A Massively Parallel Computation Strategy for FDTD: Time and Space Parallelism Applied to Electromagnetics Problems

Amir Fijany, *Member, IEEE,* Michael A. Jensen, *Member, IEEE,*
Yahya Rahmat-Samii, *Fellow, IEEE,* and Jacob Barhen, *Member, IEEE*

***Abstract*— In this paper, we present a novel strategy for incorporating massive parallelism into the solution of Maxwell's equations using finite-difference time-domain methods. In a departure from previous techniques wherein spatial parallelism is used, our approach exploits massive temporal parallelism by computing all of the time steps in parallel. Furthermore, in contrast to other methods which appear to concentrate on explicit schemes such as Yee's algorithm, our strategy uses the implicit Crank–Nicolson technique which provides superior numerical properties. We show that the use of temporal parallelism results in algorithms which offer a massive degree of coarse grain parallelism with minimum communication and synchronization requirements. Due to these features, the time-parallel algorithms are particularly suitable for implementation on emerging massively parallel multiple instruction-multiple data (MIMD) architectures. The methodology is applied to a circular cylindrical configuration, which serves as a testbed problem for the approach, to demonstrate the massive parallelism that can be exploited. We also discuss the generalization of the methodology for more complex problems.**

I. Introduction

THE application of finite-difference and finite-volume time-domain (FDTD and FVTD) methods to the solution of Maxwell's equations has been encouraged by increased efforts to model electrically large and complex radiators and scatterers. Because of the intensive computation and storage requirements associated with these techniques, however, their practical implementation for very large problems presents some challenges. In a recent survey of computational electromagnetic (CEM) techniques [1], it is suggested that a key solution to the computational power and storage requirement bottlenecks is the exploitation of a massive degree of parallelism by implementing CEM FDTD algorithms on parallel architectures. To fully exploit the computing power offered by available parallel platforms, however, existing algorithms must be reexamined with a focus on their efficiency for parallel implementation. Eventually, new algorithms may have to be developed that, from the onset, take a greater advantage of the available massive parallelism.

An emerging class of massively parallel multiple instruction-multiple data (MIMD) supercomputers, such as Intel's Delta and Paragon and CRAY's T3D, appears to set the current and future trend in massively parallel computing technology. The very large number of vector processors employed in these supercomputers couples the high level, coarse grain parallelism of the MIMD architecture with a lower level vector processing capability to provide an impressive computational throughput. The main limitation of these architectures lies in their rather limited communication structure. This feature makes them most suitable for algorithms which possess a high degree of coarse grain parallelism, require limited communication and synchronization, and involve basic operations or algorithmic processes that can be efficiently vectorized. The algorithms presented in this paper have been developed based upon this model of computation and, as such, are highly efficient for implementation on this class of massively parallel MIMD supercomputers.

A number of finite-difference schemes have been developed and used to solve Maxwell's time-domain coupled partial differential equations (PDE's) [2]–[5]. The majority of these techniques are explicit and require a sparse matrix-vector multiplication at each time step. The high degree of spatial parallelism—the exploitation of parallelism at each time step, suggested by these schemes has motivated considerable recent work on space-parallel FDTD implementations. This parallelism, however, is rather fine grain and, due to communication and synchronization requirements, it cannot be efficiently exploited by massively parallel MIMD architectures. In fact, practical implementations of one widely used technique, commonly referred to as Yee's algorithm [2], on computers such as the Hypercube [6], [7] and Transputer [8] clearly show that computational speed-up is limited since only a few processors can be efficiently employed. In contrast, as suggested in [4] and [5], these methods are highly suitable for vector processing and as such may be efficiently computed using a high degree of vectorization but a limited degree of parallelism.

In this paper, we propose a novel computational strategy which uses time-parallelism, the exploitation of parallelism in the computation of all the time steps, in the finite-difference solution of the scalar wave equation. In a clear departure from

Manuscript received April 4, 1994; revised June 14, 1995. This work was supported in part by the Jet Propulsion Laboratory under contract with the National Aeronautics and Space Administration (NASA) and by ARPA Contract DAAB07-93-C-C501.

A. Fijany and J. Barhen are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA.

M. Jensen is with the Department of Electrical and Computer Engineering, 459 CB, Brigham Young University, Provo, UT 84602 USA.

Y. Rahmat-Samii is with the Electrical Engineering Department, University of California, Los Angeles, Los Angeles, CA, 90024-1594 USA.

IEEE Log Number 9415647.

typical FDTD methods, the technique uses a fully implicit Crank–Nicolson (CN) discretization [9] and therefore offers the benefits of unconditional stability. This feature often allows a reduction in the number of time steps required to complete the simulation. The resulting time-parallel algorithm offers a high degree of coarse grain parallelism with minimum communication and synchronization requirements, making it highly efficient for implementation on massively parallel MIMD architectures. The application of our time-parallel computing approach to determine the electromagnetic behavior of fields near circular cylindrical geometries clearly reveals the massive temporal parallelism which can be efficiently exploited in the computation. It is further shown that with the availability of a larger number of processors spatial parallelism can be also exploited in the computation, resulting in a time- and space-parallel algorithm which remains highly coarse grain with a simple communication structure.

The paper is organized as follows. Section II reviews some fundamental ideas relating to the parallel solution of time-dependent PDE's and introduces the underlying concepts behind time-parallel computation. In Section III, the algorithm is applied to the circular cylinder. The performance of the time-parallel algorithm with respect to the best sequential explicit and implicit methods for the same cylinder problem is analyzed in Section IV. Generalization of the time-parallel computing approach is discussed in Section V. Finally, Section VI provides some concluding remarks.

II. Time-Parallel Algorithm for FDTD CEM

A. Parallel Solution of Time-Dependent PDE's

Maxwell's time-dependent coupled PDE's form the computational basis for the time-parallel algorithm implementation. For homogeneous, isotropic materials, these equations can be written as two decoupled second-order wave equations which assume the form

$$\frac{\partial^2 \vec{E}}{\partial t^2} = c^2 \nabla^2 \vec{E}$$
$$\frac{\partial^2 \vec{H}}{\partial t^2} = c^2 \nabla^2 \vec{H} \qquad (1)$$

where c is the speed of light and $\vec{E}$ and $\vec{H}$ represent the electric and magnetic field intensities, respectively. Depending on the problem geometry, a set of scalar hyperbolic equations can be selected from (1). These equations can be expressed in a general form as

$$\frac{\partial^2 \psi(\vec{r},t)}{\partial t^2} = \nabla^2 \psi(\vec{r},t) + g(\vec{r},t) \qquad \vec{r} \in \Omega, 0 < t < T \quad (2)$$

where Ω is a bounded domain with boundary Ω' and $g(\vec{r},t)$ indicates a time- and space-dependent source term. Using finite-difference approximations for the derivatives results in a general discretized form of (2) which may be written in a matrix form as

$$\boldsymbol{A}_1 \psi^{(m+1)} = \boldsymbol{A}_2 \psi^{(m)} + \boldsymbol{A}_3 \psi^{(m-1)} + f^{(m+1)} \qquad (3)$$

for $1 \leq m \leq M-1$ where $\psi^{(m)}$ is a column vector representing the approximate solution at the mth time step, Δt is the time step size, and $M = T/\Delta t$. The term $f^{(m+1)}$ results from the discretization of $g(\vec{r},t)$ in time and space, and $\psi^{(0)}$ and $\psi^{(1)}$ are the given initial conditions. The specific structure of the matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$, and $\boldsymbol{A}_3$ and the computation of $f^{(m+1)}$ depend upon the solution method as well as the time and space discretization strategies employed.

