Direct Finite-Element Solver of Linear Complexity for Large-Scale 3-D Electromagnetic Analysis and Circuit Extraction

Bangda Zhou and Dan Jiao, *Senior Member, IEEE*

***Abstract*—In this paper, we develop a linear-complexity direct finite-element solver for the electromagnetic analysis of general 3-D problems containing arbitrarily shaped lossy or lossless conductors in inhomogeneous materials. Both theoretical analysis and numerical experiments have demonstrated the solver's linear complexity in CPU time and memory consumption with prescribed accuracy satisfied. The proposed direct solver has successfully analyzed an industry product-level full package involving over 22.8488 million unknowns in approximately 16 h on a single core running at 3 GHz. It has also rapidly solved large-scale antenna arrays of over 73 wavelengths with 3600 antenna elements whose number of unknowns is over 10 million. The proposed direct solver has been compared with the finite-element methods that utilize the most advanced direct sparse solvers and a widely used commercial iterative finite-element solver. Clear advantages of the proposed solver in time and memory complexity, as well as computational efficiency, have been demonstrated.**

***Index Terms*—Circuit analysis, direct solvers, electromagnetic analysis, fast solvers, finite-element methods (FEMs), frequency domain, linear-complexity solvers, 3-D structures.**

I. Introduction

AMONG existing computational electromagnetic methods, the finite-element method (FEM) has been a popular choice for analyzing electromagnetic problems that involve both irregular geometries and complicated materials. The system matrix resulting from an FEM-based analysis is sparse; however, its LU factors and inverse are, in general, dense. Hence, it can be computationally challenging to solve a large-scale sparse matrix.

The multifrontal method [1] is one of the most important direct sparse solvers. Well-known sparse solver packages such as `UMFPACK` [2], `MUMPS` [3], and `Pardiso` in Intel Math Kernel Library (MKL) [4] are all based on the multifrontal method. The complexity of a multifrontal solver is dependent on the ordering used to reduce fill-ins. For 2-D FEM problems, the nested dissection ordering [5] results in a direct solution of $O(N^{1.5})$ complexity in CPU time, where N denotes the matrix size, which is also the number of degrees of freedom. This complexity is optimal in a regular grid for any ordering in exact arithmetic. For 3-D problems, a nested-dissection-based multifrontal factorization has $O(N^2)$ complexity in time and $O(N^{4/3})$ complexity in memory [5], neither of which is optimal. It has been shown that for structured grids with point or edge singularities, the computational cost of a multifrontal solver may reduce to $O(N)$ in exact arithmetic [6], [7]. In [8], a direct solution is presented for 2-D FEM-based electromagnetic analysis, the complexity of which is higher than linear. Recently, it has been proven in [9] that the sparse matrix arising from an FEM-based analysis of general 2-D or 3-D electromagnetic problems has an exact $\mathcal{H}$-matrix representation, and the inverse of this sparse matrix has an error-controlled $\mathcal{H}$-matrix representation. Based on this proof, an $\mathcal{H}$-matrix based fast direct finite-element solver is developed in [9]. The storage and time complexity of this solver are, respectively, $O(NlogN)$, and $O(N\log^2 N)$ for solving 3-D problems whose $\mathcal{H}$-matrix representations have a constant rank. A superfast multifrontal solver with hierarchically semiseparable representations has also been developed in [10]. However, as yet, no $O(N)$ (optimal) complexity direct matrix solution has been achieved for the FEM-based analysis of general 3-D problems. This is also true to a recent direct solver reported in [11]. There also exists extensive research on parallelizing direct sparse factorizations [12]. It is known that parallelization can reduce CPU run time, but it cannot reduce the computational complexity of the algorithm being parallelized.

Prevailing fast FEM-based solvers for solving large-scale problems are iterative solvers. Their best computational complexity is $O(N_{\text{it}}N_{\text{rhs}}N)$, where N_{it} is the number of iterations and N_{rhs} is the number of right-hand sides. The N_{it} is, in general, problem dependent. In addition, most of the iterative solvers rely on preconditioners to reduce N_{it}, which further increases the overall computational complexity.

In this paper, we develop a direct FEM solver of $O(N)$ (linear) complexity for general 3-D electromagnetic analysis. Its linear complexity in CPU time and memory consumption is demonstrated by theoretical analysis and numerical experiments. The proposed direct solver has successfully solved a product-level full package in inhomogeneous dielectrics having over 22.8488 million unknowns in 16 h, on a single core running at 3 GHz. Its accuracy is fully validated by both industry

Manuscript received September 20, 2014; revised December 24, 2014, April 28, 2015, July 07, 2015, and July 28, 2015; accepted August 16, 2015. Date of publication September 11, 2015; date of current version October 02, 2015. This work was supported by the National Science Foundation (NSF) under Grant 0802178 and Grant 1065318, by the Scientific Research Council (SRC) under Grant Task 1292.073, and by the Defense Advanced Research Projects Agency (DARPA) under Award N00014-10-1-0482B.

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.2015.2472003

measurements and numerical experiments. The proposed direct solver has also been compared with state-of-the-art direct sparse matrix solvers, as well as a widely used commercial iterative FEM solver. Clear advantages in computational complexity and efficiency have been demonstrated.

This paper is a significantly extended version of our conference papers [13], [14]. In [13], we developed a direct finite-element solver of linear complexity for large-scale 3-D circuit extraction in multiple dielectrics. This work was then extended to electrically large analysis in [14] by taking into account the rank's growth with electrical size in electrodynamic analysis. Neither of the solvers we reported in [13] and [14], as well as the recent application to signal and power integrity analysis in [15] and antenna analysis in [16], has yet solved the realistic industry full package simulated in this paper. Furthermore, although the comparison with `UMFPACK` shows advantages of the direct solvers in [13] and [14], we have not made comparisons with other more advanced direct sparse solvers such as `MUMPS` [3], and `Pardiso` in Intel MKL [4], which outperform `UMFPACK` in CPU time and memory consumption. In this paper, we present the detailed algorithms of an $O(N)$ direct FEM solver that outperforms `MUMPS` and `Pardiso` in computational complexity and efficiency. This solver is also capable of analyzing both large-scale electromagnetic structures such as antennas, and large-scale integrated circuits such as an industry product-level full package.

This paper is organized as follows. In Section II, we give a brief review of the vector finite-element-based analysis of electromagnetic problems, and the state-of-the-art sparse solver technologies. In Section III, we elaborate the detailed algorithm of the proposed linear-complexity direct finite-element solver. In Section IV, we present a theoretical analysis on the accuracy and complexity of the proposed direct solver. In Section V, the choice of simulation parameters is discussed. In Section VI, numerical and experimental results are presented to demonstrate the linear complexity and the superior performance of the proposed direct solver. Section VII relates to our conclusions. Throughout this paper, for quantities involved in a matrix equation, we use a boldface letter to denote a matrix, and an italicized one for a vector. A symbol of $\mathbf{A}_{i,j}$ denotes the entry at the ith row and jth column of matrix $\mathbf{A}$.

II. Preliminaries

A. Vector Finite-Element-Based Electromagnetic Analysis

The equation solved in this work is the second-order vector wave equation. A finite-element-based solution of this equation subject to boundary conditions results in the following matrix equation:

$$\mathbf{Y}\{u\} = \{b\} \tag{1}$$

where b denotes an excitation vector, u is the unknown field vector, and $\mathbf{Y}$ is the system matrix. Different from the system matrix resulting from the solution of Poisson's equation, $\mathbf{Y}$ is indefinite here, and has complex-valued eigenvalues when the dielectrics or conductors are lossy. $\mathbf{Y}$ can further be written as

$$\mathbf{Y} = -k_0^2\mathbf{T} + jk_0\mathbf{G} + \mathbf{S} \tag{2}$$

where k_0 is free-space wavenumber, $\mathbf{T}$, $\mathbf{G}$, and $\mathbf{S}$ are sparse matrices assembled from

$$\begin{aligned} \mathbf{T}_{ij}^e &= \iiint_v \varepsilon_r \mathbf{N}_i \cdot \mathbf{N}_j \mathrm{d}v \\ \mathbf{S}_{ij}^e &= \iiint_v \frac{1}{\mu_r}(\nabla \times \mathbf{N}_i)\cdot(\nabla \times \mathbf{N}_j)\mathrm{d}v \\ \mathbf{G}_{ij}^e &= \iiint_v \eta_0\sigma \mathbf{N}_i \cdot \mathbf{N}_j \mathrm{d}v + \iint_{s_o} (\hat{n}\times\mathbf{N}_i)\cdot(\hat{n}\times\mathbf{N}_j)\mathrm{d}s \end{aligned} \tag{3}$$

where μ_r, ε_r, and σ, respectively, denotes relative permeability, dielectric constant, and conductivity, s_o is the outermost boundary of the problem being simulated, $\hat{n}$ is an outward unit normal vector, and $\mathbf{N}$ is the vector basis function used for field expansion.

B. $\mathcal{H}$-Matrix

In an $\mathcal{H}$-matrix [17], the entire matrix is hierarchically partitioned, via a tree structure, into multilevel admissible blocks and inadmissible blocks. An admissible block $\mathbf{Y}^{t,s}$, with row unknowns in set t and column unknowns in s, satisfies the following admissibility condition:

$$\min\{\mathrm{diam}(\Omega_t), \mathrm{diam}(\Omega_s)\} < \eta\,\mathrm{dist}(\Omega_t, \Omega_s) \tag{4}$$

where Ω_t denotes the region containing all field unknowns in set t, Ω_s is the area where all unknowns in set s reside, $\mathrm{diam}(\cdot)$ denotes the Euclidean diameter of a set, $\mathrm{dist}(\cdot,\cdot)$ stands for the Euclidean distance between two sets, and η is a parameter greater than zero, which η can be used to control the admissibility condition. An inadmissible block does not satisfy the admissibility condition. It keeps its original full-matrix representation. However, an admissible block is represented as

$$\tilde{\mathbf{Y}}^{t\times s} = \mathbf{A}_{\#t\times k}\mathbf{B}_{\#s\times k}^T \tag{5}$$

where k is the rank, and $\#t(s)$ denotes the number of unknowns in t or s. If we sort the singular values of $\mathbf{Y}^{t\times s}$ from the largest to the smallest as $\sigma_1, \sigma_2, \ldots, \sigma_N$, the relative error of (5), as compared to its full-matrix representation, is [17]

$$\epsilon = \frac{\|\mathbf{Y}^{t\times s} - \tilde{\mathbf{Y}}^{t\times s}\|_2}{\|\mathbf{Y}^{t\times s}\|_2} = \frac{\sigma_{k+1}}{\sigma_1}. \tag{6}$$

Obviously, the accuracy of (5) can be adaptively controlled by choosing rank k based on a prescribed accuracy.

C. On the Rank

As for the rank of an $\mathcal{H}$-matrix representation of an FEM matrix, since the matrix is sparse, the rank of admissible blocks is zero in the FEM matrix. However, the key for a fast direct solution does not lie in the original matrix, but in its inverse or LU factors. The rank of the inverse or LU factors of an electromagnetic problem has been studied in [18] and [19]. It is shown that for electrically small problems, a constant rank is sufficient in representing the inverse and LU factors of the FEM matrix to achieve a desired order of accuracy irrespective of problem size. For electrically large problems, given an error bound, the rank of the inverse finite-element matrix is a constant irrespective of electrical size for 1-D problems. For 2-D problems, it grows very slowly as the square root of the logarithm of the electrical size. For 3-D problems, the rank scales linearly with the electrical size. Notice that this rank's growth rate is the growth rate of a minimal rank representation [such as the one generated by singular value decomposition (SVD)] that does not separate sources from observers in characterizing the interaction between the two. In contrast, representations that separate sources from observers do not result in a minimal-rank representation required by accuracy, and the resultant rank is a full rank asymptotically.

