7-25

SANDIA REPORT

SAND95–1322 • UC–705
Unlimited Release
Printed July 1995

The Implementation of the Upwind Leapfrog Scheme for 3D Electromagnetic Scattering on Massively Parallel Computers

Brian T. Nguyen, Scott A. Hutchinson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000

Approved for public release; distribution is unlimited.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

SAND95–1322
Unlimited Release
Printed July 1995

Distribution
Category UC–705

The Implementation of the Upwind Leapfrog Scheme for 3D Electromagnetic Scattering on Massively Parallel Computers*

Brian T. Nguyen†
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109-2118

Scott A. Hutchinson‡
Massively Parallel Computing Research Laboratory
Sandia National Laboratories
Albuquerque, NM 87185

Abstract

The upwind leapfrog scheme for electromagnetic scattering is briefly described. Its application to the 3D Maxwell's time domain equations is shown in detail. The scheme's use of upwind characteristic variables and a narrow stencil result in a smaller demand in communication overhead, making it ideal for implementation on distributed memory parallel computers. The algorithm's implementation on two message passing computers, a 1024-processor nCUBE 2 and a 1840-processor Intel Paragon, is described. Performance evaluation demonstrates that the scheme performs well with both good scaling qualities and high efficiencies on these machines.

* This work was supported in part by the Computational Science Graduate Fellowship Program of the Office of Scientific Computing in the Department of Energy and by the Applied Mathematical Sciences program, U.S. Department of Energy, Office of Energy Research, and was partially performed at Sandia National Laboratories, operated for the U.S. Department of Energy under contract No. DE-AC04-94AL85000.

† Doctoral Candidate; email: bnguyen@engin.umich.edu; telephone: 313-764-5229

‡ Parallel Computational Sciences Department; email: sahutch@cs.sandia.gov; telephone: 505-845-7996

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED
MASTER

1. Introduction. The leapfrog scheme of Yee [1] has long been a popular method for solving Maxwell's equations in the time domain. In recent years, several different techniques for the solving the same classes of problems have been published, including many finite element approaches, higher order finite difference methods and upwind biased schemes. Some examples of new finite difference schemes [2, 3] use extended differencing stencils to attain higher orders. Higher order extensions to the method described in this paper are possible [4] though they are not addressed in this report. Whereas the finite element approaches have emphasized the role of mathematically better numerical approximations for the differential operators, the upwind biased schemes first interpret the phenomena as the interaction of moving waves with specific directions of propagations and then discretize the problems so that information is moved in the correct direction. Experiences in computational fluid dynamics have shown that this approach improves the solution [5], especially in the resolution of high gradients in shock waves and boundary layers. Such high gradients are not unlike those found in high frequency electromagnetic and acoustic disturbances.

Upwind biased schemes were developed for hyperbolic systems of equations whose solutions exhibit wave-like behavior. Schemes including, but not limited to, those of Gudonov [6] and Roe [7] use Riemann solvers to determine the interaction of waves at the boundaries of discrete cells in the domain and then apply flux to the cells based upon the Riemann solutions. The principle of solving the Riemann problems at cell interfaces were applied to the time domain Maxwell's equations by Shankar et. al. [8]. This method suffers a disadvantage in comparison to Yee's leapfrog scheme in that it is numerically dissipative. While numerical dissipation is acceptable in fluid dynamics where the solution varies on the problem's geometric scale it can be overwhelming in electromagnetics whose solutions vary on an independent scale such as the electromagnetic wavelength. These wave disturbances can be too dissipative if the grid resolution is too low or prohibitively expensive if the the grid resolution is made sufficient.

Additionally, as the smallest important wavelength decreases with respect to the characteristic geometric length of the problem, the computational requirements of the solution can grow to exceed the available resources of traditional serial-vector computers. Today's scalable massively parallel (MP) computers are ideal platforms for solving such large problems. However, in order to realize the efficiency of any algorithm on such computers, the overhead required for communication must not defeat the gains provided by parallel processing. Thus, it is vital that approaches are suitable for parallelization and are well designed.

In this report, we discuss the parallelization of the upwind leapfrog method, which is a combination of the traditional leapfrog and upwinding methods and can be efficiently implemented on MP computers. The scheme is introduced in Section 2, and the set of equations actually solved is presented in Section 3. The numerical implementation of the method is presented in Section 4 and specifics of the parallel implementation are given in Section 5. In Section 6, results of a benchmark problem run on two different platforms are presented and lastly, Section 7 summarizes our findings.

2. Upwind Leapfrog Scheme. The upwind leapfrog scheme was first described by Iserles [9] and recently further developed by Roe [10], Thomas and Roe [11] and Nguyen and Roe [12]. The reader is especially referred to [10] as only a brief summary follows here.

The advantages of the upwind leapfrog schemes described in references [10, 11, 12] include their lack of numerical dissipation, low numerical dispersion, their use compact stencils and upwinded discretization of the bicharacteristic equations. The last two items are especially useful for implementation on MP computers.

All leapfrog schemes are reversible in time and therefore are non-dissipative. Analyses in [11] and [10] show the numerical dispersion characteristics to be much improved over those of the original leapfrog scheme of Yee [1]. However, since Maxwell's equations for simple media decouple when this scheme is applied, only one of the decoupled sets needs to be solved. One interpretation of Yee's staggered scheme is as a leapfrog scheme where only one of the decoupled sets is solved. The staggered placement of the solution in space and in time results because the decoupled set consists of points in staggered patterns.

The upwind leapfrog scheme uses bicharacteristic forms of Maxwell's equations, which are special combinations of the original partial differential equations. A feature of the bicharacteristic equations is the inclusion of an explicit derivative in a two-dimensional spatial-temporal subspace of the $D+1$ dimensional space of a D-dimensional problem. It is the orientation of this subspace that tells the direction and speed of propagation of the information governed by a particular bicharacteristic equation. Bicharacteristic equations can be formulated for any spatial direction since a change in direction is no more than a rotation of coordinates. But, as argued in [10], many independent bicharacteristic equations can be formulated from a finite system of PDE's because each bicharacteristic applies along a particular direction (i.e. the vector $\mathbf{n}$ in (40) and (41)), not just at a single point.

The use of bicharacteristic equations have two immediate consequences: a change in the number of equations (and unknowns) and the presence of an explicit direction associated with each equation (and unknown). For Maxwell's equations in the current implementation, twelve equations result, each pair corresponding to one of the directions $\pm x$, $\pm y$, and $\pm z$. The two unknowns for each direction correspond to the two transverse directions. (See Appendix A). Reference [10] has further discussion on bicharacteristic equations and their discretization.

