Sparse Transformations and Preconditioners for 3-D Capacitance Extraction

Shu Yan, Vivek Sarin, and Weiping Shi

***Abstract*— Three-dimensional capacitance extraction algorithms are important due to their high accuracy. However, the current 3D algorithms are slow and thus their application is limited. In this paper, we present a novel method to significantly speed up capacitance extraction algorithms based on boundary element methods, under uniform and multiple dielectrics.**

The $n \times n$ coefficient matrix in the boundary element method is dense, even when approximated with the fast multipole method, where n is the number of panels needed to discretize the conductor surfaces and dielectric interfaces. As a result, effective preconditioners are hard to obtain and iterative solvers converge slowly. In this paper, we introduce a linear transformation to convert the $n \times n$ dense coefficient matrix into a sparse matrix with $O(n)$ nonzero entries, and then use incomplete factorization to produce a very effective preconditioner. For the $k \times k$ bus crossing benchmark, our method requires at most 4 iterations, whereas previous best methods such as FastCap and HiCap require 10-20 iterations. As a result, our algorithm is up to 70 times faster than FastCap and up to 2 times faster than HiCap on these benchmarks. Additional experiments illustrate that our method consistently outperforms previous best methods by a large magnitude on complex industrial problems with multiple dielectrics.

***Index Terms*— Parasitic extraction, capacitance extraction, boundary element method, iterative method, preconditioning.**

I. Introduction

CAPACITANCE extraction is important for timing verification and signal integrity analysis of VLSI circuits, multi-chip modules, printed circuit boards and packages. Most existing methods fall into two categories: *library look-up* where the layout is divided into sections and matched against a pre-characterized library to derive the capacitance value, or *field solver* where the electromagnetic field is solved to derive the capacitance. The library methods are faster, while the field methods are more accurate. As the technology shrinks, the demand for fast and accurate tools is increasing. In this paper, we try to meet this demand by proposing a novel technique to significantly speed-up fast multipole accelerated [1] boundary element method (BEM), which is used by many field solvers such as FastCap [2], HiCap [3], the multi-scale algorithm [4], and hybrid algorithms [5].

The linear system arising from BEM is often solved by iterative methods. However, the linear system is dense, even when approximated with the fast multipole method. As a result, effective preconditioners are hard to obtain and the iterative solvers converge slowly. In this paper, we propose PHiCap, a **P**reconditioned **Hi**erarchical algorithm for **Cap**acitance extraction. PHiCap uses a linear transformation to convert the dense linear system obtained from the hierarchical algorithm [3] to an equivalent sparse system. The sparse structure is exploited to construct preconditioners based on incomplete LU or incomplete Cholesky factorizations. The transformed linear system is solved by preconditioned GMRES or CG iterative methods (see, e.g., [6]). The rate of convergence of the iterative methods increases dramatically by using these preconditioners. For benchmark examples, PHiCap uses fewer iterations and runs significantly faster than previous methods such as FastCap [2] and HiCap [3]. The number of iterations used by PHiCap is also less than the multi-scale method [4].

In addition to fast multipole accelerated BEMs, there are other fast capacitance extraction algorithms, such as the pFFT algorithm [7], the singular value decomposition (SVD) algorithm [8] and the geometric independent method [9]. We do not know if our method can be applied to speed-up these algorithms.

The paper is organized as follows: In Section II, we review the integral equation approach for capacitance extraction for uniform and multiple dielectrics. In Section III, we introduce the new algorithm. We present experimental results in Section IV and conclusions in Section V.

II. Preliminaries

To compute the self and coupling capacitances, we need to compute the conductor surface charges, given certain conductor potentials. In general, the surface charges satisfy the integral equation

$$\psi(x) = \int_{S_c \cup S_d} \sigma(x')\, G(x, x')\, d\alpha', \tag{1}$$

where $\psi(x)$ denotes the known conductor surface potential, S_c is the the conductor surfaces, S_d is the dielectric-dielectric interfaces, σ is the charge densities on S_c and S_d, $G(x, x')$ is the Green's function, $d\alpha'$ is the incremental conductor surface area, and $x' \in d\alpha'$. The Green's function $G(x, x')$ has the form

$$G(x, x') = \frac{1}{4\pi\epsilon_0 \|x - x'\|},$$

where $\|x - x'\|$ denotes the Euclidean distance between x and x'.

In addition, the interface condition

$$\epsilon_a \frac{\partial \psi_a(x)}{\partial n_a} - \epsilon_b \frac{\partial \psi_b(x)}{\partial n_a} = 0 \tag{2}$$

must be satisfied at any point $x \in S_d$. Here, ϵ_a and ϵ_b are the permittivities of the two adjacent dielectrics a and b, n_a is

This research was supported by the NSF grants CCR-0098329, CCR-0113668, EIA-0223785, and ATP grant 512-0266-2001.

Shu Yan, Department of Electrical Engineering, Texas A&M University, College Station, TX 77843, shu@ee.tamu.edu

Vivek Sarin, Department of Computer Science, Texas A&M University, College Station, TX 77843, sarin@cs.tamu.edu

Weiping Shi, Department of Electrical Engineering, Texas A&M University, College Station, TX 77843, wshi@ee.tamu.edu

the normal to the dielectric-dielectric interface at x pointing into a, and $\partial\psi_a(x)/\partial n_a$ and $\partial\psi_b(x)/\partial n_a$ are the normal component of electric field at x in a and b, respectively. Here, the equivalent charge approach [10], [11] is used to deal with multiple dielectric.