III. Proposed Linear-Complexity Direct 3-D Finite-Element Solver

In the proposed method, we do not blindly treat the entire FEM matrix and its $\mathbf{L}/\mathbf{U}$ factors as a whole $\mathcal{H}$-matrix. If we do so, the sparse linear algebra cannot be taken advantage of. Instead, we establish an elimination tree based on the nested-dissection ordering such that the zeros can be maximized in the $\mathbf{L}/\mathbf{U}$ factors. We then perform a multifrontal-based factorization of the nodes in the elimination tree from the bottom nodes to the root node level by level. By doing so, we only need to store and compute nonzeros in the $\mathbf{L}/\mathbf{U}$ factors without wasting computation and storage on zeros. The intermediate frontal and update matrices generated during the factorization procedure are all dense matrices. If using traditional full-matrix representations, the resulting complexity would be much higher than linear, like the complexity of existing sparse solvers. Instead of doing so, we form the intermediate dense matrices compactly by error-controlled $\mathcal{H}$-matrix representations, and use these representations to do fast computations. Therefore, there are two tree structures in the proposed algorithm. One is the elimination tree, the other is the local $\mathcal{H}$-tree created for each node in the elimination tree to represent and compute the intermediate dense matrices associated with the node. Since the $\mathcal{H}$-tree structure is locally created for each node in the elimination tree, all the additions and multiplications become unmatched operations, unlike those in the traditional $\mathcal{H}$-matrix arithmetic. Hence, new $\mathcal{H}$-matrix arithmetic is developed in this work to handle unmatched operations. Moreover, the aforementioned procedure organizes the factorization of the original 3-D finite-element matrix into a sequence of partial factorizations of $\mathcal{H}$-matrices corresponding to 2-D separators. Hence, the rank of the intermediate matrices follows a 2-D based growth rate with electrical size, which is much slower than a 3-D based growth rate [18], facilitating electrically large analyses.

There are six major steps in the proposed direct solver, as shown in Algorithm 1. In the next few sections, we will explain each step one by one.

A. Partition Unknowns by Nested Dissection

With the nested dissection scheme, we recursively divide a 3-D computational domain into two subdomains and one separator, obtaining $d_i^0 = \{d_{i+1}^0, d_{i+1}^1, s_{i+1}^0\}$, as illustrated in Figs. 1 and 2. Here, d_i^j denotes a domain at level i with index j, s_i^j represents a separator at level i with index j. Since separator s_{i+1}^0 completely separates domains d_{i+1}^0 and d_{i+1}^1, the matrix blocks $\mathbf{Y}_{d_{i+1}^0 d_{i+1}^1}$ and $\mathbf{Y}_{d_{i+1}^1 d_{i+1}^0}$ in the FEM system matrix are zero blocks. These zero blocks will also be preserved during the LU factorization process, hence, reducing the total number of operations.

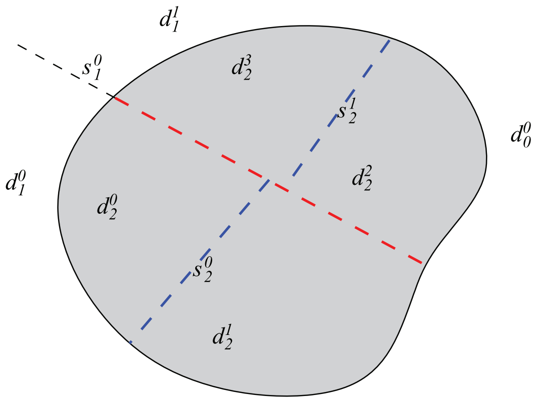


Fig. 1. Example of a level-2 nested dissection on domain d_0^0.

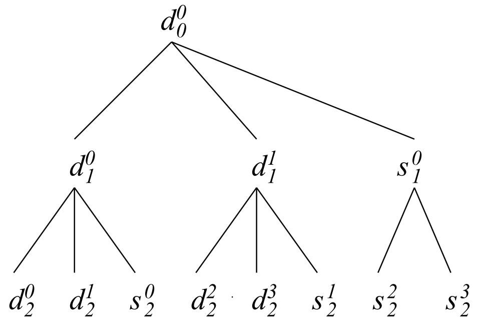


Fig. 2. Illustration of the unknown partition by a level-2 nested dissection on domain d_0^0.

Algorithm 1: Proposed Direct Solver

```
1 Partition unknowns by nested dissection.
2 Build elimination tree $\mathcal{E}_\mathcal{I}$ from nested dissection
  ordering.
3 Do symbolic factorization by elimination tree $\mathcal{E}_\mathcal{I}$.
4 Generate a local $\mathcal{H}$-matrix representation for the frontal
  matrix associated with each node in the elimination
  tree.
5 Do numerical factorization across elimination tree $\mathcal{E}_\mathcal{I}$
  by developing fast $\mathcal{H}$-matrix-based algorithms.
6 Compute the solution and do post-processing.
```

In order to generate large zero blocks, the size of the separator should be as small as possible and the size of two domains being separated should be as equal as possible. These two criteria guide the unknown partition based on the nested dissection. We also recursively divide the separator into subdomains by using geometrical bisection or nested dissection, as shown in Fig. 2. The recursive process of the nested-dissection partition continues until the number of unknowns in each domain is no greater than a pre-defined constant parameter *leafsize*, denoted by $n_{\min}$. The resultant tree shown in Fig. 2 is denoted by $\mathcal{T}_\mathcal{I}$, in which $\mathcal{I}$ represents the index set of all unknowns.

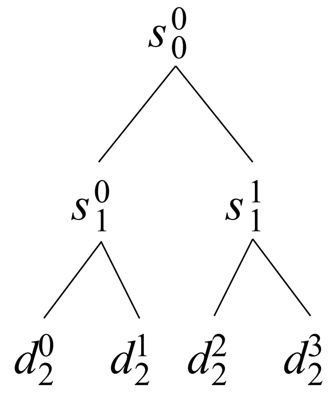


Fig. 3. Elimination tree $\mathcal{E}_{\mathcal{I}}$ of level-2 nested dissection on domain d_0^0.

B. *Construction of Elimination Tree $\mathcal{E}_{\mathcal{I}}$ From Nested Dissection Ordering*

An elimination tree, denoted by $\mathcal{E}_{\mathcal{I}}$, is used in most sparse matrix solvers [2], [20], [3], [4]. It provides a sequence of factorization or elimination. An elimination tree is defined as a structure with n nodes $\{t_0, t_1, \ldots, t_{n-1}\}$ such that node t_p is the parent node of t_j if and only if the following is satisfied:

$$p = \min\{i > j | \mathbf{L}_{t_i t_j} \neq 0\} \tag{7}$$

in which $\mathbf{L}_{t_i t_j}$ is the block in the $\mathbf{L}$ factor of the original system matrix $\mathbf{Y}$, whose row unknowns are contained in node t_i and column unknowns in node t_j.

Consider the example shown in Fig. 2, where domain d_1^0 and domain d_1^1 are completely separated by separator s_1^0. It is clear that the matrix blocks corresponding to the two domains d_1^0 and d_1^1 are completely decoupled, and hence, can be factorized independent of each other. However, they both make contributions to separator s_1^0 during the factorization process. Thus, based on (7), it can be readily verified that separator s_1^0 is the parent node of nodes d_1^0 and d_1^1 in the elimination tree. Applying this rule recursively, we can find that in the final elimination tree built for the entire unknown set, the bottom level is occupied by the domains of *leafsize*, and the upper levels are occupied by the separators of increasing size, with the topmost level being the separator of the largest size. The overall algorithm for building the elimination tree is given in Algorithm 2. To give an example, the elimination tree corresponding to Fig. 2 is illustrated in Fig. 3, where the subscripts of separator nodes have been updated to the tree level of the elimination tree. Thus, s_0^0, s_1^0, s_1^1 in Fig. 3 correspond to s_1^0, s_2^0, and s_2^1 in Fig. 1, respectively.

Different from the cluster tree used in an $\mathcal{H}$-matrix representation, an elimination tree $\mathcal{E}_{\mathcal{I}}$ satisfies the following property:

$$\bigcup_{i,j,t_i^j \in \mathcal{E}_{\mathcal{I}}} t_i^j = \mathcal{I} \tag{8}$$

where t_i^j denotes a node in the elimination tree at level i with index j. This means the nodes in the entire elimination tree constitute a disjoint partition of the entire unknown set $\mathcal{I}$. The unknowns in the children nodes are *not* contained in the parent node. In contrast, in a cluster tree, the children nodes are subsets of the parent node, and the nodes at each *tree level* form a disjoint partition of the complete unknown set. The LU factorization process is nothing but a post-order or bottom-up tree traversal of the elimination tree $\mathcal{E}_{\mathcal{I}}$. After all the nodes in the elimination tree have been factorized, the entire LU factorization is completed.

C. *Symbolic Factorization*

Due to the sparse nature of the partial differential operator, only a subset of unknowns is affected by eliminating one node in the elimination tree. In this paper, this subset is termed the *boundary cluster* of one node in the elimination tree.

Algorithm 2: Build Elimination Tree

```
1 BuildEliminationTree ($d_i^j$)
  Data: domain cluster $d_i^j \in \mathcal{T}_{\mathcal{I}}$
  Result: a cluster $t_i^j \in \mathcal{E}_{\mathcal{I}}$
2 if children ($d_i^j$) $\neq 0$ then
3     find $l$ such that $parent(d_i^j) = parent(s_i^l)$
4     $t_i^j \leftarrow s_i^l$
5     for each $k$ that $d_{i+1}^k \in children(d_i^j)$ do
6         $t_{i+1}^k \leftarrow$ BuildEliminationTree ($d_{i+1}^k$)
7 else
8     $t_i^j \leftarrow d_i^j$
9 return $t_i^j$
```

For each node t_i^j in the elimination tree, we construct a frontal matrix $\mathbf{F}_i^j$, as shown below, where b_i^j denotes the *boundary cluster* of node t_i^j,

$$\mathbf{F}_i^j = \begin{array}{c c} & \begin{array}{cc} t_i^j & b_i^j \end{array} \\ \begin{array}{c} t_i^j \\ b_i^j \end{array} & \begin{pmatrix} \tilde{\mathbf{Y}}_{t_i^j t_i^j} & \tilde{\mathbf{Y}}_{t_i^j b_i^j} \\ \tilde{\mathbf{Y}}_{b_i^j t_i^j} & \hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j} \end{pmatrix} \end{array}. \tag{9}$$

If we use $\mathrm{ans}(t_i^j)$ to represent the ancestor of node t_i^j in elimination tree $\mathcal{E}_{\mathcal{I}}$, the following is satisfied from the definition of elimination tree:

$$b_i^j \subseteq \mathrm{ans}(t_i^j). \tag{10}$$

To identify boundary unknowns contained in b_i^j, there are two approaches: algebraic approach and geometrical approach. Geometrically, boundary unknowns are the unknowns residing on the bounding box of a node (a 2-D separator or a 3-D $leafsize$ domain), which can be determined by the mesh information. The geometrical approach for finding boundary unknowns could be done along with the nested dissection based unknown partition.

In the proposed solver, we have also implemented an algebraic approach, known as *symbolic factorization*, in order to precisely determine the boundary unknowns. Since it is not efficient to use one unknown as a node to perform the symbolic factorization, we use the domains of *leafsize* as *supernodes*. These *leafsize*-domains reside on the bottom level of the elimination tree. The detailed procedure is given in Algorithm 3. As can be seen, it starts with an initialization of all b_i^j's from the original matrix $\mathbf{Y}$. The nonzero sparse pattern of $\mathbf{Y}$ is then used to fill b_i^j, which is composed of supernodes. If $\mathbf{Y}_{t_i^j c}$ is nonzero, then supernode c is added to b_i^j. After symbolic factorization, boundary unknowns contained in b_i^j for every t_i^j in elimination tree $\mathcal{E_I}$ can be correctly and precisely determined.