Although most of the current research on developing parallel techniques for solution of time-dependent PDE's appears to concentrate on parabolic equations, some of the techniques developed can be extended to the solution of hyperbolic equations. Recent trends in this arena seem to be motivated by three widely acknowledged observations [10]–[12]:

1) Explicit methods, while limited in their range of stability, are highly efficient for parallel and/or vector processing since the computation at each time step mainly involves a matrix-vector multiplication. This has motivated the development of new explicit methods which offer improved numerical properties while preserving the efficiency for parallel/vector computation [13], [14].
2) Implicit algorithms, despite their superior numerical properties, are not efficient for parallel and/or vector processing since at each time step a linear system solution is required. This has motivated new implicit techniques which improve the efficiency for space-parallel computation [11], [15].
3) The implementation of time-stepping methods is generally assumed to be strictly sequential in time, implying that the solution for time step $m+1$ cannot be obtained without first computing the solution for step m. This has motivated the investigation of new iterative techniques to increase parallelism in time [16]–[18]. The resulting algorithms are limited to parabolic equations, however, and achieve a rather limited temporal parallelism. In fact, it has been implied that simultaneous solution for all time steps is not feasible [16], [19].

In contrast, we have recently shown that by departing from these governing assumptions, it is possible to efficiently solve implicit time-stepping methods in a fully time-parallel fashion [20], [21]. In fact, the practical application of such a time-parallel algorithm to a simple two-dimensional heat problem on Intel's Touchstone Delta has shown that linear and even super-linear speed-up can be achieved by using a very large number of processors (of the order of 10^2) [22]. In the following, we discuss the implementation of the time-parallel algorithm to the CN solution of (2).

B. Temporal Parallelism

It is noteworthy that (3) simply represents a second-order inhomogeneous linear recursion (SOILR) which can be cast into a first-order one. The concept of temporal parallelism is to solve this recurrence in parallel using the recursive doubling algorithm (RDA) or cyclic reduction algorithm (CRA) [23]. Unfortunately, such an approach is not computationally practical, since both RDA and CRA compute powers and products of the matrices in (3) result in increasingly dense and ultimately full matrices. In such an event, the cost of

one full matrix–matrix multiplication alone would be much greater than the cost of any serial algorithm. Nevertheless, this observation clearly indicates that, insofar as the data dependency in the computation is concerned, the time-stepping procedure in (3) can be fully parallelized in time.

Motivated by this observation, we have developed a technique which allows efficient time-parallelization of time-stepping procedures, leading to a highly practical approach for massively parallel computation. To describe this technique, consider the CN method for solution of (2). As shown in the Appendix, (3) may be written as

$$(\boldsymbol{I}-\alpha\boldsymbol{A})\psi^{(m+1)} = 2\boldsymbol{I}\psi^{(m)} - (\boldsymbol{I}-\alpha\boldsymbol{A})\psi^{(m-1)} + f^{(m+1)} \quad (4)$$

for $1 < m < M-1$ where $\boldsymbol{I}$ is the unit matrix, α is a constant, and $\boldsymbol{A}$ is the matrix arising from the discretization of the Laplace operator. The vector $f^{(m+1)}$ contains any source or otherwise known field values in the spatial domain. Now, let the Eigenvalue/Eigenvector (EE) decomposition of the nonsingular matrix $\boldsymbol{A}$ be given by

$$\boldsymbol{A} = \boldsymbol{\Theta}\boldsymbol{\Lambda}\boldsymbol{\Theta}^{-1} \quad (5)$$

where $\boldsymbol{\Theta}$ is the set of eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix representing the set of eigenvalues of $\boldsymbol{A}$. Substituting (5) into (4) and taking $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}^{-1}$ outside the parentheses gives the expression

$$\begin{aligned}&\boldsymbol{\Theta}(\boldsymbol{I}-\alpha\boldsymbol{\Lambda})\boldsymbol{\Theta}^{-1}\psi^{(m+1)}\\ &\quad = 2\boldsymbol{I}\psi^{(m)} - \boldsymbol{\Theta}(\boldsymbol{I}-\alpha\boldsymbol{\Lambda})\boldsymbol{\Theta}^{-1}\psi^{(m-1)} + f^{(m+1)}.\end{aligned} \quad (6)$$

Multiplying both sides of (6) by the nonsingular matrix $\boldsymbol{\Theta}^{-1}$, we obtain

$$\begin{aligned}&(\boldsymbol{I}-\alpha\boldsymbol{\Lambda})\boldsymbol{\Theta}^{-1}\psi^{(m+1)}\\ &\quad = 2\boldsymbol{I}\boldsymbol{\Theta}^{-1}\psi^{(m)} \quad (\boldsymbol{I} \quad \alpha\boldsymbol{\Lambda})\boldsymbol{\Theta}^{-1}\psi^{(m-1)} \; | \; \boldsymbol{\Theta}^{-1}f^{(m+1)}.\end{aligned} \quad (7)$$

Defining a diagonal matrix $\boldsymbol{S} = (\boldsymbol{I}-\alpha\boldsymbol{\Lambda})^{-1}$ and the vectors $\tilde{\psi}^{(m)} = \boldsymbol{\Theta}^{-1}\psi^{(m)}$ and $\tilde{f}^{(m)} = \boldsymbol{S}\boldsymbol{\Theta}^{-1}f^{(m)}$ allows us to write (7) as

$$\tilde{\psi}^{(m+1)} = 2\boldsymbol{S}\tilde{\psi}^{(m)} - \tilde{\psi}^{(m-1)} + \tilde{f}^{(m+1)}. \quad (8)$$

In contrast to (4), (8) is diagonal and can therefore efficiently be solved in parallel using RDA or CRA. For example, for two- and three-dimensional problems (spatial grids of $N \times N$ and $N \times N \times N$ nodes, respectively), (8) can be computed in $O(N^2 \log M)$ and $O(N^3 \log M)$ by using $O(M)$ processors.

C. Structure of the Time-Parallel Algorithm

To facilitate the discussion of the algorithm, we subdivide it into four steps as illustrated in Fig. 1. Step 1 involves determining the EE decomposition of the matrix $\boldsymbol{A}$ and forming the matrix $\boldsymbol{S}$. The source vectors $f^{(m)}$ are computed and multiplied by $\boldsymbol{S}\boldsymbol{\Theta}^{-1}$ in Step 2 to obtain the vectors $\tilde{f}^{(m)}$. The parallel solution of the SOILR in (8) is accomplished in Step 3. Finally, the solution vectors $\psi^{(m)}$ are obtained in Step 4 by performing the matrix-vector multiplication $\boldsymbol{\Theta}\tilde{\psi}^{(m)}$.