To solve (1) and (2) numerically, the standard Galerkin scheme is used. In this approach, the conductor surfaces and dielectric-dielectric interfaces are divided into n small panels, $A_1, A_2, \ldots, A_n$, and a dense linear system is formed:

$$\begin{bmatrix} \mathbf{P_{cc}} & \mathbf{P_{cd}} \\ \mathbf{E_{dc}} & \mathbf{E_{dd}} \end{bmatrix} \begin{bmatrix} \mathbf{q_c} \\ \mathbf{q_d} \end{bmatrix} = \begin{bmatrix} \mathbf{v_c} \\ \mathbf{0} \end{bmatrix}, \tag{3}$$

where $\mathbf{q_c}$ and $\mathbf{q_d}$ denote the vector of charges on the conductor panels and dielectric-dielectric interface panels, respectively, and $\mathbf{v_c}$ denotes the vector of potentials on conductor panels. The (i,j) entry of $\mathbf{P_{cc}}$and $\mathbf{P_{cd}}$ are defined as

$$p_{ij} = \frac{1}{\text{area}(A_i)} \frac{1}{\text{area}(A_j)} \int_{A_i} \int_{A_j} G(x_i, x_j)\, d\alpha_j d\alpha_i.$$

The i-th diagonal entry of $\mathbf{E_{dd}}$ is defined as

$$e_{ii} = (\epsilon_a + \epsilon_b) \frac{1}{2a_i\epsilon_0}.$$

The off-diagonal entries of $\mathbf{E_{dd}}$ and the entries of $\mathbf{E_{dc}}$ are defined as

$$\begin{aligned} e_{ij} &= (\epsilon_a - \epsilon_b) \frac{\partial}{\partial n_a} \frac{1}{\text{area}(A_i)} \frac{1}{\text{area}(A_j)} \cdot \\ &\quad \cdot \int_{A_i} \int_{A_j} G(x_i, x_j)\, d\alpha_j d\alpha_i. \end{aligned}$$

The problem with uniform dielectric is a special case with the dielectric-dielectric interfaces removed.

III. The PHiCap Algorithm

In this section, we present PHiCap, the preconditioned hierarchical capacitance extraction algorithm. Reformulate linear system (3) as follows.

$$\mathbf{Pq} = \mathbf{v}. \tag{4}$$

We will show how to transform dense linear system (4) to a sparse system, which is then solved by a preconditioned iterative method. The algorithm is outlined below.

The PHiCap Algorithm

1) Construct the factorization $\mathbf{P} = \mathbf{J^T H J}$.
2) Transform the dense system $\mathbf{Pq} = \mathbf{v}$ to equivalent sparse system $\tilde{\mathbf{P}}\tilde{\mathbf{q}} = \tilde{\mathbf{v}}$.
3) Compute an incomplete factorization preconditioner for $\tilde{\mathbf{P}}$.
4) Solve $\tilde{\mathbf{P}}\tilde{\mathbf{q}} = \tilde{\mathbf{v}}$ by preconditioned CG or GMRES method.
5) Compute capacitance.

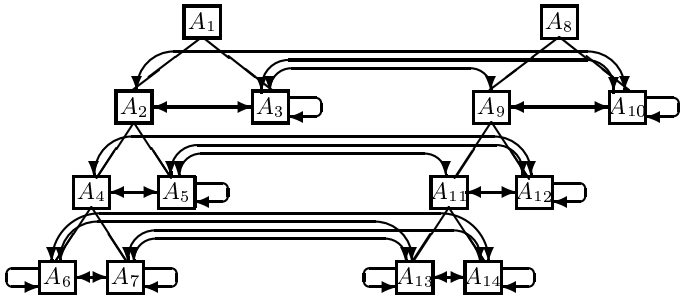


Fig. 1. The hierarchical data structure.

A. Factorization of $\mathbf{P}$ (Step 1)

We use the HiCap algorithm [3] to construct a hierarchical data structure to store the potential coefficient matrix $\mathbf{P}$. Fig. 1 shows an example of the hierarchical data structure. Each tree represents the partition of a conductor surface or a dielectric-dielectric interface. Each non-leaf node represents a panel that is further subdivided into two child panels. Each leaf node represents a panel that is not further subdivided. The coefficients are stored as links between the nodes.

In fact, the links stored in the hierarchical data structure are not exactly entries of $\mathbf{P}$, but a factorization of $\mathbf{P}$, which we now explain. Let n be the number of leaf panels and N be the total number of leaf and non-leaf panels. Let $\mathbf{H} \in \mathbb{R}^{N\times N}$ be the matrix where each nonzero entry represents a link between the corresponding panels in the hierarchical data structure. Let $\mathbf{J} \in \mathbb{R}^{N\times n}$ be the matrix representing the tree structure. Each row of $\mathbf{J}$ corresponds to a panel, either leaf or non-leaf, and each column corresponds to a leaf panel. Entry (i,j) of $\mathbf{J}$ is 1 if panel i contains the leaf panel j, and 0 otherwise. According to [3], we have the factorization: $\mathbf{P} = \mathbf{J^T H J}$. Here $\mathbf{P}$ is a dense matrix with $O(n)$ block entries and $\mathbf{H}$ is a sparse matrix with $O(n)$ nonzero entries.

