A Precorrected-FFT Method for Electrostatic Analysis of Complicated 3-D Structures

Joel R. Phillips and Jacob K. White, *Associate Member, IEEE*

***Abstract*—In this paper we present a new algorithm for accelerating the potential calculation which occurs in the inner loop of iterative algorithms for solving electromagnetic boundary integral equations. Such integral equations arise, for example, in the extraction of coupling capacitances in three-dimensional (3-D) geometries. We present extensive experimental comparisons with the capacitance extraction code FASTCAP [1] and demonstrate that, for a wide variety of geometries commonly encountered in integrated circuit packaging, on-chip interconnect and micro-electro-mechanical systems, the new "precorrected-FFT" algorithm is superior to the fast multipole algorithm used in FASTCAP in terms of execution time and memory use. At engineering accuracies, in terms of a speed-memory product, the new algorithm can be superior to the fast multipole based schemes by more than an order of magnitude.**

***Index Terms*— Capacitance extraction, dense matrix algebra, electrostatic analysis, fast Fourier transform, integral equation.**

I. Introduction

APPLICATIONS as diverse as analysis of signal integrity in integrated circuit interconnect, characterization of electrical packaging, and design of microelectromechanical systems [2] require accurate electrostatic analysis of complicated three-dimensional (3-D) structures. Recent work on techniques for rapid electrostatic analysis for capacitance extraction have been based on random-walk methods [3], partitioning heuristics combined with techniques from matrix extension theory [4], [5], finite-difference [6], [7] or finite-element methods [8], [9], or method-of-moments [10] techniques.

Algorithms using method of moments [10] or weighted residuals [11], [12] based discretizations of integral equation formulations, also known as boundary-element methods [13], are commonly used to perform electrostatic analyzes, but such approaches generate dense matrix problems which are computationally expensive to solve, and this limits the complexity of problems which can be analyzed. Multipole-accelerated iterative methods [1], [14]–[16] have recently been used to reduce the computational cost of boundary-element based methods, but these techniques are still computationally expensive and, of more immediate consequence, memory exhausting.

In [1], a fast algorithm for electrostatic analysis of 3-D structures was presented. The computation time for the algorithm was shown to grow nearly as mn, where n is the number of panels used to discretize the conductor surfaces, and m is the number of conductors. The algorithm of [1] was based on the hierarchical multipole algorithm [15], which can perform the dense matrix-vector product associated with discretized potential integral equations in order-n ($O(n)$) time and memory. In this paper, we describe a precorrected-FFT approach which can replace the fast multipole algorithm for accelerating the Coulomb potential calculation needed to perform the matrix-vector product. The central idea of the algorithm is to represent the long-range part of the Coulomb potential by point charges lying on a uniform grid, rather than by series expansions as in fast multipole algorithms [15]. This grid representation allows the fast Fourier transform (FFT) [17]–[19] to be used to efficiently perform potential computations. Because only the long-range part of the potential is represented by the grid, the grid is not coupled to the underlying discretization of the structure. Decoupling the long and short range parts of the potentials allows the algorithm to solve problems which may be discretized in a very irregular fashion in nearly optimal time.

Numerous algorithms exist for the "n-body problem" of evaluating the potential of a set of charges at all the other charge points, such as the "particle-mesh" methods (see [20] for extensive references), the fast multipole method (FMM) [15], and multigrid methods [21]. The various algorithms differ in the way the long range potential is approximated and in the way local interactions are treated. We have attempted to develop an algorithm which, like particle-mesh methods, exploits the availability of efficient discrete Fourier transform implementations while at the same time preserves the higher accuracy of the multipole-based schemes, but is also (like multigrid schemes) easily adapted to a broad class of kernels. In addition, the algorithm is, in the way local interactions are treated, particularly adapted to boundary-integral solvers.

The precorrected-FFT method, described below, is at best an $O(n \log n)$ algorithm. It is possible to construct geometries for which the performance of the precorrected-FFT algorithm is inferior to the fast multipole methods, but we demonstrate that for many structures associated with packaging, on-chip interconnect, and micro-electro-mechanical systems,

Manuscript received February 15, 1996; revised April 18, 1997. This work was supported by the Defense Advanced Research Project Agency Contract J-FBI-95-215 and Subgrant 536040-52223, by the National Science Foundation Contract MIP-8858764 A02, by an NDSEG Fellowship, and by grants from Digital Equipment Corporation, IBM, and the Consortium for Superconducting Electronics. This paper was recommended by Associate Editor K. Mayaram.

J. R. Phillips was with the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA. He is now with Cadence Design Systems, San Jose, CA 95134 USA (e-mail: jrp@cadence.com).

J. K. White is with the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Publisher Item Identifier S 0278-0070(97)09247-6.

precorrected-FFT methods are faster and use substantially less memory.

The outline of the paper is as follows. The boundary-element formulation and a standard iterative algorithm for solving the generated matrix problem are briefly reviewed in Section II. The precorrected-FFT method is described in Section III and some analysis of the algorithm performed in Section IV. Simple examples are examined in Section V to show various aspects of the algorithm. In Section VI, our precorrected-FFT method is compared to FASTCAP on a variety of realistic examples, and is shown to be faster and use substantially less memory. Finally, in Section VII, we discuss some possible extensions of the algorithm and strategies to reduce the computational complexity for very inhomogeneous problems.

II. Problem Formulation

The capacitance of an m-conductor geometry can then be summarized by an $m \times m$ symmetric matrix C, where the entry C_{ij} represents capacitive coupling between conductors j and i. To determine the jth column of the capacitance matrix, one need only solve for the surface charges on each conductor produced by raising conductor j to one volt while grounding the rest. If the conductors are embedded in a homogeneous dielectric, these m potential problems can be solved using an equivalent free space formulation in which the conductor-dielectric interfaces are replaced by a charge layer of density σ [22], [23]. The charge layer in the free space problem will be the induced charge in the original problem if σ satisfies the integral equation

$$\psi(x) = \int_{\text{surfaces}} \sigma(x') \frac{1}{4\pi\epsilon \|x - x'\|} da', \quad x \in \text{surfaces} \tag{1}$$

where $\psi(x)$ is the known conductor surface potential, da' is the differential conductor surface area, $x, x' \in \mathbf{R}^3, \epsilon$ is the dielectric constant, and $\|x\|$ is the usual Euclidean length of x. This approach may be extended to the case of piecewise-constant dielectrics [23].

A standard approach [10] to numerically solving (1) for σ is to use a piecewise constant collocation scheme. That is, the conductor surfaces are broken into n small panels, and it is assumed that on each panel i, a charge, q_i, is uniformly distributed, as in Fig. 1. Then for each panel, an equation is written which relates the known potential at the center of that ith panel, denoted $\overline{f}_i$, to the sum of the contributions to that potential from the n charge distributions on all n panels [23]. The result is a dense linear system

$$Pq = \overline{f} \tag{2}$$

where $P \in \mathbf{R}^{n \times n}$, $q \in \mathbf{R}^n$ is the vector of panel charges, $\overline{f} \in \mathbf{R}^n$ is the vector of known panel potentials, and

$$P_{ij} = \frac{1}{a_j} \int_{\text{panel}_j} \frac{1}{4\pi\epsilon \|x_i - x'\|} da' \tag{3}$$

where the collocation point x_i is the center of the ith panel and a_j is the area of the jth panel. The dense linear system

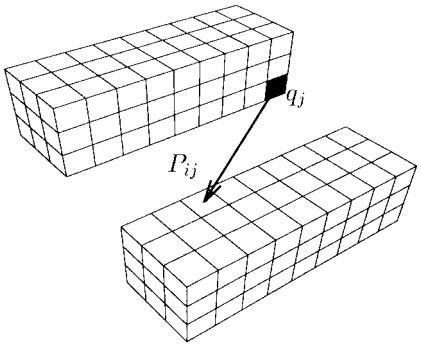


Fig. 1. Piecewise-constant collocation discretization of two conductors. Conductor surfaces are discretized into panels which support a constant charge density.

of (2) can be solved to compute panel charges from a given set of panel potentials and the capacitances can be derived by summing the panel charges.

The direct approach of solving (2) via Gaussian elimination, which requires $O(n^3)$ operations and $O(n^2)$ storage, becomes computationally intractable if the number of panels exceeds several hundred.

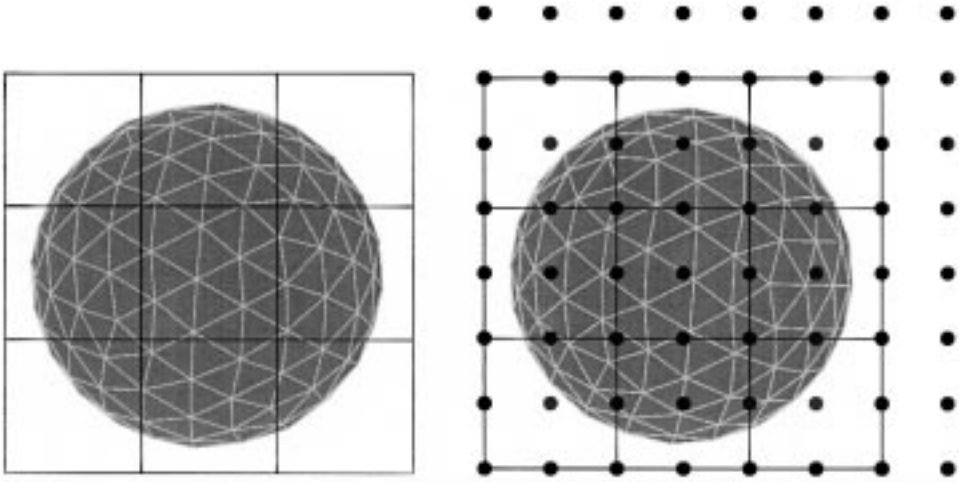
III. The Precorrected-FFT Approach

If, instead of Gaussian elimination, an iterative algorithm such as GMRES [24] is used to solve (2), then each iteration of GMRES will cost n^2 operations. This is because the matrix in (2) is dense, and therefore evaluating candidate solution vectors involves a dense matrix-vector multiply. Several sparsification techniques for P are based on the idea of directly computing only those portions of Pq associated with interactions between panels which are close to each other. The rest of Pq is then somehow approximated to accelerate the computation [15], [5], [21].

