Accuracy Directly Controlled Fast Direct Solution of General $\mathcal{H}^2$-Matrices and Its Application to Solving Electrodynamic Volume Integral Equations

Miaomiao Ma, *Graduate Student Member, IEEE*, and Dan Jiao, *Fellow, IEEE*

***Abstract*— The dense matrix resulting from an integral equation (IE)-based solution of Maxwell's equations can be compactly represented by an $\mathcal{H}^2$-matrix. Given a general dense $\mathcal{H}^2$-matrix, prevailing fast direct solutions involve approximations whose accuracy can only be indirectly controlled. In this paper, we propose new direct solution algorithms whose accuracy is directly controlled, including both factorization and inversion, for solving general $\mathcal{H}^2$-matrices. Different from the recursive inverse performed in existing $\mathcal{H}^2$-based direct solutions, this new direct solution is a one-way traversal of the cluster tree from the leaf level all the way up to the root level. The underlying multiplications and additions are carried out as they are without using formatted multiplications and additions whose accuracy cannot be directly controlled. The cluster bases and their rank of the original matrix are also updated level by level based on prescribed accuracy, without increasing computational complexity, to take into account the contributions of fill-ins generated during the direct solution procedure. For constant-rank $\mathcal{H}^2$-matrices, the proposed direct solution has a strict $O(N)$ complexity in both time and memory. For rank that linearly grows with the electrical size, the complexity of the proposed direct solution is $O(N\log N)$ in factorization and inversion time, and $O(N)$ in solution time and memory for solving volume IEs (VIEs). Rapid direct solutions of electrodynamic VIEs involving millions of unknowns have been obtained on a single CPU core with directly controlled accuracy. Comparisons with state-of-the-art $\mathcal{H}^2$-based direct VIE solvers have also demonstrated the advantages of the proposed direct solution in accuracy control, as well as achieving better accuracy with much less CPU time.**

***Index Terms*— Dense matrices, electrodynamic, electromagnetic analysis, fast direct solvers, frequency domain, $\mathcal{H}^2$-matrix, integral equations (IEs), linear complexity solvers, volume IEs (VIEs).**

I. Introduction

THE $\mathcal{H}^2$-matrix is a general mathematical framework [1]–[4] for compact representation and efficient computation of dense matrices. It can be utilized to develop fast solvers for electromagnetic analysis. Many existing fast solvers can be interpreted in this framework, although their developments may predate the $\mathcal{H}^2$-framework. For example, the matrix structure resulting from the well-known FMM-based methods [5], [6] is an $\mathcal{H}^2$-matrix. In other words, the $\mathcal{H}^2$-matrix can be viewed as an algebraic generalization of the FMM method. Multiplying an $\mathcal{H}^2$-matrix by a vector has a complexity of $O(N\log N)$ for solving electrically large surface integral equations (SIEs). In the mathematical literature, such as [7], it is shown that the storage, matrix-vector multiplication (MVM), and matrix–matrix multiplication of an $\mathcal{H}^2$-matrix can be performed in optimal $O(N)$ complexity for constant-rank cases. However, no such complexity is shown for either matrix factorization or inversion, regardless of whether the rank is a bounded constant or an increasing variable. Later in [8] and [9], fast matrix inversion and LU factorization algorithms are developed for $\mathcal{H}^2$-matrices. They have shown a complexity of $O(N)$ for solving constant-rank cases, such as electrically small and moderate problems for both SIE and volume IE (VIE) [10]–[13]; and a complexity of $O(N\log N)$ for solving electrically large VIEs [14]–[16]. There exist other significant contributions in fast direct solvers, such as [17]–[25], [32], and [33]. The focus of this paper, however, is a fast direct solution to an $\mathcal{H}^2$-matrix, as this matrix structure is general suitable for solving both dense and sparse matrices.

Despite a significantly reduced complexity, in the existing direct solutions of $\mathcal{H}^2$-matrices, such as [8] and [9], formatted multiplications and additions are performed instead of actual ones, where the cluster bases used to represent a matrix are also used to represent the matrix's inverse as well as LU factors. During the inversion and factorization procedure, only the coupling matrix of each admissible block is computed, while the cluster bases are kept to be the same as those in the original matrix. Physically speaking, such a choice of the cluster bases for the inverse matrix often constitutes an accurate choice. However, mathematically speaking, the accuracy of the inversion and factorization cannot be directly controlled. The lack of an accuracy control is also observed in the $\mathcal{H}^2$-based formatted matrix–matrix multiplication in the mathematical literature, such as [7]. If the accuracy is to be controlled, the inverse as well as the matrix–matrix multiplication algorithm must be completely changed, as the original formatted framework does not offer a mechanism to control

Manuscript received March 22, 2017; revised June 5, 2017 and July 16, 2017; accepted July 20, 2017. Date of publication August 15, 2017; date of current version January 4, 2018. This work was supported in part by the NSF under Award 1619062, in part by the SRC (Task 1292.073), and in part by DARPA under Award HR0011-14-1-0057. *(Corresponding author: Dan Jiao.)*

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/TMTT.2017.2734090

the accuracy without increasing complexity. Algorithmwise, the collect operation involves approximations, whose accuracy is not directly controlled. This operation is performed when multiplying a nonleaf block with a nonleaf block, or multiplying a nonleaf block with an admissible block, with the target block being admissible. In this operation, the four blocks, either admissible or inadmissible, are collected to a single admissible block based on the cluster bases of the original matrix. If the target block cannot be accurately represented by the original cluster bases, then an error would occur. When the accuracy of the direct solution is not satisfactory, one can only change the original $\mathcal{H}^2$-representation, i.e., change the cluster bases and/or the rank for representing the original matrix (based on a prescribed accuracy), with the hope that they can better represent the inverse and LU factors. Therefore, the accuracy of the resulting direct solution is not directly controlled. A direct solution with an explicit accuracy control should perform every multiplication and addition as it is without assuming a format of the matrix involved in the computation. Meanwhile, every operation in the direct solution should be either exact or performed based on a prescribed accuracy. This is what is pursued and achieved in this paper.

Recently, an HSS-matrix structure [26] has also been explored for electromagnetic analysis [27]–[30]. Exact-arithmetic fast factorization and inversion algorithms have been developed in [28] and [29]. In other words, the direct solution of the HSS matrix does not involve any approximation. However, the HSS matrix, as a special class of $\mathcal{H}^2$-matrix, requires only one admissible block be formed for each node in the $\mathcal{H}^2$-tree. All the off-diagonal matrices in the HSS matrix are represented as low-rank matrices. As a result, the resultant rank can be too high to be computed efficiently, especially for electrically large analysis. This is because as long as the sources and observers are separated, even though their distance is infinitely small, their interaction must be represented by a low-rank matrix to suite the structure of an HSS matrix. From this perspective, an $\mathcal{H}^2$-matrix is a more-efficient structure for general applications, since the distance between the sources and observers can be used to reduce the rank required to represent a dense matrix for a prescribed accuracy. However, the accuracy-controlled direct solution of general $\mathcal{H}^2$-matrices is still lacking in the open literature. The contribution of this paper is such an accuracy-controlled direct solution of general $\mathcal{H}^2$-matrices. This solution can be applied to solve a general $\mathcal{H}^2$-matrix, be it real- or complex-valued, symmetric or unsymmetrical. Its complexity is $O(N)$ for constant-rank $\mathcal{H}^2$-matrices in both time and memory. For rank that linearly grows with electrical size, the complexity of the proposed direct solution is $O(N\log N)$ in factorization and inversion time, $O(N)$ in solution time, and memory usage for solving VIEs. As a demonstration, we show how it is used to solve large VIEs involving millions of unknowns on a single core CPU, and with controlled accuracy.

The rest of this paper is organized as follows. In Section II, we review the mathematical background of this paper. In Section III, we present the proposed factorization and inversion algorithms for general $\mathcal{H}^2$-matrices. In Section IV, we describe the proposed matrix solution algorithm, including both backward and forward substitutions. The accuracy and the computational complexity of the proposed direct solutions are analyzed in Section V. Section VI demonstrates the application of this paper in solving electrodynamic VIEs. In Section VII, we summarize this paper.

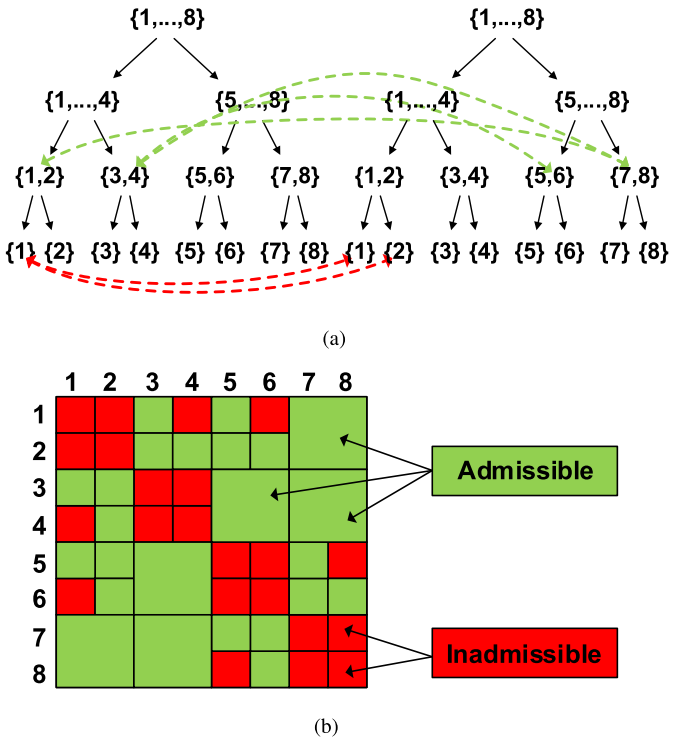


Fig. 1. Illustration of an $\mathcal{H}^2$-matrix. (a) Block cluster tree. (b) Matrix partition where admissible blocks are stored in the format of (1).

II. Background of This Paper

An $\mathcal{H}^2$-matrix [1]–[4] is generally stored in a tree structure, with each node in the tree called a cluster. The number of unknowns in each cluster at the leaf level is no greater than *leafsize*, a predefined constant. An $\mathcal{H}^2$-matrix is partitioned into multilevel admissible and inadmissible blocks based on an admissibility condition. As an example, a four-level block cluster $\mathcal{H}^2$-tree is shown in Fig. 1(a), where a green link denotes an admissible block, formed between a row cluster and a column cluster that satisfy the admissibility condition, and a red link denotes an inadmissible block. The resultant $\mathcal{H}^2$-matrix is shown in Fig. 1(b). An admissible block in an $\mathcal{H}^2$-matrix is represented as

$$\mathbf{Z}_{t,s} = (\mathbf{V}_t)_{\#t\times k}(\mathbf{S}_{t,s})_{k\times k}(\mathbf{V}_s)^T_{\#s\times k} \tag{1}$$

where $\mathbf{V}_t$ ($\mathbf{V}_s$) is called cluster basis of cluster t (s), $\mathbf{S}_{t,s}$ is called coupling matrix, # denotes the cardinality of a set, and k is rank. The cluster basis in an $\mathcal{H}^2$-matrix has a nested property. This means the cluster basis for a nonleaf cluster t, $\mathbf{V}_t$, can be expressed by its two children's cluster bases, $\mathbf{V}_{t_1}$ and $\mathbf{V}_{t_2}$, as the following:

$$(\mathbf{V}_t)_{\#t\times k} = \begin{bmatrix} (\mathbf{V}_{t_1})_{\#t_1\times k_1} & 0 \\ 0 & (\mathbf{V}_{t_2})_{\#t_2\times k_2} \end{bmatrix} \begin{bmatrix} (\mathbf{T}_{t_1})_{k_1\times k} \\ (\mathbf{T}_{t_2})_{k_2\times k} \end{bmatrix} \tag{2}$$

where $\mathbf{T}_{t_1}$ and $\mathbf{T}_{t_2}$ are called transfer matrices. In an $\mathcal{H}^2$-matrix, the number of blocks formed by a single cluster at

each tree level is bounded by a constant, C_{sp}. The size of an inadmissible block is *leafsize*, and inadmissible blocks appear only at the leaf level.

