A Fast $\mathcal{H}$-Matrix Based Direct Integral Equation Solver with Optimized $\mathcal{H}$-Partition and Minimized Rank for Large-Scale Electrodynamic Analysis

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

***Abstract*—In this paper, we will develop a fast direct integral equation solver for large-scale electrodynamic analysis. In this solver, we significantly reduce the cost of an $\mathcal{H}$-matrix-based computation of electrodynamic problems with prescribed accuracy satisfied, by constructing an efficient matrix algebra-based method that minimizes the rank of each admissible block based on accuracy; and by developing a new frequency-dependent $\mathcal{H}$-partition that minimizes the number of admissible blocks at each tree level for each frequency point. We will then develop a fast $\mathcal{H}$-matrix-based LU factorization for directly solving the dense system matrix resulting from an integral-equation-based analysis of large-scale electrodynamic problems. The proposed direct solver successfully solves dense matrices that involve more than 1 million unknowns associated with electrodynamic problems of 96 wavelengths in fast CPU time (less than 20 hours in LU factorization, 85 seconds in LU solution), modest memory consumption, and with the prescribed accuracy satisfied on a single CPU running at 3 GHz. As an algebraic method, the underlying fast techniques are kernel independent.**

***Index Terms*—Integral-equation based methods, electrodynamic analysis, direct solution, $\mathcal{H}$ matrix, large-scale analysis**

I. Introduction

THe integral equation (IE) based computational electromagnetic 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 $O(N^2)$, with N being the matrix size. When an iterative solver is used, the memory requirement remains the same, and the computing time is proportional to $O(N_{it}N^2)$, where N_{it} denotes the total number of iterations required to reach convergence. The N_{it} is, in general, problem dependent, solver dependent, and accuracy dependent. In recent years, fast solvers such as fast multipole based methods [1]–[3] and FFT-based methods [4], [5] have been developed that dramatically reduce the memory requirement of the iterative IE solvers to $O(N)$, and the CPU time to $O(N_{it}N \log N)$ for electrodynamic problems. This represents an impressive improvement as compared with conventional $O(N^3)$ or $O(N_{it}N^2)$ techniques.

Fast direct solvers have also been developed for electrodynamic problems. Most recent work can be seen from [6], [7]. LU factorization of $O(N^2)$ time complexity and $O(N^{1.5})$ memory complexity has been shown by numerical experiments. Compared to iterative solvers, direct solvers have advantages when the number of iterations is large or the number of right hand sides is large. For example, if there exist N right hand sides, each of which costs $O(N_{it}N \log N)$ operations, the total cost of the iterative solver will be $O(N_{it}N^2 \log N)$, which is expensive.

Recently, in [8], [9], we introduced and further developed the $\mathcal{H}$- and $\mathcal{H}^2$-matrix based mathematical framework for reducing the computational cost of iterative IE-based solvers for electrodynamic problems. In [10]–[12], we further demonstrate that the inverse as well as the LU factorization of a dense system matrix resulting from the IE-based analysis of static problems or problems with moderate electric sizes can be directly obtained in $O(N)$ complexity, for arbitrary 3-D structures with non-uniform materials.

The focus of this work is large-scale electrodynamic problems. For this class of problems, we develop a fast direct integral equation based solver. In this solver, we significantly reduce the computational cost of the $\mathcal{H}$-matrix-based direct matrix solution with prescribed accuracy satisfied. We propose two methods to reduce the cost of an $\mathcal{H}$-matrix based computation. One is to minimize the rank of each admissible block based on accuracy requirements. We determine the minimal rank for each admissible block by an efficient matrix algebra -ased method. The algebraic method has a linear cost for each block, and hence the computational overhead is negligible. The other method we develop for reducing the computational cost is to optimize the $\mathcal{H}$-partition to reduce the number of admissible blocks at each tree level based on accuracy requirements for each frequency point. We show that the admissibility condition used in the conventional $\mathcal{H}$-partition is empirical instead of theoretical, and hence not optimal for a given accuracy and a given frequency. We then develop a new $\mathcal{H}$-partition method which is frequency dependent, and also controlled by accuracy requirements. With the proposed new partition, the number of admissible blocks at each tree level is significantly reduced compared to that generated by the admissibility condition based, i.e. geometry based partition. The methods developed in this work for minimizing the rank and optimizing the $\mathcal{H}$-partition not only can be used in the proposed solver, but also can be used in other fast integral equation solvers to speed up their computation. Moreover, we develop an efficient $\mathcal{H}$-matrix-based LU-factorization for directly solving the dense system matrix resulting from an IE-based analysis of large-scale electrodynamic problems.

This work was supported by grants from NSF under award No. 0747578 and award No. 0702567. The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA.

This use of this work is restricted solely for academic purposes. The author of this work owns the copyright and no reproduction in any form is permitted without written permission by the author.

We also give a detailed implementation of the $\mathcal{H}$-based LU factorization, which is not reported elsewhere.

The basic idea of the proposed work was reported in [21]. In this paper, we complete it from both theoretical and numerical perspectives. The remainder of this paper is organized as follows. In Section II, we give the background of the $\mathcal{H}$-matrix based analysis of electrodynamic problems. In Section III, we analyze the storage and the computational cost of the $\mathcal{H}$-matrix based computation of electrodynamic problems. In Section IV, we propose methods to reduce the cost of the $\mathcal{H}$-matrix based computation. In Section V, we give numerical proof on the existence of an $\mathcal{H}$-matrix based representation of the inverse and LU factors of an IE based system matrix for an electrodynamic problem. In Section VI, we present a number of pseudo-codes to show a detailed implementation of the LU factorization using $\mathcal{H}$ matrices. In Section VII, we analyze total computational cost. In Section VIII, we present numerical results to demonstrate the accuracy and efficiency of the proposed direct IE solver. In Section IX, we summarize our findings.

II. BACKGROUND

The $\mathcal{H}$-matrix based methods are algebraic methods that are kernel independent. In the following, we use an electric-field integral equation as an example to illustrate the basic concept of $\mathcal{H}$-matrix based methods.

A. Electric Field Integral Equation

We consider the electric-field integral equation (EFIE) [3], [22]

$$\begin{aligned}\mathbf{E}_i|_{tan} &= \iint_S [j\omega\mu \mathbf{J}_S(\mathbf{r}) g(\mathbf{r},\mathbf{r}')]_{tan}\, dS' \\ &- \iint_S \left[\frac{j}{\omega\epsilon}\left(\nabla'\cdot\mathbf{J}_S(\mathbf{r}')\right)\nabla' g(\mathbf{r},\mathbf{r}')\right]_{tan} dS', \quad (1)\end{aligned}$$

in which Green's function $g(\mathbf{r},\mathbf{r}') = \frac{e^{-j\kappa|\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|}$, $\mathbf{J}_S$ is the induced surface current density, ω is angular frequency, κ is wave number, and subscript tan denotes tangential component.

By expanding the unknown $\mathbf{J}_S$ using the RWG basis functions [22], a Method of Moment based solution of (1) results in the following linear system of equations

$$\mathbf{G}I = V, \quad (2)$$

where

$$\begin{aligned}\mathbf{G}_{mn} &= \iint_{S_m} dS \iint_{S_n} dS' \,[j\omega\mu \mathbf{J}_m(\mathbf{r})\cdot\mathbf{J}_n(\mathbf{r}') \\ &- \frac{j}{\omega\epsilon}\left(\nabla\cdot\mathbf{J}_m(\mathbf{r})\right)\left(\nabla'\cdot\mathbf{J}_n(\mathbf{r}')\right)\Big]\, g(\mathbf{r},\mathbf{r}'), \quad (3)\end{aligned}$$

and

$$V_m = \iint_{S_m} \mathbf{J}_m(\mathbf{r})\cdot\mathbf{E}_i dS. \quad (4)$$

The conventional way to solve (2) could be very expensive, since the entries of $\mathbf{G}$ are all nonzero. In the following section, we introduce the $\mathcal{H}$ matrix as a data-sparse representation of $\mathbf{G}$, from which a significant reduction in computational cost can be achieved.

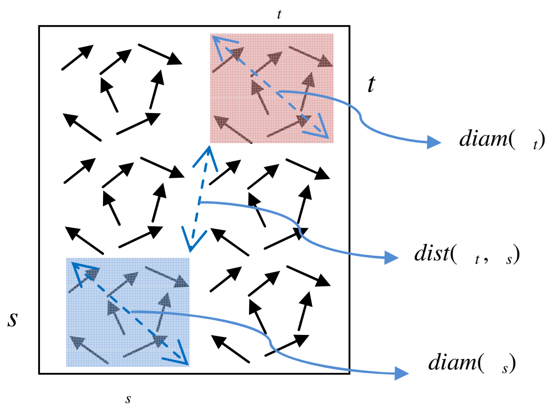


Fig. 1: An illustration of the admissibility condition.

B. Definition of the $\mathcal{H}$ Matrix

An $\mathcal{H}$ matrix is generally associated with an admissibility condition [16]. To define an admissibility condition, we denote the whole index set containing the indices of the basis functions in the computational domain by $\mathcal{I} = \{1, 2, \cdots, N\}$, where N is the total number of unknowns. Considering two subsets t and s of the $\mathcal{I}$, the admissibility condition is defined as

$$(t,s) \text{ are admissible} = \begin{cases} \text{True} & \text{if } \min\{diam(Q_t), diam(Q_s)\} \\ & \le \eta dist(Q_t, Q_s); \\ \text{False} & \text{otherwise}, \end{cases} \quad (5)$$

where, as shown in Fig. 1, $Q_{t,s}$ is the minimal subset of the space containing the supports of all basis functions belonging to t or s, $diam(\cdot)$ is the Euclidean diameter of a set, $dist(\cdot)$ is the Euclidean distance of two sets, and η is a positive parameter that can be used to control the admissibility condition. If subsets t and s do not satisfy the admissibility condition, they are called inadmissible. The admissibility condition shown in (5) is empirical rather than theoretical. It is controlled by an empirical parameter η, instead of a prescribed accuracy. In this work, we will develop a new method to partition a matrix into admissible and inadmissible blocks for electrodynamic analysis, which is controlled by accuracy requirements.

In an $\mathcal{H}$-matrix representation, an inadmissible block keeps its original full-matrix representation; while an admissible block has a factorized low-rank form. To be specific, an admissible block $\mathbf{G}^{t,s}$ that is formed by subsets t and s can be written as a factorized form

$$\mathbf{G}^{t,s} = \mathbf{A}\mathbf{B}^T, \quad (6)$$

where $\mathbf{G}^{t,s} \in \mathbb{C}^{m\times n}$, $\mathbf{A} \in \mathbb{C}^{m\times k}$, $\mathbf{B} \in \mathbb{C}^{n\times k}$, and $k \in \mathbb{N}$ is the rank of $\mathbf{G}^{t,s}$. Here, as long as k is less than the minimum of m and n, $\mathbf{G}^{t,s}$ is low rank. The k is not required to be $O(1)$.

If all the blocks $\mathbf{G}^{t,s}$ formed by the admissible (t,s) in $\mathbf{G}$ can be represented by a factorized low-rank form shown in (6), $\mathbf{G}$ has an $\mathcal{H}$-matrix representation ([16], p. 18). Clearly, to store admissible $\mathbf{G}^{t,s}$, we only need to store $\mathbf{A}$ and $\mathbf{B}$, the cost of which is $O(k(m+n))$ in contrast with the original

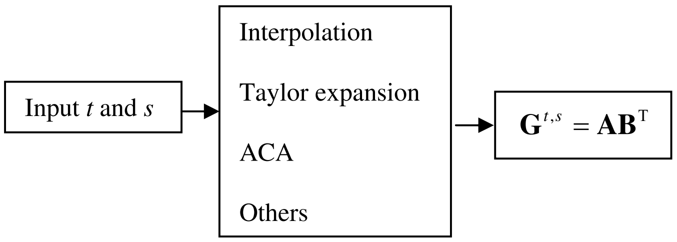


Fig. 2: A flow to obtain a rank-k matrix representation of a given dense matrix block.

storage that is $O(mn)$. Similar cost reduction in matrix-vector and matrix-matrix multiplication can be achieved. Therefore, if we are able to represent the dense matrix resulting from an IE based analysis by an $\mathcal{H}$ matrix, we can reduce the computational cost of IE-based solutions significantly. The detail of the $\mathcal{H}$-matrix representation of an IE-based system matrix is given in next subsection.

C. $\mathcal{H}$-Matrix Representation of the IE-Based System Matrix

The essential idea of an $\mathcal{H}$-matrix representation is to represent the off-diagonal matrix block that satisfies the admissibility condition by a rank-k matrix shown in (6).

There are three representative schemes that can be used to generate a rank-k matrix of an IE-based dense matrix block: interpolation, Taylor expansion, and adaptive cross approximation (ACA). In [8], [9], an interpolation scheme is used to obtain an $\mathcal{H}$-representation without any compression cost for electrodynamic kernels. The Taylor expansion can also be used to locally replace the kernel function by degenerate approximations, as shown in ([16], p. 10). Neither interpolation nor Taylor expansion involves rank compression cost. However, the rank determined by the two schemes may not be a minimal rank required by accuracy. Another approach is to directly compute a low rank approximation of the original matrix up to a prescribed accuracy. The representative method is ACA [16], [19], which is purely algebraic. In [6], [20], ACA was used to solve electromagnetic problems. For each matrix block, the cost of ACA is linear. A simple flow to obtain the $\mathcal{H}$-matrix representation of a given matrix block is shown in Fig. 2.

D. $\mathcal{H}$-Matrix Partition

To use an $\mathcal{H}$-matrix representation, we have to partition a dense matrix into admissible blocks and inadmissible blocks. An admissible block is represented by a rank-k matrix shown in (6), and an inadmissible block is represented by a full-matrix form.

In [8], [9], we show how to build a cluster tree and a block cluster tree to efficiently carry out the $\mathcal{H}$-matrix partition for a dense matrix resulting from the IE-based analysis of electrodynamic problems. We denote the cluster tree constructed for the full index set $\mathcal{I}$ by $T_{\mathcal{I}}$. We then find a disjoint partition of the index set and use this partition to create children 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. A block cluster tree, as shown in Fig. 3(a), between cluster trees $T_{\mathcal{I}}$ and $T_{\mathcal{I}}$ is constructed based on a given admissibility condition recursively. In Fig. 3(a), each link in an upper level represents an admissible block shown by a shaded block in Fig. 3(b). The number of links is bounded by the sparsity constant C_{sp}, which is the maximum number of blocks that can be formed by a cluster in a block cluster tree ([16], p. 125). 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 C_{sp}.