B. Transforming the Linear System (Step 2)

1) Overview: Our transformation is based on the factorization $\mathbf{P} = \mathbf{J^T H J}$. Since $\text{rank}(\mathbf{J}) = n$, we can always construct an orthonormal transformation $\mathbf{F} \in \mathbb{R}^{N\times N}$ (described later in this section), such that

$$\mathbf{FJ} = \begin{bmatrix} \mathbf{W} \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$. Thus,

$$\begin{aligned} \mathbf{P} &= \mathbf{J^T H J} \\ &= \mathbf{J^T (F^T F) H (F^T F) J} \\ &= (\mathbf{FJ})^{\mathbf{T}} \left(\mathbf{FHF^T}\right) (\mathbf{FJ}) \\ &= \begin{bmatrix} \mathbf{W^T} & \mathbf{0} \end{bmatrix} \left(\mathbf{FHF^T}\right) \begin{bmatrix} \mathbf{W} \\ \mathbf{0} \end{bmatrix}, \end{aligned}$$

where $\mathbf{FHF^T}$ can be represented as

$$\mathbf{FHF^T} = \begin{bmatrix} \tilde{\mathbf{P}} & \times \\ \times & \times \end{bmatrix}.$$

Here, $\tilde{\mathbf{P}}$ is a sparse $n \times n$ matrix (we show this property later), and $\times$'s denote submatrices that do not contribute to $\mathbf{P}$. Since $\mathbf{P} = \mathbf{W^T}\tilde{\mathbf{P}}\mathbf{W}$, the dense linear system (4) is transformed to the sparse system

$$\tilde{\mathbf{P}}\tilde{\mathbf{q}} = \tilde{\mathbf{v}}, \tag{5}$$

where $\tilde{\mathbf{q}} = \mathbf{W}\mathbf{q}$ and $\tilde{\mathbf{v}} = \mathbf{W^{-T}}\mathbf{v}$.

2) Constructing $\mathbf{F}$*:* We first introduce a basic transformation that is used to construct $\mathbf{F}$. Consider the matrix

$$\hat{\mathbf{J}}_\mathbf{k} = \begin{bmatrix} 1 & 1 \\ c_k & 0 \\ 0 & c_k \end{bmatrix},$$

where c_k is a constant that depends on the height k of a node in the tree. There exists an orthonormal matrix

$$\hat{\mathbf{F}}_\mathbf{k} = \begin{bmatrix} \frac{2}{\sqrt{2({c_k}^2+2)}} & \frac{c_k}{\sqrt{2({c_k}^2+2)}} & \frac{c_k}{\sqrt{2({c_k}^2+2)}} \\ \frac{c_k}{\sqrt{{c_k}^2+2}} & -\frac{1}{\sqrt{{c_k}^2+2}} & -\frac{1}{\sqrt{{c_k}^2+2}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix},$$

such that

$$\hat{\mathbf{F}}_\mathbf{k}\hat{\mathbf{J}}_\mathbf{k} = \begin{bmatrix} c_{k+1} & c_{k+1} \\ 0 & 0 \\ e_k & -e_k \end{bmatrix}, \tag{6}$$

where

$$c_{k+1} = \sqrt{\frac{c_k^2+2}{2}}, \qquad e_k = \frac{c_k}{\sqrt{2}}, \qquad k \geq 1,$$

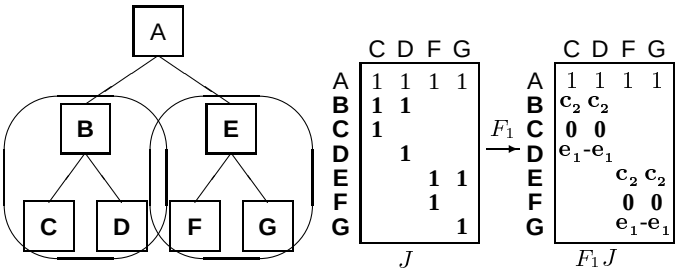
and $c_1 = 1$.

To simplify the discussion, we define an *element tree* as a tree with one root and two children. Given a hierarchical data structure and the corresponding matrix $\mathbf{J}$, the transformation is done by a depth-first traversal of the corresponding tree, propagating the transformation upward to the root. Fig. 2 illustrates the procedure. Starting from height $k = 1$, as shown in Fig. 2(a), for each *element tree* rooted at height 1, i.e., trees (B, C, D) and (E, F, G), we can identify the corresponding $\hat{\mathbf{J}}_\mathbf{1}$ blocks in $\mathbf{J}$. We construct $\mathbf{F_1}$ that transforms all $\hat{\mathbf{J}}_\mathbf{1}$ blocks to $\hat{\mathbf{F}}_\mathbf{1}\hat{\mathbf{J}}_\mathbf{1}$ blocks without changing anything else in $\mathbf{J}$. Next, as illustrated in Fig. 2(b), we identify the *element tree* at height $k = 2$, i.e., tree (A, B, E), and the corresponding $\hat{\mathbf{J}}_\mathbf{2}$ block in $\mathbf{F_1J}$. Note that the rows of A, B, and E have two instances of the $\hat{\mathbf{J}}_\mathbf{2}$ block in columns C and F and columns D and G, respectively. We construct $\mathbf{F_2}$ that transforms the $\hat{\mathbf{J}}_\mathbf{2}$ blocks to $\hat{\mathbf{F}}_\mathbf{2}\hat{\mathbf{J}}_\mathbf{2}$ blocks without changing anything else in $\mathbf{F_1J}$. In this way, the transformation is propagated to the root. Finally, as shown in Fig. 2(c), we move the nonzero rows to the top of the matrix using a permutation matrix $\mathbf{E}$. The overall transformation is given as $\mathbf{F} = \mathbf{EF_2F_1}$. It is easy to see that the nonzero rows of $\mathbf{FJ}$ correspond to the root node and nodes that are right children of other nodes. In other words, zero rows in $\mathbf{FJ}$ correspond to nodes that are left children of other nodes.