In the mathematical literature [7], for constant-rank cases, it is shown that the computational complexity of an $\mathcal{H}^2$-matrix is optimal (linear) in storage, MVM, and matrix–matrix multiplication. In [8]–[13], it is shown that $O(N)$ direct solutions can also be developed for $\mathcal{H}^2$-matrices for constant-rank cases as well as electrically moderate cases whose rank can be controlled by a rank function. For electrically large analysis, the rank of an $\mathcal{H}^2$-representation of IE kernels is not constant any more for achieving a prescribed accuracy. Different $\mathcal{H}^2$-representations also result in different ranks and their growth rates with electrical size, and hence different complexities. For example, if an FMM-based $\mathcal{H}^2$-representation is used to model an IE operator, the asymptotic rank would scale *quadratically* with the electrical size. If an interpolation-based $\mathcal{H}^2$-representation is used to model an IE operator, the asymptotic rank would scale *quadratically* with the electrical size in SIEs, and *cubically* with the electrical size in VIEs. Despite its full-rank model in SIEs, the FMM complexity is as low as $O(N\log N)$ for one MVM, because its coupling matrix is diagonal and transfer matrix is sparse. On the other hand, if one uses an SVD-based minimal-rank representation, the resultant coupling matrix is not diagonal. However, the asymptotic rank of such a minimal-rank representation only scales linearly with electrical size for general 3-D problems as shown by the analysis given in [31]. In this analysis, there are additional findings that are not utilized in the FMM-based rank analysis or a source-observer separated rank analysis, such as SVD turns to a Fourier analysis in a linear-shift invariant system, and the Fourier transform of Green's function reveals that not all the plane waves are important, and only those whose wavenumber is closest to the given wavenumber k_0 make the most important contribution. These plane waves are counted for prescribed accuracy for 1-, 2-, and 3-D problems in [31]. It is found that the growth rate with electrical size is no greater than linear.

Regardless of which $\mathcal{H}^2$-matrix is generated to represent the integral or partial differential equation operators, the proposed direct solution is equally applicable. In this paper, when we apply the proposed direct solution to solve large VIEs, we employ a minimal-rank $\mathcal{H}^2$-representation of the VIE operator using the technique we developed in [14]–[16].

III. Proposed Factorization and Inversion Algorithms

Given a general $\mathcal{H}^2$-matrix $\mathbf{Z}$, the proposed direct solution is a one-way tree traversal from the leaf level all the way up to the root level of the $\mathcal{H}^2$-tree. At each level, the factorization proceeds from the left to the right across all the clusters. This is very different from the algorithm of [9], [11], [14], and [15], where a recursive procedure is performed. To help better understand the proposed algorithm, while we present a generic algorithm, we will use the $\mathcal{H}^2$-matrix shown in Fig. 1(b), as an example to illustrate the overall procedure. In Fig. 1(b), green blocks are admissible blocks, and red ones represent inadmissible blocks.

A. Computation at the Leaf Level

1) For the First Leaf Cluster: We start from the leaf level ($l = L$). For the first cluster, whose index is denoted by $i = 1$, we perform three steps as follows.

Step 1: We compute the complementary cluster basis $\mathbf{V}_i^{\perp}$, which is orthogonal to $\mathbf{V}_i$ (cluster basis of cluster i). This can be easily done by carrying out an SVD or QL factorization on $\mathbf{V}_i$. The cost is not high; instead, it is constant at the leaf level. We then combine $\mathbf{V}_i$ with $\mathbf{V}_i^{\perp}$ to form $\widetilde{\mathbf{Q}}_i$ matrix as the following:

$$\widetilde{\mathbf{Q}}_i = [(\mathbf{V}_i^{\perp})_{\#i\times(\#i-k_l)} \ (\mathbf{V}_i)_{\#i\times k_l}] \tag{3}$$

in which k_l is the rank at level l (now $l = L$), and # denotes the number of unknowns contained in a set.

Step 2: Let $\mathbf{Z}_{\text{cur}}$ be the current matrix that remains to be factorized. For leaf cluster $i = 1$, $\mathbf{Z}_{\text{cur}} = \mathbf{Z}$. We compute a transformed matrix $\mathbf{Q}_i^H \mathbf{Z}_{\text{cur}} \overline{\mathbf{Q}}_i$, and use it to overwrite $\mathbf{Z}_{\text{cur}}$; thus

$$\mathbf{Z}_{\text{cur}} = \mathbf{Q}_i^H \mathbf{Z}_{\text{cur}} \overline{\mathbf{Q}}_i \tag{4}$$

where $\mathbf{Q}_i^H$ denotes $\mathbf{Q}_i$'s complex conjugate transpose, and

$$\mathbf{Q}_i = \text{diag}\{\mathbf{I}_1, \mathbf{I}_2, \ldots, \widetilde{\mathbf{Q}}_i, \ldots, \mathbf{I}_{\#\{\text{leaf clusters}\}}\} \tag{5}$$

is a block diagonal matrix. The ith diagonal block in $\mathbf{Q}_i$ is $\widetilde{\mathbf{Q}}_i$, and other blocks are identity matrices $\mathbf{I}$, the size of each of which is the corresponding leaf-cluster size. The subscripts in the right-hand side of (5) denote the indexes of diagonal blocks, which are also the indexes of leaf clusters, and #{leaf clusters} stands for the number of leaf clusters. If leaf cluster $i = 1$ is being computed, $\widetilde{\mathbf{Q}}_i$ appears in the first block of (5). In addition, $\overline{\mathbf{Q}}_i$ in (4) denotes the complex conjugate of $\mathbf{Q}_i$, as the cluster bases we construct are complex-valued to account for general $\mathcal{H}^2$-matrices. In (5), $\mathbf{Q}_i$ only consists of one $\widetilde{\mathbf{Q}}_i$, where i is the current cluster being computed. This is because $\widetilde{\mathbf{Q}}_i$ values for other clusters are not ready for use yet, since in the proposed algorithm, we will update cluster bases to take into account the contribution from fill-ins. This will become clear in the sequel.

The purpose of doing this step, i.e., obtaining (4), is to introduce zeros in the matrix being factorized, since

$$\widetilde{\mathbf{Q}}_i^H \times \mathbf{V}_i = \begin{bmatrix} \mathbf{0}_{(\#i-k_l)\times k_l} \\ \mathbf{I}_{k_l\times k_l} \end{bmatrix} \tag{6}$$

$$\mathbf{V}_i^T \times \overline{\widetilde{\mathbf{Q}}}_i = [\mathbf{0}_{k_l\times(\#i-k_l)} \quad \mathbf{I}_{k_l\times k_l}]. \tag{7}$$

The operation of (4) thus zeros out the first $(\#i - k_l)$ rows of the admissible blocks formed by row cluster i, as well as the first $(\#i - k_l)$ columns in $\mathbf{Z}_{\text{cur}}$, as shown by the white blocks in Fig. 2. The real computation of (4) is, hence, in the inadmissible blocks of cluster i, as shown by the four red blocks associated with cluster $i = 1$ in Fig. 2.

Step 3: After Step 2, the first $(\#i - k_l)$ rows and columns of $\mathbf{Z}_{\text{cur}}$ in the admissible blocks of cluster i become zero. Hence, if we eliminate the first $(\#i - k_l)$ unknowns of cluster i via a partial LU factorization, the admissible blocks of cluster i at these rows and columns would stay as zero, and thus not affected. As a result, the resulting L and U factors, as well

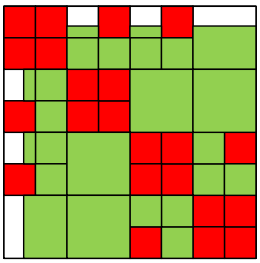


Fig. 2. Illustration of $\mathbf{Q}_1^H \mathbf{Z} \overline{\mathbf{Q}}_1$.

as the fill-in blocks generated, would only involve a constant number of blocks, which is bounded by $O(C_{\text{sp}}^2)$.

In view of the above-mentioned property, in Step 3, we perform a partial LU factorization to eliminate the first $(\#i - k_l)$ unknowns of cluster i in $\mathbf{Z}_{\text{cur}}$. Let this set of unknowns be denoted by i'. $\mathbf{Z}_{\text{cur}}$ can be cast into the following form accordingly:

$$\mathbf{Z}_{\text{cur}} = \begin{bmatrix} \mathbf{F}_{i'i'} & \mathbf{Z}_{i'c} \\ \mathbf{Z}_{ci'} & \mathbf{Z}_{cc} \end{bmatrix} \tag{8}$$

in which $\mathbf{F}_{i'i'}$ denotes the diagonal block of the $(\#i - k_l)$ unknowns, and $\mathbf{Z}_{i'c}$ is the remaining rectangular block in the row of set i'. Here, c contains the rest of the unknowns, which do not belong to i'. The $\mathbf{Z}_{ci'}$ is the transpose block of $\mathbf{Z}_{i'c}$, and $\mathbf{Z}_{cc}$ is the diagonal block formed by unknowns in set c. It is evident that $\mathbf{Z}_{i'c}$ $(\mathbf{Z}_{ci'})$ only consists of $O(C_{\text{sp}})$ inadmissible blocks, as the rest of the blocks are zero. Take leaf cluster $i = 1$ as an example. It forms inadmissible blocks with clusters $i = 2, 4, 6$, as shown by the red blocks in Fig. 2. Hence, $\mathbf{Z}_{i'c} = [\mathbf{F}_{12}, \mathbf{F}_{14}, \mathbf{F}_{16}]$ in addition to part of $\mathbf{F}_{11}$.

After performing a partial LU factorization to eliminate the unknowns contained in i', we obtain

$$\mathbf{Z}_{\text{cur}} = \begin{bmatrix} \mathbf{L}_{i'i'} & \mathbf{0} \\ \mathbf{Z}_{ci'}\mathbf{U}_{i'i'}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Z}}_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{i'i'} & \mathbf{L}_{i'i'}^{-1}\mathbf{Z}_{i'c} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{9}$$

where

$$\mathbf{F}_{i'i'} = \mathbf{L}_{i'i'}\mathbf{U}_{i'i'} \tag{10}$$

and the Schur complement is

$$\widetilde{\mathbf{Z}}_{cc} = \mathbf{Z}_{cc} - \mathbf{Z}_{ci'}\mathbf{U}_{i'i'}^{-1}\mathbf{L}_{i'i'}^{-1}\mathbf{Z}_{i'c} \tag{11}$$

in which $\mathbf{Z}_{ci'}\mathbf{U}_{i'i'}^{-1}\mathbf{L}_{i'i'}^{-1}\mathbf{Z}_{i'c}$ represents the fill-in blocks introduced due to the elimination of i'-unknowns. Because of the zeros introduced in the admissible blocks, $\mathbf{Z}_{i'c}$ and $\mathbf{Z}_{ci'}$ only consist of inadmissible blocks formed by cluster i, the number of which is bounded by constant C_{sp}. Hence, the number of fill-in blocks introduced by this step is also bounded by a constant, which is $O(C_{\text{sp}}^2)$.

