Direct Solution of General $\mathcal{H}^2$-Matrices With Controlled Accuracy and Concurrent Change of Cluster Bases for Electromagnetic Analysis

Miaomiao Ma, *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 with controlled accuracy. In this paper, we develop a new direct solution for general $\mathcal{H}^2$-matrices. The new solution possesses not only explicitly controlled accuracy but also a full change of cluster basis to efficiently account for the updates to the original matrix during the direct solution procedure. The change of cluster basis is performed concurrently with the direct solution, without increasing its computational complexity. Comparisons with the state-of-the-art direct solutions have demonstrated a much reduced CPU run time of the new solution, in addition to a significantly improved accuracy, which is also controllable. Rapid and accurate direct solutions of millions of unknowns have been obtained on a single CPU core.**

***Index Terms*— Fast direct solvers, $\mathcal{H}^2$-matrix.**

I. Introduction

MANY fast solvers have been developed to solve large-scale electromagnetic problems [1]–[12]. To meet real-world modeling and simulation challenges, there is a continued need to accelerate the computation as well as increase the accuracy and modeling capabilities of existing computational electromagnetic methods.

The $\mathcal{H}^2$-matrix is a general mathematical framework [13]–[17] for compact representation and efficient computation of dense matrices. Such a framework can be utilized to develop faster solvers of reduced computational complexity for electromagnetic analysis. In [18]–[21], the dense system matrix resulting from an integral equation (IE)-based analysis is first represented as an $\mathcal{H}^2$-matrix with controlled accuracy. Fast inversion and LU factorization algorithms are then developed to directly solve the $\mathcal{H}^2$-matrix in $O(N)$ complexity. Despite a significantly reduced complexity, the cluster bases used to represent the original dense matrix are also used to represent the inverse as well as the LU factors of the matrix. Formatted additions and multiplications are performed, whose accuracy can only be indirectly controlled. This is different from formatted additions and multiplications performed in an $\mathcal{H}$-matrix, whose format is not nested and does not involve nested cluster bases. In an $\mathcal{H}^2$-matrix, the format involves both the partition of the original matrix into admissible and inadmissible blocks, and a set of nested cluster bases used to represent admissible blocks in a nested way. When the cluster bases of the original matrix are also used to represent LU factors or inverse, the accuracy of a direct solution cannot be directly controlled.

Recently, in [22], a new fast direct solution algorithm with controlled accuracy is developed for solving general $\mathcal{H}^2$-matrices. In this algorithm, the additions and multiplications are performed as they are without using formatted operations. Each operation is either exact or controlled by a prescribed accuracy. The cluster bases of the original $\mathcal{H}^2$-matrix are changed during factorization based on accuracy. However, they are changed by appending new ones to account for the fill-in's contributions. The resulting rank, i.e., the number of cluster bases can only be higher than the original one instead of lower. If the cluster bases can be changed based on the updated matrix obtained during the direct solution procedure and with prescribed accuracy, the resulting solver can be potentially more efficient. Despite such an advantage, it is difficult to completely change the cluster bases during the direct solution procedure, since the original nested structure can be ruined, and the computational complexity would become very high.

In this paper, we develop an efficient direct solution algorithm to overcome the aforementioned challenge. The cluster bases of the original $\mathcal{H}^2$-matrix are completely changed to a new set during the factorization procedure based on accuracy. Meanwhile, the complexity of the overall algorithm is not increased. As a result, the updates to the original matrix during the direct solution procedure are efficiently and accurately represented by the new cluster bases. The proposed new direct solver has been applied to both static capacitance extraction and full-wave scattering analysis. A clear $O(N)$ complexity is observed in factorization, solution time, memory, and with a strictly controlled accuracy for constant-rank $\mathcal{H}^2$-matrices. For variable-rank $\mathcal{H}^2$-matrices such as those commonly encountered in electrically large problems, the complexity is also

Manuscript received October 31, 2018; revised February 23, 2019 and April 24, 2019; accepted April 25, 2019. Date of publication May 29, 2019; date of current version June 4, 2019. This work was supported in part by NSF under Award 1619062 and in part by DARPA under Award FA8650-18-2-7847. This paper is an expanded version from the International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization, Reykjavik, Iceland, August 8–10. *(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.2019.2914391

low and can be analytically derived based on the rank's behavior for a given problem. The solver is also shown to outperform the state-of-the-art direct solvers in both accuracy and efficiency.

The basic idea of this paper has been presented in [23]–[25]. In this paper, we have significantly expanded our conference papers to complete the algorithm development. To be more specific, in [23]–[25], we use the original admissible blocks together with fill-ins to update the cluster bases. However, when clusters are eliminated one by one during the proposed direct solution, the admissible blocks formed between the cluster being eliminated and those already eliminated have been changed at both leaf and nonleaf levels. They are different from those in the original matrix. It is challenging to use instantaneously changed admissible blocks to update cluster basis while keeping the complexity to be linear. In this paper, we successfully overcome this difficulty and develop an efficient algorithm to perform a concurrent change of cluster basis during the factorization procedure. The details of this algorithm are given in Section IV. In addition, we have demonstrated the performance of the proposed new direct solution by a large number of numerical experiments. Detailed comparisons with the state-of-the-art direct solvers in both accuracy and efficiency are also performed.

II. Preliminaries

In this section, we review the background of this paper.

A. $\mathcal{H}^2$-Matrix

An $\mathcal{H}^2$-matrix [13]–[15], [17] is generally stored in a tree structure, with each node in the tree called a cluster. This tree can be obtained by recursively dividing the entire unknown set into two subsets until the number of unknowns in each subset is no greater than leafsize, a predefined constant. Using the row and column trees, the original matrix can be partitioned into multilevel admissible and inadmissible blocks, by checking the following admissibility condition level by level between a cluster t in the row tree and a cluster s in the column tree:

$$\max\{\text{diam}(\Omega_t), \text{diam}(\Omega_s)\} \leq \eta \text{dist}(\Omega_t, \Omega_s). \tag{1}$$

Here, Ω_t (Ω_s) denotes the geometrical support of the unknown set t (s), diam$\{\cdot\}$ is the Euclidean diameter of a set, dist$\{\cdot, \cdot\}$ denotes the Euclidean distance between two sets, and η is a positive parameter that can be used to control the admissibility condition. An $\mathcal{H}^2$-matrix example is shown in Fig. 1. An admissible block in an $\mathcal{H}^2$-matrix is represented as

$$\mathbf{Z}_{t,s} = (\mathbf{V}_t)_{\#t\times k}(\mathbf{S}_{t,s})_{k\times k}(\mathbf{V}_s)^T_{\#s\times k} \tag{2}$$

where $\mathbf{V}_t$ ($\mathbf{V}_s$) is called cluster basis of cluster t (s), and $\mathbf{S}_{t,s}$ is called coupling matrix. 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

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

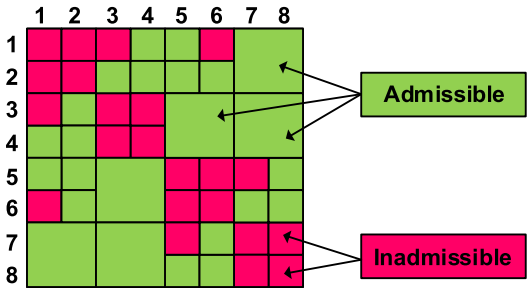


Fig. 1. Illustration of an $\mathcal{H}^2$-matrix.

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. Because of the nested relationship of cluster bases, the cluster bases only need to be stored for leaf clusters. For nonleaf clusters, transfer matrices are stored. In fact, the matrix structure resulting from a fast multipole method (FMM) or a multilevel fast multipole algorithm (MLFMA) is also an $\mathcal{H}^2$-matrix. This structure is very general, suitable for representing the IE operators encountered in electromagnetic analysis. Methods described in [19] and [29] can be employed to construct an $\mathcal{H}^2$-matrix representation efficiently.

B. Difference Between the Error Control of an $\mathcal{H}$-Algorithm and an $\mathcal{H}^2$-Algorithm

The error in the $\mathcal{H}$-matrix-based algorithm, such as the $\mathcal{H}$-LU decomposition based on the formatted addition and multiplication, can be controlled. However, this is not true for the formatted addition and multiplication performed in an $\mathcal{H}^2$-matrix-based algorithm such as [13]–[15] and [17].

In an $\mathcal{H}$-matrix, there is no concept of *nested cluster bases*. Every admissible block is represented by an $\mathbf{AB}^T$ form, where $\mathbf{A}$ and $\mathbf{B}$ are generated solely by prescribed accuracy. $\mathbf{A}$ and $\mathbf{B}$ of an admissible block do not need to have any relationship with their children or parent blocks. That is also the reason why an $\mathcal{H}$-LU algorithm does not have a linear complexity since the $\mathcal{H}$-based representation is not a nested representation.

In contrast, in an $\mathcal{H}^2$-matrix, the admissible block is represented by (2), where the cluster basis $\mathbf{V}_t$ is required to be nested. In an $\mathcal{H}^2$-matrix arithmetic, formatted multiplications and additions are performed by keeping the cluster bases of the original matrix in the target matrix. For example, when performing $\mathbf{A}_{t,r}\mathbf{B}_{r,s} = \mathbf{C}_{t,s}$, if $\mathbf{C}_{t,s}$ is admissible, then its row cluster basis is kept the same as $\mathbf{A}$'s row cluster basis, and its column cluster basis is chosen the same as $\mathbf{B}$'s column cluster basis. The accuracy of this operation is clearly not controlled. For example, $\mathbf{A} = \mathbf{F}$ is a full matrix, and $\mathbf{B} = \mathbf{V}_r\mathbf{S}_{r,s}(\mathbf{V}_s)^T$ is an admissible block; so we have $\mathbf{C} = \mathbf{F}\mathbf{V}_r\mathbf{S}_{r,s}(\mathbf{V}_s)^T$. Obviously, $\mathbf{C}$'s row cluster basis is not $\mathbf{V}_r$ any more but $\mathbf{F}\mathbf{V}_r$. Similarly, at a nonleaf level, we can run into the computations of multiplying a nonleaf block by an admissible block. Again, the product $\mathbf{C}$ should be changed in its row cluster basis.

However, if one makes a brute-force change of the cluster basis, then the nested property of the original cluster bases is completely ruined. This is very different from an $\mathcal{H}$-matrix. In an $\mathcal{H}$-matrix computation, there is no need to make the computation nested, and the low-rank form can be generated solely based on accuracy. In [22], which is accuracy controlled, the factorization algorithm has to be completely changed, as compared to [19], in order to ensure accuracy while keeping the computation nested.

C. Integral Equations for Electromagnetic Analysis

For capacitance extraction of interconnects in a single material, we consider charges on the conducting surfaces producing electric potential, which results in the following IE:

$$\phi(\vec{r}) = \int_S \frac{\rho_s(\vec{r'})}{\epsilon} g(\vec{r}, \vec{r'}) ds' \tag{3}$$

where ρ_s is the surface charge density, ϵ is the permittivity, and $g(\vec{r}, \vec{r'})$ is the free-space Green's function. A method-of-moments-based solution of (3) results in the following dense system of equations:

$$\mathbf{Z}q = v \tag{4}$$

where q vector consists of charges on each discretization patch, and right-hand side vector v is composed of the potential assigned to each conductor. In nonuniform dielectrics, (3) is augmented by the electric potential due to equivalent surface charges at the dielectric discontinuity.