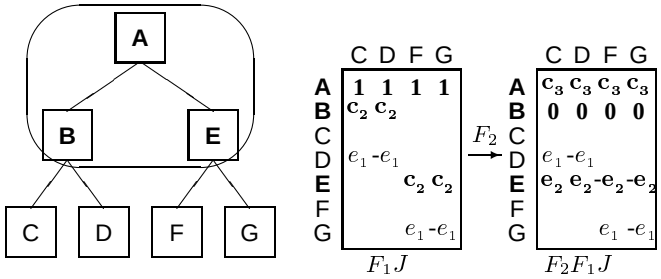
For a tree of height h, the transformation is

$$\mathbf{F} = \mathbf{EF_hF_{h-1}} \cdots \mathbf{F_2F_1},$$

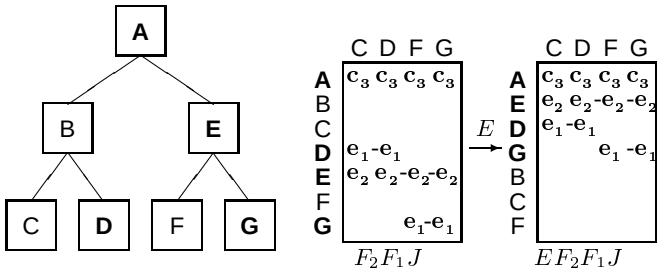
where $\mathbf{F_k}$ is constructed according to the element trees at height k. Since $\mathbf{F_1}, \mathbf{F_2}, \ldots, \mathbf{F_h}$, and $\mathbf{E}$ are orthonormal, the transformation matrix $\mathbf{F}$ is orthonormal.



(a) Transformation with F_1



(b) Transformation with F_2



(c) Permuting rows with E

Fig. 2. Construction of transformation $\mathbf{F}$ for a tree of height 2.

3) Computing $\mathbf{FHF^T}$*:* The matrix $\mathbf{H}$ is transformed into $\mathbf{FHF^T}$ by applying the transformations $\mathbf{F_k}$ as shown below

$$\mathbf{H_{k+1}} = \mathbf{F_kH_kF_k^T}, \qquad k = 1, 2, \ldots, h,$$

where $\mathbf{H_1} = \mathbf{H}$, and then by applying the permutation matrix $\mathbf{E}$:

$$\mathbf{FHF^T} = \mathbf{EH_{h+1}E^T}.$$

In the hierarchical data structure, this is done by a depth-first traversal of the tree, propagating the transformation upward, in a manner similar to the process of constructing the matrix $\mathbf{F}$.

In the transformed matrix $\mathbf{FHF^T}$, we are concerned only with the submatrix $\tilde{\mathbf{P}}$ that contains the links among root nodes and right child nodes. The matrix $\tilde{\mathbf{P}}$ can be treated as a sparse matrix with the number of nonzeros that are comparable to the number of block entries in $\mathbf{P}$(see Fig. 3).

4) Computing $\tilde{\mathbf{v}}$*:* The rows of the transformed matrix $\hat{\mathbf{F}}_\mathbf{k}\hat{\mathbf{J}}_\mathbf{k}$ are orthogonal. It follows that the rows of $\mathbf{W}$ are mutually orthogonal, and that $\mathbf{WW^T}$ is a diagonal matrix with values $2^k c_{k+1}^2$, where k is the height of the corresponding node in

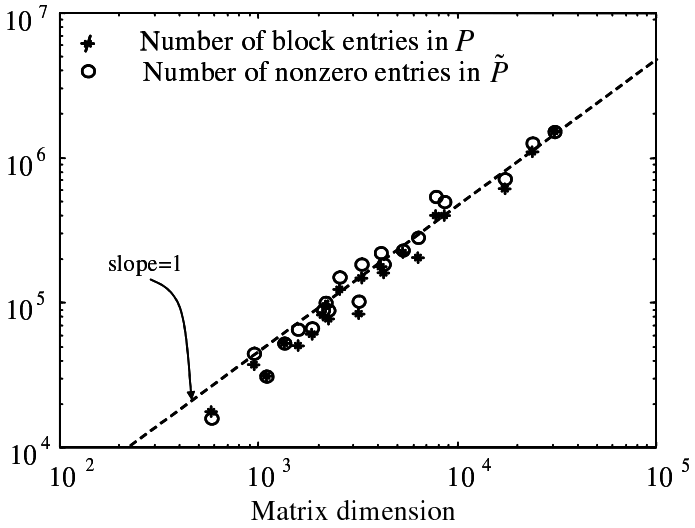


Fig. 3. The number of nonzero entries in $\tilde{\mathbf{P}}$ and the number of block entries in $\mathbf{P}$ are comparable, for various problem sizes.

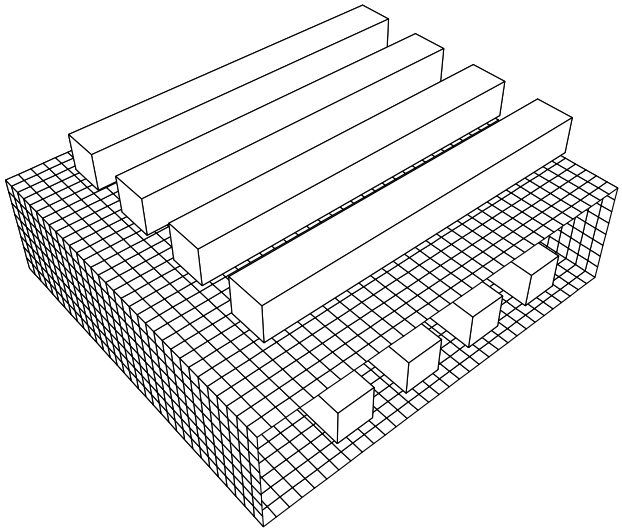


Fig. 4. 4x4 bus with two layers of dielectrics (section view).

the tree. This property is exploited when computing $\tilde{\mathbf{v}}$:

$$\tilde{\mathbf{v}} = \mathbf{W}^{-\mathbf{T}}\mathbf{v} = \left(\mathbf{W}\mathbf{W}^{\mathbf{T}}\right)^{-\mathbf{1}}\mathbf{W}\mathbf{v}.$$

Furthermore, the sum of entries in each row of $\mathbf{W}$ is zero for all nodes except the root. The sum of entries in rows corresponding to root nodes at height k is $2^k c_{k+1}$. As a result, $\tilde{\mathbf{v}}$ has nonzero entries of value c_{k+1}^{-1} in the locations corresponding to the roots of the conductor surfaces at unit potential.