To develop a faster approach to computing the matrix-vector product, consider the parallelepiped which contains a 3-D problem after it has been discretized into n panels. The parallelepiped containing the problem could be subdivided into an $k \times l \times m$ array of small cubes so that each small cube contains only a few panels. Fig. 2(a) shows a discretized sphere, with the associated space subdivided into a $3 \times 3 \times 3$ array of cubes. We refer to these small cubes as *cells*.

A possible approach to computing distant interactions is to exploit the fact that potentials at evaluation points distant from a cell can be accurately computed by representing the given cell's charge distribution using a small number of weighted point charges. If the point charges all lie on a uniform grid, for example at the cell vertices, then the computation of the potential at the grid points due to the grid charges is a discrete convolution which can be performed using the FFT. Fig. 2(b) shows a possible set of grid charges for the cell subdivisions shown in Fig. 2(a). Thus, a four step method for approximating Pq is:

1) project the panel charges onto a uniform grid of point charges;
2) compute the grid potentials due to grid charges using an FFT;



(a) (b)

Fig. 2. (a) Side view of a sphere discretized into 320 panels, with spatial decomposition into a $3 \times 3 \times 3$ array of cells. (b) Superimposed grid charges corresponding to the cell decomposition of (a), with $p = 3$. In each cell, a $3 \times 3 \times 3$ array of grid charges is used to represent the long range potential of the charged panels in the cell. Some of the grid charges are shared among cells. Note that the grid is "coarser" than the triangular panels used to discretize the sphere. The grid extends outside the problem domain because the number of grid points is required to be a factor of two.

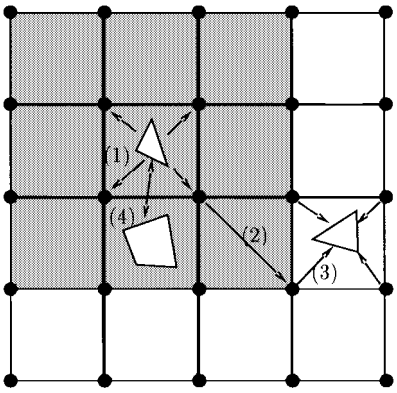


Fig. 3. A 2-D pictorial representation of the four steps of the precorrected-FFT algorithm. Interactions with nearby panels (in the grey area) are computed directly, interactions between distant panels are computed using the grid.

3) interpolate the grid potentials onto the panels;
4) directly compute nearby interactions.

This process is summarized in Fig. 3. We emphasize that the grid of point charges is introduced purely as a computational aid, it is not related to the underlying discretization of the conductors.

A. Notation

Given a set of M cells which contain the set of n panels and define the $\hat{n}$ grid points, we now describe how to compute the vector of potentials $\psi \in \mathbf{R}^n$ from the vector of panel charges $q \in \mathbf{R}^n$. $\psi_G \in \mathbf{R}^n$ will denote the contribution of the grid charges to the potentials on the n panel charges. $n(k)$ denotes the number of panels in a cell k, $q(k) \in \mathbf{R}^{n(k)}$ the restriction of the charge vector q to the indices whose corresponding panels lie in cell k and $\psi(k) \in \mathbf{R}^{n(k)}$ denotes the similar restriction of the potential vector. The variable p denotes the order of grid approximation. $\hat{q} \in \mathbf{R}^{\hat{n}}$ is the vector of grid charges, $\hat{\psi}$ the vector of grid potentials, and $\hat{q}(k) \in \mathbf{R}^{p^3}$, $\hat{\psi}(k) \in \mathbf{R}^{p^3}$ denote the restriction of $\hat{q}$ and $\hat{\psi}$, respectively, to grid points of cell k. We define $N(k)$ to be the indexes of the set of cells which are "near" cell k. W, V: $\mathbf{R}^n \rightarrow \mathbf{R}^{\hat{n}}$, yet to be defined, will refer to linear operators which project n uniformly distributed panel charges to the $\hat{n}$ grid points, and the linear operator H: $\mathbf{R}^{\hat{n}} \rightarrow \mathbf{R}^{\hat{n}}$ gives grid potentials in terms of grid charges, i.e., $\hat{\psi} = H\hat{q}$. $W(k, j)$: $\mathbf{R} \rightarrow \mathbf{R}^{p^3}$ is the nonzero part of W corresponding to charge j in cell k, $1 \leq j \leq n(k)$; $V(k, j)$ is the similar part of V, and $H(k, l)$: $\mathbf{R}^{p^3} \rightarrow \mathbf{R}^{p^3}$ is the block of H which maps grid charges $\hat{q}(l)$ of cell l to grid potentials of cell k, $\hat{\psi}(k) = H(k, l)\hat{q}(l)$. A subscript indicates an index into a matrix or vector, e.g., $q_j(k)$ is the jth entry of vector $q(k)$.

B. Projecting Onto a Grid

The first step in the description of the algorithm is to describe the construction of the grid projection operator W. For panel charges contained within a given cell, the potentials at evaluation points distant from the given cell can be accurately computed by representing the given cell's charge distribution with a small number of appropriately weighted point charges on a uniform grid throughout the given cell's volume. Fig. 2(b) shows the grid imposed on the cell structure of Fig. 2(a) when a $3 \times 3 \times 3$ array of grid charges is used to represent the charge in each cell. Note that because the grid is only used to represent the long range part of the panel potentials, the grid may be significantly coarser than the actual problem discretization.

To motivate a scheme for representing panel charges with weighted point charges lying on a grid, consider a charge distribution $\rho(x)$ contained entirely within some small volume B. The potential outside B due to ρ can be determined from the knowledge of the potential on a surface S surrounding B [25]. For example, suppose ρ is contained within a sphere S of radius a, as in Fig. 4. For all (r, θ, ϕ) with $r > a$, the

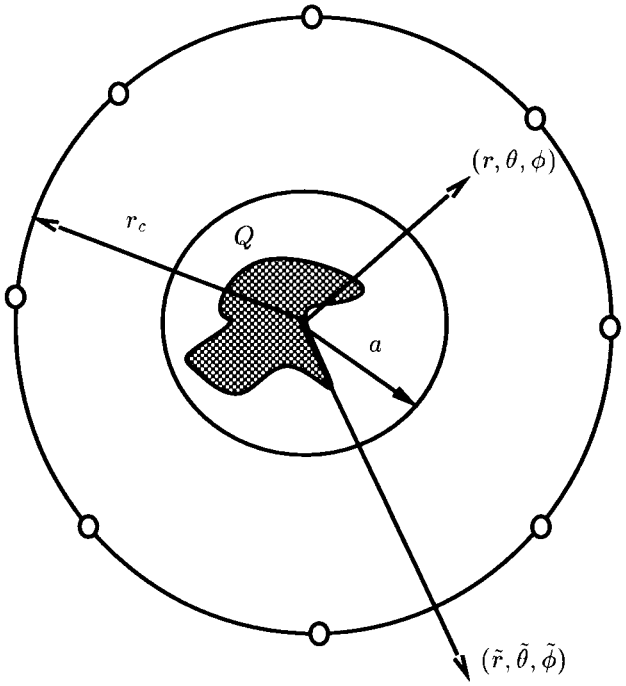


Fig. 4. Potentials $\psi(r < r_c, \theta, \phi)$ or $\psi(\bar{r} > r_c, \bar{\theta}, \bar{\phi})$ for $r, \bar{r} > a$ may be obtained from the potential at $r = r_c$.

potential ϕ can be written as a multipole expansion series

$$\phi(r, \theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{c_{lm}}{r^{l+1}} Y_{lm}(\theta, \phi) \tag{4}$$

where the Y_{lm} are spherical harmonics and the c_{lm} coefficients of expansion [25]. Since the spherical harmonics are orthogonal on a sphere, if the potential is known on any sphere S of radius $r_c > a$, the multipole moments c_{lm} can be computed as

$$c_{lm} = r_c^{l+1} \int_S d\Omega Y_{lm}^* \phi(r_c, \theta, \phi). \tag{5}$$

The above observation suggests a scheme for computing the grid charges used to represent charge in a given cell k. Suppose a $p \times p \times p$ array of grid charges is used to represent the charge in a cell. First, N_c test points are selected on the surface of a sphere of radius r_c whose center is coincident with the center of the given cell. Then, potentials due to the p^3 grid charges are forced to match the potential due to the cell's actual charge distribution (say m panel charges) at the test points, i.e.,

$$P^{gt} \hat{q}(k) = P^{qt} q(k) \tag{6}$$

where $P^{gt} \in \mathbf{R}^{N_c \times p^3}$ is the mapping between grid charges and test point potentials, given by

$$P_{i,j}^{gt} = \frac{1}{\|x_i^t - \hat{x}_j\|}. \tag{7}$$

Here x_i^t and $\hat{x}_j$ are the positions of the ith test point and the jth grid point, respectively. By construction, the relative positions of the grid charges and the test points are identical for each cell k, and therefore P^{gt} is the same for each cell.