Equation (9) can further be written in short as

$$\mathbf{Z}_{\text{cur}} = \mathbf{L}_i^l \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Z}}_{cc} \end{bmatrix} \mathbf{U}_i^l \tag{12}$$

in which the first matrix of (9), which is the $\mathbf{L}$ factor generated after a partial LU of cluster i at level l, is denoted by $\mathbf{L}_i^l$. Similarly, the rightmost matrix of (9), which is the $\mathbf{U}$ factor generated after a partial LU of cluster i at level l, is denoted by $\mathbf{U}_i^l$. The center matrix is the one that remains to be factorized. Thus, we let

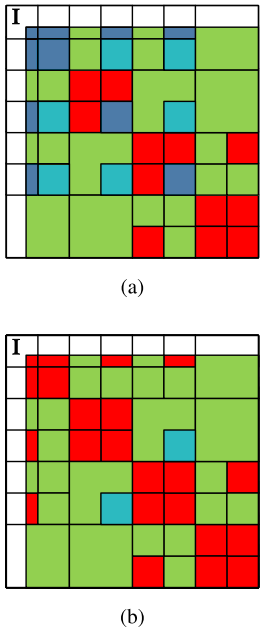


Fig. 3. (a) $\mathbf{Z}_{\text{cur}}$ after partial LU of cluster 1 with fill-in blocks marked in blue. (b) Fill-in blocks of cluster 2 turned green after cluster basis update.

$$\mathbf{Z}_{\text{cur}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Z}}_{cc} \end{bmatrix}. \tag{13}$$

This matrix is shown in Fig. 3(a), where the fill-in blocks are added upon previous $\mathbf{Z}_{\text{cur}}$. If the fill-in block is added upon an inadmissible block, we add the fill-in directly to the inadmissible block. For the example shown in Fig. 1(b), these blocks are the blocks marked in dark blue in Fig. 3(a) after the i' set of unknowns in cluster $i = 1$ is eliminated. If the fill-in appears in an admissible block, we store the fill-in block temporarily in this admissible block for future computation. These blocks are the blocks marked in light blue in Fig. 3(a). To explain in more detail, from (11), it can be seen that the fill-in block formed between cluster j and cluster k has the following form:

$$\mathbf{F}_{j,k} = \mathbf{Z}_{ji'}\mathbf{U}_{i'i'}^{-1}\mathbf{L}_{i'i'}^{-1}\mathbf{Z}_{i'k} \tag{14}$$

which is also low rank in nature. The row cluster j and column cluster k belong to the set of clusters that form inadmissible blocks with cluster i, whose number is bounded by C_{sp}. If the original block formed between j and k, $\mathbf{Z}_{\text{cur},jk}$, is admissible, no computation is needed for $\mathbf{F}_{j,k}$, i.e., we do not add it directly to the admissible block. Instead, we store it with this admissible block temporarily until cluster j is reached during the factorization. At that time, $\mathbf{F}_{j,k}$ will be used to compute an updated cluster basis for j. If the elimination of different clusters results in more than one fill-in block in admissible $\mathbf{Z}_{\text{cur},jk}$ block, the later ones are added upon existing $\mathbf{F}_{j,k}$. In other words, the fill-ins at the admissible $\mathbf{Z}_{\text{cur},jk}$ block are summed up and stored in a single $\mathbf{F}_{j,k}$.

In summary, Step 3 is to perform a partial LU factorization, as shown in (9), to eliminate the first $\#i - k_l$ unknowns in cluster i. The storage and computation cost of this step can

be analyzed as follows. In (9), the LU factorization of $\mathbf{F}_{i'i'}$ into $\mathbf{L}_{i'i'}$ and $\mathbf{U}_{i'i'}$ costs $O(leafsize^3)$ in computation, and $O(leafsize^2)$ in memory, which is hence constant. $\mathbf{Z}_{ci'}$ ($\mathbf{Z}_{i'c}$) only involves $O(C_{\text{sp}})$ inadmissible blocks. Hence, the computation of $\mathbf{Z}_{ci'}\mathbf{U}_{i'i'}^{-1}$ as well as $\mathbf{L}_{i'i'}^{-1}\mathbf{Z}_{i'c}$ is simply $O(C_{\text{sp}})$ operations, each of which scales as $O(leafsize^3)$, and hence, the cost is also constant. The results of the partial factorization, i.e., $\mathbf{L}_i^l$ and $\mathbf{U}_i^l$, are stored in $O(C_{\text{sp}})$ blocks of *leafsize*. Thus, the memory cost is also constant. As for the fill-in blocks, their number is bounded by C_{sp}^2, and each of which is of *leafsize*. Hence, the computation and storage associated with fill-in blocks are constant too, at the leaf level for each cluster.

2) For Other Leaf Clusters: Next, we proceed to the second leaf cluster $i = 2$ and other leaf clusters. For these leaf clusters, all the steps are the same as those performed for the first leaf cluster, except for adding another step before Step 1. We term this step as Step 0.

Step 0: This step is to update cluster basis $\mathbf{V}_i$ to account for the contributions of the fill-ins due to the elimination of previous clusters. If we do not update cluster bases $\mathbf{V}_i$, they are not accurate any more to represent the admissible blocks formed by cluster i, since the contents of these blocks have been changed due to the addition of the fill-in blocks. However, if we let the admissible block updated by fill-ins become a full-matrix block, i.e., an inadmissible block, the number of fill-ins would keep increasing after each step of partial factorization, and hence, the resultant complexity would be too high to tolerate. To overcome this problem, we propose to update cluster bases as follows.

For cluster i, we take all the fill-in blocks associated with its admissible blocks, and use them to update cluster basis $\mathbf{V}_i$. These fill-in blocks are shown in (14), which have been generated and stored in the corresponding admissible blocks during the elimination of previous clusters. Let them be $\mathbf{F}_{i,j_k}(k = 1, 2, \ldots, O(C_{\text{sp}}))$, where the first subscript denotes the row cluster i, and the second being the column cluster index. The number of such blocks is bounded by $O(C_{\text{sp}})$, because the number of admissible blocks formed by a single cluster, such as i, is no greater than C_{sp}. Next, we compute

$$\mathbf{G}_i = (\mathbf{I} - \mathbf{V}_i\mathbf{V}_i^H)\left(\sum\nolimits_k \mathbf{F}_{i,j_k}\mathbf{F}_{i,j_k}^H\right)(\mathbf{I} - \mathbf{V}_i\mathbf{V}_i^H)^H. \quad (15)$$

We then perform an SVD on $\mathbf{G}_i$, and truncate the singular vector matrix based on prescribed accuracy $\epsilon_{\text{fill-in}}$, obtaining

$$\mathbf{G}_i \overset{\epsilon_{\text{fill-in}}}{=} \left(\mathbf{V}_i^{\text{add}}\right)\Sigma\left(\mathbf{V}_i^{\text{add}}\right)^H. \quad (16)$$

$\mathbf{V}_i^{\text{add}}$ is the additional cluster basis we supplement, since it captures the part of the column space of the fill-in blocks, which is not present in the original $\mathbf{V}_i$. The cluster basis can then be updated for cluster i as

$$\widetilde{\mathbf{V}}_i = \begin{bmatrix}\mathbf{V}_i & \mathbf{V}_i^{\text{add}}\end{bmatrix}. \quad (17)$$

By doing so, we are able to keep the original admissible blocks to be admissible while accurately taking into account the fill-in's contributions. Hence, the number of fill-ins introduced due to the elimination of each cluster can be kept to be a constant, instead of growing. In addition, by keeping the original cluster basis $\mathbf{V}_i$ in the new bases instead of generating a completely new one, we can keep the nested relationship in the original $\mathcal{H}^2$-matrix for efficient computations at top levels. The computational cost of such a supplement of cluster bases is constant at leaf level, and scales as $O(k_l^3)$ at other levels (this will become clear soon) for each cluster.

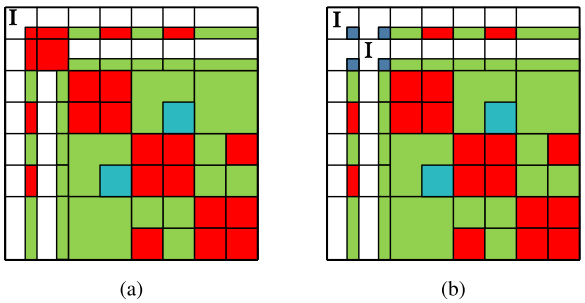


Fig. 4. (a) $\mathbf{Q}_2^H\mathbf{Z}_{\text{cur}}\overline{\mathbf{Q}}_2$. (b) $\mathbf{Z}_{\text{cur}}$ after partial LU of cluster 2.

In Fig. 3(b), Step 0 is reflected by turning the light blue blocks associated with cluster $i = 2$ in Fig. 3(a) back to green blocks, symbolizing that they become admissible again. After $\mathbf{V}_i$ is updated to $\widetilde{\mathbf{V}}_i$, we find its complementary bases, i.e., perform Step 1, to obtain

$$\widetilde{\mathbf{Q}}_i = [(\widetilde{\mathbf{V}}_i^{\perp})_{\#i\times(\#i-k_l)}\ (\widetilde{\mathbf{V}}_i)_{\#i\times k_l}]. \quad (18)$$

We then use $\widetilde{\mathbf{Q}}_i$ to generate (5), and project the current matrix to be factorized, shown in Fig. 3(b), to the matrix shown in (4). This again will zero out the first $(\#i - k_l)$ rows (columns) of the admissible blocks formed by cluster i, as shown in Fig. 4(a). We then proceed to Step 3 to perform a partial LU factorization. After this step, the matrix $\mathbf{Z}_{\text{cur}}$, which remains to be factorized, is shown in Fig. 4(b). Steps 0–3 are then repeated for the rest of the leaf clusters.

3) Updating the Coupling Matrix at the Leaf Level and the Transfer Matrix at One Level Higher: After the computation at the leaf level is finished, before proceeding to the next level, we need to update the coupling matrices at the leaf level, as well as the transfer matrices at one level higher. This is because the cluster bases have been updated at the leaf level.

For admissible blocks without fill-ins, their coupling matrices can be readily updated by appending zeros. For example, considering an admissible block formed between clusters i and k. Its coupling matrix $\mathbf{S}_{i,k}$ is updated to

$$\widetilde{\mathbf{S}}_{i,k} = \begin{bmatrix}\mathbf{S}_{i,k} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix} \quad (19)$$

because in this way

$$\widetilde{\mathbf{V}}_i\widetilde{\mathbf{S}}_{i,k}\widetilde{\mathbf{V}}_k^T = \mathbf{V}_i\mathbf{S}_{i,k}\mathbf{V}_k^T. \quad (20)$$

For admissible blocks with fill-ins, their coupling matrices are updated by adding the fill-in part onto the augmented coupling matrix, shown in (19), as

$$\widetilde{\mathbf{S}}_{i,k} = \widetilde{\mathbf{S}}_{i,k} + \widetilde{\mathbf{V}}_i^H\mathbf{F}_{i,k}\overline{\widetilde{\mathbf{V}}}_k \quad (21)$$

in which $\mathbf{F}_{i,k}$ represents the fill-in block associated with the admissible block.

We also need to update the transfer matrices at one level higher, i.e., at the $l = L-1$ level. This can be readily done by

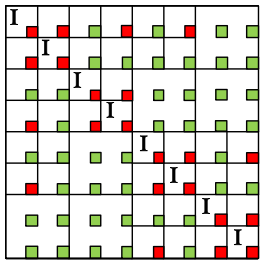


Fig. 5. $\mathbf{Z}_{\text{cur}}$ left to be factorized after leaf-level computation.

adding zeros to the original transfer matrices. To be clearer, consider a nonleaf cluster t, whose two children clusters are t_1 and t_2, and its cluster basis can be written as

$$\begin{bmatrix} \mathbf{V}_{t_1}\mathbf{T}_{t_1} \\ \mathbf{V}_{t_2}\mathbf{T}_{t_2} \end{bmatrix} = \begin{bmatrix} [\mathbf{V}_{t_1}\mathbf{V}_{t_1}^{\text{add}}] & \mathbf{0} \\ \mathbf{0} & [\mathbf{V}_{t_2}\mathbf{V}_{t_2}^{\text{add}}] \end{bmatrix} \begin{bmatrix} \mathbf{T}_{t_1} \\ \mathbf{0} \\ \mathbf{T}_{t_2} \\ \mathbf{0} \end{bmatrix} \tag{22}$$

where $\mathbf{T}_{t_1}$ and $\mathbf{T}_{t_2}$ are original transfer matrices associated with nonleaf cluster t.

The transfer matrices and coupling matrices at other levels all remain the same as before. They are not updated until corresponding levels are reached.

B. Computation at the Nonleaf Level

After all the leaf-level computation, we obtain a matrix shown in Fig. 5, which is current $\mathbf{Z}_{\text{cur}}$ left to be factorized. All the empty blocks are zero blocks. The diagonal blocks corresponding to the unknowns that have been eliminated are identity matrices shown as $\mathbf{I}$. The nonzero blocks, shown as shaded blocks, each is of size $(k_{l+1} \times k_{l+1})$, where l is current tree level, $l+1$ is the previous tree level whose computation is finished, and k_{l+1} denotes the rank at the $(l+1)$th level. The structure shown in Fig. 5 is not only true for $l = L-1$ level, i.e., one level above the leaf level, but also true for every nonleaf level l, except that the size of each block is different at different tree levels, since k_{l+1} is different especially for electrically large analysis. The computation at a nonleaf level is performed as follows.