For electrodynamic scattering analysis, the most general IE formulation would be a volume IE-based formulation, which accounts for arbitrary dielectric and conductive inhomogeneity. Considering an interconnect exposed to an external field $\vec{E}^i(\vec{r})$, based on the volume equivalence principle, the equivalent volume current $\vec{J} = j\omega(\bar{\epsilon} - \epsilon_0)\vec{E}$ radiating in the background material produces the scattered field. Thus, the total field $\vec{E}$ at any point $\vec{r}$ is equal to the sum of the incident field and the scattered field

$$\vec{E}(\vec{r}) + \nabla\phi^s(\vec{r}) + j\omega A^s(\vec{r}) = \vec{E}^i(\vec{r}) \tag{5}$$

which is expressed in the following form of the volume IE [26]:

$$\frac{\vec{D}(\vec{r})}{\bar{\epsilon}(\vec{r})} - \mu\omega^2 \int_V \kappa(\vec{r'})\vec{D}(\vec{r'}) g(\vec{r}, \vec{r'}) dv' - \nabla \int_V \nabla' \cdot \left(\frac{\kappa(\vec{r'})\vec{D}(\vec{r'})}{\epsilon_0} \right) g(\vec{r}, \vec{r'}) dv' = \vec{E}^i(\vec{r}) \tag{6}$$

where Green's function $g(\vec{r}, \vec{r'}) = e^{-jk_0|\vec{r}-\vec{r'}|}/4\pi|\vec{r}-\vec{r'}|$, ω is the angular frequency, k_0 is the free-space wavenumber, and κ is the contrast ratio defined as

$$\kappa(\vec{r}) = \frac{\bar{\epsilon}(\vec{r}) - \epsilon_0}{\bar{\epsilon}(\vec{r})} \tag{7}$$

and $\vec{D}(\vec{r})$ is

$$\vec{D}(\vec{r}) = \bar{\epsilon}(r)\vec{E} \tag{8}$$

which is related to the equivalent volume current by $\vec{J}(\vec{r}) = j\omega\kappa(\vec{r})\vec{D}(\vec{r})$. From the third term in (6), it can be seen that

$$\phi^s(\vec{r}) = -\int_V \nabla' \cdot \left(\frac{\kappa(\vec{r'})\vec{D}(\vec{r'})}{\epsilon_0} \right) g(\vec{r}, \vec{r'}) dv' \tag{9}$$

which can be further written as

$$\phi^s(\vec{r}) = \int_V \frac{\rho_v(\vec{r'})}{\epsilon_0} g(\vec{r}, \vec{r'}) dv' + \int_S \frac{\rho_s(\vec{r'})}{\epsilon_0} g(\vec{r}, \vec{r'}) ds' \tag{10}$$

where ρ_v is the density of equivalent volume charges and ρ_s is the density of the equivalent surface charges at the material discontinuity, where $\kappa(\vec{r})$ is discontinuous. In this paper, the vector basis functions employed to expand $\vec{D}$ are the Schaubert–Wilton–Glisson (SWG) bases [26].

III. Proposed Direct Solution

To make the proposed algorithm easier for understanding, we use the $\mathcal{H}^2$-matrix shown in Fig. 1, denoted by $\mathbf{Z}$, as an example to illustrate the overall procedure. However, the algorithm presented here is generic, applicable to any $\mathcal{H}^2$-matrix. In Fig. 1, green blocks are admissible and red ones are inadmissible.

A. Computation at Leaf Level

Denote the tree level of the leaf level to be L, the computations at level $l = L$ include the following steps.

1) Let $\mathbf{Z}_{cur} = \mathbf{Z}$. For each leaf cluster i, the following holds.
   a) Change the original cluster basis of $\mathbf{V}_i$ to a new cluster basis $\widetilde{\mathbf{V}}_i$ to account for fill-ins, for $i > 1$.
   b) Compute the complementary basis $\widetilde{\mathbf{V}}_i^{\perp}$ and build $\widetilde{\mathbf{Q}}_i = [\widetilde{\mathbf{V}}_i^{\perp} \quad \widetilde{\mathbf{V}}_i]$.
   c) Compute $\mathbf{Q}_i^H \mathbf{Z}_{cur} \overline{\mathbf{Q}}_i$ to introduce zeros into admissible blocks, which only involves the computation of $O(C_{sp})$ inadmissible blocks. Here, $\mathbf{Q}_i$ is block diagonal, whose ith diagonal block is $\widetilde{\mathbf{Q}}_i$, and other blocks are identity matrices.
   d) Let $\mathbf{Z}_{cur} = \mathbf{Q}_i^H \mathbf{Z}_{cur} \overline{\mathbf{Q}}_i$. Perform partial LU factorization of $\mathbf{Z}_{cur}$ to eliminate the first $\#i - k_i$ unknowns in cluster i, where k_i is the rank of $\widetilde{\mathbf{V}}_i$.
2) Update the coupling matrix of each admissible block at the leaf level.
3) Merge and permute the matrix to prepare for $\mathbf{Z}_{cur}$ for next-level computation.

The details of the aforementioned steps can be found in [22], and the overall procedure is shown in Fig. 2. In this section, we focus on the difference with [22], which is in the first step where the bases are changed. In [22], the cluster bases are updated by appending new ones to the original set. Here, we propose an efficient algorithm to completely change cluster bases as follows.

To change cluster basis $\mathbf{V}_i$ to accommodate the matrix updates, we first put all the i-related admissible matrix blocks into a single matrix denoted by $\mathbf{Z}_i$. Since the cluster bases need to be nested, the i-related matrix blocks include not only those admissible blocks formed at the same tree level as that

of cluster i but also those admissible blocks formed by the parents of i at all upper levels. Thus, we have

$$\mathbf{Z}_i = \left[\mathbf{Z}_{i,j_1}\ \mathbf{Z}_{i,j_2}\ \ldots,\ \mathbf{Z}^{L-1}_{ip,jp_1}\ \mathbf{Z}^{L-1}_{ip,jp_2} \ldots,\ \mathbf{Z}^{l_0}_{ip,jp_1}\ \mathbf{Z}^{l_0}_{ip,jp_2}\ \ldots\right] \quad (11)$$

where $\mathbf{Z}_{i,j_1}\ \mathbf{Z}_{i,j_2}\ldots$ are the admissible blocks at the tree level of cluster i, and the rest are from upper levels; the superscript represents tree level, and L is the leaf level, while l_0 is the minimum level that has admissible blocks; the ip denotes cluster i's parent clusters at level l, and $jp_1, jp_2, \ldots$ stand for the column cluster indices of the admissible blocks formed with row cluster ip.

To construct a new cluster basis for cluster i, we compute the Gram matrix of $\mathbf{Z}_i$, which is

$$\mathbf{Z}_{i,2} = \mathbf{Z}_i \mathbf{Z}_i^H. \quad (12)$$

The computation of $\mathbf{Z}_{i,2}$ appears to be expensive. In this paper, we develop a fast linear-time tree traversal algorithm to obtain $\mathbf{Z}_{i,2}$ efficiently. The detailed procedure will be elaborated in Section IV.

After computing $\mathbf{Z}_{i,2}$, we add the contributions of the fill-in blocks associated with cluster i at i's tree level, obtaining

$$\widetilde{\mathbf{Z}}_{i,2} = \mathbf{Z}_{i,2} + \sum\nolimits_m \mathbf{F}_{i,j_m}\mathbf{F}^H_{i,j_m} \quad (13)$$

in which m is the fill-in block number associated with cluster i, whose number is bounded by constant C_{sp}. Note that although we use symbol $\mathbf{F}$ to denote a fill-in block, it is not a full-matrix block. In fact, it is a low-rank block. This is because a fill-in block has a format of $\mathbf{Z}_{ci'}\mathbf{U}^{-1}_{i'i}\mathbf{L}^{-1}_{i'i'}\mathbf{Z}_{i'c}$. Since $\mathbf{Z}_{ci'}$ has a column rank of $\#i - k$ and $\mathbf{Z}_{i'c}$ has a row rank of the same, the fill-in block is low rank in nature without using any further truncation.

We then perform a singular value decomposition (SVD) on $\widetilde{\mathbf{Z}}_{i,2}$, and the resultant singular vector matrix truncated based on accuracy ϵ_{acc} is the new cluster basis for cluster i, denoted by $\widetilde{\mathbf{V}}_i$. Thus,

$$\widetilde{\mathbf{Z}}_{i,2} \overset{\epsilon_{\text{acc}}}{\approx} \widetilde{\mathbf{V}}_i \Sigma \widetilde{\mathbf{V}}_i^H. \quad (14)$$

The size of $\widetilde{\mathbf{Z}}_{i,2}$ is leafsize at leaf level. Hence, (14) can be obtained in constant complexity for each cluster using SVD. This step is reflected in Fig. 2(c), where the light blue blocks are turned into green admissible blocks for cluster 2.

B. Computation at Non-Leaf Levels

At a nonleaf level l, the matrix to be factorized is of size $2^l(2k_{l+1})$ by $2^l(2k_{l+1})$ as shown by the bottom right block in Fig. 2(h). This is because it is formed between 2^l clusters at the nonleaf level l, and each block is of size $2k_{l+1} \times 2k_{l+1}$. Obviously, this matrix is again an $\mathcal{H}^2$-matrix, as shown by green admissible blocks and red inadmissible blocks in Fig. 2(h). The difference with the leaf level is that each block at this level, 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.

Take an admissible block formed between clusters t and s at nonleaf level $l = L-1$ as an example. These blocks are shown by the green blocks in Fig. 2(h). Originally, this block has a form of $\mathbf{V}_t\mathbf{S}_{t,s}\mathbf{V}_s^T$. Due to the left multiplication with $\mathbf{Q}^H_{t_1}$ and $\mathbf{Q}^H_{t_2}$, and the right multiplication with $\overline{\mathbf{Q}}_{s_1}$ and $\overline{\mathbf{Q}}_{s_2}$, where $t_{1,2}$ and $s_{1,2}$ are the two children's of t and s, respectively, this admissible block has been zeroed out in some of its rows and columns, leaving the following part only:

$$(\text{Admissible block})^{(L-1)}_{t,s} = \mathbf{T}^{\text{new}}_t \mathbf{S}_{t,s}\left(\mathbf{T}^{\text{new}}_s\right)^T \quad (15)$$

where

$$\mathbf{T}^{\text{new}}_t = \begin{bmatrix}\widetilde{\mathbf{V}}^H_{t_1}\mathbf{V}_{t_1}\mathbf{T}_{t_1}\\ \widetilde{\mathbf{V}}^H_{t_2}\mathbf{V}_{t_2}\mathbf{T}_{t_2}\end{bmatrix} = \begin{bmatrix}\mathbf{B}^T_{t_1}\mathbf{T}_{t_1}\\ \mathbf{B}^T_{t_2}\mathbf{T}_{t_2}\end{bmatrix} \quad (16)$$

in which

$$\mathbf{B}_i = \mathbf{V}_i^T\overline{\mathbf{V}}_i \quad (17)$$

is the projection of the new leaf cluster basis of cluster i generated at the leaf level onto the original cluster basis of cluster i. As can be seen from (16), $\mathbf{B}^T_i$ can be viewed as a modified cluster basis at the leaf level.

For an admissible block at a level l higher than $L-1$, it has the following form:

$$(\text{Admissible block})^l_{t,s} = \mathbf{T}^{\text{new}}_t\mathbf{T}^{L-2}_t \ldots \mathbf{T}^l_t\mathbf{S}_{t,s}\left(\mathbf{T}^l_s\right)^T \ldots \left(\mathbf{T}^{L-2}_s\right)^T\left(\mathbf{T}^{\text{new}}_s\right)^T \quad (18)$$

where $\mathbf{T}$ without superscript *new* is the original transfer matrix, and the integer index in the superscript denotes the tree level of the transfer matrix.