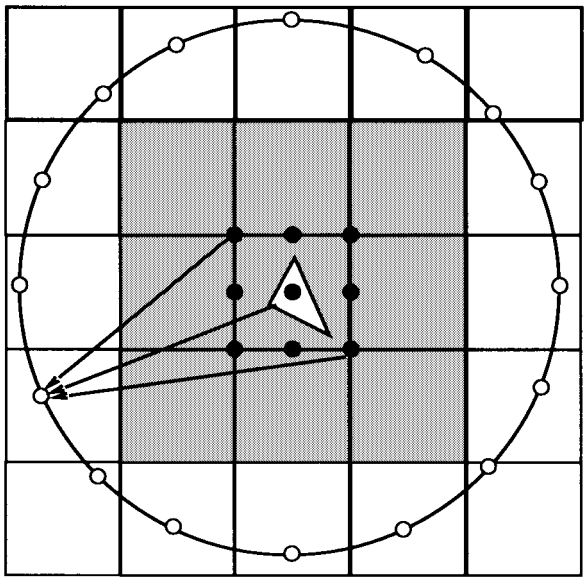


Fig. 5. Two-dimensional pictorial representation of the grid projection scheme. The black points (at $\hat{x}$) represent the grid charges ($\hat{q}$) being used to represent the triangular panel's charge density σ. The white points are the points x^t where the potential due to the black point charges and the potential due to the triangular panel's charge density are forced to match. The grid charges approximate the panel potential outside the gray region.

$P^{qt} \in \mathbf{R}^{N_c \times m}$ is the mapping between panel charges and test point potentials and is given by

$$P_{i,j}^{qt} = \int_{panel\, j} \frac{1}{\|x_i^t - x'\|} da'. \tag{8}$$

Since the collocation (6) is linear in the panel and grid charge distributions, the contribution of the jth panel in cell k to $\hat{q}(k)$ can be represented by a column vector $W(k, j)$. $W(k, j)$ is given by

$$W(k, j) = [P^{gt}]^{\dagger} P^{qt,\, j} \tag{9}$$

where $P^{qt,\, j}$ denotes the jth column of P^{qt} and $[P^{gt}]^{\dagger}$ indicates the generalized Moore–Penrose (or pseudo-) inverse [26] of $P^{gt\dagger}$. The computation of $[P^{gt}]^{\dagger}$ is done using the singular value decomposition. Since this matrix is small and is the same for each cell, the relative computational cost of performing the singular value decomposition is insignificant.

For any panel charge j in cell k, this projection operation generates a subset of the grid charges $\hat{q}(k)$. The contribution to $\hat{q}(k)$ from the charges in cell k is generated by summing over all the charges in the cell. Note that panel charges outside cell k may contribute to some of the elements of $q(k)$ in the case of shared grid charges. The grid projection scheme is summarized in Fig. 5. For an alternative approach, based on matching multipole expansion coefficients directly, see [27]. A simpler approach based on polynomial interpolation may be found in [28].

The accuracy of the above projection scheme hinges on the proper selection of the test points x^t. From (5), we expect high accuracy if the test points are chosen to be abscissas of a high-order quadrature rule [29]. It can be shown that the error in potential due to the grid-charge approximation of a charge distribution contained within a sphere of radius a, at a distance

r from the center of the distribution, is of order $(a/r)^{(M+1)/2}$ if the test points are chosen to be the nodes of a quadrature rule accurate to order M [30].

C. Computing Grid Potentials

Once the charge has been projected to a grid, the operation H, computing the potentials at the grid points due to the grid charges, is a 3-D convolution. We denote this as

$$\hat{\psi}(i,j,k) = H\hat{q} \equiv \sum_{i',j',k'} h(i-i', j-j', k-k')\hat{q}(i',j',k') \tag{10}$$

where i,j,k and i',j',k' are triplets specifying the grid points and $h(i-i', j-j', k-k')$ is the inverse distance between grid points i,j,k and i',j',k'. As will be made clear below, h(0, 0, 0) can be arbitrarily defined, and is set to zero. The above convolution can be rapidly computed by using the FFT. In practice, each convolution requires one forward and one inverse 3-D FFT. The discrete Fourier transform of the kernel matrix H, denoted $\tilde{\mathbf{H}}$, need be computed only once.

An efficient FFT implementation is central to the performance of the precorrected-FFT algorithm. The FFT is a very well-studied algorithm and many possible implementations exist. Most FFT implementations have a fairly regular nature, thus very efficient optimized code can be developed. Also, the structure of the data in a multidimensional convolution can be exploited for additional performance gains. For example, the use of the FFT to perform a linear multidimensional convolution involves embedding the data ($\hat{q}$) to be transformed into a larger data space, much of which is zero. The fact that much of the transformed data is zero can be exploited to yield a more efficient transform. In comparison, achieving optimal machine performance with fast multipole algorithms is more difficult, due to the less regular nature of the algorithms.

D. Interpolating Grid Potentials

Once the grid potentials have been computed, they must be interpolated to the panels in each cell. This process is essentially the same as the problem of representing charge on the grid, as can be seen from the following result [28].

Theorem 1: Given $\tilde{V} \in \mathbf{R}^{m\times 1}$ is an operator which projects charge onto a grid of m points, then $\tilde{V}^T$ may be interpreted as an operator which interpolates potential at grid points onto charge coordinates; conversely, given $\tilde{V}^T \in \mathbf{R}^{1\times m}$ is an operator which interpolates potential at m grid points onto charge coordinates, $\tilde{V}$ may be interpreted as an operator which projects charge onto the grid coordinates. In either case, $\tilde{V}$ and $\tilde{V}^T$ have comparable accuracy.

Proof: Let $G(x,y)$ be the Green function for a source at y, evaluated at x. Suppose that a unit charge at the point x_0 is represented by the vector of grid charges $\hat{q}$. The approximate potential $\Psi'_x(y)$ at a point y_0 is given by

$$\Psi'_x(y_0) = \sum_i G(y_0, \hat{x}_i)\hat{q} \equiv g^T\hat{q}$$

where $\hat{x}_i$ is the position of the ith grid charge and $g \in \mathbf{R}^m$, $g_i = G(y_0, \hat{x}_i)$. Conversely, suppose there is a unit charge at y_0 and the potential $\Psi_y(x_0)$ at x_0 is to be computed by interpolating potentials produced by this unit charge at the grid points $\hat{x}_i$. Then, if $\tilde{V}^T$ is the interpolation operator

$$\Psi'_y(x_0) = \sum_i \tilde{V}^T(x_0, \hat{x}_i)G(\hat{x}_i, y_0).$$

For a symmetric Green function, $\Psi_y(x_0) = G(x_0,y_0) = G(y_0,x_0) = \Psi_x(y_0)$, and

$$\sum_i \tilde{V}^T(x_0, \hat{x}_i)G(\hat{x}_i, y_0) = \tilde{V}^T g$$

so that

$$\Psi'_y(x_0) - \Psi_y(x_0) = \tilde{V}^T g - \Psi_y(x_0) = g^T\tilde{V} - \Psi_x(y_0).$$

Now suppose $\hat{q}$ is assigned the value $\tilde{V}$ in order to represent the unit point charge at x_0. Then

$$\begin{aligned}\Psi'_x(y_0) - \Psi_x(y_0) &= (g^T q_g) - \Psi_x(y_0)\\ &= g^T\tilde{V} - \Psi_x(y_0)\\ &= \Psi'_y(x_0) - \Psi_y(x_0).\end{aligned}$$ ■

When a collocation scheme is used to discretize the integral equation, the operator which interpolates potential at grid points in cell k to a charge j also in cell k is not $[W(k,j)]^T$ defined in (9). Instead, the projection operator $V(k,j)$ for a point charge located at the collocation point is computed which gives the interpolation operator $[V(k,j)]^T$. However, if a Galerkin scheme is used in the discretization then the interpolation operator is $[W(k,j)]^T$.

Thus, projection, followed by convolution, followed by interpolation gives the grid-charge approximation ψ_G to the potentials which can be represented as

$$\psi_G = V^T H W q. \tag{11}$$

If Galerkin methods are used, (11) becomes

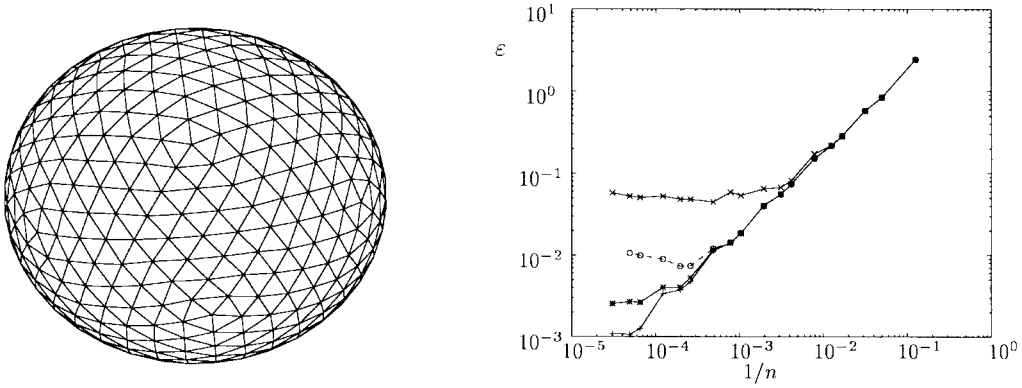
$$\psi_G = W^T H W q \tag{12}$$

and therefore the precorrected-FFT method preserves the symmetry of the Galerkin discretization for free space problems.