The admissible block cluster tree results in a matrix partition as shown in Fig. 3(b). The shaded matrix blocks are admissible blocks; the un-shaded ones are inadmissible blocks. The admissible blocks and inadmissible blocks together form a complete $\mathcal{H}$ partition. In Fig. 3(b), we also label the levels present in an $\mathcal{H}$ partition, which is shown by dashed rectangular boxes.

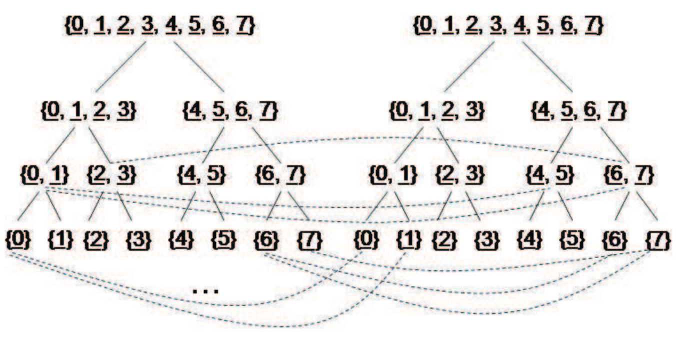
III. ANALYSIS OF STORAGE REQUIREMENTS AND OPERATION COUNTS FOR $\mathcal{H}$-MATRIX BASED COMPUTATION OF ELECTRODYNAMIC PROBLEMS

In mathematical literature [16], it is shown that storage requirements and matrix-vector multiplications using $\mathcal{H}$-matrices are of complexity $O(kN \log N)$, where k is the upper bound of the rank of the admissible blocks, i.e. blocks that represent a far-field interaction. Moreover, the inverse and LU of an $\mathcal{H}$ matrix can be obtained in $O(k^2 N \log^2 N)$ complexity.

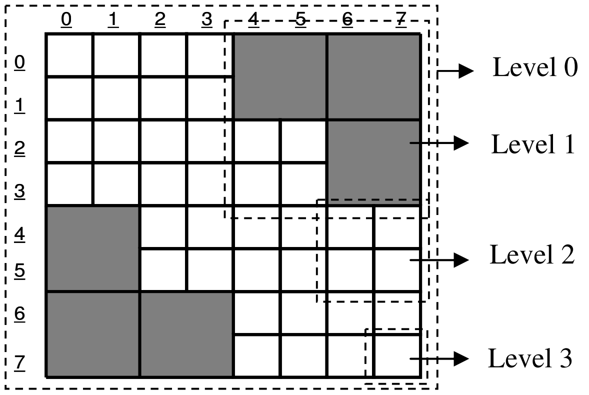
For frequency independent kernels, a constant rank k across all the admissible blocks is sufficient to generate a constant order of accuracy for the $\mathcal{H}$-based computation of the dense system matrix. This can be theoretically verified from the error bound for an $\mathcal{H}$-matrix based representation of a static kernel [16]. As a result, the k can be excluded from the complexity bounds. Thus, for static problems, the inverse and LU of an $\mathcal{H}$ matrix can be obtained in $O(N \log^2 N)$ complexity, and the storage and matrix-vector multiplications are both of complexity $O(N \log N)$.

For frequency-dependent kernels, however, a constant k, in general, cannot guarantee a constant order of accuracy across a wide range of electric sizes. Since rank k is electric size dependent, it becomes a variable that is different in each admissible block, especially in admissible blocks at different tree levels since each tree level corresponds to a different electric size. Traditionally, people use the maximal rank among all the blocks, k_{max}, to bound the complexity of $\mathcal{H}$-based computation of electrodynamic problems. In this paper, we show that the cost of an $\mathcal{H}$-matrix based computation of electrodynamic problems can be greatly reduced from the computational cost obtained based on k_{max}.

This is because first of all, using k_{max} to bound the complexity largely overestimates the cost since many blocks have a rank smaller than k_{max}; secondly, the k_{max}-based complexity is obtained based on the admissibility condition based $\mathcal{H}$-partition that is solely geometry based. By changing the $\mathcal{H}$-partition to a frequency-dependent one that is optimized based on accuracy, the complexity of $\mathcal{H}$-based computation of electrodynamic problems can be reduced. To help understand



(a)



(b)

Fig. 3: (a) A block Cluster Tree. (b) An $\mathcal{H}$-matrix partition.

the importance of $\mathcal{H}$-partition in determining the actual computational complexity, we give an example. In Fig. 4, we plot the storage of the blocks that have the largest rank between $0.5k_{max}$ and k_{max} in comparison with the storage of the rest of the blocks that have a smaller rank, for a cone sphere across a wide range of electric sizes from $\kappa a = 2$ to $\kappa a = 600$, where κ is wave number and a is the largest physical dimension of the cone sphere. It is clear that the total storage is not dominated by the cost of rank-k_{max} blocks. In fact, the storage of the rank-k_{max} blocks is much less than that of the rest of the blocks. This is because although k_{max} is the largest, the number of admissible blocks that have rank k_{max} is also the smallest, and the cost of an $\mathcal{H}$-matrix based computation is determined not only by the rank of each admissible block, but also by the number of blocks that can have such a rank. In other words, if the weight of each rank-k_i block in the storage and CPU time cost is the same, then the contribution from rank-k_{max} blocks would dominate the storage and CPU time cost. However, the weight of each rank-k_i block is not the same; it is determined by the number of blocks that can have such a rank. Since the number of admissible blocks at each tree level is determined by the $\mathcal{H}$-matrix partition, the $\mathcal{H}$-matrix partition plays an important role in reducing the computational cost. One can optimize the $\mathcal{H}$-matrix partition to reduce the computational complexity. This is an important factor that did not receive much attention in previous research. In the following, we give a more quantitative analysis.

Take the storage complexity of an $\mathcal{H}$-matrix as an example, which is also the complexity of an $\mathcal{H}$-matrix based matrix-vector multiplication. In an $\mathcal{H}$ matrix, since each admissible block $\mathbf{G}^{m_i \times n_i}$ has a factorized form $\mathbf{A}_{m_i \times k_i} \mathbf{B}^T_{n_i \times k_i}$ with rank k_i, the storage is reduced from the conventional full-matrix based $m_i \times n_i$ units to $k_i(m_i + n_i)$ units. By summing up the storage of all the admissible blocks, we obtain

$$\text{Storage} = \sum_{i=1}^{nk} k_i(m_i + n_i), \tag{7}$$

where nk is the total number of admissible blocks. The above

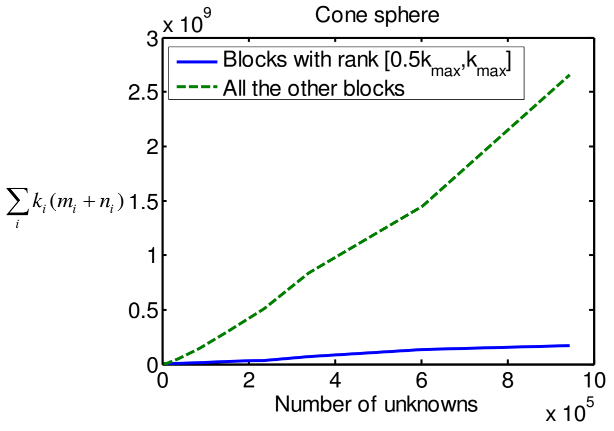


Fig. 4: Comparison between the storage of the rank ($0.5k_{max}$-k_{max}) blocks and that of the other blocks for a cone-sphere from $\kappa a = 2$ to $\kappa a = 600$.

can be evaluated as

$$\text{Storage} = \sum_{l=0}^{p} \sum_{i=1}^{nk_l} k_i(m_i + n_i), \tag{8}$$

where l is tree level, p is tree depth, and nk_l is the number of admissible blocks at level l.

Clearly, as can be seen from (7) and (8), if we use the maximal k among all the admissible blocks, k_{max}, to bound the computational complexity, we will overestimate the complexity because many admissible blocks have a rank k_i much smaller than k_{max}. Moreover, the number of admissible blocks at each tree level, nk_l, also plays an important role in determining the computational complexity. Existing admissibility condition based $\mathcal{H}$-partition is geometry based. It is not optimized for a given accuracy and a given frequency. There exists a large space to optimize the $\mathcal{H}$-partition to reduce nk_l without sacrificing accuracy.

A. *Proposed definition of average* partition *rank, k_{ave}, for the $\mathcal{H}$-based computation of electrodynamic problems*

From the aforementioned analysis, the cost of an $\mathcal{H}$-based computation of high-frequency problems is determined not

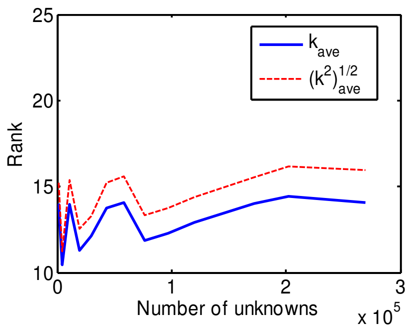


Fig. 5: Comparison of two average partition rank for a 3D PEC plate with electric size from 2 to 30 wavelengths.

only by the block rank k_i but also by the number of admissible blocks at each tree level, i.e. $\mathcal{H}$-partition. In light of this fact, we propose a new parameter, average *partition* rank k_{ave}, in order to analyze the computational cost of an $\mathcal{H}$-based computation of high-frequency problems in a more objective way. We define average *partition* rank k_{ave} as the following:

$$k_{ave} = \frac{\sum_{i=1}^{nk} k_i(m_i + n_i)}{\sum_{i=1}^{nk}(m_i + n_i)}, \tag{9}$$

where nk is the total number of admissible blocks, k_i is the rank of the i-th admissible block, m_i and n_i are the number of rows and columns of the admissible block. From the above definition, it is clear that k_{ave} is not only related to the block rank k_i, but also related to the $\mathcal{H}$-partition. Different partitions can result in different k_{ave}. That is why we name k_{ave} as average *partition* rank to distinguish it from the *block* rank that does not contain any information about partitions. Different from maximal block rank k_{max}, the k_{ave} can be reduced by changing $\mathcal{H}$-partition, which will become clear by numerical results shown in Section IV.

Similarly, we can define an average *square partition* rank as the following.

$$(k^2)_{ave} = \frac{\sum_{i=1}^{nk} k_i^2(m_i + n_i)}{\sum_{i=1}^{nk}(m_i + n_i)}. \tag{10}$$

From (10) and (9), it can be seen that the $(k^2)_{ave}$ is an $O(k_{ave}^2)$ quantity. As an example, in Fig. 5, we plot k_{ave} and $\sqrt{(k^2)_{ave}}$ for a 3D conducting plate from 2 to 30 wavelengths. It can be seen that these two average rank follow each other closely.

B. Parameters that can be used to reduce the cost of $\mathcal{H}$-based computation of electrodynamic problems

By using (9), the storage complexity (7) can be written as

$$\text{Storage} = k_{ave}\sum_{i=1}^{nk}(m_i + n_i), \tag{11}$$

The summation of the m_i and n_i over all the admissible blocks can be evaluated as the following:

$$\sum_{i=1}^{nk}(m_i + n_i) \le \sum_{l=0}^{p}\sum_{i=1}^{nk_l}(\frac{N}{2^l}\times 2) = \sum_{l=0}^{p} nk_l(\frac{N}{2^l}\times 2), \tag{12}$$

in which we use the fact that the row/column dimension of a block at level l is $\frac{N}{2^l}$, where $l = 0$ represents the root level. The above should be evaluated based on actual nk_l, which is problem dependent and partition dependent. Substituting (12) into (11), we obtain

$$\text{Storage} \le k_{ave}\sum_{l=0}^{p} nk_l(\frac{N}{2^l}\times 2). \tag{13}$$

Thus, it becomes clear how to reduce the storage of $\mathcal{H}$-based computation of electrodynamic problems: we should reduce k_{ave} and nk_l.

To analyze the CPU time cost, take the matrix-matrix multiplication as an example. The cost for each admissible block involved in the multiplication is $C_{sp}{k_i}^2(m_i + n_i)$ ([16], pp. 127-130). By summing up the cost of all the admissible blocks across all the tree levels, we obtain

$$\begin{aligned}\text{Operation counts} &= C_{sp}\sum_{l=0}^{p}\sum_{i=1}^{nk}{k_i}^2(m_i + n_i)\\ &\le C_{sp}\sum_{l=0}^{p}(k^2)_{ave}\sum_{i=1}^{nk}(m_i + n_i)\ \text{(From (10))}\\ &\le C_{sp}(k^2)_{ave}\sum_{l=0}^{p}\sum_{i=1}^{nk}(m_i + n_i)\\ &\le C_{sp}k_{ave}^2\sum_{l=0}^{p}\sum_{l=0}^{p} nk_l(m_i + n_i),\end{aligned} \tag{14}$$

where p is the tree depth that is proportional to $\log N$. In the last inequality, we use the fact that $(k^2)_{ave}$ is an $O(k_{ave}^2)$ quantity, as shown in Section III-A. Again, from the above, it can be seen that by reducing k_{ave} and nk_l, we can reduce the cost of an $\mathcal{H}$-based computation. It is worth mentioning that the above analysis of storage and time cost is valid for any $\mathcal{H}$-partition.

IV. PROPOSED METHODS FOR REDUCING THE COMPUTATIONAL COST OF $\mathcal{H}$-MATRIX BASED DIRECT SOLUTION OF ELECTRODYNAMIC PROBLEMS

From (13) and (14), it can be seen that to reduce the computational cost of an $\mathcal{H}$-based computation of electrodynamic problems, we have to reduce k_{ave} and nk_l. In next two subsections, we propose methods to minimize k_{ave} for a given partition and optimize $\mathcal{H}$-partition to minimize nk_l based on accuracy requirements. The number of admissible blocks at each tree level, nk_l, can be qualitatively measured by C_{sp}, which is the maximal number of blocks that can be formed by a cluster.

A. Proposed method for minimizing average partition *rank k_{ave} based on a prescribed accuracy for a given $\mathcal{H}$-partition*

Given an $\mathcal{H}$-partition, to minimize k_{ave}, we determine a minimal rank based on a prescribed accuracy for each admissible block. The reason is obvious. If each admissible block has a minimal rank, the resultant average *partition* rank k_{ave} for the given $\mathcal{H}$ partition is also minimized.