If we view the current nonleaf level $L-1$ being computed as the leaf level of the tree that remains to be factorized, then from (15) and (18), it can be seen that $\mathbf{T}^{\text{new}}_t$ is the new leaf-level cluster basis, while the transfer matrices at upper levels remain the same as before. We also note that the new leaf-level cluster basis $\mathbf{T}^{\text{new}}_t$ is of dimension $O(2k_{l+1} \times k_l)$, whereas $\mathbf{S}_{t,s}$ is of size $O(k_l \times k_l)$.

The above is for nonleaf level $L-1$. For an arbitrary nonleaf level l_c, we start the computation from the following bottom-level (we can view it as current leaf level) admissible blocks:

$$(\text{Admissible block})^{(l_c)}_{t,s} = \mathbf{T}^{\text{new}}_t\mathbf{S}_{t,s}\left(\mathbf{T}^{\text{new}}_s\right)^T \quad (19)$$

where

$$\mathbf{T}^{\text{new}}_t = \begin{bmatrix}\widetilde{\mathbf{T}}^{\text{new},H}_{t_1}\mathbf{T}^{\text{new}}_{t_1}\mathbf{T}_{t_1}\\ \widetilde{\mathbf{T}}^{\text{new},H}_{t_2}\mathbf{T}^{\text{new}}_{t_2}\mathbf{T}_{t_2}\end{bmatrix} = \begin{bmatrix}\mathbf{B}^T_{t_1}\mathbf{T}_{t_1}\\ \mathbf{B}^T_{t_2}\mathbf{T}_{t_2}\end{bmatrix} \quad (20)$$

in which

$$\mathbf{B}_i = \left(\mathbf{T}^{\text{new}}_i\right)^T\overline{\widetilde{\mathbf{T}}}^{\text{new}}_i \quad (21)$$

is the projection of the new cluster basis of cluster i generated at the previous level (l_c+1 level) onto the original cluster basis of cluster i at the previous level (l_c+1 level). Similar to (18), for an admissible block at a level l higher than l_c, it has the following form:

$$(\text{Admissible block})^l_{t,s} = \mathbf{T}^{\text{new}}_t\mathbf{T}^{l_c-1}_t \ldots \mathbf{T}^l_t\mathbf{S}_{t,s}\left(\mathbf{T}^l_s\right)^T \ldots \left(\mathbf{T}^{l_c-1}_s\right)^T\left(\mathbf{T}^{\text{new}}_s\right)^T \quad (22)$$

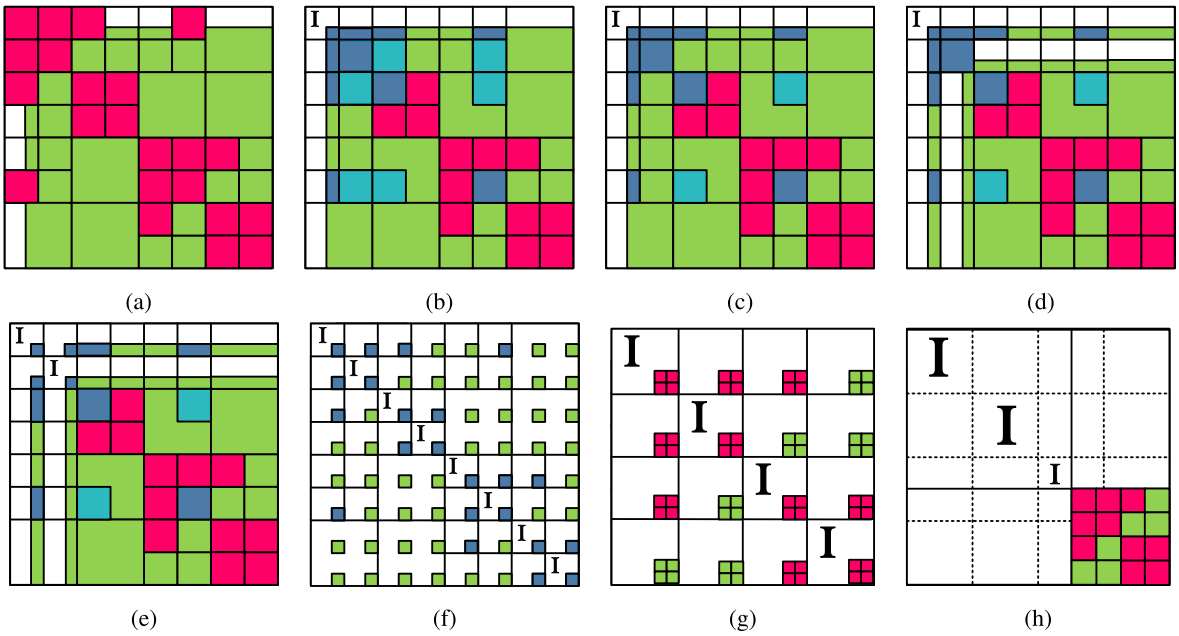


Fig. 2. Illustration of the overall procedure of the proposed direct solution. (a) $\mathbf{Q}_1^H \mathbf{Z} \overline{\mathbf{Q}}_1$. (b) $\mathbf{Z}_{cur}$ after partial LU of cluster 1 with fill-in blocks marked in blue. (c) Fill-in blocks of cluster 2 turned green after cluster basis update. (d) $\mathbf{Q}_2^H \mathbf{Z}_{cur} \overline{\mathbf{Q}}_2$. (e) $\mathbf{Z}_{cur}$ after partial LU of cluster 2. (f) $\mathbf{Z}_{cur}$ left to be factorized after leaf-level computation. (g) Merging to next level. (h) Permuting to obtain $\mathbf{Z}_{cur}$ to be factorized at next level.

where $\mathbf{T}$ without superscript *new* is the original transfer matrix.

Since, at every level, we are factorizing an $\mathcal{H}^2$-matrix, the computation at a nonleaf level l follows what is done at the leaf level, which is summarized in the following.

1) For each cluster i, the following holds.
   a) Change transfer matrix $\mathbf{T}_i^{\text{new}}$ to a new $\widetilde{\mathbf{T}}_i$ to account for fill-ins (algorithm described in Section IV).
   b) Compute the complementary basis $\widetilde{\mathbf{T}}_i^{\perp}$ and build $\widetilde{\mathbf{Q}}_i = [\widetilde{\mathbf{T}}_i^{\perp} \quad \widetilde{\mathbf{T}}_i]$.
   c) Compute $\mathbf{Q}_i^H \mathbf{Z}_{cur} \overline{\mathbf{Q}}_i$ to introduce zeros into the admissible blocks of i, which only involves the computation of $O(C_{sp})$ blocks of rank size.
   d) Let $\mathbf{Z}_{cur} = \mathbf{Q}_i^H \mathbf{Z}_{cur} \overline{\mathbf{Q}}_i$. Perform a partial LU factorization of $\mathbf{Z}_{cur}$ to eliminate the first $(2k_{l+1} - k_l)$ unknowns in cluster i, where k_l is the rank of $\widetilde{\mathbf{T}}_i$.
2) Update the coupling matrix of each admissible block at current level.
3) Merge and permute the matrix to prepare for $\mathbf{Z}_{cur}$, the matrix that remains to be factorized.

The aforementioned computation is continued level by level until we reach the minimal level $l = l_0$ that has admissible matrix blocks.

C. Overall Factorization

The overall procedure results in the following factorization of $\mathbf{Z}$:

$$\begin{aligned} \mathbf{Z} &= \mathcal{L}\mathcal{U}, \\ \mathcal{L} &= \left\{ \prod_{l=L}^{l_0} \left[\left(\prod_{i=1}^{2^l} \mathbf{Q}_i^l \mathbf{L}_i^l \right) \mathbf{P}_l^T \right] \right\} \mathbf{L}, \\ \mathcal{U} &= \mathbf{U} \left\{ \prod_{l=l_0}^{L} \left[\mathbf{P}_l \left(\prod_{i=2^l}^{1} \mathbf{U}_i^l \mathbf{Q}_i^{l\,T} \right) \right] \right\} \end{aligned} \tag{23}$$

where $\mathbf{Q}_i^l$ denotes $\mathbf{Q}_i$ at level l. 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 $\mathbf{L}_i^l$ has $O(C_{sp})$ blocks of leafsize at the leaf level, and $O(C_{sp})$ 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 at level l_0. The nonidentity blocks in $\mathbf{L}$ and $\mathbf{U}$ are of size $O(2^{l_0} k_{l_0})$ by $O(2^{l_0} k_{l_0})$. For $l_0 = 1$, it is simply $2k_{l_0}$ by $2k_{l_0}$. 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. The factorization shown in (23) also results in an explicit form of $\mathbf{Z}$'s inverse. We can solve the linear equation $\mathbf{Z}x = B$ in two ways. One is to use the inverse to multiply right-hand side B. The other is to perform a forward and backward substitution based on (23), as shown in [22].

IV. Algorithm for Concurrent Change of Cluster Bases During Matrix Factorization

In this section, we detail the proposed algorithm on how to change the cluster basis concurrently with the matrix factorization, so that the update to the original matrix during factorization can be accurately and efficiently represented.

A. Concurrent Change of Cluster Bases During the Leaf-Level Computation

At the leaf level, for each cluster, we assemble a $\mathbf{Z}_i$ matrix as shown in (11), which is repeated in the following for convenience of explanation:

$$\mathbf{Z}_i = \left[\mathbf{Z}_{i,j_1} \, \mathbf{Z}_{i,j_2} \, \ldots \, \mathbf{Z}_{ip,jp_1}^{L-1} \, \mathbf{Z}_{ip,jp_2}^{L-1} \, \ldots \, \mathbf{Z}_{ip,jp_1}^{l_0} \, \mathbf{Z}_{ip,jp_2}^{l_0} \, \ldots \right]. \tag{24}$$

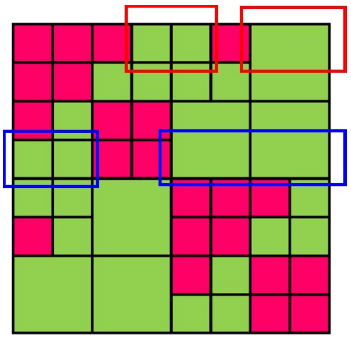


Fig. 3. Illustration of $\mathbf{Z}_i$ for $i = 1$ and $i = 4$.

This is nothing but all the admissible blocks formed by cluster i at its tree level as well as those in i's parent levels. In Fig. 3, we give two examples: $\mathbf{Z}_i$ for cluster $i = 1$ highlighted in red rectangular box and that for cluster $i = 4$ highlighted in blue box. In [23]–[25], we used the admissible blocks in the original matrix to construct $\mathbf{Z}_i$. In this case, $\mathbf{Z}_{i,j_m}$ and $\mathbf{Z}^l_{ip,jp_m}$ in (24) have the following expressions:

$$\mathbf{Z}_{i,j_m} = \mathbf{V}_i \mathbf{S}_{i,j_m} \mathbf{V}^T_{j_m}, \quad m = 1, 2, \ldots, O(C_{sp}) \tag{25}$$

$$\mathbf{Z}^l_{ip,jp_m} = \mathbf{V}_i \mathbf{T}_i^{L-1} \ldots \mathbf{T}^l_i \mathbf{S}^l_{ip,jp_m} \mathbf{V}^T_{jp_m}, \quad l = L-1, L-2, \ldots, l_0 \tag{26}$$

where $\mathbf{T}^l_i$ denotes the transfer matrix associated with cluster i or its parent cluster at level l.