Algorithm 3: Symbolic Factorization

```
1  Initialize all $b_i^j$ from $\mathbf{Y}$
2  for $i = L \to 0, L$ is the tree depth of $\mathcal{E_I}$ do
3      for each $t_i^j \in \mathcal{E_I}$ do
4          for each $p \in b_i^j$ do
5              $k \leftarrow$ level of $p$
6              $m \leftarrow$ index of $p$
7              for each $p \neq q \in b_i^j$ do
8                  $b_k^m += p$
```

D. $\mathcal{H}$-Matrix Representations of All Frontal Matrices $\mathbf{F}_i^j$

Algorithm 4: Construction of $\mathcal{H}$-matrix Representations of All $\mathbf{F}_i^j \in \mathcal{E_I}$

```
1  for $i = L \to 0, L$ is the tree depth of $\mathcal{E_I}$ do
2      for each $t_i^j \in \mathcal{E_I}$ do
3          Build $\mathcal{H}$-matrix representation of $\tilde{\mathbf{Y}}_{t_i^j t_i^j}$
4          if $i \neq 0$ then
5              Build $\mathcal{H}$-matrix representation of $\tilde{\mathbf{Y}}_{t_i^j b_i^j}$
6  for $i = L \to 0, L$ is the tree depth of $\mathcal{E_I}$ do
7      for each $t_i^j \in \mathcal{E_I}$ do
8          Fill $\tilde{\mathbf{Y}}_{t_i^j t_i^j}$ with sparse matrix from FEM
9          if $i \neq 0$ then
10             Fill $\tilde{\mathbf{Y}}_{t_i^j b_i^j}$ with sparse matrix from FEM
11 for $i = L \to 1, L$ is the tree depth of $\mathcal{E_I}$ do
12     for each $t_i^j \in \mathcal{E_I}$ do
13         $\tilde{\mathbf{Y}}_{b_i^j t_i^j} \leftarrow \tilde{\mathbf{Y}}'_{t_i^j b_i^j}$
14 for $i = L \to 1, l$ is the tree depth of $\mathcal{E_I}$ do
15     for each $t_i^j \in \mathcal{E_I}$ do
16         Link $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$
```

After symbolic factorization, the row cluster $\{t_i^j, b_i^j\}$, as well as the column cluster of each frontal matrix $\mathbf{F}_i^j$, shown in (9), are determined for node t_i^j in elimination tree $\mathcal{E_I}$. In each frontal matrix $\mathbf{F}_i^j$, there are four blocks, $\tilde{\mathbf{Y}}_{t_i^j t_i^j}, \tilde{\mathbf{Y}}_{t_i^j b_i^j}, \tilde{\mathbf{Y}}_{b_i^j t_i^j}$, and $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$. The first three can be generated as local matrix blocks, which are associated with node t_i^j only since they are not needed in upper level computations. However, the last one, $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$, which is known as the *update matrix* [3], has to communicate with the nodes at the upper levels of the elimination tree.

Different from traditional $\mathcal{H}$-matrix based computations, where the $\mathcal{H}$-matrix structures to be operated on are compatible, here, the structure of update matrix $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$ is not compatible with either local blocks $\{\tilde{\mathbf{Y}}_{t_i^j b_i^j}, \tilde{\mathbf{Y}}_{b_i^j t_i^j}\}$ or frontal matrices residing on the upper levels of the elimination tree. If we make the $\mathcal{H}$-matrix structure of the update matrix $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$ compatible with that of the local blocks $\{\tilde{\mathbf{Y}}_{t_i^j b_i^j}, \tilde{\mathbf{Y}}_{b_i^j t_i^j}\}$, then we have to perform non-conformal additions of the update matrix upon corresponding upper-level blocks, which is not computationally efficient. Hence, we choose not to make the $\mathcal{H}$-matrix structure of the update matrix compatible with that of local matrices $\{\tilde{\mathbf{Y}}_{t_i^j b_i^j}, \tilde{\mathbf{Y}}_{b_i^j t_i^j}\}$. Instead, we form the update matrix $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$ by grouping links to corresponding blocks in the upper level frontal matrices. Thus, $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$ is composed of multiple pointers that link the update matrix generated at node t_i^j to corresponding blocks in the $\mathcal{H}$-matrix representations of the nodes where b_i^j belongs.

Due to the symmetry of the FEM matrix, the frontal matrix $\mathbf{F}_i^j$ is also symmetric and so is its $\mathcal{H}$-matrix representation. Hence, we only build the $\mathcal{H}$-matrix representations of $\tilde{\mathbf{Y}}_{t_i^j t_i^j}$ and $\tilde{\mathbf{Y}}_{t_i^j b_i^j}$ and fill them with sparse matrices from the FEM. $\tilde{\mathbf{Y}}_{b_i^j t_i^j}$ can be constructed by a transpose copy of $\tilde{\mathbf{Y}}_{t_i^j b_i^j}$. The overall process is detailed in Algorithm 4.

Algorithm 5: Build $\mathcal{H}$-matrix Representation of $\tilde{\mathbf{Y}}_{pq}$

```
1  $\mathcal{H}$-Construction $(p, q)$
2  if $\tilde{\mathbf{Y}}_{pq}$ is admissible then
3      Mark $\tilde{\mathbf{Y}}_{pq}$ as admissible
4  else if $\tilde{\mathbf{Y}}_{pq}$ is inadmissible then
5      Mark $\tilde{\mathbf{Y}}_{pq}$ as inadmissible
6  else
7      if $\frac{1}{\alpha} \leq \frac{\#(p)}{\#(q)} \leq \alpha$ then
8          for each child $r \in p$ do
9              for each child $s \in q$ do
10                 $children(r, s)$ of $\tilde{\mathbf{Y}}_{pq} \leftarrow$
                   $\mathcal{H}$-Construction $(r, s)$
11     else
12         if $\#(p) \geq \#(q)$ then
13             for each child $r \in p$ do
14                 $children(r)$ of $\tilde{\mathbf{Y}}_{pq} \leftarrow$
                   $\mathcal{H}$-Construction $(r, q)$
15         else
16             for each child $s \in q$ do
17                 $children(s)$ of $\tilde{\mathbf{Y}}_{pq} \leftarrow$
                   $\mathcal{H}$-Construction $(p, s)$
18 return $\tilde{\mathbf{Y}}_{pq}$
```

To construct the $\mathcal{H}$-matrix structure for each node in the elimination tree, we use the supernodes and devise an adaptive subdivision scheme, which are different from the conventional $\mathcal{H}$-matrix construction method [17]. For each frontal matrix $\mathbf{F}_i^j$, we build a local cluster tree for t_i^j, and b_i^j, respectively, with the leaf size of the cluster tree defined by $h_{\min}$, termed *h-leafsize* to distinguish it from *leafsize*. The *leafsize* is the size of a bottom-level node in the elimination tree, whereas *h-leafsize* is the number of supernodes contained in the leaf node of an $\mathcal{H}$-matrix cluster tree. The local cluster tree is a binary tree, the recursive subdivision of which is performed based on the geometrical bisection along the largest dimension. After the local cluster tree is built for each node in the elimination tree, we build the block cluster tree [17] from two local cluster trees. One is the row tree and the other is the column tree, such as $t_i^j \times t_i^j$ for $\tilde{\mathbf{Y}}_{t_i^j t_i^j}$, and $t_i^j \times b_i^j$ for $\tilde{\mathbf{Y}}_{t_i^j b_i^j}$. In practice, the cardinality of b_i^j is larger than that of t_i^j. If we use the conventional way to generate the block cluster tree, the matrix structure will become very skewed, which is not amenable for fast computation. Hence, we perform an adaptive subdivision. We define a positive constant $\alpha > 1$ to determine the structure of the block cluster tree. If the following condition is satisfied for block $\tilde{\mathbf{Y}}_{pq}$:

$$\frac{1}{\alpha} \leq \frac{\#(p)}{\#(q)} \leq \alpha \tag{11}$$

block $\tilde{\mathbf{Y}}_{pq}$ is a *non-skewed matrix*. Otherwise, $\tilde{\mathbf{Y}}_{pq}$ is a *skewed matrix*. If a matrix is a skewed matrix, during the construction of the block cluster tree, the division will be performed on the cluster with a larger size, as shown in Algorithm 5. Following Algorithms 4 and 5, we can build the $\mathcal{H}$-matrix structure for every frontal matrix $\mathbf{F}_i^j$ in the elimination tree $\mathcal{E}_{\mathcal{I}}$.

Notice that the frontal matrices are not first computed as dense matrices, and then compressed into $\mathcal{H}$-matrices. Instead, they are stored and computed as $\mathcal{H}$-matrices directly. The initial filling of the $\mathcal{H}$-based frontal matrices is straightforward since all admissible blocks are zero, and only inadmissible blocks are filled in their exact form. During the level-by-level factorization procedure, the contents of the $\mathcal{H}$-matrix based frontal matrices are updated from $\mathcal{H}$-arithmetic-based fast multiplications and additions instead of from full-matrix-based computations. Therefore, the frontal matrices in the proposed algorithm are never first computed as dense matrices and then compressed into $\mathcal{H}$-matrices. If so, the complexity of the proposed direct solver cannot be linear.

E. Numerical Factorization and Solution by Fast $\mathcal{H}$-Matrix Algorithms Developed for the Proposed Direct Solver

In the proposed solver, numerical factorization is done by partial factorization of each frontal matrix $\mathbf{F}_i^j$ in the elimination tree $\mathcal{E}_{\mathcal{I}}$ in a bottom-up tree traversal procedure. As shown in Algorithm 6, for each frontal matrix $\mathbf{F}_i^j$ in (9), we first compute LU factorization of $\tilde{\mathbf{Y}}_{t_i^j t_i^j}$ (Step 4), then solve $\mathbf{U}_{t_i^j b_i^j}$ in Step 6, and then compute $\mathbf{L}_{b_i^j t_i^j}$ in Step 7. All these three steps can be done by the $\mathcal{H}$-LU algorithm in [17], [9] since the computations at the three steps are local without the need for the communications with higher levels of the elimination tree. With the $\mathcal{H}$-structure known for the node–node interaction $\tilde{\mathbf{Y}}_{t_i^j t_i^j}$ and the node–boundary interaction $\tilde{\mathbf{Y}}_{t_i^j b_i^j}$, as well as its transpose (see Section III-D), Steps 4, 6, and 7 can be readily performed with the $\mathcal{H}$-LU algorithm. The only difference is that the $\mathcal{H}$-LU algorithm in this work is applied to the local $\mathcal{H}$-matrix representation of every node in the elimination tree, while in [9], the algorithm is used for the entire matrix that is represented by a global $\mathcal{H}$-matrix.

Algorithm 6: Numerical Factorization of Y

```
1 for $i = L \to 0, L$ is the tree depth of $\mathcal{E}_{\mathcal{I}}$ do
2     for each $t_i^j \in \mathcal{E}_{\mathcal{I}}$ do
3         Collect frontal matrix $\mathbf{F}_i^j$
4         Compute $\mathbf{L}_{t_i^j t_i^j}$ and $\mathbf{U}_{t_i^j t_i^j}$ by $\mathcal{H}$-matrix based LU
          factorization, $\tilde{\mathbf{Y}}_{t_i^j t_i^j} = \mathbf{L}_{t_i^j t_i^j} \cdot \mathbf{U}_{t_i^j t_i^j}$
5         if $i \neq 0$ then
6             Compute $\mathbf{U}_{t_i^j b_i^j}$ by solving
              $\mathbf{L}_{t_i^j t_i^j} \cdot \mathbf{U}_{t_i^j b_i^j} = \tilde{\mathbf{Y}}_{t_i^j b_i^j}$
7             Compute $\mathbf{L}_{b_i^j t_i^j}$ by solving
              $\mathbf{L}_{b_i^j t_i^j} \cdot \mathbf{U}_{t_i^j t_i^j} = \tilde{\mathbf{Y}}_{b_i^j t_i^j}$
8             Update $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$ by
              $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j} = \hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j} - \mathbf{L}_{b_i^j t_i^j} \cdot \mathbf{U}_{t_i^j b_i^j}$
```

The last step (Step 8 in Algorithm 6) is performed to generate the update matrix, and also merge it with relevant frontal matrices in the upper levels of the elimination tree. Since for achieving linear complexity we do not generate a global $\mathcal{H}$-matrix structure, but build $\mathcal{H}$-matrix representations of all the intermediate dense matrices, specifically the $\tilde{\mathbf{Y}}_{t_i^j t_i^j}$, $\tilde{\mathbf{Y}}_{t_i^j b_i^j}$, and $\tilde{\mathbf{Y}}_{b_i^j t_i^j}$ blocks of each frontal matrix shown in (9), the updating/merging step is dominant by operations with unmatched dimensions and structures. In contrast, conventional $\mathcal{H}$-matrix arithmetic requires matched $\mathcal{H}$-matrix structures of the operands. We therefore develop new $\mathcal{H}$-matrix arithmetic to handle unmatched cases present in the proposed direct solver. Below, we provide a detailed description of the algorithm.