1) Merging and Permuting: In $\mathbf{Z}_{\text{cur}}$, whose structure is shown in Fig. 5, we permute the matrix, and merge the four blocks of a nonleaf cluster, formed between its two children clusters and themselves, to a single block, as shown by the transformation from Figs. 5 to 6(a). After this, we further permute the unknowns that have not been eliminated to the end. This results in a matrix shown in Fig. 6(b). As a result, the bottom-right block becomes the matrix that remains to be factorized. This matrix is of size $2^l(2k_{l+1}) \times 2^l(2k_{l+1})$, because it is formed between 2^l clusters at the nonleaf level l, and each cluster is of size $2k_{l+1}$. Obviously, this matrix is again an $\mathcal{H}^2$-matrix, as shown by green admissible blocks and red inadmissible blocks in Fig. 6(b). The difference with the leaf level is that each block, be its inadmissible or admissible, now is of size $2k_{l+1} \times 2k_{l+1}$. Hence, in the proposed direct solution algorithm, at each nonleaf level, the computation is performed on the matrix blocks whose size is the *rank* at that level, and hence efficient.

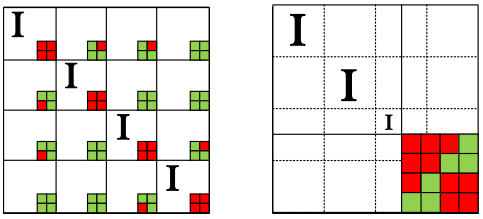


Fig. 6. (a) Merging to next level. (b) Permuting to obtain $\mathbf{Z}_{\text{cur}}$ to be factorized at next level.

Now, each admissible block is of size only $2k_{l+1} \times 2k_{l+1}$. Its expression is different from that at the leaf level. Consider an admissible block formed between clusters t and s at a nonleaf level l, as shown by the green blocks in Fig. 6(b). It has the following form:

$$(\text{Admissible block})_{t,s}^{(l)} = \mathbf{T}_t \mathbf{S}_{t,s} \mathbf{T}_s^T \tag{23}$$

where $\mathbf{T}_t$ is the transfer matrix for nonleaf cluster t, whose size is $O(2k_{l+1} \times k_l)$, $\mathbf{T}_s$ is the transfer matrix for cluster s whose size is also $O(2k_{l+1} \times k_l)$, and $\mathbf{S}_{t,s}$ is the coupling matrix whose size is $O(k_l \times k_l)$.

Since the matrix block left to be factorized after merging and permutation is again an $\mathcal{H}^2$-matrix, we can apply the same procedure performed at the leaf level, i.e., from Step 0 to Step 3, to the nonleaf level. In addition, from (23), it can also be seen clearly that now, transfer matrix $\mathbf{T}$ at a nonleaf level plays the same role as cluster basis $\mathbf{V}$ at the leaf level. Therefore, wherever $\mathbf{V}$ is used in the leaf-level computation, we replace it by $\mathbf{T}$ at a nonleaf level. In the following, we will briefly go through Steps 0–3 performed at a nonleaf level, as the essential procedure is the same as that in the leaf level.

2) Steps 0 to 3:

a) Step 0 at the nonleaf level: This step is to update transfer matrix $\mathbf{T}_i$ to account for the contributions of the fill-ins due to the elimination of previous clusters.

For cluster i, we take all the fill-in blocks associated with its admissible blocks, and use them to update transfer matrix $\mathbf{T}_i$. These fill-in blocks are shown in (14), which have been generated and stored during the factorization of previous clusters. Let them be $\mathbf{F}_{i,j_k} (k = 1, 2, \ldots, O(C_{\text{sp}}))$, where the first subscript denotes the row cluster i. Different from the fill-ins at leaf level, $\mathbf{F}_{i,j_k}$ now is of $(2k_{l+1} \times 2k_{l+1})$ size at a nonleaf level l. We then compute

$$\mathbf{G}_i = (\mathbf{I} - \mathbf{T}_i\mathbf{T}_i^H) \left(\sum\nolimits_k \mathbf{F}_{i,j_k} \mathbf{F}_{i,j_k}^H \right) (\mathbf{I} - \mathbf{T}_i\mathbf{T}_i^H)^H \tag{24}$$

where all the k admissible blocks formed by cluster i are considered. Performing an SVD on $\mathbf{G}_i$, and truncating the singular vector matrix based on prescribed accuracy $\epsilon_{\text{fill-in}}$, we obtain

$$\mathbf{G}_i \overset{\epsilon_{\text{fill-in}}}{=} \left(\mathbf{T}_i^{\text{add}}\right) \Sigma \left(\mathbf{T}_i^{\text{add}}\right)^H. \tag{25}$$

$\mathbf{T}_i^{\text{add}}$ is hence the additional column space we should supplement in the transfer matrix. The new transfer matrix $\widetilde{\mathbf{T}}_i$ can then be obtained as

$$\widetilde{\mathbf{T}}_i = \begin{bmatrix} \mathbf{T}_i & \mathbf{T}_i^{\text{add}} \end{bmatrix}. \tag{26}$$

The cost of this step for each cluster at level l is $O(k_l{}^3)$.

b) Step 1 at a nonleaf level: We compute the complementary cluster basis $\widetilde{\mathbf{T}}_i^{\perp}$, which is orthogonal to $\widetilde{\mathbf{T}}_i$. This computation is not expensive, as it costs only $O({k_l}^3)$ at level l. We then combine $\widetilde{\mathbf{T}}_i$ with $\widetilde{\mathbf{T}}_i^{\perp}$ to form $\widetilde{\mathbf{Q}}_i$ matrix as the following:

$$\widetilde{\mathbf{Q}}_i = \left[(\widetilde{\mathbf{T}}_i^{\perp})_{2k_{l+1}\times(2k_{l+1}-k_l)} \; (\widetilde{\mathbf{T}}_i)_{2k_{l+1}\times k_l} \right]. \tag{27}$$

c) Step 2 at a nonleaf level: With $\widetilde{\mathbf{Q}}_i$ computed, we build a block diagonal matrix

$$\mathbf{Q}_i = \text{diag}\{\mathbf{I}_1, \mathbf{I}_2, \ldots, \widetilde{\mathbf{Q}}_i, \ldots, \mathbf{I}_{\#\{\text{clusters at level } l\}}\} \tag{28}$$

using which we construct

$$\mathbf{Z}_{\text{cur}} = \mathbf{Q}_i^H \mathbf{Z}_{\text{cur}} \overline{\mathbf{Q}}_i \tag{29}$$

where $\mathbf{Z}_{\text{cur}}$ only consists of the block that remains to be factorized, as shown by the bottom-right matrix block in Fig. 6(b), which is not an identity. Similar to that in the leaf level, by doing so, we zero out first $(2k_{l+1} - k_l)$ rows of the admissible blocks formed by row cluster i, as well as their transpose entries in $\mathbf{Z}_{\text{cur}}$. The real computation of (29) is hence in the inadmissible blocks of nonleaf cluster i, the cost of which is $O(k_l)^3$.

d) Step 3 at a nonleaf level: We perform a partial LU factorization to eliminate the first $(2k_{l+1} - k_l)$ unknowns of cluster i in $\mathbf{Z}_{\text{cur}}$. Let this set of unknowns be denoted by i'. We obtain

$$\mathbf{Z}_{\text{cur}} = \begin{bmatrix} \mathbf{L}_{i'i'} & \mathbf{0} \\ \mathbf{Z}_{ci'}\mathbf{U}_{i'i'}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Z}}_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{i'i'} & \mathbf{L}_{i'i'}^{-1}\mathbf{Z}_{i'c} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{30}$$

where $\mathbf{F}_{i'i'} = \mathbf{L}_{i'i'}\mathbf{U}_{i'i'}$, and $\widetilde{\mathbf{Z}}_{cc}$ is a Schur complement. Each fill-in block is again either directly added, if the to-be added block is inadmissible, or stored temporarily in the corresponding admissible block. Equation (30) can further be written in short as

$$\mathbf{Z}_{\text{cur}} = \mathbf{L}_i^l \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Z}}_{cc} \end{bmatrix} \mathbf{U}_i^l. \tag{31}$$

At a nonleaf level l, each of $\mathbf{L}_{i'i'}$ and $\mathbf{U}_{i'i'}$ is of size at most $(2k_{l+1} - k_l) \times (2k_{l+1} - k_l)$. $\mathbf{Z}_{i'c}$ $(\mathbf{Z}_{ci'})$ has only C_{sp} blocks, which are also inadmissible, each of which is of size at most $2k_{l+1} \times (2k_{l+1} - k_l)$. The Schur complement involves C_{sp}^2 fill-in blocks, each of which is of size at most $2k_{l+1}$ by $2k_{l+1}$. Therefore, generating the L and U factors, as well as computing the fill-in blocks, only involves a constant number of computations, each of which scales as $O({k_l}^3)$ at a nonleaf level. The storage of the $\mathbf{L}$ and $\mathbf{U}$ factors, as well as the fill-in blocks, only takes $O({k_l}^2)$ multiplied by constant $O(C_{\text{sp}}^2)$.

3) Updating the Coupling Matrix at Current Level and the Transfer Matrix at One Level Higher: After the computation at the current nonleaf level l is finished, before proceeding to the next level, $l-1$, we need to update the coupling matrices at current level, as well as the transfer matrices at one level higher. This is because the transfer matrices have been updated at the current level.

Similar to the procedure at the leaf level, for admissible blocks without fill-ins, their coupling matrices can be readily updated by appending zeros, as shown in (19). For admissible blocks with fill-ins, their coupling matrices are updated by adding the fill-in part onto the augmented coupling matrix $\widetilde{\mathbf{S}}_{i,k}$ as

$$\widetilde{\mathbf{S}}_{i,k} = \widetilde{\mathbf{S}}_{i,k} + \widetilde{\mathbf{T}}_i^H \mathbf{F}_{i,k} \overline{\widetilde{\mathbf{T}}}_k \tag{32}$$

in which $\mathbf{F}_{i,k}$ represents the fill-in block associated with the admissible block formed between clusters i and k.

We also need to update the transfer matrices at one level higher, i.e., at the $l-1$ level. This can be readily done by adding zeros to the original transfer matrices, corresponding to the matrix block of $\mathbf{T}_i^{\text{add}}$.

C. Summary

The aforementioned computation at the nonleaf level is continued level by level until we reach the minimal level that has admissible blocks, denoted by l_0. The overall procedure can be realized by the pseudocode shown in Algorithm 1, where the detail of each step can be found from aforementioned Sections III-A and III-B.

Algorithm 1 Proposed Direct Solution of General $\mathcal{H}^2$-Matrices

```
Let Z_cur = Z
for l = L to l_0 do
  for cluster : i =1 to 2^l do
    Update cluster basis (Step 0)
    Obtain projection matrix Q_i (Step 1)
    Do Z_cur = Q_i^H Z_cur conj(Q_i) (Step 2)
    Perform partial LU Z_cur = L_i^l Z_cnt U_i^l (Step 3)
    Let Z_cur = Z_cnt
  end for
  Update coupling matrices at level l
  Update transfer matrices at level l − 1
  Permute and merge small k_l × k_l matrices to next level
end for
Do LU on the final block, Z_cur = LU.
```

The aforementioned direct solution procedure results in the following factorization of $\mathbf{Z}$:

$$\begin{aligned} \mathbf{Z} &= \mathcal{L}\mathcal{U} \\ \mathcal{L} &= \left\{ \prod_{l=L}^{l_0} \left[\left(\prod_{i=1}^{2^l} \mathbf{Q}_i^l \mathbf{L}_i^l \right) \mathbf{P}_l^T \right] \right\} \mathbf{L} \\ \mathcal{U} &= \mathbf{U} \left\{ \prod_{l=l_0}^{L} \left[\mathbf{P}_l \left(\prod_{i=2^l}^{1} \mathbf{U}_i^l \mathbf{Q}_i^{l\,T} \right) \right] \right\}. \end{aligned} \tag{33}$$

As mentioned earlier, $\mathbf{Q}_i^l$ denotes $\mathbf{Q}_i$ at level l. For leaf level, it is shown in (5), and for a nonleaf level, it is shown in (28). Each $\mathbf{Q}_i^l$ has only one block that is not identity, whose size is $2k_{l+1} \times 2k_{l+1}$ at the nonleaf level, and $leafsize \times leafsize$ at the leaf level. Each $\mathbf{L}_i^l$ is shown in (12) or (31), which has $O(C_{\text{sp}})$ blocks of *leafsize* at the leaf level, and $O(C_{\text{sp}})$ blocks of $O(k_{l+1})$ size at a nonleaf level. The same is true to $\mathbf{U}_i^l$. $\mathbf{L}$ and $\mathbf{U}$ without subscripts and superscripts are the $\mathbf{L}$ and $\mathbf{U}$ factors of the final matrix that remains to be factorized in level l_0. The nonidentity blocks in $\mathbf{L}$ and $\mathbf{U}$ are of size $O(2^{l_0}k_{l_0})$. The $\mathbf{P}_l$ matrix is a permutation matrix that merges small $k_l \times k_l$ matrices at level l to one level higher.