The uniqueness of each bicharacteristic equation to a particular direction and the maintenance of this uniqueness in discretizing them with an appropriate upwind scheme enables the present approach to propagate distinct information in a more physically consistent manner than central difference or finite element schemes.

3. Bicharacteristic Governing Equations. The twelve bicharacteristic governing equations derived in Appendix A are

$$\left[\frac{\partial}{\partial t} \pm c\frac{\partial}{\partial x}\right] G_y^{\pm x} - \frac{\partial H_x}{\partial z} \mp c\frac{\partial D_x}{\partial y} = -J_y \tag{1}$$

$$\left[\frac{\partial}{\partial t} \pm c\frac{\partial}{\partial x}\right] G_z^{\pm x} + \frac{\partial H_x}{\partial y} \mp c\frac{\partial D_x}{\partial z} = -J_z \tag{2}$$

$$\left[\frac{\partial}{\partial t} \pm c\frac{\partial}{\partial y}\right] G_x^{\pm y} + \frac{\partial H_y}{\partial z} \mp c\frac{\partial D_y}{\partial x} = -J_x \tag{3}$$

$$\left[\frac{\partial}{\partial t} \pm c\frac{\partial}{\partial y}\right] G_z^{\pm y} - \frac{\partial H_y}{\partial x} \mp c\frac{\partial D_y}{\partial z} = -J_z \tag{4}$$

$$\left[\frac{\partial}{\partial t} \pm c\frac{\partial}{\partial z}\right] G_x^{\pm z} - \frac{\partial H_z}{\partial y} \mp c\frac{\partial D_z}{\partial x} = -J_x \tag{5}$$

$$\left[\frac{\partial}{\partial t} \pm c\frac{\partial}{\partial z}\right] G_y^{\pm z} + \frac{\partial H_z}{\partial x} \mp c\frac{\partial D_z}{\partial y} = -J_y \tag{6}$$

Each equation applies in either the $\pm x$, $\pm y$ or $\pm z$ direction as indicated by the first operator.

With the exception of two additional cross terms per equation, they are very similar to the standard second order, one-dimensional advection equation for which the upwind leapfrog scheme was first developed [9]:

$$\left[\frac{\partial}{\partial t} + a\frac{\partial}{\partial x}\right] u = 0 \tag{7}$$

The cross terms are the results of the multidimensional coupling of non-planar waves or waves that are not perfectly aligned and propagating along one of the coordinate directions. As seen below, these cross terms do not present a problem when the leapfrog scheme is naturally extended to multiple dimensions. The directions of propagation are explicitly indicated by the second derivative in the first operator and the sense of propagation by the the sign. These equations are physically consistent (in a discrete sense) with the directions of propagation of the physical wave as well as the combinations of propagated variables.

4. Numerical Implementation. In the one dimensional upwind leapfrog scheme, the variables are stored at the points and the stencil is centered around a cell (Figure 1). Information traversing the cell is used to update the points forming the cell. For example, a difference centered at $j - \frac{1}{2}$ is used to update right going information at point j and a difference centered at $j + \frac{1}{2}$ is used to update left going information at the same point. The natural multidimensional extension of this is to store variables on the faces (planar segments in 3D and line segments in 2D). Figure 2 shows, for the 3D problem, how each face stores at its centroid the tangential components of the characteristic variable vectors (or alternatively, primitive variable vectors). Four variables are stored

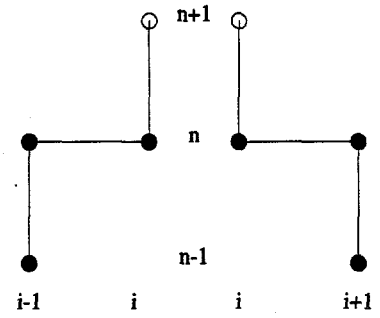

FIG. 1. *Upwind Leapfrog stencils in 1D*

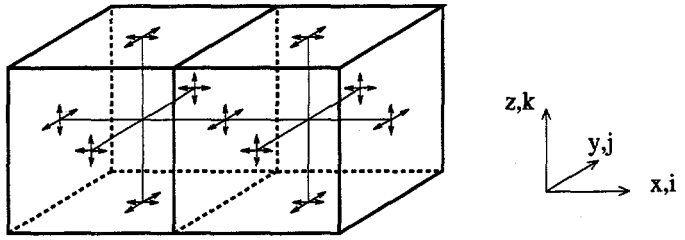

FIG. 2. *Upwind Leapfrog Stencils in 3D*

at each face, two tangential components for each direction a wave can pass through the face.

With the cell centered scheme described in the next section, only the solution at two time steps needs to be stored since the solution at step $n-1$ can be discarded just prior to storing the solution at step $n+1$. However, the current code is written to store three time steps in order to facilitate ongoing experimentations.

The bicharacteristic equations operate directly on the characteristic variables. To save having to convert between primitive and characteristic variables at every time step, the characteristic variables are stored. Conversion to primitive variables is still required for the cross terms in (1), but no reverse conversion is necessary after the characteristic variables are updated.

The lines connecting opposing face centroids in Figure 2 indicate where differencing takes place. These cell centered differences are used to update the solutions on each face. Time differencing is taken as the average of two differences, the forward difference between step $n+1$ and step n and the backward difference between step n and step $n-1$, shown in Figure 1. For example, (1) can be discretized as

$$(8)\quad \frac{G_i^{n+1} - G_i^n + G_{i\mp1}^n - G_{i\mp1}^{n-1}}{2\Delta t} + c\frac{G_i^n - G_{i\mp1}^n}{\Delta x} - \left[\frac{\partial H_x}{\partial z}\right]_{i\mp\frac{1}{2}}^n \mp \left[\frac{\partial D_x}{\partial y}\right]_{i\mp\frac{1}{2}}^n = -J_{y,i\mp\frac{1}{2}}^n$$

where the super- and subscripts on $G_y^{\pm D_x}$ have been omitted to reduce clutter. This approximation to (1) is second order accurate in space and time. Note that the half-integer indexed derivatives are obtained by cell centered central differencing along the legs directed in the y- and z-axis (Figure 2). The half-integer indexed current density can be taken as the cell averaged current density.

Within each cell, the algorithm determines how the incoming waves interact as they traverse the cell. The scattered waves then become outgoing waves to be updated on the faces. Each cell can only update the variables on the insides of its constituent faces and relies on neighboring cells to update the other side.

