Layered $\mathcal{H}$-Matrix Based Inverse and LU Algorithms for Fast Direct Finite-Element-Based Computation of Electromagnetic Problems

Haixin Liu and Dan Jiao, *Senior Member, IEEE*

***Abstract*—A layered $\mathcal{H}$-matrix based inverse algorithm and a layered $\mathcal{H}$-matrix based LU factorization and solution algorithm are developed to accelerate the direct solution of the finite element matrix resulting from electromagnetics-based analysis of general 3-D problems. In these algorithms, the direct solution of the original 3-D system matrix is transformed to the direct solution of multiple 2-D problems. The size of the matrix to be computed is thus reduced from a 3-D size to a 2-D size. Moreover, the growth rate of the rank of the inverse finite element matrix with electric size is reduced from a 3-D based growth rate to a 2-D based growth rate. Numerical experiments have demonstrated the clear advantages of the proposed direct solvers over a state-of-the-art multifrontal based direct sparse solver as well as existing fast $\mathcal{H}$-matrix based direct solvers, in both CPU time and memory consumption, for finite-element analysis of general 3-D electromagnetic problems.**

***Index Terms*—$\mathcal{H}$-matrix, direct matrix solution, finite element methods, electromagnetic analysis.**

I. Introduction

A FINITE-ELEMENT METHOD (FEM) based analysis of a complex electromagnetic problem generally results in a large-scale system matrix. Although the matrix is sparse, its direct solution can be a computational challenge when the matrix size is large. The optimal operation count of a direct solution to an FEM matrix of size N is shown to be $O(N^{1.5})$ for 2-D problems, which is achieved by nested dissection [1]. For 3-D problems, the time complexity of a nested-dissection based sparse matrix solver is $O(N^2)$ [1]. A local-global solution modes based fast direct solution is developed in [2] for the FEM based analysis of 2-D wave problems. State-of-the-art finite-element-based solvers rely on iterative approaches to solve a large-scale 3-D problem. The resulting computational complexity is $O(N_{\rm it}N_{\rm rhs}N)$, where $N_{\rm it}$ is the number of iterations and $N_{\rm rhs}$ the number of right hand sides. When $N_{\rm it}$ or $N_{\rm rhs}$ is large, an iterative solver becomes inefficient.

Manuscript received May 07, 2012; revised August 29, 2012; accepted November 15, 2012. Date of publication December 03, 2012; date of current version February 27, 2013. This work was supported by the NSF under awards 0747578 and 0702567.

H. Liu was with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA. He is now with Oracle Corporation, Santa Clara, CA 94065 USA (e-mail: djiao@purdue.edu).

D. Jiao is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA (e-mail: djiao@purdue.edu).

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

Digital Object Identifier 10.1109/TAP.2012.2230236

In [3]–[5], an $\mathcal{H}$-matrix-based mathematical framework [6]–[9] was introduced to accelerate the direct solution of the FEM-based linear system of equations for large-scale electromagnetic analysis. It is proved in [5] that the sparse matrix resulting from a finite-element-based analysis of electrodynamic problems can be represented by an $\mathcal{H}$ matrix without any approximation, and the inverse of this sparse matrix has a data-sparse $\mathcal{H}$-matrix approximation with error well controlled. Based on this proof, an $\mathcal{H}$-matrix-based direct finite-element solver having $O(kN\log N)$ storage cost and $O(k^2N\log^2 N)$ operation counts is developed for solving electrodynamic problems, where rank k is adaptively determined based on a prescribed accuracy for each low-rank block in the inverse or LU factors of the finite-element matrix. To satisfy a prescribed accuracy, the rank k required in a static analysis is shown to be a constant regardless of matrix size. For electrodynamic analysis, it is shown in [10], [11] that for a prescribed error bound, the rank of the inverse finite-element matrix is a constant irrespective of electric size for analyzing 1-D electrodynamic problems; for 2-D electrodynamic problems, the rank grows very slowly with electric size as square root of the logarithm of the electric size of the problems; for 3-D electrodynamic problems, the rank scales linearly with the electric size. Since N scales as electric size cube in a 3-D finite-element based analysis, k is proportional to $N^{1/3}$. Although k is not a constant any more, the resultant time complexity of the $\mathcal{H}$-matrix-based direct finite element solver is still lower than existing best complexity for 3-D electrodynamic analysis enabled by nested dissection, which is $O(N^2)$, for directly solving the finite element matrix.

In this work, in light of the fact that the rank of a 2-D electrodynamic problem grows with the electric size much more slowly than the rank of a 3-D problem, we propose layered $\mathcal{H}$-matrix-based algorithms to transform the original 3-D problem to multiple 2-D problems to solve. As a result, the rank's growth rate with electric size is significantly reduced from a 3-D based growth rate to a 2-D based growth rate. The storage cost of the $\mathcal{H}$-matrix-based direct finite element solver is reduced from $O(k_{\rm 3D}N\log N)$ to $O(k_{\rm 2D}M\log M)$, and the time cost is reduced from $O(k_{\rm 3D}^2N\log^2 N)$ to $O(k_{\rm 2D}^2N\log^2 M)$, where M is the number of unknowns in a single 2-D problem, which can be orders of magnitude smaller than N, and $k_{\rm 2D}$ is the rank of a 2-D problem, which grows with electric size very slowly as compared to the 3-D rank $k_{\rm 3D}$. If the structure is periodic, the time cost of the proposed direct solver is $\log_2 pO(k_{\rm 2D}^2M\log^2 M)$, where p is the number of periods.

For a general 3-D problem having an equal dimension along each of the x-, y-, and z-directions, M is proportional to $N^{2/3}$, and k_{2D} is in the order of $\sqrt{\log(N^{1/3})}$. As a result, the time complexity of the proposed direct finite element solver is $\mathrm{O}(N\log^3 N)$ in inverse and LU factorization, while the storage complexity is less than $\mathrm{O}(N)$ for obtaining the solution in any 2-D domain of interest.

The rest of the paper is organized as follows. In Section II, we introduce the background of an $\mathcal{H}$-matrix based direct FEM solver for electrodynamic analysis. In Section III, we present the proposed layered $\mathcal{H}$-matrix-based inverse algorithm. In Section IV, we elaborate a layered $\mathcal{H}$-matrix-based LU factorization and solution algorithm. In Section V, the rank and the complexity of the proposed direct solvers are analyzed. In Section VI, a further complexity reduction for periodic structures is shown. In Section VII, numerical results are presented to demonstrate the accuracy and the efficiency of the proposed direct FEM solvers. Section VIII relates to our conclusions.

II. Preliminaries

In this section, we give an overview of the basics of an $\mathcal{H}$-matrix based direct finite element solver.

A. Formulate an FEM System Matrix

Consider the second-order vector wave equation:

$$\nabla\times\left(\frac{1}{\mu}\nabla\times\vec{E}\right)-k_0^2\overline{\varepsilon}_r\,\vec{E}=-jk_0Z_0\,\vec{J} \tag{1}$$

subject to boundary conditions:

$$\hat{n}\times\vec{E}=\vec{P}\ \text{ on } \mathrm{S}_1 \tag{2}$$

$$\frac{1}{\mu_r}\hat{n}\times\left(\nabla\times\vec{E}\right)+\gamma_e\hat{n}\times\left(\hat{n}\times\vec{E}\right)=\vec{U}\ \text{ on } \mathrm{S}_2. \tag{3}$$

An FEM-based solution to the above boundary value problem results in a linear system of equation [14]:

$$\mathbf{Y}\{E\}=b \tag{4}$$

where b is the right hand side, $\mathbf{Y}$ is sparse and can be written as

$$\mathbf{Y}=-k_0^2\mathbf{T}+\mathbf{S}+\mathbf{B}, \tag{5}$$

in which

$$\begin{aligned}\mathbf{T}&=\iiint_V[\overline{\varepsilon}_r\mathbf{N}_i\cdot\mathbf{N}_j]dV\\ \mathbf{S}&=\iiint_V\left[\frac{1}{\mu_r}(\nabla\times\mathbf{N}_i)\cdot(\nabla\times\mathbf{N}_j)\right]dV\\ \mathbf{B}&=\iint_{s_2}[\gamma_e(\hat{n}\times\mathbf{N}_i)\cdot(\hat{n}\times\mathbf{N}_j)]dS,\end{aligned} \tag{6}$$

where V denotes the computational domain, and $\mathbf{N}$ is the vector basis used to expand unknown $\mathbf{E}$. In (6), $\mathbf{T}$ is known to be a mass matrix, and $\mathbf{S}$ is a stiffness matrix. $\mathbf{T}$ is positive definite, $\mathbf{S}$ is semi-positive definite, and the combined system $\mathbf{Y}$ is, in general, indefinite.

B. Build Cluster Tree and Block Cluster Tree

To construct an $\mathcal{H}$-matrix representation of $\mathbf{Y}$ and its inverse, we first build a cluster tree and then a block cluster tree [8]. A cluster tree is used to efficiently store the $\mathcal{H}$-matrix-based representation of the FEM system matrix as well as its inverse. Before detailing the procedure of building a cluster tree, we introduce the following notations:

- $\mathcal{I}:=\{1,2,\ldots,N\}$ is the index set containing the indices of all the basis functions used to discretize (1). A cluster tree constructed based on $\mathcal{I}$ is denoted by $T_{\mathcal{I}}$.
- Ω_i is the support of the i-th vector basis function $\vec{N}_i$. Ω_i can be chosen to be the bounding box that comprises all the elements sharing $\vec{N}_i$.
- For a subset t of $\mathcal{I}$, Ω_t is defined as: $\Omega_t:=\cup_{i\in t}\Omega_i$, which is a union of the supports of all the basis functions in t.

To build a cluster tree, we start from the whole index set $\mathcal{I}:=\{1,2,\ldots,N\}$, which is the root cluster denoted by $\mathcal{I}_1^{(0)}$ (the superscript represents the level of the set and the subscript denotes the index of the set at this level). We then choose the coordinate direction of maximal extent and split set $\mathcal{I}$ into two subsets $\mathcal{I}_1^{(1)}$ and $\mathcal{I}_2^{(1)}$. This process continues until the size of each subset is smaller than a pre-determined parameter $n_{\min}$ (leafsize), which is used to control the depth of the cluster tree.