The preceding description is for a balanced tree in which all leaf nodes are at height 1. An unbalanced tree can be embedded into a balanced tree by adding additional dummy nodes, and the above procedure can be followed. An efficient implementation can be developed by avoiding the actual construction of dummy nodes.

C. Solving the Transformed System (Steps 3-4)

For problems in uniform medium, the sparse linear system (5) is symmetric. We use the incomplete Cholesky factorization with no fill [6] to compute the preconditioner. Preconditioned Conjugate Gradients method is used to solve the system. For problems with multiple dielectrics, the sparse linear system is nonsymmetric. The preconditioner is computed from an incomplete LU factorization with no fill [6]. We use right preconditioned GMRES method to solve the system.

D. Computing Capacitance (Step 5)

Capacitance can be computed from $\tilde{\mathbf{q}}$ directly without computing $\mathbf{q}$. Recall that $\tilde{\mathbf{q}} = \mathbf{W}\mathbf{q}$, and that the rows of $\mathbf{W}$ corresponding to root nodes at height k have identical nonzero entries with value c_{k+1}. Thus, a root node entry in $\tilde{\mathbf{q}}$ is c_{k+1} times the sum of all the leaf panel charges in that tree. Capacitance can be computed by adding the root node entries of each conductor in $\tilde{\mathbf{q}}$ after scaling them by corresponding factors c_{k+1}^{-1}.

E. Complexity Analysis

The complexity of constructing the factorization of $\mathbf{P}$ in Step 1 is $O(n)$ [3]. The transformation of the linear system in Step 2 usually takes $O(nh)$ time, where h is the height of the tree. Normally, $h = O(\log n)$. Since the number of nonzeros in $\tilde{\mathbf{P}}$ is $O(n)$, the incomplete factorization can be done in $O(n)$ time. Each iteration requires a matrix-vector product with $\tilde{\mathbf{P}}$, and a solution of the lower and upper triangular systems of the preconditioner. Thus, Steps 3 and 4 take $O(n)$ time when the number of iterations is small. Capacitance can be computed in constant time. The overall complexity of this algorithm is normally $O(n \log n + mn)$, where m is the number of conductors.

IV. Experimental Results

We compare PHiCap with the following algorithms: FastCap with expansion order 2, FastCap with expansion order 1, and HiCap. Other methods, such as SVD [8] and pFFT [7], exhibit performance that is similar to FastCap. No benchmark experiments were reported for the geometric independent method [9]. In [3], HiCap algorithm can only solve problems in uniform media. To make the comparison complete, we extend HiCap to the multiple dielectrics case. The algorithms are executed on a Sun UltraSPARC Enterprise 4000. Unless otherwise noted, the iterations are terminated when the relative residual norm of the preconditioned system is reduced below 10^{-2}.

The first set of benchmarks are $k \times k$ bus crossing structures from [2]. Each bus is scaled to $1m \times 1m \times (2k+1)m$. The distance between the adjacent buses in the same layer is $1m$ and the distance between the two bus layers is $2m$. For the uniform dielectric cases, the permittivity is assumed to be ϵ_0. For the multiple dielectric cases, see Fig. 4, the medium surrounding the upper layer conductors has permittivity $3.9\epsilon_0$ and the medium surrounding the lower layer conductors has permittivity $7.5\epsilon_0$. The shaded box represents the interface of the two layers.

Tables I and II compare the four algorithms. PHiCap is the fastest and uses much less memory compared to FastCap. The current implementation of PHiCap uses more memory compared to HiCap because of the additional storage needed for the transformed system $\tilde{\mathbf{P}}$. The storage requirement can be reduced by computing $\tilde{\mathbf{P}}$ directly, since it is not necessary to construct $\mathbf{H}$. In HiCap and PHiCap algorithms, the discretization threshold P_ϵ is chosen to be 0.008. This ensures that the relative error in the capacitance matrix computed by PHiCap is below 3%, which is acceptable in practice. The relative error in the capacitance matrix $\mathbf{C}'$, which is computed by HiCap or PHiCap algorithms, is defined as $\|\mathbf{C}-\mathbf{C}'\|_F/\|\mathbf{C}\|_F$, where $\|\cdot\|_F$ denotes the Frobenius norm. As per standard practice, $\mathbf{C}$ is computed by FastCap with expansion order 2.

Table III shows the first and second rows of the capacitance matrix computed by PHiCap and FastCap. It is easy to see that for the self capacitances and significant coupling capacitances, where a coupling capacitance is considered significant if it is greater than 10% of the self capacitance, PHiCap's error is mostly within 3%, with respect to FastCap with expansion order 2. The error for the small coupling capacitances is sometimes large, which is acceptable since the small coupling capacitances have minor influence on the circuit performance. Fig. 5 shows the error distribution of the self capacitances and the significant coupling capacitances for the six benchmark examples in Table I and II.

TABLE I

COMPARISON FOR UNIFORM DIELECTRIC. TIME IS CPU SECONDS, ITERATION IS AVERAGE FOR SOLVING ONE CONDUCTOR, MEMORY IS MB, AND ERROR IS WITH RESPECT TO FASTCAP (ORDER=2).