E. Precorrecting

The difficulty with the above three steps is that the calculations using the FFT on the grid do not accurately approximate the nearby interactions. In $\hat{\psi}$ of (11), the portions of P_q associated with neighboring cell interactions have already been computed, though this close interaction has been poorly approximated in the projection/interpolation. A more accurate calculation of interactions between nearby panels is needed, but it is also necessary to remove or avoid the inaccurate contribution from the use of the grid. This is a general difficulty with grid-based potential calculation methods and a variety of correction methods have been proposed [20], [28], [31] the details of which usually depend on the problem being solved, the interpolation scheme, and the nature of the grid solver.

Because our algorithm works directly with the Green function, and because the iterative solver requires that many



(a) (b)

Fig. 6. (a) A sphere discretized into 960 panels. The discretization is refined by subdividing the spherical triangle defined by the panel vertices into four triangular panels, whose vertices are the midpoints of the edges of the original spherical triangle. (b) Lines show integrated charge error for the sphere with Dirichlet condition of (17). Solid line shows errors for grid code, (x) $p = 2$ (*) $p = 3$ (+) $p = 4$. Dashed line connecting (o) shows error for $l = 2$ multipole scheme.

potential evaluations are performed for a given panel configuration, it is possible to treat nearby panel interactions exactly, without sacrificing algorithmic efficiency. We accurately treat interactions between panels close together by modifying the way nearby interactions are computed, a step we refer to as precorrection.

In particular, denote as $P(k,l)$ the portion of P associated with the interaction between neighboring cells k and l, $V(k)$ and $W(l)$ the matrices formed from the columns $V(k,j)$ and $W(k,j)$, respectively, and denote $\psi_{(k|l)}$ as the panel potentials in cell k due to the charges q_l in cell l. Then

$$\psi_{G,(k|l)} = V(k)^T H(k,l) W(l) q_l \tag{13}$$

is the grid-approximation to $\psi_{(k|l)}$, which is inaccurate. Subtracting this approximation and then adding the correct contribution

$$\psi_{(k|l)} = \psi_{G,(k|l)} + (P_{k,l} - V(k)^T H_{k,l} W(l)) q_l \tag{14}$$

produces the accurate result $P(k,l)q_l$.

This may be efficiently accomplished by defining

$$\tilde{P}(k,l) \equiv P(k,l) - V(k)^T H(k,l) W(l) \tag{15}$$

to be the "precorrected" direct interaction operator. When used in conjunction with the grid charge representation $\tilde{P}(k,l)$ results in exact calculation of the interactions of panels which are close. Assuming that the Pq product will be computed many times in the inner loop of an iterative algorithm, $\tilde{P}$ will be expensive to initially compute, but will cost no more to subsequently apply than P.

F. Complete Algorithm

Combining the above steps leads to the precorrected-FFT algorithm, which rapidly computes the Pq dense matrix-vector product. Using the above notation, the algorithm can be described as two steps. The first step is to compute

$$\psi_G = V^T H W q. \tag{16}$$

W and V are sparse interpolation operators and H can be represented in a sparse manner via the FFT. The second step is to add in the corrected direct interactions, to obtain the panel potentials $\psi(k)$ for each cell k

$$\psi(k) = \psi_G(k) + \sum_{l \in N(k)} \tilde{P}(k,l) q_l. \tag{17}$$

Because for each k, $N(k)$ is a small set and each matrix $\tilde{P}(k,l)$ is also small, this second step is also a sparse operation. The complete algorithm follows in pseudocode form:

Precorrected-FFT Algorithm to Compute Pq

```
/* Projection Step */
  Set q̂ = 0
  For each cell k = 1 to M {
    For each panel j in cell k, j = 1 to n(k) {
      Add q̂(k) = q̂(k) + W(k, j)q_j(k)
    }
  }
/* Convolution Step */
  Compute Q̂ = FFT(q̂)
  Compute Ψ̂ = H̃Q̂
  Compute ψ̂ = FFT^{-1}(Ψ̂)
/* Interpolation Step */
  Set ψ = 0
  For each cell k = 1 to M {
    For each panel j in cell k, j = 1 to n(k) {
      Add ψ_j(k) = ψ_j(k) + [V(k, j)]^T ψ̂_j(k)
    }
  }
/* Nearby Interactions */
  For each cell k = 1 to M {
    For each cell l in N(k)
      ψ(k) = ψ(k) + P̃(k, l)q(l)
    }
  }
```

Thus, the effect of this algorithm is to replace the operation

$$\psi \leftarrow Pq$$

where P is a dense matrix, with the operation

$$\psi \leftarrow [\tilde{P} + V^T H W] q$$

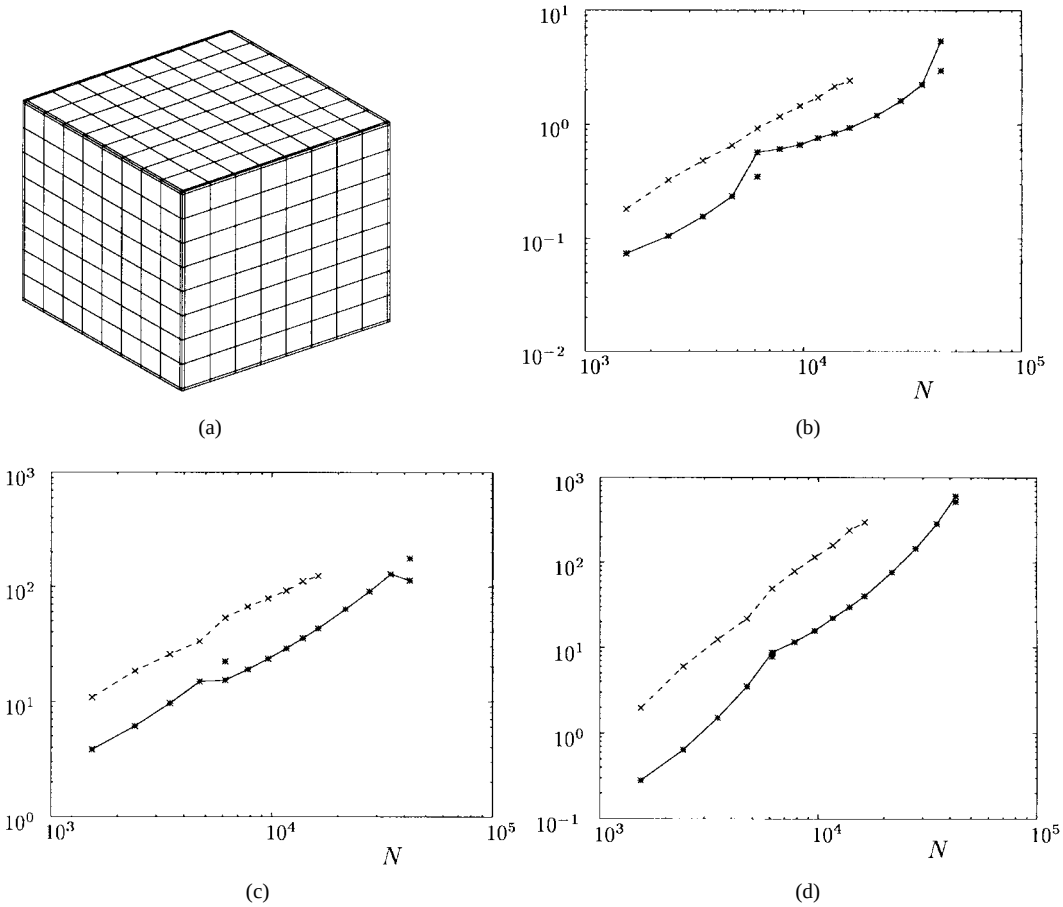


Fig. 7. The cube example. (a) Discretization of the cube. (b) CPU time, in seconds, needed for the fast multipole (dashed line connecting "×") and precorrected-FFT algorithms ("*") to compute a matrix-vector product. For the precorrected-FFT algorithm, different results are possible depending on whether speed or memory usage is to be optimized. The solid line connects runs with grid sizes chosen to minimize memory use. Note the speed-memory product is fairly independent of grid size. (c) Memory, in Mb, needed by the fast multipole and precorrected-FFT algorithms. (d) Product of (b) and (c).

where all the matrices $\tilde{P}, V, H$, and W possess sparse representations.

G. Grid Selection

Before the algorithm has been completely specified, it necessary to specify how panels are selected for inclusion in direct interaction regions and how the grid size is selected. That is, for each cell k the set $N(k)$ must be specified. To insure that interactions between panels which are close together are treated accurately, at a minimum it is necessary to compute interactions between panels in cells which are near-neighbors of each other via direct products. The near-neighbors of a cell β are defined to be all the cells which have a vertex in common with cell β (thus a cell is a near-neighbor of itself). We have included only near-neighbor interactions in the computational experiments of Sections V and VI.

The worst case accuracy of the grid representation is a function of the ratio of the cell radius to the radius of the direct interaction region [30]. Thus, once the direct-interaction region has been specified to be near-neighbor cells, the selection of the cell size, and hence the grid spacing is purely a matter of computational efficiency. The cost of direct interactions will decrease monotonically as the cells are made smaller, but the number of grid points will increase, so the cost of the FFT will increase monotonically. This implies that the total cost of the algorithm will have an easily determined global minimum for some grid spacing. For a given grid spacing and panel configuration, the memory and computation time needed by the precorrected-FFT algorithm can be estimated cheaply, so the optimal grid spacing can be obtained by starting with a small number of grid points and increasing the number until a minimum CPU or memory estimate, as appropriate, is reached. In addition, we have generally required that the number of grid points be a factor of two, in order to exploit the most efficient FFT implementations.