A block cluster tree $T_{\mathcal{I}\times\mathcal{J}}$ is built from two cluster trees $T_{\mathcal{I}}$ and $T_{\mathcal{J}}$. Each block cluster $b\in T_{\mathcal{I}\times\mathcal{J}}$ has a form $b=(t,s)$ with clusters $t\in T_{\mathcal{I}}$ and $s\in T_{\mathcal{J}}$, and b,t,s at the same level. Given an admissibility condition:

$$\min\{\mathrm{diam}(\Omega_t),\mathrm{diam}(\Omega_s)\}\le\eta\,\mathrm{dist}(\Omega_t,\Omega_s), \tag{7}$$

where $\mathrm{diam}(\cdot)$ is the Euclidean diameter of a set, $\mathrm{dist}(\cdot,\cdot)$ is the Euclidean distance between two sets, and η is a positive constant. The block cluster tree can be constructed by testing the admissibility condition of the blocks level by level starting with $\mathrm{Root}(T_{\mathcal{I}})$ and $\mathrm{Root}(T_{\mathcal{J}})$, and descending in the tree. If at one level, clusters t and s satisfy the admissibility condition, the block cluster (t,s) is admissible and its children blocks do not need to be checked. If the clusters t and s do not satisfy the admissibility condition, their children are examined. The above process continues until the leaf level is reached. A leaf of block cluster tree $T_{\mathcal{I}\times\mathcal{J}}$ is either admissible or non-admissible. The leaves of $T_{\mathcal{I}\times\mathcal{J}}$ constitute an admissible partition: $\mathcal{P}:=\mathcal{L}(T_{\mathcal{I}\times\mathcal{J}})$. They can be located at different levels of the block cluster tree, thus having different matrix sizes.

C. Generate an $\mathcal{H}$-Matrix Representation of the FEM System Matrix

As shown in the subsection above, there are two kinds of leaves in a block cluster tree $T_{\mathcal{I}\times\mathcal{J}}$: non-admissible leaves and

admissible leaves. In an $\mathcal{H}$-matrix, matrix blocks in non-admissible leaves are stored in a full matrix form, namely all the matrix entries are stored without any approximation. Matrix blocks in admissible leaves $\mathbf{M}_{t\times s}$ defined by the row cluster t and column cluster s are stored in a factorized form: $\mathbf{M}_{t\times s} = \mathbf{A}\mathbf{B}^{\mathrm{T}}$, where $\mathbf{A}$ is a $t \times k$ matrix and $\mathbf{B}$ is an $s \times k$ matrix, with k being the blockwise rank. The rank k can be used to control the accuracy of the approximation. The definition of the set of $\mathcal{H}$-matrices on the block cluster tree $T_{\mathcal{I}\times\mathcal{J}}$ with admissible partition $\mathcal{P} := \mathcal{L}(T_{\mathcal{I}\times\mathcal{J}})$ and blockwise rank k can be written as:

$$\mathcal{H}(T_{\mathcal{I}\times\mathcal{J}}, k) = \{\mathbf{A} \in \mathbb{C}^{\mathcal{I}\times\mathcal{J}} : \operatorname{rank} A_b \le k \text{ for all admissible } b \in \mathcal{P}\},$$

where A_b denotes the matrix corresponding to block b. In mathematical literature of the $\mathcal{H}$-matrices, k is, in general, treated as a constant that does not depend on N. When using $\mathcal{H}$-matrix based mathematics for electrodynamic analysis, the rank k of the *inverse* finite element matrix is electric size, and hence N dependent in 2- and 3-D analysis [10], [11]. However, k is still less than N, thus being low rank [5], [10], [11].

In the process of constructing an $\mathcal{H}$-matrix to represent the original FEM matrix, all the nonzero matrix entries in the FEM matrix are stored in non-admissible leaves and admissible leaves do not need to be filled because the corresponding matrix elements are zero. In other words, all the admissible blocks in the original FEM matrix have a zero rank [5]. Thus, an FEM matrix can be represented exactly by an $\mathcal{H}$-matrix without approximation, and the assembly of an FEM matrix has optimal complexity $\mathrm{O}(N)$.

D. $\mathcal{H}$-Matrix Inverse and LU Decomposition

After the $\mathcal{H}$-matrix representation is generated for the FEM matrix, an $\mathcal{H}$-matrix based direct solution can be computed by inverting $\mathbf{Y}$ or factorizing $\mathbf{Y}$ into $\mathbf{L}$ and $\mathbf{U}$ factors.

Rewriting the FEM matrix $\mathbf{Y}$ in the following form:

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{pmatrix}. \tag{8}$$

Its inverse can be computed hierarchically using (9), shown at the bottom of the page, where $\mathbf{S} = \mathbf{Y}_{22} \oplus (-\mathbf{Y}_{21} \otimes \mathbf{Y}_{11}^{-1} \otimes \mathbf{Y}_{12})$, and all the additions ($\oplus$) and multiplications ($\otimes$) are performed based on $\mathcal{H}$-matrix arithmetic [6]–[9], which is much faster than the conventional matrix additions and multiplications. For example, for a dense matrix of size N, an $\mathcal{H}$-based matrix addition has $O(N \log N)$ complexity, whereas an $\mathcal{H}$-based matrix-matrix multiplication has $O(N \log^2 N)$ complexity. The $\mathcal{H}$-LU-based direct solution [5] has two components: 1) $\mathcal{H}$-based recursive LU factorization; 2) matrix solution by $\mathcal{H}$-based backward and forward substitution.

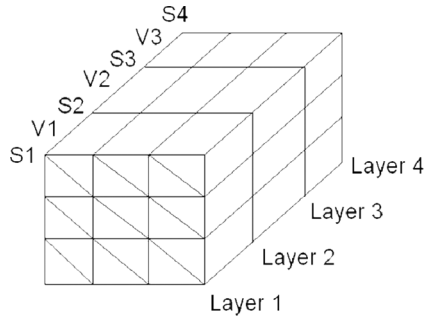


Fig. 1. A 3-D problem sliced into layers.

The cost of storage and CPU time for an $\mathcal{H}$-matrix based inverse as well as LU factorization are shown to be $O(k_{3\mathrm{D}} N \log N)$, and $O(k_{3\mathrm{D}}^2 N \log^2 N)$ respectively in [5] for solving 3-D electrodynamic problems, where $k_{3\mathrm{D}}$ is the rank of the inverse finite element matrix for a prescribed accuracy.

III. Proposed Layered $\mathcal{H}$-Inverse Based Direct Finite Element Solver

Given a general 3-D electromagnetic problem, we transform the inverse of the original 3-D FEM matrix of size N to the inverses of L 2-D problems, the size of each of which is bounded by M with $LM = O(N)$. By doing so, we reduce the size of the matrices to be computed from N to M. In addition, we reduce the growth rate of the rank from the original 3-D growth rate to a 2-D growth rate, thus further accelerating the $\mathcal{H}$-matrix based fast direct solution of the FEM matrix. The details of the proposed algorithm are given below.

Since an FEM matrix is sparse, its direct solution can be decomposed into the direct solution of 2-D problems in many ways. Here, we decompose the original 3-D solution into the solution of 2-D layers. We slice a 3-D problem into layers. The material in each layer is not required to be uniform in the proposed algorithm. It can be arbitrarily non-uniform. We categorize unknowns into two groups: surface unknowns and volume unknowns as shown in Fig. 1. Surface unknowns are on the interfaces that separate layers, while volume unknowns are internal to each layer. The mesh in each layer can be different from that in other layers. It can be a triangular prism element or a tetrahedral element based mesh. The resultant number of unknowns can be different in different layers. If the unknowns are ordered layer by layer, the resultant FEM matrix is, also, layered as shown in Fig. 2, where shaded blocks are non-zero blocks and others are all zeros. Although equal-size blocks are shown, each block can have a different size.

$$\mathbf{Y}^{-1} = \begin{pmatrix} \mathbf{Y}_{11}^{-1} \oplus \mathbf{Y}_{11}^{-1} \otimes \mathbf{Y}_{12} \otimes \mathbf{S}^{-1} \otimes \mathbf{Y}_{21} \otimes \mathbf{Y}_{11}^{-1} & -\mathbf{Y}_{11}^{-1} \otimes \mathbf{Y}_{12} \otimes \mathbf{S}^{-1} \\ -\mathbf{S}^{-1} \otimes \mathbf{Y}_{21} \otimes \mathbf{Y}_{11}^{-1} & \mathbf{S}^{-1} \end{pmatrix} \tag{9}$$

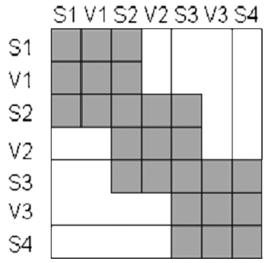


Fig. 2. FEM matrix resulting from a layered ordering.

When a structure is complicated, it can be cumbersome to partition the structure into layers geometrically. In this case, a numerical partition can be employed. In other words, the layers can be numerically formed instead of geometrically identified. In a numerical way, regardless of the structure and its mesh, we can start from the set that includes all the unknown indexes, then partition this set into two disjoint subsets along one direction based on the physical coordinates of these unknowns. Each subset represents one part of the original structure, which is then partitioned to two disjoint subsets again. This process continues until the required single-layer size is reached. For example, one can set the number of unknowns in each single layer to be no greater than a predefined number, M. Then once the number of unknowns in a subset is reduced to M, we stop partitioning this subset. After this process, the entire unknown set is partitioned into L subsets, where $L \sim N/M$. The unknowns in each subset, the size of which is $O(M)$, correspond to the unknowns in a single layer. In other words, each subset constitutes a single 'layer'. With the unknown indexes known in each layer, the matrix blocks corresponding to each layer and the interaction between this layer and its adjacent two layers can be identified numerically.