Given an accuracy requirement, singular value decomposition (SVD) is the most accurate method to obtain the minimum rank that can meet the accuracy requirement for an admissible block. However, if we directly apply SVD to the original full matrix to obtain its $\mathcal{H}$-matrix representation, although the resultant rank is minimal, the computational cost is high.

On the other hand, we can use interpolation, the Taylor expansion, and ACA-based approaches to efficiently convert a full matrix block to an $\mathcal{H}$-matrix representation. However, the resultant rank is, in general, not the minimal one that is necessary to satisfy the accuracy requirement. In other words, the resultant rank can be much larger than what is necessary to satisfy a prescribed accuracy.

In this paper, we first use ACA+ ([16], pp. 71-74), which is a variant of ACA [19], to efficiently compute an $\mathcal{H}$-matrix representation. We then apply SVD to the $\mathcal{H}$-matrix representation to determine the actual rank that is needed to satisfy the accuracy requirement [17]. By doing so, we keep the advantages of both SVD and ACA-based methods. The resultant rank is minimal, and meanwhile it is obtained in linear complexity for each block. In the following, we give more details of the proposed approach.

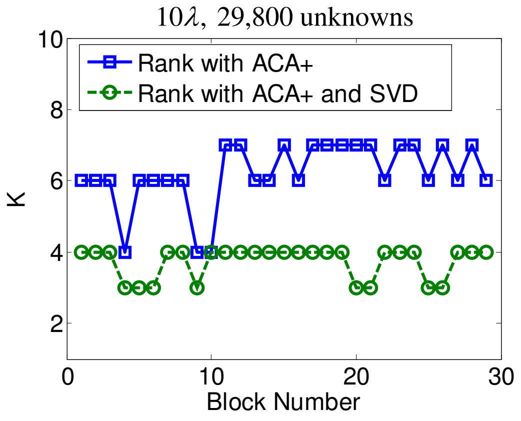
First, we use ACA+ to numerically obtain a factorized form of an admissible block. The ACA+ involves less storage and computational cost than ACA. The detailed procedure of ACA+ is very similar to that of the conventional ACA. The difference between them is as follows. At the beginning of an ACA+ algorithm, a reference row and a reference column of the original matrix are chosen to determine where to start the pivot search. A row and column pivot index is then determined from the reference ones. In the subsequent steps, the reference row and column can still be used. But if they are chosen as a pivot index, a new reference row and a new reference column has to be chosen. This method only requires assemble k rows and k columns of an admissible block, where k is the rank determined by a certain accuracy requirement ϵ. The output of an ACA+ algorithm is $\tilde{\mathbf{G}}^{m,n} = \mathbf{A}_{m,k}\mathbf{B}_{n,k}^T$, where k is, in general, much less than m and n. The ACA+ algorithm terminates when

$$\left\| \mathbf{G} - \tilde{\mathbf{G}} \right\| = \left\| \mathbf{G} - \mathbf{A}\mathbf{B}^T \right\| \leq \epsilon \left\| \mathbf{G} \right\| \tag{15}$$

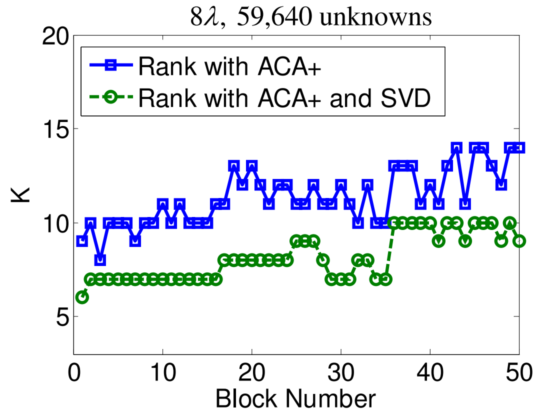
is satisfied. Therefore, the error of the resultant $\mathcal{H}$-matrix representation is bounded by ϵ.

After the ACA+ is completed, we obtain a factorized form $\mathbf{A}_{m,k}\mathbf{B}_{n,k}^T$. For such a factorized form, SVD can be efficiently performed by a reduced SVD ([16], p. 108). The resultant computational cost is $O(k^2(m+n))$, which is linear.

To test the effectiveness of the proposed approach to determining the minimal rank of an admissible block, we simulated a 3D conducting plate of 10 wavelengths, and a 3D conducting sphere of 8 wavelengths, respectively. The ϵ in ACA+ and SVD was set to be 10^{-4}. In Fig. 6, we plot the rank distributions with the proposed scheme (ACA+ and SVD) and with ACA+ only in the lowest $\mathcal{H}$-partition level that has an admissible block. This level is the closest to the root of a block cluster tree, and hence the admissible blocks therein have the largest electric size. The horizontal axis of Fig. 6 is the admissible block index. It can be seen from Fig. 6 that by using the proposed method, the rank in most admissible blocks is reduced by half. Moreover, the same accuracy is achieved. The accuracy is measured by $\left\| \mathbf{G} - \tilde{\mathbf{G}} \right\| / \left\| \mathbf{G} \right\|$, where a Frobenius norm is used. The accuracy is 1.021689×10^{-5} without the proposed rank minimization, and 1.079537×10^{-5} with the proposed minimization for the plate example. Clearly, the rank is reduced without sacrificing accuracy. The same is true for the sphere example. The accuracy is 1.931219×10^{-5} without the proposed rank minimization, and 1.996355×10^{-5} with the proposed minimization. In addition, with the proposed method, the new rank distribution becomes more uniform across all the admissible blocks as can be seen from Fig. 6.



(a)



(b)

Fig. 6: Rank distribution in the lowest level of an $\mathcal{H}$-partition generated with $\epsilon = 10^{-4}$. (a) A 10 λ perfect electrically conducting (PEC) plate.(b) An 8 λ PEC sphere.

B. Proposed method for optimizing $\mathcal{H}$-matrix partition to minimize the number of admissible blocks at each tree level

Existing $\mathcal{H}$-matrix partition is based on the admissibility condition as shown in (5). It is not an optimal one for frequency dependent kernels. This can also be understood from the fact that the admissibility condition (5) is frequency independent. Physically speaking, whether the interaction between two regions can be represented by a low-rank block or not is frequency dependent. For example, for a given accuracy, it is possible that an off-diagonal block that is inadmissible at certain frequencies becomes admissible at a lower frequency. It is also possible that multiple small admissible blocks can be merged into a single admissible block with prescribed accuracy. In other words, they start to become admissible in a lower level of an inverted block cluster tree when the frequency changes. However, the admissibility condition given in (5) is empirical instead of theoretical. It is controlled by an empirical parameter η, instead of a prescribed accuracy.

In the following, we show our proposed algorithm that can optimize an $\mathcal{H}$-matrix partition based on a prescribed accuracy. The resultant $\mathcal{H}$-matrix partition is frequency dependent. It significantly reduces the number of admissible blocks at each tree level, with the prescribed accuracy satisfied.

The proposed $\mathcal{H}$-matrix partition algorithm for a prescribed accuracy ϵ_{opt} is shown in (16). A new error tolerance ϵ_{opt} is introduced instead of reusing ϵ in (15) to facilitate separated accuracy control of the $\mathcal{H}$-partition optimization and the rank minimization for each admissible block. In (16), the original partition given by (5) is used as an initial guess from which we construct an optimized partition.

```
$\mathcal{H}$-Partition Optimization
Procedure $\mathcal{H} - \mathcal{P}_{opt}(\mathcal{P}, \epsilon_{opt})$
    (Input $\mathcal{P}$ is the original $\mathcal{H}$ partition,
    output $\mathcal{P}$ is overwritten by an optimized $\mathcal{H}$ partition)
      If $\mathcal{P}$ is a non-leaf off-diagonal matrix block
        for $(i = 0; i < 4; i++)$
          if $\mathcal{P}(i)$ is an inadmissible block
            Rk_Factor$(\mathcal{P}(i), \epsilon_{opt})$
          end if
          if $\mathcal{P}(i)$ is a non-leaf block
            $\mathcal{H} - \mathcal{P}_{opt}(\mathcal{P}(i), \epsilon_{opt})$
          end if
        end for
        If all blocks in $\mathcal{P}$ are admissible blocks
          Merge_Rkblocks $(\mathcal{P}, \epsilon_{opt})$
        end if
      end if                                                    (16)
```

In (16), the function **Rk_Factor** is to factorize a full-matrix block to a rank-k matrix shown in (6) based on a prescribed accuracy. The procedure is the combined ACA+ with SVD, which is detailed in section IV-A. The function **Merge_Rkblocks** is to merge multiple small admissible blocks to a single one based on the prescribed accuracy. To give an example, four admissible sub-blocks can be merged into one admissible block as follows.

$$\begin{bmatrix} \mathbf{A}_1(f_1)\mathbf{B}_1(f_1)^T & \mathbf{A}_2(f_2)\mathbf{B}_2(f_2)^T \\ \mathbf{A}_3(f_3)\mathbf{B}_3(f_3)^T & \mathbf{A}_4(f_4)\mathbf{B}_4(f_4)^T \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{A}_1(f_1) \\ 0 \end{bmatrix}\begin{bmatrix} \mathbf{B}_1(f_1) \\ 0 \end{bmatrix}^T + \begin{bmatrix} \mathbf{A}_2(f_2) \\ 0 \end{bmatrix}\begin{bmatrix} 0 \\ \mathbf{B}_2(f_2) \end{bmatrix}^T +$$

$$\begin{bmatrix} 0 \\ \mathbf{A}_3(f_3) \end{bmatrix}\begin{bmatrix} \mathbf{B}_3(f_3) \\ 0 \end{bmatrix}^T + \begin{bmatrix} 0 \\ \mathbf{A}_4(f_4) \end{bmatrix}\begin{bmatrix} 0 \\ \mathbf{B}_4(f_4) \end{bmatrix}^T =$$

$$\tilde{\mathbf{A}}_1(f_1)\tilde{\mathbf{B}}_1(f_1)^T + \tilde{\mathbf{A}}_2(f_2)\tilde{\mathbf{B}}_2(f_2)^T + \tilde{\mathbf{A}}_3(f_3)\tilde{\mathbf{B}}_3(f_3)^T +$$

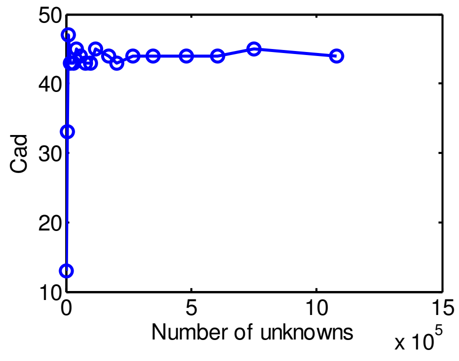
$$\tilde{\mathbf{A}}_4(f_4)\tilde{\mathbf{B}}_4(f_4)^T \overset{\epsilon_{opt}}{=} \mathbf{A}\mathbf{B}^T, \quad (17)$$

where the $f_i (i = 1, 2, 3, 4)$ denotes the electric size the blocks are associated with, and the addition in the final step is carried out by the truncated addition operation in ([16], p. 110), with the new rank k determined based on the accuracy requirement ϵ_{opt}. Different from static problems, in electrodynamic problems, when the size of the admissible block increases, its rank also increases in general. Consequently, although each merging operation reduces four admissible blocks to one block, it also increases the rank of the resultant block. Therefore, we should check which one is computationally more efficient: merging or not merging. This can be assessed by comparing the storage requirement or computational operations of the merged block with that of the four children admissible blocks. If the former is less than the latter, we perform merging; otherwise, we do not perform merging, instead we keep the original four blocks. To be more specific, we check whether $k_{merge}(m_{merge} + n_{merge}) \leq \sum_{j=1}^{4} k_j(m_j + n_j)$ is satisfied or not, where k_{merge} is the rank of the big block resulting from the merging operation, $m_{merge}(n_{merge})$ is the row(column) dimension of the block, and k_j is the rank of the children admissible blocks. If it is satisfied, we merge blocks based on accuracy requirements; if not, we keep the original blocks. By doing so, the cost of storage and computation based on the resultant new $\mathcal{H}$ matrix is minimized for the prescribed accuracy at each frequency.

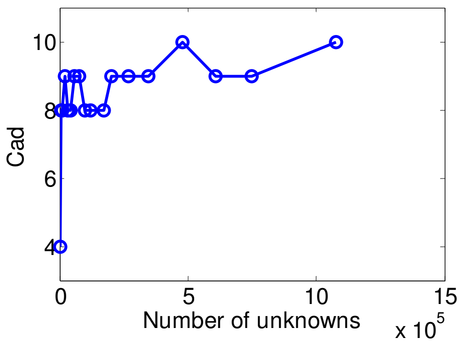
In (16), we only perform two basic operations: making an inadmissible block in the off-diagonal part admissible based on a prescribed accuracy via function **Rk_Factor**, or merging small admissible blocks to be a big admissible block based on the accuracy requirement through function **Merge_Rkblocks**. Both operations do not increase the number of inadmissible blocks. In fact, the number of inadmissible blocks is even reduced due to the first operation. Therefore, given an $\mathcal{H}$-partition, the proposed optimization algorithm does not increase the number of inadmissible blocks. Thus, if the number of admissible blocks is reduced, the total number of blocks is also reduced.

To validate the effectiveness of the proposed $\mathcal{H}$-partition optimization algorithm, we simulated a conducting plate from 2λ to 60λ, the number of unknowns of which is from 1,160 to 1,078,800. The ϵ_{opt} was chosen as 10^{-3}. In Fig. 7, we plot the maximum number of admissible blocks that can be formed by one cluster in a block cluster tree (C_{ad}) in the original $\mathcal{H}$-partition, and that in the optimized $\mathcal{H}$-partition. Clearly, the C_{ad} is reduced significantly. Since the proposed $\mathcal{H}$-partition optimization does not increase the number of inadmissible blocks as analyzed above, the C_{sp} is also reduced significantly.

