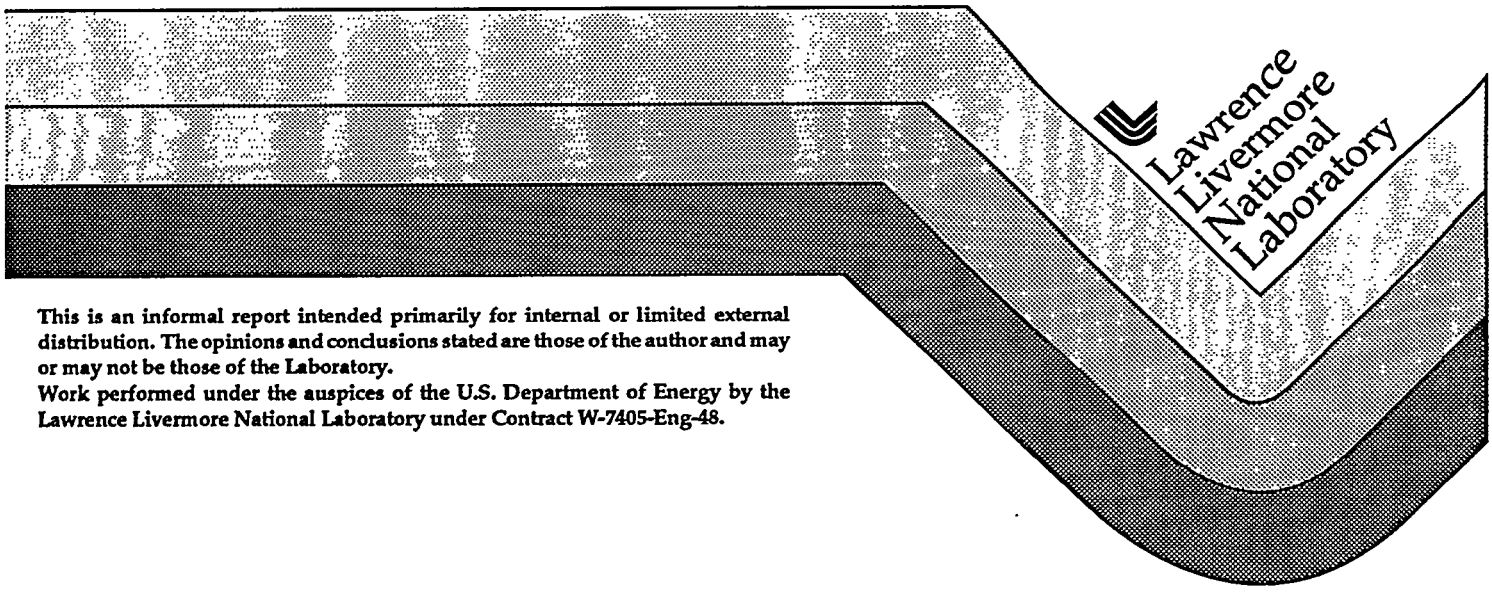
UCRL-ID-120581

DSI3D - RCS

Theory Manual

Niel Madsen
David Steich
Grant Cook
Bill Eme

March 16, 1995



Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

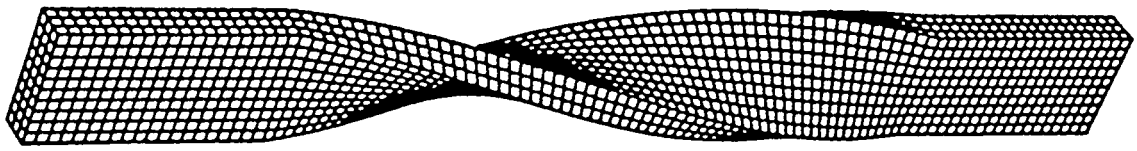
DISCLAIMER

DISCLAIMER

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

DSI3D - RCS

Theory Manual



March 1995

Niel Madsen, David Steich, Grant Cook, Bill Eme

Lawrence Livermore National Laboratory

Table of Contents

Introduction

The DSI3D-RCS code is designed to numerically evaluate radar cross sections on complex objects by solving Maxwell's curl equations in the time-domain and in three space dimensions. The code has been designed to run on the new parallel processing computers as well as on conventional serial computers.

