrightarrow{r}') - \frac{j}{\omega\varepsilon}(\nabla' \cdot \vec{J}_s(\overrightarrow{r}'))\nabla' g(\overrightarrow{r}, \overrightarrow{r}') \right]_{\tan} ds' \tag{1}$$

$$\int\int_{S_m} \vec{J}_m(\overrightarrow{r}) \cdot \vec{E}_i(\overrightarrow{r}) ds = \sum_{n=1}^{N} I_n \int\int_{S_m} ds \int\int_{S_n} ds' \left[j\omega\mu\vec{J}_m(\overrightarrow{r}) \cdot \vec{J}_n(\overrightarrow{r}') - \frac{j}{\omega\varepsilon}(\nabla \cdot \vec{J}_m(\overrightarrow{r}))(\nabla' \cdot \vec{J}_n(\overrightarrow{r}')) \right] g(\overrightarrow{r}, \overrightarrow{r}') \tag{2}$$

and N is the total number of basis functions. Equation (2) can be written in a matrix equation format

$$\mathbf{G}I = V \tag{3}$$

where

$$\begin{aligned}\mathbf{G}_{mn} &= \iint_{S_m} ds \iint_{S_n} ds' \Big[j\omega\mu \vec{f}_m(\vec{r}) \cdot \vec{f}_n(\vec{r}') \\ &\quad - \frac{j}{\omega\varepsilon} (\nabla \cdot \vec{f}_m(\vec{r}))(\nabla' \cdot \vec{f}_n(\vec{r}')) \Big] g(\vec{r}, \vec{r}') \\ V_m &= \iint_{S_m} \vec{f}_m(\vec{r}) \cdot \vec{E}_i(\vec{r}) ds. \end{aligned} \tag{4}$$

A straightforward approach to solving (3) can be very expensive, since matrix $\mathbf{G}$ is dense in the sense that all entries are nonzero. Our approach is to approximate $\mathbf{G}$ by a matrix (with error well controlled), which can be stored in a data-sparse format, i.e., $\mathbf{G}$ can be specified by few parameters, from which a significant reduction in complexity can be achieved.

III. Proposed $\mathcal{H}^2$-Matrix Based Fast IE Solver

In the proposed IE solver, first, we approximate $\mathbf{G}$ by an $\mathcal{H}^2$ matrix with accuracy well controlled, the detail of which is illustrated in Section III-A. Second, we explore the use of a block cluster tree to efficiently capture the nested hierarchical dependencies present in the $\mathcal{H}^2$-matrix-based representation of $\mathbf{G}$ (Section III-B). Then, we introduce a rank function to control the accuracy without compromising computational complexity (Section III-C). Next, based on the rank function, we identify a reduced set of parameters of $O(N)$ to compactly represent $\mathbf{G}$ (Section III-D), which paves the way to the fast matrix-vector multiplication described in Section III-E.

A. $\mathcal{H}^2$-Matrix Representation and Its Error Bound

Denoting the index set of the basis functions used in the discretization of (1) by $\mathcal{I} := \{1, 2, \ldots, N\}$. We fix two subsets t and s of $\mathcal{I}$ and define the corresponding domains Ω_t and Ω_s as the union of the supports of the basis functions

$$\Omega_t := \bigcup_{i \in t} \mathrm{supp}(\vec{f}_i), \qquad \Omega_s := \bigcup_{i \in s} \mathrm{supp}(\vec{f}_i) \tag{5}$$

in which "supp" denotes the domain that is occupied by the basis function. If t and s are far away from each other (the criterion will be established soon), the original kernel function $g(\vec{r}, \vec{r}')$ in (1) can be replaced by a degenerate approximation

$$\tilde{g}^{t,s}(\vec{r}, \vec{r}') = \sum_{v \in K^t} \sum_{\mu \in K^s} g(\xi_v^t, \xi_\mu^s) L_v^t(\vec{r}) L_\mu^s(\vec{r}') \tag{6}$$

where $K := \{v \in \mathrm{N}^d : v_i \le p \text{ for all } i \in \{1, \ldots, d\}\} = \{1, \ldots, p\}^d$, $d = 1, 2, 3$, for 1-, 2-, and 3-D problems, respectively; p is the number of interpolation points; $(\xi_v^t)_{v \in K^t}$ is a family of interpolation points in t; $(\xi_\mu^s)_{\mu \in K^s}$ is a family of interpolation points in s; and $(L_v^t)_{v \in K^t}$ and $(L_\mu^s)_{\mu \in K^s}$ are the corresponding Lagrange polynomials satisfying $L_v(\xi_\tau) = \delta_{v,\tau}$ for all $v, \tau \in K^t$, and $L_\mu(\xi_\tau) = \delta_{\mu,\tau}$ for all $\mu, \tau \in K^s$.

The advantage of the degenerate approximation is twofold. First, the double integral in (4) is separated in two single integrals

$$\begin{aligned}&\tilde{\mathbf{G}}_{mn}^{t,s} \\ &:= \sum_{v \in K^t} \sum_{\mu \in K^s} j\omega\mu g(\xi_v^t, \xi_\mu^s) \iint_{S_m} \vec{f}_m(\vec{r}) L_v^t(\vec{r}) ds \\ &\quad \cdot \iint_{S_n} \vec{f}_n(\vec{r}') L_\mu^s(\vec{r}') ds' - \sum_{v \in K^t} \sum_{\mu \in K^s} \frac{j}{\omega\varepsilon} g(\xi_v^t, \xi_\mu^s) \\ &\quad \times \iint_{S_m} (\nabla \cdot \vec{f}_m(\vec{r})) L_v^t(\vec{r}) ds \iint_{S_n} (\nabla' \cdot \vec{f}_n(\vec{r}')) L_\mu^s(\vec{r}') ds'. \\ &\quad \text{for } m \in t,\ n \in s,\ v \in K^t,\ \mu \in K^s. \end{aligned} \tag{7}$$

Second, the submatrix $\tilde{\mathbf{G}}^{t,s}$ can be represented in a factorized form

$$\begin{aligned}\tilde{\mathbf{G}}^{t,s} &:= \mathbf{V}^t \mathbf{S}^{t,s} \mathbf{V}^{s^{\mathrm{T}}},\ \mathbf{V}^t \in \mathbb{C}^{t \times 2K^t}, \\ &\qquad \mathbf{S}^{t,s} \in \mathbb{C}^{2K^t \times 2K^s},\ \mathbf{V}^s \in \mathbb{C}^{s \times 2K^s} \end{aligned} \tag{8}$$

where

$$\begin{aligned}&\mathbf{V}^t = [\mathbf{V}_1^t \quad \mathbf{V}_2^t],\ \mathbf{V}^s = [\mathbf{V}_1^s \quad \mathbf{V}_2^s],\ \mathbf{S}^{t,s} = \begin{bmatrix} \mathbf{S}_1^{t,s} & 0 \\ 0 & \mathbf{S}_2^{t,s} \end{bmatrix} \\ &\mathbf{V}_1^t, \mathbf{V}_2^t \in \mathbb{C}^{t \times K^t}, \mathbf{V}_1^s, \mathbf{V}_2^s \in \mathbb{C}^{s \times K^s}, \mathbf{S}_1^{t,s}, \mathbf{S}_2^{t,s} \in \mathbb{C}^{K^t \times K^s} \end{aligned} \tag{9}$$

and

$$\begin{aligned}\mathbf{V}_{1mv}^t &= \iint_{S_m} \vec{f}_m(\vec{r}) L_v^t(\vec{r}) ds \\ \mathbf{V}_{2mv}^t &= \iint_{S_m} (\nabla \cdot \vec{f}_m(\vec{r})) L_v^t(\vec{r}) ds \\ \mathbf{V}_{1n\mu}^s &= \iint_{S_n} \vec{f}_n(\vec{r}') L_\mu^s(\vec{r}') ds' \\ \mathbf{V}_{2n\mu}^s &= \iint_{S_n} (\nabla' \cdot \vec{f}_n(\vec{r}')) L_\mu^s(\vec{r}') ds' \\ \mathbf{S}_{1v\mu}^{t,s} &= j\omega\mu g(\xi_v^t, \xi_\mu^s),\ \ \mathbf{S}_{2v\mu}^{t,s} = \frac{-j}{\omega\varepsilon} g(\xi_v^t, \xi_\mu^s) \\ &\qquad m \in t,\ n \in s,\ v \in K^t,\ \mu \in K^s. \end{aligned} \tag{10}$$

Clearly, the rank of the matrix $\tilde{\mathbf{G}}^{t,s}$ is at most $2\#K^t$ or $2\#K^s$ regardless of the cardinality of t and s (assuming $\#t(\#s)$ is larger than $2\#K^t(2\#K^s)$). For example, if $p = 2$ and $d = 2$ are used in both t and s, the rank of $\tilde{\mathbf{G}}^{t,s} = 2 \times p^d$ is 8 irrespective of the cardinality of t and s.

To estimate the error bound of the low-rank approximation in (8), we define a strong admissibility condition [24] as

$$\begin{aligned}&(t,\ s) \text{ are admissible} \\ &:= \begin{cases} \text{True}, & \text{if } \max\{\mathrm{diam}(\Omega_t), \mathrm{diam}(\Omega_s)\} \le \eta \mathrm{dist}(\Omega_t, \Omega_s) \\ \text{False}, & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

in which $\mathrm{diam}(\cdot)$ is the Euclidean diameter of a set, $\mathrm{dist}(\cdot, \cdot)$ is the Euclidean distance of two sets, and η is a parameter that can be used to control the accuracy of the low-rank approximation. This condition ensures that we are dealing with a region

where the Green's function is expected to be smooth or at least separable.