	FastCap (order=2)	FastCap (order=1)	HiCap	PHiCap
4x4 Bus with Uniform Dielectric				
Time	18.6	19	0.5	0.4
Iteration	8	14.9	9	3
Memory	25.7	16.7	0.7	1.0
Error	—	0.05%	2.1%	2.0%
Panel	2736	2736	1088	1088
6x6 Bus with Uniform Dielectric				
Time	113.9	68.5	2.4	1.5
Iteration	14.4	14.5	11.9	3.2
Memory	62.5	40.3	1.9	2.9
Error	—	1.1%	2.1%	2.2%
Panel	5832	5832	3168	3168
8x8 Bus with Uniform Dielectric				
Time	206	204	7.3	3.4
Iteration	12	21.9	13.0	3.9
Memory	112	67	3.1	4.9
Error	—	1.0%	3.1%	3.0%
Panel	10080	10080	4224	4224

The second set of benchmarks are complex industrial circuits containing 8 layers of dielectrics and 48, 68 and 116 conductors, respectively. The smallest case of 48 conductors is shown in Fig. 6. The results are in Table IV. FastCap cannot solve these examples because of prohibitive time and memory requirements. PHiCap displays near-optimal preconditioning on these experiments. Fig. 7 shows that the residual norm decreases rapidly for PHiCap. In contrast, the decrease is slower for HiCap. As a result, PHiCap requires much less time to solve the problem.

TABLE II

COMPARISON FOR MULTIPLE DIELECTRICS. TIME IS CPU SECONDS, ITERATION IS AVERAGE FOR SOLVING ONE CONDUCTOR, MEMORY IS MB, AND ERROR IS WITH RESPECT TO FASTCAP (ORDER=2).

	FastCap (order=2)	FastCap (order=1)	HiCap	PHiCap
4x4 Bus with Two Layer Dielectrics				
Time	63	36	1.7	2
Iteration	13	14	9	3
Memory	68	39	3.0	3.9
Error	—	0.6%	1.0%	1.0%
Panel	3456	3456	2120	2120
6x6 Bus with Two Layer Dielectrics				
Time	162	104	10.4	5.8
Iteration	17.1	17	11.3	3
Memory	92	61	6.3	8.3
Error	—	0.5%	1.0%	1.2%
Panel	5448	5448	4120	4120
8x8 Bus with Two Layer Dielectrics				
Time	324	197	32	15
Iteration	18	18	12.8	3
Memory	133	86.9	11.5	15.3
Error	—	0.0%	1.4%	1.4%
Panel	7968	7968	6784	6784

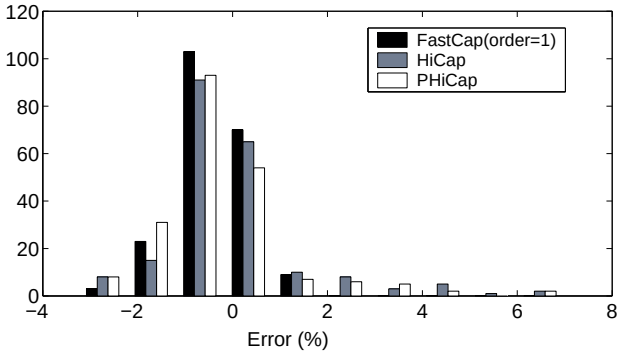


Fig. 5. Error distribution of self capacitance and significant coupling capacitance for the 6 examples in Table I and II.

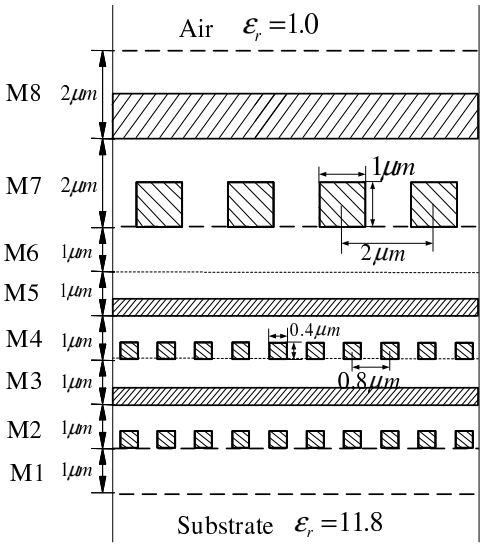


Fig. 6. Example with 48 metal conductors and 8 dielectric layers. Metal wires are shaded. Relative permittivity of M1 is 3.9, M2 through M6 is 2.5, and M7 and M8 is 7.0. Layers M2 through M5 have 10 conductors each whereas layers M7 and M8 have 4 conductors each.

TABLE III

FIRST TWO ROWS OF CAPACITANCE MATRIX COMPUTED BY PHICAP AND FASTCAP (ORDER=2) FOR 4X4 BUS WITH UNIFORM DIELECTRIC.

	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}
FastCap	405.54	-137.54	-12.02	-8.07	-48.40	-40.26	-40.17	-48.48
PHiCap	405.78	-139.30	-18.89	0.10	-48.05	-39.77	-39.84	-46.60
	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}	C_{27}	C_{28}
FastCap	-137.54	468.23	-132.66	-11.89	-40.15	-32.59	-32.54	-40.20
PHiCap	-139.32	468.27	-129.49	-18.30	-39.96	-31.49	-31.21	-41.04

TABLE IV

COMPARISON OF HICAP AND PHICAP FOR COMPLEX MULTIPLE DIELECTRIC PROBLEMS SHOWN IN FIG. 6. TIME IS CPU SECONDS, ITERATION IS AVERAGE FOR SOLVING ONE CONDUCTOR, AND MEMORY IS MB. FASTCAP WAS UNABLE TO SOLVE THESE PROBLEMS.