The DSI3D-RCS code is unique for the following reasons:

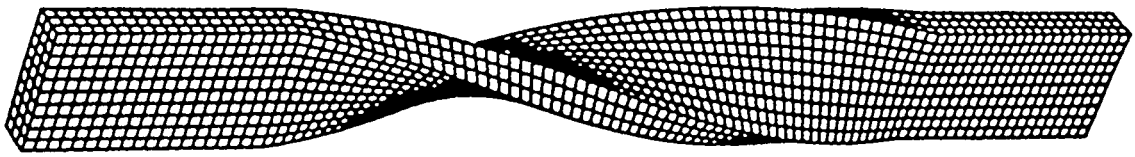
- Allows the use of unstructured non-orthogonal grids,
- Allows a variety of cell or element types,
- Reduces to be the Finite Difference Time Domain (FDTD) method when orthogonal grids are used,
- Preserves charge or divergence locally (and globally),
- Is conditionally stable,
- Is non-dissipative,
- Is accurate for non-orthogonal grids.

This method is derived using a Discrete Surface Integration (DSI) technique[1]. As formulated, the DSI technique can be used with essentially arbitrary unstructured grids composed of convex polyhedral cells. This implementation of the DSI algorithm allows the use of unstructured grids that are composed of combinations of non-orthogonal hexahedrons, tetrahedrons, triangular prisms and pyramids. This algorithm reduces to the conventional FDTD method when applied on a structured orthogonal hexahedral grid.

Overview

Discrete Surface Integration Method

We begin by assuming that we wish to solve Maxwell's curl equations on an irregular three-dimensional domain R that has a boundary surface denoted by S. We will also assume that the domain R has been discretized into convex polyhedrons. The figure below shows a twisted waveguide discretized using hexahedral cells.



Twisted waveguide discretized using distorted hexahedral cells.

This is an example of a problem type that we wish to consider that could not be easily solved using the conventional orthogonal grid FDTD method. Maxwell's curl equations are given by:

$$\frac{\partial \mathbf{D}}{\partial t} = \nabla \times \mathbf{H} \tag{1}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \tag{2}$$

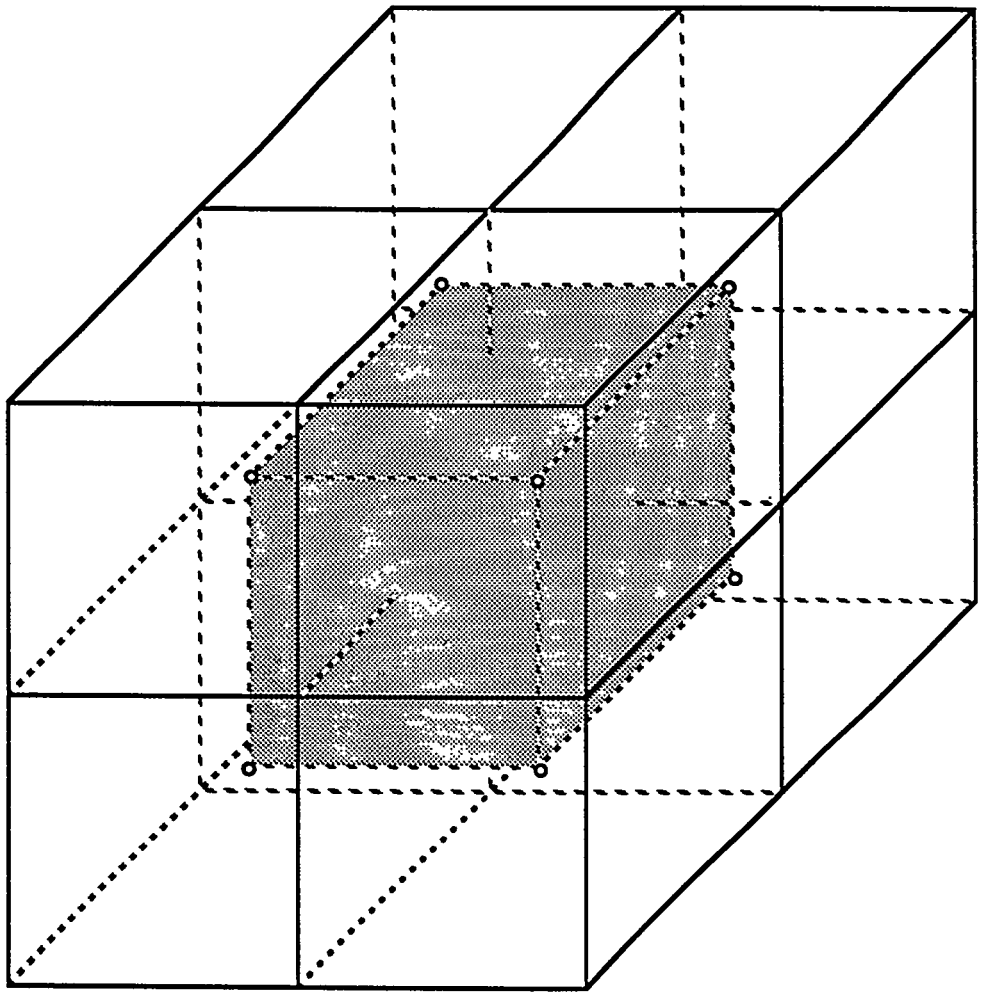
where for linear isotropic materials the vectors **D**, **E**, **B** and **H** are related by the constitutive relationships