When the $\mathcal{H}$-partition changes, the average *partition* rank k_{ave} also changes. Thus, in addition to C_{sp}, we need to examine the product of k_{ave} and C_{sp} to assess the success of the proposed $\mathcal{H}$-partition optimization algorithm, since it is

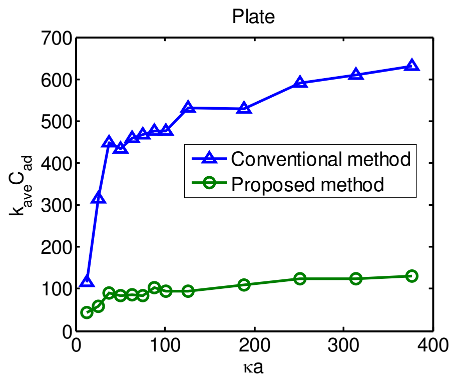


(a)

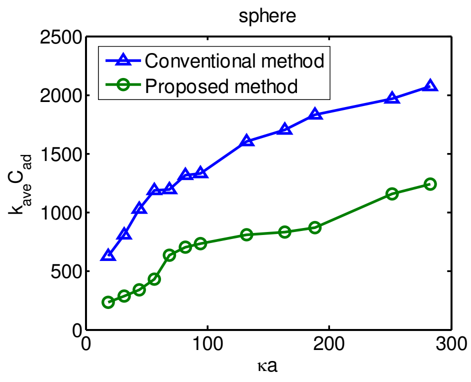


(b)

Fig. 7: C_{ad} versus N in the simulation of a conducting plate from 2λ to 60λ. (a) Original $\mathcal{H}$-partition. (b) Optimized $\mathcal{H}$-partition.



(a)



(b)

Fig. 8: Comparison between $k_{ave}C_{sp}$ generated by conventional method and that generated by the proposed method. (a) A PEC plate from 2 λ to 60 λ. (b) A sphere from 2 λ to 45 λ.

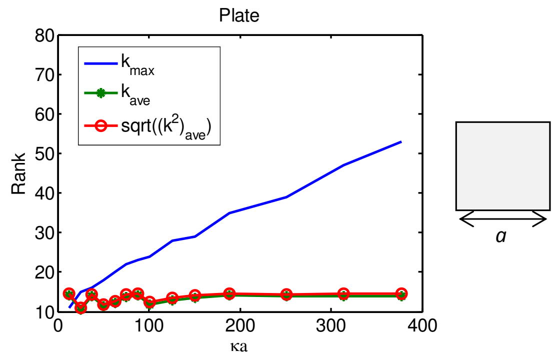
$k_{ave}C_{sp}$ that determines the storage and time cost as can be seen from (13) and (14). In Fig. 8(a), we plot $k_{ave}C_{ad}$ for the plate example simulated above with respect to κa, where κ is the wave number, and a is the side length of the plate. We compare the $k_{ave}C_{ad}$ generated by the proposed method, and that generated by the conventional method which is based on an ACA-based rank scheme and an admissibility condition based $\mathcal{H}$-partition. It is clear that the proposed method greatly reduces $k_{ave}C_{ad}$. In Fig. 8(b), we plot $k_{ave}C_{ad}$ for a sphere of diameter a with respect to κa from 2 λ to 45 λ. Again, $k_{ave}C_{ad}$ is greatly reduced by the proposed method. Since the proposed $\mathcal{H}$-partition optimization does not increase the number of inadmissible blocks as analyzed above, the $k_{ave}C_{sp}$ is also reduced significantly.

Although C_{sp} is a parameter that can be used to qualitatively measure the number of blocks formed by an $\mathcal{H}$-partition, it does not give a *quantitative* measurement of the number of blocks obtained by the $\mathcal{H}$-partition. To provide a quantitative analysis, in Fig. 9(a), we plot the exact number of blocks with respect to tree level generated by the original $\mathcal{H}$-partition which is based on the admissibility condition in comparison with the number of blocks generated by the proposed partition, for a 15λ sphere. In Fig. 9(b), we plot the same for a 40λ plate. Clearly, compared to the original partition, the proposed partition significantly reduces the number of blocks at each tree level, and hence significantly reducing the computational cost. In addition, across the entire tree depth, it is observed that the number of blocks is not necessarily reduced by half when one ascends the inverted tree by one level from leaf level $l = p$ to root level $l = 0$, which can be seen from both the conventional partition and the proposed partition.

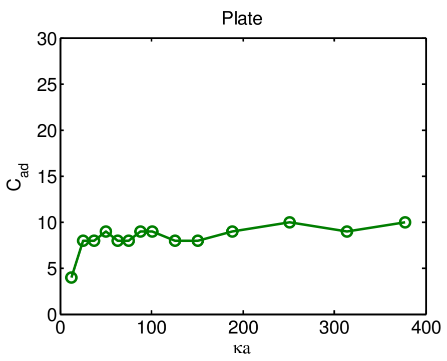
C. Study on the dependence of k_{ave} and C_{sp} of the proposed methods with respect to scatterer shape

We varied the scattering shape continuously from plate, Sierpiski gasket, cylinder, open cone, cone sphere, to sphere. We plotted the maximal rank k_{max} versus electric size for these scatterers. For comparison, we also plotted the average *partition* rank k_{ave} obtained by the proposed methods for the same accuracy. As can be seen from the figures in the left column of Fig. 10 and Fig. 11, for all these scatterers, k_{ave} is much smaller compared to k_{max}. In addition, the rate of the change of k_{ave} with respect to electric size is much slower than that of k_{max}. Theoretically speaking, this is because k_{ave} is not only determined by the rank of each admissible block, but also determined by the number of admissible blocks at each tree level, and hence the $\mathcal{H}$-partition.

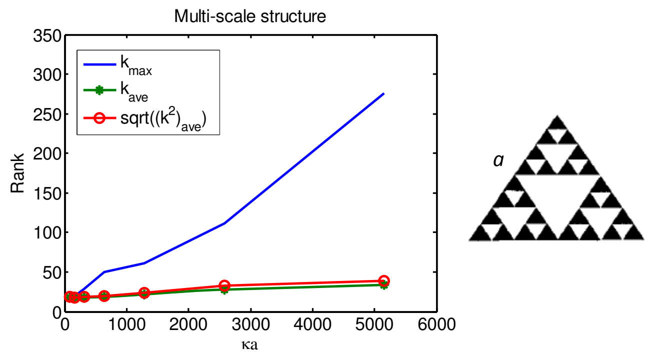
In the right column of Fig. 10 and Fig. 11, we plot the number of admissible blocks obtained by the proposed method with respect to electric size for a variety of scatter shapes. The dependence of the C_{sp} on the scatter shape is shown to be



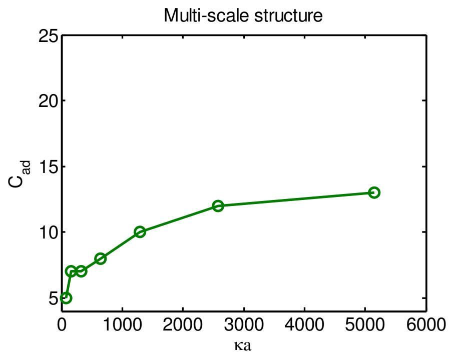
(a)



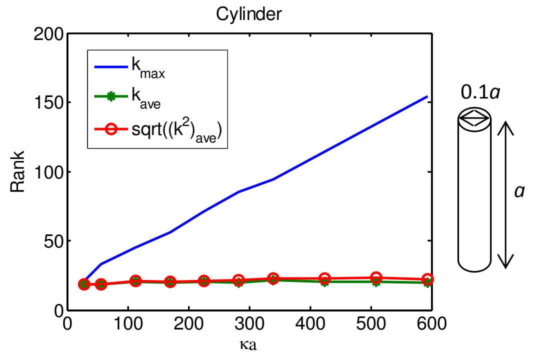
(b)



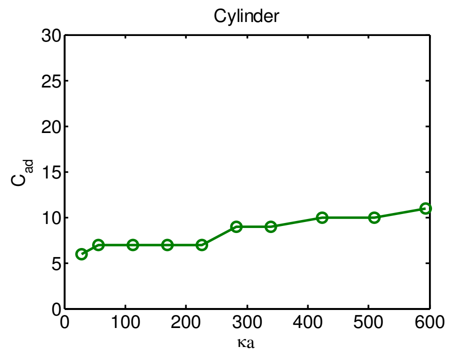
(c)



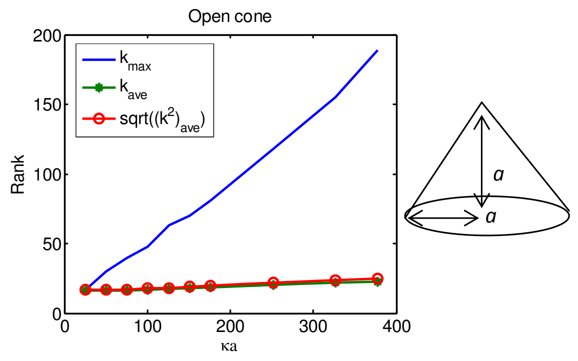
(d)



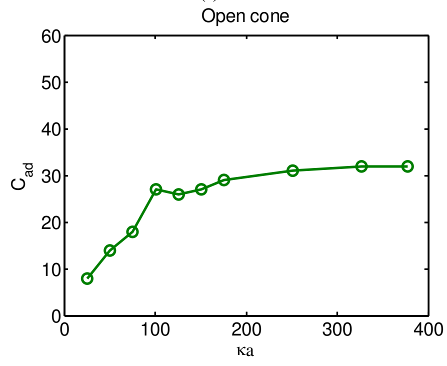
(e)



(f)



(g)



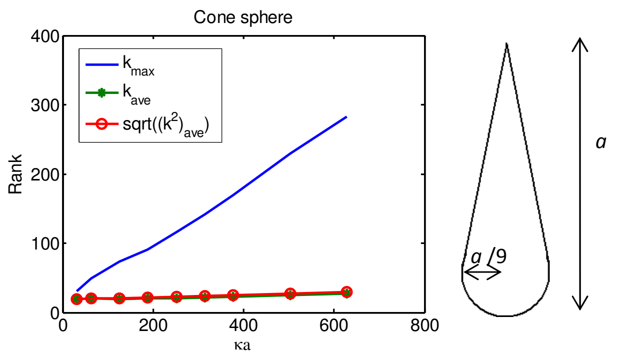
(h)

Fig. 10: The k_{ave} and C_{sp} generated by the proposed method with respect to electric size for a variety of scatterer shapes. (a-b) Plate. (c-d) Sierpiski gasket. (e-f) Cylinder. (g-h) Open cone.

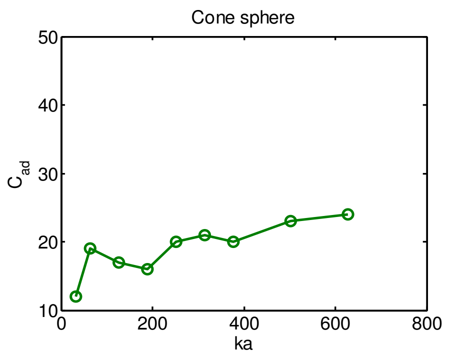
little. In addition, with proposed methods, both k_{ave} and C_{sp} are minimized to be small compared to N.

D. Study on the dependence of k_{ave} and C_{sp} on accuracy requirements

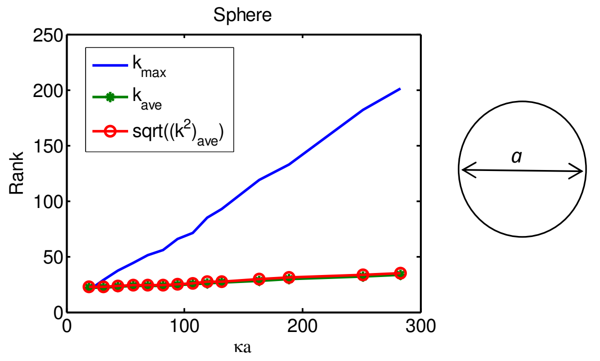
We also tested the dependence of k_{ave} and C_{sp} with respect to accuracy. The results in Fig. 10 and Fig. 11 were



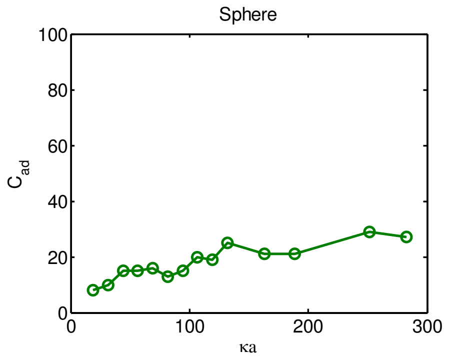
(a)



(b)



(c)



(d)

Fig. 11: The k_{ave} and C_{sp} generated by the proposed method with respect to electric size for a variety of scatterer shapes (Continued from Fig. 10). (a-b) Cone sphere. (c-d) Sphere.

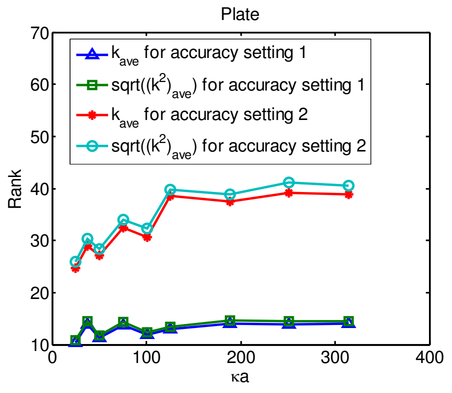
generated based on $\epsilon = 10^{-4}$ and $\epsilon_{opt} = 10^{-3}$, where ϵ is a parameter shown in (15) for controlling the accuracy of rank reduction, and ϵ_{opt} is a parameter shown in (16) for controlling the accuracy of the $\mathcal{H}$-partition optimization. To test the dependence with respect to accuracy, we set $\epsilon = 10^{-7}$ and $\epsilon_{opt} = 10^{-6}$. We used a plate and a sphere as examples, and tested the dependence of k_{ave} and C_{sp} generated by the proposed methods with respect to accuracy across a wide range of electric sizes. As can be seen from Fig. 12, with the accuracy requirement increased, k_{ave} and $\sqrt{(k^2)_{ave}}$ increase. However, the increase is small compared to the three orders of magnitude increase in accuracy. In addition, the dependence of k_{ave} and $\sqrt{(k^2)_{ave}}$ with respect to electric size is similar to what is observed for a lower order of accuracy. In addition, with the accuracy requirement increased, the number of admissible blocks also increases. Again, the increase is minor compared to the three orders of magnitude increase in accuracy. Moreover, the frequency dependence of C_{sp} is also similar to that for a lower order of accuracy.