As mentioned in Section III-D, $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$ is constructed by grouping links to corresponding blocks in the upper level frontal matrices, i.e., pointing them to the right blocks of the $\mathcal{H}$-matrices of the upper-level frontal matrices. Let us denote $\mathbf{C}_{st}$ as the $\mathcal{H}$-matrix of one upper-level frontal matrix where $\hat{\tilde{\mathbf{Y}}}_{b_i^j b_i^j}$ contributes. Since each frontal matrix has a unique local $\mathcal{H}$-matrix structure instead of sharing a global one in common, the row cluster of $\mathbf{C}_{st}$ may not match that of $\mathbf{L}_{b_i^j t_i^j}$ locally built for node t_i^j, and the column cluster of $\mathbf{C}_{st}$ may not match that of $\mathbf{U}_{t_i^j b_i^j}$. The unmatched clusters are clusters that do not share either the same set of unknowns or the same subdivision. This is very different from the traditional $\mathcal{H}$-based addition given in mathematical literature, where the clusters are matched. Hence, we extend the original $\mathcal{H}$-matrix algorithms to handle the matrix operations with unmatched row and column clusters. Basically, we need to perform

$$\mathbf{C}_{st} = \mathbf{C}_{st} \oplus \mathbf{A}_{pq}\mathbf{B}_{uv} \tag{12}$$

where s and t are, respectively, the row cluster and the column cluster of matrix $\mathbf{C}_{st}$ and $s \neq p, q \neq u, t \neq v$. Since the multiplication of $\mathbf{A}_{pq}$ and $\mathbf{B}_{uv}$ contributes to $\mathbf{C}_{st}$, we have $s \cap p = d, q \cap u = e, t \cap v = f$, and d, e, f are shared unknown sets that are non-null. Therefore, to perform an unmatched operation, we first find the shared unknown sets d, e, and f; then perform the matrix operation as follows:

$$\mathbf{C}_{df} = \mathbf{C}_{df} \oplus \mathbf{A}_{de}\mathbf{B}_{ef}. \tag{13}$$

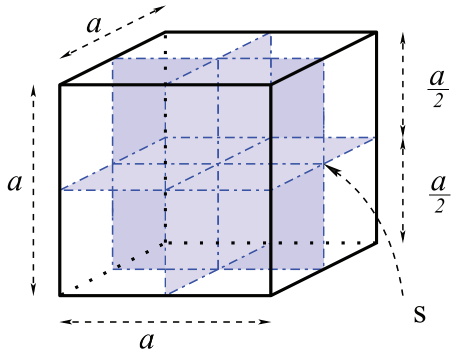
Algorithm 7: Unmatched Updating/Merging Algorithm

```
1 MulAddPro($\mathbf{C}_{st}, \mathbf{A}_{pq}, \mathbf{B}_{uv}$)
2 if $\mathbf{C}_{st}$ is non-leaf then
3     For each child block $\mathbf{S}$ in $\mathbf{C}_{st}$, do
4     MulAddPro($\mathbf{S}, \mathbf{A}_{pq}, \mathbf{B}_{uv}$)
5 else
6     Find the shared unknown sets $d, e$, and $f$
7     Do MulAdd($\mathbf{C}_{df}, \mathbf{A}_{de}, \mathbf{B}_{ef}$)
```

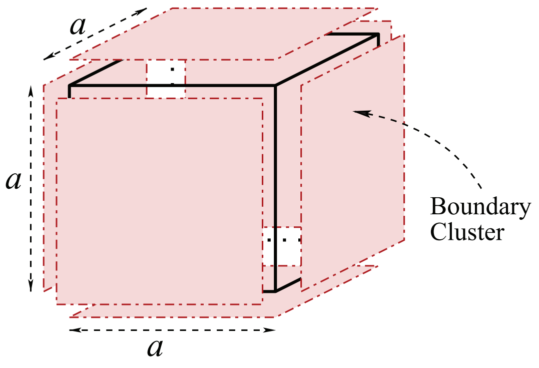
The pseudo-code of the unmatched updating/merging is given in Algorithm 7, where function MulAdd$(\mathbf{C}_{df}, \mathbf{A}_{de}, \mathbf{B}_{ef})$ is to perform the matrix operation with matched row and column clusters. Even though the clusters match each other, the matrices may not share the same $\mathcal{H}$-matrix structure in common. In the proposed algorithm, this is taken into consideration by modifying the basic $\mathcal{H}$-based algorithm of computing $\mathbf{C} = \mathbf{C} + \mathbf{AB}$ to handle the structure difference. The $\mathcal{H}$-matrix-based addition and multiplication are implemented based on supernodes. Furthermore, we develop a *collect* operation to handle unmatched matrix operations related to admissible blocks, as shown in Step 3 of Algorithm 6. Consider matrix operation $\mathbf{C} \mathrel{+}= \mathbf{A}$, if $\mathbf{C}$ is admissible and $\mathbf{A}$ is much smaller than $\mathbf{C}$, then adding them directly using reduced SVD is not efficient. Hence, we store the contribution from matrix $\mathbf{A}$ first without adding it immediately. When $\mathbf{C}$ need to be factorized, all the small contributions will be added together and then added upon $\mathbf{C}$. This addition is performed by collecting children matrix blocks to one block level by level through an accuracy controlled addition. This process continues until we reach the level of $\mathbf{C}$.

The LU solution for one right-hand side can be done as follows, where all multiplications and additions are performed based on the aforementioned new $\mathcal{H}$-matrix algorithms suitable for the proposed direct solution:

```
//forward, $\mathbf{L}y = b$
for each $t_i^j \in \mathcal{E}_\mathcal{I}$, bottom-up
        solve $\mathbf{L}_{t_i^j t_i^j} \cdot b_{t_i^j} = b_{t_i^j}$
        update $b, b_{b_i^j} =$
            $b_{b_i^j} - \tilde{\mathbf{Y}}_{b_i^j t_i^j} \cdot b_{t_i^j}$
end
//backward, $\mathbf{U}x = y$
for each node $t_i^j \in \mathcal{E}_\mathcal{I}$, top-down
    update $b, b_{t_i^j} =$
        $b_{t_i^j} - \tilde{\mathbf{Y}}_{t_i^j b_i^j} \cdot b_{b_i^j}$
    solve $\mathbf{U}_{t_i^j t_i^j} \cdot b_{t_i^j} = b_{t_i^j}$
end                                                   (14)
```



(a)



(b)

Fig. 4. Illustration of three separators and their boundary cluster. (a) Separator. (b) Boundary.

IV. Accuracy and Complexity Analysis

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

A. Accuracy

In the proposed solver, each intermediate update matrix and frontal matrix is represented by an $\mathcal{H}$-matrix. The reason why such an $\mathcal{H}$-representation exists is as follows. The update and frontal matrices are associated with the Schur complement. The Schur complement can be written as $\mathbf{Y}_{22} - \mathbf{Y}_{21}\tilde{\mathbf{Y}}_{11}^{-1}\mathbf{Y}_{12}$, where $\tilde{\mathbf{Y}}_{11}^{-1}$ is the inverse of either an original FEM matrix block or the Schur complement obtained at the previous level. The original FEM matrix has an exact $\mathcal{H}$-representation and its inverse has an $\mathcal{H}$-representation with controlled accuracy [9]. As a result, all the intermediate Schur complements and their inverses as well as LU factors can be represented by $\mathcal{H}$-matrices, the accuracy of which can be controlled by ϵ, as shown in (6).

B. Time and Memory Complexity

Consider a general 3-D problem having $N = n^3$, where n is the number of unknowns along each dimension. By nested dissection, we recursively divide the computational domain into eight subdomains, as shown in Fig. 4(a). The three shaded surface separators, as a group, are denoted by one separator cluster s, and the boundary cluster of s contains all the unknowns residing on the six surfaces that completely enclose the separator cluster s. The depth of the elimination tree is $L = \log_2 n$. With the bottom level of the tree denoted by $l = 0$, in Table I, we list level-by-level the number of unknowns contained in each node

TABLE I
SIZE AND NUMBER OF NODES AT EACH LEVEL OF $\mathcal{E}_{\mathcal{I}}$

level l	$\#(t_i^j)$	num	$\#(b_i^j)$
L	$3\times(2^L)^2$	8^0	$\emptyset$
$L-1$	$3\times(2^{L-1})^2$	8^1	$\sim 6\times(2^{L-1})^2$
$L-2$	$3\times(2^{L-2})^2$	8^2	$\sim 6\times(2^{L-2})^2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$3\times(2^0)^2$	8^L	$6\times(2^0)^2$

of the elimination tree $(\#(t_i^j))$, the number of nodes (num), and the number of unknowns contained in the boundary cluster of the node $(\#(b_i^j))$.

For each node t_i^j in the elimination tree, the dimension of $\mathbf{F}_i^j$ (frontal matrix) is the sum of the cardinality of the node and that of its boundary cluster shown as follows:

$$\dim\left(\mathbf{F}_i^j\right) = \#\left(t_i^j\right) + \#\left(b_i^j\right). \tag{15}$$

It can be seen clearly from Fig. 4(b) that the cardinality of boundary cluster b_i^j is proportional to $6a^2$, while the cardinality of the separator cluster is proportional to $3a^2$, where a is the side length of the domain. Therefore, we have

$$\#\left(b_i^j\right) = c\#\left(t_i^j\right) \tag{16}$$

where c is a constant. Hence, the dimension of frontal matrix $\mathbf{F}_l^j$ at level l is approximately $9\times(2^l\times 2^l)$.

Let m be $\dim(\mathbf{F}_l^j)$, the computational cost associated with each node in the elimination tree, with conventional methods, scales as $O(m^3)$ since the frontal matrix is a dense matrix. However, by the fast $\mathcal{H}$-matrix based arithmetic, the complexity of dealing with the dense frontal matrix is reduced to $O(r_l^2 m\log^2 m)$ in time and $O(r_l m\log m)$ in memory [9], where r_l denotes the rank at the lth level. As a result, for each frontal matrix $\mathbf{F}_l^j$ at level l, the time complexity of LU factorization is

$$\begin{aligned}P_{lu}\left(F_l^j\right) &= O\left(r_l^2 2^l\times 2^l\log^2(2^l\times 2^l)\right)\\ &= O\left(r_l^2 2^{2l}(2l)^2\right).\end{aligned} \tag{17}$$

Each intermediate dense frontal matrix is the Schur complement of the original sparse FEM matrix in a 2-D domain (a surface separator). This matrix is the original FEM matrix in the 2-D domain superposed with the contribution from its children domains after the children domains are eliminated. Since the rank of $(\mathbf{A}+\mathbf{B})^{-1}$ is no greater than the rank of either $\mathbf{A}^{-1}$ or $\mathbf{B}^{-1}$, the rank of each intermediate dense frontal matrix is bounded by the rank of the inverse of the original sparse FEM matrix in a 2-D domain. Hence, the rank obeys the 2-D-based growth rate with electrical size rather than a 3-D-based growth rate. Therefore, the rank r_l is proportional to the square root of the logarithm of the electrical size of the 2-D surface [18], [19], and hence, $r_l = \text{rank}_{2\text{D}} = O(\sqrt{\log 2^l}) = O(\sqrt{l})$. Substituting it into the above, we obtain

$$P_{lu}\left(F_l^j\right) = O(l2^{2l}(2l)^2). \tag{18}$$

As a result, the overall complexity of the LU factorization of the proposed solver is

$$\begin{aligned}P_{lu}(\mathbf{Y}) &= \sum_{l,j,t_l^j\in\mathcal{E}_{\mathcal{I}}} P_{lu}(\mathbf{F}_l^j)\\ &\sim \sum_{l=0}^{L} 8^{(L-l)}\times l2^{2l}(2l)^2\\ &= N\sum_{l=0}^{L}\left(\frac{4l^3}{2^l}\right)\\ &= O(N)\end{aligned} \tag{19}$$

which is linear complexity. The last equality in the above holds true because the denominator grows with l much faster than the numerator. Similarly, we have the complexity of the memory consumption of the proposed solver as the following:

$$\begin{aligned}M_{lu}(\mathbf{Y}) &\sim \sum_{l=0}^{L} 8^{(L-l)}\times l2^{2l}(2l)\\ &= N\sum_{l=0}^{L}\left(\frac{2l^2}{2^l}\right)\\ &= O(N)\end{aligned} \tag{20}$$

which is also linear.