$$\mathbf{D} = \varepsilon \mathbf{E}$$
$$\mathbf{B} = \mu \mathbf{H}$$

The linear isotropic material properties are: ε, the permittivity, and μ, the permeability. Like the conventional FDTD method, the DSI methods can be generalized to treat more complex materials. However, for the purposes of this discussion, we will assume that the linear isotropic material properties are piecewise constant over the domain R. We will also assume that S is a perfectly conducting surface, i.e., $E_{tan} = 0$. This is sufficient to guarantee that the problem is well-posed.

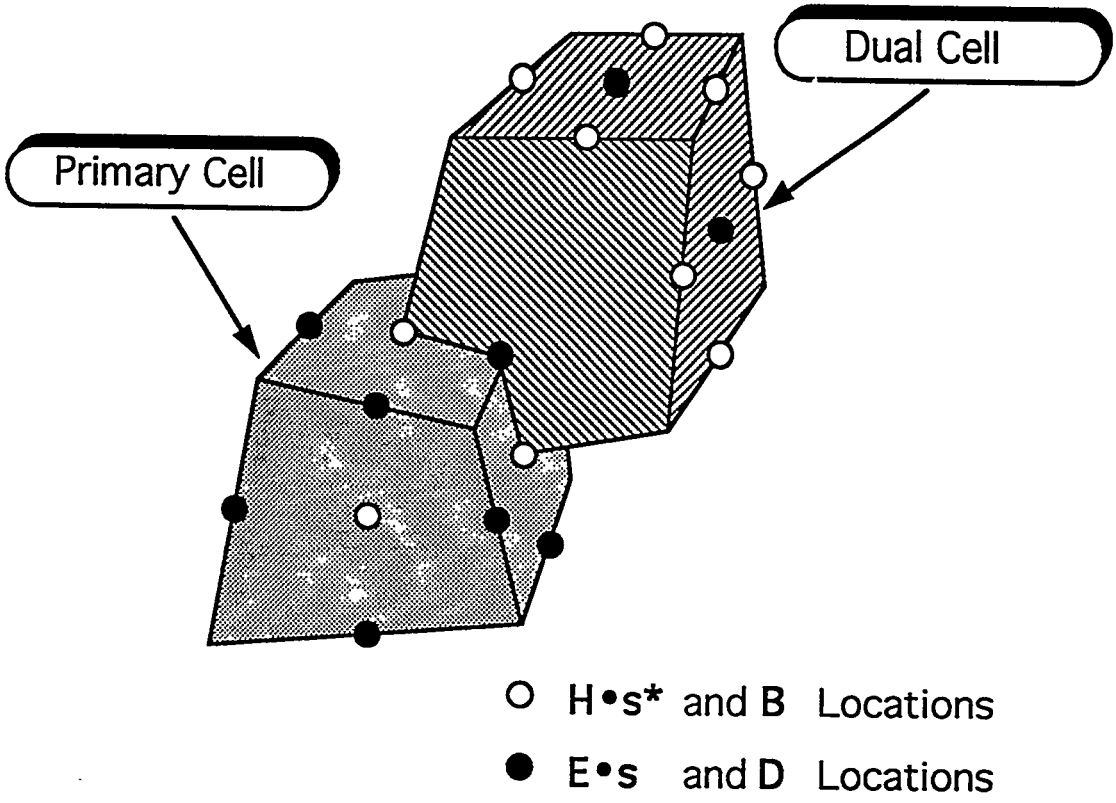
We will derive our new DSI algorithm for unstructured grids formed from convex polyhedral cells. We restrict the choice of cell types to convex polyhedrons whose edges are straight lines. The faces of the polyhedrons are not necessarily planar and we make the assumption that any face in the assembled grid is shared by at most two cells. These are very weak restrictions and allow a great deal of flexibility.