V. On the Existence of the $\mathcal{H}$-Matrix Representation of $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU Factors

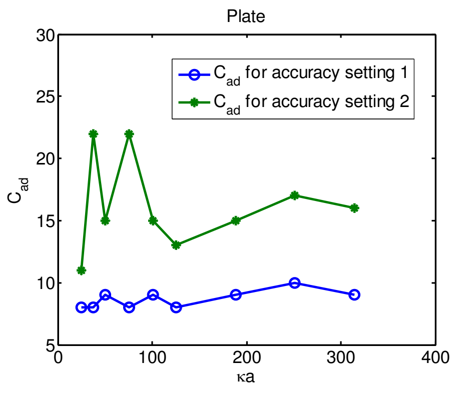
In this section, we numerically prove the existence of an $\mathcal{H}$-matrix representation of the inverse of $\mathbf{G}$ and $\mathbf{G}$'s LU factors by examining their rank distributions. The detailed procedure is as follows: we directly compute $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU factors without introducing any approximation; we then use SVD to obtain the rank distribution in $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU factors. Since the computation requires a complete full-matrix form of $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU factors, it is not practical to use a problem having a large electric size as an example. We thus considered a conducting plate of 6λ and a conducting sphere of 4λ. We computed the rank of each off-diagonal block that satisfies the admissibility condition (5) at the lowest level of a block cluster tree. The accuracy requirement in SVD is set to be 10^{-4}. In Fig. 13, we plot the rank distribution measured by $k/\min(m,n)$ with respect to the matrix block index. It is clear that the off-diagonal blocks that satisfy the admissibility condition in $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU factors are low rank. Therefore, not only the original matrix $\mathbf{G}$ can be represented by an $\mathcal{H}$-matrix, but also $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU factors can be represented by an $\mathcal{H}$-matrix. Furthermore, the rank distribution of $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU factors is very similar to that of $\mathbf{G}$. In addition, both $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU factors have a smaller rank than the original matrix, with the rank of $\mathbf{G}$'s LU factors being the smallest. This suggests that the $\mathcal{H}$-matrix partition constructed for the original matrix $\mathbf{G}$ is equally applicable to $\mathbf{G}^{-1}$ and $\mathbf{G}$'s LU factors. In Fig. 13, $\mathbf{U}$'s rank distribution is plotted to represent $\mathbf{L}$'s and $\mathbf{U}$'s rank distribution since these two have a very similar rank distribution, with $\mathbf{U}$'s rank being slightly larger.

VI. Proposed Fast Implementation of LU Factorization

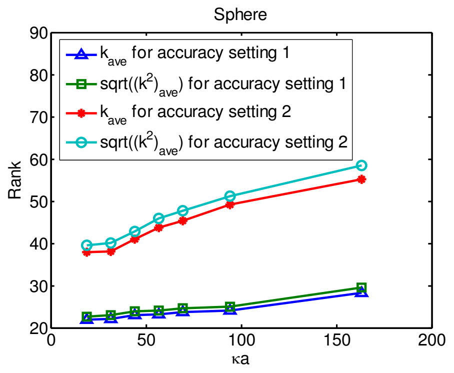
In this section, we show how to perform a fast LU factorization using the $\mathcal{H}$-matrix based representation of $\mathbf{G}$ and $\mathbf{G}$'s LU factors. Based on the findings in the section above, we



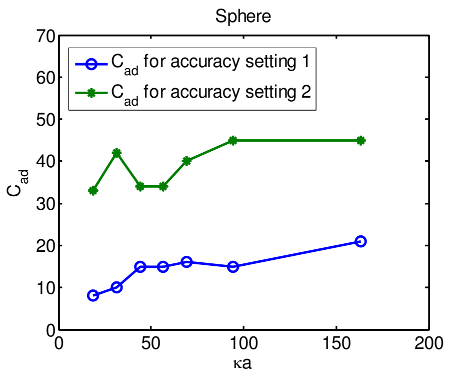
(a)



(b)



(c)



(d)

Fig. 12: The dependence of k_{ave} and C_{sp} generated by the proposed method with respect to accuracy requirements over a wide range of electric sizes (Accuracy setting 1: $\epsilon = 10^{-4}$ and $\epsilon_{opt} = 10^{-3}$; Accuracy setting 2: $\epsilon = 10^{-7}$ and $\epsilon_{opt} = 10^{-6}$). (a-b) Plate. (c-d) Sphere.

use the $\mathcal{H}$-partition constructed for **G** for the $\mathcal{H}$-partition of **L** and **U**.

The $\mathcal{H}$-based LU factorization has been discussed in ([16], p. 119). However, no detailed implementation is given. In the following, we give a number of pseudo-codes to show a fast implementation of $\mathcal{H}$-based LU factorization, which is not reported anywhere else. It is worth mentioning that [6] did an ACA-based LU factorization, the procedure of which is very different from the proposed one.

A. LU factorization basics

Given an IE-based system matrix **G**, we cast it into a form

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{bmatrix}. \quad (18)$$

The LU decomposition can be recursively computed by the following equation:

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{11} & 0 \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} \\ 0 & \mathbf{U}_{22} \end{bmatrix} = \mathbf{LU}. \quad (19)$$

B. Proposed fast implementation of the LU factorization

We developed a pseudo-code shown in (20) to recursively perform LU factorization.

```
LU-Decomposition G=LU
Procedure $\mathcal{H}$-LU(G)
    (G is the input matrix overwritten by L and U)
      If G is a non-leaf block
        $\mathcal{H}-\mathbf{LU}(\mathbf{G}_{11}) \rightarrow \mathbf{L}_{11}, \mathbf{U}_{11}$,
        Solve_LX($\mathbf{L}_{11}, \mathbf{G}_{12}$) $\rightarrow \mathbf{U}_{12}$,
        Solve_XU($\mathbf{G}_{21}, \mathbf{U}_{11}$) $\rightarrow \mathbf{L}_{21}$,
        $-\mathbf{L}_{21} \times \mathbf{U}_{12} + \mathbf{G}_{22} \rightarrow \mathbf{G}_{22}$,
        $\mathcal{H}-\mathbf{LU}(\mathbf{G}_{22}) \rightarrow \mathbf{L}_{22}, \mathbf{U}_{22}$,
      else
        Full-LU(G)                                        (20)
```

The underlying algorithm is as follows. When **G** is a non-leaf matrix block, we recursively call (20) until $\mathbf{G}_{11}$ is a full matrix block. We then directly compute the LU factors of the $\mathbf{G}_{11}$ using a full-matrix-based LU factorization, which generates $\mathbf{L}_{11}$ and $\mathbf{U}_{11}$. Next, we call function **Solve_LX** shown in (21) and **Solve_XU** to compute $\mathbf{U}_{12}$, and $\mathbf{L}_{21}$ respectively.

Algorithm for Solving a Lower Triangular System
LX = G, with G being an $\mathcal{H}$ matrix

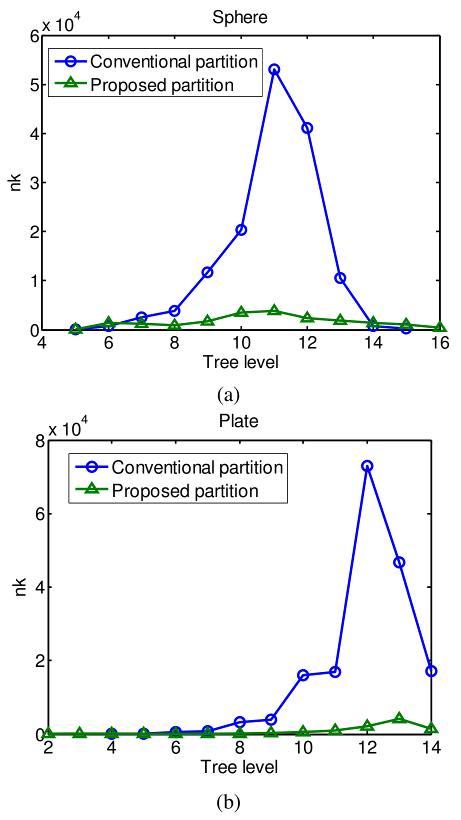


Fig. 9: Comparison between the number of blocks generated by the conventional geometry-based partition and that by the proposed optimal partition with respect to tree level. (a) A 15λ sphere. (b) A 40λ sphere.

```
Procedure Solve_LX(L,G)
  ($\mathbf{L}$ and $\mathbf{G}$ are input matrices, $\mathbf{G}$ is overwritten by $\mathbf{X}$)
    If $\mathbf{L}$ is a non-leaf block
      If $\mathbf{G}$ is a non-leaf block
        $\mathbf{Solve\_LX}(\mathbf{L}_{11}, \mathbf{G}_{11})$, $\mathbf{Solve\_LX}(\mathbf{L}_{11}, \mathbf{G}_{12})$
        $-\mathbf{L}_{21} \times \mathbf{G}_{11} + \mathbf{G}_{21} \rightarrow \mathbf{G}_{21}$, $\mathbf{Solve\_LX}(\mathbf{L}_{22}, \mathbf{G}_{21})$
        $-\mathbf{L}_{21} \times \mathbf{G}_{12} + \mathbf{G}_{22} \rightarrow \mathbf{G}_{22}$, $\mathbf{Solve\_LX}(\mathbf{L}_{22}, \mathbf{G}_{22})$
      else if $\mathbf{G}$ is an admissible block
        $\mathbf{Solve\_LF}(\mathbf{L}, \mathbf{A})$
      else
        $\mathbf{Solve\_LF}(\mathbf{L}, \mathbf{G})$
      end if
    else
      $\mathbf{Full\_LX}(\mathbf{L}, \mathbf{G})$
      (Solve a full-matrix triangular system)
    end if                                                    (21)
```

The **Solve_LX(L,G)** is to solve a lower triangular system $\mathbf{LX} = \mathbf{G}$, where $\mathbf{L}$ and $\mathbf{G}$ are input matrices having $\mathcal{H}$-representations, and $\mathbf{X}$ is the solution. The **Solve_XU(G, U)** is to solve an upper triangular system, which can be derived in a similar fashion as (21). In (21), a function **Solve_LF** is called. Similar to **Solve_LX**, **Solve_LF** also solves a triangular system. The difference is that the right-hand-side matrix for **Solve_LX** is an $\mathcal{H}$ matrix, whereas that for **Solve_LF** is a full matrix. The pseudo-code of **Solve_LF** is given in (22).

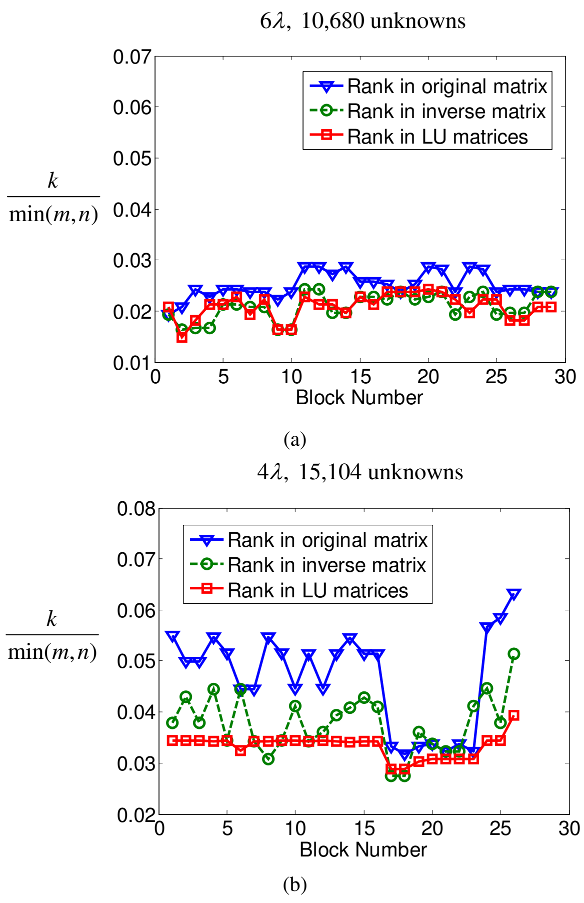


Fig. 13: Rank distributions in $\mathbf{G}$, $\mathbf{G}^{-1}$, and $\mathbf{G}$'s LU factors. (a) PEC plate. (b) PEC sphere.

Algorithm for Solving a Lower Triangular System $\mathbf{LX} = \mathbf{F}$, with F being a Full Matrix

```
Procedure Solve_LF(L,F)
  ($\mathbf{L}$ and $\mathbf{F}$ are input matrices, $\mathbf{F}$ is overwritten by $\mathbf{X}$)
    If $\mathbf{L}$ is a non-leaf block
      $\mathbf{Solve\_LF}(\mathbf{L}_{11}, \mathbf{F}_1)$
      $-\mathbf{L}_{21} \times \mathbf{F}_1 + \mathbf{F}_2 \rightarrow \mathbf{F}_2$
      $\mathbf{Solve\_LF}(\mathbf{L}_{22}, \mathbf{F}_2)$
    else
      $\mathbf{Full\_LX}(\mathbf{L}, \mathbf{F})$
      (Solve a full-matrix triangular system)
    end if                                                    (22)
```

In the final step of (20), we use $\mathbf{U}_{12}$ and $\mathbf{L}_{21}$ to update $\mathbf{G}_{22}$, and then call (20) recursively until $\mathbf{L}_{22}$ and $\mathbf{U}_{22}$ are computed. As can be seen from (19) to (22), efficient LU factorization relies on efficient block multiplication and block addition. In next subsection, we show how to efficiently perform these two operations for a prescribed accuracy.