We then compute Gram matrix, $\mathbf{Z}_{i,2}$, which is

$$\mathbf{Z}_{i,2} = \mathbf{Z}_i \mathbf{Z}^H_i. \tag{27}$$

Due to the nested property of admissible blocks as can be seen from (26), $\mathbf{Z}_{i,2}$ can be efficiently obtained through a top-down tree traversal procedure. To explain, substituting (25) and (26) into (24), and then (27), we obtain

$$\mathbf{Z}_{i,2} = \mathbf{V}_i \mathbf{S}^i_{\text{sum}} \mathbf{V}^H_i \tag{28}$$

where

$$\mathbf{S}^i_{\text{sum}} = \begin{cases} \widetilde{\mathbf{S}}^i_{\text{sum}}, & l(i) = l_0 \\ \widetilde{\mathbf{S}}^i_{\text{sum}} + \mathbf{T}_i \mathbf{S}^{ip}_{\text{sum}} \mathbf{T}_i{}^H, & l_0 < l(i) \le L \end{cases} \tag{29}$$

in which $l(i)$ denotes the tree level of cluster i, ip is the immediate parent cluster of i, $\mathbf{T}_i$ denotes the transfer matrix between i and ip, l_0 is the minimum level that has admissible blocks, and

$$\widetilde{\mathbf{S}}^i_{\text{sum}} = \sum_{j=1}^{O(C_{sp})} \mathbf{S}_{i,j} \mathbf{B}_j \mathbf{B}^H_j \mathbf{S}^H_{i,j} \tag{30}$$

with

$$\mathbf{B}_j \mathbf{B}^H_j = \mathbf{V}^T_j \overline{\mathbf{V}}_j = \mathbf{I} \tag{31}$$

for unitary cluster basis $\mathbf{V}_j$. Obviously, $\widetilde{\mathbf{S}}^i_{\text{sum}}$ includes all of the admissible blocks at i's own level, and $\mathbf{S}_{i,j}$ are coupling matrices of these blocks. In contrast, nontilted $\mathbf{S}^i_{\text{sum}}$ includes both $\widetilde{\mathbf{S}}^i_{\text{sum}}$ and the aggregated contribution from i's parent admissible blocks, as can be seen from (29).

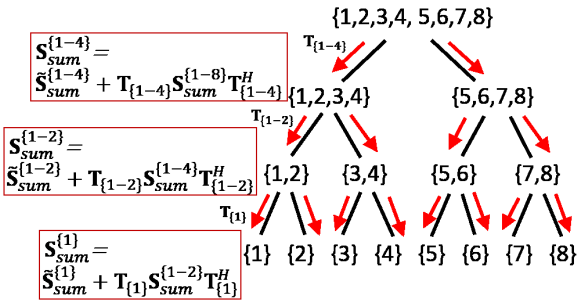


Fig. 4. Illustration of top-down tree traversal for computing $\mathbf{S}^i_{\text{sum}}$.

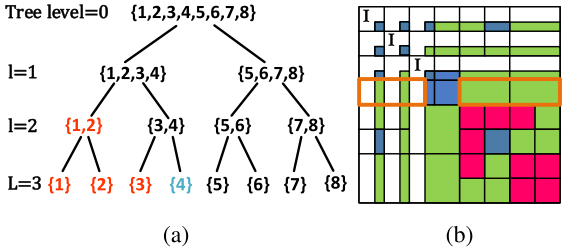


Fig. 5. (a) $\mathcal{H}^2$-tree with eliminated and uneliminated nodes. (b) $\mathbf{Z}_i$ matrix (orange box) for $i = 4$.

To compute (28) efficiently, we need to obtain $\mathbf{S}^i_{\text{sum}}$ for each cluster i, be it nonleaf or leaf. To do so, as can be seen from (29), we need to first compute $\widetilde{\mathbf{S}}^i_{\text{sum}}$ for each cluster i. $\widetilde{\mathbf{S}}^i_{\text{sum}}$ can be obtained for all clusters i in linear time for constant-rank $\mathcal{H}^2$-matrices, as the computational cost of (30) is $O(C_{sp})k^3$ for each cluster, and there are in total $O(N)$ clusters. After obtaining $\widetilde{\mathbf{S}}^i_{\text{sum}}$, based on (29), the nontilted $\mathbf{S}^i_{\text{sum}}$ for all nonleaf and leaf clusters can be obtained by performing a top-down traversal of the cluster tree from level l_0 down to the leaf level. This procedure is shown in Fig. 4. First, starting from level l_0 (which is the level of cluster $i = \{1-8\}$ in Fig. 4), for a cluster i at this level, $\mathbf{S}^i_{\text{sum}} = \widetilde{\mathbf{S}}^i_{\text{sum}}$, since there are no parent-level admissible blocks. In Fig. 4, $\mathbf{S}^{\{1-8\}}_{\text{sum}} = \widetilde{\mathbf{S}}^{\{1-8\}}_{\text{sum}}$. Such an $\mathbf{S}^i_{\text{sum}}$ is then split to its two children clusters, which is to perform $\mathbf{T}_i \mathbf{S}^{ip}_{\text{sum}} \mathbf{T}_i{}^H$ for its left and right child clusters i, respectively. As a result, $\mathbf{S}^i_{\text{sum}}$ for the two children clusters can be obtained. As can be seen from Fig. 4, $\mathbf{S}^{\{1-4\}}_{\text{sum}}$ is obtained by adding $\widetilde{\mathbf{S}}^{\{1-4\}}_{\text{sum}}$ with $\mathbf{T}_{\{1-4\}} \mathbf{S}^{\{1-8\}}_{\text{sum}} \mathbf{T}_{\{1-4\}}{}^H$. Such a procedure continues until we reach the leaf level. Since each split and addition operation of $(+\mathbf{T}_i \mathbf{S}^{ip}_{\text{sum}} \mathbf{T}_i{}^H)$ costs $O(k^3)$ and there are $O(N)$ clusters in the tree, the whole process of obtaining $\mathbf{S}^i_{\text{sum}}$ after $\widetilde{\mathbf{S}}^i_{\text{sum}}$ is computed and also costs $O(N)$ operations.

After computing $\mathbf{S}^i_{\text{sum}}$, and thereby the Gram matrix shown in (28) for each leaf cluster, we add the fill-in's contributions as shown in (13). We then perform an SVD, from which the new cluster basis of each leaf cluster is obtained. Since the matrix on which SVD is performed is of leafsize and there are $O(N)$ such matrices, the total cost of this step is also linear.

However, during the factorization procedure, when eliminating a cluster i, among all of its admissible blocks shown in (24), some have already been changed due to the computation

performed on previous clusters. To see this point clearly, take cluster $i = 4$ shown in Fig. 5(a) as an example. When this cluster is being eliminated, clusters $i = 1, 2, 3$, colored in red, are already eliminated, while clusters $i = 5, 6, 7, 8$, colored in black, are not eliminated yet. The admissible blocks formed between cluster $i = 4$ and eliminated clusters, which are $\mathbf{Z}_{4,1}$ and $\mathbf{Z}_{4,2}$ shown in orange box, are different from those in the original matrix, because cluster bases of $i = 1, 2$ have been changed, and the $\mathbf{Z}_{4,1}$ and $\mathbf{Z}_{4,2}$ blocks have been multiplied from the right by $\overline{\mathbf{Q}}_1$ and $\overline{\mathbf{Q}}_2$ respectively.

If we number the clusters based on their elimination order in sequence, then, for $j_m > i$ and $jp_m > i_p$, the admissible blocks $\mathbf{Z}_{i,j_m}$ and $\mathbf{Z}^l_{ip,jp_m}$ are formed with uneliminated clusters. They have the same form as those shown in (25) and (26). However, the admissible blocks $\mathbf{Z}_{i,j_m}$ and $\mathbf{Z}^l_{ip,jp_m}$ for $j_m < i$ and $jp_m < i_p$, i.e., those formed with eliminated clusters are not the same as those in the original $\mathcal{H}^2$-matrix $\mathbf{Z}$ anymore. Instead, they have the following form due to $\mathbf{Q}$ matrix multiplication performed in Step 1:

$$\widetilde{\mathbf{Z}}_{i,j_m} = \mathbf{V}_i \mathbf{S}_{i,j_m} \mathbf{V}^T_{j_m} \overline{\overline{\mathbf{V}}}_{j_m}, \quad j_m < i \tag{32}$$

$$\widetilde{\mathbf{Z}}^l_{ip,jp_m} = \mathbf{V}_i \mathbf{T}_i^{L-1} \ldots \mathbf{T}_i^l \mathbf{S}^l_{ip,jp_m} \mathbf{V}^T_{jp_m} \overline{\overline{\mathbf{V}}}_{jp_m}, \quad jp_m < ip. \tag{33}$$

To highlight the fact that the aforementioned blocks are different from those in the original matrix, in (32) and (33), we use $\widetilde{\mathbf{Z}}$ instead of $\mathbf{Z}$ to represent these blocks. Similarly, we update (24) as follows to reflect the fact that some of the underlying blocks have been changed:

$$\widetilde{\mathbf{Z}}_i = \big[\widetilde{\mathbf{Z}}_{i,j_1}\ \widetilde{\mathbf{Z}}_{i,j_2}\ \ldots, \mathbf{Z}_{i,j_m}, \ldots, \widetilde{\mathbf{Z}}^{L-1}_{ip,jp_1}\ \widetilde{\mathbf{Z}}^{L-1}_{ip,jp_2} \ldots, \mathbf{Z}^{L-1}_{ip,jm}, \ldots, \widetilde{\mathbf{Z}}^{l_0}_{ip,jp_1}\ \mathbf{Z}^{l_0}_{ip,jp_2}\ \ldots\big]. \tag{34}$$

We then use (34) to compute the Gram matrix of $\mathbf{Z}_i$, which becomes

$$\mathbf{Z}_{i,2} = \widetilde{\mathbf{Z}}_i \widetilde{\mathbf{Z}}_i^H = \mathbf{V}_i \mathbf{S}^{i,\text{new}}_{\text{sum}} \mathbf{V}_i^H \tag{35}$$

where

$$\mathbf{S}^{i,\text{new}}_{\text{sum}} = \begin{cases} \widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}}, & l(i) = l_0 \\ \widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}} + \mathbf{T}_i \mathbf{S}^{ip,\text{new}}_{\text{sum}} \mathbf{T}_i{}^H, & l_0 < l(i) \le L. \end{cases} \tag{36}$$

Using nested property, the above can still be computed via a top-down tree traversal procedure similar to Fig. 4. However, $\widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}}$ now is different from (30). We can divide $\widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}}$ into two parts as

$$\widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}} = \widetilde{\mathbf{S}}^i_{\text{sum},1} + \widetilde{\mathbf{S}}^i_{\text{sum},2}. \tag{37}$$

The first part, denoted by $\widetilde{\mathbf{S}}^i_{\text{sum},1}$, is from the admissible blocks formed between cluster i and eliminated clusters at i's tree level, as shown in (32) and (33). The second part, denoted by $\widetilde{\mathbf{S}}^i_{\text{sum},2}$, is from the admissible blocks formed between cluster i and uneliminated clusters, whose expressions are given in (25) and (26). For the first part, since the admissible block has a form of (32), we compute it as

$$\widetilde{\mathbf{S}}^i_{\text{sum},1} = \sum_{j\,(j<i)} \mathbf{S}_{i,j} \mathbf{B}^{\text{new}}_j \big(\mathbf{B}^{\text{new}}_j\big)^H \mathbf{S}^H_{i,j} \tag{38}$$