It is interesting that the optimal grid size may occasionally be such that the number of grid charges, $\hat{n}$, is *larger* than the original number of panel charges n. This may be the case even when the grid spacing is larger than the underlying panel sizes, that is, when the grid is "coarser" than the panel discretization. Such a case may occur, for example, for a finely discretized cube surface, where the grid must fill the 3-D space of the cube's interior. However, the overall algorithm may still be quite effective, since the cost of the FFT is $O(\hat{n}\log\hat{n})$, with a constant factor of $O(10)$. Thus if $\hat{n} \simeq n$ and n is large, the cost of the FFT is less than that of the direct product by a factor of nearly $O(n)$, and so the algorithm may have $\hat{n} > n$

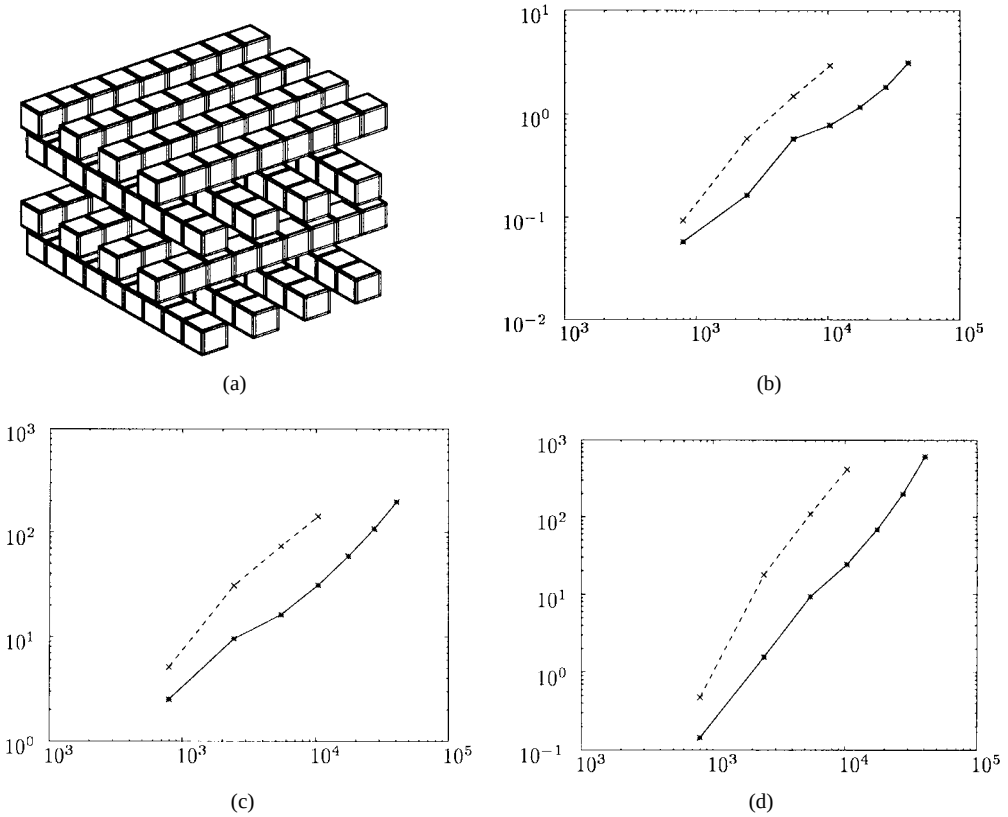


Fig. 8. The bus crossing example. (a) Larger problems are generated by adding more bus lines. (b) CPU time, in seconds, needed for the fast multipole (dashed line connecting "×") and precorrected-FFT algorithms (solid line connecting "*") to compute a matrix-vector product. (c) Memory, in Mb, needed by the fast multipole and precorrected-FFT algorithms. (d) Product of (b) and (c).

by a fairly significant factor and still possess an advantage over the direct computation.

IV. Analysis of the Precorrected-FFT Algorithm

In this section, we analyze the complexity of the precorrected-FFT algorithm and give some comparisons with other approaches.

A. Comparison to Fast Multipole Algorithms

First we compare the efficiency of the grid representation used in the precorrected-FFT algorithm to the multipole expansions used in the fast multipole method. Both the fast multipole algorithm and the precorrected-FFT algorithm obtain efficiency by representing the long-range part of the potential of a group of charges by an expression which can be used at multiple evaluation points, but the algorithms differ in the way they cluster sets of charges together to form single expressions.

Again consider subdividing the parallelepiped containing the entire 3-D problem domain into a $k \times l \times m$ array of cells. Then, the collocation approach above can be used to generate point charge approximations for charge distributions in every cell, effectively projecting the charge density onto a 3-D grid. For example, if the representative point charges are placed at the cell vertices, then the panel charge distribution will be projected to a $(k+1)\times(l+1)\times(m+1)$ uniform grid. Fast multipole algorithms also effectively create a uniform grid by constructing multipole expansions at the center of each cell, but due to sharing, the point charge approach can be more efficient. For example, when representing the potential of a panel by charges at the cell vertices, there are eight free coefficients which may be varied to obtain an optimal representation, and there will be $(k+1)(l+1)(m+1)$ terms in the entire domain. On average, there is only one grid-charge per cubic cell, since a point charge at a cell vertex is used to represent charge in the eight cells which share that vertex. By contrast, as no sharing occurs in the multipole representation, if there are q coefficients in the multipole expansion which represents the potential of the charges in the cell, the total number of terms in the domain will be $q(k+1)(l+1)(m+1)$. For an equivalent number of total terms in the domain, we expect the grid representation to be more accurate. Conversely, for roughly equivalent accuracy, we may choose $q \simeq 8$, but then the total number of multipole terms will be significantly higher than for the grid representation.

B. Performance for Homogeneous Problems

From the analysis of the preceding section, we expect the grid representation to be locally more efficient than the use of multipole expansions. However, our current implementation of the precorrected-FFT algorithm may be globally less efficient, as the grid representation is introduced throughout space, even where no panels are present. Thus, whereas for a problem containing n panels, the fast multipole algorithm can perform a potential evaluation for all of the panels in $O(n)$ operations, regardless of the panel distribution [32], no such guarantee

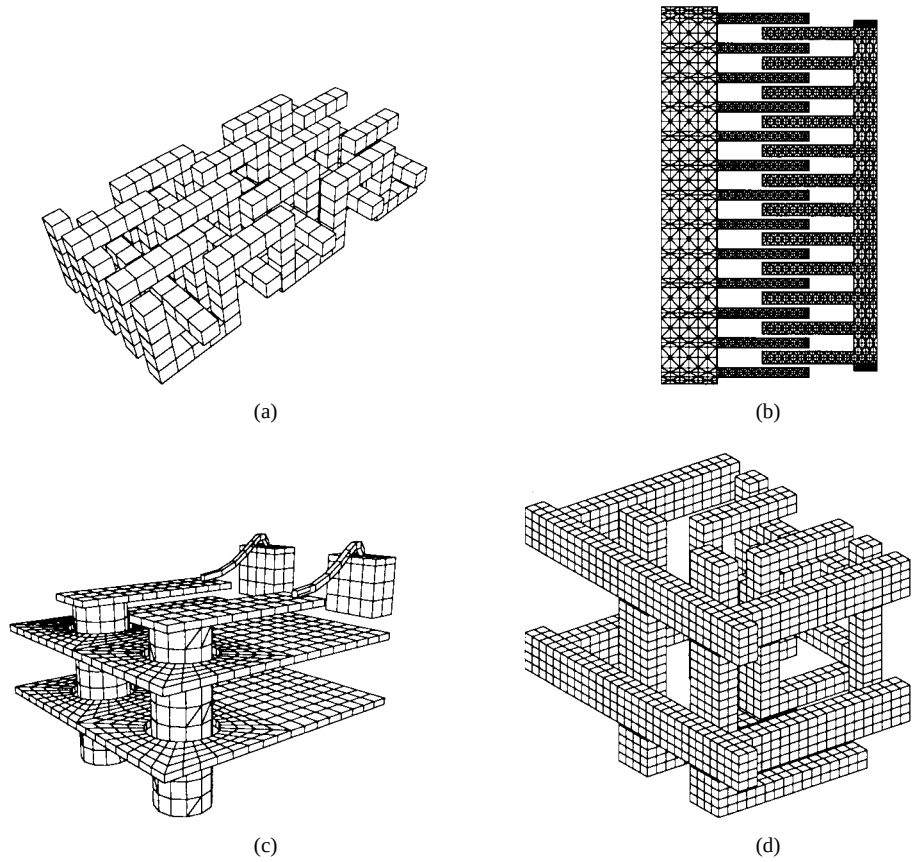


Fig. 9. Several realistic capacitance extraction problems. (a) The woven bus example (woven 5 × 5). (b) The comb drive example (comb). (c) The via example (via). (d) The SRAM example (SRAM).

is available for the precorrected-FFT algorithm. However, it is possible to establish a weaker complexity result for the precorrected-FFT method.

Theorem 2: For a homogeneous distribution of n panels and a given prescribed accuracy, the precorrected-FFT method requires $O(n \log n)$ operations to perform a potential calculation.