From [28, p. 328], let $g \in C^{\infty}(Q_t \times Q_s)$ where Q_t and Q_s are axis-parallel bounding boxes such that $\Omega_t \in Q_t$ and $\Omega_t \in Q_s$ hold, such that there are positive real constants C_g and γ_g satisfying

$$||\partial_j^n g||_{\infty,Q_t\times Q_s} \le C_g \gamma_g^n n! \tag{12}$$

the error of the interpolated kernel function $\tilde{g}$ is bounded by

$$||g(r,r') - \tilde{g}^{(t,s)}(r,r')||_{\infty,Q_{t\times}Q_s} \le 8e(2d)(\Lambda_p)^{2d} p C_g \cdot[1+\gamma_g \mathrm{diam}(Q_t \times Q_s)] \left[1+\frac{2}{\gamma_g \mathrm{diam}(Q_t\times Q_s)\}}\right]^{-p} \tag{13}$$

where Λ_p is a constant related to p and the interpolation scheme, and as shown in (14) at the bottom of the page.

The parameters C_g and γ_g in (12) are dependent on the kernel function g. For an electrodynamic kernel, we derive them as follows.

From $g(\vec{r},\vec{r}') = e^{-j\kappa R}/4\pi R$, where $R = |\vec{r}-\vec{r}'| = \sqrt{(x-x')^2+(y-y')^2+(z-z')^2}$, we have

$$|\partial_j R| = \left|\frac{j-j'}{R}\right| \le 1, \quad j = x,y,z,x',y',z'. \tag{15}$$

Therefore,

$$\begin{aligned}||\partial_j^n g||_{\infty,Q_t\times Q_s} &\le \frac{1}{4\pi}\left[\frac{\kappa^n}{R}+\frac{n\kappa^{n-1}}{R^2}+\frac{n(n-1)\kappa^{n-2}}{R^3}\right.\\ &\quad \left.+\cdots+\frac{n(n-1)2\kappa}{R^n}+\frac{n!}{R^{n+1}}\right]\end{aligned} \tag{16}$$

where κ is the wave number.

To identify C_g and γ_g, we did the following derivation:

$$\begin{aligned}&||\partial_j^n g||_{\infty,Q_t\times Q_s}\\ &\le \frac{1}{4\pi}\left[\frac{\kappa^n}{R}+\frac{n\kappa^{n-1}}{R^2}+\frac{n(n-1)\kappa^{n-2}}{R^3}\right.\\ &\quad \left.+\cdots+\frac{n(n-1)2\kappa}{R^n}+\frac{n!}{R^{n+1}}\right]\\ &= \frac{1}{4\pi R^{n+1}}[(\kappa R)^n + n(\kappa R)^{n-1}\\ &\quad + n(n-1)(\kappa R)^{n-2}+\cdots+n!]\\ &= n!\frac{1}{4\pi R^{n+1}}\left[\frac{(\kappa R)^n}{n!}+\frac{n(\kappa R)^{n-1}}{n!}\right.\\ &\quad \left.+\frac{n(n-1)(\kappa R)^{n-2}}{n!}+\cdots+\frac{n!}{n!}\right]\\ &\le n!\frac{1}{4\pi R^{n+1}}\left[\frac{(\kappa R)^n}{1}+\frac{n(\kappa R)^{n-1}}{1!}\right.\\ &\quad \left.+\frac{n(n-1)(\kappa R)^{n-2}}{2!}+\cdots+\frac{n!}{n!}\right]\\ &= n!\frac{1}{4\pi R^{n+1}}[\kappa R+1]^n\\ &= n!\frac{1}{4\pi R}\left[\frac{\kappa R+1}{R}\right]^n = n!\frac{1}{4\pi R}\left[\kappa+\frac{1}{R}\right]^n\\ &\le \frac{1}{4\pi \mathrm{dist}(Q_t,Q_s)}\left[\kappa+\frac{1}{\mathrm{dist}(Q_t,Q_s)}\right]^n n!.\end{aligned} \tag{17}$$

Here we kept both terms, κ and $1/\mathrm{dist}(Q_t,Q_s)$, to facilitate the error analysis for both static and dynamic problems. Comparing (17) to (12), we obtain

$$C_g = \frac{1}{4\pi \mathrm{dist}(Q_t,Q_s)}, \quad \gamma_g = \kappa + \frac{1}{\mathrm{dist}(Q_t,Q_s)}. \tag{18}$$

Hence, from (13), we have

$$\begin{aligned}&||g(\vec{r},\vec{r}') - \tilde{g}^{(t,s)}(\vec{r},\vec{r}')||_{\infty,Q_{t\times}Q_s}\\ &\le 8e(2d)(\Lambda_p)^{2d} p \frac{1}{4\pi \mathrm{dist}(Q_t,Q_s)}\\ &\quad \cdot\left[1+\left(\kappa+\frac{1}{\mathrm{dist}(Q_t,Q_s)}\right)\mathrm{diam}(Q_t\times Q_s)\right]\\ &\quad \cdot\left[1+\frac{2}{\left(\kappa+\frac{1}{\mathrm{dist}(Q_t,Q_s)}\right)\mathrm{diam}(Q_t\times Q_s)}\right]^{-p}.\end{aligned} \tag{19}$$

If the admissibility condition given in (11) is satisfied, i.e.,

$$\max\{\mathrm{diam}(Q_t),\mathrm{diam}(Q_s)\} \le \eta \mathrm{dist}(Q_t,Q_s).$$

From (19) and (14), we obtain

$$\begin{aligned}&||g(\vec{r},\vec{r}') - \tilde{g}^{(t,s)}(\vec{r},\vec{r}')||_{\infty,Q_{t\times}Q_s}\\ &\le \frac{4ed}{\pi}(\Lambda_p)^{2d} p \frac{1}{\mathrm{dist}(Q_t,Q_s)}\\ &\quad \cdot\left[1+\sqrt{2}\kappa\eta \mathrm{dist}(Q_t,Q_s)+\sqrt{2}\eta\right]\\ &\quad \cdot\left[1+\frac{2}{\sqrt{2}\kappa\eta \mathrm{dist}(Q_t,Q_s)+\sqrt{2}\eta}\right]^{-p}.\end{aligned} \tag{20}$$

Clearly, exponential convergence with respect to p can be obtained irrespective of the electric size $\kappa\eta \mathrm{dist}(Q_t,Q_s)$ and the choice of η. In addition, given a required level of accuracy, when the electric size increases, the error of the $\mathcal{H}^2$-matrix

$$\begin{aligned}\mathrm{diam}(Q_t\times Q_s) &= \max\{||b_1-b_2||_2 : b_1,b_2 \in Q_t\times Q_s\} = \max\{||(t_1,s_1)-(t_2,s_2)||_2 : (t_1,t_2)\in Q_t,(s_1,s_2)\in Q_s\}\\ &\le \sqrt{\max\{||t_1-t_2||_2^2+||s_1-s_2||_2^2 : (t_1,t_2)\in Q_t,(s_1,s_2)\in Q_s\}}\\ &\le \sqrt{\mathrm{diam}(Q_t)^2+\mathrm{diam}(Q_s)^2} \le \sqrt{2}\max\{\mathrm{diam}(Q_t),\mathrm{diam}(Q_s)\}.\end{aligned} \tag{14}$$

approximation can be controlled to the same level either by decreasing η to maintain a constant $\kappa\eta\mathrm{dist}(Q_t, Q_s)$, or by adaptively increasing the number of interpolation points p, or by the combination of both. Certainly, all these accuracy control approaches could render the computational complexity increasing with electric size, like what has been observed in FMM-based solutions of electrodynamic problems. Since the number of multipoles needs to be increased when one ascends an inverted tree, the complexity of FMM-based solutions of electrodynamic problems is shown to be $O(N \log N)$ in contrast to the $O(N)$ complexity in static applications. The increased complexity with electric size is also true for a direct application of $\mathcal{H}^2$-matrix-based techniques. In Section III.C, we will show an approach that can be used to control the accuracy of $\mathcal{H}^2$-matrix-based solutions to the same order in a range of electric sizes without compromising computational complexity.

The low rank approximation of $\mathbf{G}$ in (8) forms an $\mathcal{H}^2$-matrix-based representation of $\mathbf{G}$ [25] if the same space of polynomials are used across t and s. It enables an efficient computation of matrix-vector multiplication. Also, the $\mathcal{H}^2$-matrix approximation is kernel independent. In addition, it is worth mentioning that the $\mathcal{H}^2$-matrix representation in (8) is constructed directly without any compression cost.

In addition to the $\mathcal{H}^2$-matrix-based low-rank approximation, we explore the use of a cluster tree and a block cluster tree [26], [27] to efficiently capture a nested hierarchical dependence present in the structure of matrix $\mathbf{G}$, which is described in next section.

B. Cluster Tree and Block Cluster Tree Construction

For the index set of the basis functions $\mathcal{I} := \{1, 2, \ldots, N\}$, we construct a cluster tree $T_{\mathcal{I}}$, which is a tree with vertex set V and edge set E. Each vertex in the tree is called as a cluster. The label of cluster t is denoted by $\hat{t}$. The set of sons for a cluster $t \in T_{\mathcal{I}}$ is denoted by $\mathrm{sons}(t)$, which is the union of $\{w \in V | (t, w) \in E\}$. For any cluster $t \in T_{\mathcal{I}}$, either $\mathrm{sons}(t) = \emptyset$ or $t = \dot{\bigcup}_{w \in sons(t)} w$, where $\dot{\bigcup}$ denotes a disjoint union. The root of the tree is the index set $\mathcal{I} := \{1, 2, \ldots, N\}$. The uniquely determined predecessor (father) of a non-root cluster $t \in T_{\mathcal{I}}$ is denoted by $\mathcal{F}(t)$. The levels of the tree $T_{\mathcal{I}}$ are defined by

$$T_{\mathcal{I}}{}^{(0)} := \{\mathcal{I}\}, \quad T_{\mathcal{I}}{}^{(l)} := \{t \in T_{\mathcal{I}} | \mathcal{F}(t) \in T_{\mathcal{I}}{}^{(l-1)}\} \quad \text{for } l \in \mathbb{N}. \tag{21}$$

To construct a cluster tree, we start from the root cluster which is the full index set. We then find a disjoint partition of the index set and use this partition to create son clusters. We continue this procedure until the index number in each cluster is less than the *leafsize* which is a parameter to control the depth of the tree. Clusters with indices no more than *leafsize* are leaves. The set of leaves of $T_{\mathcal{I}}$ is denoted by $\mathcal{L}_{\mathcal{I}}$. The leaves of $T_{\mathcal{I}}$ on level l are denoted by $\mathcal{L}_{\mathcal{I}}^{(l)} := \mathcal{L}_{\mathcal{I}} \cap T_{\mathcal{I}}{}^{(l)}$.