A. Introduction of Notations

We denote surface unknowns and volume unknowns in the i-th layer by $\mathrm{S}i$ and $\mathrm{V}i$, respectively. We denote the maximal number of unknowns on a single surface by M_{s}, the maximal number of volume unknowns in a single layer by M_{V}, and the total number of layers by L.

We use $\mathbf{Y}_{11}^{(l)}$, $\mathbf{Y}_{12}^{(l)}$, $\mathbf{Y}_{21}^{(l)}$, and $\mathbf{Y}_{22}^{(l)}$ with $l = 1, 2, \ldots, L$ to respectively represent the $\mathbf{Y}_{11}$, $\mathbf{Y}_{12}$, $\mathbf{Y}_{21}$, and $\mathbf{Y}_{22}$ matrix block at the l-th level. Fig. 3 illustrates $\mathbf{Y}_{11}^{(l)}$ and $\mathbf{Y}_{22}^{(l)}$ at different levels. As can be seen from Fig. 3(a), $\mathbf{Y}_{11}^{(1)}$ is the first-layer matrix formed by (S1, V1) unknowns; $\mathbf{Y}_{11}^{(2)}$ is the matrix block of the first two layers; and $\mathbf{Y}_{11}^{(3)}$ is the matrix block of the first three layers. At the L-th level, $\mathbf{Y}_{11}^{(L)}$ becomes the entire FEM matrix $\mathbf{Y}$. From Fig. 3(b), it is clear that the $\mathbf{Y}_{22}^{(l)}$ at different levels are all of single-layer size. For example, $\mathbf{Y}_{22}^{(1)}$ is the second-layer matrix formed by (S2, V2) unknowns; and $\mathbf{Y}_{22}^{(2)}$ is the third-layer matrix formed by (S3, V3) unknowns. From Fig. 3, the $\mathbf{Y}_{12}^{(l)}$ and $\mathbf{Y}_{21}^{(l)}$ at different levels can be identified correspondingly.

We employ $\mathbf{Y}^{i,i}$ to represent the original FEM matrix in the i-th layer, i.e., the diagonal block formed by $(\mathrm{S}i, \mathrm{V}i)$ unknowns. Correspondingly, $\mathbf{Y}^{i,i+1}$ stands for the off-diagonal block formed between the i-th layer unknowns and the $(i+1)$-th layer unknowns. Different from $\mathbf{Y}^{i,i}$, the $(\mathbf{Y}^{-1})^{i,i}$ represents the inverse FEM matrix in the i-th layer.

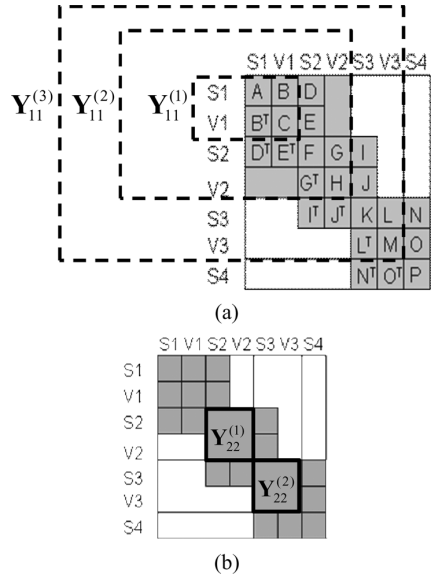


Fig. 3. Illustration of $\mathbf{Y}_{11}^{(l)}$ and $\mathbf{Y}_{22}^{(l)}$ at different levels. (a) $\mathbf{Y}_{11}^{(l)}$ $(l = 1, 2, 3)$. (b) $\mathbf{Y}_{22}^{(l)}$ $(l = 1, 2)$.

B. Proposed Layered $\mathcal{H}$-Inverse

Based on (9), we compute the inverse of an FEM matrix level by level, from the first level that only includes the first-layer matrix block to the entire matrix level that includes all the layers. Take the layered FEM matrix shown in Fig. 3(a) as an example, in the first level, $\mathbf{Y}_{11}^{(1)}$ is the first-layer matrix block formed by unknowns in the first layer. By performing an $\mathcal{H}$-matrix-based inverse of $\mathbf{Y}_{11}^{(1)}$, we obtain the inverse of the first-layer matrix block $(\mathbf{Y}_{11}^{(1)})^{-1}$. At the second level, we proceed to obtain the inverse of the first two layers, $(\mathbf{Y}_{11}^{(2)})^{-1}$, by using (9). In doing so, as can be seen from Fig. 3(a), $\mathbf{Y}_{12}^{(1)}$ is the matrix block formed between unknown set (S1 and V1) and unknown set (S2 and V2), $\mathbf{Y}_{21}^{(1)}$ is $\mathbf{Y}_{12}^{(1)T}$, and $\mathbf{Y}_{22}^{(1)}$ is the matrix formed by unknowns in the second layer, i.e., (S2 and V2). After computing (9), we obtain $(\mathbf{Y}_{11}^{(2)})^{-1}$, the inverse of the first two layers as illustrated in Fig. 4, where all the blocks are now nonzero, denoted by shaded blocks. We then proceed to compute $(\mathbf{Y}_{11}^{(3)})^{-1}$, the inverse of the first 3 layers. The $\mathbf{Y}_{11}$ block at this level, $\mathbf{Y}_{11}^{(2)}$, is the matrix block formed by the first two layers, the inverse of which has been computed; $\mathbf{Y}_{12}^{(2)}$ is the matrix block formed between unknown set (S1, V1, S2, V2) and unknown set (S3 and V3), $\mathbf{Y}_{21}^{(2)}$ is $\mathbf{Y}_{12}^{(2)T}$, and $\mathbf{Y}_{22}^{(2)}$ is the matrix formed by unknowns in the third layer, i.e., (S3 and V3). The $\mathbf{Y}_{22}^{(2)}$ consists of four matrix blocks $\mathbf{K}, \mathbf{L}, \mathbf{L}^{\mathrm{T}}$, and $\mathbf{M}$ in the original FEM matrix as shown in Fig. 5 which displays the inverted matrix $(\mathbf{Y}_{11}^{(2)})^{-1}$ together with the remaining FEM matrix to be inverted.

Based on (9), to compute $((\mathbf{Y}_{11}^{(3)})^{-1})_{22}$, which is the 22-block in the inverse matrix $(\mathbf{Y}_{11}^{(3)})^{-1}$, two steps are performed. First, we compute $\mathbf{Y}_{22}^{(2)} = \mathbf{Y}_{22}^{(2)} \oplus (-\mathbf{Y}_{21}^{(2)} \otimes (\mathbf{Y}_{11}^{(2)})^{-1} \otimes \mathbf{Y}_{12}^{(2)})$, then compute its inverse. The first step is illustrated in Fig. 6, where empty blocks are zero blocks and shaded blocks are non-zero blocks. Due to the zero blocks in $\mathbf{Y}_{21}^{(2)}$ and $\mathbf{Y}_{12}^{(2)}$, we notice that

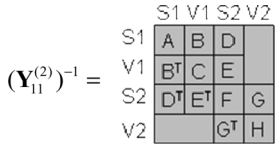


Fig. 4. The inverse matrix of the first two layers with every block nonzero. (The original FEM matrix blocks are also shown, which are overwritten by inverse blocks).

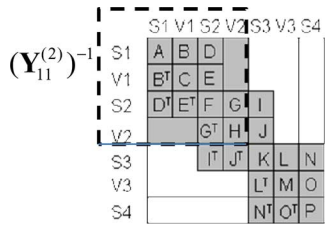


Fig. 5. The FEM matrix after computing the inverse matrix for the first two layers. What is outside the dashed box is the original FEM matrix that remains to be inverted.

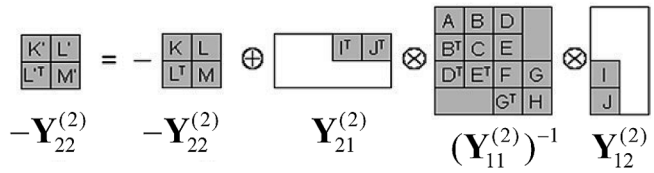


Fig. 6. Illustration of the operation $\mathbf{Y}_{22}^{(2)} = \mathbf{Y}_{22}^{(2)} \oplus (-\mathbf{Y}_{21}^{(2)} \otimes (\mathbf{Y}_{11}^{(2)})^{-1} \otimes \mathbf{Y}_{12}^{(2)})$.

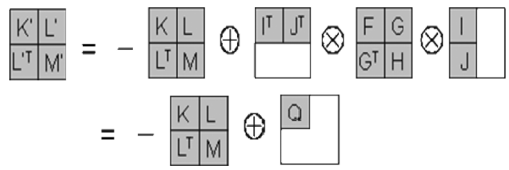


Fig. 7. Actual operations in $\mathbf{Y}_{22}^{(2)} = \mathbf{Y}_{22}^{(2)} \oplus (-\mathbf{Y}_{21}^{(2)} \otimes (\mathbf{Y}_{11}^{(2)})^{-1} \otimes \mathbf{Y}_{12}^{(2)})$, where $\mathbf{Q} = (\mathbf{I}^{\mathrm{T}}\mathbf{F} + \mathbf{J}^{\mathrm{T}}\mathbf{G})\mathbf{I} + (\mathbf{I}^{\mathrm{T}}\mathbf{G} + \mathbf{J}^{\mathrm{T}}\mathbf{H})\mathbf{J}$.

only $\mathbf{F}, \mathbf{G}, \mathbf{G}^{\mathrm{T}}$, and $\mathbf{H}$ blocks of $(\mathbf{Y}_{11}^{(2)})^{-1}$ are involved in the multiplication. All the other blocks in $(\mathbf{Y}_{11}^{(2)})^{-1}$ are not needed in the multiplication. These four blocks are, in fact, the 22-block of $(\mathbf{Y}_{11}^{(2)})^{-1}$, i.e., $((\mathbf{Y}_{11}^{(2)})^{-1})_{22}$. Thus, we only need to store $((\mathbf{Y}_{11}^{(2)})^{-1})_{22}$ that are of single-layer dimension instead of the entire $(\mathbf{Y}_{11}^{(2)})^{-1}$. In addition to single-layer storage, the matrix multiplication cost is also of single-layer cost as can be seen from Fig. 7. The above is true for the computation at any level. Therefore, when proceeding from one level to the other level. We only need to keep the 22-block of the inverse computed at the previous level.