C. Fast implementation of the block multiplication $\mathbf{G}^b = \mathbf{G}^{b1} \times \mathbf{G}^{b2}$

We give a pseudo-code of computing $\mathbf{G}^b = \mathbf{G}^{b1} \times \mathbf{G}^{b2}$ in (23).

```
Recursive Multiplication Algorithm
Procedure $\mathcal{H}$-mult($\mathbf{G}^{b1}, \mathbf{G}^{b2}, \mathbf{G}^{b}, \epsilon_{LU}$)
    If ($\mathbf{G}^{b1}, \mathbf{G}^{b2}, \mathbf{G}^{b}$ are all non-leaf blocks)
      $for(i = 0; i < 2; i++)$
        $for(j = 0; j < 2; j++)$
          $for(k = 0; k < 2; k++)$
            $\mathcal{H}$-mult($\mathbf{G}^{b1}(i,k), \mathbf{G}^{b2}(k,j), \mathbf{G}^{b}(i,j)$)
    else if($\mathbf{G}^{b}$ is a non-leaf block,
    $\mathbf{G}^{b1}$ or $\mathbf{G}^{b2}$ is a leaf block)
      Multiply_RK($\mathbf{G}^{b1}, \mathbf{G}^{b2}, \tilde{\mathbf{G}}^{b}$)
      $\mathbf{G}^{b} \overset{\epsilon_{LU}}{=} \tilde{\mathbf{G}}^{b} + \mathbf{G}^{b}$
    else if $\mathbf{G}^{b}$ is an admissible block
      Multiply_RK($\mathbf{G}^{b1}, \mathbf{G}^{b2}, \mathbf{G}^{b}$)
    else if $\mathbf{G}^{b}$ is an inadmissible block
      Multiply_Full($\mathbf{G}^{b1}, \mathbf{G}^{b2}, \mathbf{G}^{b}$)
    end if                                                    (23)
```

where, b, b_1, and b_2 represent three blocks in the same level of an $\mathcal{H}$-partition, ϵ_{LU} represents a prescribed accuracy. If $\mathbf{G}^b$, $\mathbf{G}^{b1}$, and $\mathbf{G}^{b2}$ are all non-leaf blocks, we recursively call (23). If one of $\mathbf{G}^{b1}$ and $\mathbf{G}^{b2}$ is a leaf block, or $\mathbf{G}^b$ is an admissible block, we call function **Multiply_Rk** shown in (24) to compute an admissible product. In (23), the addition is performed based on the prescribed accuracy ϵ_{LU}, which is denoted by $\overset{\epsilon_{LU}}{=}$. The detailed procedure of the addition is given in the following subsection.

```
Procedure Multiply_RK($\mathbf{G}^{b1}, \mathbf{G}^{b2}, \mathbf{G}^{b}, \epsilon_{LU}$)
  if $\mathbf{G}^{b1}$ and $\mathbf{G}^{b2}$ are both non-leaf blocks
    $for(i = 0; i < 2; i++)$
      $for(j = 0; j < 2; j++)$
        $for(k = 0; k < 2; k++)$
          Multiply_RK($\mathbf{G}^{b1}(i,k), \mathbf{G}^{b2}(k,j), \tilde{\mathbf{G}}^{b}(i,j)$)
          $\mathbf{G}^{b} \overset{\epsilon_{LU}}{=} \tilde{\mathbf{G}}^{b} + \mathbf{G}^{b}$
          ($\tilde{\mathbf{G}}^{b}$ is a non-leaf block)
  else if $\mathbf{G}^{b1}$ or $\mathbf{G}^{b2}$ is an admissible block
    $\mathbf{G}^{b1}\mathbf{A}\mathbf{B}^T \rightarrow (\mathbf{G}^{b1}\mathbf{A})\mathbf{B}^T = \tilde{\mathbf{A}}_b\tilde{\mathbf{B}}^T = \tilde{\mathbf{G}}^{b}$
    ($\tilde{\mathbf{G}}^{b}$ is an admissible block)
    $\mathbf{G}^{b} \overset{\epsilon_{LU}}{=} \tilde{\mathbf{G}}^{b} + \mathbf{G}^{b}$
  else if $\mathbf{G}^{b1}$ or $\mathbf{G}^{b2}$ is an inadmissible block
    $\mathbf{G}^{b1}\mathbf{G}^{b2} = \mathbf{G}^{b1}\mathbf{F} \overset{\epsilon_{LU}}{\rightarrow} (\mathbf{G}^{b1}\mathbf{A})\mathbf{B}^T = \tilde{\mathbf{A}}_b\tilde{\mathbf{B}}^T = \tilde{\mathbf{G}}^{b}$
    $\mathbf{G}^{b} \overset{\epsilon_{LU}}{=} \tilde{\mathbf{G}}^{b} + \mathbf{G}^{b}$
  end if                                                    (24)
```

In (24), there are two multiplication cases. One is to multiply an admissible block $\mathbf{G}^{b1}$ by an admissible block of an $\mathbf{AB}^T$ form, for which we can compute $\mathbf{G}^{b1}\mathbf{A}$ as a new $\mathbf{A}$. The other multiplication case is to multiply $\mathbf{G}^{b1}$ by a full matrix block $\mathbf{F}$, for which we can first apply SVD to $\mathbf{F}$ to generate a form $\mathbf{AB}^T$ based on the prescribed accuracy ϵ_{LU}. If $\mathbf{G}^b$ is a full matrix block, a normal full matrix multiplication is computed. The additions in (24) again are performed based on ϵ_{LU}.

D. Fast implementation of the block addition $\mathbf{G}^b = \mathbf{G}^{b1} + \mathbf{G}^{b2}$

Two cases are involved in the addition operations.

Case 1: If $\mathbf{G}^b$, $\mathbf{G}^{b1}$, and $\mathbf{G}^{b2}$ have the same $\mathcal{H}$-partition, the addition can be done using the following procedure. (a) If three blocks are all full matrices, we simply add two full matrices up. (b) If three blocks are all admissible matrices, for example, $\mathbf{G}^{b1} = \mathbf{A}_{b1}\mathbf{B}_{b1}^T$ with rank k_1, $\mathbf{G}^{b2} = \mathbf{A}_{b2}\mathbf{B}_{b2}^T$ with rank k_2, and $\mathbf{G}^b = \mathbf{A}_b\mathbf{B}_b^T$, the $\mathbf{G}^b = \mathbf{G}^{b1} + \mathbf{G}^{b2}$ can be realized by a truncated addition operation using the approach shown in ([16], p. 110). The rank k of the resultant $\mathbf{G}^b$ is adaptively determined by the prescribed accuracy ϵ_{LU}. (c) If three blocks are all non-leaf blocks, the addition can be carried out by summing over all the inadmissible blocks using (a), and all the admissible ones using (b) respectively.

Case 2: If the three blocks do not share the same partition, we convert the $\mathcal{H}$-matrix partitions of $\mathbf{G}^{b1}$ and $\mathbf{G}^{b2}$ both into the partition of $\mathbf{G}^b$. Take the block $\mathbf{G}^{b1}$ as an example. If $\mathbf{G}^{b1}$ is an admissible block but $\mathbf{G}^b$ is a non-leaf block that has four admissible subblocks, we convert $\mathbf{G}^{b1}$ by the following formula

$$\mathbf{G}^{b1} = \mathbf{A}_{b1}\mathbf{B}_{b1}^T = \begin{bmatrix} \tilde{\mathbf{A}}_1 \\ \tilde{\mathbf{A}}_2 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{B}}_1 \\ \tilde{\mathbf{B}}_2 \end{bmatrix}^T = \begin{bmatrix} \tilde{\mathbf{A}}_1\tilde{\mathbf{B}}_1^T & \tilde{\mathbf{A}}_1\tilde{\mathbf{B}}_2^T \\ \tilde{\mathbf{A}}_2\tilde{\mathbf{B}}_1^T & \tilde{\mathbf{A}}_2\tilde{\mathbf{B}}_2^T \end{bmatrix} = \tilde{\mathbf{G}}^{b1}. \tag{25}$$

where $\tilde{\mathbf{G}}^{b1}$ contains four admissible sub-blocks, which is exactly equal to $\mathbf{G}^{b1}$. The opposite procedure, where $\mathbf{G}^b$ is an admissible block while $\mathbf{G}^{b1}$ contains four admissible sub-blocks, can be performed by the scheme shown in (17).

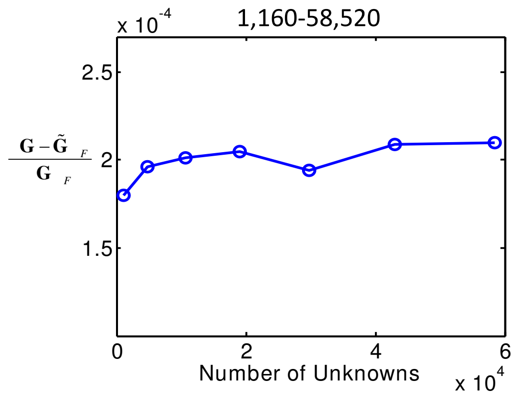
VII. TOTAL COMPUTATIONAL COST ANALYSIS

Three numerical procedures are involved in the proposed direct IE solver: rank minimization, $\mathcal{H}$-partition optimization, and LU-based direct matrix solution. We analyze the computational cost of each in the following.

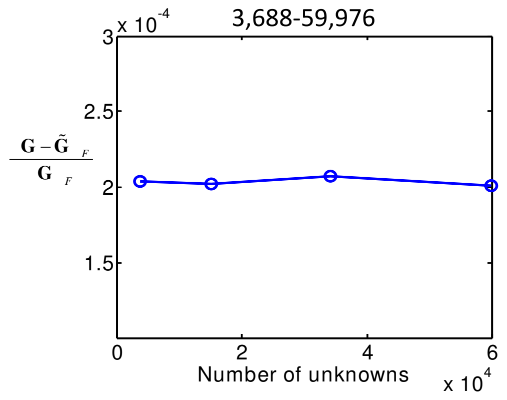
The cost of the rank minimization scheme for *each admissible block*, described in section IV-A, is linear. This is because both ACA+ and reduced SVD have a linear cost for each admissible block [16], [19].

For the $\mathcal{H}$-partition optimization shown in (16), two basic operations are involved. One is the factorization of inadmissible blocks. This operation is carried out by the function **Rk_Factor** based on ACA+ and SVD. The other is the conversion of non-leaf blocks into an admissible block shown in (17). This operation is carried out by the function **Merge_Rkblocks** by using an SVD based truncated addition. Since both ACA+ and SVD have a linear cost for each matrix block, the two basic operations involved in (16) also have a linear cost for each matrix block.

From procedure (20), it can be seen that at the leaf level, the computation of the recursive LU factorization essentially includes a full-matrix LU factorization, a full-matrix solution of a lower triangular system, and a full-matrix solution of an upper triangular system, all of which have the same computational cost as a full-matrix block multiplication. At all the other levels, a number of block-block multiplications are computed, which have the same recursive pattern as that in an $\mathcal{H}$-based matrix-matrix multiplication. Therefore, the $\mathcal{H}$-based LU factorization has the same cost as $\mathcal{H}$-based multiplication, which is shown in (14). The $\mathcal{H}$-based LU solution has the same cost as the $\mathcal{H}$-based matrix-vector multiplication, and hence storage, which is derived in (13). The proposed methods minimize $k_{ave}C_{sp}$, and hence k_{ave} and nk_l based on accuracy to reduce the computational cost.



(a)



(b)

Fig. 14: Accuracy of the proposed $\mathcal{H}$-matrix representation. (a) PEC plate from 2 λ to 14 λ. (b) PEC sphere from 2 λ to 8 λ.

VIII. NUMERICAL RESULTS

To test the performance of the proposed direct IE solver, we simulated a PEC (perfect electrically conducting) plate, a PEC sphere, and a PEC cylinder from a small number of unknowns to over 1 million unknowns, from small electric sizes to over 95 wavelengths. In all these examples, $\eta = 1$ and *leafsize* $= 32$ were used. Note that $\eta = 1$ was used to generate an initial $\mathcal{H}$-partition, which was later replaced by the optimized $\mathcal{H}$-partition by the method proposed in Section IV-B. The error tolerance ϵ_{opt} used in the $\mathcal{H}$-partition optimization was set as 10^{-3}. The error tolerance ϵ_{LU} used in the LU factorization was 10^{-2}. The computer used was a Dell's PowerEdge 6950s server with 8222SE AMD Opteron processors. Double precision was employed in all the simulations.

First, we tested the accuracy of the $\mathcal{H}$-matrix representation obtained by the proposed rank minimization and partition optimization methods. The error of the $\mathcal{H}$-matrix representation is measured by $\left\|\mathbf{G} - \tilde{\mathbf{G}}\right\| / \|\mathbf{G}\|$, where $\tilde{\mathbf{G}}$ is the $\mathcal{H}$-matrix representation of the original $\mathbf{G}$, and the Frobenius norm is used. We simulated a PEC plate from 2 λ to 14 λ, and a PEC sphere from 2 λ to 8 λ. Fig. 14 shows the accuracy of the $\mathcal{H}$-matrix representation generated from the proposed method. Excellent accuracy can be observed in the entire range of electric sizes. Since the assessment of the matrix error requires the knowledge of the full matrix $\mathbf{G}$, we did not simulate larger problem sizes in this testing case.

Next, we tested the accuracy and efficiency of the proposed direct LU based IE solver. The first example is a conducting sphere illuminated by a normally incident plane wave. The electric size of the sphere is from 2 λ to 45 λ. The discretization results in unknowns from 3,688 to 1,152,368. The average partition rank k_{ave} resulting from the proposed rank minimization and $\mathcal{H}$-partition optimization methods is shown in Fig. 15(a) in the entire range of electric sizes. It can be seen that k_{ave} is minimized to be a small number compared to N. In Fig. 15(b), we plot the memory cost of the proposed solver. In 15(c) and (d), we plot the CPU time of the LU decomposition, and LU solution, respectively. It is clear that the memory and CPU cost of the proposed direct solver scale more favorably with N compared to conventional direct solvers. To test the accuracy, in Fig. 16(a) and (b), we plot the E-plane bi-static RCS simulated for 12 λ and 26 λ, respectively. The RCS generated with $\epsilon_{LU} = 10^{-3}$ is shown to agree very well with the analytical Mie-Series solution. In Fig. 16(c), we plot the solution error. Less than 6% error is observed in the entire frequency band.