V. Choice of Simulation Parameters

In the proposed solver, there are four simulation parameters: *leafsize*, *h-leafsize*, η, and truncation error ϵ. All these parameters are constant, and hence, regardless of their choices, they do not affect the complexity of the proposed direct solver. However, better choices of these simulation parameters can help reduce absolute run time and memory consumption.

In our implementation, the *leafsize* is usually chosen proportional to the number of basis functions contained in each element. The *h-leafsize* is determined based on the matrix size that can be handled efficiently in a full-matrix format. As for the choice of η and ϵ, we use the following metric. Consider an admissible block $\mathbf{C}\in\mathbb{C}^{m\times n}$. Due to its rank-$k$ representation, the storage cost of $\mathbf{C}$ becomes $(m+n)k$ instead of mn. In order to save memory, we should ensure

$$k < \frac{mn}{m+n}. \tag{21}$$

Therefore, we can define the following ratio to evaluate the memory efficiency of an $\mathcal{H}$-matrix representation:

$$\beta = \frac{mn}{(m+n)k}. \tag{22}$$

We use (22) to adjust η and ϵ so that there are as many admissible blocks as possible with $\beta > 1$. When simulating a suite of structures of increasing size, we usually use the smallest structure to determine the simulation parameters, and use them for all structures in the same simulation suite.

VI. Numerical Results

The proposed direct solver has been used to simulate a variety of complicated 3-D circuits with inhomogeneous lossy conductors and dielectrics from small to over 22 million unknowns on a single core. It has also been used to analyze large-scale antennas. Its computational performance has been benchmarked with existing direct FEM solvers that employ most advanced direct sparse solutions, and also with a widely used commercial iterative FEM solver. In addition to numerical validation, the accuracy of the proposed direct solver has also been validated by full-package measurements provided by a semiconductor company.

A. Cavity-Backed Microstrip Patch Antenna and Antenna Arrays

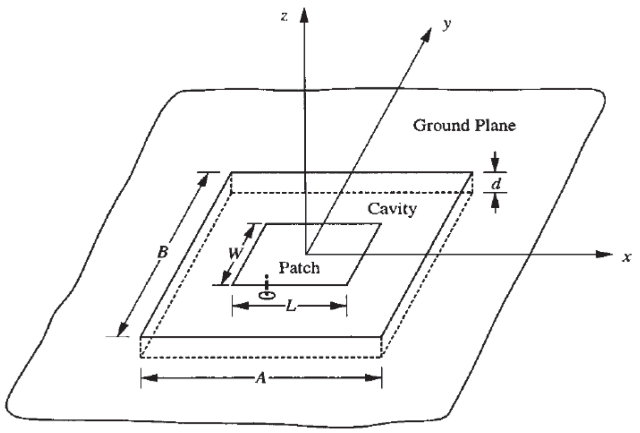
We first validated the accuracy of the proposed solver on a cavity-backed patch antenna [21], [22], whose geometrical parameters are illustrated in Fig. 5(a) and given in the caption. The substrate has a dieletric constant of 2.17 and loss tangent of 0.0015. The top truncation boundary is placed 0.25 cm away from the antenna patch, which is terminated by an open (Neumann) boundary condition. The ground plane is of size 15 cm $\times$ 10 cm, with the center area being the cavity-backed patch antenna. The four sides of the ground plane are truncated by the Neumann boundary condition. The input impedance (both resistance and reactance) of the antenna from 1 to 4 GHz extracted from the proposed method is shown in Fig. 5, which agrees very well with the results generated from the finite-element boundary-integral (FE-BI) method. This result is also shown to agree very well with measured data given in [22].

We then simulated an array of such a patch antenna structure, and also increased the array element number from 2 by 2 to 34 by 34, resulting in 14 449 to 3.47 million unknowns. The frequency is 2.75 GHz. In Fig. 6, we plot the factorization time, memory, and solution error with respect to N for two choices of truncation error, $\epsilon = 1e-4$, and $1e-8$, respectively. The solution error is measured by relative residual $||\mathbf{YX}-\mathbf{B}||/||\mathbf{B}||$, where $\mathbf{B}$ denotes the right-hand-side matrix whose column dimension ranges from 2 to 400. From Fig. 6, it can be seen clearly that the proposed method exhibits linear complexity in both CPU time and memory consumption with good accuracy achieved in the entire unknown range. The smaller error at the early stage is due to the fact that many blocks are full-matrix blocks when the unknown number is small, and the admissible block number has not saturated. In addition, when truncation error is set to be smaller, the solution error measured by relative residual also becomes smaller. Meanwhile, the complexity of the proposed solver remains linear regardless of the choice of accuracy parameter.

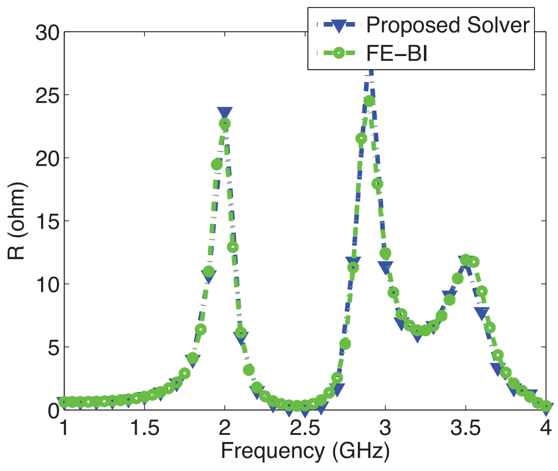
In this example, we also simulated even larger electrical sizes and examined the maximal rank across a wide range of electrical size from small to 73 wavelengths. As can be seen from Fig. 7, the rank grows slowly with electrical size, and the growth rate agrees with the theoretical bound given in [18] and [19]. For the 73-wavelength case involving 3600 antenna elements and 10 147 169 unknowns, the CPU time of the proposed solver is shown to be less than 4000 s for factorization and the memory cost is less than 6.4 GB, with a truncation error of $1e-4$.

B. Large-Scale Inductor Array

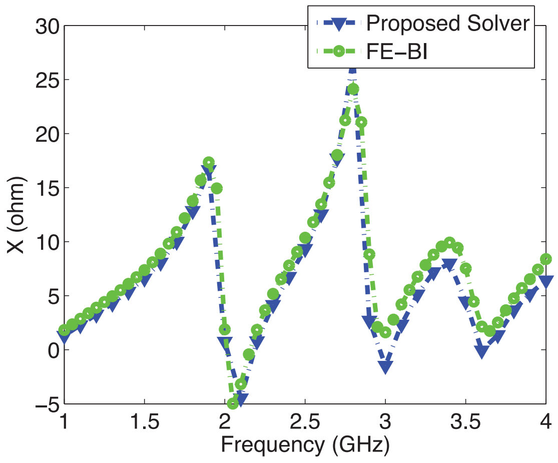
Next, a large-scale 3-D inductor array [9], [13] is simulated at 100 GHz. The detailed geometrical and material data of a single inductor element can be found in [9]. Notice that the same



(a)



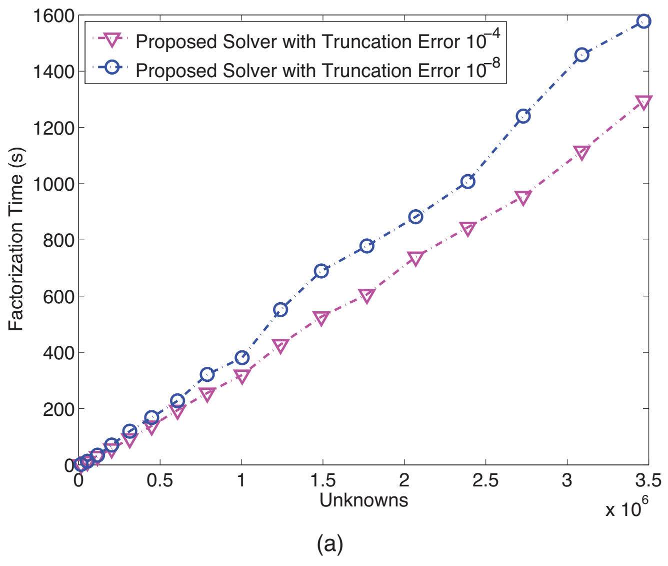
(b)

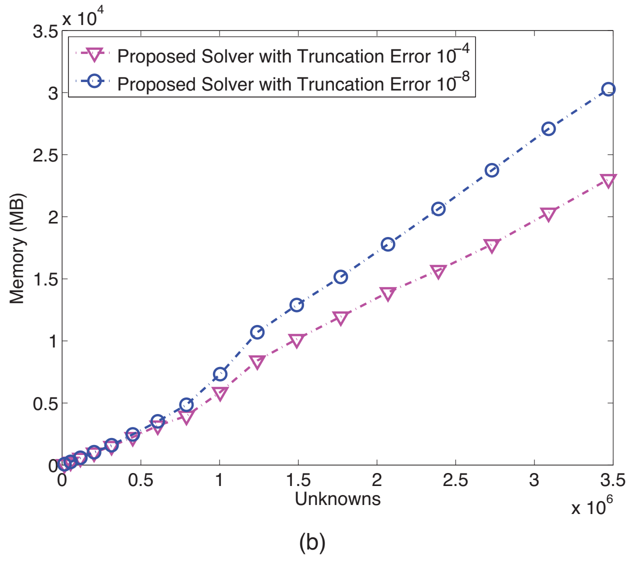


(c)

Fig. 5. Simulation of a cavity-backed microstrip patch antenna in comparison with reference FE-boundary integral results. (a) Illustration of the structure (after [21] and [22], $A = 7.5$ cm, $B = 5.1$ cm, $L = 5$ cm, $W = 3.4$ cm, $d = 0.08779$ cm. The location of load is shown in the figure with origin located at patch center). (b) Input resistance R. (c) Input reactance X.

structure was simulated in [13], but at a 10$\times$ lower frequency. A UNIX server having a single AMD CPU core at 2.8 GHz is used. The mesh density is fixed, while the array-element number is increased from 2×2 to 14×14, resulting in an unknown number from 117 287 to 5 643 240. The electrical size of the largest array is around 15.6 wavelengths. The conductors are also discretized





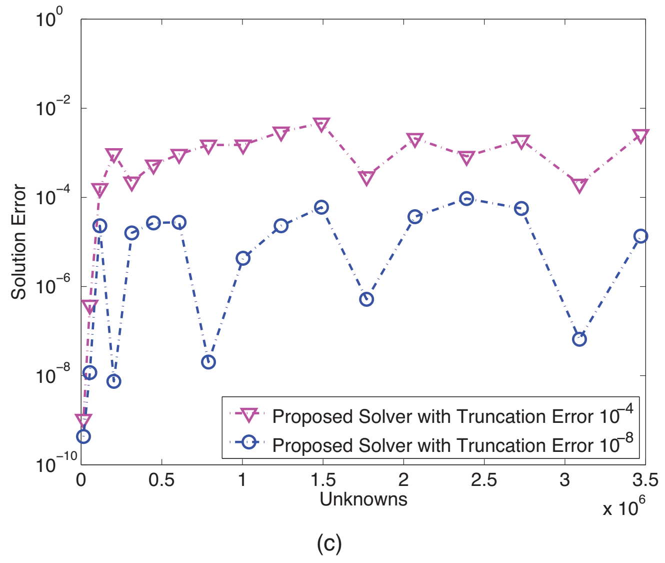


Fig. 6. Simulation of a suite of patch antenna arrays from 14 449 to 3.47 million unknowns at 2.75 GHz for two choices of truncation error. (a) LU factorization time. (b) Memory. (c) Solution error.

because of finite conductivity 5.8e + 7 S/m. The simulation parameters used are leafsize $= 8$, h-leafsize $= 16$, and $\eta = 3$. The

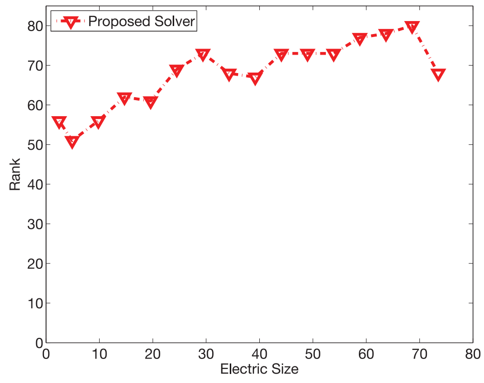


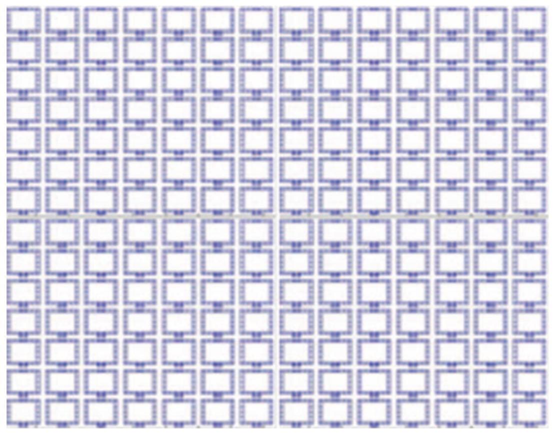
Fig. 7. Rank versus electrical size of the large-scale patch antenna arrays.