The cell centered update approach was chosen because it can be easily vectorized and it minimizes the amount of data that must be communicated between processors in distributed computers. Each face borders two cells and is updated by both, but the variables updated by each cell are independent, thus no vector dependencies exist which might inhibit vectorization.

5. Parallel Implementation. For two major reasons, the second order upwind leapfrog scheme is well suited for implementation on massively parallel message passing computers: 1) it uses a narrow non-overlapping stencil and 2) it "knows" the direction of wave propagation.

The method's narrow, non-overlapping stencil allows the grid to be partitioned along a plane of faces such that no stencil straddles a processor boundary. Larger stencils such as Deveze's extension to Yee's scheme [2] which spans three stencils in space, would require several layers of data to be exchanged between neighboring processors. Additionally, large stencils that require data from diagonally located neighbors also require processors to communicate with diagonal neighbors, increasing the number of communication calls from 6 to 26 in 3D.

The ability to work directly with characteristic variables also helps to minimize communication overhead. Since each variable propagates in a definite direction from one side of a face to the other and the grid can be partitioned at the faces, only variables known to propagate across faces on the partition boundary need to be communicated. In the present 3D implementation, although 12 variables are solved per cell, only two variables are sent and two variables are received per cell on the partition interface (except, of course, cells on the corners and edges of partitions).

5.1. Partitioning Methods. Given the 3D monolithic grid of the domain to be simulated, the partitioning process divides it by making cuts along constant index planes and assigning a subset to each processor. Figure 3 shows this in 2D, where the dividing planes are represented by the lines of medium thickness. Notice that each face stores variables that come from each side, as indicated by the circles lying on both sides of each face. Each processor updates its interior data, including the interior data of its boundary faces. Exterior data are obtained from the domain boundary condition or the neighboring processors.

The processors are logically arranged into a "processor grid" which is a 3D grid mapping dimension for dimension onto the 3D problem grid. Each partition in the 3D analogy of Figure 3 represents a point on the partition grid. Each processor handles the partition of the domain grid that maps directly to its location in the processor grid.

The first stage of the partitioning determines the dimension of the processor grid, that is, how many layers of processors are to be in the i, j and k directions given the number of processors available. The second stage is to identify where in the processor

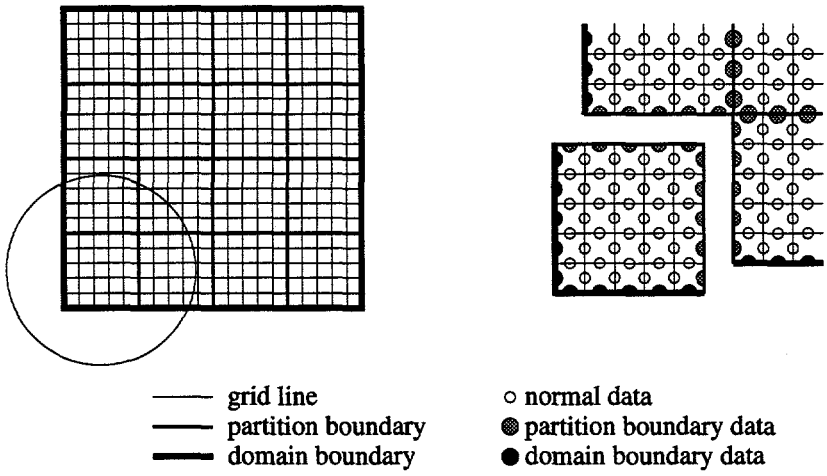

FIG. 3. *2D Domain Partitioning*

grid each processor belongs. This essentially requires a simple heuristic to assign the processor grid coordinates to each processor. Finally, the ranges of indices for which each processor is responsible are computed.

The first and second stages are performed differently depending on the communication architecture of the target MP machine. The code has been implemented on the nCUBE 2 and the Intel Paragon and the methods which take advantage of each computer's specific architecture are described separately below. The third stage is a relatively trivial but tedious procedure. It is only important to note that in each of the i, j and k directions, the thickest and thinnest partitions differ by no more than one layer of cells. This creates an optimal load balance for a partitioning scheme that uses constant index planar cuts to divide the domain grid. In the worst case, this approach yields a ratio of the heaviest loaded processor's work to the lightest loaded processor's work of

$$\frac{(S_i+1)(S_j+1)(S_k+1)}{S_iS_jS_k}$$

where S_l is the thickness of the smallest slice in the l direction.

5.1.1. nCUBE 2. The nCUBE 2 computer uses a hypercube communication interconnect. For an h-dimensional hypercube, the available number of processors is 2^h. Each additional dimension of the hypercube doubles the number of processors.

The procedure to determine the size of the processor grid begins with a zero-dimensional hypercube (one processor). For each additional hypercube dimension, the computational dimension (i, j or k) with the greatest ratio

$$r = \frac{\text{domain grid size}}{\text{processor grid size}} \tag{9}$$

is selected to receive the doubling of processor grid size.

Once the dimensions of the processor grid is determined, the processor's position in the processor grid is found by the standard gray code algorithm implemented in the

function *nodetogrid* in the nCUBE 2 parallel library. This algorithm takes advantage of the hypercube and assures that neighboring partitions in the domain grid are neighbors in the hardware, minimizing contention for message passing routes and the distance (hops) individual messages must travel.

5.1.2. Paragon. The Intel Paragon computer provides a $b \times h$ rectangular array of processors. In this two-dimensional *mesh* architecture, a processor has at most four immediate neighbors and communication with other processors requires the message to travel through multiple processors (hops). Thus, an optimal mapping of the problem grid to the processor mesh is not easily found.

Determining the size of the (3D) processor grid proceeds similarly to the procedure used for the nCUBE 2 but instead of doubling the dimension with the largest r, this dimension is increased by just one layer. The number of processors required for each additional layer is the product of the remaining two dimensions of the current processor grid. If this procedure exceeds the bh processors available without hitting it exactly, the subroutine cannot proceed. It suggests a more compatible value for bh and returns.

Each processor's position in the processor grid is determined by natural ordering, thus there is no useful relationship established between the rectangular processor array and the 3D processor grid. This can potentially lead to contention, but the Paragon under SUNMOS has sufficient bandwidth (approximately 100 to 150 MB per second) so that no significant contention should occur. The aspect ratio of the processor array may also induce contention, therefore it is kept as square as is possible. This is limited by a maximum height of $h = 16$ for each Paragon cabinet.