To give an example, consider a rectangular plate as shown in Fig. 1(a). The basis functions used to expand the unknown current are drawn as small arrows. Each basis function is associated with one unknown. In total there are 80 unknowns involved in the computational domain. We split the computational domain into two subdomains as shown in Fig. 1(b). We continue to split until the number of unknowns in each sub-domain is less than or equal to the *leafsize*, which is 10 in this example. As a result, we generate a cluster tree as shown in Fig. 2. To simplify the notation, we use $\underline{i}$ to represent all the basis functions in region i. For this example, each region includes ten basis functions.

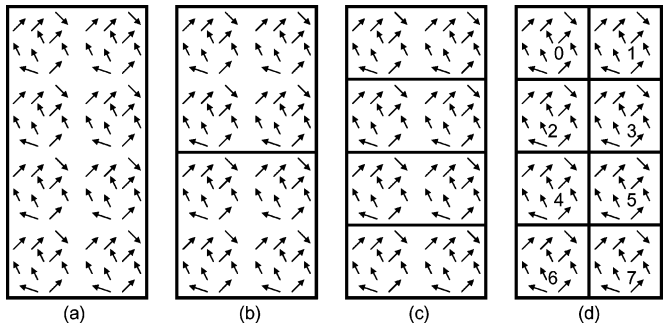


Fig. 1. Geometric partitioning for the construction of a cluster tree.

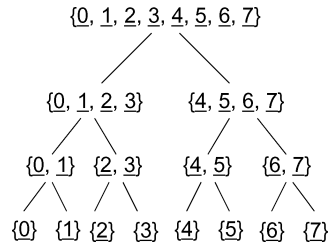


Fig. 2. Cluster tree.

A good cluster tree should make blocks become admissible as soon as possible. The admissibility of a block depends on the diameters of the supports of the involved basis functions and on the distance between the supports as shown in (11). We try to choose the cluster tree in a way that the diameters shrink quickly.

Let $T_{\mathcal{I}}$ and $T_{\mathcal{J}}$ be cluster trees for index sets $\mathcal{I}$ and $\mathcal{J}$. $T_{\mathcal{I}} \times T_{\mathcal{J}}$ will form a block cluster tree labeled by $T_{\mathcal{I} \times \mathcal{J}}$ and its nodes are called blocks. A level-consistent block cluster tree $T_{\mathcal{I} \times \mathcal{J}}$ is a special cluster tree for the index set $\mathcal{I} \times \mathcal{J}$ that satisfies 1) $\mathrm{Root}(T_{\mathcal{I} \times \mathcal{J}}) = (\mathrm{Root}(T_{\mathcal{I}}), \mathrm{Root}(T_{\mathcal{J}}))$; 2) each node $b \in T_{\mathcal{I} \times \mathcal{J}}$ has the form $b = (t, s)$ for clusters $t \in T_{\mathcal{I}}$ and $s \in T_{\mathcal{J}}$, and $\mathrm{level}(b) = \mathrm{level}(t) = \mathrm{level}(s)$; and 3) for each node $b = (t, s) \in T_{\mathcal{I} \times \mathcal{J}}$ with nonzero number of sons, $\mathrm{sons}(b)$ are all the combinations of sons of t and sons of s. The vertices of $T_{\mathcal{I} \times \mathcal{J}}$ are called block clusters.

A block cluster tree $T_{\mathcal{I} \times \mathcal{J}}$ is called admissible with respect to an admissibility condition if (t, s) is admissible, or $\mathrm{sons}(t) = \emptyset$, or $\mathrm{sons}(s) = \emptyset$, holds for all leaves (t, s) of $T_{\mathcal{I} \times \mathcal{J}}$. The set of the leaves is denoted by $\mathcal{L}_{\mathcal{I} \times \mathcal{J}}$. Based on the admissibility condition, the set of leaves of $T_{\mathcal{I} \times \mathcal{J}}$ is split into

$$\mathcal{L}^+{}_{\mathcal{I} \times \mathcal{J}} := \{b = (t, s) \in \mathcal{L}_{\mathcal{I} \times \mathcal{J}} : (t, s) \text{ are admissible}\}$$
$$\text{and } \mathcal{L}^-{}_{\mathcal{I} \times \mathcal{J}} := \mathcal{L}_{\mathcal{I} \times \mathcal{J}} \backslash \mathcal{L}^+{}_{\mathcal{I} \times \mathcal{J}} \tag{22}$$

i.e., admissible and inadmissible leaves. The admissible leaves are represented by low rank matrices as shown in (8), whereas the inadmissible ones are stored and computed by using full matrix representation as those in a dense matrix.

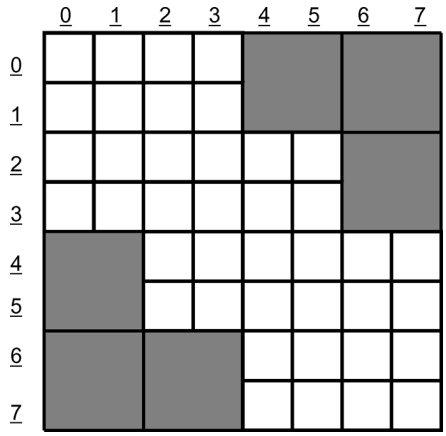


Fig. 3. $\mathcal{H}^2$-matrix structure.

Constructing an admissible block cluster tree from cluster trees $T_{\mathcal{I}}$ and $T_{\mathcal{J}}$ and a given admissibility condition can be done recursively: We test blocks level by level starting with $\mathrm{Root}(T_{\mathcal{I}})$ and $\mathrm{Root}(T_{\mathcal{J}})$, and descending in the tree. Given two clusters $t \in T_{\mathcal{I}}$ and $s \in T_{\mathcal{J}}$, we check the admissibility. If the two clusters are admissible, we are done. If they are not admissible, we repeat the procedure for all combinations of sons of t and sons of s.

The admissible block cluster tree defines an $\mathcal{H}$-Matrix structure. Consider the cluster tree $T_{\mathcal{I}}$ shown in Fig. 2. In our applications, $\mathcal{I} = \mathcal{J}$, the block cluster tree is hence formed between $T_{\mathcal{I}}$ and $T_{\mathcal{I}}$. Imagine another $T_{\mathcal{I}}$ is placed in parallel with the original $T_{\mathcal{I}}$ in Fig. 2. We start from $\mathrm{Root}(T_{\mathcal{I}})$ and $\mathrm{Root}(T_{\mathcal{I}})$, and test the admissibility between clusters $t \in T_{\mathcal{I}}$ and $s \in T_{\mathcal{I}}$ level by level. Clearly, at level 0, the two root clusters are not admissible. Hence, we descend to level 1. We test the admissibility between $\{\underline{0}, \underline{1}, \underline{2}, \underline{3}\}$ and $\{\underline{0}, \underline{1}, \underline{2}, \underline{3}\}$, between $\{\underline{0}, \underline{1}, \underline{2}, \underline{3}\}$ and $\{\underline{4}, \underline{5}, \underline{6}, \underline{7}\}$ (due to the symmetry, the other two combinations of the sons of the root clusters need not to be repeated). Because of the geometric proximity between the two clusters $\{\underline{0}, \underline{1}, \underline{2}, \underline{3}\}$ and $\{\underline{4}, \underline{5}, \underline{6}, \underline{7}\}$ as can be seen from Fig. 1, no admissible blocks are found. Hence, we descend to level 2. We test the admissibility between $\{\underline{0}, \underline{1}\}$ and $\{\underline{0}, \underline{1}\}$, $\{\underline{0}, \underline{1}\}$ and $\{\underline{2}, \underline{3}\}$, $\{\underline{0}, \underline{1}\}$ and $\{\underline{4}, \underline{5}\}$, $\{\underline{0}, \underline{1}\}$ and $\{\underline{6}, \underline{7}\}$, $\{\underline{2}, \underline{3}\}$ and $\{\underline{4}, \underline{5}\}$, $\{\underline{2}, \underline{3}\}$ and $\{\underline{6}, \underline{7}\}$, $\{\underline{4}, \underline{5}\}$ and $\{\underline{6}, \underline{7}\}$, $\{\underline{6}, \underline{7}\}$ and $\{\underline{6}, \underline{7}\}$. We find that $\{\underline{0}, \underline{1}\}$ and $\{\underline{4}, \underline{5}\}$ are admissible, $\{\underline{0}, \underline{1}\}$ and $\{\underline{6}, \underline{7}\}$ are admissible, and $\{\underline{2}, \underline{3}\}$ and $\{\underline{6}, \underline{7}\}$ are admissible, given an admissibility condition. Hence, we can stop without testing the admissibility of their sons. For other cluster blocks that are inadmissible, we descend to level 3 and repeat the same procedure.

The aforementioned procedure of constructing a block cluster tree results in a matrix structure shown in Fig. 3. Each block cluster corresponds to a matrix block. The shaded matrix blocks are admissible blocks in which the $\mathcal{H}^2$-matrix-based low-rank approximation is used; the unshaded ones are inadmissible blocks in which a full matrix representation is employed.