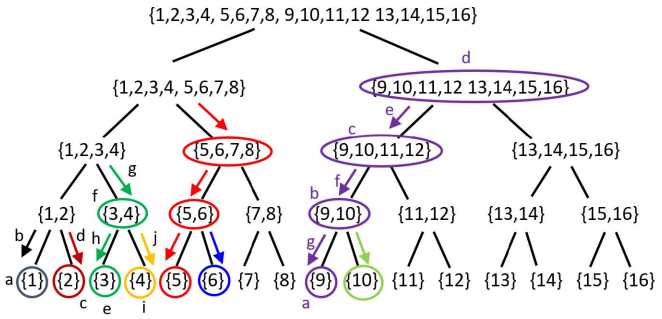


Fig. 6. Illustration of the procedure for instantaneously computing $\mathbf{S}^{i,\text{new}}_{\text{sum}}$.

where $\mathbf{B}^{\text{new}}_j$ is the projection of the new cluster basis onto the original cluster basis as shown in the following:

$$\mathbf{B}^{\text{new}}_j = \mathbf{V}^T_j \overline{\overline{\mathbf{V}}}_j. \tag{39}$$

For a nonleaf cluster j, using the nested property of cluster basis $\mathbf{V}_j$, $\mathbf{B}^{\text{new}}_j$ can be computed from $\mathbf{B}$ of its two children clusters, $\mathbf{B}^{\text{new}}_{j1}$ and $\mathbf{B}^{\text{new}}_{j2}$, as $\mathbf{B}^{\text{new}}_j = [\mathbf{T}^T_1 \mathbf{B}^{\text{new}}_{j1}\ \mathbf{T}^T_2 \mathbf{B}^{\text{new}}_{j2}]$. Here, $\mathbf{T}_1$ and $\mathbf{T}_2$ are the left and right transfer matrices of a nonleaf cluster j. $\mathbf{B}^{\text{new}}_j$ is nothing but the rightmost part of (32) and (33). Hence, $(\mathbf{B}^{\text{new}}_j)^T$ can be viewed as the new cluster basis for j. For the second part, since the admissible block has its original form shown in (25), we have

$$\widetilde{\mathbf{S}}^i_{\text{sum},2} = \sum_{j\,(j>i)} \mathbf{S}_{i,j} \mathbf{B}_j \mathbf{B}^H_j \mathbf{S}^H_{i,j} = \sum_{j\,(j>i)} \mathbf{S}_{i,j} \mathbf{S}^H_{i,j}. \tag{40}$$

Like $\widetilde{\mathbf{S}}^i_{\text{sum}}$, $\widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}}$ can be obtained for all clusters in linear time. However, unlike $\widetilde{\mathbf{S}}^i_{\text{sum}}$, the $\widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}}$ cannot be generated in advance before the factorization, since the updated admissible blocks formed by i are not known until we reach the step of eliminating cluster i. Hence, $\widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}}$ should be generated instantaneously during the factorization procedure, when it is time to eliminate cluster i. The same holds true for the top-down process of splitting the parent cluster's $\mathbf{S}^{i,\text{new}}_{\text{sum}}$ to its children clusters to obtain $\mathbf{S}^{i,\text{new}}_{\text{sum}}$ for each cluster i. In what follows, we use the cluster tree shown in Fig. 6 to illustrate the entire process of obtaining $\mathbf{S}^{i,\text{new}}_{\text{sum}}$ for each leaf cluster.

We number leaf clusters from left to right as $1, 2, 3, \ldots$, which is also the order of elimination. For the first cluster $i = 1$, we first compute its $\widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}}$ denoted by a circle around cluster $i = 1$ in Fig. 6. Since no clusters have been eliminated, $\widetilde{\mathbf{S}}^{i,\text{new}}_{\text{sum}}$ is the same as that in the original matrix, thus $\widetilde{\mathbf{S}}^i_{\text{sum}}$. We then check whether $\mathbf{S}^{ip,\text{new}}_{\text{sum}}$ is known or not. Since the immediate parent cluster of $i = 1$ has not been updated either in its admissible blocks, $\mathbf{S}^{ip,\text{new}}_{\text{sum}} = \mathbf{S}^{ip}_{\text{sum}}$. As a result, (36) can be readily computed for cluster $i = 1$. The arrow from $i = 1$'s parent cluster to $i = 1$ represents the $(+\mathbf{T}_i \mathbf{S}^{ip,\text{new}}_{\text{sum}} \mathbf{T}_i{}^H)$ operation in (36), which is to split $\mathbf{S}^{ip,\text{new}}_{\text{sum}}$ to its child, and then add it with the child's $\widetilde{\mathbf{S}}^{\text{new}}_{\text{sum}}$.

For cluster $i = 2$, since, when it is factorized, cluster $i = 1$ has been factorized, we compute its $\widetilde{\mathbf{S}}^i_{\text{sum},1}$ using (38), where $\mathbf{B}^{\text{new}}_1$ is employed. Meanwhile, we compute its $\widetilde{\mathbf{S}}^i_{\text{sum},2}$ using (40), which captures the admissible blocks formed between $i = 2$ and uneliminated clusters. The sum of the

two makes $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$, which can be different from that in the original matrix, i.e., $\widetilde{\mathbf{S}}_{\text{sum}}^{i}$. Note that considering all eliminated and uneliminated clusters, there are at most $O(C_{sp})$ clusters that can form admissible blocks with a cluster. The step of computing $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$ for $i = 2$ is denoted by a circle surrounding cluster $i = 2$ in Fig. 6. Since i's immediate parent cluster ip has a known $\mathbf{S}_{\text{sum}}^{ip,\text{new}}$, which is equal to $\mathbf{S}_{\text{sum}}^{ip}$, summing up $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$ and the second term in (36) (denoted by the red arrow pointing from $i = 2$'s parent to $i = 2$), we obtain $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ for cluster $i = 2$.

We then proceed to cluster $i = 3$. When this cluster is factorized, both clusters $i = 1$ and $i = 2$ have been factorized, and we hence compute its $\widetilde{\mathbf{S}}_{\text{sum},1}^{i}$ using (38) and $\widetilde{\mathbf{S}}_{\text{sum},2}^{i}$ using (40), the sum of which makes $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$. As for the parent cluster's contribution to $i = 3$, since now the parent cluster of clusters 1 and 2, cluster $i = \{1, 2\}$, has an updated $\mathbf{B}_i^{\text{new}}$, $\widetilde{\mathbf{S}}_{\text{sum}}^{\{3,4\},\text{new}}$ would have to be computed, as it may be different from the original one. Thus, we compute $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$ for $i = 3$'s parent cluster $\{3, 4\}$, the step of which is denoted by the circle around $\{3, 4\}$, and then $\mathbf{S}_{\text{sum}}^{\{3,4\},\text{new}}$ can be found from (36) by realizing that $\mathbf{S}_{\text{sum}}^{ip,\text{new}} = \mathbf{S}_{\text{sum}}^{\{1,2,3,4\}}$, which is cluster $\{1, 2, 3, 4\}$'s $\mathbf{S}_{\text{sum}}$ in the original matrix. After $\mathbf{S}_{\text{sum}}^{\{3,4\},\text{new}}$ is found, $\mathbf{S}_{\text{sum}}^{i=3,\text{new}}$ can be readily computed by splitting $\mathbf{S}_{\text{sum}}^{\{3,4\},\text{new}}$'s contribution to it, denoted by the arrow labeled as h.

For cluster $i = 4$, we first compute its $\widetilde{\mathbf{S}}_{\text{sum}}^{4,\text{new}}$. We then check whether its parent cluster's $\mathbf{S}_{\text{sum}}^{\text{new}}$ is known or not, which turns out to be known since $\mathbf{S}_{\text{sum}}^{\{3,4\},\text{new}}$ has been found when cluster $i = 3$ is computed. Thus, a single splitting of $\mathbf{S}_{\text{sum}}^{\{3,4\},\text{new}}$ to $i = 4$ and adding it with $\widetilde{\mathbf{S}}_{\text{sum}}^{4,\text{new}}$ yields $\mathbf{S}_{\text{sum}}^{\{4\},\text{new}}$. This step is represented by the yellow arrow in Fig. 6.

Algorithm 1 Proposed Algorithm for Computing $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ for All Leaf Clusters

for $cluster : i$ =1 **to** 2^L **do**
  Compute $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$ (using (37)).
  $l = l(i)$: Let l to be the tree level of cluster i
  $j = i$
  while $\mathbf{S}_{\text{sum}}^{\text{parent}(j),\text{new}}$ is not known **do**
    $j = \text{parent}(j)$: Let next j be j's parent
    Compute $\widetilde{\mathbf{S}}_{\text{sum}}^{j,\text{new}}$.
  end while
  for $l = l(j)$ **to** $l(i) - 1$ **do**
    $k = \text{parent}(i, l)$: Let k be i's parent at level l
    Compute $\mathbf{S}_{\text{sum}}^{k,\text{new}} = \widetilde{\mathbf{S}}_{\text{sum}}^{k,\text{new}} + \mathbf{T}_k \mathbf{S}_{\text{sum}}^{\text{parent}(k),\text{new}} \mathbf{T}_k{}^H$.
  end for
  Compute $\mathbf{S}_{\text{sum}}^{i,\text{new}} = \widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}} + \mathbf{T}_i \mathbf{S}_{\text{sum}}^{\text{parent}(i),\text{new}} \mathbf{T}_i{}^H$.
end for

The algorithm shown in Algorithm 1 summarizes the overall procedure of instantaneously computing $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ for all leaf clusters. Basically, for an arbitrary leaf cluster i, we first compute its $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$. We then check whether $\mathbf{S}_{\text{sum}}^{ip,\text{new}}$ is known or not. If known, then using (36), $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ is obtained for leaf cluster i. If not, that means ip's admissible blocks have been changed, and thus we compute ip's $\widetilde{\mathbf{S}}_{\text{sum}}^{\text{new}}$. After that, we move one level up and check whether $\mathbf{S}_{\text{sum}}^{\text{new}}$ is known for ip's parent cluster. We continue to do so until we reach the level where i's parent cluster has a known $\mathbf{S}_{\text{sum}}^{\text{new}}$. This cluster's $\mathbf{S}_{\text{sum}}^{\text{new}}$ is known either because it is the first cluster at that level, and thus $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ is equal to the original $\mathbf{S}_{\text{sum}}^{i}$, or because it is at the minimal level where admissible block exists, and thus $\mathbf{S}_{\text{sum}}^{i,\text{new}} = \widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$, or because it has been obtained during previous cluster computation. For example, for cluster $i = 4$ shown in Fig. 6, its $\mathbf{S}_{\text{sum}}^{ip,\text{new}}$ is known from the computation done for cluster $i = 3$. As another example, for cluster $i = 9$, we have to reach its parent cluster at level $L-4$, where $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ is known. After that, we perform a top-down tree traversal to split $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ to children clusters and compute $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ of children clusters level by level down until we reach leaf level. The sequence of computation is marked in purple for cluster $i = 9$ using letters from a to g.