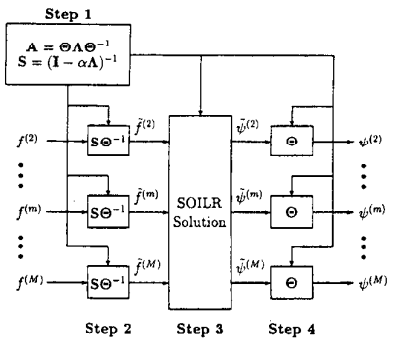


Fig. 1. Flow diagram illustrating the computational structure of the time-parallel algorithm.

Several key issues relating to these steps must be addressed before applying the approach to an example problem. To begin, it is important that the computation in Step 1 typically only generates $\boldsymbol{\Theta}$ and $\boldsymbol{\Lambda}$, and therefore the matrix $\boldsymbol{\Theta}^{-1}$ must be explicitly computed. This is simply performed if $\boldsymbol{A}$ is symmetric since $\boldsymbol{\Theta}^{-1} = \boldsymbol{\Theta}^T$ (where T indicates transpose). If $\boldsymbol{A}$ is nonsymmetric, the fact that $\boldsymbol{A}^T = (\boldsymbol{\Theta}^{-1})^T\boldsymbol{\Lambda}\boldsymbol{\Theta}^T$ suggests that the EE decomposition of $\boldsymbol{A}^T$ will yield $\boldsymbol{\Theta}^{-1}$. This can be performed in parallel with the EE decomposition of $\boldsymbol{A}$ without increasing the overall computational complexity. Because the computation in Step 1 is space dependent, it needs to be performed only once for a given problem geometry.

The transformations in both Steps 2 and 4 are completely decoupled and can be performed in parallel using $O(M)$ processors with no interprocessor communication. The communication required in the computation of Step 3 has a rather simple structure and can be efficiently implemented on MIMD parallel architectures since it involves the exchange of large vectors among processors. For practical temporal and spatial grid sizes, the computational cost of Step 3 is much less than that of Steps 2 and 4. This implies that the overall complexity of the time-parallel algorithm is dominated by the computations of Steps 2 and 4 which can be fully parallelized in time. This also illustrates the highly decoupled structure, coarse grain size, and vector nature of the scheme.

These features of the time-parallel algorithm have motivated us to identify the class of problems to which the scheme may be efficiently applied. Two key requirements for the efficient application of the method are:

1) an efficient scheme for determination of the eigenpairs of $\boldsymbol{A}$,
2) an efficient scheme for the matrix-vector multiplications in Steps 2 and 4.

The first issue motivates us to seek problems which exploit a high degree of parallelism in the EE decomposition of $\boldsymbol{A}$. The second issue is particularly important since multiplication of a dense matrix by a vector leads to excessively high computational costs. This fact motivates us to exploit the sparse structure of $\boldsymbol{A}$ and to express $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}^{-1}$ as products of highly sparse matrices to reduce the complexity of these steps.

The implications of these two issues will be clarified through application of the algorithm to a representative problem.

D. Absorbing Boundary Condition

For many problems, particularly enclosed problems involving resonant cavities or wave-guiding structures, the time-parallel algorithm may be readily applied since typically either a Dirichlet or Neumann condition is enforced on the outer spatial domain boundary. For open problems encountered in radiation and scattering, however, the issue of incorporating an absorbing boundary condition (ABC) [24] at the outer domain boundary must be addressed. This ABC is a computational grid termination which allows waves to exit the domain without undergoing artificial reflections at the grid boundary. For the time-parallel algorithm, proper choice of an ABC is of critical importance. This arises from the fact that the derivation of the scheme depends upon the fundamental assumption that all matrices in (3) and (4) are simultaneously diagonalizable. While this assumption holds for conventional boundary conditions (i.e., Dirichlet, Neumann, Mixed, and Periodic), it is not necessarily valid for all existing forms of ABC's.

In light of this fact, additional research is under way to identify improved techniques for including nonreflecting boundary conditions within the framework of the time-parallel algorithm. One potential ABC involves solving the problem once for a Dirichlet and once for a Neumann boundary condition at the outer domain boundary [24] and then averaging the results to give the desired solution [24], [25]. This scheme not only maintains the diagonalizability of (4) but also is highly suitable for parallel computation since the two solutions can be performed in parallel. Our investigation, however, has shown that this ABC suffers from inaccuracies when the temporal duration is long enough for multiple reflections to occur. A second possible strategy which we have developed makes use of a time- and space-parallel solution of (2) using a first order ABC [24]. This procedure has recently been presented by the authors [26].

III. Algorithm Implementation for a Circular Cylinder

Although the high-level structure of the time-parallel algorithm as outlined in Section II-C is the same for various applications, some of the details of the implementation may change from problem to problem. In this section, we illustrate the method by applying it to the CN solution of Maxwell's equations for circular cylindrical geometries. The concepts illustrated here can be used to compute the behavior of coaxial cavity configurations or the scattering from circular cylinders depending on the choice of boundary conditions used. The key area of emphasis in this demonstration involves determining the EE decomposition of $\boldsymbol{A}$ and diagonalizing the resulting CN matrix equation.

A. Laplace Operator in Polar Coordinates

Let $\rho_0 = (q-1)\Delta\rho$ and $\rho_1 = (p+1)\Delta\rho$, where $K = p - q + 1$, represent the radii of the cylinder and the edge of the finite computational domain, as shown in Fig. 2. In this case, the domain Ω of (2) is an annulus between ρ_0 and ρ_1. Also let ψ represent E_z, the $\hat{z}$-polarized electric field surrounding the circular cylinder. The polar grid is defined by the points $(\rho = i\Delta\rho, \phi = j\Delta\phi)$ for $q-1 \le i \le p+1$ and $1 \le j \le N$, where $\Delta\phi = 2\pi/N$. Using this notation along with the convention that $\psi(i\Delta\rho, j\Delta\phi, m\Delta t) = \psi_{i,j}^{(m)}$, the five-point finite-difference approximation of the Laplace operator becomes

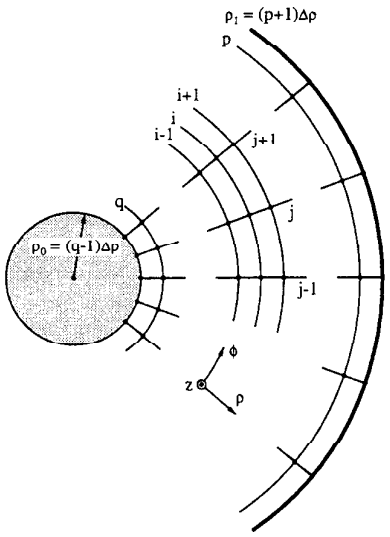


Fig. 2. Geometry and gridding for the CN solution of Maxwell's equations near a perfectly conducting circular cylinder.

$$\begin{aligned}\nabla^2\psi(i\Delta\rho, j\Delta\phi, m\Delta t) &= \frac{1}{(\Delta\rho)^2}\left\{\left(1+\frac{1}{2i}\right)\psi_{i+1,j}^{(m)} + \left(1-\frac{1}{2i}\right)\psi_{i-1,j}^{(m)} - 2\left[1+\frac{1}{(i\Delta\phi)^2}\right]\psi_{i,j}^{(m)} + \frac{1}{(i\Delta\phi)^2}\left[\psi_{i,j+1}^{(m)} + \psi_{i,j-1}^{(m)}\right]\right\}. \end{aligned} \tag{9}$$

If the field scattered from the cylinder is to be computed, the boundary condition

$$\psi_{q-1,j}^{(m)} = -E_z^{\text{inc}}(\rho_0, j\Delta\phi, m\Delta t) \tag{10}$$

must be satisfied on the inner domain boundary where E_z^{inc} represents the incident plane wave (assuming TM^z incidence). In the CN matrix equation, this is accomplished by placing the known boundary field values into the source vector $f^{(m)}$.