5.2. Parallel Algorithm. A serial implementation of the present scheme would involve only a normal update followed by boundary condition setting and source term effects. In the parallel implementation, these steps are followed by the boundary data exchange step. Each processor gathers into a buffer data needed by its neighbors and send that data to the appropriate neighbor. It then receives the data from its neighbors.

Across each face, four characteristic variables are propagated, two transverse components each way. The faces lying on the partition interfaces are the only ones that require data from the adjacent partition (Figure 3). On these faces, the process updates the two characteristic variables coming from inside the cell. After updating, the boundary data exchange procedure provides the remaining two variables.

A pseudocode for the update procedure for each partition is as follows:

```
For each update {

  For all cells of the partition {
    Update outgoing variables on faces of cell
  }

  Add source term effects to updated solution
```

```
For all sides of the partition {
  If the side is a computational boundary {
    set boundary condition as normal
  }
  Else {
    Copy information leaving the partition through the boundary
    Send (non-blocking) information to neighboring partition
  }
}

While there is an unresolved partition boundary {
  Receive information coming from any neighbor partition
  Decide where the data should go
  Copy received information into appropriate solution memory
  }
}
```

6. Performance Results. As mentioned above, the computers used in this study were the nCUBE 2 and the Intel Paragon at the Massively Parallel Computing Research Laboratory at Sandia National Laboratories. On the Intel Paragon, all runs were made using the SUNMOS operating system developed at Sandia.

6.1. Test Problem. The test problem chosen to evaluate the performance of the implementation is that of a radiating dipole. As written, the code is capable of simulating electromagnetic scattering in linear, isotropic, inhomogeneous (cellwise homogeneous) media. However, free space was chosen as the medium. The dipole was placed in one of the partitions while the remaining partitions remain source free. At the domain grid boundary, the incoming characteristic variables are set to zero. The main update section requires 131 floating point operations (flops) per cell per time step.

Two problem sizes were used in this study, one fixed size problem and one scaled size problem. In the fixed size problem, the maximum size that can fit on one processor was determined. This size was fixed as the problem was run on an increasing number of processors. In the scaled size problem, the maximum size that can fit on one processor was used for a single processor run. For multiprocessor runs, the problem size was increased proportional with the number of processors.

The time of each run was determined by a timing the entire integration loop 512 time steps. This loop does not include initialization procedures such as allocating the processors, reading input, allocating memory, etc. During the runs, no residuals were computed and no input/output operations were called. The processors were all synchronized before the main integration loop and no explicit synchronizations were performed during the loop. The functions *clock* and *dclock* were used to mark the times during the runs on the nCUBE 2 and the Paragon, respectively.

Two measures of parallel efficiency were computed as follows. The flop rate is computed from the estimate of flops performed in the solver and boundary condition

handler and the time required. The speed-up ratio is defined as

$$S_p = \frac{\text{flop rate for } p \text{ processors}}{\text{flop rate for 1 processor}} \tag{10}$$

and the parallel efficiency is defined as

$$E_p = \frac{S_p}{p} \tag{11}$$

where p is the number of processors.

6.2. nCUBE 2 Results. For the constant size problem, a grid of 19x19x19 cells was used to utilize the 3.75 MB available on each processor. The scaled size problem was run with a partition of 19x19x19 cells per processor. Tables 1 and 2 show, respectively, the results for the nCUBE 2.

number of processors	grid size (cells)	time (sec.)	flop rate (Mflop/s)	S_p	E_p
1	19x19x19	462.2	.996	1.0000	1.0000
2	19x19x19	246.7	1.865	1.8738	.9369
4	19x19x19	131.3	3.505	3.5207	.8802
8	19x19x19	70.3	6.540	6.5704	.8213
16	19x19x19	36.4	12.635	12.6938	.7934
32	19x19x19	19.1	24.073	24.1839	.7557
64	19x19x19	10.4	44.266	44.4717	.6949
128	19x19x19	6.7	68.365	68.6819	.5366

TABLE 1
nCUBE constant size problem results

The efficiencies for the constant size problem drop quickly (Figure 4), as expected, because the processors are not fully utilized. For the scaled size problem, efficiency remained above 97.5% for all the runs up to 1024 processors and drops slightly below 97.5% for 1024 processors (Figure 5). The maximum computational speed achieved by utilizing 1024 processors was 993 Mflops/s. In comparison, the maximum computation rate for double precision mathematics on this machine is 2.5 Gflops/s. Figure 5 shows that the reduction in parallel efficiency tapers off with the number of processors. The addition of more processors should incur reduced communication overhead according to the trend of Figure 5.

6.3. Intel Paragon Results. For the constant size problem, a grid of 33x33x34 cells was used needing roughly 16 MBytes of memory on a single node. The scaled size problem was run with a partition of 33x33x34 cells per processor. Tables 3 and 4 show, respectively, the results for the Paragon runs.

For the constant size problem, the inefficient use of the processors is evident in E_p as contention dominates, leading to irregular results such as the parallel efficiencies for