The second example is a 3D conducting plate, the electric size of which is from 2 λ to 60 λ. The discretization results in 1,160 unknowns to 1,078,800 unknowns. The average partition rank k_{ave} resulting from the proposed rank minimization and partition optimization methods is shown in Fig. 17(a). It is

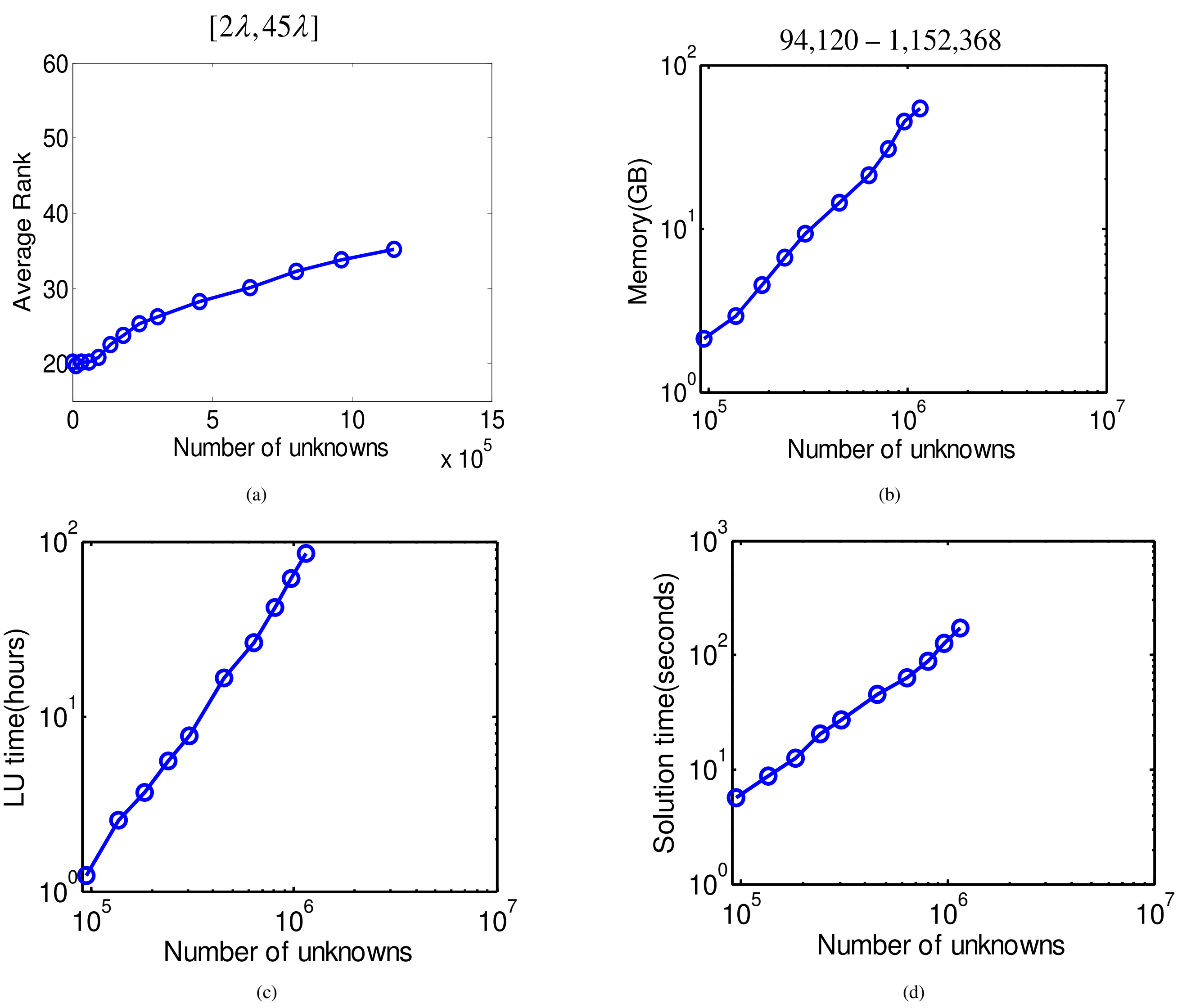
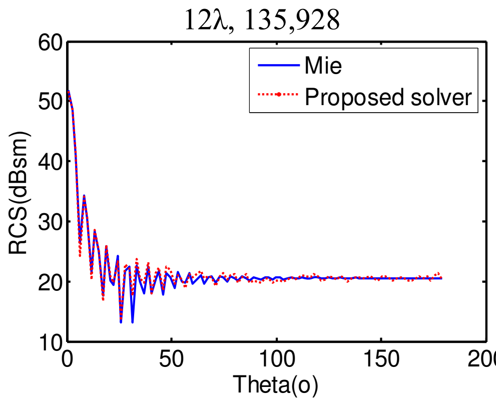


Fig. 15: Simulation of a PEC sphere from 2 λ to 45 λ with unknowns from 3,688 to 1,152,368. (a) Average partition rank k_{ave}. (b) Memory requirement. (c) LU factorization time. (d) LU solution time.

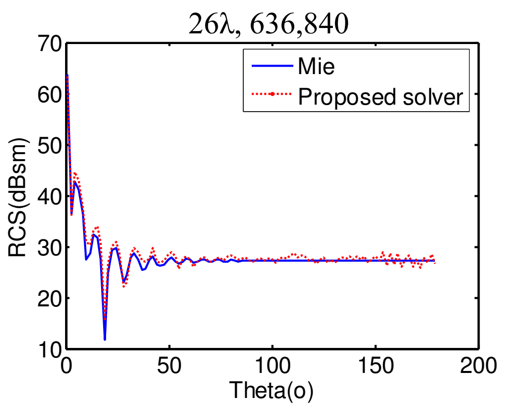
clear that the average partition rank is minimized to be a small number, in addition, for the plate example, it is controlled to be almost a constant in the entire range of electric sizes. The C_{sp} for this example can be seen from Fig. 7(b). In Fig. 17(b), we plot the memory cost of the proposed direct IE solver. In Fig. 17(c), we plot the solution error of the proposed LU solver measured by $\left\|\tilde{\mathbf{G}}I - V\right\| / \|V\|$. Good accuracy can be observed. In addition, the accuracy is kept to be almost a constant in the entire range. From Fig. 17(d) to (f), we plot the CPU time of the $\mathcal{H}$-matrix construction, LU decomposition, and LU solution, respectively. The computation for the 60 λ case having over 1 million unknowns was finished within 10-hour LU decomposition time, 55-second LU solution time, and costing 31.5 GB storage only.

The last example is a conducting cylinder, the length of which is from 2 λ to 96 λ. The ratio of length to radius is 20. The number of unknowns is from 1,391 to 1,075,200. In Fig. 18(a), we plot the average partition rank k_{ave} resulting from the proposed rank minimization and partition optimization methods in the entire electric-size range. In Fig. 18(b), we plot C_{ad} versus the electric size. Without the proposed $\mathcal{H}$-partition optimization, the C_{ad} is between 47 and 51. Clearly, C_{ad}, and hence C_{sp} is reduced greatly. In Fig. 19(a), we plot the solution error with respect to the number of unknowns. Good accuracy is observed. Note that the error is controllable by ϵ, ϵ_{opt}, and ϵ_{LU}. If better accuracy is required, it can be obtained by reducing the error tolerances. From Fig. 19(b) to (d), we show the computational cost of the proposed direct LU solver in LU factorization, LU Solution, and storage. The LU factorization for the 1,075,200 unknown case costs less than 20 hours and 37.6 GB memory. The LU solution time is 85 seconds only. Compared to results obtained for similar examples reported in open literature, the proposed direct IE solver is much more efficient in both CPU time and memory consumption, even though double precision was used for computation.

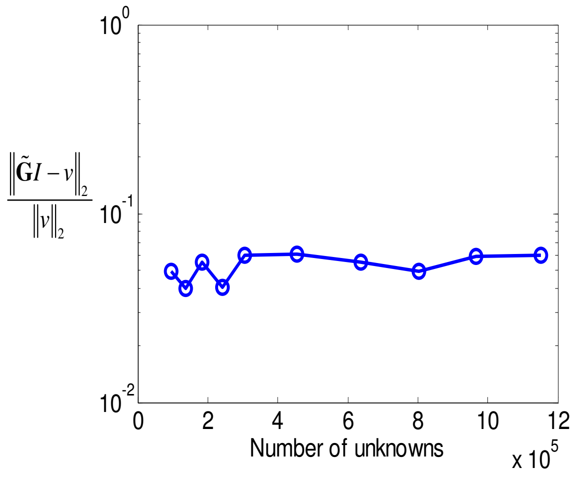
The above direct matrix solutions were all generated based on $\epsilon = 10^{-4}$, $\epsilon_{opt} = 10^{-3}$, and $\epsilon_{LU} = 10^{-2}$. To test the



(a)



(b)



(c)

Fig. 16: (a) RCS of a conducting sphere simulated by the proposed solver at 12 λ. (b) RCS of a conducting sphere simulated by the proposed solver at 26 λ. (c) Solution error in the entire frequency band from 2 λ to 45λ.

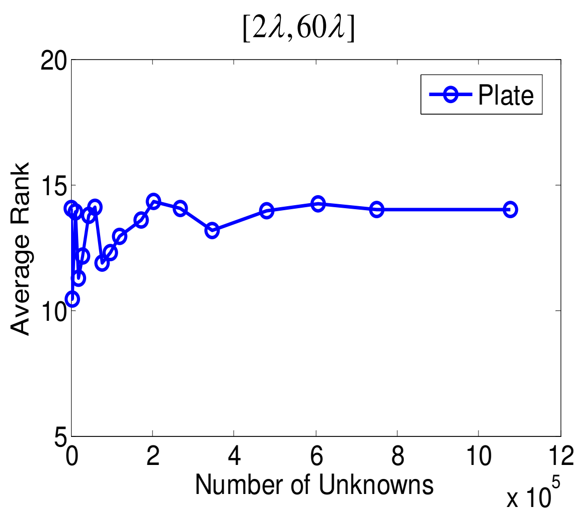
performance of the proposed direct solver for a higher order of accuracy, we set $\epsilon = 10^{-5}$, $\epsilon_{opt} = 10^{-4}$, and $\epsilon_{LU} = 10^{-3}$. We simulated the same plate example simulated in Fig. 17. In Fig. 20(a), we plot the solution error with respect to the number of unknowns. An excellent accuracy can be observed in the entire range of electric sizes, where the worst error is shown to be less than 0.33%, in comparison with the 3% error achieved by the previous accuracy setting. From Fig. 20(b) to (d), we show the computational cost of the proposed direct LU solver in memory, LU factorization, and LU solution with respect to N.

IX. Conclusion

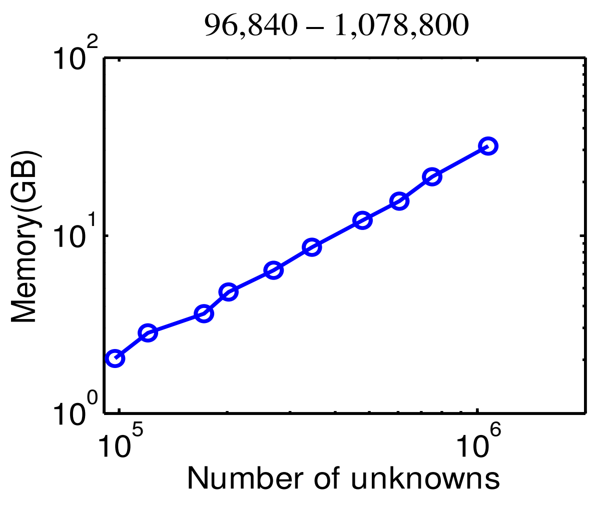
In this work, we show that the cost of an $\mathcal{H}$-based computation of high-frequency problems is determined not only by the block rank k_i (the rank of each admissible block) but also by the number of admissible blocks that have rank k_i, and hence $\mathcal{H}$-partition. By optimizing the $\mathcal{H}$-partition based on accuracy for each frequency to minimize the number of admissible blocks at each tree level, the computational cost of an $\mathcal{H}$-based computation of high-frequency problems can be significantly reduced. We show that there exists a large space to optimize the $\mathcal{H}$-matrix partition for frequency dependent problems. Existing $\mathcal{H}$-matrix partition is based on a geometry-based admissibility condition. This condition is controlled by an empirical parameter instead of a prescribed accuracy. The resultant partition is not optimal, especially for electrodynamic problems. We hence develop a new $\mathcal{H}$-partition method that is frequency dependent, and also directly controlled by accuracy requirements. With the proposed new partition, the number of admissible blocks at each tree level is significantly reduced compared to that generated by the conventional geometry based $\mathcal{H}$-partition, thus leading to a significant reduction in computational cost.

In addition, the actual number of the block rank should be determined and minimized based on accuracy requirements. This paper provides an efficient matrix algebra based method to perform this task with negligible computational overhead.

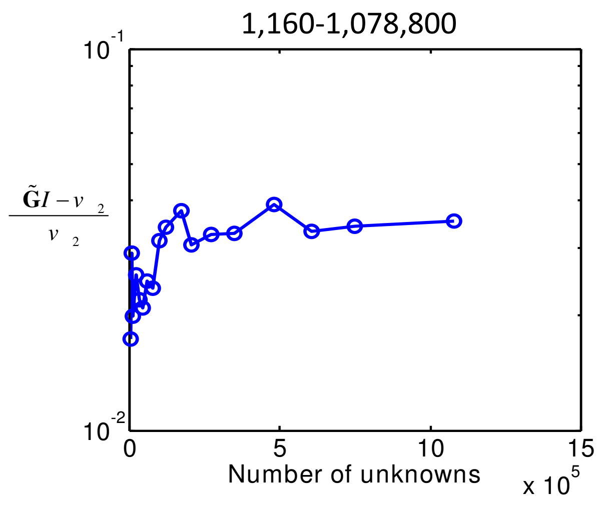
Moreover, we developed an efficient $\mathcal{H}$-based LU-factorization for directly solving the dense system matrix resulting from an IE-based analysis of large-scale electrodynamic problems. The dense matrix involving over 1 million unknowns formulated for a 96-wavelength problem is factorized in fast CPU run time, modest memory usage, and



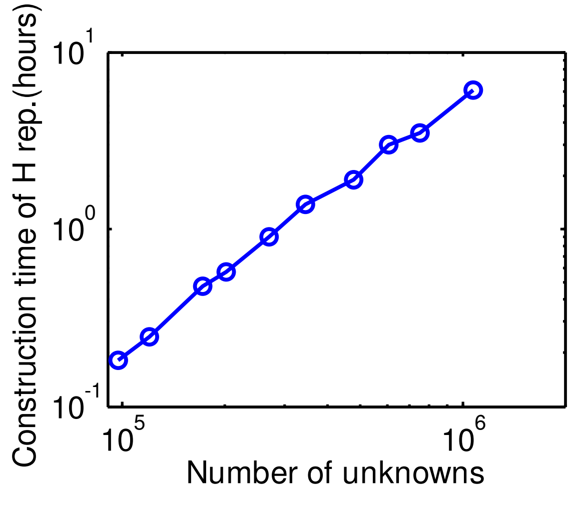
(a)



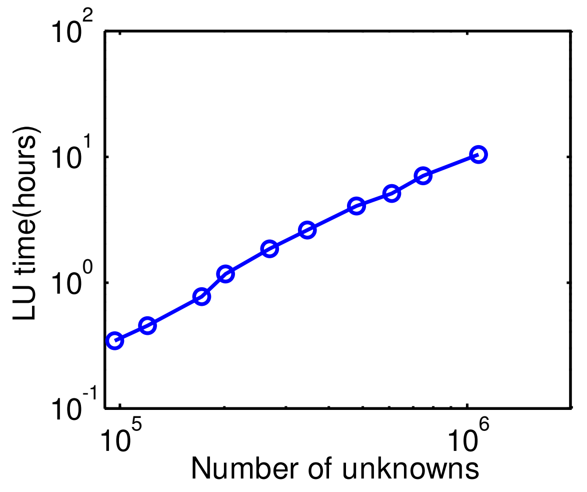
(b)



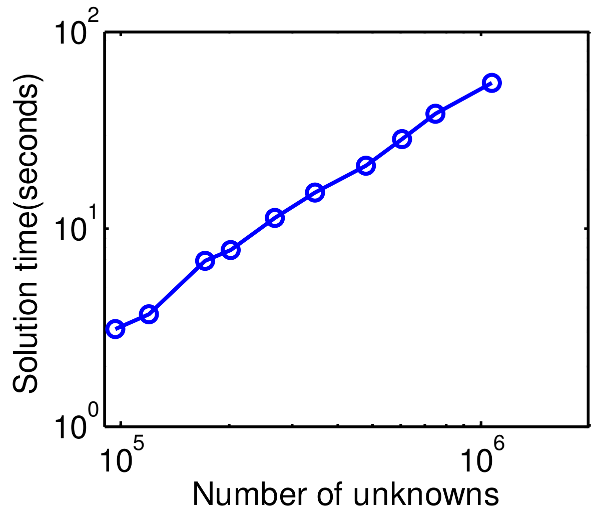
(c)



(d)



(e)



(f)

Fig. 17: Simulation of a PEC plate from 2 λ to 60 λ. (a) Average partition rank. (b) Memory cost. (c) Solution error. (d) $\mathcal{H}$-matrix construction time. (e) LU factorization time. (f) LU solution time.

with prescribed accuracy satisfied. The proposed methods for minimizing the rank and optimizing the $\mathcal{H}$-partition not only can be used in the proposed direct solver, but also can be employed in other fast integral equation solvers for both iterative and direct solutions.