B. EE Decomposition: Dirichlet Boundary Condition

With a Dirichlet boundary condition at ρ_1 in (9), the matrix $\boldsymbol{A} \in \boldsymbol{R}^{KN\times KN}$ in (4) becomes block tridiagonal with the form

$$(\Delta\rho)^2\boldsymbol{A} = \text{Tridiag}\left[\left(1-\frac{1}{2i}\right)\boldsymbol{I} \quad \boldsymbol{B}_i \quad \left(1+\frac{1}{2i}\right)\boldsymbol{I}\right] \tag{11}$$

for $q \leq i \leq p$ where $\boldsymbol{B}_i = \beta_i \boldsymbol{B} - 2\boldsymbol{I} \in \boldsymbol{R}^{N\times N}$ and $\beta_i = (1/i\Delta\phi)^2$. The matrix $\boldsymbol{B}$ is given by

$$\boldsymbol{B} = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & & & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 \\ 1 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}. \tag{12}$$

With $\boldsymbol{A}$ written in the form of (11), it becomes straightforward to determine its EE decomposition. It can be shown [27, p. 253] that

$$\boldsymbol{B} = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{-1} \tag{13}$$

where $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ are the direct and inverse one-dimensional discrete Fourier transform (DFT) operators and $\boldsymbol{D} = \text{Diag}\{-4\sin^2[(j-1)\pi/N]\}, 1 \leq j \leq N$. From the definition of $\boldsymbol{B}_i$, it follows that $\boldsymbol{B}_i$ and $\boldsymbol{B}$ share the same set of eigenvectors but have a different set of eigenvalues. In light of this, the EE decomposition of $\boldsymbol{B}_i$ is given by

$$\boldsymbol{B}_i = \boldsymbol{F}\lambda_i\boldsymbol{F}^{-1} \tag{14}$$

where $\lambda_i = \text{Diag}\{\lambda_{i,j}\} = \beta_i\boldsymbol{D} - 2\boldsymbol{I}, 1 \leq j \leq N$, such that

$$\lambda_{i,j} = -4\beta_i \sin^2[(j-1)\pi/N] - 2. \tag{15}$$

Using (14), the matrix $\boldsymbol{A}$ can be written as

$$(\Delta\rho)^2\boldsymbol{A} = \mathcal{F}\mathcal{R}\mathcal{F}^{-1} \tag{16}$$

where

$$\mathcal{R} = \text{Tridiag}[(1-1/2i)\boldsymbol{I} \quad \lambda_i \quad (1+1/2i)\boldsymbol{I}] \tag{17}$$

$$\mathcal{F} = \text{Diag}\{\boldsymbol{F}, \boldsymbol{F}, \cdots, \boldsymbol{F}\} \in \boldsymbol{R}^{KN\times KN}. \tag{18}$$

Since the block elements of $\mathcal{R}$ are diagonal, it can be transformed to a block diagonal matrix $\mathcal{T}$ using

$$\mathcal{R} = \mathcal{P}\mathcal{P}^T\mathcal{R}\mathcal{P}\mathcal{P}^T = \mathcal{P}(\mathcal{P}^T\mathcal{R}\mathcal{P})\mathcal{P}^T = \mathcal{P}\mathcal{T}\mathcal{P}^T \tag{19}$$

where $\mathcal{P}$ is the orthogonal permutation matrix which arises in the two-dimensional DFT. The matrix $\mathcal{T}$ is given by

$$\mathcal{T} = \text{Diag}\{\boldsymbol{T}_j\} \in \boldsymbol{R}^{KN\times KN} \tag{20}$$

$$\boldsymbol{T}_j = \text{Tridiag}[(1-1/2i) \quad \lambda_{i,j} \quad (1+1/2i)] \tag{21}$$

for $q \leq i \leq p$ and $1 \leq j \leq N$. Let the EE decomposition of $\boldsymbol{T}_j$ be given by

$$\boldsymbol{T}_j = \boldsymbol{Q}_j\boldsymbol{\Lambda}_j\boldsymbol{Q}_j^{-1}. \tag{22}$$

It is noteworthy that since $\boldsymbol{T}_j$ is a sign symmetric matrix, $\boldsymbol{\Lambda}_j$ will be a real matrix which can be efficiently computed using, for example, the RT subroutine provided by EISPACK [28]. Also, following our discussion in Section II-C, $\boldsymbol{Q}_j^{-1}$ may be explicitly obtained by performing the EE decomposition of $\boldsymbol{T}_j^T$. Now, defining $\mathcal{Q} = \text{Diag}\{\boldsymbol{Q}_j\}$ and $\boldsymbol{\Lambda} = \text{Diag}\{\boldsymbol{\Lambda}_j\}, 1 \leq j \leq N$, it follows that

$$\mathcal{T} = \mathcal{Q}\boldsymbol{\Lambda}\mathcal{Q}^{-1}. \tag{23}$$

Substituting (23) into (19) and (16), we obtain

$$(\Delta\rho)^2\boldsymbol{A} = \mathcal{F}\mathcal{P}\mathcal{Q}\boldsymbol{\Lambda}\mathcal{Q}^{-1}\mathcal{P}^T\mathcal{F}^{-1} = \boldsymbol{\Theta}\boldsymbol{\Lambda}\boldsymbol{\Theta}^{-1} \tag{24}$$

where $\boldsymbol{\Theta} = \mathcal{F}\mathcal{P}\mathcal{Q}$.

C. EE Decomposition: Neumann Boundary Condition

For a homogeneous Neumann boundary condition at the outer domain boundary, we place $\psi_{p+2,j} = \psi_{p,j}$ into (9) for $i = p+1$ to obtain the block tridiagonal matrix $\hat{\boldsymbol{A}} \in \boldsymbol{R}^{K'N\times K'N}$ $(K' = K+1)$ which assumes the form

$$(\Delta\rho)^2\hat{\boldsymbol{A}} = \text{Tridiag}\left[\gamma_i\boldsymbol{I} \quad \boldsymbol{B}_i \quad \left(1+\frac{1}{2i}\right)\boldsymbol{I}\right] \tag{25}$$

for $q \leq i \leq p+1$ where

$$\gamma_i = \begin{cases} 1 - \dfrac{1}{2i} & q \leq i \leq p \\ 2 & i = p+1. \end{cases} \tag{26}$$