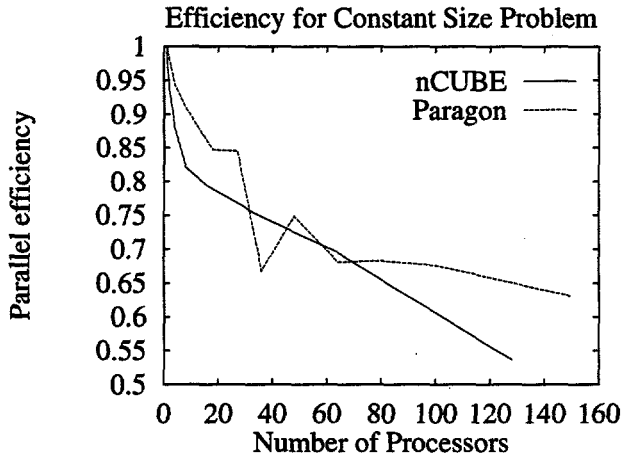

FIG. 4. *Parallel Efficiency of Constant Size Problem*

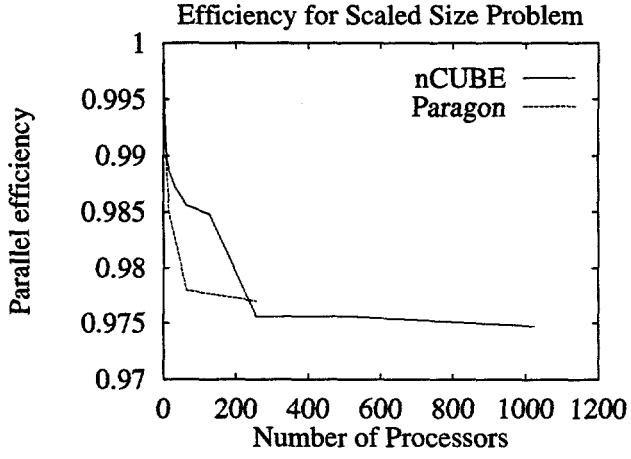

FIG. 5. *Parallel Efficiency of Scaled Size Problem*

number of processors	grid size (cells)	time (sec.)	flop rate (Mflop/s)	S_p	E_p
1	19x19x19	462.2	.996	1.0000	1.0000
2	19x19x38	463.3	1.986	1.9952	.9976
4	19x38x38	465.0	3.957	3.9756	.9939
8	38x38x38	466.7	7.886	7.9228	.9904
16	38x38x76	468.4	15.715	15.8198	.9887
32	38x76x76	470.0	31.323	31.5910	.9872
64	76x76x76	471.7	62.423	63.0812	.9856
128	76x76x152	471.7	124.485	126.0483	.9848
256	76x152x152	473.7	248.610	249.7636	.9756
512	152x152x152	473.7	497.210	499.5195	.9756
1024	152x152x304	474.2	993.478	998.1084	.9747

TABLE 2
nCUBE scaled size problem results

48, 64 80 and 100 processors being greater than that for 36 processors (Table 1 and Figure 4).

The more realistic usage is illustrated in the scaled size problem (Table 4 and Figure 5), where each processor's memory is fully utilized. Here, the efficiency remains above 97.5% up to 256 processors. More importantly, efficiency does not decrease as the number of processors increase. A maximum flop rate of 1.76 Gflops/s. is achieved with 256 processors.

Like the behavior on the nCUBE 2, reduction in parallel efficiency tapers off quickly so that further parallelism does not incur significant additional communication overhead (Figure 5).

7. Summary and Conclusions. The application of the second order leapfrog scheme to the 3D time domain Maxwell's equations was presented along with its implementation on MP message passing computers. In addition to the previously demonstrated ability to efficiently propagate disturbances with low numerical dispersion, this scheme is now shown to have excellent parallel features. The test results presented show that the implementation's efficiency remained close to or above 97.5% for machine sizes of up to 1024 processors for the nCUBE 2 computer and 256 processors for the Intel Paragon. Further, scaled speed-up values of 998 for 1024 nCUBE 2 processors and 250 for 256 Intel Paragon processors were also demonstrated. These results were obtained using scaled problems which used the maximum memory capacities of 3.75 MBytes per processor on the nCUBE 2 and the 16 MBytes per processor on the Intel Paragon.

processor array size	grid size (cells)	time (sec.)	flop rate (Mflop/s)	S_p	E_p
1x1	33x33x34	353.72	7.02	1.00	1.000
1x2	33x33x34	179.67	13.82	1.97	0.984
2x2	33x33x34	93.67	26.51	3.78	0.944
2x4	33x33x34	48.65	51.05	7.27	0.909
3x6	33x33x34	23.21	106.97	15.24	0.847
3x9	33x33x34	15.50	160.24	22.24	0.845
6x6	33x33x34	14.72	168.69	24.03	0.668
6x8	33x33x34	9.84	252.34	35.95	0.749
8x8	33x33x34	8.12	305.58	43.55	0.681
8x10	33x33x34	6.47	383.51	54.66	0.683
10x10	33x33x34	5.24	473.97	67.55	0.676
10x15	33x33x34	3.73	664.54	94.72	0.631

TABLE 3
Paragon constant size problem results

processor array size	grid size (cells)	time (sec.)	flop rate (Mflop/s)	S_p	E_p
1x1	33x33x34	353.17	7.03	1.00	1.000
2x2	33x66x68	356.09	27.90	3.97	0.992
4x4	66x66x136	358.60	110.80	15.76	0.985
8x8	132x132x136	361.22	440.00	62.57	0.978
16x16	132x264x272	361.29	1759.69	250.25	0.977

TABLE 4
Paragon scaled size problem results

A. Derivation of Bicharacteristic Governing Equations. The following derivation of the bicharacteristic form of Maxwell's equations assumes a linear, isotropic nondispersive media. It uses a well known characteristics analysis procedure involving an eigensystem analysis. In addition, we define a set of characteristic variables and formulate of the bicharacteristic equations.

A.1. Eigensystem Analysis. The common form of Maxwell's equations is

$$\frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} = \nabla \times \mathbf{H} \tag{12}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \tag{13}$$

In the form of a single vector, Maxwell's equations can be written as

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial K}{\partial z} = S. \tag{14}$$

where

$$Q = \begin{pmatrix} B_x \\ B_y \\ B_z \\ D_x \\ D_y \\ D_z \end{pmatrix} \tag{15}$$

$$F = \begin{pmatrix} 0 \\ -E_z \\ E_y \\ 0 \\ H_z \\ -H_y \end{pmatrix} \tag{16}$$

$$G = \begin{pmatrix} E_z \\ 0 \\ -E_x \\ -H_z \\ 0 \\ H_x \end{pmatrix} \tag{17}$$

$$K = \begin{pmatrix} -E_y \\ E_x \\ 0 \\ H_y \\ -H_x \\ 0 \end{pmatrix} \tag{18}$$

$$S = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -J_x \\ -J_y \\ -J_z \end{pmatrix} \tag{19}$$

This vector equation can be written as a one dimensional equation for a single wave propagating along any unit vector **n**:

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial n} = S \tag{20}$$

where now F is defined as

$$F = \begin{pmatrix} E_z n_y - E_y n_z \\ E_x n_z - E_z n_x \\ E_y n_x - E_x n_y \\ H_y n_z - H_z n_y \\ H_z n_x - H_x n_z \\ H_x n_y - H_y n_x \end{pmatrix} \tag{21}$$

Equation (20) is a one-dimensional equation along any arbitrary direction. Equation (14) can be recovered from (20) by using the chain rule relationships

$$\frac{\partial}{\partial x} = n_x \frac{\partial}{\partial n} \tag{22}$$

$$\frac{\partial}{\partial y} = n_y \frac{\partial}{\partial n} \tag{23}$$

$$\frac{\partial}{\partial z} = n_z \frac{\partial}{\partial n} \tag{24}$$

The Jacobian matrix of F in (21) with respect to independent variables Q in (15) is

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & -n_z/\epsilon & n_y/\epsilon \\ 0 & 0 & 0 & n_z/\epsilon & 0 & -n_x/\epsilon \\ 0 & 0 & 0 & -n_y/\epsilon & n_x/\epsilon & 0 \\ 0 & n_z/\mu & -n_y/\mu & 0 & 0 & 0 \\ -n_z/\mu & 0 & n_x/\mu & 0 & 0 & 0 \\ n_y/\mu & -n_x/\mu & 0 & 0 & 0 & 0 \end{pmatrix} \tag{25}$$

where ϵ and μ are the permittivity and permeability of the medium.