Proof: Given that the computational domain is a parallelepiped containing n panels, again assume space has been divided into an array of $k \times l \times m$ cells, and that the panel distribution is homogeneous on the scale of the cell size. That is, the number of panels per cell, N_c, is bounded independent of n, with klm of order-n. Finally, assume that the grid in each cell is a $p \times p \times p$ array. There are three components in the cost of the precorrected-FFT algorithm: the cost of direct interactions, the cost of grid projection and interpolation, and the cost of the FFT. The cost of the direct interactions will be $O(N_c^2 \times klm) = O(klm) = O(n)$. The cost of the grid projection will be $O(np^3) = O(n)$. Finally, the cost of the FFT will be $O(p^3 klm \log p^3 klm) = O(n \log n)$. Summing these costs results in the final complexity of $O(n \log n)$. ■

Since the grid spacing is typically less fine than the underlying surface discretization, for a typical problem the precorrected-FFT algorithm has $O(n \log n)$ complexity for problems with considerable inhomogeneity in the fine surface discretization, as long as the panel distribution is homogeneous at a very coarse level. As will be seen in Section VI, many structures arising in practice satisfy this "coarsely homogeneous" condition.

C. Comparison to Other Grid-Based Methods

In order to solve the underlying potential-theoretic problem, the precorrected-FFT algorithm introduces a uniform grid which covers the problem domain volume and so it is instructive to compare the precorrected-FFT algorithm with other methods that introduce volumetric grids.

First, most other methods which use a grid to represent the solution throughout space, such as finite-difference methods, finite-element methods, or integral equation methods which directly exploit the convolutional properties of the kernel via the FFT [33]–[35], introducing a space-filling grid which must also accurately represent the complicated problem geometry. These two conflicting requirements generally result in either restricted geometries or a very large number of unknowns that in turn limits the size of the problem that can be effectively solved.

In contrast, as shown in Fig. 2, the grid introduced by the precorrected-FFT algorithm is geometrically unrelated to the underlying surface discretization of the geometry. In general the number of panels in a surface discretization is much smaller than the number of elements in a volume representation, so we expect the precorrected-FFT algorithm to be more efficient, than, for example, finite-difference approaches.

TABLE I
STATISTICS FOR FASTCAP ORDER-2, GRID-2,3 CODES FOR $1/r$ GREEN FUNCTION. SETUP, SOLVE, AND CPU TIMES ARE IN SECONDS ON DEC AXP 3000/900, MEMORY IN MEGABYTES. m IS NUMBER OF CONDUCTORS IN PROBLEM, EACH CONDUCTOR REQUIRES A SEPARATE LINEAR SYSTEM SOLUTION

Example[m]	Panels	Code	Setup	Solve	CPU	Memory	% err/rel	% err/diag
via [4]	6120	FFT[$p=2$]	10.85	22.18	33.03	15.83	1.41	0.81
		FFT[$p=3$]	10.60	61.96	72.56	20.92	0.068	0.026
		FASTCAP	18.56	101.32	119.88	55.82	0.369	0.152
woven5x5[10]	9360	FFT[$p=2$]	5.8	125.2	131.0	20.35	7.06	1.65
		FFT[$p=3$]	32.63	282.4	315.03	49.49	1.60	0.048
		FASTCAP	37.37	656.3	693.67	103.8	16.9	0.431
cube[1]	14406	FFT[$p=2$]	4.80	8.80	13.6	25.03	0.105	0.105
		FFT[$p=3$]	14.37	9.93	24.3	36.55	0.003	0.003
		FASTCAP	34.59	28.78	63.37	115.81	0.024	0.024
bus3x6[6]	6480	FFT[$p=2$]	6.38	18.52	24.9	11.92	1.30	0.99
		FFT[$p=3$]	6.59	44.70	51.29	15.74	0.033	0.015
		FASTCAP	21.75	184.2	205.95	75.49	1.89	0.164
bus3x8[6]	11520	FFT[$p=2$]	5.67	52.1	57.77	20.22	1.10	0.416
		FFT[$p=3$]	15.2	82.86	98.06	32.64	0.021	0.021
		FASTCAP	30.52	328.38	358.9	119.9	1.96	0.177
SRAM[6]	3944	FFT[$p=2$]	3.92	11.83	15.75	7.71	0.90	0.45
		FFT[$p=3$]	8.30	28.02	36.32	16.70	0.046	0.023
		FASTCAP	11.89	81.22	93.11	38.92	0.62	0.082
comb[3]	19424	FFT[$p=2$]	28.9	91.5	120.3	50.3	2.11	1.74
		FFT[$p=3$]	23.2	315.8	339.0	71.1	0.056	0.043
		FASTCAP	70.2	399.3	469.5	211.0	0.12	0.11
woven15[30]	82080	FFT[$p=2$]	134.2	6152.8	6287.0	246.3	–	–
cube[1]	126150	FFT[$p=2$]	73.6	127.5	201.1	224.7	–	–

Additionally, most other 3-D grid-based approaches necessarily have a complexity of k^3, if k is the number of basis elements along a side. The precorrected-FFT method analyzed here uses $O(n \simeq k^2)$ basis elements in the underlying surface discretization, and the complexity is $O(n) \rightarrow O(n^{1.2}) = O(k^2) \rightarrow O(k^{2.4})$ (see Sections IV-B and V-B, and [30]). At $k = 50$ basis elements per dimension, corresponding only to a 15 000 panel problem, $k^{3.0}$ exceeds $k^{2.4}$ by more than a factor of 20.

In short, because of the decoupling of short range interactions from the long range interactions treated by the grid, the precorrected-FFT method can efficiently utilize fast potential solvers without sacrificing the ability to represent complicated surface geometries in a compact manner.

V. REFERENCE EXAMPLES

In this section, we examine a variety of simple examples to evaluate the performance of the precorrected-FFT algorithm. We start by examining the errors introduced by the grid projection method and then we examine the efficiency of the overall precorrected method.

A. Empirical Error Analysis

As described above, in the precorrected-FFT algorithm, the interaction between panels in neighboring cells is computed exactly, but more distant interactions are approximated by extrapolation, convolution, and then interpolation using the grid. To demonstrate that the errors due to using the grid are well controlled, we present an empirical error study based on an analytically solvable potential problem borrowed from [32]. If (1) is solved on a sphere with given potential

$$\psi(\theta, \phi) = -\frac{\cos\theta}{2} \tag{18}$$

the analytically computable charge distribution is

$$\sigma(\theta, \phi) = -\frac{3\cos\theta}{8\pi}. \tag{19}$$

To estimate the error introduced by the grid approximations in the precorrected-FFT method, the sphere can be discretized, as in Fig. 6, and the charges q_i on each panel i computed. The approximations introduced by the grid-charge approximation to long-range interactions will become evident as the discretization is refined, since eventually these errors will dominate over the discretization error. One relative measure of the error is

$$\varepsilon = \frac{1}{Q}\sum_i |q_i - a_i\sigma(x_i)| \tag{20}$$

where the sum runs over all panels i, x_i is the centroid of panel i, a_i the area of that panel, q_i the charge on the panel,

and Q the exact total charge on the sphere. Fig. 6(b) shows that for the low-order piecewise-constant collocation scheme, as the discretization of the sphere is refined [see Fig. 6(a)], the integrated error ε decreases proportional to $1/n$. The multipole or precorrected-FFT approximation errors are evident when ε ceases to decrease as n is increased. For example, for the $2 \times 2 \times 2$ grid-charge representation is used in each cell ($p = 2$), ε ceases to decrease below about 0.05. This indicates that the $p = 2$ grid charge scheme introduces errors into the integrated charge calculation of about 5%. Similarly, we expect the $p = 3$ scheme to be accurate to almost a tenth of a percent. We have also shown results for the $l = 2$ multipole approximation, which from this experiment we expect to be intermediate in accuracy between the grid $p = 3$ and $p = 2$ approximations.

B. Effects of Inhomogeneity

The fast multipole algorithms used in the FASTCAP program compute matrix-vector products in $O(n)$ operations regardless of the distribution of panels on the discretized surfaces [32], but this is not true of the precorrected-FFT method. As described, the use of the FFT implies that the algorithm computes matrix-vector products in at best $O(n \log n)$ operations, and attains this optimum only for fairly homogeneous distributions of panels (see Section IV-B). That is, for problems where the panels are distributed in a roughly uniform manner throughout space, the precorrected-FFT method should be efficient. In contrast, for inhomogeneous problems which consist of clusters of panels separated by large areas of open space, inefficiency may be expected. Therefore it is important to quantify the performance penalty induced in the precorrected-FFT method by problem inhomogeneity.

A simple approach to generating an example which is inhomogeneous is to refine the discretization of a cube. The cube example is intended to serve as a model for typical boundary-element discretizations of surfaces. As the discretization is refined, problems with increasing numbers of panels will be generated. The precorrected-FFT algorithm must place grid charges in the empty interior of the cube, which causes the CPU time and memory required by the algorithm to increase faster than $O(n \log n)$. As n increases, relatively more panels are near the surfaces of the cube relative to the interior, i.e., the problem inhomogeneity increases. Thus, at some large n, the fast multipole methods will be superior to the precorrected-FFT method. We wish to determine how effective the precorrected-FFT method is for reasonable size problems, and at what n it would become advantageous to use the fast-multipole methods.

Fig. 7 shows the comparison of the precorrected-FFT method at $p = 3$ to the fast-multipole based code FASTCAP, at $l = 2$, for the cube example. The discretization of the cube is refined to generate more panels, and the performance of the two codes compared as the problem size increases. Three figures are shown. Fig. 7(b) shows the time required for each code to compute a matrix-vector product, Fig. 7(c) shows the amount of memory needed by each code, and Fig. 7(d) shows a figure of merit which is the product of required memory and the time needed for a potential calculation. The product is important to consider when analyzing the precorrected-FFT method because, as is clear from the figure, speed can be traded for memory by manipulating the size of the region the grid-charge approximation covers. The CPU and memory figures for the precorrected-FFT method are observed to grow irregularly with problem size. This is because our specific implementation of the method requires the number of grid-charges along one side of the computational domain to be a power of two. The solid line in the figures shows results when the number of grid charges along a side was selected to optimize (see Section III-G) the speed-memory product, which is observed to grow smoothly. Two cases in Fig. 7(a) are evident where the code would have been considerably faster had a different number of grid-charges been used. However, as Fig. 7(b) shows, the memory required would have been greater in each case.