The similarities between the matrices $\hat{\boldsymbol{A}}$ and $\boldsymbol{A}$ in (11) imply that the process for computing their EE decompositions is similar. To this end, we define the matrices $\hat{\mathcal{F}}$ and $\hat{\mathcal{P}}$ similar to the matrices $\mathcal{F}$ and $\mathcal{P}$, but of dimension $K'N \times K'N$. The EE decomposition of $\hat{\boldsymbol{A}}$ becomes

$$(\Delta\rho)^2\hat{\boldsymbol{A}} = \hat{\mathcal{F}}\hat{\mathcal{P}}\hat{\mathcal{Q}}\hat{\boldsymbol{\Lambda}}\hat{\mathcal{Q}}^{-1}\hat{\mathcal{P}}^T\hat{\mathcal{F}}^{-1} \tag{27}$$

where $\hat{\mathcal{T}} = \text{Diag}\{\hat{\boldsymbol{T}}_j\}, 1 \leq j \leq N$ and

$$\hat{\boldsymbol{T}}_j = \text{Tridiag}\left[\gamma_i \quad \lambda_{i,j} \quad \left(1+\frac{1}{2i}\right)\right] \tag{28}$$

for $q \leq i \leq p+1$. The matrix $\hat{\mathcal{Q}} = \text{Diag}\{\hat{\boldsymbol{Q}}_j\}$ is computed from the N EE decompositions

$$\hat{\boldsymbol{T}}_j = \hat{\boldsymbol{Q}}_j\hat{\boldsymbol{\Lambda}}_j\hat{\boldsymbol{Q}}_j^{-1}. \tag{29}$$

D. Algorithm Structure for the Circular Cylinder

It is important to note that by using the factored form of the eigenvector matrices $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}^{-1}$ as formulated above, completion of the EE decomposition of $\boldsymbol{A}$ is tantamount to performing the EE decomposition of $2N$ matrices of dimension $K \times K$ and can therefore be performed in parallel. Additionally, the flow charts in Fig. 3 demonstrate the procedure for performing the matrix-vector products in Steps 2 and 4 as well as the SOILR solution in Step 3 using the factored matrices. These figures imply that an additional level of spatial parallelism may be exploited in the computation, an issue which is explored more fully in Section IV. Note that the SOILR in Fig. 3(b) is given by

$$\tilde{\psi}_j^{(m+1)} = 2\boldsymbol{S}_j\tilde{\psi}_j^{(m)} - \tilde{\psi}_j^{(m-1)} + \tilde{f}_j^{(m+1)} \tag{30}$$

where $\boldsymbol{S} = \text{Diag}\{\boldsymbol{S}_j\}$.

IV. Algorithm Performance: Circular Cylinder

A. Computational Complexity: Time-Parallel

In analyzing the performance of the time-parallel algorithm, we recall that M represents the number of time steps, and K and N denote the number of radial and azimuthal cells, respectively, in the spatial grid. We assume that M processors are available to fully exploit temporal parallelism. For typical values of M (of the order of 10^3) and K and N (of the order of 10^2), we have $M \gg K$ and $M \gg N$. Table I illustrates the

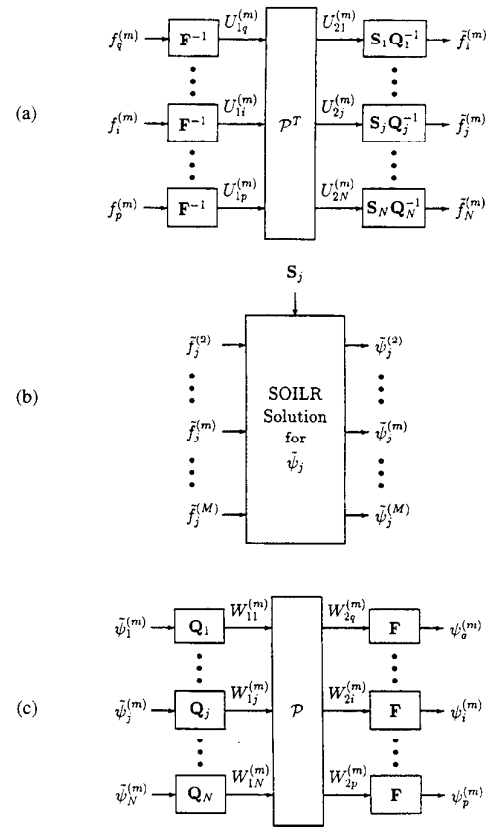


Fig. 3. Flow diagram illustrating the space-parallelism in the computation of (a) Step 2, (b) Step 3, and (c) Step 4.

TABLE I
COMPUTATIONAL COMPLEXITIES OF THE DIFFERENT STEPS IN THE TIME PARALLEL ALGORITHM

Operation	Cost	Operation	Cost
EE of $\mathbf{T}_j$	$O(K^2)$	SOILR for $\tilde{\psi}_j$	$O(K \log M)$
$U_{1i}^{(m)} = \mathbf{F}^{-1} f_i^{(m)}$	$O(N \log N)$	$W_{1j}^{(m)} = \mathbf{Q}_j \tilde{\psi}_j^{(m)}$	$O(K^2)$
$\tilde{f}_j^{(m)} = \mathbf{S}_j \mathbf{Q}_j^{-1} U_{2j}^{(m)}$	$O(K^2)$	$\psi_i^{(m)} = \mathbf{F} W_{2i}^{(m)}$	$O(N \log N)$

computational complexity associated with each of the major operations outlined in Fig. 3 for the time parallel algorithm applied to the circular cylinder.

Using the expressions in Table I, the overall complexity of the algorithm assuming $O(M)$ processors and $M \geq 2N$ can be determined. The EE decomposition of the $2N$ matrices $\boldsymbol{T}_j$ and $\boldsymbol{T}_j^T$ can be performed in parallel at a total cost of $O(K^2)$. The K one-dimensional DFT's and N multiplications $\tilde{f}_j^{(m)} = \boldsymbol{S}_j \boldsymbol{Q}_j^{-1} U_{2j}^{(m)}$ for Step 2 require an overall cost of $O(KN \log N + K^2 N)$. In Step 3, since the SOILR must be performed for N vectors $\tilde{\psi}_j^{(m)}$, the total SOILR cost (using RDA or CRA [23]) is $O(KN \log M)$. Since the cost of Step 4 is the same as that of Step 2, the total computational complexity of the time-parallel algorithm while fully exploiting temporal parallelism is given by

$$C_{TP} = a_1 K^2 N + a_2 KN \log N + a_3 KN \log M \qquad (31)$$

where a_1, a_2, and a_3 are constants and lower-degree terms have been neglected.

B. Computational Complexity: Time- and Space-Parallel

With the availability of a larger number of processors, spatial parallelism can also be exploited in the computation. The operations shown in Fig. 3 reveal the structure of the spatial parallelism in the algorithm.

For a time- and space-parallel computation we consider using ML processors where $L = \text{Max}(N, K)$. With this strategy, the computation of K one-dimensional DFT's and N matrix-vector products in Steps 2 and 4 can now be done at a cost of $O(N \log N)$ and $O(K^2)$, respectively. Solution of the SOILR's in Step 3 results in a complexity of $O(K \log M)$. The computational cost of the time- and space-parallel implementation of the algorithm is therefore given as

$$C_{TSP} = b_1 K^2 + b_2 N \log N + b_3 K \log M \qquad (32)$$

where b_1, b_2, and b_3 are constants.