	48 conductors		68 conductors		116 conductors	
	HiCap	PHiCap	HiCap	PHiCap	HiCap	PHiCap
Time	533	122	3011	389	12930	2391
Iteration	18.7	2.8	25.3	3.0	36.8	5.1
Memory	43	59	115	161	406	570
Panel	19840	19840	42912	42912	138552	138552

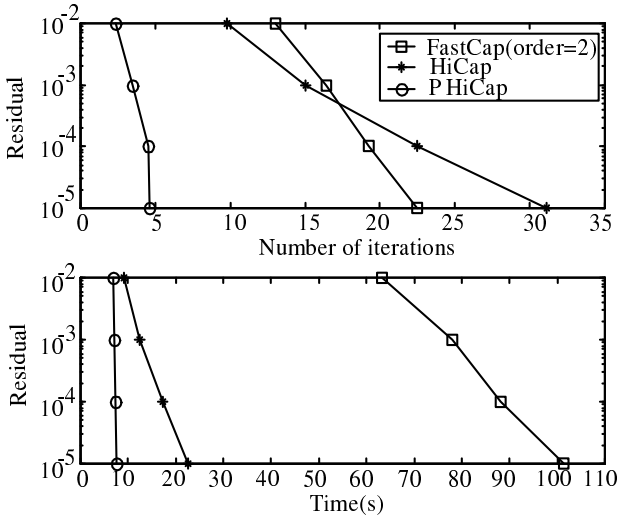


Fig. 7. Comparison of the rate of convergence of preconditioned GMRES method.

The third benchmark we studied is the parallel-plate problem. It is well known that this problem yields ill-conditioned systems when the two plates are very close to each other. We consider the problem with plate size $10\mathrm{m} \times 10\mathrm{m}$ and distance between two plates $0.1\mathrm{m}$. The results in Table V indicate that PHiCap performs very well on these problems too.

The multi-scale method [4] uses a similar idea to sparsify the dense matrix $\mathbf{P}$. However, there are important differences between the multi-scale method and our method. The multi-scale method is based on high-order FMM, whereas our method is based on HiCap. It was shown that the hierarchical approach in HiCap is more efficient and kernel independent [3]. The multi-scale method uses a block diagonal preconditioner, while ours uses incomplete Cholesky or LU factorizations. In addition, the multi-scale method has been applied to uniform dielectric only. For the $k \times k$ bus crossing

TABLE V

COMPARISON OF HICAP AND PHICAP FOR PARALLEL PLATES OF SIZE $10\mathrm{m} \times 10\mathrm{m}$ AND DISTANCE $0.1\mathrm{m}$. TIME IS CPU SECONDS AND ITERATION IS AVERAGE FOR SOLVING ONE CONDUCTOR.

	Coarse Discretization $P_\epsilon = 0.01$		Fine Discretization $P_\epsilon = 0.005$	
	Time	Iteration	Time	Iteration
HiCap	100.2	54	523.5	72.5
PHiCap	53.3	6	265.8	6

benchmarks, we compare the number of iterations needed by the two methods to reduce the residual norm below 10^{-9}. Table VI shows that the number of iterations required by PHiCap is less than the multi-scale method. The growth in iterations with the increase of the problem size is negligible for PHiCap. A more detailed comparison was not possible because only the number of iterations were reported in [4].

V. CONCLUSIONS

This paper proposes PHiCap, a preconditioned hierarchical algorithm for capacitance extraction. PHiCap transforms the dense linear system into a sparse system and then solves it by a preconditioned iterative method. The sparse structure allows construction of inexpensive but highly effective preconditioners based on incomplete factorization techniques. The dense-to-sparse transformation used in PHiCap is applicable to multipole-based methods as well, where the linear system can be represented by a block matrix. Numerical experiments demonstrate the superiority of PHiCap over FastCap and HiCap in terms of the number of iterations of the solver and the overall running time. Experiments on the $k \times k$ bus crossing benchmark show that PHiCap is up to 70 times faster than FastCap (order=2), up to 60 times faster than FastCap (order=1), and up to 2 times faster than HiCap. For complex industrial problems with multiple dielectrics, PHiCap is 4-8 times faster than HiCap. FastCap is not able to solve these problems due to their sizes.

ACKNOWLEDGMENTS

The authors wish to thank Sani Nassif for helpful suggestions and industrial examples.

REFERENCES

[1] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems.* Cambridge, MA: MIT Press, 1988.
[2] K. Nabors and J. White, "FastCap: A multipole accelerated 3-d capacitance extraction program," *IEEE Trans. on CAD*, pp. 1447–1459, 1991.
[3] W. Shi, J. Liu, N. Kakani, and T. Yu, "A fast hierarchical algorithm for 3-d capacitance extraction," *IEEE Trans. on CAD*, pp. 330–336, 2002.

TABLE VI

COMPARISON OF THE NUMBER OF ITERATIONS OF MULTI-SCALE METHOD AND PHICAP FOR $k \times k$ BUS CROSSING BENCHMARK. STOPPING CRITERIA FOR ITERATIVE SOLVER IS 10^{-9}.

	1×1	2×2	4×4	6×6	8×8
Multi-Scale	12	17	18	18	18
PHiCap	8	8	8	9	11

[4] J. Tausch and J. White, "A multiscale method for fast capacitance extraction," *Proc. DAC*, pp. 537–542, 1999.

[5] M. Beattie and L. Pileggi, "Electromagnetic parasitic extraction via a multipole method with hierarchical refinement," *Proc. ICCAD*, pp. 437–444, 1999.

[6] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA: SIAM, 2003.

[7] J. R. Phillips and J. White, "A precorrected FFT method for capacitance extraction of complicated 3-d structures," *IEEE Trans. CAD*, pp. 1059–1072, 1997.

[8] S. Kapur and D. E. Long, "IES^3: A fast integral equation solver for efficient 3-dimensional extraction," *Proc. ICCAD*, pp. 448–455, 1997.

[9] ——, "Large-scale capacitance calculation," *Proc. DAC*, pp. 744–749, 2000.

[10] S. M. Rao, T. K. Sarkar, and R. F. Harrington, "The electrostatic field of conducting bodies in multiple dielectric media," *IEEE Trans. on Microwave Theory Tech.*, pp. 1441–1448, 1984.