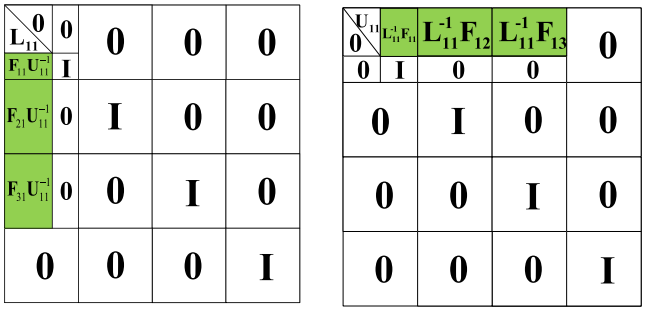


Fig. 7. (a) $\mathbf{L}_1$ matrix. (b) $\mathbf{U}_1$ matrix.

The factorization shown in (33) also results in an explicit form of $\mathbf{Z}$'s inverse, which can be written as

$$\mathbf{Z}^{-1} = \left\{\prod_{l=L}^{l_0}\left[\left(\prod_{i=1}^{2^l}\overline{\mathbf{Q}}_i^l(\mathbf{U}_i^l)^{-1}\right)\mathbf{P}_l^T\right]\right\}\mathbf{U}^{-1} \times \mathbf{L}^{-1}\left\{\prod_{l=l_0}^{L}\left[\mathbf{P}_l\left(\prod_{i=2^l}^{1}(\mathbf{L}_i^l)^{-1}\mathbf{Q}_i^{l\,H}\right)\right]\right\}. \quad (34)$$

In addition, in the proposed direct solution, one can also flexibly stop at a level before l_0 is reached. This may be desired if at a tree level, the number of zeros introduced is small as compared with the block size at that level. Although one can still proceed to the next level, the efficiency gain may be smaller than that gained if one just stops at the current level, and finishes the factorization. This typically happens at a few levels lower than the l_0 level.

IV. Proposed Matrix Solution (Backward and Forward Substitution) Algorithm

After factorization, the solution of $\mathbf{Z}x = b$ can be obtained in two steps: $\mathcal{L}y = b$, which can be called as a forward substitution, and $\mathcal{U}x = y$ called as a backward substitution. Rigorously speaking, since projection matrix $\mathbf{Q}_i^l$ and permutation matrix $\mathbf{P}_l$ are involved in $\mathcal{L}$ and $\mathcal{U}$ factors, the matrix solution here is not a conventional forward/backward substitution.

A. Forward Substitution

In this step, we solve

$$\mathcal{L}y = b. \quad (35)$$

From (34), it can be seen that (35) can be solved as

$$y = \mathcal{L}^{-1}b = \mathbf{L}^{-1}\left\{\prod_{l=l_0}^{L}\left[\mathbf{P}_l\left(\prod_{i=2^l}^{1}(\mathbf{L}_i^l)^{-1}\mathbf{Q}_i^{l\,H}\right)\right]\right\}b. \quad (36)$$

Take a three-level $\mathcal{H}^2$-matrix as an example, (36) is nothing but

$$y = \mathbf{L}^{-1}\mathbf{P}_1\mathbf{L}_4^{-1}\mathbf{Q}_4^H\mathbf{L}_3^{-1}\mathbf{Q}_3^H\mathbf{L}_2^{-1}\mathbf{Q}_2^H\mathbf{L}_1^{-1}\mathbf{Q}_1^H b. \quad (37)$$

Obviously, three basic operations are involved in the forward substitution: multiplying $(\mathbf{Q}_i^l)^H$ by a vector, multiplying $(\mathbf{L}_i^l)^{-1}$ by a vector, and $\mathbf{P}_l$-based permutation.

The algorithm of the forward substitution is shown in Algorithm 2. $(\mathbf{L}_i^l)^{-1}b$ in Step 4 of this algorithm only involves $O(C_{\text{sp}})$ MVMs of *leafsize* or rank size associated with cluster i at level l. To see this point more clearly, we can use the $\mathbf{L}_1$ matrix shown in Fig. 7(a) as an example, where $\mathbf{L}_1$ is resultant from the partial factorization of cluster 1 in an $\mathcal{H}^2$-matrix. It can be seen that $\mathbf{L}_1$ has only $O(C_{\text{sp}})$ blocks that need to be computed and stored, each of which is *leafsize* at the leaf level, or rank size at a nonleaf level l. These blocks are the inadmissible blocks formed with cluster i. For such type of $\mathbf{L}_i$, $\mathbf{L}_i^{-1}b$ can be readily computed. For $\mathbf{L}_1$ shown in Fig. 7(a), $\mathbf{L}_1^{-1}b$ is

$$\mathbf{L}_1^{-1}b = \begin{bmatrix} \begin{bmatrix} \mathbf{L}_{11}^{-1} & \mathbf{0} \\ -\mathbf{T}_{11}\mathbf{L}_{11}^{-1} & \mathbf{I} \end{bmatrix} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \begin{bmatrix} -\mathbf{T}_{21}\mathbf{L}_{11}^{-1} & \mathbf{0} \end{bmatrix} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \begin{bmatrix} -\mathbf{T}_{31}\mathbf{L}_{11}^{-1} & \mathbf{0} \end{bmatrix} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix} \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \widetilde{b}_{1,1} \\ b_{1,2} - \mathbf{T}_{11}\widetilde{b}_{1,1} \end{bmatrix} \\ b_2 - \mathbf{T}_{21}\widetilde{b}_{1,1} \\ b_3 - \mathbf{T}_{31}\widetilde{b}_{1,1} \\ b_4 \end{bmatrix} \quad (38)$$

where $\mathbf{T}_{i1} = \mathbf{F}_{i1}\mathbf{U}_{11}^{-1}$ $(i = 1, 2, 3)$, and $\widetilde{b}_{1,1} = \mathbf{L}_{11}^{-1}b_{1,1}$. Therefore, this operation actually only involves $O(C_{\text{sp}})$ MVMs related to clusters 1–3, which are those clusters that form inadmissible blocks with cluster 1, the number of which is bounded by C_{sp}. After forward substitution, the right-hand side b vector is overwritten by solution y.

Algorithm 2 Forward Substitution $b \leftarrow \mathcal{L}^{-1}b$

1: **for** $l = L$ **to** l_0 **do**
2: **for** $cluster : i = 1$ **to** 2^l **do**
3: Update b's b_i part by $b_i \leftarrow (\widetilde{\mathbf{Q}}_i^l)^H b_i$.
4: Do $b \leftarrow (\mathbf{L}_i^l)^{-1}b$
5: **end for**
6: Permute b by front multiplying with $\mathbf{P}_l$
7: **end for**
8: Do $b \leftarrow \mathbf{L}^{-1}b$

B. Backward Substitution

After finishing forward substitution, we solve

$$\mathcal{U}x = y. \quad (39)$$

From (34), it can be seen that (39) can be solved as

$$x = \mathcal{U}^{-1}y = \left\{\prod_{l=L}^{l_0}\left[\left(\prod_{i=1}^{2^l}\overline{\mathbf{Q}}_i^l\mathbf{U}_i^{l\,-1}\right)\mathbf{P}_l^T\right]\right\}\mathbf{U}^{-1}y. \quad (40)$$

For the same three-level $\mathcal{H}^2$-matrix example, (40) is nothing but

$$x = \overline{\mathbf{Q}}_1\mathbf{U}_1^{-1}\overline{\mathbf{Q}}_2\mathbf{U}_2^{-1}\overline{\mathbf{Q}}_3\mathbf{U}_3^{-1}\overline{\mathbf{Q}}_4\mathbf{U}_4^{-1}\mathbf{P}_1^T\mathbf{U}^{-1}y. \quad (41)$$

The backward substitution starts from minimal admissible block level l_0, and finishes at the leaf level, as shown in Algorithm 3. $(\mathbf{U}_i^l)^{-1}y$ in Step 5 is performed efficiently similar to how (38) is used to compute $(\mathbf{L}_i^l)^{-1}b$. There are also three basic operations in Algorithm 3: $(\mathbf{U}_i^l)^{-1}$-based MVM, $\overline{\mathbf{Q}}_i^l$-based MVM, and $\mathbf{P}_l$-based permutation. After the backward substitution is done, y is overwritten by solution x.

Algorithm 3 Backward Substitution $y \leftarrow \mathcal{U}^{-1}y$

1: Do $y \leftarrow \mathbf{U}^{-1}y$
2: **for** $l = l_0$ **to** L **do**
3: Permute y by front multiplying $\mathbf{P}_l^T$
4: **for** *cluster* $i = 2^l$ **to** 1 **do**
5: Do $y \leftarrow (\mathbf{U}_i^l)^{-1}y$
6: Update $y_i \leftarrow (\overline{\mathbf{Q}}_i^l)y_i$.
7: **end for**
8: **end for**

V. Accuracy and Complexity Analysis

In this section, we analyze the accuracy and computational complexity of the proposed direct solver.

A. Accuracy

When generating an $\mathcal{H}^2$-matrix to represent the original dense matrix, the accuracy is controlled by $\epsilon_{\mathcal{H}^2}$, which is the same as the accuracy parameter used in [16, eqs. (6), (7), (11), and (15)]. When computing the direct solution, the accuracy is controlled by $\epsilon_{\text{fill-in}}$. This is different from [9], [11], and [15], since it is directly controlled.

As can be seen from Section III, there are no approximations involved in the proposed solution procedure, except that when we update the cluster bases to account for the contribution from the fill-ins, we use (16) or (25) to truncate the singular vectors. The accuracy of this step is controlled by parameter $\epsilon_{\text{fill-in}}$, which can be set to any desired value.

B. Time and Memory Complexity

From (33), it is evident that the whole computation involves $\mathbf{Q}_i^l$, $\mathbf{L}_i^l$, and $\mathbf{U}_i^l$ for each cluster i at every level l, the storage and computation of each of which are, respectively, $O(k_l^2)$ and $O(k_l^3)$ at a nonleaf level, and constant at the leaf level. Recall that $\mathbf{Q}_i^l$ is obtained from Steps 0 and 1, and $\mathbf{L}_i^l$ and $\mathbf{U}_i^l$ are obtained from Step 3, whose underlying blocks are generated in Step 2. As for the permutation matrix $\mathbf{P}_l$, it is sparse involving only $O(2^l)$ integers to store at level l, and hence having an $O(N)$ cost in both memory and time.

From another perspective, the overall procedure of the proposed direct solution is shown in Algorithm 1. It involves $O(L)$ levels of computation. At each level, there are 2^l clusters. For each cluster, we have performed four steps of computation from Step 0, 1, 2, to 3, as shown by the tasks listed at the end of lines 4–7 in Algorithm 1. Each of these steps costs $O(k_l)^3$ for each cluster, as analyzed in Section III. After each level is done, we update coupling matrices, and transfer matrices at one-level higher, the cost of which is also $O(k_l)^3$ for every admissible block, or cluster.

The factorization and inversion complexity of the proposed direct solutions, hence, can be evaluated as

$$\textbf{Fact./Inverse Complexity} = \sum_{l=0}^{L} C_{\text{sp}}^2 2^l k_l^3 = C_{\text{sp}}^2 \sum_{l=0}^{L} 2^l k_l^3 \tag{42}$$

since at every level, there are 2^l clusters, and each cluster is associated with $O(C_{\text{sp}}^2)$ computations, and each operation costs $O(k_l^3)$. Similarly, the solution time and memory only involve $O(k_l^2)$ computations and storage for each cluster. Hence, their complexities are

$$\textbf{Solution/Mem. Complexity} = \sum_{l=0}^{L} C_{\text{sp}} 2^l k_l^2 = C_{\text{sp}} \sum_{l=0}^{L} 2^l k_l^2. \tag{43}$$

Recall that k_l is the rank at tree level l. Hence, (42) and (43) show that the overall complexity is a function of rank k_l.