Interestingly, even the space-parallel implementation of Steps 2–4 results in a coarse grain computation with a rather low communication complexity. The space-parallel computation of Step 3 can be performed in a fully decoupled fashion with no communication requirement. In this step, each processor performs the operations for parallel computation of its corresponding SOILR on vectors of dimension K. In Steps 2 and 4, each processor performs an FFT on vectors of dimension N and a multiplication of a $K \times K$ matrix by a $K \times 1$ vector.

The permutations in Steps 2 and 4, however, require communication among processors in the space-parallel computation. Consider Step 2 [Fig. 3(a)] as an example and let $N = K$. Before the permutation, each processor (e.g., processor i) computes the vector $U_{1i}^{(m)}$. If these vectors are considered as the columns of a matrix $\mathcal{U}$, then the permutation corresponds to transposing $\mathcal{U}$. In this case, processor i (which initially contained the ith column of matrix $\mathcal{U}$) will receive the ith row of $\mathcal{U}^T$. The complexity of such a data communication is a function of the processors interconnection structure. With K processors interconnected through a Hypercube topology, the complexity of this matrix transposition is of $O(K \log K)$ (see for example [29]). This implies that, with such interconnection topology, even the space-parallel implementation of Steps 2 and 4 remains highly compute bound since its computation complexity of $O(K^2)$ is greater than its communication complexity of $O(K \log K)$.

C. Computational Speed-up

The speed-up of the time-parallel and time- and space-parallel algorithms can be measured with respect to the best sequential explicit method (Yee's algorithm) and the best sequential implementation of the CN method for the problem. The application of Yee's algorithm to the circular cylinder essentially requires a matrix-vector multiplication at each time step. Because the $KN \times KN$ matrix involved in this operation is highly sparse, it can be shown that the computational cost

C_{SEY} of the sequential implementation of this algorithm is given as

$$C_{SEY} = c_1 M' K N \tag{33}$$

where c_1 is constant and M' is the number of time steps required to achieve the same level of accuracy as for the CN method. Due to the stability constraints, M' may be much greater than M. The speed-up of the time-parallel algorithm with respect to the sequential implementation of Yee's algorithm is then given by

$$SP_1 = \frac{C_{SEY}}{C_{TP}} = \frac{c_1 M'}{a_1 K} = O\left(\frac{M'}{K}\right) \tag{34}$$

where it is assumed here and in the following expressions that $K > \log N$ and $K > \log M$.

The speed-up of the time- and space-parallel algorithm with respect to the sequential implementation of Yee's algorithm is given approximately by

$$SP_2 = \frac{C_{SEY}}{C_{TSP}} = \frac{c_1 M'}{b_1 K/N} = O\left(\frac{M'N}{K}\right) = O(M') \tag{35}$$

where the last equality holds when $O(K) = O(N)$.

The sequential solution of (4) at each time step involves two vector additions, each at a cost of $O(KN)$, and one matrix-vector multiplication to form the right-hand side vector. By exploiting the structure of $\boldsymbol{A}$, the matrix-vector multiplication can also be performed with a cost of $O(KN)$. The solution vector $\psi^{(m+1)}$ is subsequently obtained by solving a linear system. Since the matrices $(I - \alpha \boldsymbol{A})$ and $\boldsymbol{A}$ have a similar structure, this linear system solution is equivalent to the solution of the Poisson equation in polar coordinates which can be performed using the fast Poisson solver [30] with a complexity of $O(KN \log N)$. It then follows that the cost of the best sequential algorithm for solution of (4), denoted as C_{SCN}, is given by

$$C_{SCN} = d_1 M K N \log N + d_2 M K N \tag{36}$$

where d_1 and d_2 are constants. The speed-up of the time-parallel algorithm with respect to the best sequential algorithm for computation of (4) is then given approximately by

$$SP_3 = \frac{C_{SCN}}{C_{TP}} = \frac{d_1 M \log N}{a_1 K} = O\left(\frac{M \log N}{K}\right). \tag{37}$$

The speed-up of the time- and space-parallel algorithm with respect to the best sequential algorithm for computation of (4) is obtained approximately as

$$SP_4 = \frac{C_{SCN}}{C_{TSP}} = \frac{d_1 M \log N}{a_1 K/N} = O\left(\frac{MN \log N}{K}\right) = O(M \log N). \tag{38}$$

Table II summarizes the asymptotic computational complexity of the serial and parallel algorithms and the number of processors required. As can be seen from (34)–(38), the time-parallel and time- and space-parallel implementation of our algorithm leads to a massive speed-up in the computation while resulting in highly coarse grain parallel computation with simple communication and synchronization requirements.

TABLE II
COMPARISON OF SERIAL AND PARALLEL ALGORITHMS

	Sequential		Parallel	
	Explicit (Yee's)	Implicit CN	Time Parallel	Time/Space Parallel
Computational Complexity	$O(M'KN)$	$O(MKN \log N)$	$O(K^2N)$	$O(K^2)$
Number of Processors	-	-	$O(M)$	$O(ML)$

For typical values of M' (of the order of 10^4), M (of the order of 10^3), and K and N (of the order of 10^2), the time-parallel implementation of our algorithm leads to more than two orders-of-magnitude speed-up in the computation over the best sequential explicit and implicit methods. An even more impressive speed-up is possible using the time- and space-parallel implementation of the algorithm which, when used with a massive number of processors, is several orders-of-magnitude faster than the best sequential algorithm.

It is noteworthy that the time-parallel algorithm performance increases for problems requiring a much larger number of time steps. This follows from the fact that time dependence occurs only in the computation of Step 3, with a dependency of $O(\log M)$, while the dominant parts of the computation (Steps 2 and 4) are fully decoupled in time and are therefore independent of M.

V. GENERALIZATION OF TIME-PARALLEL APPROACH

Provided that the matrix $\boldsymbol{A}$ is nonsingular, the diagonalization process of Section II can be applied to any time-stepping procedure for solution of Maxwell's equations. Hence, the main issue in generalization of the approach is not the domain of applicability but rather the computational efficiency. As indicated in Section II-C, efficient application of the time-parallel approach requires fast schemes for computing the eigenpairs of $\boldsymbol{A}$ and multiplying the matrix of eigenvectors by a vector. We have identified three main techniques for accomplishing these tasks for more general problems.

A. Analytical Expressions

For a few simple cases involving regular domains, an analytical expression for the eigenpairs of $\boldsymbol{A}$ is known *a priori*. Well-known examples are cases involving two- and three-dimensional square and cubical domains. This approach allows extremely efficient application of the time-parallel algorithm. It appears, however, that the domain of applicability for this approach remains quite limited.

B. Divide and Conquer

The second technique exploits the specific structure of $\boldsymbol{A}$ by reducing the computation of eigenpairs of $\boldsymbol{A}$ to the computation of eigenpairs of a set of simpler matrices. Such a divide and conquer approach was applied to the problem in Section III. As seen in that example, this technique is highly efficient with an optimal computational complexity for most cases and furthermore can be accomplished in parallel. Additionally, the resulting matrix of eigenvectors can be efficiently multiplied by a vector by using the factored matrix form.