This matrix (25) has the eigenvalues

$$\lambda = 0, 0, c, c, -c, -c \tag{26}$$

where $c = 1/\sqrt{\epsilon\mu}$ is the speed of light.

The eigenvectors for the two stationary waves, associated with $\lambda = 0$ and called the $D0$ and $B0$ waves, are simple. The right eigenvector of the $B0$ wave and the left eigenvector of the $D0$ wave are linearly dependent and can be written

$$r^{B0} = l^{D0} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ n_x \\ n_y \\ n_z \end{pmatrix} \tag{27}$$

and similarly, for the right eigenvector of the $D0$ wave and the left eigenvector of the $B0$ wave, we have

$$r^{D0} = l^{B0} = \begin{pmatrix} n_x \\ n_y \\ n_z \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{28}$$

Note that the right-left eigenvector pairs are orthonormal for both of the stationary waves.

The remaining eigenvalues belong to four moving waves, two along $+c$ and two along $-c$. Since these waves must be transverse, the two waves in each direction must correspond to two field directions orthogonal to $\mathbf{n}$.

The eigenvectors corresponding to the moving waves are more difficult to express. Since each eigenvalue is repeated, each of their eigenvectors can lie in a two-dimensional subspace of the six-dimensional space of the unknowns. This leads to a certain degree of ambiguity in writing the eigenvectors, beyond the fact that they can vary within an arbitrary multiplicative constant. Some ways of writing the left eigenvectors are

$$l^{\pm D_x} = \begin{pmatrix} 0 \\ \pm Y n_z \\ \mp Y n_y \\ n_y^2 + n_z^2 \\ -n_x n_y \\ -n_x n_z \end{pmatrix}, \quad l^{\pm D_y} = \begin{pmatrix} \mp Y n_z \\ 0 \\ \pm Y n_x \\ -n_x n_y \\ n_x^2 + n_z^2 \\ -n_y n_z \end{pmatrix}, \quad l^{\pm D_z} = \begin{pmatrix} \pm Y n_y \\ \mp Y n_x \\ 0 \\ -n_x n_z \\ -n_y n_z \\ n_x^2 + n_y^2 \end{pmatrix} \tag{29}$$

$$l^{\pm B_x} = \begin{pmatrix} n_y^2 + n_z^2 \\ -n_x n_y \\ -n_x n_z \\ 0 \\ \mp Z n_z \\ \pm Z n_y \end{pmatrix}, \quad l^{\pm B_y} = \begin{pmatrix} -n_x n_y \\ n_x^2 + n_z^2 \\ -n_y n_z \\ \pm Z n_z \\ 0 \\ \mp Z n_x \end{pmatrix}, \quad l^{\pm B_z} = \begin{pmatrix} -n_x n_z \\ -n_y n_z \\ n_x^2 + n_y^2 \\ \mp Z n_y \\ \pm Z n_x \\ 0 \end{pmatrix} \tag{30}$$

where the upper signs correspond to the $+c$ eigenvalue and the lower signs correspond to the $-c$ eigenvalue. The $\pm$ signs in the superscripts indicate the same correspondence. The designations "D" and "B" will lead to different characteristic variable vectors defined below, and the x, y and z subscripts in the superscripts will lead to orthogonal components of the characteristic variable vectors, but for now, these are simply different names for the different vectors. Note that six different left eigenvectors are offered for each wave, but only two are linearly independent.

Similarly, some ways of writing the right eigenvectors of the moving waves are

$$r^{\pm B_x} = \begin{pmatrix} n_y^2 + n_z^2 \\ -n_x n_y \\ -n_x n_z \\ 0 \\ \mp Y n_z \\ \pm Y n_y \end{pmatrix}, \quad r^{\pm B_y} = \begin{pmatrix} -n_x n_y \\ n_x^2 + n_z^2 \\ -n_y n_z \\ \pm Y n_z \\ 0 \\ \mp Y n_x \end{pmatrix}, \quad r^{\pm B_z} = \begin{pmatrix} -n_x n_z \\ -n_y n_z \\ n_x^2 + n_y^2 \\ \mp Y n_y \\ \pm Y n_x \\ 0 \end{pmatrix} \tag{31}$$

$$r^{\pm D_x} = \begin{pmatrix} 0 \\ \pm Z n_z \\ \mp Z n_y \\ n_y^2 + n_z^2 \\ -n_x n_y \\ -n_x n_z \end{pmatrix}, \quad r^{\pm D_y} = \begin{pmatrix} \mp Z n_z \\ 0 \\ \pm Z n_x \\ -n_x n_y \\ n_x^2 + n_z^2 \\ -n_y n_z \end{pmatrix}, \quad r^{\pm D_z} = \begin{pmatrix} \pm Z n_y \\ \mp Z n_x \\ 0 \\ -n_x n_z \\ -n_y n_z \\ n_x^2 + n_y^2 \end{pmatrix}. \tag{32}$$

where the same conventions described above have been followed.

A.2. Characteristic Variables. Despite the above ambiguity in the choices of eigenvectors, it is possible to resolve electromagnetic disturbances into waves by considering inner products of any left eigenvector with the vector Q.

Since the given left and right eigenvectors of the stationary waves are orthonormal, the strengths of these waves are the inner products of the the left eigenvectors with Q, and the distribution of each wave in the space of Q is proportional to its right eigenvector.

Upon premultiplying (20) by l^{B0} and by l^{D0}, and using the definitions

$$G^{D0} = \mathbf{n} \cdot \mathbf{D} = D_n \tag{33}$$

and

$$G^{B0} = \mathbf{n} \cdot \mathbf{B} = B_n \tag{34}$$

we get

$$\frac{\partial}{\partial t} G^{B0} = 0 \tag{35}$$

$$\frac{\partial}{\partial t} G^{D0} = -\mathbf{n} \cdot \mathbf{I} \tag{36}$$

stating that a locally one-dimensional wave must be transverse since components of B and D along $\mathbf{n}$ are steady, except for the electric field from the divergence of the current field.