C. Rank Function

Two cluster trees $T_{\mathcal{I}}$ and $T_{\mathcal{J}}$ with different sizes are shown in Fig. 4.

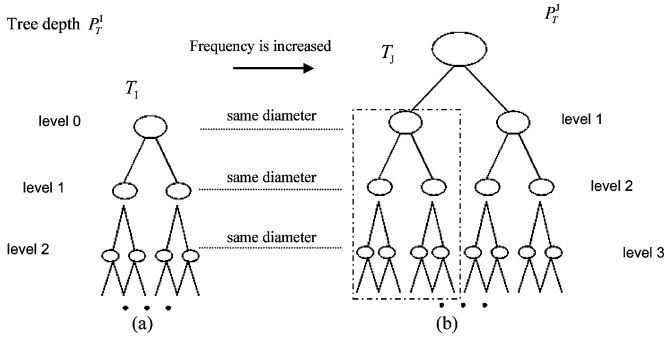


Fig. 4. Two cluster trees with different sizes.

Consider cluster tree $T_{\mathcal{I}}$ shown in Fig. 4(a). From the properties of a cluster tree, it is known that the lower the level is, the larger the cluster size is. Therefore, when the level is lower, the admissible blocks formed in that level are larger. Hence, $1+2/\left[(\kappa+1/\mathrm{dist}(Q_t,Q_s))\mathrm{diam}(Q_t \times Q_s)\right]$ in (19) becomes smaller, which leads to a slower convergence rate with respect to p. If $p = p1$ is used in level 3 where a relative error ε is obtained, larger p is required in level 2 to make the approximation accurate up to the same level.

Consider the case when frequency is increased. Since the number of unknowns increases with frequency, the cluster tree constructed for the same problem is enlarged as shown in Fig. 4(b). Assuming that $T_{\mathcal{I}}$ and $T_{\mathcal{J}}$ have the same *leaf-size*, then clusters in level $(P_T^{\mathcal{I}} - i)$ of $T_{\mathcal{I}}$ have the similar diameters as the clusters in level $(P_T^{\mathcal{J}} - i)$ of $T_{\mathcal{J}}$ with $i = 0, 1, \ldots, \min(P_T^{\mathcal{I}}, P_T^{\mathcal{J}})$. Therefore, $T_{\mathcal{I}}$ can be viewed as a sub-tree of $T_{\mathcal{J}}$ as shown in Fig. 4(b). Hence, increasing frequency is equivalent to increasing the tree depth. In order to keep the same order of accuracy across all tree levels, polynomial order p should be increased when larger admissible blocks appear in the lower level. This suggests that we adaptively decide the rank: for small clusters, a lower order approximation is sufficient, while a larger cluster requires a higher order p to keep the same accuracy.

Based on the above analysis, if the number of interpolation points p, and hence the rank of an admissible block, is increased when ascending an inverted tree, it is feasible to achieve the same level of accuracy across all tree levels, and hence across a range of frequencies. Thereby, we can define a polynomial order function which decreases with tree level. Enlightened by [26], this function is defined as

$$p(b) = \hat{a} + \hat{b}(L - l(b)) \tag{23}$$

where

$$\begin{aligned} L &= L_{\min} = \min\{level(\tau) : \tau \in \mathcal{L}_{\mathcal{I}}\} \\ l(b) &= level(t) = level(s) \\ p(b) &= \hat{a} \text{ if } L \le l(b) \end{aligned} \tag{24}$$

and $\hat{a}$, $\hat{b}$ are two constants. Given a cluster t in an $\mathcal{H}^2$ tree, its rank $k_{\mathrm{var}}(t)$ can be determined from (23) as

$$k_{\mathrm{var}}(t) = p^d(t) = [\hat{a} + \hat{b}(L - l(t))]^d \tag{25}$$

where $d = 1, 2, 3$, for 1-, 2-, and 3-D problems, respectively. Since the coefficients of this rank function are all constants, when N increases with frequency or electric size, these coefficients do not change. This is the key reason why a same set of parameters can render the accuracy the same in the range of electric sizes while still keeping the complexity to the same order. In the next section, we show that the cost of storage and matrix-vector multiplication is linear by using (25).

D. Compact Representation of $\mathbf{G}$ in $O(N)$ Parameters

Based on the rank function introduced in the above section, we can show that $\mathbf{G}$ can be compactly represented by $O(N)$ parameters. Before deriving the compact representation, we introduce the following concepts and notations.

(1) Block cluster trees constructed for an $\mathcal{H}$-matrix or $\mathcal{H}^2$-matrix have an important property: for each cluster $t \in T_{\mathcal{I}}$, the cardinality of the sets $col(t) := \{s \in T_{\mathcal{J}} : (t,s) \in T_{\mathcal{I}\times\mathcal{J}}\}$ and $row(s) := \{t \in T_{\mathcal{I}} : (t,s) \in T_{\mathcal{I}\times\mathcal{J}}\}$ is bounded by a constant C_{sp} [24], which is called as sparsity constant.
(2) In our construction, each non-leaf cluster t has two sons, i.e., $\#\text{sons}(t) = 2$.
(3) The total number of clusters in the cluster tree is bounded by $2N$.

All the admissible blocks in $\mathbf{G}$ are represented by a factorized form as given in (8), which is repeated as

$$\tilde{\mathbf{G}}^{t,s} := \mathbf{V}^t \mathbf{S}^{t,s} \mathbf{V}^{s^{\mathrm{T}}}, \quad \mathbf{V}^t \in \mathbb{C}^{t\times 2K^t},\ \mathbf{S}^{t,s} \in \mathbb{C}^{2K^t\times 2K^s},\ \mathbf{V}^s \in \mathbb{C}^{s\times 2K^s}.$$

Hence, all the admissible blocks are uniquely defined by $\mathbf{V} = (\mathbf{V}^t)_{t\in T_{\mathcal{I}}}$ and $(\mathbf{S}^{t,s})_{t\in T_{\mathcal{I}}, s\in T_{\mathcal{I}}}$.

Apparently, $\mathbf{V}^t$ needs to be stored for each cluster $t \in T_{\mathcal{I}}$. In fact, $\mathbf{V}^t$ only needs to be stored for each *leaf* cluster since $\mathbf{V}^t$ is nested. The nested property is the key factor that enables us to reduce the complexity of an IE-based solution for electrodynamic problems. The reason why $\mathbf{V}^t$ is nested is given as below.

Consider cluster t' which is a son of t. We use the same space of polynomials for all clusters. Hence, $L_v^t(\vec{r})$ in (10) can be written as

$$L_v^t(\vec{r}) = \sum_{v'\in K^{t'}} E_{v'v}^{t'} L_{v'}^{t'}(\vec{r}) \tag{26}$$

where

$$E_{v'v}^{t'} = L_v^t(\xi_{v'}^{t'}). \tag{27}$$

As a result

$$\begin{aligned}\mathbf{V}_{1mv}^t &= \iint_{S_m} \vec{J}_m(\vec{r}) L_v^t(\vec{r}) ds \\ &= \sum_{v'\in K^{t'}} E_{v'v}^{t'} \iint_{S_m} \vec{J}_m(\vec{r}) L_{v'}^{t'}(\vec{r}) ds \\ &= \sum_{v'\in K^{t'}} E_{v'v}^{t'} \mathbf{V}_{1mv'}^{t'} = (\mathbf{V}_1^{t'} \mathbf{E}^{t'})_{mv}. \end{aligned} \tag{28}$$

where $\mathbf{E}^{t'} \in \mathbb{C}^{Kt'\times Kt}$. Similarly, $\mathbf{V}_{2mv}^t = (\mathbf{V}_2^{t'} \mathbf{E}^{t'})_{mv}$. Hence, assuming that $sons(t) = \{t_1, t_2\}$ with $t_1 \neq t_2$, we have

$$\mathbf{V}^t = \begin{pmatrix} \mathbf{V}^{t1}\mathbf{E}^{t1} \\ \mathbf{V}^{t2}\mathbf{E}^{t2} \end{pmatrix} = \begin{pmatrix} \mathbf{V}^{t1} & \\ & \mathbf{V}^{t2} \end{pmatrix} \begin{pmatrix} \mathbf{E}^{t1} \\ \mathbf{E}^{t2} \end{pmatrix}. \tag{29}$$

This means that we only need to store the matrices $\mathbf{V}^t$ for leaf clusters t and use the transfer matrices $\mathbf{E}$ to represent them implicitly for all other clusters. Since the transfer matrices $\mathbf{E}$ only require $(\#K^{t'})(\#K^t) = k_{\text{var}}(t')k_{\text{var}}(t)$ units of storage, while the matrices $\mathbf{V}^t$ require $(\#\hat{t})(\#K^t) = (\#\hat{t})k_{\text{var}}(t)$ units where $k_{\text{var}}(t) \ll (\#\hat{t})$ in general, the nested representation is memory efficient.

Thus, $\mathbf{V} = (\mathbf{V}^t)_{t\in T_{\mathcal{I}}}$ can be stored as follows.