[11] K. Nabors and J. White, "Multipole-accelerated capacitance extraction algorithm for 3-d structures with multiple dielectrics," *IEEE Trans. on CAS*, pp. 946–954, 1992.

Responses to the reviewer's comments

Review Number 3.
****
Comments to the Author

The method is an extension of their previous work, HiCap, which is to certain extend can be regarded as the h- adaptive mesh refinement technique, but with pre-determined error estimate based on the relative distant between interacting panels. It was demonstrated in their previous article that HiCap is a very fast algorithm for capacitance extraction problems.

In this article, they improve their solver by implementing preconditioners, specifically via the incomplete Cholesky factorization, and incomplete LU factorization, that helps to improve its convergence rate. Although, these two approaches of devising preconditioners are not novel, their implementations to fast integral solvers are not obvious. Here, the authors present a systematic approach that transforms the original linear system, based on the original HiCap algorithm, to another sparse form that allows the devising of the preconditioners. Hence, their main contribution here is the transformation of the linear system.

Here are my comments and suggestions to the authors:
1) The first paragraph in the introduction is supposed to give an overview of the various fast algorithms, but it is not properly written. First, HiCap is a hierarchical based algorithm, but definitely do not accelerated via FMM. And it is quite meaningless to use another algorithm for reference [4]. Likewise, it is also meaningless to say that pFFT and SVD are not FMM based algorithm. The last sentence on the geometric independent method [8] of the paragraph is also sluggishly written, as it is not very informative about the method. Hence, I would hope that the authors would improve on this part.

Author's response: We rewrote the introduction part. We outlined the two categories of existing capacitance extraction methods, and explained that it is hard to find effective preconditioners for the existing BEM based methods because of the dense structure of the linear system. Then, we outlined our contribution, including the sparse transformations and effective preconditioners.

2) In the second paragraph of the introduction, the claim that their algorithm is significantly faster than many previous fast algorithms, including FastCap, HiCap, SVD, multi-scale and pFFT methods, is not evident. In this article, there are only explicit comparisons made between their new approach with FastCap and HiCap. Even for the comparison between their method and the multi-scale method, their experimental results only showed that their approach can converge to the desire accuracy in lesser iterations. But that does not means that it can be significantly faster in terms of CPU time. So in my opinion, it is not appropriate to make such strong claim of their new approach.

Author's response: We have revised the second paragraph of the introduction. We claim that the new method is faster than FastCap and HiCap, and the new method converges faster than multi-scale method. This claim is supported by the experiments in section IV. In reference [4], CPU time is not reported, and therefore, we can only compare number of iterations with multi-scale method. We do not compare the new method with SVD, pFFT and the algorithm in [9] directly, since we have no access to those tools.

3) The first paragraph of the preliminary section, i.e. regarding the computing of capacitance matrix, is redundant, as it is too fundamental. The authors may also want to consider revising or

combining the subsection A and B, since the main difference between the two sets of problems is the presence of the dielectric-dielectric layers for case B. And it is quite evident that the solution method for case A is the similar to case B with the dielectric-dielectric layer removed.

Author's response: As suggested by the reviewer, we simplified the part regarding the computing of the capacitance matrix to one sentence. We have also combined subsections II.A and II.B.

4) In section III, the authors can also consider to further summarize the first step of the PHiCap algorithm, which is essentially the HiCap algorithm [2]. Readers should be able to have the copy of the article [2] easily.

Author's response: As suggested by the reviewer, we have simplified the first step of PHiCap algorithm.

5) One major problem that I found quite disturbing is the accuracy of the method. In this article, and also their previous work [2], the authors have used the Frobenius norm of the capacitance matrix as a gauge to justify the accuracy of their methods. First, I would like to point out that it can be quite misleading to use the capacitance solution as an indication of the accuracy of fast solvers, including FMM. It was pointed out in a recent article (ET Ong, HP Lee and KM Lim, A Parallel Fast Fourier Transform on Multipoles (FFTM) Algorithm for Electrostatics Analysis of 3D Structures, IEEE Trans. CAD, vol. 23, no. 7, pp. 1063, 2004) that the capacitance solution can be very accurate even thought the primary solution (i.e. surface charge solution) is not as accurate. In general, it was observed that the capacitance solution can be up to 2 order more accurate than the surface charge solution (measured in L2 norm). This is mainly due to the global nature of the capacitance variable, where one is only interested in the summation of the surface charge solution. Furthermore, the used of the Frobenius norm of the capacitance matrix would further averaged the overall errors. Take for example the capacitance solutions in Table II of [2], where HiCap reported an error of only 1.8 in terms of the Frobenius norm, but as far as the capacitance entries were concerned, the errors can be as large as about 20 for C14. Hence, I think the authors may have too over concerned with the efficiency of the algorithm, but fail to address the accuracy of the method properly. And I suppose it is very important that a good numerical method should be able to account for both the efficiency and accuracy aspects properly.

Author's response: We agree with the reviewer that Frobenius norm tends to average the overall errors. To show the accuracy of each capacitance entry, we have added Table III and Fig.5. Table III includes individual capacitance of the first and second rows for the 4X4 bus uniform medium case, for both PHiCap and FastCap. Fig 5 is the error distribution of self-capacitance and significant coupling capacitance for the bus crossing examples. From the table and figure, it can be seen that for the self and significant coupling capacitance entries, the accuracy is very good. Although the error for small coupling capacitances is sometimes large, it is acceptable for circuit analysis purpose, since the small coupling capacitances have minor influence on the circuit performance. In practical circuit analysis applications, small coupling capacitances are usually dropped to maintain a manageable problem size.
As pointed out by the reviewer, the capacitance solution can not be used as an accuracy indicator for the fast solver, when the concern is the charge distribution. However, for the purpose of capacitance extraction, it is sufficient to compute the net charge on each conductor. Therefore, we believe that it is appropriate to use the capacitance solution to evaluate the approach.