In Fig. 6, a circle around a cluster denotes the computation of its $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$, and an arrow denotes a *split and add* operation from its parent $\mathbf{S}_{\text{sum}}^{i,\text{new}}$. The sequence of computation is also marked in letters from a to i for the computation of $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ for clusters from $i = 1$ to $i = 4$. Obviously, it is different from that shown in Fig. 4, which is performed in advance before the matrix factorization. The procedure shown in Fig. 6 is performed concurrently with the factorization. Thus, $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$ is computed based on the updated admissible blocks at the time when cluster i is factorized. In Fig. 6, we also use different colors to denote the computation associated with the generation of $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ for a leaf cluster i. For example, the circles and arrows in red are the computations performed for cluster $i = 5$; and those in purple are the computations performed for cluster $i = 9$. Since each arrow in Fig. 6 is associated with a cost of $O(k^3)$, each circle is associated with a cost of $O(k^3)C_{sp}$, and there are in total $O(N)$ clusters in the tree, and the overall cost is $O(N)$.

Before we move to next level, we update the $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$ for each cluster which is now $\widetilde{\mathbf{S}}_{\text{sum},1}^{i,\text{new}}$ only as all clusters at the current level have been eliminated. After that, we obtain $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ for each cluster, which can now be performed based on the procedure shown in Fig. 4 since every block is known. This $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ will be used as the initial $\mathbf{S}_{\text{sum}}^{i}$ to start the process of new cluster basis computation at the next nonleaf level.

B. Concurrent Change of Cluster Bases at a Nonleaf Level

At a nonleaf level l, the computation of changing cluster bases is similar to that at the leaf level, because we can view the current nonleaf level l as the new leaf level and $\mathbf{T}_t^{\text{new}}$ shown in (15) and (19) as new leaf level cluster bases. Meanwhile, the coupling matrices at this level and up and the transfer matrices at upper levels all stay the same as those in the original matrix.

Similar to the leaf-level computation, for every cluster at level l (viewed as current leaf level), we assemble all of the i-related blocks into $\mathbf{Z}_i$ like (24), which includes the i-related admissible blocks at not only i's tree level but also i's parent levels. Similar to leaf level's situation, some of the admissible blocks are formed between cluster i and eliminated clusters, while the others are from uneliminated clusters. Note that now the row dimension of these blocks is only $2k_{l+1}$ as can be

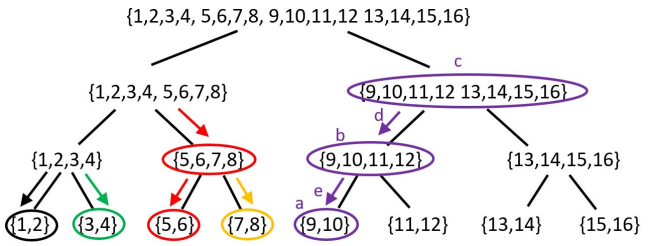


Fig. 7. Illustration of instantaneously computing $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ during a nonleaf level factorization.

seen from (15). The Gram matrix for nonleaf $\mathbf{Z}_i$ can then be computed as

$$\mathbf{Z}_{i,2} = \mathbf{Z}_i \mathbf{Z}_i^H = \mathbf{T}_i^{\text{new}} \mathbf{S}_{\text{sum}}^{i,\text{new}} \mathbf{T}_i^{\text{new},H} \tag{41}$$

where $\mathbf{S}_{\text{sum}}^{i,\text{new}}$ is obtained using a process similar to that at the leaf level. The procedure is shown in Fig. 7. Again, we instantaneously compute $\widetilde{\mathbf{S}}_{\text{sum}}^{i,\text{new}}$ and $\mathbf{S}_{\text{sum}}^{i,\text{new}}$. Now, we have

$$\widetilde{\mathbf{S}}_{\text{sum},1}^{i} = \sum_{j\,(j<i)} \mathbf{S}_{i,j} \mathbf{B}_j^{\text{new}} \left(\mathbf{B}_j^{\text{new}}\right)^H \mathbf{S}_{i,j}^H \tag{42}$$

where $\mathbf{B}_j^{\text{new}}$ is

$$\mathbf{B}_j^{\text{new}} = \left(\mathbf{T}_j^{\text{new}}\right)^T \overline{\mathbf{T}}_j^{\text{new}} \tag{43}$$

and

$$\widetilde{\mathbf{S}}_{\text{sum},2}^{i} = \sum_{j\,(j>i)} \mathbf{S}_{i,j} \mathbf{B}_j \mathbf{B}_j^H \mathbf{S}_{i,j}^H \tag{44}$$

with

$$\mathbf{B}_j = \left(\mathbf{T}_j^{\text{new}}\right)^T \overline{\mathbf{T}}_j^{\text{new}}. \tag{45}$$

Here, different from (40), at a nonleaf level, $\mathbf{B}_j \mathbf{B}_j^H$ may not be identity any more.

We then add fill-in's contributions to update the cluster basis of cluster i by computing

$$\widetilde{\mathbf{Z}}_{i,2} = \mathbf{Z}_{i,2} + \sum\nolimits_m \mathbf{F}_{i,j_m} \mathbf{F}_{i,j_m}^H \tag{46}$$

where $\mathbf{F}_{i,j_m}$ are fill-in blocks associated with the admissible blocks of cluster i. $\widetilde{\mathbf{Z}}_{i,2}$ is a small matrix whose size is rank k. Its SVD can be readily performed. The resultant singular vector matrix truncated based on prescribed accuracy ϵ_{acc} is the new transfer matrix $\widetilde{\mathbf{T}}_i^{\text{new}}$, and thus

$$\widetilde{\mathbf{Z}}_{i,2,O(k_l)\times O(k_l)} \overset{\epsilon_{\text{acc}}}{\approx} \widetilde{\mathbf{T}}_i^{\text{new}} \Sigma' \widetilde{\mathbf{T}}_i^{\text{new},H}. \tag{47}$$

The overall computational cost is again linearly proportional to the number of clusters in the $\mathcal{H}^2$-tree being handled at level l. As a result, the overall computational cost of changing cluster basis is $O(2^{L-1})$ at tree level $L-1$, $O(2^{L-2})$ at tree level $L-2$, and so on. Hence, the total computational cost is $O(N)$.

V. Accuracy and Complexity Analysis

The accuracy of the proposed direct solution is controlled by ϵ_{acc}. This is different from [18] and [19], since it is directly controlled. As can be seen from the previous sections, there are no approximations involved in the proposed solution, except that when we change the cluster bases to account for the contribution from the fill-ins, we use (14) to truncate the singular vectors at the leaf level and (47) to truncate the singular vectors at a nonleaf level. The accuracy of this step is controlled by parameter ϵ_{acc}, which can be set to any desired value.

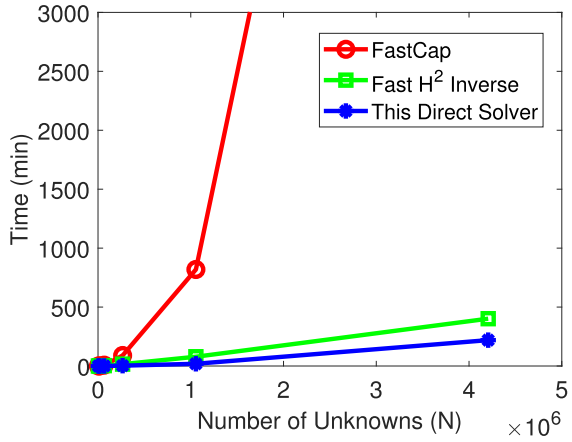
From (23), it is evident that the whole computation involves $\mathbf{Q}_i^l$, $\mathbf{L}_i^l$, and $\mathbf{U}_i^l$ for cluster i at level l, the storage and computation of each of which are, respectively, $O(k_l^2)$ and $O(k_l^3)$ at a nonleaf level and constant at the leaf level. Since there are $O(N)$ clusters in a tree, the total cost of generating these factors is linear for constant-rank $\mathcal{H}^2$ matrices. 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. The change of the cluster basis costs $O(N)$ in time and memory also as analyzed in Section IV. As a result, the time and memory complexity of the proposed direct solution are linear for constant-rank $\mathcal{H}^2$-matrices. For variable-rank $\mathcal{H}^2$, the complexity of the proposed direct solution can also be analytically derived based on rank's behavior.

VI. Simulation Results

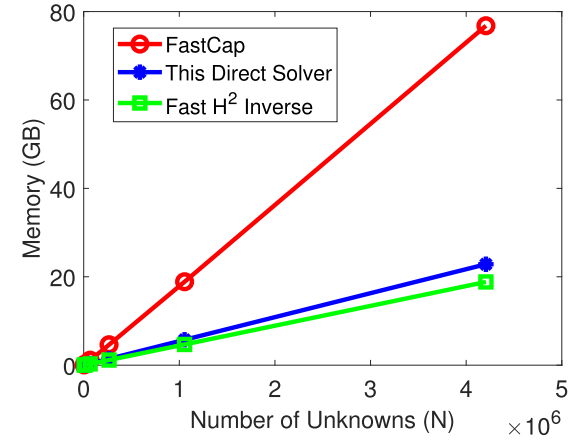
The accuracy and complexity of the proposed direct solver are examined by performing both capacitance extraction and full-wave electromagnetic analysis of large-scale structures. The computer used has an Intel(R) Xeon(R) CPU running at 3.00 GHz, and only a *single core* is employed for carrying out the computation to demonstrate the low computational complexity of this direct solver.

A. Two-Layer Cross Bus

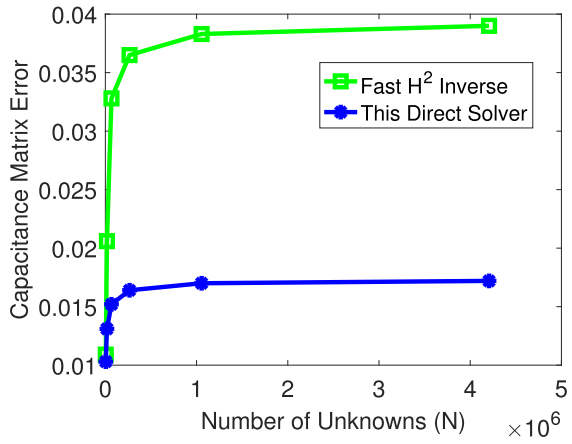
The first example is a two-layer cross bus structure in a uniform dielectric, as shown in [19]. In each layer, there are m conductors, and each conductor has a dimension of $1\times1\times(2m+1)$ m^3. We simulate a suite of such structures with $m = 16, 32, 64, 128$, and 512, respectively, whose number of unknowns N is, respectively, 17 152, 67 072, 265 216, 1 055 232, and 4 206 592. The parameters used in the proposed direct solver are $leafsize = 20$, $\eta = 1$, and $\epsilon_{\mathcal{H}^2} = 10^{-4}$ for $\mathcal{H}^2$-matrix construction and $\epsilon_{\text{acc}} = 10^{-8}$ for direct solution. The resultant rank in the $\mathcal{H}^2$-matrix as well as the proposed direct solution is found to be a constant no greater than 5 at each tree level. FastCap [2] and the $O(N)$ inverse solver in [19] are also used to simulate the same example. The expansion order used in FastCap is 3. In Fig. 8(a), we compare the total CPU run time of the three solvers as a function of N. Clear linear scaling can be observed from the proposed direct solver and that of [19]. Moreover, the proposed direct solver is shown to take less CPU time. We also compare the memory usage of the three solvers in Fig. 8(b). The fast $\mathcal{H}^2$-inverse solver keeps original $\mathcal{H}^2$-matrix memory, which needs least memory. The proposed solver costs only slightly more memory than the fast $\mathcal{H}^2$-inverse method due to cluster basis change during factorization, while both of them are much more memory efficient than FastCap. We have also used the capacitance matrix extracted from FastCap with expansion order 3 as the reference to compare the accuracy of the proposed direct solver with that of the fast $\mathcal{H}^2$-inverse. The



(a)



(b)



(c)

Fig. 8. Capacitance extraction of a two-layer cross bus interconnect. (a) Time complexity. (b) Memory complexity. (c) Capacitance matrix error.

error is assessed by $||\mathbf{C} - \mathbf{C}_{\text{ref}}||/||\mathbf{C}_{\text{ref}}||$, where $\mathbf{C}_{\text{ref}}$ is from FastCap and $\mathbf{C}$ is from either this direct solver or the fast $\mathcal{H}^2$-inverse. As can be seen from Fig. 8(c), the proposed direct solver exhibits better accuracy while using less CPU time.