TABLE II
COMPARISON OF FASTCAP AND GRID CODES. FIGURES ARE RATIOS OF REQUIRED RESOURCES. "PRODUCT" IS PRODUCT OF CPU AND MEMORY FIGURE

Example	Grid Order	Setup	Solve	CPU	Memory	Product
via	$p = 2$	0.58	0.22	0.28	0.28	0.078
	$p = 3$	0.57	0.61	0.61	0.37	0.23
woven5x5	$p = 2$	0.16	0.19	0.19	0.20	0.037
	$p = 3$	0.87	0.43	0.45	0.48	0.22
cube	$p = 2$	0.14	0.31	0.21	0.22	0.046
	$p = 3$	0.42	0.34	0.38	0.32	0.12
bus3x6	$p = 2$	0.29	0.10	0.12	0.16	0.019
	$p = 3$	0.30	0.24	0.25	0.21	0.052
bus3x8	$p = 2$	0.19	0.16	0.16	0.17	0.027
	$p = 3$	0.50	0.25	0.27	0.27	0.074
SRAM	$p = 2$	0.33	0.15	0.17	0.20	0.034
	$p = 3$	0.70	0.34	0.39	0.43	0.17
comb	$p = 2$	0.41	0.23	0.26	0.24	0.062
	$p = 3$	0.33	0.80	0.72	0.34	0.24

Analysis of the trend of Fig. 7(a) reveals that the CPU time needed to solve the cube problem grows as about $O(n^{1.15})$, where n is the number of panels, faster than the $O(n)$ expected asymptotically for the fast multipole method. However, for all the problems analyzed, the precorrected-FFT method was superior in terms of CPU time and memory required. We may obtain the approximate point at which the algorithms cross over by extrapolating the data in Fig. 7(c). Assuming that the CPU time and memory of the multipole method grow as $O(n)$, and that the CPU time and memory required by the precorrected-FFT method grows as $O(n^{1.2})$ [30], then in terms of the speed-memory product the precorrected-FFT method will be superior to the fast-multipole method until n is, at least, several million panels. We estimate over 30 gigabytes of memory would be needed to solve such a problem.

The cube example demonstrates that problems exist for which the precorrected-FFT algorithm is inferior to the fast-multipole methods. This example, however, is somewhat artificial, as very large capacitance extraction problems are not

TABLE III
COMPARISON OF CAPACITANCE EXTRACTION ALGORITHMS. FIGURES IN PARENTHESES ARE ESTIMATES

Example		CPU Usage				Memory Usage	
Name	Panels[cond]	P/FFT	FASTCAP	Dir. Iterative	LU Decomp	P/FFT	Direct
via	6120[4]	1.1 min	2.0 min	(5.6 min)	(1.9 hrs)	21 Mb	(286 Mb)
woven5x5	9360[10]	5.2 min	12 min	(42 min)	(6.9 hrs)	50 Mb	(668 Mb)
woven15	82080[30]	1.7 hrs	-	(11.5 days)	(194 days)	246 Mb	(50.2 Gb)
cube	126150[1]	3.3 min	-	(8.4hrs)	(2.7 yrs)	225 Mb	(119 Gb)

usually due to very fine discretizations of a few surfaces, but rather by fixing a discretization level, and solving problems which involve increasingly more complicated structures. For a situation which better models problems from VLSI interconnect analysis, consider a bus crossing example, as in Fig. 8. In this example, a series of stacked bus problems are solved. The faces of each bus line are broken into quadrilateral sections, and the quadrilaterals discretized by division into a central panel and five edge panels. In order to generate larger and larger problems, we consider b levels of bus wires, each level having b wires.

From Fig. 8 it is clear that the computational cost of the algorithm grows nearly linearly with problem size, as predicted in Section IV-B. For the size problems considered, the precorrected-FFT method with $p = 3$ enjoys an advantage of more than a factor of three in terms of computational cost and roughly a factor of four in memory utilization over the fast multipole method using order-2 expansions. With these parameter values, however, from Fig. 6 we expect the precorrected-FFT method to be considerably more accurate.

VI. REALISTIC EXAMPLES

In this section, we present results comparing the FASTCAP program to the precorrected-FFT method for computing capacitances of several 3-D geometries. As a preconditioner has not yet been implemented in the precorrected-FFT algorithm, all comparisons were performed without FASTCAP's preconditioner. Fig. 9 shows four realistic 3-D structures: a woven bus structure, a bus crossing structure, a via structure, and part of an SRAM memory cell. We have compared the multipole-based code FASTCAP, using multipole expansion order $l = 2$, to the grid based methods with $p = 2$ and $p = 3$. To estimate the accuracy of the computed capacitances, we have compared the results to the grid-code run using $p = 6$, which we expect to introduce errors into the calculation which are very small compared to the $p = 2$, $p = 3$ grid codes or the multipole $l = 2$ code. As a check on this assumption we also performed the calculations using the fast multipole algorithm and sixth order multipole expansions. Taking the $p = 6$ capacitances to be exact, we have calculated both the maximum relative errors in the computed capacitance coefficients, as well as the maximum over all rows of the capacitance matrix of the largest error in the row as a fraction of that row's diagonal capacitance. Table I shows the computation times, memory required, and error estimates for each problem. All experiments were run on a DEC AXP3000/900, with 256 megabytes of physical memory.

The table indicates that multipole expansions of order 2 are usually enough to give relative accuracy of one percent or so in the calculated capacitances. In terms of relative errors in the computed capacitances, the $p = 2$ grid code appears to be comparable to the $l = 2$ multipole code, and somewhat inferior when the error is measured as a percentage of the diagonal capacitance. The $p = 3$ grid code clearly has uniformly superior error properties. These results are in accord with the sphere example considered previously in Fig. 6.

Table II shows explicit performance comparisons of the $l = 2$ multipole code to the $p = 2$ grid code, which has comparable accuracy, as well as to the more-accurate $p = 3$ grid codes. At $p = 3$, the precorrected-FFT method can be as much as four times faster and can use as little as one fifth the memory of FASTCAP. In terms of the speed-memory product, the grid-based code at $p = 3$ was superior by a factor ranging from four to 20. At $p = 2$, the performance advantage of the grid-code was even more significant. The CPU advantage of the method ranged from nearly four to more than eight, the memory advantage from four to six, and the product from 12 to 52.

The two final entries in Table I are worthy of note. Using the $p = 2$ grid representation, from which we expect about 2–4% accuracy, it was possible to analyze two very large problems. The first is a 15×15 wire woven bus crossing, shown in Fig. 10, which has over 80 000 panels in the discretization. The second is the cube, discretized into about 125 000 panels. The precorrected-FFT method was able to perform a single solution (one row in the capacitance matrix) in only about 3 min. More importantly, both problems could be solved in the available physical memory, which was not possible using FASTCAP.

We note that all the examples considered here generate fairly well conditioned linear systems, so no preconditioner was necessary to secure convergence in a reasonable time. However, use of a preconditioner could further reduce the computation times required for the linear system solution, and enable solution of less well conditioned problems.

Finally, we wish to emphasize that, regardless of whether the precorrected-FFT or fast-multipole based approaches are used, the advantage of the accelerated schemes over traditional algorithms is tremendous. Table III compares the computation time and memory needed by algorithms based on LU-factorization via Gaussian elimination, iterative solution using direct (explicit) matrix-vector products, and iterative solution using accelerated matrix-vector products, as described in this paper. The statistics for the direct algorithms were

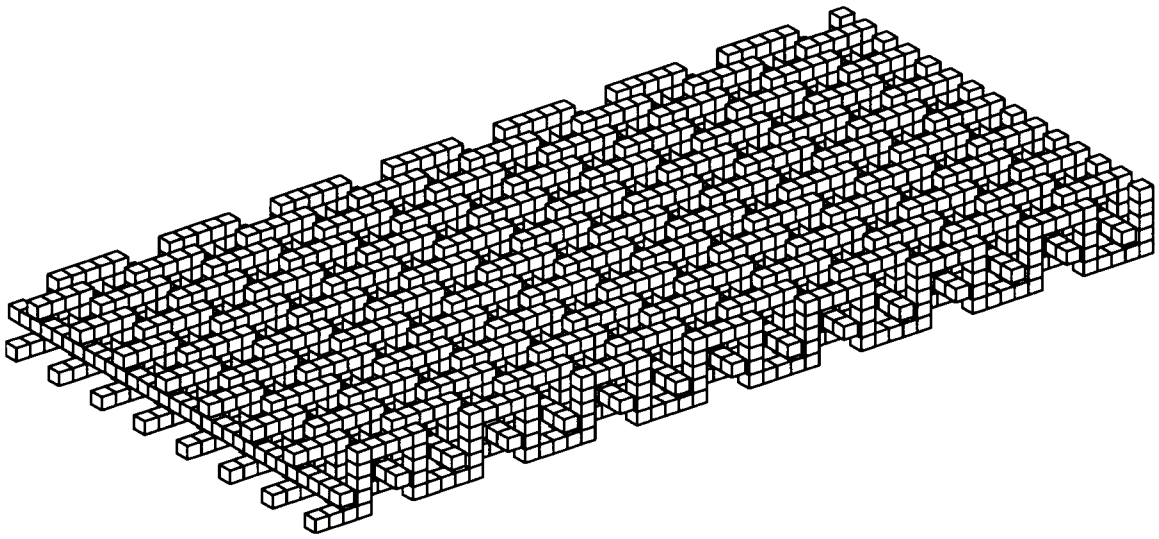


Fig. 10. The large woven bus structure. The 30 conductor structure is formed from fifteen conductors woven around fifteen straight conductors. The actual discretization is finer than shown in the above figure. There are 82 080 panels in the actual discretization.

estimated by extrapolating timings of computations performed by MATLAB, and by assuming that storage is in double precision floating point words. For the largest problems, the speedups can be two orders of magnitude over iterative solution with direct products, and nearly *six* orders of magnitude over Gaussian-elimination, with memory savings of a factor of 500.