In a conventional scheme, for the computation associated with the i-th level, let $M = Ms + Mv, \mathbf{Y}_{22}^{(i)}$ is an $O(M) \times O(M)$ matrix, $\mathbf{Y}_{21}^{(i)}$ is an $O(M) \times O[(i-1)M]$ matrix, $\mathbf{Y}_{11}^{(i)}$ is an $O[(i-1)M] \times O[(i-1)M]$ matrix, and $\mathbf{Y}_{12}^{(i)}$ is an $O[(i-1)M] \times O(M)$ matrix. The size of matrices involved in the computation keeps increasing with the level. In contrast, in the proposed layered inverse, as shown in Fig. 7, $\mathbf{Y}_{21}^{(i)}, ((\mathbf{Y}_{11}^{(i)})^{-1})_{22}, \mathbf{Y}_{12}^{(i)}$, and $\mathbf{Y}_{22}^{(i)}$ all have a single-layer size. To be specific, $\mathbf{Y}_{22}^{(i)} = \mathbf{Y}^{i+1,i+1}; \mathbf{Y}_{21}^{(i)} = \mathbf{Y}^{i+1,i}$; and $\mathbf{Y}_{12}^{(i)} = \mathbf{Y}_{21}^{(i)T}$. More importantly, only $((\mathbf{Y}_{11}^{(i)})^{-1})_{22}$, which is the inverse in the i-th single layer, denoted by $\tilde{\mathbf{Y}}^{i,i}$, is used in the computation of inverse in the $(i+1)$-layer. This process continues until the inverse in the last layer is computed. Thus, the computation for each layer only involves the operation of matrices of a single-layer size. Therefore, the computational cost for each layer is fixed. In addition, the total storage is the storage of two-layer matrices since it can be reused for each layer.

The pseudo-code for obtaining the inverse in the last layer is shown in (10) at the bottom of the page. In this code, the storage space opened up for $\mathbf{Y}_{11}, \mathbf{Y}_{12}, \mathbf{Y}_{21}$, and $\mathbf{Y}_{22}$ are reused for each layer. The function $\mathcal{H}$-inverse $(\mathbf{Y}_{11}, \mathbf{X}_{11})$ takes $\mathbf{Y}_{11}$ as the input, which is then overwritten by its inverse $\mathbf{Y}_{11}^{-1}$ at the output. The $\mathbf{X}_{11}$ is a temporary space used during the inverse procedure [5]. At the end of the computation, the inverse of the last layer, $(\mathbf{Y}^{-1})^{L,L}$, is stored in $\mathbf{Y}_{22}$. In each layer, the surface and volume unknowns are partitioned based on the admissibility

Pseudo code for computing layered $\mathcal{H}-$inverse

Procedure Layered_$\mathcal{H}$inverse$(\mathbf{Y}, \mathbf{X})$
1) $i = 1$;
Let $\mathbf{Y}_{11} = \mathbf{Y}^{i,i}, \mathbf{Y}_{22} = \mathbf{Y}^{i+1,i+1}; \mathbf{Y}_{21} = \mathbf{Y}^{i+1,i}; \mathbf{Y}_{12} = \mathbf{Y}_{21}^{T}$
2) $\mathcal{H}-$inverse$(\mathbf{Y}_{11}, \mathbf{X}_{11})$
for $i = 2$ to $L-1$
3) $\mathbf{Y}_{22} = \mathbf{Y}_{22} \oplus (-\mathbf{Y}_{21} \otimes \mathbf{Y}_{11} \otimes \mathbf{Y}_{12})$
4) $\mathcal{H}-$inverse$(\mathbf{Y}_{22}, \mathbf{X}_{22})$
5) $\mathbf{Y}_{11} = \mathbf{Y}_{22}$
6) $\mathbf{Y}_{22} = \mathbf{Y}^{i+1,i+1}; \mathbf{Y}_{21} = \mathbf{Y}^{i+1,i}; \mathbf{Y}_{12} = \mathbf{Y}^{i,i+1}$
end for
7) $\mathbf{Y}_{22} = \mathbf{Y}_{22} \oplus (-\mathbf{Y}_{21} \otimes \mathbf{Y}_{11} \otimes \mathbf{Y}_{12})$
8) $\mathcal{H}-$inverse$(\mathbf{Y}_{22}, \mathbf{X}_{22})$

(10)

condition, and all the matrix additions and multiplications involved in the computation of (10) are performed based on the $\mathcal{H}$-matrix based fast arithmetic [6]–[9].

In addition to the inverse in the last layer $(\mathbf{Y}^{-1})^{L,L}$, the aforementioned algorithm can be used to efficiently obtain the inverse in any i-th layer $(\mathbf{Y}^{-1})^{i,i}$, where i is from 1 to L. If the layer is in between the first and the last layer, the aforementioned numerical procedure can be performed in a top-down bottom-up manner until along both directions the operations have reached the i-th layer. At this point, the original i-th layer FEM matrix $\mathbf{Y}^{i,i}$ will be superposed with $-\mathbf{Y}^{i,i-1} \otimes \tilde{\mathbf{Y}}^{i-1,i-1} \otimes \mathbf{Y}^{i-1,i}$ obtained from the top-down layered operations as well as $-\mathbf{Y}^{i,i+1} \otimes \tilde{\mathbf{Y}}^{i+1,i+1} \otimes \mathbf{Y}^{i+1,i}$ obtained from the bottom-up layered operations. The superposed matrix will then be inverted to obtain $(\mathbf{Y}^{-1})^{i,i}$.

IV. Proposed Layered $\mathcal{H}$-LU Based Direct FEM Solver

Compared to inversion, an LU factorization based direct solution is more efficient when the number of right hand sides is less than N. Therefore, we also develop a layered $\mathcal{H}$-LU based direct solver to complement the layered $\mathcal{H}$-inverse based direct solver. This algorithm can be used to efficiently obtain the factorized $\mathbf{L}$ and $\mathbf{U}$ in any selected layer of interest.

Consider a non-leaf cluster t in the cluster tree $T_{\mathcal{I}}$ of an $\mathcal{H}$-based representation of the FEM matrix $\mathbf{Y}$. The matrix block $\mathbf{Y}_{tt}$ can be subdivided into four sub blocks:

$$\mathbf{Y}_{tt} = \begin{pmatrix} \mathbf{Y}_{t_1t_1} & \mathbf{Y}_{t_1t_2} \\ \mathbf{Y}_{t_2t_1} & \mathbf{Y}_{t_2t_2} \end{pmatrix}, \tag{11}$$

where t_1 and t_2 are the children of t in the cluster tree $T_{\mathcal{I}}$.

Assuming $\mathbf{Y}$ can be factorized into $\mathbf{L}$ and $\mathbf{U}$ matrices, $\mathbf{Y}$ can also be written as:

$$\begin{aligned} \mathbf{Y}_{tt} = \mathbf{L}_{tt}\mathbf{U}_{tt} &= \begin{pmatrix} \mathbf{L}_{t_1t_1} & 0 \\ \mathbf{L}_{t_2t_1} & \mathbf{L}_{t_2t_2} \end{pmatrix} \begin{pmatrix} \mathbf{U}_{t_1t_1} & \mathbf{U}_{t_1t_2} \\ 0 & \mathbf{U}_{t_2t_2} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{L}_{t_1t_1}\mathbf{U}_{t_1t_1} & \mathbf{L}_{t_1t_1}\mathbf{U}_{t_1t_2} \\ \mathbf{L}_{t_2t_1}\mathbf{U}_{t_1t_1} & \mathbf{L}_{t_2t_1}\mathbf{U}_{t_1t_2} + \mathbf{L}_{t_2t_2}\mathbf{U}_{t_2t_2} \end{pmatrix}. \end{aligned} \tag{12}$$

By comparing (11) and (12), it can be seen that the LU factorization can be computed recursively by the following four steps:

1) Compute $\mathbf{L}_{t_1t_1}$ and $\mathbf{U}_{t_1t_1}$ by $\mathcal{H}$-LU factorization $\mathbf{Y}_{t_1t_1} = \mathbf{L}_{t_1t_1}\mathbf{U}_{t_1t_1}$;
2) Compute $\mathbf{U}_{t_1t_2}$ by solving $\mathbf{L}_{t_1t_1}\mathbf{U}_{t_1t_2} = \mathbf{Y}_{t_1t_2}$;
3) Compute $\mathbf{L}_{t_2t_1}$ by solving $\mathbf{L}_{t_2t_1}\mathbf{U}_{t_1t_1} = \mathbf{Y}_{t_2t_1}$;
4) Compute $\mathbf{L}_{t_2t_2}$ and $\mathbf{U}_{t_2t_2}$ by $\mathcal{H}$-LU factorization $\mathbf{L}_{t_2t_2}\mathbf{U}_{t_2t_2} = \mathbf{Y}_{t_2t_2} - \mathbf{L}_{t_2t_1}\mathbf{U}_{t_1t_2}$.

All the additions and multiplications involved in the four steps are performed by $\mathcal{H}$-matrix-based fast arithmetic [6]–[9].

Next, we use the layered FEM matrix shown in Fig. 8 as an example to elaborate the proposed layered $\mathcal{H}$-LU. Without loss of generality, assuming that the $\mathbf{L}$ and $\mathbf{U}$ factors of the last layer are of interest. Now consider non-leaf cluster t that includes all the unknowns in the first three layers from S1 to V3. Its two children clusters are t_1 that includes all the unknowns in the first two layers from S1 to V2, and t_2 that includes unknowns from S3 to V3. The corresponding FEM matrix blocks are denoted

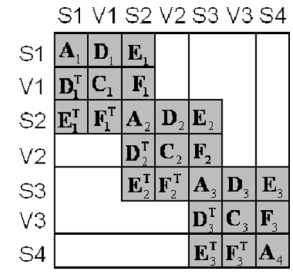


Fig. 8. FEM matrix with layered ordering.