1) For each *leaf* cluster $t \in T_{\mathcal{I}}$, we store the matrix $\mathbf{V}^t$, which requires $O(k_{\text{var}}(t))\#\hat{t}$ units of storage:

$$St(\text{all leaf clusters}) = \sum_{t\in\mathcal{L}_{\mathcal{I}}} O(k_{\text{var}}(t))\#\hat{t}. \tag{30}$$

Since $L_{\min} \le l(t)$ is satisfied, we obtain

$$k_{\text{var}}(t) = \hat{a}^d. \tag{31}$$

Since the index sets corresponding to the leaves of $T_{\mathcal{I}}$ form a partition of $\mathcal{I}$, we have

$$St(\text{all leaf clusters}) = \sum_{t\in\mathcal{L}_{\mathcal{I}}} O(k_{\text{var}}(t))\#\hat{t} = O(\hat{a}^d)\cdot N. \tag{32}$$

2) For each non-leaf cluster $t \in T_{\mathcal{I}}$, we store the transfer matrices $\mathbf{E}^{t'}$ for all $t' \in sons(t)$, which requires $O(k_{\text{var}}(t)k_{\text{var}}(t'))$ storage units. Since each cluster has two sons in our tree construction

$$\begin{aligned} &St(\text{all nonleaf clusters}) \\ &= \sum_{t\in T_{\mathcal{I}}\setminus\mathcal{L}_{\mathcal{I}}} \sum_{t'\in sons(t)} O(k_{\text{var}}(t)k_{\text{var}}(t')) \\ &\le 2\sum_{t\in T_{\mathcal{I}}\setminus\mathcal{L}_{\mathcal{I}}} O(k_{\text{var}}^2(t)) \le 2O\left(\sum_{l=0}^{L}\sum_{t\in T_{\mathcal{I}}^{(l)}} [\hat{a}+\hat{b}(L-l)]^{2d}\right) \\ &\le 2O\left(\sum_{l=0}^{L}[\hat{a}+\hat{b}(L-l)]^{2d}\cdot 2^l\right) \\ &\le 2O((\hat{a}+\hat{b})^{2d})\sum_{l=0}^{L}(1+L-l)^{2d}\cdot 2^l \\ &\le 2O((\hat{a}+\hat{b})^{2d})\cdot 2^L\cdot\sum_{l=0}^{L}(l+1)^{2d}\cdot 2^{-l}. \end{aligned} \tag{33}$$

In the following, we prove that

$$(l+1)^{2d} \le \left(\frac{2d}{\ln 1.5}\right)^{2d}\cdot 1.5^l \tag{34}$$

is satisfied for any of $d = 1, 2, 3$.

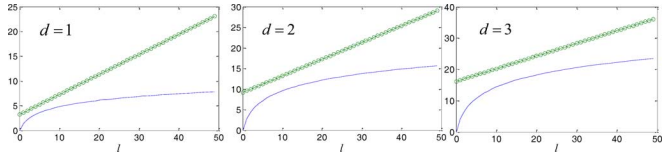


Fig. 5. Illustration of two functions: "o" represents $y1 = \ln\{(2d/\ln 1.5)^{2d} \cdot 1.5^l\}$, "−" represents $y2 = \ln\{(l+1)^{2d}\}$.

From Fig. 5, it is clear that

$$y1 = \ln\left\{\left(\frac{2d}{\ln 1.5}\right)^{2d} \cdot 1.5^l\right\} = \ln\left\{\left(\frac{2d}{\ln 1.5}\right)^{2d}\right\} + l\ln(1.5)$$

and

$$y2 = \ln\{(l+1)^{2d}\} = 2d\ln(l+1)$$

both increase with l, but $y1$ increases faster than $y2$. Therefore, (34) is proved.

Based on (34), (33) becomes

$$\begin{aligned} &St(\text{all nonleaf clusters}) \\ &\le 2O((\hat{a}+\hat{b})^{2d}) \cdot 2^L \cdot \sum_{l=0}^{L}\left(\frac{2d}{\ln 1.5}\right)^{2d} \cdot 1.5^l \cdot 2^{-l} \\ &\le 2 \cdot O\left(\left(\frac{2d(\hat{a}+\hat{b})}{\ln 1.5}\right)^{2d}\right) \cdot N \sum_{l=0}^{L}\left(\frac{3}{4}\right)^l \\ &\le 2N \cdot O\left(\left(\frac{2d(\hat{a}+\hat{b})}{\ln 1.5}\right)^{2d}\right) \cdot 4 \\ &\le 8O\left(\left(\frac{2d(\hat{a}+\hat{b})}{\ln 1.5}\right)^{2d}\right) \cdot N \end{aligned} \tag{35}$$

and hence the storage of all the non-leaf clusters scales with N linearly.

In addition to $\mathbf{V} = (\mathbf{V}^t)_{t\in T_{\mathcal{I}}}$, we need to store $(\mathbf{S}^{t,s})_{t\in T_{\mathcal{I}}, s\in T_{\mathcal{I}}}$ for admissible blocks. For each admissible block b, the coupling matrix $\mathbf{S}^b$ requires $O({k_{\text{var}}}^2(b))$ units of storage. Hence,

$$\begin{aligned} &St(\text{all admissible blocks}) \\ &= \sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}} O({k_{\text{var}}}^2(b)) \le \sum_{t\in T_{\mathcal{I}}}\sum_{s\in col(t)} O({k_{\text{var}}}^2(b)) \\ &\le C_{sp}\sum_{t\in T_{\mathcal{I}}} O({k_{\text{var}}}^2(t)) \le 4C_{sp}O\left(\left(\frac{2d(\hat{a}+\hat{b})}{\ln 1.5}\right)^{2d}\right) \cdot N. \end{aligned} \tag{36}$$

The last inequality in (36) is obtained via the same steps given in (33).

For each inadmissible block b, the admissibility of $T_{\mathcal{I}\times\mathcal{I}}$ implies that t and s have to be leaves of $T_{\mathcal{I}}$, respectively. So the matrix $\mathbf{G}^{t,s}$ requires at most $O(\#\hat{t}\#\hat{s})$ units of storage.

$$\begin{aligned} &St(\text{all inadmissible blocks}) \\ &= \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}} O(\#\hat{t}\#\hat{s}) \le \sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}} O(n_{\min}^2) \\ &\le \sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{I}}} O(n_{\min}^2) \\ &\le \sum_{t\in T_{\mathcal{I}}}\sum_{s\in col(t)} O(n_{\min}^2) \le C_{sp}\sum_{t\in T_{\mathcal{I}}} O(n_{\min}^2) \\ &= C_{sp}O(n_{\min}^2)(\#T_{\mathcal{I}}) \le 2C_{sp}O(n_{\min}^2)N. \end{aligned} \tag{37}$$

Summing over (32), (35), (36), (37), since C_{sp}, $n_{\min}$, $\hat{a}$, $\hat{b}$, d are constants, clearly, matrix $\mathbf{G}$ can be represented by $O(N)$ parameters and stored in $O(N)$ complexity.

E. Matrix-Vector Multiplication in $O(N)$ Operations

Multiplying $\mathbf{G}$ with its admissible blocks represented by (8) by a vector x can be performed in four steps.

1) Forward transformation: Compute $x^s := {\mathbf{V}^s}^{\mathrm{T}} x|_{\hat{s}}$ for all clusters $s \in T_{\mathcal{I}}$.
2) Multiplication: Compute $y^t := \sum_{s\in R^t} \mathbf{S}^{t,s}x^s$ for all clusters $t \in T_{\mathcal{I}}$, where $R^t := \{s \in T_{\mathcal{I}} : (t,s) \in \mathcal{L}^+(T_{\mathcal{I}\times\mathcal{I}})\}$, i.e., R^t contains all clusters s such that (t,s) is an admissible leaf of the block cluster tree.
3) Backward transformation: Compute $y \in \mathbb{C}^{\mathcal{I}}$ defined by $y_i := \sum_{t,i\in\hat{t}} (\mathbf{V}^t y^t)_i$.
4) Inadmissible blocks: These blocks are treated the same as those in the original matrix $\mathbf{G}$, i.e., a full-matrix-vector multiplication is performed.

The time complexity of the matrix-vector multiplication based on the proposed rank function is analyzed as follows.

1) *Forward Transformation*: Let $s \in T_{\mathcal{I}}$, if $\text{sons}(s) = \emptyset$, i.e., s is a leaf cluster of $T_{\mathcal{I}}$, we compute $x^s := {\mathbf{V}^s}^{\mathrm{T}} x|_{\hat{s}}$ directly. The cost is $O(k_{\text{var}}(s))\#\hat{s}$ operations for each leaf. Summing over all the leaves, we obtain

$$\sum_{s\in\mathcal{L}_{\mathcal{I}}} O(k_{\text{var}})\#\hat{s} \le O(\hat{a}^d)N. \tag{38}$$

If s is a non-leaf cluster, we can first compute for $x^{s'}$ all $s' \in sons(s)$. Since $\mathbf{V} = (\mathbf{V}^s)_{s\in T_{\mathcal{I}}}$ is nested, we have

$$x^s = {\mathbf{V}^s}^{\mathrm{T}} x|_{\hat{s}} = \sum_{s'\in sons(s)} ({\mathbf{V}^{s'}\mathbf{E}^{s'}})^{\mathrm{T}} x|_{\hat{s}'} = \sum_{s'\in sons(s)} ({\mathbf{E}^{s'}})^{\mathrm{T}} x^{s'}. \tag{39}$$

Therefore, we can use the vectors $x^{s'}$ to compute x^s, i.e., for each non-leaf cluster, we can use the contribution from its two sons to obtain x^s, the cost of which is

$O(k_{\text{var}}(s)k_{\text{var}}(s'))$. Hence, the total complexity of the forward transformation is

$$\begin{aligned}&\text{Comp}\left(\sum_{s\in T_{\mathcal{I}}}\mathbf{V}^{s^{\mathrm{T}}}x|_{\hat{s}}\right)\\&=\sum_{s\in\mathcal{L}_{\mathcal{I}}}O(k_{\text{var}}(s))\#\hat{s}+\sum_{\substack{s\in T_{\mathcal{I}}\setminus\mathcal{L}_{\mathcal{I}}\\ s'\in sons(s)}}O(k_{\text{var}}(s)k_{\text{var}}(s'))\\&\le O(\hat{a}^d)N+\sum_{s\in T_{\mathcal{I}}}O(k_{\text{var}}^2(s))\\&\le O(\hat{a}^d)N+4O\left(\left(\frac{2d(\hat{a}+\hat{b})}{\ln 1.5}\right)^{2d}\right)N.\end{aligned}\tag{40}$$