The characteristic variable associated with the moving waves are determined by the inner product of the above left eigenvectors. Since only two can be independent,

the choice of eigenvectors must be chosen carefully to obtain a concise definition for the characteristic variable vector. The eigenvectors designated with superscripts "D" will be used here. Mixing arbitrary combinations of valid eigenvectors will produce results that are difficult to interpret.

Due to the similarity between the $+c$ and $-c$ waves, it is possible to treat them simultaneously. We premultiply Q with the left eigenvectors in (29), obtaining the wave strengths

$$G_x^{\pm D} = l^{\pm D_x} \cdot Q = (n_y^2 + n_z^2)D_x - n_x(n_y D_y + n_z D_z) \mp Y(n_y B_z - n_z B_y) \tag{37}$$

$$G_y^{\pm D} = l^{\pm D_y} \cdot Q = (n_x^2 + n_z^2)D_y - n_y(n_x D_x + n_z D_z) \mp Y(-n_x B_z + n_z B_x) \tag{38}$$

$$G_z^{\pm D} = l^{\pm D_z} \cdot Q = (n_x^2 + n_y^2)D_z - n_z(n_x D_x + n_y D_y) \mp Y(n_x B_y - n_y B_x) \tag{39}$$

which are the components of the "characteristic variable vector"

$$\mathbf{G}^{D\pm} = \mathbf{D}_t \mp Y\mathbf{n} \times \mathbf{B} \tag{40}$$

where the subscript t stands for the transverse (to $\mathbf{n}$) components. Again, of equations (37) through (39), only two equations for the c^+ characteristic and two for the c^- characteristic can be used. By definition, $\mathbf{G}^{D+}$ is transverse to $\mathbf{n}$. It actually represents not one but two waves, since the two transverse components are independent.

In the above, the l^D eigenvectors have been used. Working with the l^B eigenvectors instead would yield different wave strengths and the characteristic variable vector

$$\mathbf{G}^{B\pm} = \mathbf{B}_t \pm Z\mathbf{n} \times \mathbf{D} \tag{41}$$

but the results, in terms of the vectors $\mathbf{D}$ and $\mathbf{B}$ would be the same. Additionally, one can show

$$\mathbf{n} \times \mathbf{G}^{B\pm} = \mp Z\mathbf{G}^{D\pm} \tag{42}$$

Normally, the changes to Q by any wave are determined by the right eigenvector of that wave, if the set of right and left eigenvectors are chosen to be orthonormal. Due to the arbitrariness mentioned above, this would be difficult in the present case. However, the distribution of changes in Q space can be directly backed out by the definitions of the characteristic variables (33, 34, 40).

Directly from (35) and (36), we know that

$$\Delta D_n = \Delta B_n = 0 \tag{43}$$

as expected of a transverse wave. From here on, we will drop the "Δ". All appropriate quantities are understood to be changes.

Knowing the solutions on the right and left sides of the face, we compute two tangential components of

$$\mathbf{G}_R^{D-} = \mathbf{D}_{tR} + Y\mathbf{n} \times \mathbf{B}_R \tag{44}$$

and two tangential components of

$$\mathbf{G}_L^{D+} = \mathbf{D}_{tL} - Y\mathbf{n} \times \mathbf{B}_L \tag{45}$$

These vector variables are tangent to the face, by (43). From these, we find

$$\mathbf{D}_t = \frac{1}{2}(\mathbf{G}_L^{D+} + \mathbf{G}_R^{D-}) \tag{46}$$

$$\mathbf{B}_t = \frac{Z}{2}\mathbf{n} \times (\mathbf{G}_L^{D+} - \mathbf{G}_R^{D-}) \tag{47}$$

on the face without having to use the right eigenvectors. Using the right eigenvectors would yield the same results.

One may attempt to rewrite (46) and (47) to reveal the right eigenvectors (by splitting the contributions from the $+c$ and $-c$ waves). Combining these equations, one gets

$$\begin{pmatrix} B_x \\ B_y \\ B_z \\ D_x \\ D_y \\ D_z \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{D} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} Z(n_y G_z^{D+} - n_z G_y^{D+}) \\ Z(n_z G_x^{D+} - n_x G_z^{D+}) \\ Z(n_x G_y^{D+} - n_y G_x^{D+}) \\ G_x^{D+} \\ G_y^{D+} \\ G_z^{D+} \end{pmatrix} \tag{48}$$

which is the weighted sum of the three possibilities in (32). But of the three weights, G_x^{D+}, G_y^{D+} and G_z^{D+}, only two are independent wave strengths. It is impossible, but fortunately not important, to separate the vector in (48) into two contributions of the two independent waves without diminishing its generality for arbitrary $\mathbf{n}$.

A.3. Derivation of Bicharacteristic Forms. Knowing the variable vector $\mathbf{G}^{D\pm}$ given by (40) and the associated direction vector $\mathbf{n}$ is equivalent to knowing $\mathbf{D}$ and $\mathbf{B}$. A set of equations governing the components of $\mathbf{G}^{D\pm}$ can be found by taking the appropriate combinations of equations that govern the variables appearing in $\mathbf{G}^{D\pm}$. As suggested by (40), combining the transverse (to $\mathbf{n}$) part of (12) with the cross product of $\mathbf{n}$ and (13) yields an equation governing $\mathbf{G}^{D\pm}$. Upon simplification this equation is

$$\left[\frac{\partial}{\partial t} + c\frac{\partial}{\partial n}\right] \mathbf{G}^{D\pm} + \mathbf{J} = c[\nabla_t(\mathbf{n} \cdot \mathbf{D}) - Y\mathbf{n} \times \nabla(\mathbf{n} \cdot \mathbf{B})] \tag{49}$$

where ∇_t is the gradient in the transverse plane.