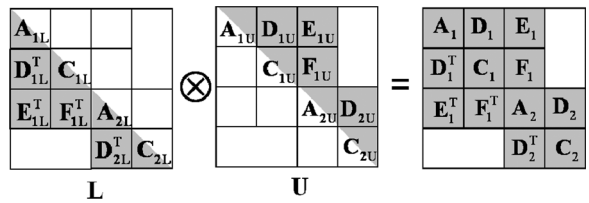


Fig. 9. $\mathbf{L}$ and $\mathbf{U}$ factors of the first two layers.

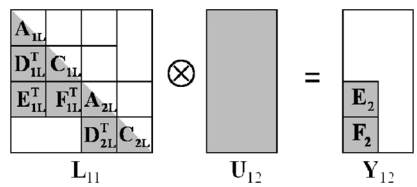


Fig. 10. Illustration of operation $\mathbf{L}_{11}\mathbf{U}_{12} = \mathbf{Y}_{12}$.

by $\mathbf{Y}_{11}$, and $\mathbf{Y}_{22}$ respectively. We first perform an $\mathcal{H}$-LU factorization $\mathbf{Y}_{11} = \mathbf{L}_{11}\mathbf{U}_{11}$ to obtain the $\mathbf{L}$ and $\mathbf{U}$ factors of $\mathbf{Y}_{11}$ as shown in Fig. 9. We start from the first layer, and then proceed to the second layer. With the LU factorization of the first 2 layers computed as shown in Fig. 9, we proceed to compute the LU factorization of the first 3 layers. The factorized matrices $\mathbf{L}$ and $\mathbf{U}$ shown in the left hand side of Fig. 9 will be used for the computation of the LU factorization of the next-level matrix.

Next, to obtain the LU factorization of cluster t corresponding to the matrix of the first three layers, three steps are performed:

1) Solve $\mathbf{L}_{11}\mathbf{U}_{12} = \mathbf{Y}_{12}$;
2) Solve $\mathbf{L}_{21}\mathbf{U}_{11} = \mathbf{Y}_{21}$;
3) Do LU factorization $\mathbf{L}_{22}\mathbf{U}_{22} = \mathbf{Y}_{22} - \mathbf{L}_{21}\mathbf{U}_{12}$.

The first step is depicted in Fig. 10. A closer look of Fig. 10 reveals that due to zero blocks in $\mathbf{Y}_{12}$, only $\mathbf{A}_{2\mathrm{L}}$, $\mathbf{D}_{2\mathrm{L}}^{\mathrm{T}}$, and $\mathbf{C}_{2\mathrm{L}}$ blocks in $\mathbf{L}_{11}$ are involved in the process of solving for $\mathbf{U}_{12}$. All the other blocks in $\mathbf{L}_{11}$ are not needed in the solution. Thus, we only need to store three blocks that are of single-layer dimension instead of the entire $\mathbf{L}_{11}$ when proceeding from one layer to the next layer. Not only the cost of storage is reduced to single-layer cost, but also the computation cost, as can be seen from Fig. 11. Similarly, Step 2 for computing L_{21} can be performed with single-layer cost as illustrated in Fig. 13 instead of using the conventional scheme shown in Fig. 12. Since both $\mathbf{L}_{21}$ and $\mathbf{U}_{12}$ are single-layer matrices, the computational cost of Step 3 is also of single-layer cost.

In the conventional $\mathcal{H}$-LU factorization scheme, for the computation associated with the ith layer, $\mathbf{Y}_{22}$ is an $O(M)\times O(M)$ matrix, $\mathbf{Y}_{21}$ is an $O(M)\times O[(i-1)M]$ matrix, Y_{11} is an $O[(i-1)M]\times O[(i-1)M]$ matrix, and Y_{12} is an $O[(i-1)M]\times O(M)$

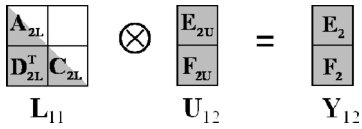


Fig. 11. Actual operations for solving $\mathbf{L}_{11}\mathbf{U}_{12} = \mathbf{Y}_{12}$.

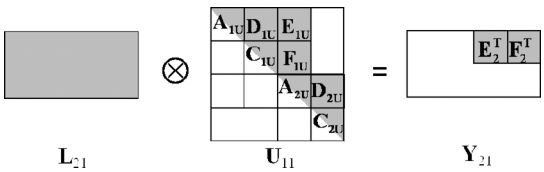


Fig. 12. Illustration of operation $\mathbf{L}_{21}\mathbf{U}_{11} = \mathbf{Y}_{21}$.

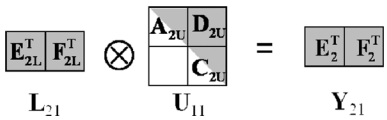


Fig. 13. Actual operations for solving $\mathbf{L}_{21}\mathbf{U}_{11} = \mathbf{Y}_{21}$.

matrix, where M is the number of unknowns in a single layer. Except for $\mathbf{Y}_{22}$, the size of the other three matrix blocks $\mathbf{Y}_{21}$, $\mathbf{Y}_{11}$ and $\mathbf{Y}_{12}$ keeps increasing with the layer index. In contrast, in the proposed layered LU factorization, $\mathbf{Y}_{21}$, $\mathbf{Y}_{11}$ and $\mathbf{Y}_{12}$ all have a fixed size of $O(M) \times O(M)$. More importantly, only the factorized $\mathbf{Y}_{22}$ in the previous layer is used in the computation of the LU factorization of $\mathbf{Y}_{22}$ at the current layer. The LU factors of $\mathbf{Y}_{22}$ at the current layer will then be used for the $\mathbf{Y}_{22}$ factorization at the next layer. The computation for each layer only involves the operation of matrices with a single-layer size, and only requires the $\mathbf{Y}_{22}$ matrix from the previous-layer computation. This procedure stops when the LU factorization of $\mathbf{Y}_{22}$ for the last layer is computed. The computational cost for each layer is in the same order and the total storage is just the storage of two-layer matrices since the storage can be reused for each layer.

The pseudo-code of the proposed layered $\mathcal{H}$-LU factorization for obtaining the LU factors in the last layer is shown in (13), at the bottom of the page. In each layer, the surface and volume unknowns are partitioned based on the admissibility condition, the $\mathcal{H}$-matrix based fast operations are performed for the matrices of a single-layer size.

V. Rank and Complexity Analysis

In this Section, we analyze the rank required by the proposed layered $\mathcal{H}$-matrix algorithms as well as their computational complexity.

A. Rank Analysis

From Sections III and IV, it can be seen that the proposed layered $\mathcal{H}$-inverse and LU algorithms transform the direct solution of the original $O(N)$ 3-D problem to the solution of multiple single-layer problems, each of which is essentially a 2-D problem since the third dimension (along layer growth direction) has a constant number of discretization elements. In a single layer, this number is one.

In each single layer i, as can be seen from (10), the essential operation performed by the proposed layered $\mathcal{H}$-inverse is to compute the inverse of the Schur complement, i.e., $(\mathbf{Y}_{22} - \mathbf{Y}_{21}\tilde{\mathbf{Y}}_{11}^{-1}\mathbf{Y}_{12})^{-1}$, where $\mathbf{Y}_{22}$ is the original FEM matrix in the i-th single layer being computed, the unknowns of which are Si and Vi, while $\mathbf{Y}_{21}$ is the original FEM matrix block formed between unknowns in the i-th layer and the unknowns in the $(i-1)$-th layer, and $\tilde{\mathbf{Y}}_{11}$ is the Schur complement of the FEM matrix in the $(i-1)$-th layer. Each of $\mathbf{Y}_{22}, \mathbf{Y}_{21}, \tilde{\mathbf{Y}}_{11}^{-1}$, and $\mathbf{Y}_{12}$ has a single-layer size.

To analyze the rank required by the proposed algorithm, we rewrite $(\mathbf{Y}_{22} - \mathbf{Y}_{21}\tilde{\mathbf{Y}}_{11}^{-1}\mathbf{Y}_{12})^{-1}$ as below:

$$\left(\mathbf{Y}_{22} - \mathbf{Y}_{21}\tilde{\mathbf{Y}}_{11}^{-1}\mathbf{Y}_{12}\right)^{-1} = \mathbf{Y}_{22}^{-1}\left(\mathbf{I} - \mathbf{Y}_{21}\tilde{\mathbf{Y}}_{11}^{-1}\mathbf{Y}_{12}\mathbf{Y}_{22}^{-1}\right)^{-1}. \quad (14)$$

Since the rank of a matrix product is smaller or equal to the minimum rank of the two matrices being multiplied, the rank of (14) is bounded by the rank of $\mathbf{Y}_{22}^{-1}$. Since $\mathbf{Y}_{22}$ is the original single-layer FEM matrix that is the FEM matrix formed for a 2-D single-layer problem, the rank of the proposed layered algorithms is governed by the rank of a 2-D problem (k_{2D}). As