2) *Multiplication of the Coupling Matrix:* The multiplication of $\mathbf{S}^{t,s}x^s$ requires $O(k_{\text{var}}^2(b))$ operations, and has to be performed for each $s\in R^t$, hence

$$\begin{aligned}&\text{Comp}\left(\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}}\mathbf{S}^{t,s}x^s\right)\\&=\sum_{b=(t,s)\in\mathcal{L}^+_{\mathcal{I}\times\mathcal{I}}}O(k_{\text{var}}^2(b))\le\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{I}}}O(k_{\text{var}}^2(b))\\&\le\sum_{t\in T_{\mathcal{I}}}\sum_{s\in col(t)}O(k_{\text{var}}^2(b))\le C_{sp}\sum_{t\in T_{\mathcal{I}}}O(k_{\text{var}}^2(b))\\&\le 4C_{sp}O\left(\left(\frac{2d(\hat{a}+\hat{b})}{\ln 1.5}\right)^{2d}\right)\cdot N.\end{aligned}\tag{41}$$

3) *Backward Transformation*: For each leaf cluster of $T_{\mathcal{I}}$, we compute $\mathbf{V}^t y^t$ directly, the cost of which is $O(k_{\text{var}}(t))\#\hat{t}$ operations. For each non-leaf cluster, we add its contribution back to its two sons. For example, consider cluster t' which is a son of t. We have

$$\mathbf{V}^t y^t=(\mathbf{V}^{t'}\mathbf{E}^{t'}y^t).\tag{42}$$

Hence, $(\mathbf{V}^{t'}y^{t'})_i+(\mathbf{V}^t y^t)_i=\mathbf{V}^{t'}(y^{t'}+\mathbf{E}^{t'}y^t)_i$, meaning the contribution of $\mathbf{V}^t$ to y_i can be efficiently taken into consideration by adding $\mathbf{E}^{t'}y^t$ back to its son, the cost of which is $O(k_{\text{var}}^2(t))$ only. Hence, the total complexity is

$$\begin{aligned}&\text{Comp}\left(\sum_{t\in T_{\mathcal{I}}}\mathbf{V}^t y^t\right)\\&=\sum_{t\in\mathcal{L}_{\mathcal{I}}}O(k_{\text{var}}(t))\#\hat{t}+\sum_{\substack{t\in T_{\mathcal{I}}\setminus\mathcal{L}_{\mathcal{I}}\\ t'\in sons(t)}}O(k_{\text{var}}(t)k_{\text{var}}(t'))\\&\le O(\hat{a}^d)N+4O\left(\left(\frac{2d(\hat{a}+\hat{b})}{\ln 1.5}\right)^{2d}\right)N.\end{aligned}\tag{43}$$

4) *Multiplication of Non-Admissible Blocks:* The complexity of multiplying each non-admissible block by a vector is $O(n_{\min}^2)$, summing over all the non-admissible blocks, we obtain

$$\begin{aligned}&\text{Comp}\left(\sum_{b=(t,s)\in\mathcal{L}^-_{\mathcal{I}\times\mathcal{I}}}\mathbf{G}^{t,s}x|_s\right)\\&\le\sum_{b=(t,s)\in\mathcal{L}_{\mathcal{I}\times\mathcal{I}}}O(n_{\min}^2)\\&\le\sum_{t\in T_{\mathcal{I}}}\sum_{s\in col(t)}O(n_{\min}^2)\le 2C_{sp}O(n_{\min}^2)N.\end{aligned}\tag{44}$$

Adding the aforementioned four steps, the matrix-vector multiplication only requires $O(N)$ operations.

F. Choice of Simulation Parameters

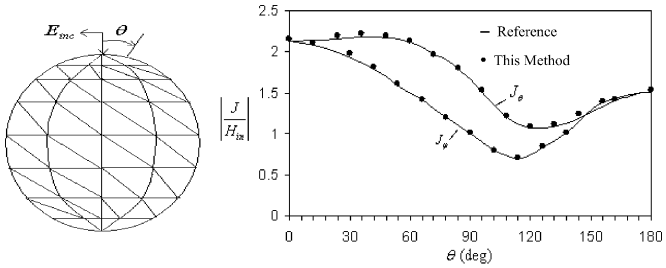
There are only four simulation parameters to choose in the proposed method: η, *leafsize* $n_{\min}$, $\hat{a}$ and $\hat{b}$. The $\hat{a}$ and $\hat{b}$ are used to determine p as can be seen from (23). As shown in (20), the smaller η is and the larger p is, the better the accuracy is, but if η is chosen to be too small, and p is chosen to be too large, the simulation will become inefficient. Therefore, there exists a tradeoff between accuracy and efficiency. Generally speaking, for static problems, $1\le\eta\le 2$ is enough for achieving a good accuracy; for high-frequency problems, $0.2\le\eta\le 1$ can be chosen. The *leafsize* $n_{\min}$ and $\hat{a}$ can be determined together. Given a required level of accuracy across a range of frequencies, $\hat{a}$ can be determined by achieving the accuracy requirement at the lower end of the frequency range, and $n_{\min}$ can be chosen based on $n_{\min}\ge 0.5\hat{a}^d$. This can help make $\mathcal{H}^2$-based representation most efficient in both memory and CPU time. The parameter $\hat{b}$ can be chosen between 0 and $\hat{a}$.

IV. Numerical Results

To demonstrate the accuracy and efficiency of the proposed $\mathcal{H}^2$-matrix-based IE solver, we simulated a conducting sphere, a conducting plate, and a half sphere of various electric sizes. In addition, we simulated a Sierpiski Gasket from 1074 unknowns to 263 234 unknowns, the electric size of the longest side length of which is from 1 wavelength to 64 wavelengths.

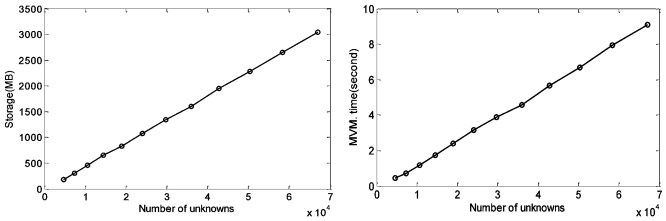
A. Conducting Sphere

First, we validated the proposed IE solver on a 0.2λ conducting sphere. The spherical surface was discretized into 1140 triangular elements. The cluster tree was built by cardinality balanced construction of the index set $\mathcal{I}=\{1,2,\dots N\}$. The construction yielded a block cluster tree T whose leaves partitioned the product index set $\mathcal{I}\times\mathcal{I}$. The parameter η used for the admissibility condition was chosen as 1 and the *leafsize* $n_{\min}$ was chosen as 20. Each admissible block of the $\mathcal{H}^2$-matrix was assembled by interpolation. The number of interpolation points, p, was 2 by choosing $\hat{a}=2$ and $\hat{b}=0$. In Fig. 6(b), the current distribution along the principal cuts obtained by the proposed



(a) (b)

Fig. 6. (a) Conducting sphere. (b) Current distribution ($p = 2$, $\|I - \tilde{I}\| / \|I\| = 0.03$).



(a) (b)

Fig. 7. Simulation of a conducting sphere. (a) Memory consumption. (b) CPU time cost.

TABLE I
ACCURACY OF $\mathcal{H}^2$-MATRIX-BASED REPRESENTATION $\tilde{\mathbf{G}}$ VERSUS ELECTRIC SIZE

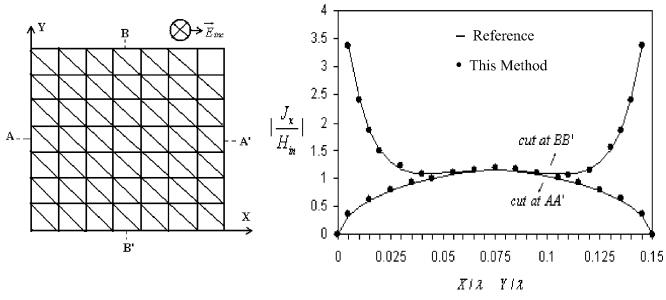
sphere_size/ λ	N	$\frac{\|\mathbf{G}-\tilde{\mathbf{G}}\|}{\|\mathbf{G}\|}$
3	4680	1.187729e-05
5	10620	6.509749e-05
7	16650	8.463785e-05
9	29700	4.841575e-05
11	42840	4.192058e-05
13	58380	9.650041e-05
15	67050	8.327470e-05

approach was plotted with respect to angle θ. Excellent agreement with the reference solution reported in [23] is observed.

Next, we investigated the accuracy, storage requirement, and CPU time cost of the proposed IE solver in a range of electric sizes from 3λ to 15λ. The simulation parameters were chosen as $\eta = 0.8$, $leafsize = 63$, $\hat{a} = 5$, and $\hat{b} = 1$. In Table I, the error of the $\mathcal{H}^2$-matrix-based representation of system matrix $\mathbf{G}$ was listed with respect to the electric size and the number of unknowns. Clearly, good accuracy is observed. More importantly, the error is kept to the same order in the whole range. In Fig. 7(a), the storage in MB of the proposed solver was plotted with respect to the number of unknowns, and hence the electric size, from which linear cost can be clearly observed. In Fig. 7(b), the CPU time per iteration was plotted versus the number of unknowns. Again, a linear cost is observed.