For constant-rank $\mathcal{H}^2$-matrices, since k_l is a constant irrespective of matrix size, the complexity of the proposed direct solution is strictly $O(N)$ in both CPU time and memory consumption, because

For constant k_l:

$$\text{Fact./Inverse Complexity} = C_{\text{sp}}^2 k_l^3 \sum_{l=0}^{L} 2^l = O(N) \tag{44}$$

$$\text{Solution/Mem. Complexity} = C_{\text{sp}} k_l^2 \sum_{l=0}^{L} 2^l = O(N). \tag{45}$$

Constant-rank $\mathcal{H}^2$-matrices can be used to represent the integral operators arising from small or modest electrical sizes with well-controlled accuracy.

For electrodynamic analysis, to ensure a prescribed accuracy, the rank becomes a function of electrical size, and thereby tree level. Different $\mathcal{H}^2$-representations can result in different complexities, because their rank's behavior is different. Using a minimal-rank $\mathcal{H}^2$-representation, as shown by [31], the rank grows linearly with electrical size for general 3-D problems. In a VIE, k_l is hence proportional to the cubic root of matrix size at level l, because this is the electrical size at level l. Hence, for a VIE, (42) and (43) become

For k_l linearly growing with electrical size:

VIE Factorization/Inverse Complexity

$$= C_{\text{sp}}^2 \sum_{l=0}^{L} 2^l \left[\left(\frac{N}{2^l} \right)^{\frac{1}{3}} \right]^3 = O(N \log N) \tag{46}$$

VIE Solution/Memory Complexity

$$= C_{\text{sp}} \sum_{l=0}^{L} 2^l \left[\left(\frac{N}{2^l} \right)^{\frac{1}{3}} \right]^2 = O(N) \tag{47}$$

irrespective of the electrical size.

VI. Numerical Results

In order to demonstrate the accuracy and low computational complexity of the proposed direct solution of general $\mathcal{H}^2$-matrices, we apply it to solve VIEs for electromagnetic analysis. Both circuit and large-scale scattering problems are simulated. The VIE formulation we use is based on [34] with SWG vector bases for expanding electric flux density in each tetrahedral element. First, the scattering from a dielectric sphere, whose analytical solution is known, is simulated to validate the proposed direct solver. Then, a variety of

TABLE I

DIRECT SOLUTION ERROR MEASURED BY RELATIVE RESIDUAL ϵ_{rel} OF THE DIELECTRIC SPHERE AS A FUNCTION OF $\epsilon_{\text{fill-in}}$

$\epsilon_{fill-in}$	Ex. 1	Ex. 2	Ex. 3	Ex. 4
10^{-6}	0.01%	0.002%	0.003%	0.011%
10^{-5}	0.04%	0.006%	0.01%	0.04%
10^{-4}	0.11%	0.015%	0.024%	0.1%

large-scale examples involving over one million unknowns are simulated on a single CPU core to examine the accuracy and complexity of the proposed direct solver. The $\mathcal{H}^2$-matrix for each example is constructed based on the method described in [14]–[16]. The accuracy parameter $\epsilon_{\mathcal{H}^2}$ used is 10^{-3} for the $\mathcal{H}^2$-matrix construction. The accuracy parameter $\epsilon_{\text{fill-in}}$ used in the direct solution is varied to examine the error controllability of the proposed direct solution. The computer used has an Intel Xeon CPU E5-2690 v2 running at 3.00 GHz, and only a *single core* is employed for carrying out the computation to examine the computational complexity of the solver. Frobenius norm is used whenever norm is calculated.

A. Scattering From a Dielectric Sphere

We simulate a dielectric sphere of radius a with different electric sizes k_0a and dielectric constant ϵ_r. The incident electric field is a normalized $-z$ propagating plane wave polarized along the x-direction. This incident field is also used in the following examples. During the $\mathcal{H}^2$-matrix construction, $leafsize = 25$ and $\eta = 1$ are used. The accuracy parameter in the direct solution is set to be $\epsilon_{\text{fill-in}} = 10^{-6}$. In Fig. 8, the radar cross section (RCS) (RCS in dBsm) of the sphere is plotted as a function of θ for zero azimuth for $k_0a = 0.408, \epsilon_r = 36$; $k_0a = 0.408, \epsilon_r = 4$; $k_0a = 0.816, \epsilon_r = 4$; and $k_0a = 1.632, \epsilon_r = 4$, respectively. They all show good agreement with the analytical Mie Series solution, which validates the proposed direct solution.

We have also investigated the accuracy of the proposed direct solution as a function of the accuracy-control parameter $\epsilon_{\text{fill-in}}$. In Table I, we list the relative residual error for all the four sphere examples simulated using $\epsilon_{\text{fill-in}} = 10^{-4}, 10^{-5}$, and 10^{-6}, respectively. Examples 1–4 in Table I correspond to Fig. 8(a)–(d), respectively. As can be seen from Table I, the smaller the $\epsilon_{\text{fill-in}}$ value, the better the solution accuracy, verifying the error controllability of the proposed direct solution. We have also compared RCS obtained from this direct solver against the analytical Mie Series solution for $\epsilon_{\text{fill-in}} = 10^{-4}, 10^{-5}$, and 10^{-6}. The difference only appears after the fourth or fifth digit.

B. Large-Scale Dielectric Slab

1) Simulation Using the Proposed Direct Solution: We then simulate a dielectric slab with $\epsilon_r = 2.54$ at 300 MHz, which is also simulated in [14] and [15]. The thickness of the slab is fixed to be $0.1\lambda_0$. The width and length are simultaneously increased from $4\lambda_0$, $8\lambda_0$, $16\lambda_0$, to $32\lambda_0$. With a mesh size of $0.1\lambda_0$, the resultant N ranges from 22 560 to 1 434 880 for this suite of slab structures. The *leafsize* is chosen to be 25 and $\eta = 1$. The $\epsilon_{\text{fill-in}}$ is set to be 10^{-2}, 10^{-4}, and 10^{-6},

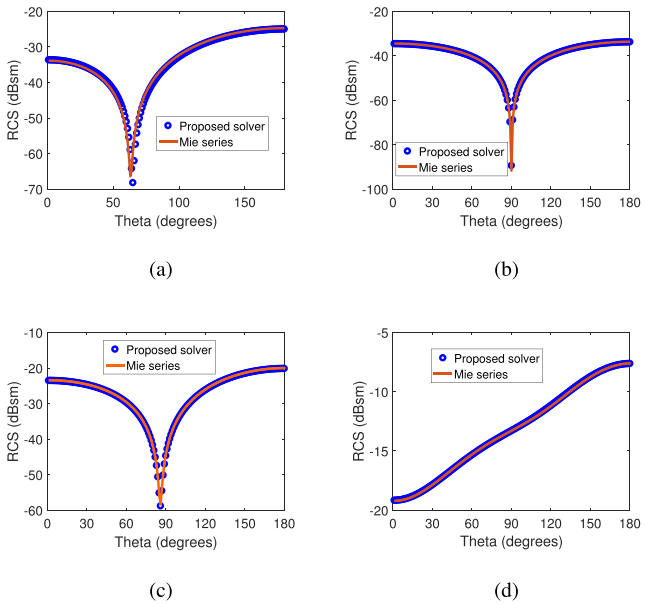


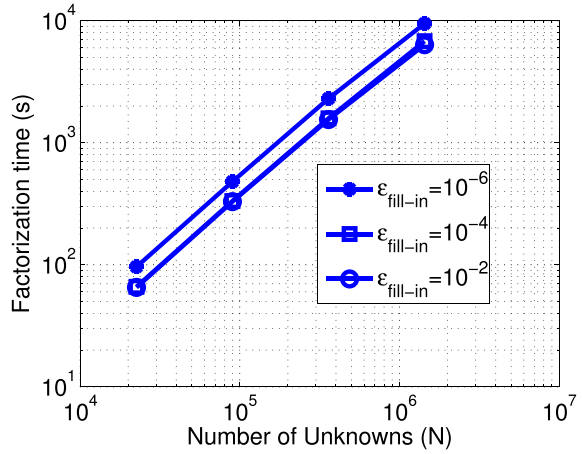
Fig. 8. Simulated RCS of a dielectric sphere for different electric sizes and dielectric constants. (a) $k_0a = 0.408$, $\epsilon_r = 36.0$. (b) $k_0a = 0.408$, $\epsilon_r = 4.0$. (c) $k_0a = 0.816$, $\epsilon_r = 4.0$. (d) $k_0a = 1.632$, $\epsilon_r = 4.0$.

respectively, to examine the solution accuracy, computational complexity, and error controllability of the proposed direct solution.

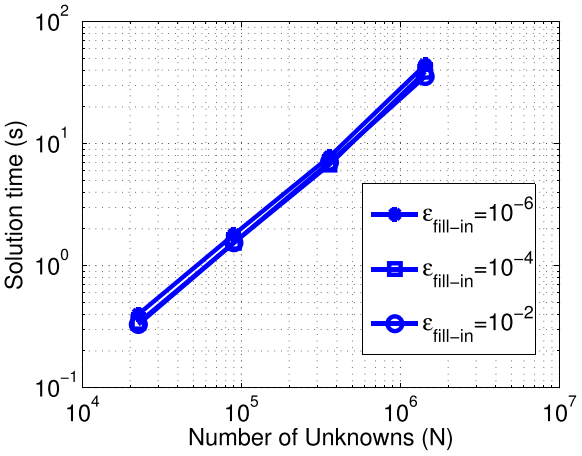
Notice that this example represents the 2-D problem characteristics. Based on [31], the rank's growth rate with electric size is lower than linear, and being a square root of the log-linear of the electric size. Substituting such a rank's growth rate into the complexity analysis shown in (42) and (43), we will obtain linear complexity in both memory and time. The rank numerically found is 12, 17, 28, and 49, respectively, for the four slab examples, whose growth rate with electrical size is clearly no greater than linear.

In Fig. 9(a), we plot the factorization time with respect to N, for all three different choices of $\epsilon_{\text{fill-in}}$. It is clear that the smaller the $\epsilon_{\text{fill-in}}$ value, the larger the factorization time. However, the complexity remains the same as linear regardless of the choice of $\epsilon_{\text{fill-in}}$. In addition, the factorization time is not increased much by decreasing $\epsilon_{\text{fill-in}}$. This is because $\epsilon_{\text{fill-in}}$ is used to append the original cluster bases with additional ones to capture the fill-in's contribution. When the number of additional cluster bases added is small, the cost is still dominated by the original number of cluster bases. The solution time and memory cost are plotted in Fig. 9(b) and (c), respectively. Obviously, both scale linearly with the number of unknowns. The error of the proposed direct solution is measured by computing the relative residual $\epsilon_{\text{rel}} = ||\mathbf{Z}_{\mathcal{H}^2}x - b||/||b||$, where $\mathbf{Z}_{\mathcal{H}^2}$ is the input $\mathcal{H}^2$-matrix to be solved. The relative residual ϵ_{rel} of the proposed direct solution is listed in Table II as a function of $\epsilon_{\text{fill-in}}$. Excellent accuracy can be observed in the entire unknown range. Furthermore, the accuracy can be controlled by $\epsilon_{\text{fill-in}}$, and overall smaller $\epsilon_{\text{fill-in}}$ results in better accuracy.

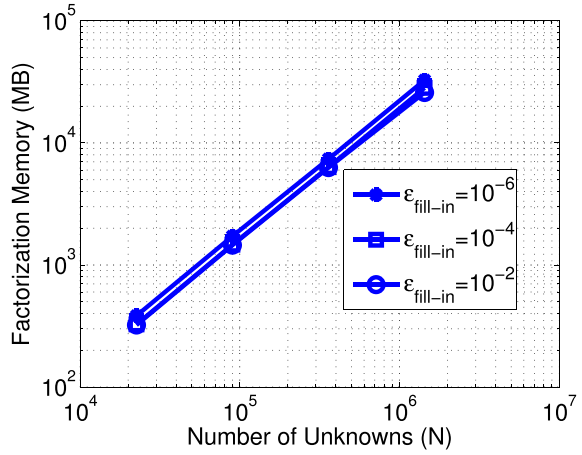
2) Comparison With Direct Solvers in [14]–[16]: Since this example is also simulated in our previous work [14]–[16], we have compared the two direct solvers in CPU run time



(a)



(b)



(c)

Fig. 9. Solver performance of a large-scale dielectric slab for different choices of $\epsilon_{\text{fill-in}}$. (a) Factorization time versus N. (b) Solution time versus N. (c) Memory versus N.

and accuracy. $\epsilon_{\text{fill-in}} = 10^{-5}$ is used. For a fair comparison, we set up this solver to solve the same $\mathcal{H}^2$-matrix solved in [16], and also use the same computer. As can be seen from Table III, the proposed new direct solution takes much less time than that of [14]–[16]. The speedup is more than one order of magnitude in large examples. Even though the factorization time is shown, as can be seen from (34), the inversion time in the proposed algorithm is similar to factorization time. In addition, because of a direct error control, the error of the proposed solution is also much less than that of [14]–[16].