Pseudo−code for computing layered $\mathcal{H}-$LU
Procedure Layered$-\mathcal{H}-$LU($\mathbf{Y}$)
1) $i = 1$, Let $\mathbf{Y}_{11} = \mathbf{Y}^{i,i}, \mathbf{Y}_{22} = \mathbf{Y}^{i+1,i+1}; \mathbf{Y}_{21} = \mathbf{Y}^{i+1,i}; \mathbf{Y}_{12} = \mathbf{Y}_{21}^{T}$
2) $\mathcal{H}-$LU ($\mathbf{Y}_{11}$).
for $i = 2$ to L
3) solve $\mathbf{L}_{11}\mathbf{U}_{12} = \mathbf{Y}_{12}$
4) solve $\mathbf{L}_{21}\mathbf{U}_{11} = \mathbf{Y}_{21}$
5) $\mathcal{H}-$LU($\mathbf{Y}_{22} - \mathbf{L}_{21}\mathbf{U}_{12}$).
6) $\mathbf{Y}_{11} = \mathbf{Y}_{22}$
7) $\mathbf{Y}_{22} = \mathbf{Y}^{i+1,i+1}; \mathbf{Y}_{21} = \mathbf{Y}^{i+1,i}; \mathbf{Y}_{12} = \mathbf{Y}^{i,i+1}$
end for
(13)

analyzed in [10], [11], different from a 3-D rank, k_{2D} grows very slowly with electric size as square root of the logarithm of the electric size. Moreover, the $\tilde{\mathbf{Y}}_{11}^{-1}$ in (14) is the inverse of the Schur complement of the FEM matrix in the $(i-1)$-th layer. This Schur complement is again the original FEM matrix in the $(i-1)$-th layer superposed with the contribution from the layers proceeding the $(i-1)$-th layer. Its inverse's rank is no greater than the rank of the inverse of the original FEM matrix in the $(i-1)$-th layer. Thus, the rank of $\tilde{\mathbf{Y}}_{11}^{-1}$ is also governed by k_{2D}, and so is the rank of the matrix product $\mathbf{Y}_{21}\tilde{\mathbf{Y}}_{11}^{-1}\mathbf{Y}_{12}$. As a result, not only (14) is governed by a 2-D rank, but also the matrix $(\mathbf{Y}_{22}-\mathbf{Y}_{21}\tilde{\mathbf{Y}}_{11}^{-1}\mathbf{Y}_{12})$ itself. Since in the proposed layered inverse algorithm, at every layer, (14) is computed with corresponding $\mathbf{Y}_{22}, \mathbf{Y}_{21}, \tilde{\mathbf{Y}}_{11}^{-1}$, and $\mathbf{Y}_{12}$ for each layer. The aforementioned rank analysis applies to the computation at every layer.

After proving the rank required for the proposed layered $\mathcal{H}$ -inverse computation is k_{2D} instead of k_{3D}, next, we prove that the $\mathbf{L}$ and $\mathbf{U}$ factors of an FEM matrix share the same rank as the FEM matrix inverse, and therefore their rank is also k_{2D}.

Since the FEM matrix $\mathbf{Y}$ can be factorized into $\mathbf{L}$ and $\mathbf{U}$ factors as $\mathbf{Y}=\mathbf{LU}$, we have $\mathbf{Y}^{-1}=\mathbf{U}^{-1}\mathbf{L}^{-1}$, and thus $\mathbf{L}^{-1}=\mathbf{U}\mathbf{Y}^{-1}$. We have proven in [5] that the FEM matrix inverse has an $\mathcal{H}$-matrix representation and can be written in a low-rank factorized form $\mathbf{Y}^{-1}=\mathbf{A}\mathbf{B}^{\mathbf{T}}$. Hence, $\mathbf{L}^{-1}=(\mathbf{UA})\mathbf{B}^{\mathbf{T}}$, where $\mathbf{UA}$ is an $N\times k$ matrix, k is the rank of the $\mathcal{H}$-matrix representation of the FEM matrix inverse. As a result, $\mathbf{L}^{-1}$ has an $\mathcal{H}$-matrix representation with the same rank k as the FEM matrix inverse. Because $\mathbf{L}^{-1}$ is an $\mathcal{H}$-matrix, $\mathbf{U}=\mathbf{L}^{-1}\mathbf{Y}$ is also an $\mathcal{H}$-matrix, the rank of which is bounded by the rank of the FEM matrix inverse. Similarly, we can prove that $\mathbf{L}$ can also be represented by an $\mathcal{H}$-matrix. Therefore, the previous rank analysis for the layered $\mathcal{H}$-inverse is equally applicable to the proposed layered $\mathcal{H}$-LU.

B. Storage Complexity

In the proposed layered $\mathcal{H}$-inverse and layered $\mathcal{H}$-LU algorithms, from operations at all previous layers, we only need to store the inverse of a single-layer size. Although such a matrix is a dense matrix of size M, by using the $\mathcal{H}$-matrix based data-sparse representation and applying the rank result from the subsection above, it can be stored in $O(k_{2D}M\log M)$ units. For a general 3-D problem having an equal dimension along each of the x, y, and z direction, M is proportional to $N^{2/3}$, and k_{2D} is in the order of $\sqrt{\log(N^{1/3})}$. As a result, the storage complexity is less than $\mathrm{O}(N)$.

C. Time Complexity

The computation of the layered $\mathcal{H}$-inverse and layered $\mathcal{H}$-LU is done layer by layer. For each layer the computational complexity is the same, which involves two $\mathcal{H}$-matrix based multiplications, one $\mathcal{H}$-matrix based addition, and one conventional $\mathcal{H}$-inverse. All these operations are performed on the matrices of size M. Assuming there are L layers, the total time cost is hence

$$\begin{aligned}\text{CPU Time Cost} &= LComp(one\_layer)\\ &\sim LO\left(k_{2D}^2 M\log^2 M\right)\\ &\sim O\left(k_{2D}^2 N\log^2 M\right)\end{aligned} \tag{15}$$

where N is the total number of unknowns. It can be seen that this complexity is much smaller than the conventional $\mathcal{H}$-matrix based direct solution of a 3-D electrodynamic problem.

For a general 3-D problem having an equal dimension along each of the x-, y-, and z-directions, M is proportional to $N^{2/3}$, and k_{2D} is in the order of $\sqrt{\log(N^{1/3})}$. As a result, the time complexity of the proposed direct finite element solver is $\mathrm{O}(N\log^3 N)$ in inverse and LU factorization.

VI. Further Acceleration for Periodic Structures

Many physical structures are periodic in nature. For example, antenna arrays, memory arrays, power grids, clock trees, etc. To solve a periodic structure, we can first use the proposed layered $\mathcal{H}$-inverse or layered $\mathcal{H}$-LU to reduce a one-period system matrix to a single-layer matrix associated with the top-most surface and the bottom-most surface of the period. The essential procedure of this reduction is given in [12] using conventional dense matrix operations. This reduced matrix is valid for all other periods in the structure. Assuming there are p periods in the structure to be solved, we can then conduct $\log_2(p)$ times periodic reduction to reduce p single-layer matrices into one single-layer system matrix as shown in Fig. 14.

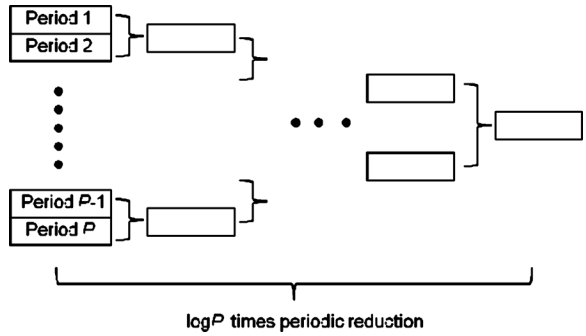


Fig. 14. Periodic reduction of the structure with p periods.

Since we only need to store a single-layer matrix, the storage complexity of the proposed acceleration for periodic structures is the same as that analyzed in Section V, while the CPU cost will be further reduced. Assume the total number of periods of the whole structure is p. In total, we only need to perform $\log_2(p)$ times periodic reduction to obtain the final single-layer matrix, each of which costs $O(k_{2D}^2 M\log^2 M)$ operations. As a result, the total time cost is $\log_2(p)O(k_{2D}^2 M\log^2 M)$.

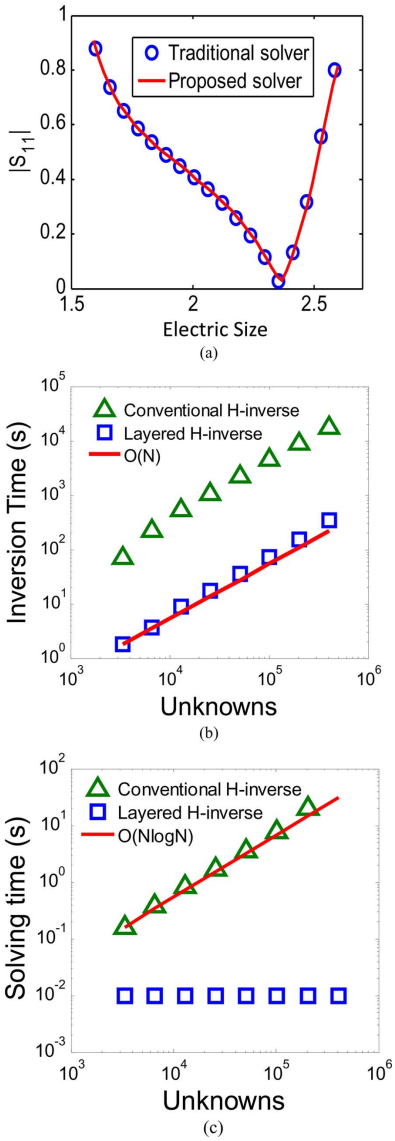


Fig. 15. Simulation of a dielectric-loaded waveguide problem from 1.2 to 64 wavelengths using the proposed layered $\mathcal{H}$-matrix inverse. (a) $|S_{11}|$; (b) CPU time for direct inverse; (c) Solving time.

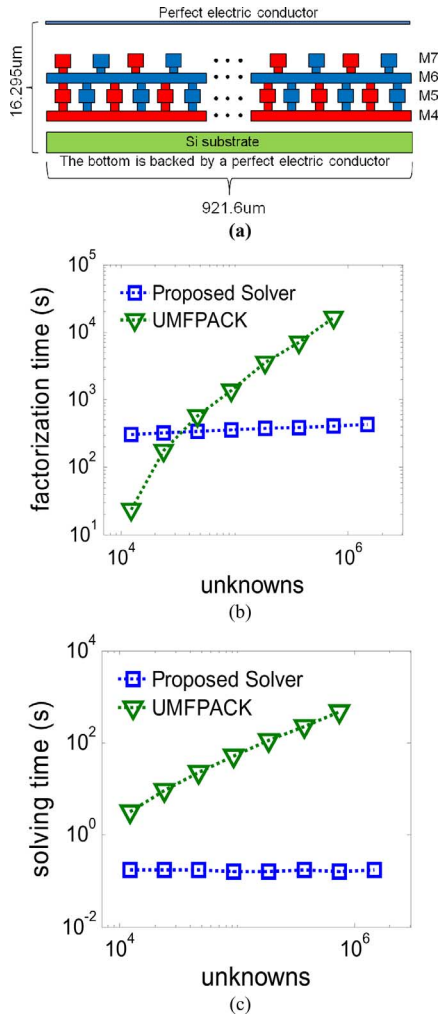


Fig. 16. Simulation of an on-chip interconnect structure using the proposed layered $\mathcal{H}$-LU direct solver. (a) Side view of the structure; (b) Factorization time; (c) Solving time.