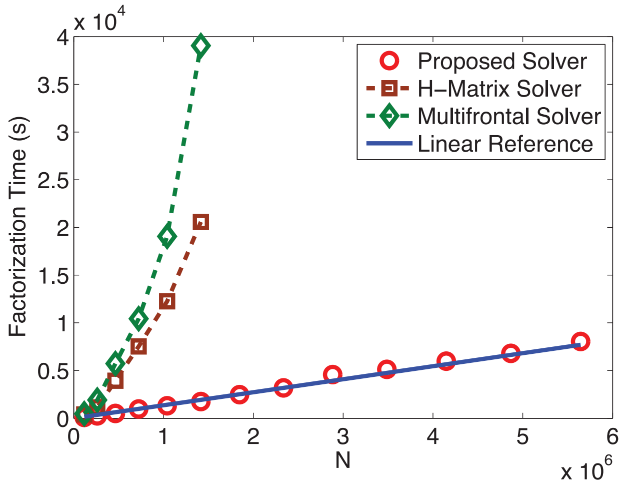
truncation error ϵ is set to be $6e - 5$. In Fig. 8(b), (c), and (d), the factorization time, memory, and accuracy (assessed by relative residual) of the proposed solver is, respectively, plotted versus N. As can be seen, good accuracy is achieved across the entire unknown range. For a higher order accuracy, instead of reducing ϵ, one can add a few steps of iterative refinement, which is a common approach used in direct sparse solvers. For example, by adding a few (less than ten) steps of iterative refinement after the solution is obtained, the accuracy can be improved to the 10^{-10} level, as shown by triangular marks in Fig. 8(d). Meanwhile, the cost of iterative refinement is negligible since $\mathbf{L}$ and $\mathbf{U}$ have been computed. For comparison, a state-of-the-art multifrontal based solver [2] and an $\mathcal{H}$-matrix based direct FEM solver [9] are also used to simulate the same example. From Fig. 8, it can be seen that neither the multifrontal solver, nor the conventional $\mathcal{H}$-matrix solver is capable of simulating larger than 7×7 arrays due to their large memory requirements. In contrast, the proposed direct solver greatly outperforms the two solvers in time and memory with excellent accuracy achieved in the entire unknown range. More importantly, the solver demonstrates a clear linear complexity, and hence, is capable of solving much larger problems. The proposed direct solver was also compared with a commercial-grade iterative FEM solver available to us on PC platforms. For the 7 $\times$ 7 array having 98 right-hand sides, on a 2.4-GHz Intel CPU, the proposed direct solver only cost 3313 s to solve the problem with a matrix of size $N = 1\,415\,130$, whereas the iterative FEM solver cost 8102 s to solve the same problem with $N = 661\,682$ without discretizing conductors.

C. Simulation of System-Level Signal and Power Integrity Problems

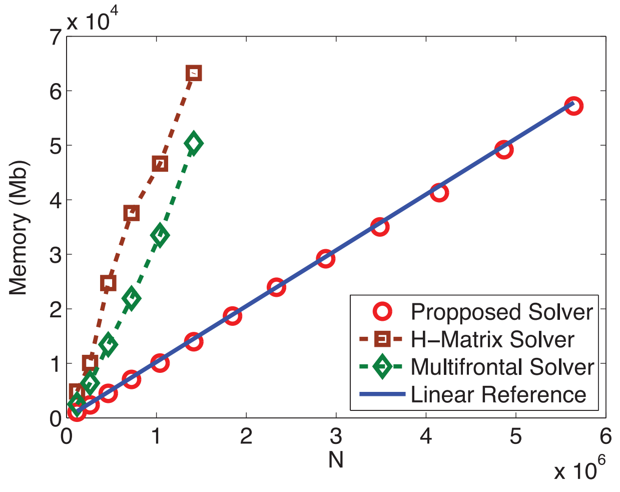
To examine the capability of the proposed direct finite-element solver in solving real-world large-scale problems, an IBM product-level full package, a picture of which can be seen in [23], is simulated from 100 MHz to 30 GHz. The package consists of 92 k unique lines, pins, shapes, and other elements [24], [23], having eight metal layers and seven dielectric layers. The metal has finite conductivity of $5.8e + 7$ S/m, and the dielectric layers have different dielectric constants.

We first develop a geometrical processing module to obtain the geometry and material data from the board file of the full IBM package. The layout of the full package reproduced by our geometrical processing module is plotted in Fig. 9, in which the blue region (in online version) is occupied by metals,

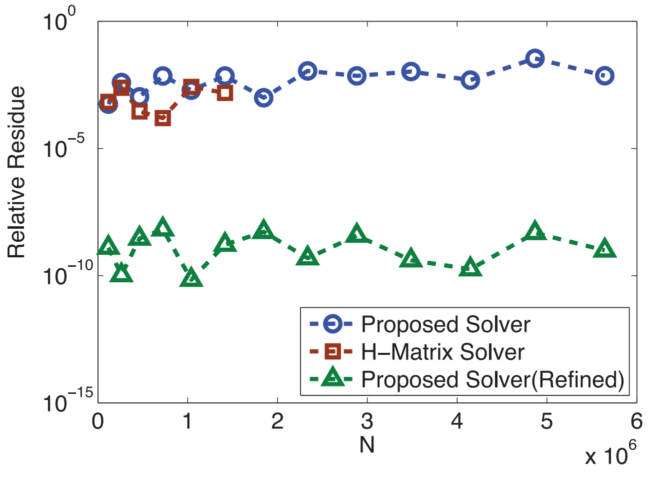

(a)



(b)



(c)



(d)

Fig. 8. Simulation of a suite of 3-D inductor arrays from 117 287 to 5 643 240 unknowns at 100 GHz. (a) Illustration of an 14 by 14 array. (b) LU factorization time versus N (number of unknowns). (c) Memory. (d) Accuracy.

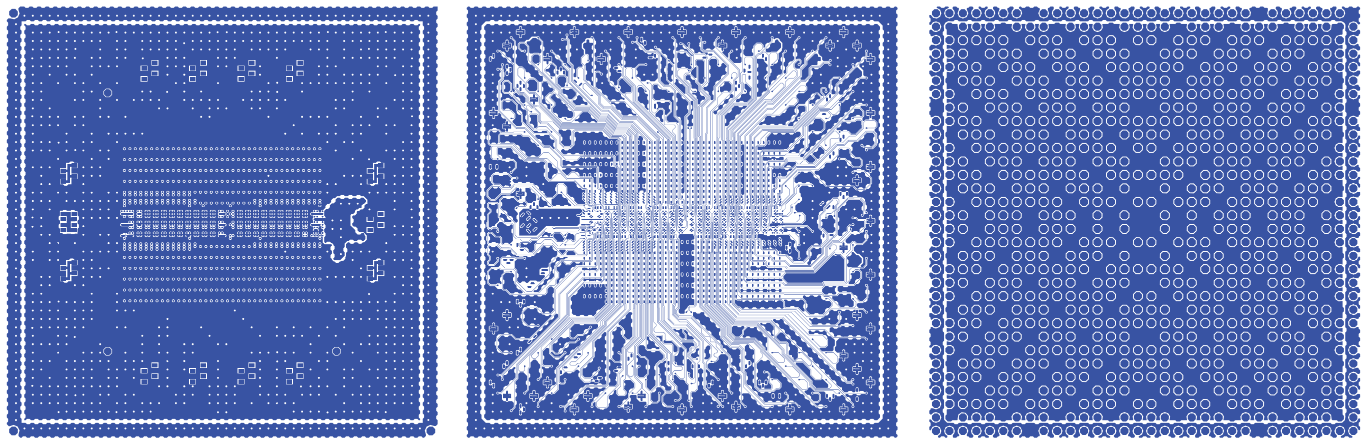

(a) (b) (c)

Fig. 9. Layout of a product-level package in different layers. (a) Layer 0. (b) Layer 2. (c) Layer 14.

whereas the white region signifies a dielectric region. The even-numbered layers are metal layers, while the odd-numbered layers only contain dielectrics and vias for connection. Notice that a *metal layer* in semiconductor industry does not refer to a layer fully occupied by metal. It only means a metal layer in the processing technology in which the metals are present and can be etched to make conductors of all kinds of shapes. Between conductors in the metal layer is still dielectrics. The entire layout is meshed into triangular prism elements.

We then simulate a suite of 19 substructures of the full package, as illustrated in Fig. 10. The size of these structures progressively increases from the smallest one of 500 μm in

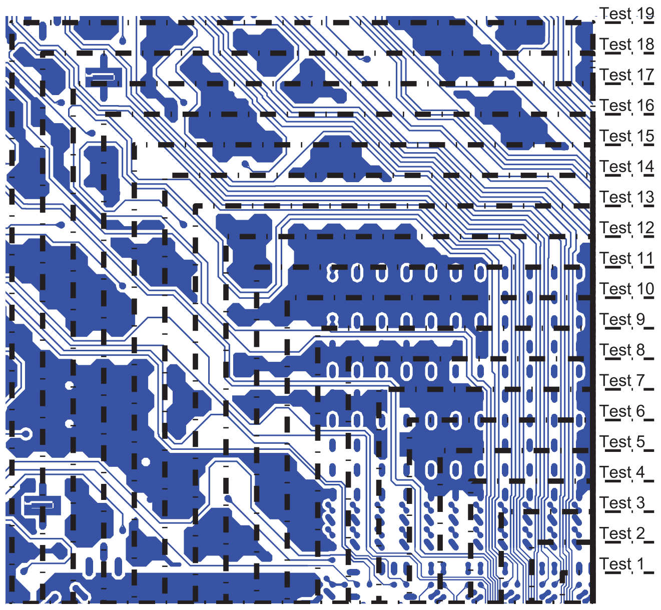


Fig. 10. 19 structures generated for solver performance verification.

width, 500 μm in length, and four layers in thickness, to the largest one of 9500 μm in width, and 9500 μm in length. The discretization results in from 31 276 to 15 850 600 unknowns. The simulation parameters used are leafsize $=$ 8, h-leafsize $=$ 16, η $=$ 3, and truncation error ϵ $=$ 10^{-6}. The computer used for simulation has 64-GB memory with a single CPU core running at 3 GHz.

In Fig. 11, we plot the CPU time and memory consumption of the proposed solver with respect to N in comparison with the direct finite-element solver that employs the most advanced direct sparse solvers. These solvers include `SuperLU 4.3` [20], `UMFPACK 5.6.2` [2], `MUMPS 4.10.0` [3], and `Pardiso` in Intel MKL 12.0.0 [4]. For a fair comparison, all the solvers are executed on the same machine that has an AMD 8222SE Opteron processor running at 3 GHz with 64-GB system memory. All the solvers including the proposed solver are compiled with the Intel Compiler if their source codes are available. For `Pardiso` whose source code is not available, we directly use its binary library in Intel MKL, which is a highly optimized kernel. In addition, optimization flag $-O3$ was used for all solvers when compiling. Intel MKL is used for all `blas` and `lapack` related routines; and MUMPS is run in its in-core mode.