VII. Conclusion

In this paper, we presented a precorrected-FFT approach to reducing the CPU time required to compute accurate coupling capacitances of complicated 3-D structures. As the examples above demonstrate, the precorrected-FFT method is well tuned to the problems associated with integrated circuit packaging, on-chip interconnect, and micro-electro-mechanical systems. In particular, the CPU time-memory product for the precorrected-FFT algorithm can be more than an order of magnitude smaller than that of the FASTCAP program.

A major advantage of this method is that it is based on the FFT and local interpolation operators, rather than on spherical-harmonics based shifting operators as in the fast multipole method. Thus, a range of kernels can be treated in the method while still preserving the high order of accuracy of multipole-based representations. For example, the approach can be combined with modified Green function techniques for handling ground planes, symmetry planes, or flat dielectric interfaces, with only minimal modification to the algorithm [36]. The algorithm is also capable of handling the Helmholtz kernel associated with full-wave electromagnetic analysis [30].

The major drawback of our algorithm is its poor performance on very inhomogeneous problems. The use of the FFT to evaluate the grid potentials is very inefficient in this case, since most of the data in the transform is zero. Future work will focus on more efficient algorithms to evaluate the grid potentials in the very inhomogeneous case, such as applying the algorithm of Section III recursively to obtain a multigrid-like algorithm. Such an algorithm would, in principle, inherit the efficiency of the grid-based representation discussed in this paper, while at the same time achieving the near $O(n)$ complexity of the fast multipole methods.

Acknowledgment

The authors are grateful to Dr. K. Nabors for making his FASTCAP program available. Many of the experiments were conducted by using substantial portions of the FASTCAP program. In addition, they would like to thank C. L. Berman and C. Anderson of IBM, Yorktown Heights, NY, and L. Greengard of the Courant Institute for useful discussions.

References

[1] K. Nabors and J. White, "FASTCAP: A multipole accelerated 3-D capacitance extraction program," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1447–1459, Nov. 1991.

[2] S. D. Senturia, R. M. Harris, B. P. Johnson, S. Kim, K. Nabors, M. A. Shulman, and J. K. White, "A computer-aided design system for microelectromechanical systems (MEMCAD)," *IEEE J. Microelectromech. Syst.*, vol. 1, pp. 3–13, Mar. 1992.

[3] Y. L. Le Coz and R. B. Iverson, "A stochastic algorithm for high speed capacitance extraction in integrated circuits," *Solid State Electron.*, vol. 35, no. 7, pp. 1005–1012, 1992.

[4] P. Dewilde and Z. Q. Ning, *Models for Large Integrated Circuits.* Boston, MA: Kluwer, 1990.

[5] A. J. van Genderen and N. P. Van Der Meijs, "Hierarchical extraction of 3-D interconnect capacitances in large regular VLSI structures," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1993.

[6] A. H. Zemanian, R. P. Tewarson, C. P. Ju, and J. F. Jen, "Three-dimensional capacitance computations for VLSI/ULSI interconnections," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 1319–1326, Dec. 1989.

[7] A. Seidl, H. Klose, M. Svoboda, J. Oberndorfer, and W. Rösner, "CAPCAL—A 3-D capacitance solver for support of CAD systems," *IEEE Trans. Computer-Aided Design* vol. 7, pp. 549–556, May 1988.

[8] P. E. Cottrell and E. M. Buturla, "VLSI wiring capacitance," *IBM J. Res. Develop.*, vol. 29, pp. 277–287, May 1985.

[9] T. Chou and Z. J. Cendes, "Capacitance calculation of IC packages using the finite element method and planes of symmetry," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1159–1166, Sept. 1994.

[10] R. F. Harrington, *Field Computation by Moment Methods.* New York: MacMillan, 1968.

[11] S. H. Crandall, *Engineering Analysis.* New York: McGraw-Hill, 1956.
[12] L. Collatz, *The Numerical Treatment of Differential Equations*,3rd ed. New York: Springer-Verlag, 1966.
[13] C. A. Brebbia, J. C. F. Telles, and L. C. Wrobel, *Boundary Element Techniques.* Berlin, Germany: Springer-Verlag, 1984.
[14] V. Rokhlin, "Rapid solution of integral equations of classical potential theory," *J. Comput. Phys.*, vol. 60, pp. 187–207, Sept. 15, 1985.
[15] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems.* Cambridge, MA: MIT Press, 1988.
[16] V. Jandhyala, E. Michielssen, and R. Mittra, "Multipole-accelerated capacitance computation for 3-D structures in a stratified dielectric medium using a closed form Green's function," *Int. J. Microwave Millimeter-Wave Computer-Aided Eng.*, vol. 5, no. 2, pp. 68–78, 1995.
[17] E. O. Brigham, *The Fast Fourier Transform and Its Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1988.
[18] C. F. van Loan, *Computational Frameworks for the Fast Fourier Transform.* Philadelphia, PA: SIAM, 1992.
[19] J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex Fourier series," *Math. Comp.*, vol. 19, pp. 297–301, 1965.
[20] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles.* New York: Adam Hilger, 1988.
[21] A. Brandt and A. A. Lubrecht, "Multilevel matrix multiplication and fast solution of integral equations," *J. Comput. Phys.*, vol. 90, pp. 348–370, 1990.
[22] A. E. Ruehli and P. A. Brennan, "Efficient capacitance calculations for three-dimensional multiconductor systems," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-21, pp. 76–82, Feb. 1973.
[23] S. M. Rao, T. K. Sarkar, and R. F. Harrington, "The electrostatic field of conducting bodies in multiple dielectric media," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-32, pp. 1441–1448, Nov. 1984.
[24] Y. Saad and M. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 856–869, July 1986.
[25] J. D. Jackson, *Classical Electrodynamics.* New York: Wiley, 1975.
[26] R. A. Horn and C. R. Johnson, *Matrix Analysis.* Cambridge, MA: Cambridge Univ. Press, 1985.
[27] L. Berman, "Grid-multipole calculations," *SIAM J. Sci. Stat. Comput.*, vol. 16, pp. 1082–1091, Sept. 1995.
[28] A. Brandt, "Multilevel computations of integral transforms and particle interactions with oscillatory kernals," *Comput. Phys. Commun.* vol. 65, pp. 24–38, 1991.
[29] A. D. McLaren, "Optimal numerical integration on a sphere," *Math. Comput.*, vol. 17, pp. 361–383, 1963.
[30] J. R. Phillips, "Error and complexity analysis for a collocation-grid-projection plus precorrected-FFT algorithm for solving potential integral equations with Laplace or Helmholtz kernels," in *Proc. 1995 Copper Mountain Conf. Multigrid Methods*, Apr. 1995.
[31] C. R. Anderson, "A method of local corrections for computing the velocity field due to a distribution of vortex blobs," *J. Comput. Phys.*, vol. 62, pp. 111–123, 1986.
[32] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White, "Multipole accelerated preconditioned iterative methods for three-dimensional potential integral equations of the first kind," *SIAM J. Sci. Stat. Comput.*, vol. AP-15, pp. 713–735, May 1994.
[33] D. T. Borup and O. P. Gandhi, "Calculation of high-resolution SAR distributions in biological bodies using the FFT algorithm and conjugate gradient method," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, pp. 417–419, 1985.
[34] T. K. Sarkar, E. Arvas, and S. M. Rao, "Application of FFT and the conjugate gradient method for the solution of electromagnetic radiation from electrically large and small conducting bodies," *IEEE Trans. Antennas Propagat.*, vol. AP-34, pp. 635–640, 1986.
[35] J. R. Phillips, M. Kamon, and J. White, "An FFT-based approach to including nonideal ground planes in a fast 3-D inductance extraction program," in *Proc. Custom Int. Circuits Conf.*, May 1993.
[36] J. R. Phillips and J. K. White, "Efficient capacitance extraction of 3D structures using generalized pre-corrected FFT methods," in *Proc. IEEE 3rd Topical Meeting on Electrical Performance of Electronic Packaging*, Nov. 1994.

Joel R. Phillips received B.S. degrees in physics and electrical engineering, the M.S. degree in electrical engineering, and the Ph.D. degree in electrical engineering and computer science, all from the Massachusetts Institute of Technology, Cambridge, in 1991, 1993, and 1997, respectively.

In 1997, he joined Cadence Design Systems, San Jose, CA. His research interests include computational electromagnetics, algorithms for analysis, simulation and modeling of RF circuits and systems, nonlinear dynamics, and numerical methods for solution of integral equations.

Jacob K. White (S'80–M'83–A'88) received the B.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, and the S.M. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley.

He worked at the IBM T. J. Watson Research Center, Yorktown Heights, NY, from 1985 to 1987, was the Analog Devices Career Development Assistant Professor at MIT from 1987 to 1989. He is currently a Professor at MIT, and his research interests are in serial and parallel numerical algorithms for problems in circuit, interconnect, device, and microelectromechanical system design.

Dr. White was a 1988 Presidential Young Investigator, and an associate editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF CIRCUITS AND SYSTEMS from 1992 until 1996.