VII. Numerical Results

A. Dielectric-Loaded Waveguide

To validate the proposed layered $\mathcal{H}$-matrix based inverse and compare its performance with the existing $\mathcal{H}$-matrix based inverse [5], we simulate a dielectric-loaded waveguide example, which was also simulated in [5]. The simulation parameters are chosen as: $\text{leafsize} = 32$ and admissibility constant $\eta = 1$. Fig. 15(a) shows the $|S_{11}|$ computed using the proposed layered $\mathcal{H}$ -inverse from 1.5 to 2.7 wavelengths. It can be seen that the computed $|S_{11}|$ agrees very well with the reference result. We then test the computational complexity of the proposed layered inverse by increasing the length of the waveguide as well as the dielectric load. The resultant electric size ranges from 1.2 to 64 wavelengths. Since the number of unknowns increases along the length direction, the number of unknowns in each layer, M, is a constant. As a result, the time complexity of the proposed layered $\mathcal{H}$-inverse is $O(N)$ for simulating this waveguide example, based on the theoretical complexity analysis given in Section V. As can be seen from Fig. 15(b), the CPU time complexity of the proposed layered $\mathcal{H}$-inverse agrees very well with the theoretical prediction. In addition, the CPU time of the proposed inverse is orders of magnitude smaller than that of the existing $\mathcal{H}$-inverse for FEM-based analysis. In Fig. 15(c), we compare the CPU time cost by the proposed solver and that by the existing $\mathcal{H}$-matrix direct solver for solving one right hand side. Unlike the existing $\mathcal{H}$-matrix direct solver, the proposed layered direct solver only needs to solve the unknowns in one layer to obtain S-parameters for this example, and hence the solution time is constant, which is independent of the number of layers. In addition, the storage of the proposed layered direct inverse is constant 3 MB for simulating this example.

B. Large-Scale On-Chip Interconnect

The proposed layered $\mathcal{H}$-LU direct solver is validated by simulating a complex 3-D on-chip interconnect structure. The side view of this interconnect structure is illustrated in Fig. 16(a). The top and bottom plates are perfect electric conductors (PEC).

TABLE I
S_{11} COMPUTED BY UMFAPCK AND THE PROPOSED SOLVER

Period	UMFPACK	Proposed solver
1	-6.584841e-01+1.129228e-02j	-6.584841e-01+1.129228e-02j
2	-6.562240e-01+1.424318e-02j	-6.562240e-01+1.424318e-02j
4	-6.509921e-01+1.948776e-02j	-6.509921e-01+1.948776e-02j
8	-6.399806e-01+2.872324e-02j	-6.399806e-01+2.872323e-02j
16	-6.171684e-01+4.248704e-02j	-6.171684e-01+4.248704e-02j
32	-5.744331e-01+5.490002e-02j	-5.744331e-01+5.490001e-02j
64	-5.138672e-01+4.946683e-02j	-5.138672e-01+4.946681e-02j
128	Out of memory	-4.605729e-01+1.623589e-02j

In between, there are four metal layers: M4, M5, M6 and M7. VCC wires (red wires) and VSS wires (blue wires) in different metal layers are connected through vias at each intersection. The wires and vias are made of copper. Different metal layers and dielectric layers are filled by dielectrics with different dielectric constants. The length of one period is 7.2 um. There are 12 ports of interest in this structure, with 6 ports at the near end and 6 ports at the far end. Since the structure of this interconnect example is periodic along length growth direction, the algorithm described in Section VI is used to efficiently reduce one-period system matrix into a single-layer matrix. The periodic reduction is then performed $\log_2(p)$ times to obtain one single-layer matrix containing near-end surface unknowns and far-end surface unknowns. By solving the resultant single-layer matrix, $\mathbf{S}$ parameters can be obtained. The state-of-the-art sparse matrix solver UMFPACK [13] is also used to compute the S-parameters of this example. The simulation is conducted at 5 GHz. The number of periods p along the length direction is increased from 1 to 128. The resultant number of unknowns ranges from 12 250 to 1 483 291 and the length of the structure is from 7.2 um to 921.6 um. The simulation parameters are chosen as: $\text{leafsize} = 32$ and admissibility constant $\eta = 1$. The rank k is adaptively determined. In Table I, we compare the S_{11} computed by UMFPACK and the proposed layered $\mathcal{H}$-LU based direct solver. Very good agreement with UMFPACK results can be observed. UMFPACK failed for the 128-period case since it ran out of memory on our machine. In contrast, the memory used by the proposed layered $\mathcal{H}$-LU based direct solver is constant 131 MB for all the testing cases from 1 period to 128 periods. The comparison with UMFPACK is shown in Fig. 16(b) and (c). In Fig. 16(b), the proposed layered $\mathcal{H}$-LU based direct solver demonstrates a much lower complexity in factorization as compared with UMFPACK. In Fig. 16(c), the CPU time required to solve 12 right hand sides for obtaining 12-port S parameters is plotted. As can be seen clearly, the proposed solver outperforms UMFPACK in CPU time. Since the number of unknowns is increased by increasing the number of periods, while the factorization as well as the solution cost for 1 period and those for 128 periods are the same in the proposed algorithm, we observe a constant CPU time cost in Fig. 16(b) and (c).

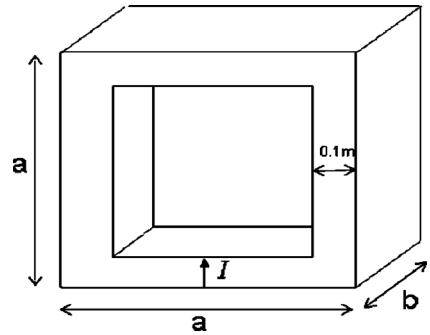


Fig. 17. Illustration of the structure.

C. A 3-D Structure With M Increasing With N

In previous two examples, the number of unknowns in a single layer, M, does not grow with N. In the third example, we consider a 3-D structure shown in Fig. 17, where M grows with N. The front and back of the structure is backed by a Neumann-type boundary condition, and the other four walls are PEC. The inner walls are also PEC. The fill-in material between inner and outer metallic walls is air. The dimension of this structure is $a \times a \times b$, where $b = a/10$. The distance between the inner walls and outer walls is 0.1 m. In the simulation, a is increased from 0.2 m to 1 m, the corresponding electric size of which is from 2 to 10 wavelengths. As a result, the number of unknowns is increased along every of x-, y-, and z-directions. The mesh size is chosen to be 10 elements per wavelength. The excitation I is placed on the front surface between the inner wall and the outer wall, as illustrated in Fig. 17. The simulation parameters are chosen as: $\text{leafsize} = 32$ and admissibility constant $\eta = 1$. In Fig. 18(a), we plot the relative error of the solution obtained by the proposed direct LU solver in comparison with the result from UMFPACK. It can be seen that the error is well controlled across the entire unknown range. The error is smaller when the number of unknowns is small because many matrix blocks are inadmissible blocks and only a few are admissible blocks. When the number unknowns increases to a certain level, the number of admissible blocks generated has become saturated, and hence the error starts to saturate to the prescribed level. In Fig. 18(b), we plot the memory usage of the proposed direct solver. The storage complexity of the proposed solver is shown to agree well with the theoretical analysis. It is much smaller than linear complexity since only a 2-D matrix needs to be stored for computing a 3-D problem in the proposed layered algorithms. We also examine the total CPU time of the proposed solver and plot it versus N in Fig. 18(c). The total CPU time clearly correlates well with the theoretical complexity data depicted by the dotted line. In Fig. 19, we plot the maximum rank adaptively determined by the proposed direct solver during the computation. It can be seen that the rank increases slowly with the electric size of the structure. It is bounded by the theoretical prediction of the 2-D rank, which is shown by the dotted line.

D. A Microstrip Patch Antenna Example

Developed as a general solver to Maxwell's equations, the proposed fast direct finite element solution can be employed for

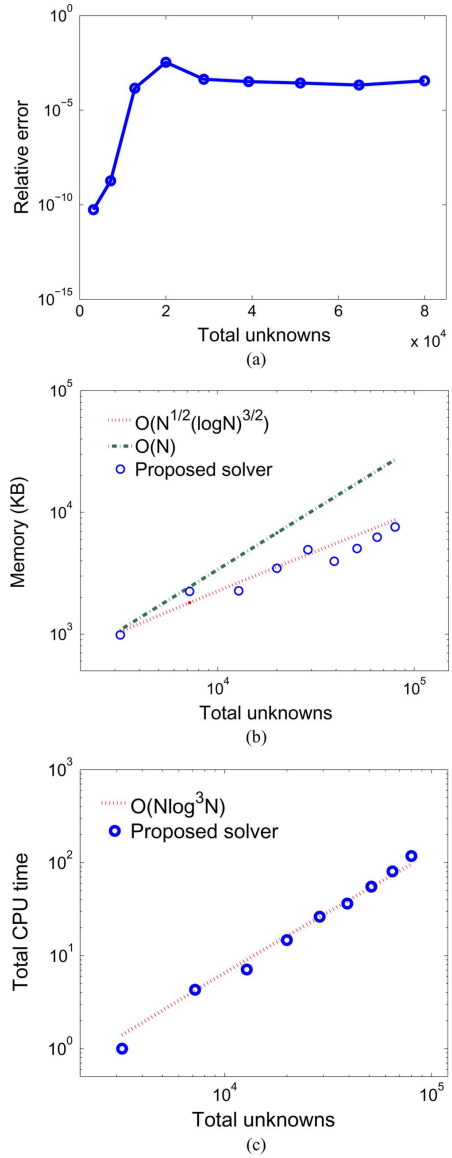


Fig. 18. Performance of the proposed direct finite element solver. (a). Accuracy. (b) Memory. (c) Total CPU time.