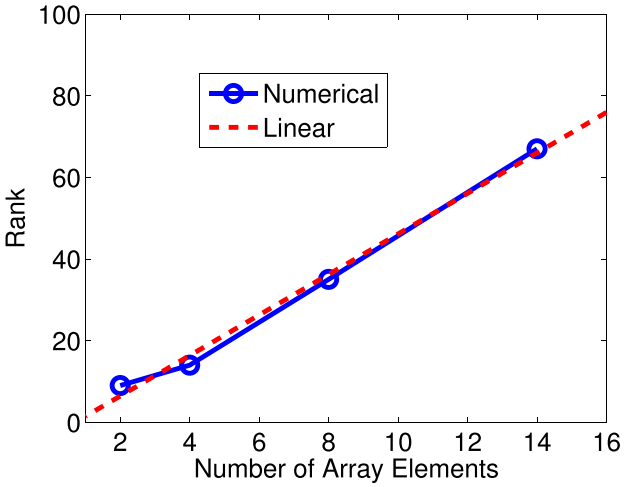


Fig. 10. Rank of the dielectric cube array example versus number of array elements.

TABLE II
DIRECT SOLUTION ERROR MEASURED BY RELATIVE RESIDUAL, ϵ_{rel}, FOR THE DIELECTRIC SLAB EXAMPLE

$\epsilon_{fill-in}$	10^{-2}	10^{-4}	10^{-6}
ϵ_{rel} $(N=22,560)$	0.86%	0.19%	0.03%
ϵ_{rel} $(N=89,920)$	1.32%	0.25%	0.057%
ϵ_{rel} $(N=359,040)$	3.59%	1.88%	0.66%
ϵ_{rel} $(N=1,434,880)$	2.68%	1.05%	0.56%

TABLE III
PERFORMANCE COMPARISON BETWEEN THIS SOLVER WITH $\epsilon_{\text{fill-in}} = 10^{-5}$ AND [14]–[16] FOR THE DIELECTRIC SLAB EXAMPLE

N	89,920	359,040
Factorization (s) [This]	353	1,662
Solution (s) [This]	1.68	7.07
Inversion (s) [14]–[16]	2.75e+03	1.65e+04
Relative Residual [This]	0.18%	0.40%
Inverse Error [14]–[16]	3.9%	7.49%

C. Large-Scale Array of Dielectric Cubes

1) Simulation With the Proposed Direct Solution: Next, we simulate a large-scale array of dielectric cubes at 300 MHz. The relative permittivity of the cube is $\epsilon_r = 4.0$. Each cube is of size $0.3\lambda_0 \times 0.3\lambda_0 \times 0.3\lambda_0$. The distance between adjacent cubes is kept to be $0.3\lambda_0$. The number of the cubes is increased along the x-, y-, and z-direction simultaneously from 2 to 14, thus producing a 3-D cube array from $2\times2\times2$ to $14\times14\times14$ elements. The number of unknowns N is, respectively, 3024, 24 192, 193 536, and 1 037 232 for these arrays. The *leafsize* is 25 and $\eta = 1$. The $\epsilon_{\text{fill-in}}$ is chosen as 10^{-3}, 10^{-4}, and 10^{-5}, respectively.

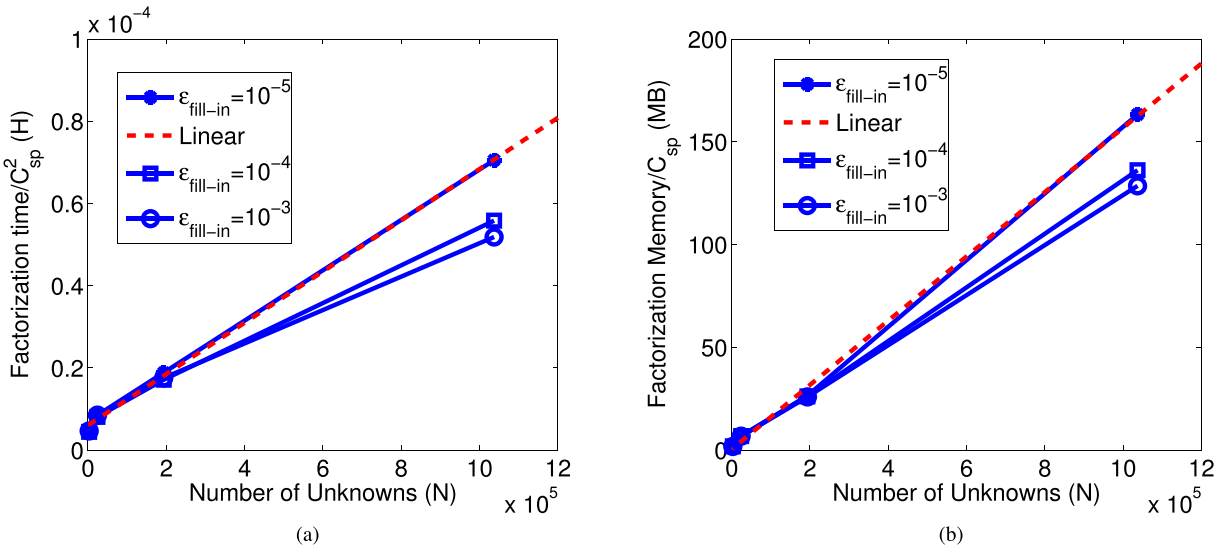


Fig. 11. Solver performance of a suite of cube arrays for different choices of $\epsilon_{\text{fill-in}}$. (a) Factorization time versus N. (b) Memory versus N.

TABLE IV
C_{sp} AS A FUNCTION OF N FOR THE DIELECTRIC CUBE ARRAY

N	3,024	24,192	193,536	378,000	1,037,232
C_{sp}	16	42	95	327	270

For the cubic growth of unknowns for 3-D problems, we observe that constant C_{sp} is not saturated until in the order of millions of unknowns, as can be seen from Table IV. It is thus important to analyze the performances of the proposed direct solver as (Memory or Solution time)$/C_{\text{sp}}$ and (Factorization time)$/C_{\text{sp}}^2$, respectively. As for the rank of this example, we plot the maximal rank of each unknown case in Fig. 10. It can be clearly seen that the rank's growth rate with the number of array elements, thereby electrical size, is no greater than linear. For such a growth rate, the resultant complexities are given in (42) and (43). In Fig. 11(a) and (b), we plot the direct factorization time normalized by C_{sp}^2, and the storage cost normalized with C_{sp} with respect to N. As can be seen, their scaling rate with N agrees very well with our theoretical complexity analysis regardless of the choice of $\epsilon_{\text{fill-in}}$. The relative residual ϵ_{rel} of the proposed direct solution is listed in Table V for this example, which again reveals excellent accuracy and error controllability of the proposed direct solution.

2) Comparison With Direct Solvers in [14]–[16]: Since this example with dielectric constant $\epsilon_r = 2.54$ is also simulated in our previous work [16], we have compared the two direct solvers in CPU run time and accuracy. As can be seen from Table VI, the proposed new direct solution again takes much less time, while achieving better accuracy.

D. On-Chip Interconnects

We have also performed a full-wave VIE simulation of on-chip interconnects. A suite of large-scale on-chip bus structures from a 4×4 to 64×64 is simulated at 30 GHz, with an x-polarized incident electric field. The conductivity

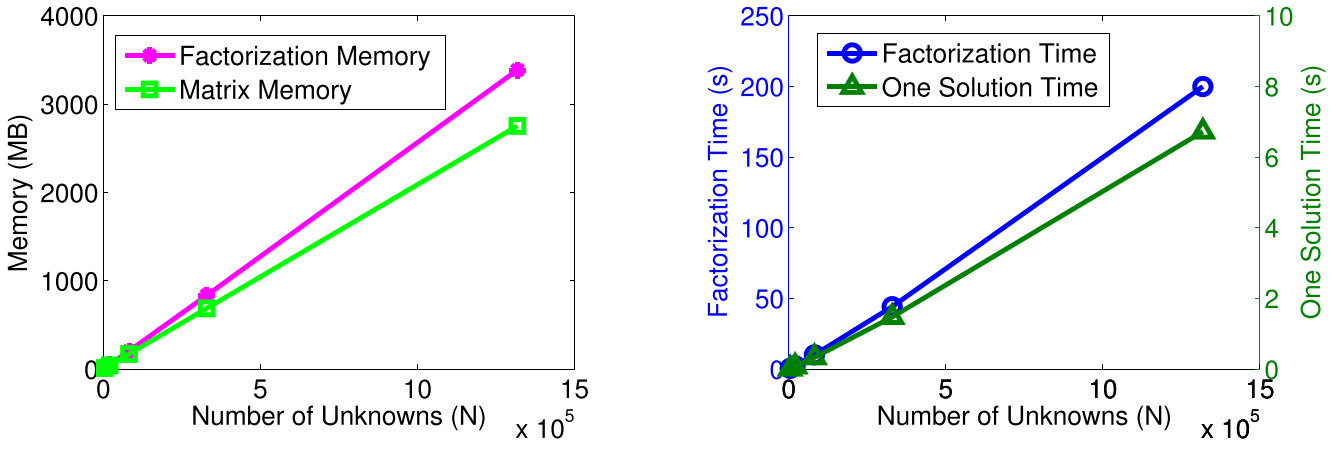
TABLE V
DIRECT SOLUTION ERROR MEASURED BY RELATIVE RESIDUAL FOR THE CUBE ARRAY EXAMPLE

$\epsilon_{fill-in}$	10^{-3}	10^{-4}	10^{-5}
ϵ_{rel} $(N=3,024)$	0.27%	0.14%	0.09%
ϵ_{rel} $(N=24,192)$	0.74%	0.31%	0.15%
ϵ_{rel} $(N=193,536)$	2.22%	0.88%	0.36%
ϵ_{rel} $(N=1,037,232)$	5.32%	4.02%	1.3%

TABLE VI
PERFORMANCE COMPARISON WITH [14]–[16] FOR THE DIELECTRIC CUBE ARRAY EXAMPLE

N	3024	24,192	193,536
Factorization (s) [This]	4.24	52.6	503
Solution (s) [This]	0.03	0.3	3.07
Inversion (s) [14]–[16]	13.95	444.01	1.15e+04
Relative Residual [This]	0.04%	0.13%	0.13%
Inverse Error [14]–[16]	0.9%	1.73%	3.03%

of the metal is 5.8×10^7 S/m. The dimensions of each bus are 1 μm $\times$ 1 μm $\times$ 20 μm. The horizontal distance between the centers of two neighboring buses is 20 μm. And the vertical distance is 40 μm. Each bus is discretized into 322 unknowns. The total number of unknowns ranges from 5152 to 1 318 912. For the $\mathcal{H}^2$ tree construction, we set *leafsize* to be 25 and $\eta = 1$, with $\epsilon_{\mathcal{H}^2} = 10^{-4}$. The accuracy parameter for controlling the direct solution is set to be $\epsilon_{\text{fill-in}} = 10^{-4}$. In Fig. 12(a), we plot the memory cost of both storing the original matrix (labeled matrix), and that for factorizing the matrix (labeled factorization). In Fig. 12(b), we plot both factorization time and solution time for one right-hand side as a function of N. Clear linear complexities can be observed in both CPU time and memory consumption. The over one-million unknown case is factorized in about 200 s, and the solution for one right-hand side only takes 7 s. The accuracy



(a) (b)

Fig. 12. Simulation of a suite of lossy on-chip buses. (a) Memory of original matrix and memory required for factorization as a function of N. (b) Time complexity.