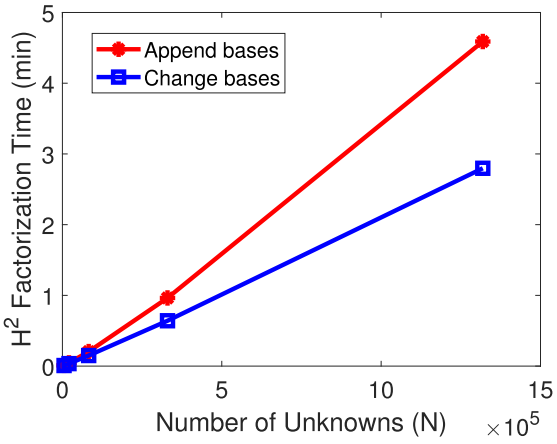
B. On-Chip Interconnects

We have also performed a full-wave VIE simulation of on-chip interconnects. A suite of large-scale on-chip bus structures from 4×4 to 64×64 are simulated at 30 GHz with an x-polarized incident electric field. The conductivity of the metal is 5.8×10^7 S/m. The dimensions of each bus are $1\ \mu\text{m} \times 1\ \mu\text{m} \times 20\ \mu\text{m}$. The horizontal distance between the centers of two neighboring buses is $20\ \mu$m. And the vertical distance is $40\ \mu$m. 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 $\epsilon_{\text{acc}} = 10^{-4}$. In Fig. 9(b), we compare the memory cost versus N of [22] with this new solver. In Fig. 9(a), we compare factorization time as a function of N. Clear linear complexities can be observed in CPU time and memory consumption for both solvers. Note that the figures are plotted in a linear scale.

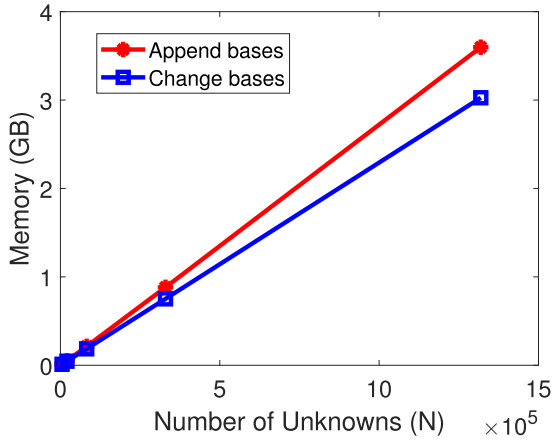
TABLE I

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
This solver	1.008e-5	1.005e-5	1.004e-5	1.003e-5	1.002e-5

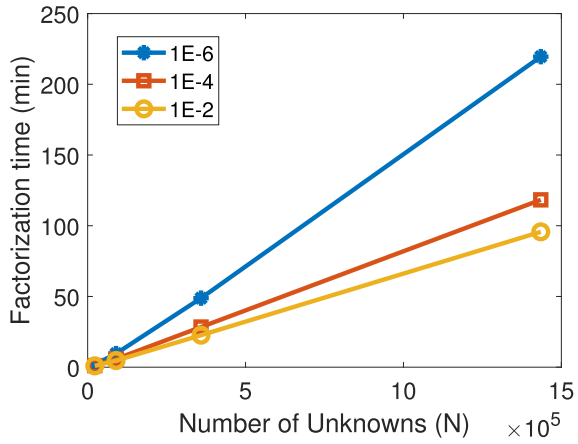


(a)

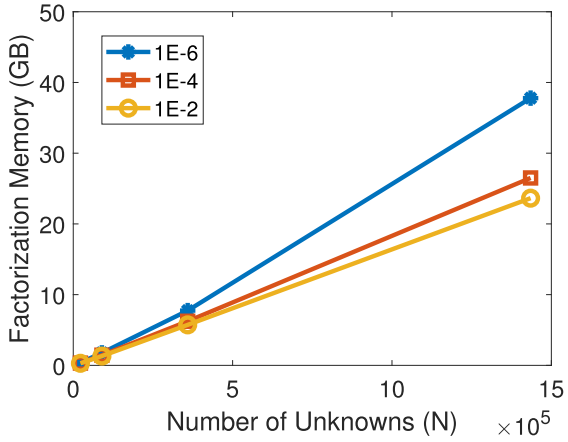


(b)

Fig. 9. Solver performance of lossy on-chip buses. (a) Factorization time versus N. (b) Memory versus N.



(a)



(b)

Fig. 10. Solver performance of a suite of dielectric slab for different choices of ϵ_{acc}. (a) Factorization time versus N. (b) Memory versus N.

However, in [22], the cluster bases are appended instead of completely changed during matrix factorization. The same accuracy parameter 10^{-4} is used to append the original cluster bases to take into account the fill-ins contribution during the factorization. Specifically, after deducting the fill-in block's component in the original space of cluster bases from the fill-in block, we truncate the remainder using 10^{-4} tolerance to determine additional cluster bases to append. As can be seen from Fig. 9(a) and (b), this new direct solver with a concurrent change of the cluster basis is computationally more efficient. The accuracy of the proposed direct solver is assessed by relative residual and listed in Table I. Excellent accuracy can be observed in the entire unknown range.

C. Large-Scale Dielectric Slab

We then simulate a dielectric slab with $\epsilon_r = 2.54$ at 300 MHz, which is also simulated in [28]. The thickness of the slab is fixed to be $0.1\lambda_0$. The width and the length are simultaneously increased from $4\lambda_0$, $8\lambda_0$, and $16\lambda_0$, 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.

TABLE II

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

ϵ_{acc}	10^{-2}	10^{-4}	10^{-6}
ϵ_{rel} $(N=22,560)$	0.0218	0.0014	0.0002
ϵ_{rel} $(N=89,920)$	0.0216	0.0019	0.0018
ϵ_{rel} $(N=359,040)$	0.0241	0.0053	0.0072
ϵ_{rel} $(N=1,434,880)$	0.1325	0.0063	0.0027

TABLE III

PERFORMANCE COMPARISON BETWEEN THIS SOLVER WITH $\epsilon_{\text{ACC}} = 10^{-4}$ AND [29] AND FOR THE DIELECTRIC SLAB EXAMPLE

N	89,920	359,040
Factorization (s) [This]	349	1703
Solution (s) [This]	1.55	6.85
Inversion (s) [29]	2.75e+03	1.65e+04
Relative Residual [This]	0.19%	0.53%
Inverse Error [29]	3.9%	7.49%

The *leafsize* is chosen to be 25 and $\eta = 1$. ϵ_{acc} is set to be 10^{-2}, 10^{-4}, and 10^{-6}, respectively, to examine the solution accuracy, computational complexity, and error controllability of the proposed direct solution.

In Fig. 10(a), we plot the factorization time with respect to N for all three different choices of ϵ_{acc}. It is clear that the smaller the ϵ_{acc}, the larger the factorization time. However, the complexity remains the same as linear regardless of the choice of ϵ_{acc}. The memory cost is plotted in Fig. 10(b). 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 in the equation to be solved. The relative residual ϵ_{rel} of the proposed direct solution is listed in Table II as a function of ϵ_{acc}. Excellent accuracy can be observed in the entire unknown range. Furthermore, the accuracy can be controlled by ϵ_{acc}, and in general, smaller ϵ_{acc} results in better accuracy. Since this example is also simulated in our previous work [28], [29], we have compared the two direct solvers in CPU run time and accuracy. $\epsilon_{\text{acc}} = 10^{-4}$ is used. For a fair comparison, we set up this solver to solve the same $\mathcal{H}^2$-matrix solved in [29] 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 [29], which is a direct $\mathcal{H}^2$-matrix inversion. The speedup is about one order of magnitude in large examples. In addition, because of a direct error control, the error of the proposed solution is also much less than that of [29].

D. Large-Scale Array of Dielectric Cubes

We also 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- directions simultaneously from 2 to 14, thus producing a 3-D cube array from $2 \times 2 \times 2$ to $14 \times 14 \times 14$ elements. The number of unknowns N is, respectively, 3024, 24 192, 193 536, and 1 037 232 for these arrays. During the construction of

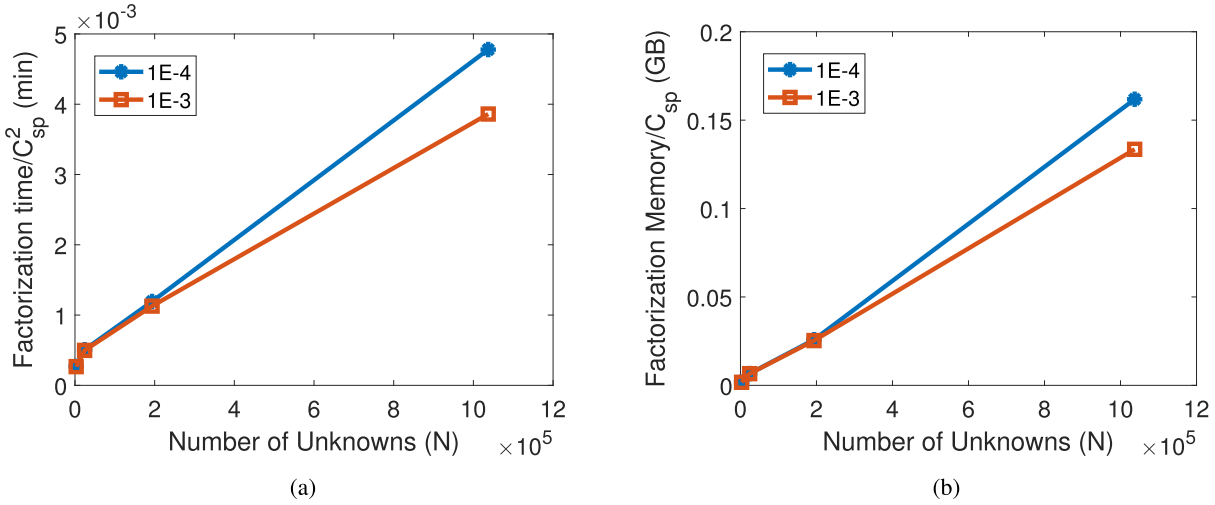


Fig. 11. Solver performance of a suite of cube array from $2 \times 2 \times 2$ to $14 \times 14 \times 14$. (a) Time scaling versus N. (b) Memory scaling versus N.