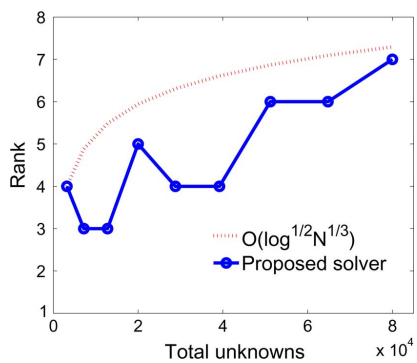


Fig. 19. Maximum rank of the proposed direct solver of a 3-D structure from 2 to 10 wavelengths.

a variety of electromagnetic applications. To demonstrate this fact, in the last example, we consider a radiation problem of

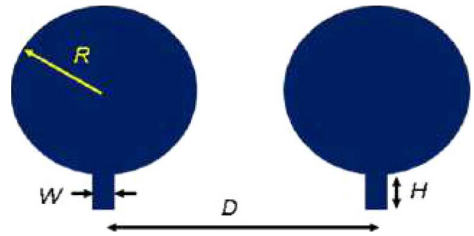


Fig. 20. Illustration of a 2-element microstrip patch antenna array.

TABLE II
COMPARISON OF S-PARAMETERS OF A PATCH ANTENNA EXAMPLE

S-para.	$\mathcal{H}$-LU	Proposed layered $\mathcal{H}$-LU
S11	-6.722305e-01 + 7.384733e-01j	-6.722299e-01 + 7.384739e-01j
S12	2.159460e-02 + 1.969724e-02j	2.159414e-02 + 1.969679e-02j
S21	2.159460e-02 + 1.969724e-02j	2.159414e-02 + 1.969679e-02j
S22	-6.722305e-01 + 7.384733e-01j	-6.722299e-01 + 7.384739e-01j

a microstrip patch antenna array having two elements. In recent years, various antenna configurations have been explored to realize on-package wireless communication as an alternative solution to the wire-based interconnects. One configuration is shown in Fig. 20, where two circular conducting patches, each of which having a radius of 0.5 mm and metal thickness of 0.015 mm, are etched on a dielectric substrate of relative permittivity 3.4. The thickness of the substrate is 0.03 mm. The substrate is backed by a 0.015 mm-thick ground plane made of copper. The conductivity of the copper used is $5.8\mathrm{e}+7$ S/m. The metallic circular patch is made of the same type of copper. In Fig. 20, D is 1.2 mm; $W = H = 0.1$ mm. Above the two patches is a dielectric layer of thickness 0.65 mm and relative permittivity 3.4.

Two solvers are used to simulate this microstrip patch antenna example. One is the proposed layered $\mathcal{H}$-LU algorithm, the other is the conventional $\mathcal{H}$-LU algorithm. The frequency simulated is 100 GHz. We excite one patch and assess its radiation to the adjacent patch by extracting 2-port S-parameters. In Table II, we list the S-parameters obtained from both solvers, excellent agreement can be observed. The proposed layered $\mathcal{H}$-LU algorithm only cost 132.79 s in CPU time and 49 MB memory to finish the simulation while the conventional $\mathcal{H}$-LU solver takes 351.46 s and 1.3 GB memory. The leafsize used is 32, the η is chosen as 1, and the relative error used for adaptive truncation in $\mathcal{H}$-based computation is set as 1e-8.

VIII. CONCLUSION

In this work, we develop both layered $\mathcal{H}$ -inverse and layered $\mathcal{H}$-LU based direct FEM solvers for 3-D electromagnetic analysis. Comparing with existing $\mathcal{H}$-matrix-based direct solvers, the storage cost of the proposed layered solvers is reduced from $\mathrm{O}(k_{3\mathrm{D}}N\log N)$ to $\mathrm{O}(k_{2\mathrm{D}}M\log M)$ and the time cost is reduced from $\mathrm{O}(k_{3\mathrm{D}}^2N\log^2 N)$ to $\mathrm{O}(k_{2\mathrm{D}}^2N\log^2 M)$. Since the number of single-layer unknowns M is orders of magnitude smaller than N, the proposed $\mathcal{H}$-matrix-based

layered direct solver is much faster than the existing $\mathcal{H}$-matrix-based direct solvers. Moreover, the rank's growth rate with electric size for electrodynamic analysis is also reduced from a 3-D based growth rate to a 2-D based growth rate. If the structure is periodic, the time cost is further reduced to $\log_2 pO(k_{2\mathrm{D}}^2 M \log^2 M)$, where p is the number of periods.

For a general 3-D problem having a similar dimension along each of the x-, y-, and z-directions, M is proportional to $N^{2/3}$, and $k_{2\mathrm{D}}$ is in the order of $\sqrt{\log(N^{1/3})}$. As a result, the time complexity of the proposed direct finite element solver is $\mathrm{O}(N \log^3 N)$ in inverse and LU factorization, while the storage complexity is less than $\mathrm{O}(N)$ for obtaining the solution in any 2-D domain of interest. Numerical experiments and a comparison with state-of-the-art sparse matrix solvers have demonstrated the validity and the performance of the proposed direct FEM solvers.

REFERENCES

[1] A. George, "Nested dissection of a regular finite element mesh," *SIAM J. Numer. Analy.*, vol. 10, no. 2, pp. 345–363, Apr. 1973.

[2] J. Choi, R. J. Adams, and F. X. Canning, "Sparse factorization of finite element matrices using overlapped localizing solution modes," *Microw. Opt. Technol. Lett.*, vol. 50, no. 4, pp. 1050–1054, 2008.

[3] H. Liu and D. Jiao, "A direct finite-element-based solver of significantly reduced complexity for solving large-scale electromagnetic problems," presented at the IEEE Int. Microwave Symp. (IMS), Jun. 2009.

[4] H. Liu and D. Jiao, "Performance analysis of the $\mathcal{H}$-matrix-based fast direct solver for finite-element-based analysis of electromagnetic problems," presented at the IEEE Int. Symp. on Antennas and Propagation, 2009.

[5] H. Liu and D. Jiao, "Existence of $\mathcal{H}$-matrix representations of the inverse finite-element matrix of electrodynamic problems and $\mathcal{H}$-based fast direct finite-element solvers," *IEEE Trans. Microw. Theory Tech.*, vol. 58, no. 12, pp. 3697–3709, Dec. 2010.

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

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

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

[9] L. Grasedyck and W. Hackbusch, "Construction and arithmetics of $\mathcal{H}$-matrices," *Computing*, vol. 70, no. 4, pp. 295–344, Aug. 2003.

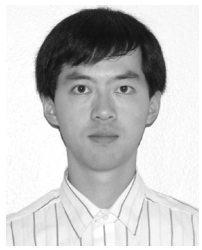
[10] H. Liu and D. Jiao, "A theoretical study on the Rank's dependence with electric size of the inverse finite element matrix for large-scale electrodynamic analysis," presented at the IEEE Int. Symp. on Antennas and Propagation, Jul. 2012.

[11] H. Liu and D. Jiao, "A theoretical study on the rank's dependence with electric size of the inverse finite element matrix for large-scale electrodynamic analysis," Nov. 2011, TR-ECE-11-20 School of Electrical and Computer Engineering, Purdue University [Online]. Available: http://docs.lib.purdue.edu/ecetr/425

[12] D. Jiao, S. Chakravarty, and C. Dai, "A layered finite-element method for electromagnetic analysis of large-scale high-frequency integrated circuits," *IEEE Trans. Antennas Propag.*, vol. 55, no. 2, pp. 422–432, Feb. 2007.

[13] UMFPACK5.0 [Online]. Available: http://www.cise.ufl.edu/research/sparse/umfpack/

[14] J. M. Jin, *The Finite Element Method in Electromagnetics*, 2nd ed. New York: Wiley, 2002.



Haixin Liu received the B.S. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2006 and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 2012.

From 2008 to 2012, he worked in the On-Chip Electromagnetics Group, Purdue University, as a Research Assistant. He was a Co-op Engineer with Global Foundries Inc. in 2009. In 2012, he joined Oracle Corporation, CA, as a Senior Engineer. His research interests include computational electromagnetics, high-performance VLSI CAD, fast numerical methods for very large scale IC and package problems.

Dr. Liu was a recipient of the Best Student Paper Award at the International Annual Review of Progress in Applied Computational Electromagnetics in 2011. He serves as a reviewer for the Applied Computational Electromagnetics Society journal and conference.

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

She then worked at the Technology Computer-Aided Design (CAD) Division, Intel Corporation, until September 2005, as a Senior CAD Engineer, Staff Engineer, and Senior Staff Engineer. In September 2005, she joined Purdue University, West Lafayette, IN, USA, as an Assistant Professor with the School of Electrical and Computer Engineering, where she is now a tenured Associate Professor. She has authored two book chapters and over 170 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 bio-electromagnetics.

Dr. Jiao was among the 85 engineers selected throughout the nation for the National Academy of Engineering's 2011 US Frontiers of Engineering Symposium. She was the 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 Start up Award (which recognizes an outstanding new faculty member of the School of Electrical and Computer Engineering, Purdue University), a 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 (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 in recognition of her work on the industry-leading BroadSpice modeling/simulation capability for designing high-speed microprocessors, packages, and circuit boards, the Intel Corporation's Technology CAD Divisional Achievement Award for the development of innovative full-wave solvers for high frequency IC design, the 2002 Intel Corporation's Components Research the Intel Hero Award (Intel-wide she was the tenth recipient) for the timely and accurate 2-D and 3-D full-wave simulations, the Intel Corporation's LTD Team Quality Award for her outstanding contribution to the development of the measurement capability and simulation tools for high frequency on-chip crosstalk, 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. She is an Associate Editor of the IEEE TRANSACTIONS ON COMPONENTS, PACKAGING, AND MANUFACTURING TECHNOLOGY.