C. General Sparse Matrix Techniques

For problems where these methods can not be applied, the highly sparse structure of $\boldsymbol{A}$ must be exploited to efficiently compute its eigenpairs [31]. Because the key objective is to multiply the matrix of eigenvectors by a vector, it is more efficient to obtain this matrix in a factored form to improve the efficiency of this matrix-vector multiplication. This is a clear departure from most conventional sparse eigenproblem techniques wherein the explicit computation of the matrix of eigenvectors is sought.

VI. Conclusion

In this paper, we have introduced a novel time-parallel approach for solving Maxwell's equations using FDTD techniques. The algorithm provides a massive degree of coarse grain parallelism with simple communication and synchronization requirements. Furthermore, in contrast to previous work which has emphasized the use of explicit FDTD methods, this approach exploits the superior numerical properties of the implicit CN method. The application of the algorithm to solution of a testbed problem has illustrated the massive speed-up that can be achieved by exploiting temporal and spatial parallelism. Work is currently underway to implement the algorithm on the Touchstone Delta supercomputer for the circular cylinder problem discussed in this paper as well as for other problems. Preliminary results from these implementations have been presented in [26]. A more extensive review of the algorithm performance will be presented in a future correspondence.

In general, however, even the exploitation of full temporal parallelism alone requires a number of processors which, by far, exceeds that offered by the current generation of massively parallel MIMD architectures. Future generations of these architectures are expected to employ many thousands of processors and to achieve a Teraflop computing capability. Our preliminary results clearly point to a new direction in massively parallel CEM which would enable efficient application of these future architectures to various CEM problems. To achieve this goal, however, further research work is needed to extend the domain of efficient applicability of our approach. To this end, the discussion in Section V provides a useful framework for further application of the time-parallel computing approach.

Appendix
CN Discretization

Let D^2 represent the finite-difference approximation to the Laplace operator ∇^2. With this notation, the CN approximation to (2) at the point (i,j) in a two-dimensional space may be written as

$$\frac{\psi_{i,j}^{(m+1)} - 2\psi_{i,j}^{(m)} + \psi_{i,j}^{(m-1)}}{\Delta t^2} = \frac{c^2}{2}[D^2\psi_{i,j}^{(m+1)} + D^2\psi_{i,j}^{(m-1)}] + g_{i,j}^{(m)} \quad (39)$$

where $\psi_{i,j}^{(m)}$ represents the unknown field at time $t = m\Delta t$ at the discretization point. Combining terms with equal time indexes, (39) may be expressed in the form

$$\left[1 - \frac{c^2\Delta t^2}{2}D^2\right]\psi_{i,j}^{(m+1)} = 2\psi_{i,j}^{(m)} - \left[1 - \frac{c^2\Delta t^2}{2}D^2\right]\psi_{i,j}^{(m-1)} + g_{i,j}^{(m)}. \quad (40)$$

If we now spatially order the unknowns $\psi_{i,j}^{(m)}$ into a column vector and express the D^2 operator as the matrix $\boldsymbol{A}$, we obtain

$$(\boldsymbol{I} - \alpha\boldsymbol{A})\psi^{(m+1)} = 2\boldsymbol{I}\psi^{(m)} - (\boldsymbol{I} - \alpha\boldsymbol{A})\psi^{(m-1)} + f^{(m+1)} \quad (41)$$

where $\alpha = c^2\Delta t^2/2$ and $f^{(m+1)}$ may contain values of $\psi_{i,j}$ which are known in the domain as well as the discretized source term.

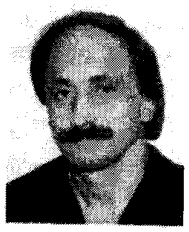
Acknowledgment

The support and encouragement of Dr. P. Messina, Director of California Institute of Technology's Center for Advanced Computing Research, is greatly acknowledged.

References

[1] E. K. Miller, "Solving bigger problems by decreasing the operation count and increasing the computation bandwidth," in *Proc. IEEE*, Special Issue on *Electromagnetics*, vol. 79, pp. 1493–1504, Oct. 1991.
[2] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. Antennas Propagat.*, vol. AP-14, pp. 302–307, May 1966.
[3] A. H. Mohammadian, V. Shankar, and W. F. Hall, "Computation of electromagnetic scattering and radiation using a time-domain finite-volume discretization procedure," *Comput. Phys. Commun.*, vol. 68, pp. 175–196, Oct. 1991.
[4] V. Shankar, A. H. Mohammadian, W. F. Hall, and R. Erickson, "CFD spinoff: Computational electromagnetics for radar cross section (RCS) studies," in *Proc. AIAA Applied Aerodynamics Conf.*, 1990.
[5] A. Taflove and K. R. Umashankar, "Review of FD-TD numerical modeling of electromagnetic wave scattering and radar cross section," in *Proc. of IEEE*. Special Issue on *Radar Cross Section of Complex Objects*, vol. 77, pp. 682–699, May 1989.
[6] R. H. Calalo *et al.*, "Hypercube matrix computation task," *JPL Pub., 88-31*, Aug. 1988.
[7] J. E. Patterson *et al.*, "Concurrent electromagnetic scattering analysis," in *Proc. AIAA Computers in Aerospace VII Conf.*, Oct. 1989.
[8] W. J. Buchanan, N. K. Gupta, and J. M. Arnold, "Simulation of radiation from a microstrip antenna using three dimensional finite-difference time-domain (FDTD) method," in *Proc. IEE 8th Int. Conf. Antennas Propagat.*, Heriot-Watt Univ., UK, 1993, pp. 639–642.
[9] H. Vinh, H. A. Dwyer, and C. P. Van Dam, "Finite-difference algorithms for time-domain Maxwell's equations—A numerical approach to RCS analysis," in *Proc. 23rd AIAA Plasmadynamics & Laser Conf.*, July 1992, pp. 1–8.
[10] J. M. Ortega and R. G. Voigt, *Solution of Partial Differential Equations on Vector and Parallel Computers*. Philadelphia, PA: SIAM, 1984.
[11] E. Gallopoulos and Y. Saad, "On the parallel solution of parabolic equations," in *Proc. ACM Int. Conf. Supercomputing*, June 1989, pp. 17–28.
[12] S. Vandewalle, R. Van Driessche, and R. Piessens, "The parallel performance of standard parabolic marching schemes," *Int. J. High Speed Computing*, vol. 3, pp. 1–29, 1991.
[13] G. Rodrigue, "A parallel first-order method for parabolic partial differential equations," in *High-Speed Computation*, J. S. Kowalik, Ed. Springer-Verlag, 1984, pp. 329–342.
[14] D. J. Evans, "Alternating group explicit methods for the diffusion equation," *Appl. Math. Modeling*, vol. 9, pp. 201–206, 1985.