To span a d-dimensional space, a minimum of d linearly independent directions must be taken by $\mathbf{n}$. For the cartesian grid used, natural choices are the coordinate directions. For each of these directions, there are two vector variables, one propagating forward and one backward. Each vector variable has $d-1$ transverse components. Therefore, in a 3D, directional splitting scheme, there are twelve equations governing

twelve different scalar components of characteristic variable vectors. They are obtained by letting $\mathbf{n}$ be $\pm\mathbf{x}$, $\pm\mathbf{y}$ and $\pm\mathbf{z}$ in equation 49. These are shown here, expanded to reveal the primitive variables:

$$\frac{\partial}{\partial t}\left(D_y \pm \frac{H_z}{c}\right) \pm c\frac{\partial}{\partial x}\left(D_y \pm \frac{H_z}{c}\right) - \frac{\partial H_x}{\partial z} \mp c\frac{\partial D_x}{\partial y} = -J_y \tag{50}$$

$$\frac{\partial}{\partial t}\left(D_z \mp \frac{H_y}{c}\right) \pm c\frac{\partial}{\partial x}\left(D_z \mp \frac{H_y}{c}\right) + \frac{\partial H_x}{\partial y} \mp c\frac{\partial D_x}{\partial z} = -J_z \tag{51}$$

$$\frac{\partial}{\partial t}\left(D_x \mp \frac{H_z}{c}\right) \pm c\frac{\partial}{\partial y}\left(D_x \mp \frac{H_z}{c}\right) + \frac{\partial H_y}{\partial z} \mp c\frac{\partial D_y}{\partial x} = -J_x \tag{52}$$

$$\frac{\partial}{\partial t}\left(D_z \pm \frac{H_x}{c}\right) \pm c\frac{\partial}{\partial y}\left(D_z \pm \frac{H_x}{c}\right) - \frac{\partial H_y}{\partial x} \mp c\frac{\partial D_y}{\partial z} = -J_z \tag{53}$$

$$\frac{\partial}{\partial t}\left(D_x \pm \frac{H_y}{c}\right) \pm c\frac{\partial}{\partial z}\left(D_x \pm \frac{H_y}{c}\right) - \frac{\partial H_z}{\partial y} \mp c\frac{\partial D_z}{\partial x} = -J_x \tag{54}$$

$$\frac{\partial}{\partial t}\left(D_y \mp \frac{H_x}{c}\right) \pm c\frac{\partial}{\partial z}\left(D_y \mp \frac{H_x}{c}\right) + \frac{\partial H_z}{\partial x} \mp c\frac{\partial D_z}{\partial y} = -J_y \tag{55}$$

REFERENCES

[1] K. S. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media," *IEEE Transactions on Antennas and Propagation*, vol. AP-14, pp. 302–307, May 1966.

[2] T. Deveze, "High Order F.D.T.D. algorithm to Reduce Numerical Dispersion and Staircasing," in *10th Annual Review of Progress in Applied Computational Electromagnetics*, pp. 61–68, Applied Computational Electromagnetics Society, March 1994.

[3] F. Q. Hu, M. Y. Hussaini, and J. Manthey, "Low-Dissipation and -Dispersion Runge-Kutta Schemes for Computational Acoustics," Tech. Rep. 94-65, ICASE, July 1994.

[4] J. P. Thomas, C. Kim, and P. L. Roe, "Progress Towards a New Computational Scheme for Aeroacoustics," AIAA Paper 95-1758, 1995.

[5] B. van Leer, J. L. Thomas, and P. L. Roe, "A Comparison of Numerical Flux Formulas for the Euler and Navier-Stokes Equations," AIAA Paper 87-1104, 1987.

[6] S. K. Godunov, "A finite-difference method for the numerical computation and discontinuous solutions of the equations of fluid dynamics," *Matematicheskii Sbornik*, vol. 47, pp. 271–306, 1959.

[7] P. L. Roe, "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes," *Journal of Computational Physics*, vol. 43, pp. 357–3762, 1981.

[8] V. Shankar, A. Mohammadian, W. Hall, and R. Erickson, "CFD Spinoff – Computational Electromagnetics for Radar Cross Section (RCS) Studies," AIAA Paper 90-3055, 1990.

[9] A. Iserles, "Generalised leapfrog schemes," *IMA Journal of Numerical Analysis*, vol. 6, 1986.

[10] P. L. Roe, "Linear Bicharacteristic Schemes without Dissipation," Tech. Rep. 94-65, ICASE, July 1994.

[11] J. P. Thomas and P. L. Roe, "Development of Non-Dissipative Numerical Schemes for Computational Aeroacoustics," AIAA Paper 93-3382, 1993.

[12] B. T. Nguyen and P. L. Roe, "Application of an Upwind Leapfrog Method for Electromagnetics," in *10th Annual Review of Progress in Applied Computational Electromagnetics*, pp. 446–458, Applied Computational Electromagnetics Society, March 1994.

EXTERNAL DISTRIBUTION:

William Camp
Cray Research Park
655F Lone Oak Drive
Eagan, MN 55121

Steven P. Castillo
Klipsch Dept. of Electrical & Computer Engineering
New Mexico State University
Box 30001
Las Cruces, NM 88003-0001

Horst Gietl
nCUBE Deutschland
Hanauer Str. 85
8000 Munchen 50
Germany

Satya Gupta
Intel SSD
Bldg. CO6-09, Zone 8
14924 NW Greenbrier Parkway
Beaverton, OR 97006

Barbara Helland
Computational Science Graduate Fellowship Program
124 Wilhelm Hall
Ames, IA 50011-3020

Mike Heroux
Cray Research Park
655F Lone Oak Drive
Eagan, MN 55121

Fred Howes
US Department of Energy
OSC, ER-30, GTN
Washington, DC 20585

Kwong T. Ng
Klipsch Dept. of Electrical & Computer Engineering
New Mexico State University
Box 30001
Las Cruces, NM 88003-0001

Brian T. Nguyen (10)
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109-2118

Elliott Schulman
nCUBE Corp.
3575 9th St.
Boulder, Co. 80304

INTERNAL DISTRIBUTION:

1	MS 0360	A.R.C. Westwood, 1000
1	MS 1186	Mark L. Keifer, 1242
1	MS 1153	Larry D. Bacon, 1248
1	MS 0321	Ed Barsis, 1400
1	MS 1111	Sudip Dosanjh, 1421
10	MS 1111	Scott Hutchinson, 1421
1	MS 1111	Martin Lewitt, 1421
1	MS 1111	John Shadid, 1421
1	MS 1109	Ted Barragy, 1424
1	MS 1109	Robert Benner, 1424

1	MS 1109	Art Hale, 1424
1	MS 1109	Rob Leland, 1424
1	MS 0865	Marvin E. Morris, 2753
1	MS 0865	Roy E. Jorgenson, 2753
1	MS 0425	Raymond Zazworksky, 4115
1	MS 0750	David L. Alumbaugh, 6116
1	MS 0965	Brian C. Brock, 9211
1	MS 1166	Joseph Kotulski, 9352
1	MS 1166	Douglas Riley, 9352
1	MS 9018	Central Technical Files, 8523-2
5	MS 0899	Technical Library, 13414
1	MS 0619	Print Media, 12615
2	MS 0100	Document Processing, 7613-2 For DOE/OSTI