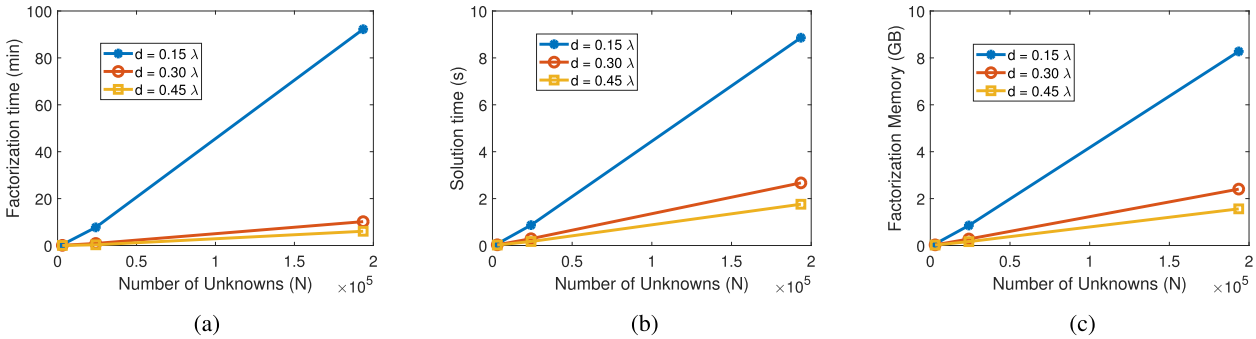


Fig. 12. Time and memory scaling with N for three different distances for the cube array example. (a) Factorization time. (b) Solution time. (c) Memory.

TABLE IV

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

N	3024	24192	193536	1037232
$\epsilon_{acc} = 10^{-3}$	0.0046	0.0113	0.0177	0.0645
$\epsilon_{acc} = 10^{-4}$	0.0012	0.0022	0.0036	0.0306

TABLE V

MATRIX ERROR FOR THE CUBE ARRAY EXAMPLE AS A FUNCTION OF CUBE DISTANCE

Distance; N	3024	24192	193536
0.15 λ	1.687400e-03	5.871952e-03	1.279663e-02
0.30 λ	2.135456e-03	5.335353e-03	1.042830e-02
0.45 λ	2.219728e-03	4.999455e-03	8.788409e-03

$\mathcal{H}^2$-matrix, we set leafsize $= 25$, $\eta = 1$, and $\epsilon_{\mathcal{H}^2} = 10^{-4}$. The ϵ_{acc} is chosen as 10^{-3} and 10^{-4}, respectively. The time and memory performance is shown in Fig. 11(a) and (b) and the errors are shown in Table IV. 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 ϵ_{acc}.

The distance between two adjacent cubes in this example is $0.3\lambda_0$. To examine the effect of the distance on the complexity of the proposed solver, we simulated the example with the other two distances: $0.15\lambda_0$ and $0.45\lambda_0$, respectively. Since for a given admissibility condition (1), although the number of admissible blocks becomes less if adjacent objects are closer, thus the absolute run time and memory usage will increase, the complexity of the solver stays the same since the parameters related to the matrix partition (into inadmissible and admissible blocks) are constant independent of N. These parameters, such as the number of admissible blocks that can be formed by one cluster, can become larger or smaller; however, they are the same constant for different N values. Hence, they do not change the rate at which the CPU run time and memory scale with N. This can be seen clearly from Fig. 12, where we plot the factorization time, the solution time, and the memory as a function of N for three different distances between adjacent cubes. We can observe that when the distance is closer, the total run time and memory usage increase. However, the scaling rate, i.e., linear or quadratic or others, with N stays the same.

The accuracy is also examined for this example as a function of cube distance. As can be seen from Tables V and VI, both the $\mathcal{H}^2$-representation and the direct solution are

TABLE VI
DIRECT SOLUTION ERROR MEASURED BY RELATIVE RESIDUAL FOR THE CUBE ARRAY EXAMPLE AS A FUNCTION OF CUBE DISTANCE

$Distance; N$	3024	24192	193536
0.15 λ	0.0047	0.0115	0.0242
0.30 λ	0.0046	0.0113	0.0177
0.45 λ	0.0056	0.0089	0.0160

accurate. Furthermore, for the same structure, regardless of the distance between adjacent cubes (this corresponds to one vertical column in Tables V and VI), the accuracy is kept at the same order. This is also understood, since the accuracy of the proposed solver is controlled, and also solely controlled by $\epsilon_{\mathcal{H}^2}$ for the construction and ϵ_{acc} for the direct solution independent of the distance between cubes and other structure specifics. The results in Tables V and VI are generated based on $\epsilon_{\mathcal{H}^2} = 10^{-4}$ and $\epsilon_{\text{acc}} = 10^{-3}$. The accuracy can also be changed to another desired value by setting a different error tolerance.

VII. CONCLUSION

In this paper, we develop a new direct solution for general $\mathcal{H}^2$-matrices. This new direct solution features not only a directly controlled accuracy but also a complete change of cluster basis that is done concurrently with the factorization to efficiently represent the update to the original $\mathcal{H}^2$-matrix. The complexity of the overall direct solution, however, is still kept the same as before without being increased, through the efficient algorithms developed in this paper. For example, its complexity is linear for factorizing constant-rank $\mathcal{H}^2$-matrices, and O(NlogN) for factorizing electrically large VIEs whose rank increases with electrical size. Millions of unknowns are directly solved on a single-core CPU in fast CPU run time. Comparisons with the state-of-the-art $\mathcal{H}^2$-based direct solvers have demonstrated the accuracy of this new direct solution as well as its significantly improved computational efficiency. The essential idea of this paper can also be applied to develop an $\mathcal{H}^2$-matrix–matrix multiplication algorithm of controlled accuracy. In addition, it can also be applied to solve other IEs and partial differential equations.

REFERENCES

[1] W. C. Chew, *Fast and Efficient Algorithms in Computational Electromagnetics*. Norwood, MA, USA: Artech House, 2001.

[2] K. Nabors and J. White, "FastCap: A multipole accelerated 3-D capacitance extraction program," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 11, pp. 1447–1459, Nov. 1991.

[3] Z. Zhu, B. Song, and J. K. White, "Algorithms in FastImp: A fast and wide-band impedance extraction program for complicated 3-D geometries," in *Proc. 40th Design Automat. Conf.*, Jun. 2003, pp. 712–717.

[4] W. Shi, J. Liu, N. Kakani, and T. Yu, "A fast hierarchical algorithm for 3-D capacitance extraction," in *Proc. 35th DAC Design Automat. Conf.*, Jun. 1998, pp. 212–217.

[5] R. J. Adams, Y. Xu, X. Xu, J.-S. 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.

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

[8] A. Heldring, J. M. Rius, J. M. Tamayo, J. Parrón, 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.

[9] 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.

[10] 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.

[11] 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

[12] A. B. Manić, A. P. Smull, F.-H. Rouet, X. S. Li, and B. M. Notaroš, "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, May 2017.

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

[14] W. Hackbusch, "A sparse H-matrix arithmetics. Part II: Application to multi-dimensional problems," *Computing*, vol. 64, no. 1, pp. 21–47, 2000.

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

[16] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Fast algorithms for hierarchically semiseparable matrices," *Numer. Linear Algebra Appl.*, vol. 17, pp. 953–976, Dec. 2010.

[17] S. Börm, *Efficient Numerical Methods for Non-Local Operators: H^2-Matrix Compression, Algorithms and Analysis*. Zurich, Switzerland: European Math. Soc., 2010.

[18] W. Chai, D. Jiao, and C.-K. 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.

[19] 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.

[20] 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.

[21] 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.

[22] M. Ma and D. Jiao, "Accuracy directly controlled fast direct solution of general H^2-matrices and its application to solving electrodynamic volume integral equations," *IEEE Trans. Microw. Theory Techn.*, vol. 66, no. 1, pp. 35–48, Jan. 2018.

[23] M. Ma and D. Jiao, "Accuracy controlled direct integral equation solver of linear complexity with change of basis for large-scale interconnect extraction," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Jun. 2018, pp. 197–200.

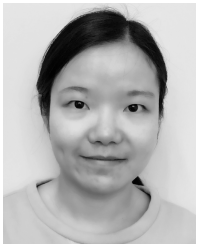
[24] M. Ma and D. Jiao, "Linear-complexity direct integral equation solver with explicit accuracy control for large-scale interconnect extraction," in *Proc. Int. Appl. Comput. Electromagn. Soc. Symp. (ACES)*, Mar. 2018, pp. 1–2.

[25] M. Ma and D. Jiao, "Direct solution of general $\mathcal{H}^2$-matrix with controlled accuracy and change of cluster bases for large-scale electromagnetic analysis," in *Proc. IEEE MTT-S Int. Conf. Numer. Electromagn. Multiphys. Modeling Optim. (NEMO)*, Aug. 2018, pp. 1–4.

[26] 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 Propaga.*, vol. 32, no. 1, pp. 77–85, Jan. 1984.

[27] 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.

[28] D. Jiao and S. Omar, "Minimal-rank $\mathcal{H}^2$-matrix-based iterative and direct volume integral equation solvers for large-scale scattering analysis," in *Proc. IEEE Int. Symp. Antennas Propag.*, Jul. 2015, pp. 740–741.
[29] S. Omar, M. Ma, and D. Jiao, "Low-complexity direct and iterative volume integral equation solvers with a minimal-rank H^2-representation for large-scale three-dimensional electrodynamic analysis," *IEEE J. Multiscale Multiphys. Comput. Tech.*, vol. 2, pp. 210–223, 2017.



Miaomiao Ma (GS'16–M'18) 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 a member of the On-Chip Electromagnetics Group, School of Electrical and Computer Engineering, Purdue University. Her current research interests include computational electromagnetics, fast and high-capacity numerical methods, and scattering analysis.

Ms. Ma was a recipient of the Cadence's Women in Technology Scholarship in 2018. She was also the recipient of the Honorable Mention Award from the IEEE International Symposium on Antennas and Propagation in 2016 and 2018, respectively, and the Best Student Paper Award from the IEEE International Conference on Wireless Information Technology and Systems (ICWITS) and Applied Computational Electromagnetics (ACES) 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, until 2005, as a Senior CAD Engineer, a Staff Engineer, and a Senior Staff Engineer. In 2005, she joined the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, as an Assistant Professor, where she is currently a Professor. She has authored over 300 papers in refereed journals and international conferences. Her current research interests include computational electromagnetics, high-frequency digital, analog, mixed-signal, and RF integrated circuit (IC) 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 was a recipient of the 2010 Ruth and Joel Spira Outstanding Teaching Award, the 2008 National Science Foundation (NSF) CAREER Award, the 2006 Jack and Cathie Kozik Faculty Startup Award (which recognizes an outstanding new faculty member of the School of Electrical and Computer Engineering, Purdue University), the 2006 Office of Naval Research (ONR) Award under the Young Investigator Program, the 2004 Best Paper Award presented at the Intel Corporation's annual corporate-wide technology conference (the 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, 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 is the Chair of the Paper Awards Committee of the IEEE Antennas and Propagation (AP) Society in 2019. She has been the Chair of Women in Engineering of the IEEE AP Society since 2018. She is the General Chair of the 2019 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO). She was among the 21 women faculty selected across the country as the 2014–2015 Fellow of the Executive Leadership in Academic Technology and Engineering (ELATE) at Drexel, a national leadership program for women in the academic STEM fields. 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 has served as a Reviewer for many IEEE journals and conferences. She is an Associate Editor of the IEEE JOURNAL ON MULTISCALE AND MULTIPHYSICS COMPUTATIONAL TECHNIQUES.