[15] S. M. Serbin, "A scheme for parallelizing certain algorithms for the linear inhomogeneous heat equation," *SIAM J. Sci. Stat. Comput.*, vol. 13, pp. 449–458, Mar. 1992.
[16] D. E. Womble, "A time-stepping algorithm for parallel computers," *SIAM J. Sci. Stat. Comput.*, vol. 11, pp. 824–837, 1990.
[17] G. Horton and R. Knirsch, "A time-parallel multigrid-extrapolation method for parabolic partial differential equations," *Parallel Computing*, vol. 18, pp. 21–29, 1992.
[18] S. Vandewalle and R. Piessens, "Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic differential equations," *SIAM J. Sci. Stat. Comput.*, vol. 13, pp. 1330–1346, Nov. 1992.
[19] G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, NJ: Prentice-Hall, 1973.
[20] A. Fijany, "Time-parallel algorithms for solution of linear parabolic PDE's," in *Proc. Int. Conf. Parallel Processing (ICPP)*, vol. III, pp. 51–55, Aug. 1993.
[21] A. Fijany, J. Barhen, and N. Toomarian, "A computational strategy for exploitation of massive temporal parallelism in solution of time-dependent PDE's," in *Proc. Toward Teraflop Computing and New Grand Challenge Applications Conf.*, Baton Rouge, LA, Feb. 1994, pp. 60–71.
[22] N. Toomarian, A. Fijany, and J. Barhen, "Time parallel solution of linear partial differential equations on the Intel Touchstone Delta supercomputer," *Concurrency: Practice and Experience*, vol. 6, pp. 641–652, Dec. 1994.
[23] R. Hockney and C. Jesshope, *Parallel Computers*. Adam Hilger, 1981.
[24] D. Givoli, "Non-reflecting boundary conditions," *J. Comput. Phys.*, vol. 94, pp. 1–29, May 1991.
[25] X. Zhang, J. Fang, K. K. Mei, and Y. Liu, "Calculations of the dispersive characteristics of microstrips by the time-domain finite difference method," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 263–267, Feb. 1988.
[26] M. A. Jensen, Y. Rahmat-Samii, and A. Fijany, "A novel scheme for massively parallel solution of Maxwell's equations using FDTD," in *Proc. 11th Ann. Rev. Progress Applied Computational Electromagnetics*. Monterey, CA, Mar. 1995, pp. 592–599.
[27] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*. Philadelphia: SIAM, 1992.
[28] B. T. Smith *et al.*, *Matrix Eigensystem Routines-Eispack Guide*, 2nd ed. New York: Springer Verlag, 1976.
[29] O. A. McBryan and E. F. Van De Velde, "Hypercube algorithms and implementation," *SIAM J. Sci. Stat. Comput.*, vol. 8, pp. 227–287, Mar. 1987.
[30] P. N. Swarztrauber and R. A. Sweet, "The direct solution of discrete Poisson equation on a disk," *SIAM J. Numer. Anal.*, vol. 10, pp. 900–907, Oct. 1973.
[31] S. Pissanetzky, *Sparse Matrix Technology*. New York: Academic, 1984.



Amir Fijany (M'88) received the B.S. and M.S. degrees in electrical engineering from the University of Teheran, Iran, in 1980, and the D.E.A. and Ph.D. degrees from the University of Paris XI (Orsay), France, in 1981 and 1988, respectively.

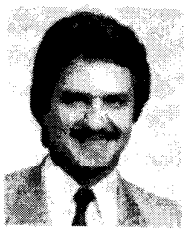
Since July 1987 he has been a Member of Technical Staff at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena. His research activities include parallel algorithms, computer architecture, and application of high performance computing to scientific and Grand Challenge problems. He has authored/coauthored over 30 articles in archival journals and conferences as well as 3 book chapters. He is the coeditor of a book entitled "Parallel Computation Systems for Robotics: Algorithms and Architectures" (World Scientific Publishing Co., 1992). He holds four (two pending) U.S. patents on the design of parallel architectures for robotics and signal processing applications.

Dr. Fijany has received numerous NASA Technical Innovation Awards. He is the recipient of two Major NASA Monetary Awards and Special Certificate of Recognition for the development of scientific contributions with significant value to the advancement of the aerospace technology program of NASA.

Michael A. Jensen (S'92–M'94) received the B.S. (summa cum laude) and M.S. degrees in electrical engineering from Brigham Young University (BYU), Provo, UT, in 1990 and 1991, respectively, and the Ph.D. in electrical engineering from the University of California, Los Angeles (UCLA) in 1994.

From 1989–1991 he was a graduate Research Assistant in the Lasers and Optics Laboratory at BYU. In 1990 he received a National Science Foundation Graduate Fellowship. From 1991–1994, he worked with the Antenna Laboratory at UCLA as a graduate Student Researcher. He is currently an Assistant Professor in the Electrical and Computer Engineering Department at BYU. His main research interests include radiation and propagation for personal communications, numerical electromagnetics, parallel processing, and optical fiber communication.

Dr. Jensen is a member of Eta Kappa Nu and Tau Beta Pi.



Yahya Rahmat-Samii (S'73–M'75–SM'79–F'85) received the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois, Champaign-Urbana.

He is a Professor of Electrical Engineering at the University of California, Los Angeles. He has been a Senior Research Scientist at NASA's Jet Propulsion Laboratory/California Institute of Technology since 1978, where he contributed significantly to the advancement of antenna technology for space programs. He was a Guest Professor at the Technical University of Denmark (TUD) in the summer of 1986. He has also been a consultant to many aerospace companies. He is listed in *Who's Who in America*, *Who's Who in Frontiers of Science and Technology*, and *Who's Who in Engineering*. He has authored or coauthored, over 320 technical journal articles and conference papers and has written one book and chapters in 13 books. He has made pioneering contributions to the developments of near-field plane-polar and bi-polar antenna measurements, microwave holographic diagnostics, mobile satellite communication antennas, reflector surface compensation, multireflector antenna diffraction analysis and synthesis, scattering and radiation from complex objects, RCS computations, singularity in dyadic Green's function, high power microwave (HPM) antennas, EMP and aperture penetration, the Spectral Theory of Diffraction (STD), and GTD.

Dr. Rahmat-Samii is a Fellow of IAE (1986) and was the 1984 recipient of the Henry Booker Award of URSI. He was appointed an IEEE Antennas and Propagation Society Distinguished Lecturer and presented lectures internationally. He was an elected IEEE AP-S AdCom member for the second term and has been an Associate Editor of the IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION and the society's *Magazine*. He is currently the President of IEEE AP-S. He was the Chairman of the IEEE Antennas and Propagation Society of Los Angeles in 1987–1989. In 1989, his chapter received the Antennas and Propagation Best Chapter Award from the AP Society. He is one of the three International Editors of the IEE book series on Electromagnetics and Antennas. He is also one of the Editors of the *Journal of Electromagnetic Waves and Applications* and *Electromagnetics*. He was one of the Directors of AMTA (Antenna Measurements Technique Association) and the Electromagnetics Society. For his contributions he has received numerous NASA Certificate of Recognitions and recently earned the JPL Team NASA's Distinguished Group Achievement Award. In 1992 and 1995, he was the recipient of the Best Application Paper Award (Wheeler Award) for a paper published in this TRANSACTIONS in 1991 and 1994. He is a member of Commissions A, B, and J of USNC/URSI, Sigma Xi, Eta Kappa Nu, and the Electromagnetics Academy.

Jacob Barhen (M'84), photograph and biography not available at the time of publication.