In Fig. 11, the CPU time of each solver is plotted up to the largest N that the solver can handle on the given computing platform. It is clear that the proposed direct finite-element solver is much more efficient than the other solvers in both CPU time and memory consumption. More importantly, the proposed direct solver scales linearly with N in both time and memory complexity, whereas the complexity of the other direct solvers is shown to be much higher. With its optimal complexity, the proposed direct solver takes less than 1.6 h to solve the large 15.8 million unknown case on a single core. To examine the accuracy of the proposed solver, in Fig. 11(c), we plot its relative residual $\|\mathbf{YX}-\mathbf{B}\|/\|\mathbf{B}\|$ with respect to N. Excellent accuracy can be observed. Note the last point in Fig. 11(b) is due to the fact that the computer used has only 64-GB memory. The computation

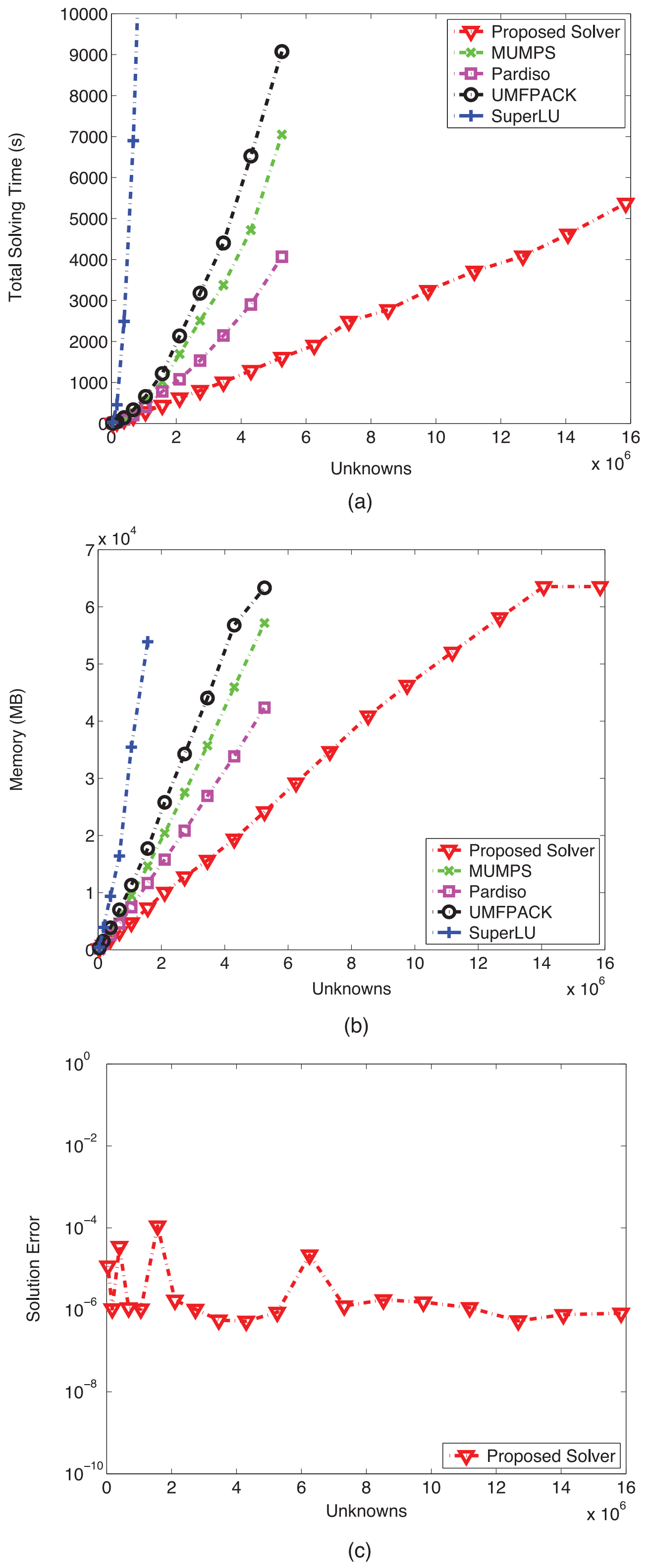


Fig. 11. Complexity and performance verification of the proposed direct solver for simulating a product-level package from 31 276 to 15 850 600 unknowns. (a) Time complexity. (b) Memory complexity. (c) Solution error (defined as relative residual $\|\mathbf{YX}-\mathbf{B}\|/\|\mathbf{B}\|$).

is still finished because the operating system manages to collect memory required for computation.

We also examined the CPU cost as a function of accuracy of the proposed method. We set the truncation error ϵ to be

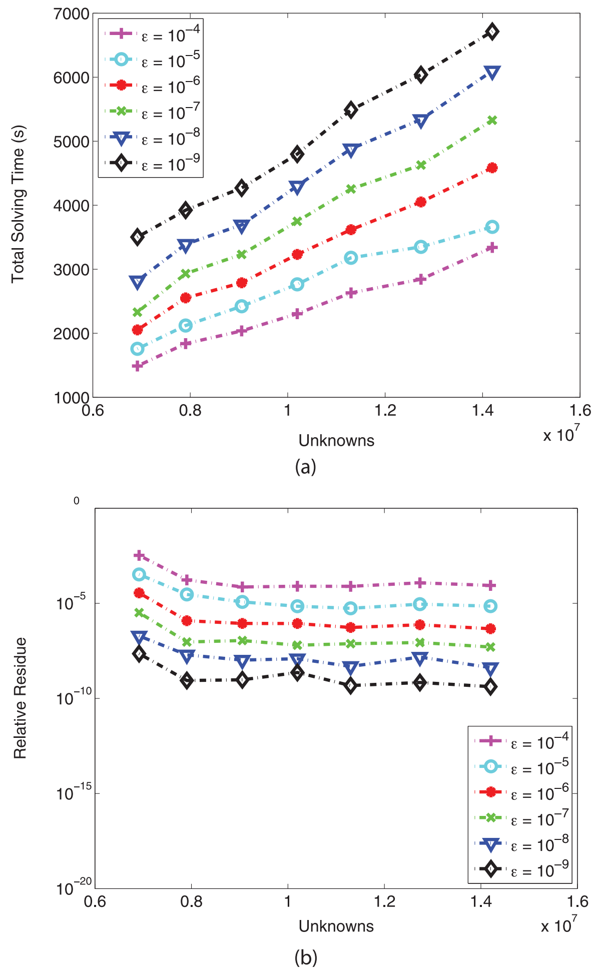


Fig. 12. CPU time versus N as a function of accuracy. (a) CPU time. (b) Solution error.

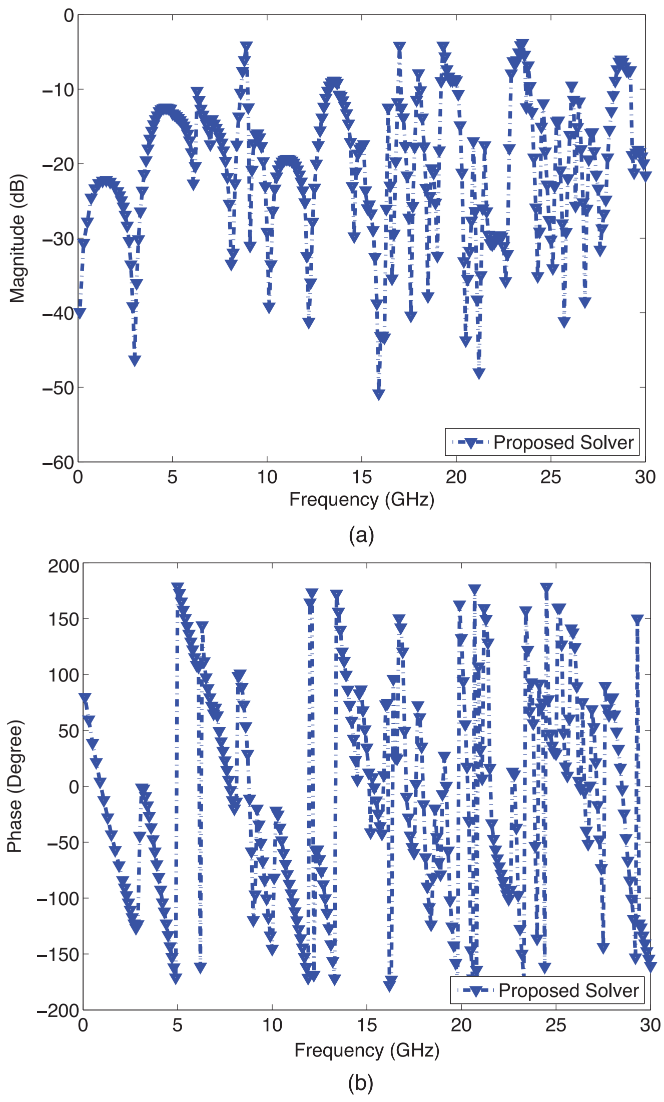


Fig. 13. S-parameters measured at the input and the output ports versus frequency. (a) Magnitude. (b) Phase.

$10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}$, and 10^{-9}, respectively. For each accuracy setting, we simulated a suite of the IBM plasma structures. The CPU time corresponding to different accuracy settings with respect to N is shown in Fig. 12(a), and the solution error in terms of relative residual is shown in Fig. 12(b). It is evident that the accuracy of the proposed direct solution can be controlled by parameter ϵ without sacrificing the linear complexity of the proposed direct solver.

Next, we benchmark the accuracy of the proposed solver with measurements. The critical circuits measured involve eight interconnects having 16 ports for which S-parameters were extracted from 100 MHz to 30 GHz by the proposed solver. Since the measurements were done in the time domain, which covered a broad frequency range from 30 GHz all the way down to zero frequency, for low-frequency data below 100 MHz, we employ the method in [25] to obtain field solutions and thereby S-parameters. Basically, we can use one solution obtained at a frequency where the FEM has not broken down to accurately obtain the FEM solutions at any breakdown frequency, as shown in [25]. Above the breakdown frequency, there is a range of frequencies (up to 2.1 GHz) where the FEM matrix is very ill conditioned although not singular yet. In this range, we set the truncation error as $5e-9$ to generate accurate results, while above this range we use $5e-6$ as the truncation error for all frequencies. It is also worth mentioning that the ill-conditioning issue of an FEM matrix is not as severe as that in an integral equation solver. This is because the FEM matrix is sparse, at any point of the direct solution procedure, one only needs to factorize a small frontal matrix instead of the big entire system matrix. This is because the elimination of any unknown only affects a subset of unknowns; whereas in a dense matrix resulting from an integral equation formulation, the elimination of any unknown affects all the other unknowns.

Since the interconnects are routed from the topmost layer (chip side) to the bottom-most layer (BGA side) [23], [24], vertically, the entire stack of the eight metal layers and seven interlayer dielectrics is simulated by the proposed solver. Horizontally, the area simulated is progressively enlarged until the simulation results do not have any noticeable change. The resultant number of unknowns is 3 149 880. It takes the proposed direct solver less than 3.3 h and 29-GB peak memory at each frequency to extract the 16 by 16 S-parameter matrix, on a single Intel Xeon E5410 CPU running at 2.33 GHz. With this set of S-parameters, one can study the time-domain behavior of the interconnects under any source and load conditions. In Fig. 13, we plot the crosstalk between the near end of line 6 located at

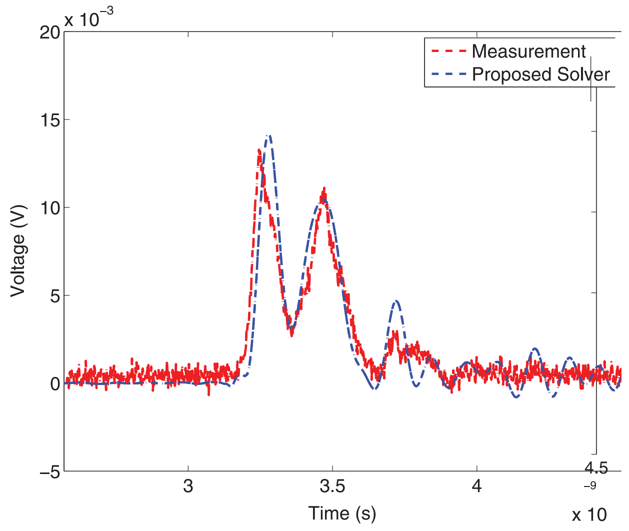


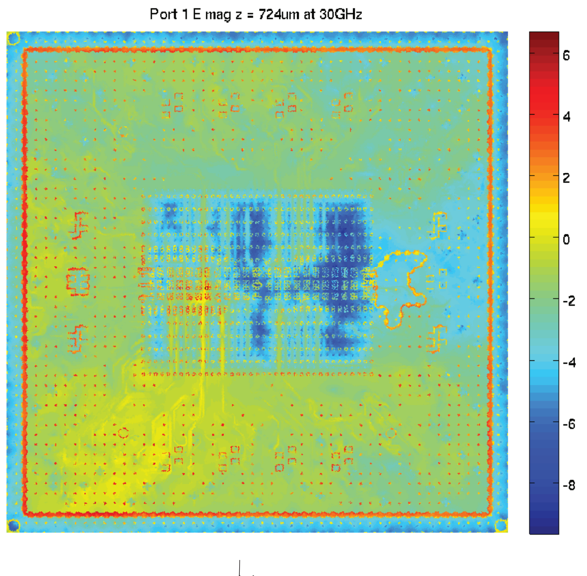
Fig. 14. Time-domain correlation with full-package measurements.

the chip side and the far end of line 2 located at the bottom BGA side, with all the other ports left open, from 100 MHz to 30 GHz. Since the measurements are performed in the time domain [24], [23], in Fig. 14 we plot the voltage obtained from the proposed solver in comparison with the full-package measurements. Very good agreement is observed. The voltage is measured at the far end of line 2 located at the bottom BGA side, with the near-end of line 6 on the chip side excited by a step function.