References

[1] V. Rokhlin, "Rapid solution of integral equations of scattering in two dimensions," *J. Comput. Phys.*, vol. 86, pp. 419–439, Feb 1990.

[2] J. Song, C. Lu, and W. Chew, "Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects," *IEEE Trans. Antennas Propag.*, vol. 45, no. 10, pp. 1488–1493, Oct 1997.

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

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

[5] E. Bleszynski, M. Bleszynski, and T. Jarozewicz, "AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems," *Radio Sci.*, vol. 31, no. 10, pp. 1225-1251, Sept.-Oct. 1996.

[6] J. Shaeffer, "Direct Solve of Electrically Large Integral Equations for Problem Sizes to 1 M unknowns," *IEEE Trans. Antennas Propag.*, vol. 56, no. 8, pp. 2306–2313, Aug. 2008.

[7] 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. Antennas Propag.*, vol. 56, no. 8, pp. 2427–2441, Aug. 2008.

[8] W. Chai and D. Jiao, "An $\mathcal{H}^2$-Matrix-Based Integral-Equation Solver of Reduced Complexity and Controlled Accuracy for Solving Electrodynamic Problems," *IEEE Trans. Antennas Propag.*, vol. 57, no. 10, pp. 3147–3159, Oct. 2009.

[9] W. Chai and D. Jiao, "$\mathcal{H}$ and $\mathcal{H}^2$-Matrix-Based Fast Integral-Equation Solvers for Large-Scale Electromagnetic Analysis," *IET Microwaves, Antennas and Propagation*, vol. 4, no. 10, pp. 1583–1596, 2010.

[10] W. Chai, D. Jiao, and C. K. Koh "A Direct Integral-Equation Solver of Linear Complexity for Large-Scale 3D Capacitance and Impedance Extraction," *46th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 752–757, July, 2009.

[11] W. Chai and D. Jiao, "An LU Decomposition Based Direct Integral Equation Solver of Linear Complexity and Higher-Order Accuracy for Large-Scale Interconnect Extraction," *IEEE Trans. Advanced Packaging*, vol. 33, no. 4, pp. 794–803, 2010.

[12] W. Chai and D. Jiao, "Dense Matrix Inversion of Linear Complexity for Integral-Equation Based Large-Scale 3-D Capacitance Extraction," *IEEE Trans. Microw. Theory Tech*, vol. 59, no. 10, pp. 2404–2421, October, 2011.

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

[14] W. Hackbusch and B. Khoromskij, "A sparse matrix arithmetic. Part II: Application to multi-dimensional problems," *Computing*, vol. 64, pp. 21–47, 2000.

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

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

[17] L. Grasedyck, "Adaptive recompression of H-matrices for BEM," *Computing*, vol. 74, no. 3, pp. 205-223, 2005.

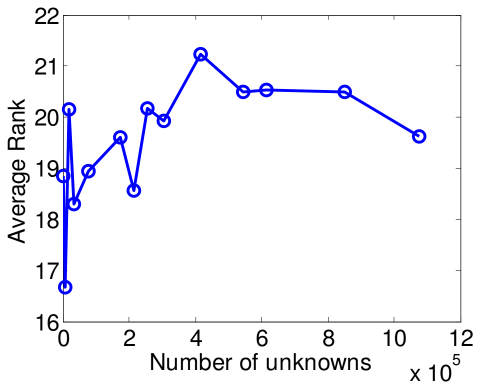
[18] W. Hackbusch, B. Khoromskij, and S. Sauter, "On $\mathcal{H}^2$-matrices," *Lecture on Applied Mathematics*, H. Bungartz, R. Hoppe, and C. Zenger, eds., pp. 9-29, 2000.

[19] M. Bebendorf, "Approximation of boundary element matrices," *Numer. Math.*, vol. 86, no. 4, pp. 565–589, 2000.

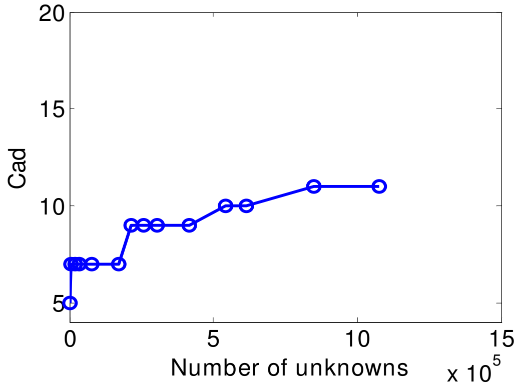
[20] K. Zhao, M. N. Vouvakis, and J. Lee, "The Adaptive Cross Approximation Algorithm for Accelerated Method of Moments Computations of EMC Problems," *IEEE Trans. EMC*, vol. 47, no. 4, pp. 763–773, 2005.

[21] W. Chai and D. Jiao, "A Complexity-Reduced $\mathcal{H}$-Matrix Based Direct Integral Equation Solver with Prescribed Accuracy for Large-Scale Electrodynamic Analysis," *the IEEE International Symposium on Antennas and Propagation*, 4 pages, July 2010.

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

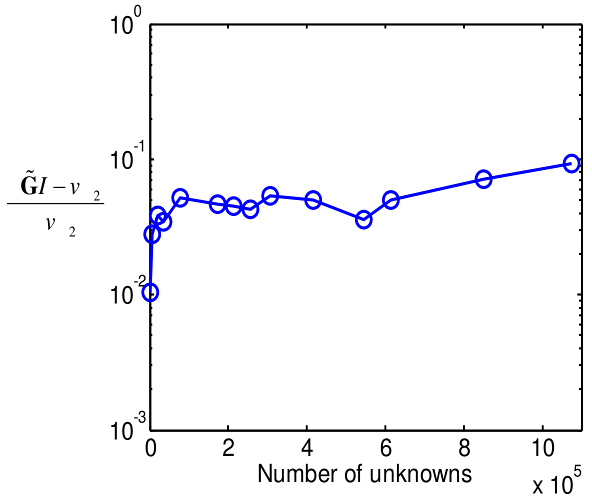


(a)

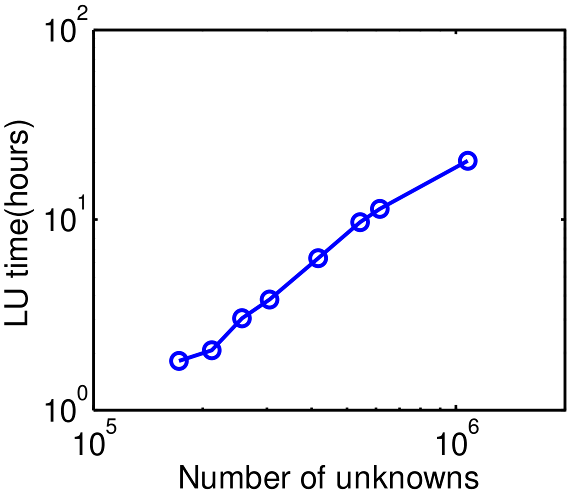


(b)

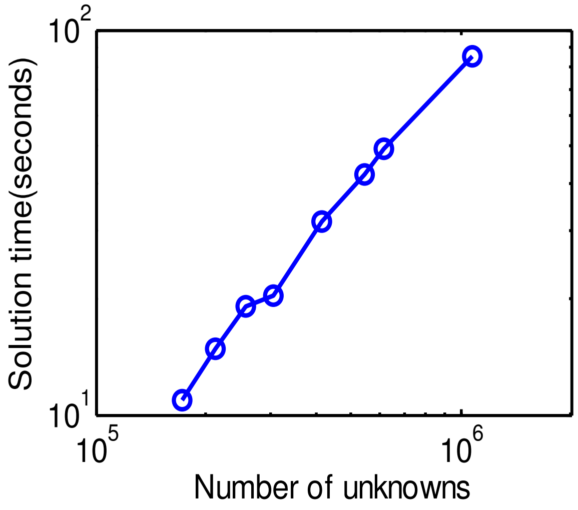
Fig. 18: Simulation of a PEC cylinder from 2 λ to 96 λ. (a) k_{ave} versus N. (b) C_{ad} versus N.



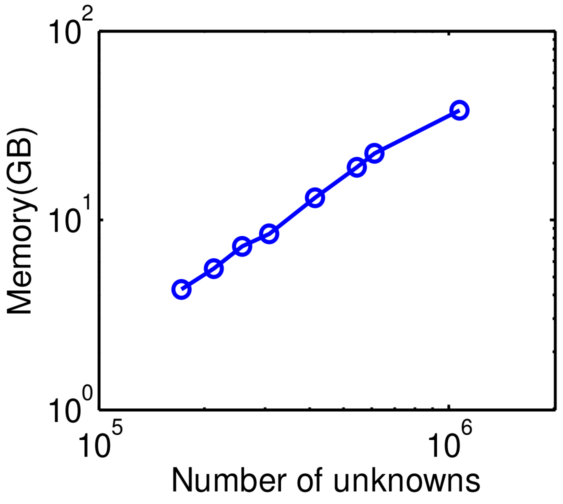
(a)



(b)

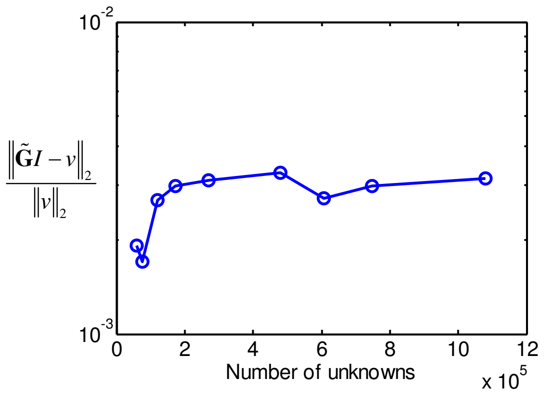


(c)

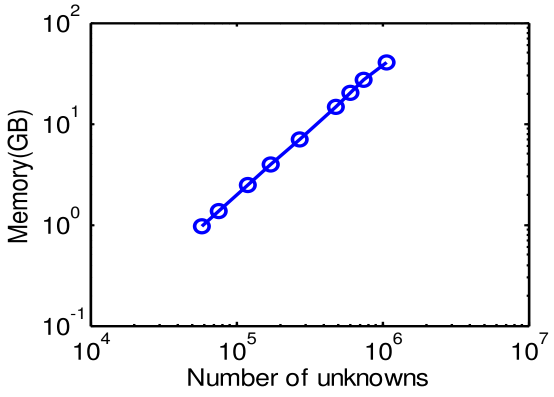


(d)

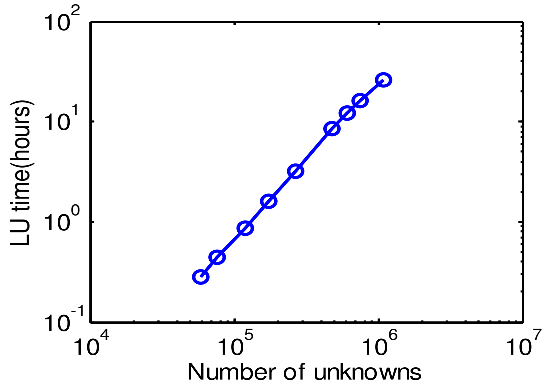
Fig. 19: Simulation of a PEC cylinder from 2 λ to 96 λ. (a) Solution error. (b) LU factorization time. (c) LU solution time. (d) Memory Cost.



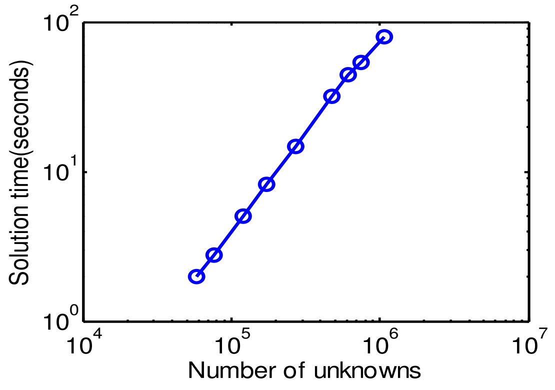
(a)



(b)



(c)



(d)

Fig. 20: Simulation of a PEC plate from 2 λ to 60 λ based on a higher-order accuracy setting ($\epsilon = 10^{-5}$, $\epsilon_{opt} = 10^{-4}$, and $\epsilon_{LU} = 10^{-3}$). (a) Solution error. (b) Memory Cost. (c) LU factorization time. (d) LU solution time.