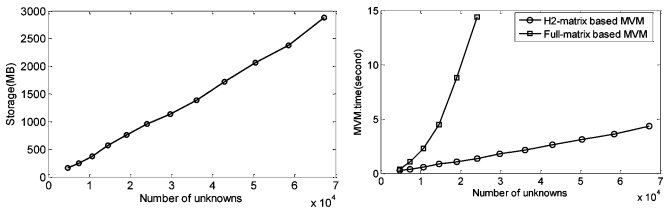
B. Conducting Square Plate

We then validated the proposed IE solver on a 0.15λ plate illuminated by a normally incident plane wave shown in Fig. 8(a).



(a) (b)

Fig. 8. (a) Conducting square plate. (b) Current distribution ($p = 2$, $\|I - \tilde{I}\| / \|I\| = 0.025$).



(a) (b)

Fig. 9. Simulation of a conducting plate. (a) Memory consumption as a function of unknown number. (b) CPU time as a function of unknown number.

The plate was discretized into 1160 triangular elements. The cluster tree was built by cardinality balanced construction of the index set $\mathcal{I} = \{1, 2, \ldots N\}$. The parameter η used for the admissibility condition was chosen as 1 and the *leafsize* $n_{\min}$ was chosen as 20. The number of interpolation points, p, was 2 by choosing $\hat{a} = 2$ and $\hat{b} = 0$. Fig. 8(b) shows the current distribution along two principal cuts of the square plate generated by using the $\mathcal{H}^2$-matrix-based approach in comparison with the reference result presented in [23], which reveals an excellent agreement.

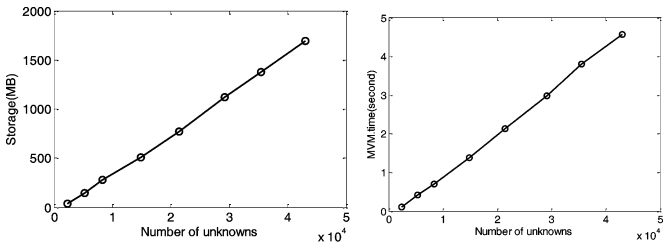
Next we simulated the same plate from 4λ to 20λ. The simulation parameters were chosen as $\eta = 1$, $leafsize = 20$, $\hat{a} = 5$, and $\hat{b} = 2$. In Table II, we listed the error of the $\mathcal{H}^2$ matrix-based $\mathbf{G}$ with respect to the number of unknowns. Once again, good accuracy is observed. In addition, the accuracy is observed to be in the same order in the whole range. In Fig. 9, the memory consumption and CPU time were plotted with respect to the number of unknowns, respectively. Both showed linear scaling with the number of unknowns. In Fig. 9(b), for comparison, we also plotted the CPU time required by a full-matrix-based matrix-vector multiplication. The advantage of the proposed method can be clearly seen.

C. Half Conducting Sphere

The third example is a half sphere with skew incidence from the concave side. The simulation parameters were chosen as $\eta = 0.8$, $leafsize = 63$, $\hat{a} = 4$, and $\hat{b} = 1$. In Table III, the error of the $\mathcal{H}^2$-matrix-based $\tilde{\mathbf{G}}$ was listed with respect to the electric size of the half sphere and the number of unknowns. A constant order of accuracy can be observed in the entire range. Different from the simulation performed for example A, in which $\hat{a}$ was chosen as 5. In this example, $\hat{a}$ was chosen as 4. Good accuracy is still obtained as can be seen from Table III. In Fig. 10, we plotted

TABLE II
ACCURACY OF $\mathcal{H}^2$-MATRIX-BASED REPRESENTATION $\tilde{\mathbf{G}}$ WITH RESPECT TO ELECTRIC SIZE

Plate_size/ λ	N	$\frac{\|\mathbf{G}-\tilde{\mathbf{G}}\|}{\|\mathbf{G}\|}$
4	4720	4.989619e-05
6	7400	1.292243e-06
8	10680	5.996929e-05
10	14560	2.263827e-05
12	24560	2.080167e-05
14	29800	4.424070e-05
16	36080	7.751298e-05
18	50130	3.690129e-05
20	67600	4.131000e-05



(a) (b)

Fig. 10. Simulation of a half sphere. (a) Memory consumption as a function of unknown number. (b) Time complexity.

the memory and CPU time cost. Once again, linear scaling is observed.

D. Sierpiski Gasket

The last example was a multiscale structure, Sierpiski gasket, as shown in Fig. 11. The electric size of the smallest triangle patch remained constant. The number of geometrical details was increased with frequency. For example, as shown in Fig. 11, one iteration was added as the frequency doubles, while the electric size of the smallest patch was always kept to be 1λ. Seven iterations were performed, resulting in 1074 to 262 434 unknowns. The electric size of the longest side length ranged from 1λ to 64λ.

The simulation parameters were chosen as $\eta = 1$, $leafsize = 20$, $\hat{a} = 4$, and $\hat{b} = 1$. In Table IV, the error of the $\mathcal{H}^2$-matrix-based $\tilde{\mathbf{G}}$ was listed with respect to the electric size of the longest side length of the Sierpiski gasket. A constant order of accuracy can be observed in the entire range. For the case of 64λ that involved 262 434 unknowns, the computation of the matrix error was impossible since the full-matrix representation required a memory that was beyond what our computer can offer. We hence checked the maximal admissible block error, which was defined as $||\mathbf{G}^{(t,s)} - \tilde{\mathbf{G}}^{(t,s)}||/\max(||\mathbf{G}^{(t,t)}, \mathbf{G}^{(s,s)}||)$. It can be proven that the total matrix error is bounded from above by the maximal admissible block error. In Table IV, we list the maximal admissible block error with respect to the electric size. Clearly, a constant order of accuracy can be observed in the entire range.

TABLE III
ACCURACY OF $\mathcal{H}^2$-MATRIX -BASED REPRESENTATION $\tilde{\mathbf{G}}$ WITH RESPECT TO ELECTRIC SIZE

half_sphere_size/ λ	N	$\frac{\|\mathbf{G}-\tilde{\mathbf{G}}\|}{\|\mathbf{G}\|}$
3	2340	4.247538e-04
5	5310	2.965317e-04
7	8320	4.336523e-04
9	14850	6.425261e-04
11	21420	2.175265e-04
13	29190	5.464219e-04
15	35520	6.394084e-04
17	43010	7.004305e-04

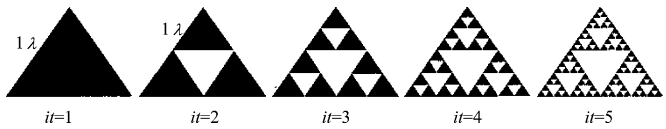
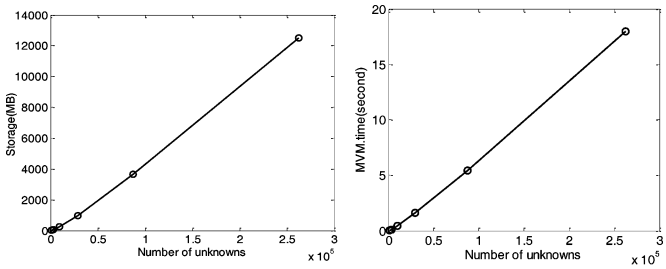


Fig. 11. Illustration of Sierpiski gasket at different iterations.



(a) (b)

Fig. 12. Simulation of a Sierpiski gasket. (a) Memory consumption as a function of unknown number. (b) Time complexity.

TABLE IV
ACCURACY OF THE $\mathcal{H}^2$-MATRIX -BASED REPRESENTATION $\tilde{\mathbf{G}}$ WITH RESPECT TO ELECTRIC SIZE

it	*Total_electric_size*/ λ	N	$\frac{\|\mathbf{G}-\tilde{\mathbf{G}}\|}{\|\mathbf{G}\|}$	Maximal Admissible Block Error $\frac{\|\mathbf{G}^{(t,s)}-\tilde{\mathbf{G}}^{(t,s)}\|}{\max(\|\mathbf{G}^{(t,t)},\mathbf{G}^{(s,s)}\|)}$
2	2	1074	1.455915e-05	2.956683e-05
3	4	3234	2.195113e-05	6.478373e-05
4	8	9714	2.088980e-05	1.077240e-04
5	16	29154	3.358065e-05	6.762226e-05
6	32	87474	5.794269e-05	1.135951e-04
7	64	262434	---	3.574245e-05

Since the total matrix error is no greater than the maximal admissible block error, the accuracy of the total matrix is also constant across the entire range from 2λ to 64λ. In Fig. 12, we plot the memory and CPU time of the proposed solver with respect to the number of unknowns. Linear cost again can be clearly seen.

V. CONCLUSION

Integral-equation-based methods generally lead to dense systems of linear equations. The resulting matrices, although dense,

can be specified by a reduced set of parameters. In this paper, we introduce $\mathcal{H}^2$-matrix as a general mathematical framework to accelerate IE-based computation of electrodynamic problems. $\mathcal{H}^2$-matrix-based methods are kernel independent, and hence suitable for any IE-based formulation. In addition, the $\mathcal{H}^2$-matrix-based framework allows for the acceleration of both iterative and direct IE solvers [29], [30], which cannot be easily done in the framework of other fast IE solvers.