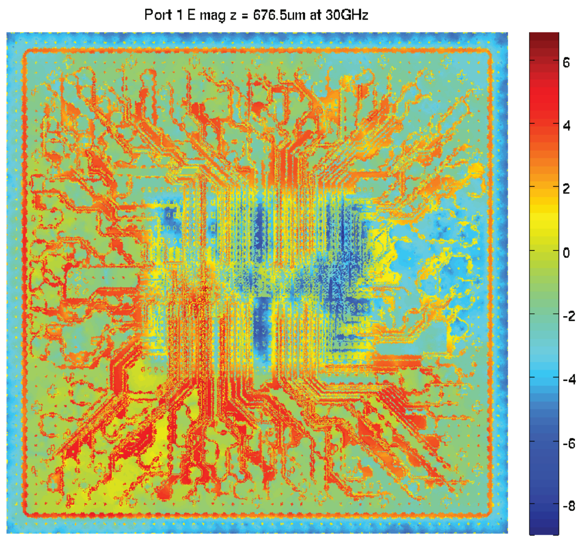
We have also simulated the *full* IBM package with the entire package area and the full stack of 15 dielectric layers taken into account. The total number of unknowns generated is 22.8488 million, where a fine resolution of 4 μm is used for critical circuits, a resolution of 20 μm for intermediate ones, and 95 μm elsewhere. It should be noted that we cannot scale the CPU time cost in simulating the above 15.8 million unknown case to obtain the time of the full-package case because the former only consists of four dielectric layers. Hence, the constant in front of N is different in the two examples. With $1e-6$ truncation error, the CPU time for the factorization and solution of the full package structure is found to be 58 976 s ($\sim$16 h), and the memory consumption is 224.9 GB, on a PowerEdge R620 server having 256-GB system memory with a single Intel Xeon E5-2690 3.0 GHz CPU. In Fig. 15, corresponding to Fig. 9, we plot the magnitude of electric field across the whole package area in different layers in log scale. The solution error of all 22.8488 million unknowns, measured in relative residual, is found to be 0.000360501.

VII. Conclusion

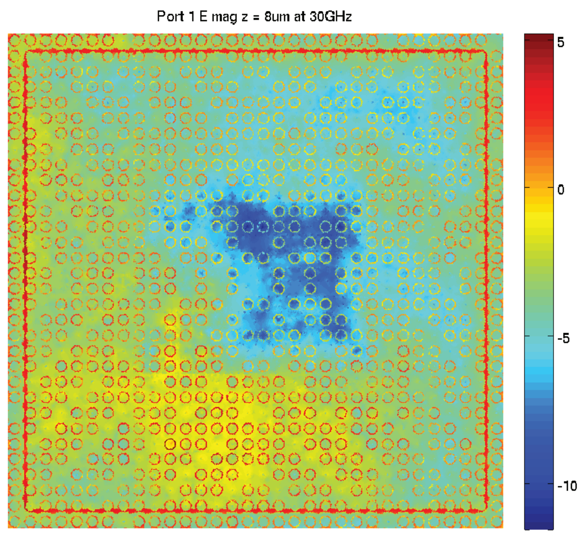
A linear-complexity direct sparse solver has been developed for FEM-based electromagnetic analysis of general 3-D problems containing arbitrarily shaped and inhomogeneous conductors and dielectrics. The computational complexity of the proposed direct solver has been theoretically proven and numerically verified for analyzing both circuit problems and electrically large problems. Comparisons with both state-of-the-art direct FEM solvers that employ most advanced sparse solvers such as `SuperLU 4.3` [20], `UMFPACK 5.6.2` [2], `MUMPS 4.10.0` [3], and `Pardiso` in Intel MKL [4], and a commercial iterative FEM solver have demonstrated the clear advantages



(a)



(b)



(c)

Fig. 15. Electric field distribution of a product-level full package in different layers simulated by the proposed solver at 30 GHz. (a) Layer 0. (b) Layer 2. (c) Layer 14.

of the proposed direct solver. In addition to numerical validation, the accuracy of the proposed direct solver has also been validated by measurements provided by industry. Recently, this work has also been extended to efficient LU solutions with many right-hand sides [26], and the solution of general sparse matrices without mesh information [27].

REFERENCES

[1] J. W. Liu, "The multifrontal method for sparse matrix solution: Theory and practice," *SIAM Rev.*, vol. 34, no. 1, pp. 82–109, 1992.

[2] T. A. Davis, "Algorithm 832: Umfpack v4. 3—An unsymmetric-pattern multifrontal method," *ACM Trans. Math. Softw.*, vol. 30, no. 2, pp. 196–199, 2004.

[3] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent, "Multifrontal parallel distributed symmetric and unsymmetric solvers," *Comput. Methods Appl. Mech. Eng.*, vol. 184, no. 2, pp. 501–520, 2000.

[4] Intel Math Kernel Library. ver. 12.0.0, Intel Corporation, Santa Clara, CA, USA, 2011.

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

[6] P. Gurgul, "A linear complexity direct solver for h-adaptive grids with point singularities," *Procedia Comput. Sci.*, vol. 29, pp. 1090–1099, 2014.

[7] D. Goik, K. Jopek, M. Paszynski, A. Lenharth, D. Nguyen, and K. Pingali, "Graph grammar based multi-thread multi-frontal direct solver with galois scheduler," *Procedia Comput. Sci.*, vol. 29, pp. 960–969, 2014.

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

[9] 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 Techn.*, vol. 58, no. 12, pp. 3697–3709, Dec. 2010.

[10] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Superfast multifrontal method for large structured linear systems of equations," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 3, pp. 1382–1411, 2009.

[11] H. Liu and D. Jiao, "Layered $\mathcal{H}$-matrix based inverse and LU algorithms for fast direct finite-element-based computation of electromagnetic problems," *IEEE Trans. Antennas Propag.*, vol. 61, no. 3, pp. 1273–1284, Mar. 2013.

[12] O. Schenk, K. Gartner, and W. Fichtner, "Efficient sparse LU factorization with left-right looking strategy on shared memory multiprocessors," *BIT Numer. Math.*, vol. 40, no. 1, pp. 158–176, 2000.

[13] B. Zhou, H. Liu, and D. Jiao, "A direct finite element solver of linear complexity for large-scale 3-D circuit extraction in multiple dielectrics," in *Proc. 50th ACM/EDAC/IEEE Design Automat. Conf.*, 2013, pp. 1–6.

[14] B. Zhou and D. Jiao, "A linear complexity direct finite element solver for large-scale 3-D electromagnetic analysis," in *IEEE AP-S Int. Symp.*, 2013, pp. 1684–1685.

[15] B. Zhou and D. Jiao, "Direct full-wave solvers of linear complexity for system-level signal and power integrity co-analysis," in *IEEE Signal Power Integrity Int. Conf.*, 2014, pp. 721–726.

[16] B. Zhou and D. Jiao, "Direct finite element solver of linear complexity for analyzing electrically large problems," in *2015 31st Int. Rev. Progr. Appl. Comput. Electromagn.*, 2014, pp. 1–2.

[17] S. Börm, L. Grasedyck, and W. Hackbusch, "Hierarchical matrices," Max Planck Inst. Math. Sci., Leipzig, Germany, Lecture Note 21, 2003.

[18] 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," in *IEEE Int. Antennas Propag. Symp.*, 2012, pp. 1–2.

[19] W. Chai and D. Jiao, "A 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.

[20] X. S. Li, "An overview of superlu: Algorithms, implementation, and user interface," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 302–325, 2005.

[21] D. Jiao and J. Jin, "Fast frequency-sweep analysis of cavity-backed microstrip patch antennas," *Microw. Opt. Technol. Lett.*, vol. 22, no. 6, pp. 389–393, 1999.

[22] J.-M. Jin, *The Finite Element Method in Electromagnetics*. New York, NY, USA: Wiley, 2002.

[23] J. D. Morsey *et al.*, "Massively parallel full-wave modeling of advanced packaging structures on bluegene supercomputer," in *IEEE Electron. Compon. Technol. Conf.*, 2008, pp. 1218–1224.

[24] J. Morsey, "Documents on IBM plasma package structure," IBM, Hopewell Junction, NY, USA, 2013.

[25] J. Zhu and D. Jiao, "A fast full-wave solution that eliminates the low-frequency breakdown problem in a reduced system of order one," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 2, no. 11, pp. 1871–1881, Nov. 2012.

[26] B. Zhou and D. Jiao, "Linear-complexity direct finite element solver accelerated for many right hand sides," in *IEEE AP-S Int. Symp.*, 2014, pp. 1383–1384.

[27] B. Zhou and D. Jiao, "Linear-complexity direct finite element solver for irregular meshes and matrices without mesh," in *IEEE AP-S Int. Symp.*, 2015, pp. 1–2.

Bangda Zhou received the B.S. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2010, and is currently working toward the Ph.D. degree in electrical and computer engineering at Purdue Univesity, West Lafayette, IN, USA.

He is currently with the and works in the On-Chip Electromagnetics Group, School of Electrical and Computer Engineering, Purdue University. His research interests include computational electromagnetics, high-performance computing for the analysis of very large-scale integrated circuits and package problems, and inverse design of electromagnetic structures.

Mr. Zhou was the recipient of the Best Student Paper Award of the 2015 International Annual Review of Progress in Applied Computational Electromagnetics (ACES), the Best Paper in Session Award of the 2014 SRC TECHCON conference, and an Honorable Mention Award and a Best Student Paper Finalist Award of the IEEE International Symposium on Antennas and Propagation in 2013 and 2014, respectively.

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 joined the Technology Computer-Aided Design (CAD) Division, Intel Corporation, until September 2005, where she was 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. She is currently a Professor with Purdue University. She has authored 3 book chapters and over 220 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 very large scale integration (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 has served as a reviewer for many IEEE publications and conferences. She is an associate editor for the IEEE TRANSACTIONS ON COMPONENTS, PACKAGING, AND MANUFACTURING TECHNOLOGY. She was the recipient of the 2013 S. A. Schelkunoff Prize Paper Award of the IEEE Antennas and Propagation Society, which recognizes the Best Paper published in the IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION during the previous year. She has been named a University Faculty Scholar by Purdue University since 2013. She was among the 85 engineers selected throughout the nation for the National Academy of Engineerings 2011 U.S. 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 Logic Technology Development (LTD) Divisional Achievement Award, the Intel Corporation Technology CAD Divisional Achievement Award, the 2002 Intel Corporation Components Research Award, the Intel Hero Award (Intel-wide she was the tenth recipient), the Intel Corporation LTD Team Quality Award, and the 2000 Raj Mittra Outstanding Research Award presented by the University of Illinois at Urbana–Champaign.