TABLE VII
DIRECT SOLUTION ERROR MEASURED BY RELATIVE RESIDUAL ϵ_{rel} FOR THE FULL-WAVE VIE INTERCONNECT SIMULATION EXAMPLE

N	5152	20,608	82,432	329,728	1,318,912
ϵ_{rel}	0.029%	0.029%	0.029%	0.029%	0.029%

TABLE VIII
DIRECT SOLUTION ERROR MEASURED AGAINST BRUTE-FORCE LU SOLUTIONS FOR THE FULL-WAVE VIE INTERCONNECT SIMULATION EXAMPLE

$\epsilon_{fill-in}$	10^{-2}	10^{-4}	10^{-6}
Solution Error ($N=5,152$)	0.0228%	0.0188%	0.0182%
Solution Error ($N=20,608$)	0.0228%	0.0186%	0.018%

of the proposed direct solver is assessed by relative residual and listed in Table VII. Excellent accuracy can be observed in the entire unknown range.

We have also compared the accuracy of this fast direct solution against that of a brute-force LU factorization. Certainly, this comparison can only be made for smaller number of unknowns because of the high complexity of the brute-force LU. Three different $\epsilon_{\text{fill-in}}$ values are examined, and the error measured by $||x - x_{\text{ref}}||/||x_{\text{ref}}||$ is listed in Table VIII, where x_{ref} is directly solved from the original dense matrix in exact arithmetic LU, x is from this fast solver, and Frobenius norm is used. Again, excellent accuracy is observed.

VII. CONCLUSION

In this paper, we develop fast factorization and inversion algorithms for general $\mathcal{H}^2$-matrices. The key features of this new direct solution are its directly controlled accuracy and low computational complexity. The entire direct solution does not involve any approximation, except that the fill-in block is compressed to update the cluster basis. However, this compression is well controlled by accuracy parameter $\epsilon_{\text{fill-in}}$. For constant-rank $\mathcal{H}^2$-matrices, the proposed direct solution has an $O(N)$ complexity in both time and memory. For rank growing with the electrical size linearly, the proposed direct solution has an $O(N\log N)$ complexity in factorization and inversion, and $O(N)$ in solution time and memory when solving VIEs. The proposed direct solution has been applied to solve electrically large VIEs whose kernel is oscillatory and complex-valued, which has been difficult to solve. Millions of unknowns are directly solved on a single core CPU in fast CPU run time. Comparisons with state-of-the-art $\mathcal{H}^2$-based direct VIE solvers have demonstrated the accuracy of this new direct solution as well as significantly improved computational efficiency. Being a generic direct solution to an $\mathcal{H}^2$-matrix, this paper can also be applied to solve other IEs and partial differential equations.

REFERENCES

[1] W. Hackbusch, "A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices," *Computing*, vol. 62, no. 2, pp. 89–108, 1999.

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

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

[4] S. Börm, *Efficient Numerical Methods for Non-local Operators: $\mathcal{H}^2$-Matrix Compression Algorithms, Analysis*. Zurich, Switzerland: Eur. Math. Soc., 2010.

[5] V. Rokhlin, "Diagonal forms of translation operators for the Helmholtz equation in three dimensions," *Appl. Comput. Harmon. Anal.*, vol. 1, p. 8293, Dec. 1993.

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

[7] S. Börm, "H^2-matrix arithmetics in linear complexity," *Computing*, vol. 77, pp. 1–28, Feb. 2006.

[8] W. Chai, D. Jiao, and C. C. Koh, "A direct integral-equation solver of linear complexity for large-scale 3D capacitance and impedance extraction," in *Proc. 46th ACM/EDAC/IEEE Design Automat. Conf.*, Jul. 2009, pp. 752–757.

[9] 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 Techn.*, vol. 59, no. 10, pp. 2404–2421, Oct. 2011.

[10] 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. Adv. Packag.*, vol. 33, no. 4, pp. 794–803, Nov. 2010.

[11] W. Chai and D. Jiao, "Direct matrix solution of linear complexity for surface integral-equation based impedance extraction of complicated 3-D structures," *Proc. IEEE*, vol. 101, no. 2, pp. 372–388, Feb. 2013.

[12] W. Chai and D. Jiao, "Linear-complexity direct and iterative integral equation solvers accelerated by a new rank-minimized H^2-representation for large-scale 3-D interconnect extraction," *IEEE Trans. Microw. Theory Techn.*, vol. 61, no. 8, pp. 2792–2805, Aug. 2013.

[13] S. Omar and D. Jiao, "A linear complexity direct volume integral equation solver for full-wave 3-D circuit extraction in inhomogeneous materials," *IEEE Trans. Microw. Theory Techn.*, vol. 63, no. 3, pp. 897–912, Mar. 2015.

[14] S. Omar and D. Jiao, "$O(N)$ iterative and $O(N\log N)$ direct volume integral equation solvers for large-scale electrodynamic analysis," in *Proc. Int. Conf. Electromagn. Adv. Appl. (ICEAA)*, Aug. 2014, pp. 593–596.

[15] D. Jiao and S. Omar, "Minimal-rank H^2-matrix based iterative and direct volume integral equation solvers for large-scale scattering analysis," in *Proc. IEEE Int. Symp. Antennas Propag.*, Jul. 2015, pp. 1–2.

[16] S. Omar and D. Jiao. (Mar. 2017). "$O(N)$ iterative and $O(N\log N)$ fast direct volume integral equation solvers with a minimal-rank H^2-representation for large-scale 3-D electrodynamic analysis." [Online]. Available: https://arxiv.org/abs/1703.01175

[17] P. G. Martinsson and V. Rokhlin, "A fast direct solver for scattering problems involving elongated structures," *J. Comput. Phys.*, vol. 221, pp. 288–302, Jan. 2007.

[18] R. J. Adams, Y. Xu, X. Xu, J. Choi, 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.

[19] 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.

[20] E. Winebrand and A. Boag, "A multilevel fast direct solver for EM scattering from quasi-planar objects," in *Proc. Int. Conf. Electromagn. Adv. Appl.*, 2009, pp. 640–643.

[21] L. Greengard, D. Gueyffier, P.-G. Martinnson, and V. Rokhlin, "Fast direct solvers for integral equations in complex three-dimensional domains," *Acta Numer.*, vol. 18, pp. 243–275, May 2009.

[22] A. Heldring, J. M. Rius, J. M. Tamayo, J. Parron, and E. Ubeda, "Multiscale compressed block decomposition for fast direct solution of method of moments linear system," *IEEE Trans. Antennas Propag.*, vol. 59, no. 2, pp. 526–536, Feb. 2011.

[23] A. Freni, P. De Vita, P. Pirinoli, L. Matekovits, and G. Vecchi, "Fast-factorization acceleration of MoM compressive domain-decomposition," *IEEE Trans. Antennas Propag.*, vol. 59, no. 12, pp. 4588–4599, Dec. 2011.

[24] Y. Brick, V. Lomakin, and A. Boag, "Fast direct solver for essentially convex scatterers using multilevel non-uniform grids," *IEEE Trans. Antennas Propag.*, vol. 62, no. 8, pp. 4314–4324, Aug. 2014.

[25] H. Guo, Y. Liu, J. Hu, and E. Michielssen. (2016). "A butterfly-based direct integral equation solver using hierarchical lu factorization for analyzing scattering from electrically large conducting objects." [Online]. Available: https://arxiv.org/abs/1610.00042

[26] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Fast algorithms for hierarchically semiseparable matrices," *Numer. Linear Algebra Appl.*, vol. 17, no. 6, pp. 953–976, 2010.

[27] A. B. Manić, B. M. Notaroš, F. H. Rouet, and X. S. Li, "Efficient EM scattering analysis based on MoM, HSS direct solver, and RRQR decomposition," in *Proc. IEEE Int. Symp. Antennas Propag.*, Jul. 2015, pp. 1660–1661.

[28] M. Ma and D. Jiao, "HSS-matrix based fast direct volume integral equation solver for electrodynamic analysis," in *Proc. IEEE Int. Conf. Wireless Inf. Technol. Syst. (ICWITS)*, Mar. 2016, pp. 1–2.

[29] M. Ma and D. Jiao, "New HSS-factorization and inversion algorithms with exact arithmetic for efficient direct solution of large-scale volume integral equations," *Proc. IEEE Int. Symp. Antennas Propag.*, Jun. 2016, pp. 1567–1568.

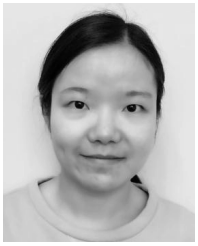
[30] A. B. Manić, A. P. Smull, B. M. Notaroš, F. H. Rouet, and X. S. Li, "Efficient scalable parallel higher order direct MoM-SIE method with hierarchically semiseparable structures for 3-D scattering," *IEEE Trans. Antennas Propag.*, vol. 65, no. 5, pp. 2467–2478, Feb. 2017.

[31] W. Chai and D. Jiao, "Theoretical study on the rank of integral operators for broadband electromagnetic modeling from static to electrodynamic frequencies," *IEEE Trans. Compon., Packag. Manuf. Technol.*, vol. 3, no. 12, pp. 2113–2126, Dec. 2013.

[32] S. Ambikasaran and E. Darve. (2014). "The inverse fast multipole method." [Online]. Available: https://arxiv.org/abs/1407.1572

[33] P. Coulier, H. Pouransari, and E. Darve. (2016). "The inverse fast multipole method: Using a fast approximate direct solver as a preconditioner for dense linear systems." [Online]. Available: https://arxiv.org/abs/1407.1572

[34] D. H. Schaubert, D. R. Wilton, and A. W. Glisson, "A tetrahedral modeling method for electromagnetic scattering by arbitrarily shaped inhomogeneous dielectric bodies," *IEEE Trans. Antennas Propag.*, vol. AP-32, no. 1, pp. 77–85, Jan. 1984.



Miaomiao Ma (GS'16) received the B.S. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, in 2014. She is currently pursuing the Ph.D. degree at Purdue University, West Lafayette, IN, USA.

She is also with the School of Electrical and Computer Engineering, Purdue University, as a member of the On-Chip Electromagnetics Group. Her current research interests include computational electromagnetics, fast and high-capacity numerical methods, and scattering analysis.

Ms. Ma was a recipient of the Honorable Mention Award of the IEEE International Symposium on Antennas and Propagation in 2016. She was also the recipient of the Best Student Paper Award of the IEEE International Conference on Wireless Information Technology and Systems and Applied Computational Electromagnetics in 2016.

Dan Jiao (M'02–SM'06–F'16) received the Ph.D. degree in electrical engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2001.

She was with the Technology Computer-Aided Design (CAD) Division, Intel Corporation, Santa Clara, CA, USA, until 2005, as a Senior CAD Engineer, a Staff Engineer, and a Senior Staff Engineer. In 2005, she joined Purdue University, West Lafayette, IN, USA, as an Assistant Professor with the School of Electrical and Computer Engineering, where she is currently a Professor. She has authored 3 book chapters and over 260 papers in refereed journals and international conferences. Her current research interests include computational electromagnetics, high-frequency digital, analog, mixed-signal, and RF integrated circuit design and analysis, high-performance VLSI CAD, modeling of microscale and nanoscale 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 bioelectromagnetics.

Dr. Jiao was the recipient of the 2013 S. A. Schelkunoff Prize Paper Award of the IEEE Antennas and Propagation Society, which recognizes the Best Paper published in the IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION during the previous year. She has been named a University Faculty Scholar by Purdue University since 2013. She was among the 85 engineers selected throughout the nation for the National Academy of Engineering's 2011 U.S. Frontiers of Engineering Symposium. She was a recipient of the 2010 Ruth and Joel Spira Outstanding Teaching Award, the 2008 National Science Foundation CAREER Award, the 2006 Jack and Cathie Kozik Faculty Start Up Award (which recognizes an Outstanding New Faculty Member of the School of Electrical and Computer Engineering, Purdue University), the 2006 Office of Naval Research Award under the Young Investigator Program, the 2004 Best Paper Award presented at the Intel Corporation's Annual Corporate-Wide Technology Conference (Design and Test Technology Conference) for her work on generic broadband model of high-speed circuits, the 2003 Intel Corporation's Logic Technology Development (LTD) Divisional Achievement Award, the Intel Corporation's Technology CAD Divisional Achievement Award, the 2002 Intel Corporation's Components Research the Intel Hero Award (Intel-wide she was the tenth recipient), the Intel Corporation's LTD Team Quality Award, and the 2000 Raj Mittra Outstanding Research Award presented by the University of Illinois at Urbana–Champaign. She has served as a reviewer for many IEEE journals and conferences.