The error bound was derived for the $\mathcal{H}^2$-matrix-based representation of an electrodynamic problem. It was shown that exponential convergence with respect to the number of interpolation points can be achieved irrespective of the electric size. The impact of error on the computational complexity was studied in detail. Our finding is summarized as the following. In general, an $\mathcal{H}^2$ matrix can be stored in $O(N)$ units, and an $\mathcal{H}^2$-matrix-based matrix-vector multiplication can be performed in $O(N)$ operations. However, when applying $\mathcal{H}^2$-matrix-based methods to electrodynamic problems, in order to keep the error constant across a range of electric sizes, we have to increase the cost of both memory consumption and CPU time. This is similar to what has been observed in an FMM-based method, which has $O(N)$ complexity in static applications and $O(N \log N)$ complexity in dynamic applications. However, the accuracy of an $\mathcal{H}^2$ matrix based representation of an electrodynamic problem is a complicated function of the number of interpolation points p, and hence the rank, as shown in (20). In addition the accuracy is also a function of η. Indeed, if one blindly adjusts η and p with respect to N to control the accuracy, it is difficult to justify $O(N)$ complexity since not only N changes, but also other parameters change when sweeping a range of electric sizes. However, in this paper, we showed a viable approach to tackle this problem. The rank was defined as a function instead of a constant. In addition, this function has constant coefficients which do not change with N. By introducing such a function, it becomes feasible to use a same set of parameters, η, *leafsize*, $\hat{a}$, and $\hat{b}$ (the coefficients of the rank function) as shown in this work, to achieve linear cost in both CPU time and memory consumption while still keeping the error to the same order in a range of electric sizes. This has been verified by our complexity analysis as well as numerical experiments. In this paper, we studied rank as a function of tree level, and we tested electric sizes of our current interest from small electric sizes to 64 wavelengths. Certainly, η can also be defined as a function, and larger electric sizes could be explored. These are our future research topics.

REFERENCES

[1] *Fast and Efficient Algorithms in Computational Electromagnetics*, W. C. Chew, J. M. Jin, E. Michielssen, and J. M. Song, Eds.. Norwood, MA: Artech House, 2001.

[2] N. N. Bojarski, "k-space formulation of the electromagnetic scattering problem," Air Force Avionics Lab., 1971, Tech. Rep. AFAL-TR-71-75.

[3] E. Bleszynski, M. Bleszynski, and T. Jarozewicz, "AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems," *Radio Sci.*, vol. 31, pp. 1225–1251, Sep.-Oct. 1996.

[4] F. Ling, V. I. Okhamtovski, W. Harris, S. McCracken, and A. Dengi, "Large-scale broad-band parasitic extraction for fast layout verification of 3D RF and mixed-signal on-chip structures," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 1, pp. 264–273, Jan. 2005.

[5] A. E. Yilmaz, J. M. Jin, and E. Michielssen, "A parallel FFT-accelerated transient field-circuit simulator," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 9, pp. 2851–2865, Sep. 2005.

[6] S. Kapur and D. E. Long, "IES^3: A fast integral equation solver for efficient 3-dimensional extraction," in *IEEE/ACM Int. Conf. Comput.-Aided Design Dig.*, Nov. 1997, pp. 448–455.

[7] M. Seo and J. F. Lee, "A single-level low rank IE-QR algorithm for PEC scattering problems using EFIE formulation," *IEEE Trans. Antennas Propag.*, vol. 52, no. 8, pp. 2141–2146, Aug. 2004.

[8] D. Gope and V. Jandhyala, "Efficient solution of EFIE via low-rank compression of multilevel predetermined interactions," *IEEE Trans. Antennas Propag.*, vol. 53, no. 10, pp. 3324–3333, Oct. 2005.

[9] R. J. Burkholder and J. F. Lee, "Fast dual-MGS block-factorization algorithm for dense MoM matrices," *IEEE Trans. Antennas Propag.*, vol. 52, no. 7, pp. 1693–1699, Jul. 2004.

[10] J. Shaeffer, "Direct solve of electrically large integral equations for problem sizes to 1 M unknowns," *IEEE Trans. Antenna Propag.*, vol. 56, no. 8, pp. 2306–2313, Aug. 2008.

[11] R. J. Adams, Y. Xu, X. Xu, S. D. Gedney, and F. X. Canning, "Modular fast direct electromagnetic analysis using local-global solution modes," *IEEE Trans. Antenna Propag.*, vol. 56, no. 8, pp. 2427–2441, Aug. 2008.

[12] W. Hackbusch and B. Khoromskij, "A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices," *Computing*, vol. 62, pp. 89–108, 1999.

[13] W. Hackbusch and B. N. Khoromskij, "A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems," *Computing*, vol. 64, pp. 21–47, 2000.

[14] S. Borm, "Introduction to hierarchical matrices with applications," *EABE*, vol. 27, pp. 403–564, 2003.

[15] H. Forster, "Fast boundary methods for magnetostatic interactions in micromagnetics," *IEEE. Trans. Magn.*, vol. 39, no. 5, pp. 2513–2515, Sep. 2003.

[16] J. Ostrowski, "Fast BEM-solution of Laplace problems with $\mathcal{H}$-matrices and ACA," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 627–630, 2006.

[17] W. Hackbusch, B. Khoromskij, and S. Sauter, "On $\mathcal{H}^2$–matrices," in *Lecture on Applied Mathematics*, H. Bungartz, R. Hoppe, and C. Zenger, Eds. Munich, Germany: Springer, 2000, pp. 9–29.

[18] S. Borm, "Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices," *Computing*, vol. 69, pp. 1–35, 2002.

[19] S. Borm, "$\mathcal{H}^2$-matrices — Multilevel methods for the approximation of integral operators," *Comput. Visual. Sci.*, vol. 7, pp. 173–181, 2004.

[20] L. Banjai and W. Hackbush, "Hierarchical matrix techniques for low- and high-frequency Helmholtz problems," *IMA J. Numerical Anal.*, vol. 28, pp. 46–79, 2008.

[21] W. Chai and D. Jiao, "An $\mathcal{H}$-matrix-based method for reducing the complexity of integral-equation-based solutions of electromagnetic problems," in *Proc. IEEE Int. Symp. Antennas Propag.*, Jun. 2008.

[22] W. Chai and D. Jiao, "An $\mathcal{H}^2$-matrix-based integral-equation solver of linear-complexity for large-scale full-wave modeling of 3D circuits," in *Proc. IEEE 17th Conf. Elect. Perform. Electron. Packag. (EPEP)*, Oct. 2008, pp. 283–286.

[23] S. M. Rao and D. R. Wilton, "Electromagnetic scattering by surfaces of arbitrary shape," *IEEE Trans. Antennas Propag.*, vol. AP-30, no. 3, pp. 409–418, May 1982.

[24] S. Borm, L. Grasedyck, and W. Hackbusch, "Hierarchical matrices," *Lecture Note 21 of the Max Planck Institute for Mathematics in the Sciences*, 2003.

[25] S. Borm, "$\mathcal{H}^2$-matrix approximation of integral operators by interpolation," *Appl. Numer. Math.*, vol. 43, pp. 129–143, 2002.

[26] S. Sauter, "Variable order panel clustering," *Computing*, vol. 64, pp. 223–261, 2000.

[27] L. Grasedyck, "Adaptive geometrically balanced clustering of $\mathcal{H}$-matrices," *Computing*, vol. 73, pp. 1–23, 2004.

[28] S. Borm and L. Grasedyck, "Low-rank approximation of integral operators by interpolation," *Computing*, vol. 72, pp. 325–332, 2004.

[29] H. Liu, W. Chai, and D. Jiao, "An $\mathcal{H}$-matrix-based fast direct solver for finite-element-based analysis of electromagnetic problems," in *2009 Int. Annu. Rev. Progress in Appl. Comput. Electromagn. (ACES)*, Mar. 2009, p. 5.

[30] W. Chai and D. Jiao, "An $\mathcal{H}^2$-matrix-based direct integral-equation solver of linear complexity for solving electrodynamic problems," in *IEEE Int. Symp. Antennas Propag.*, Jun. 2009, p. 4.

Wenwen Chai received the B.S. degree from the University of Science and Technology of China, Hefei, in 2004 and the M.S. degree from the Chinese Academy of Sciences, Beijing, in 2007, both in electrical engineering. She is currently pursuing the Ph.D. degree in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN.

She currently works in the On-Chip Electromagnetics Group at Purdue University. Her research is focused on computational electromagnetics, high-performance VLSI CAD, fast and high-capacity numerical methods.

Dan Jiao (S'00–M'02–SM'06) received the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign in October 2001.

She then worked in the Technology CAD Division at Intel Corporation until September 2005 as Senior CAD Engineer, Staff Engineer, and Senior Staff Engineer. In September 2005, she joined Purdue University, West Lafayette, IN, as an Assistant Professor in the School of Electrical and Computer Engineering. She is now an Associate Professor. She has authored two book chapters and over 100 papers in refereed journals and international conferences. Her current research interests include computational electromagnetics, high-frequency digital, analog, mixed-signal, and RF IC design and analysis, high-performance VLSI CAD, modeling of micro- and nano-scale circuits, applied electromagnetics, fast and high-capacity numerical methods, fast time-domain analysis, scattering and antenna analysis, RF, microwave, and millimeter wave circuits, wireless communication, and bio-electromagnetics.

Dr. Jiao received NSF CAREER Award in 2008. In 2006, she received the Jack and Cathie Kozik Faculty Start up Award, which recognizes an outstanding new faculty member in Purdue ECE. She also received an ONR award through Young Investigator Program in 2006. In 2004, she received the Best Paper Award from Intel's annual corporate-wide technology conference (Design and Test Technology Conference) for her work on generic broadband model of high-speed circuits. In 2003, she won the Intel Logic Technology Development (LTD) Divisional Achievement Award in recognition of her work on the industry-leading BroadSpice modeling/simulation capability for designing high-speed microprocessors, packages, and circuit boards. She was also awarded the Intel Technology CAD Divisional Achievement Award for the development of innovative full-wave solvers for high frequency IC design. In 2002, she was awarded by Intel Components Research the Intel Hero Award (Intel-wide she was the tenth recipient) for the timely and accurate two- and three- dimensional full-wave simulations. She also won the Intel LTD Team Quality Award for her outstanding contribution to the development of the measurement capability and simulation tools for high frequency on-chip cross-talk. She was the winner of the 2000 Raj Mittra Outstanding Research Award given her by the University of Illinois at Urbana-Champaign. She has served as the reviewer for many IEEE journals and conferences.