The DSI method requires the use of a dual grid. The dual grid and its structure are completely derivable from a knowledge of the primary grid. For each cell of the primary grid, we define its barycenter to be a node of the dual grid. The barycenter of a cell is located at the average of the coordinates of the nodes which define the cell. We construct edges of the dual grid by connecting barycenters of adjacent cells with straight lines passing through each of the interior cell faces of the primary grid. The barycenters of two cells will be connected (i.e., form a dual edge) if and only if the two cells share a common face. For primary cell faces which lie on the problem boundary, S, we form a corresponding half dual edge by joining the cell barycenter to a point on the face by moving from the cell barycenter in the direction of the face area normal vector. There is a one-to-one correspondence between the nodes, edges, faces, and cells of the primary grid to the cells, faces, edges and nodes of the dual grid, respectively. The dual face associated with a primary edge has as its perimeter the dual edges associated with all of the primary faces which share the given primary edge. The dual cell associated with a primary node has as its surface the dual faces which correspond to all of the primary edges which share the primary node. Though not necessary for the definition of the new algorithm, we recommend that the variations in grid sizes and angles be sufficiently smooth so that the primary and dual edges actually intersect their corresponding dual and primary faces, respectively. Degradation of solution accuracy has been observed when this condition is not met. The following figure shows an eight cell hexahedral primary grid and its one interior dual cell.



Primary grid consisting of eight hexahedral cells and its one interior dual cell.

Our DSI solution variables will be associated with the edges and faces of the primary grid and also with the edges and faces of the dual grid. The quantity associated with a primary cell edge is the projection of the electric field vector onto that edge, i.e., $\mathbf{E} \cdot \mathbf{s}$, where $\mathbf{s}$ is the primary cell edge vector. The magnetic field projection $\mathbf{H} \cdot \mathbf{s}^*$ is associated with a dual cell edge where $\mathbf{s}^*$ is the dual cell edge vector. In addition, with each primary grid face we will associate a full magnetic field vector **B**, and with each dual grid face we will associate a full electric displacement vector **D**. We will denote with an asterisk, *, geometric quantities associated with the dual grid. The following figure depicts these associations.



Discrete electric and magnetic field variable locations relative to the primary and dual grid cells.

We remark that these associations of field quantities with the primary and dual grid locations are entirely reciprocal and that the respective locations of the magnetic and electric field quantities could be interchanged. The particular choice of which field quantities to associate with each grid is best determined by deciding which field quantities one desires to have on the exterior boundary surfaces where the boundary conditions will be imposed. Since we are assuming that our domain R is surrounded by a perfect electric conductor, we will associate the electric and magnetic field quantities as described above. For open region problems, the choice for the location of the field quantities will depend on the particular radiation boundary condition algorithm used.

We will now describe the equations and algorithmic process used to advance in time the magnetic field vectors which are associated with each primary grid face. We assume that the time variable has been discretized by the choice of a time step size, Δt . Superscripts on field quantities will denote their time state with $\mathbf{D}^k = \mathbf{D}(t^k) = \mathbf{D}(k\Delta t)$. As we will be using a leapfrog style time integration method, the magnetic field vectors will be associated with half-integer times, $t^{k+\frac{1}{2}}$, and the electric field vectors will be associated with integer times, t^k. For a particular primary cell face, we define the area-normal vector to be $\mathbf{N} = \int \mathbf{n}\, dS$, where $\mathbf{n}$ is a unit surface normal defined by the right-hand rule in relation to a specified circulation around the perimeter of the cell face. It is easily shown that $\mathbf{N}$ is uniquely determined by the perimeter of the surface and is independent of the actual interior surface shape. This fact allows for simple computations of $\mathbf{N}$ using

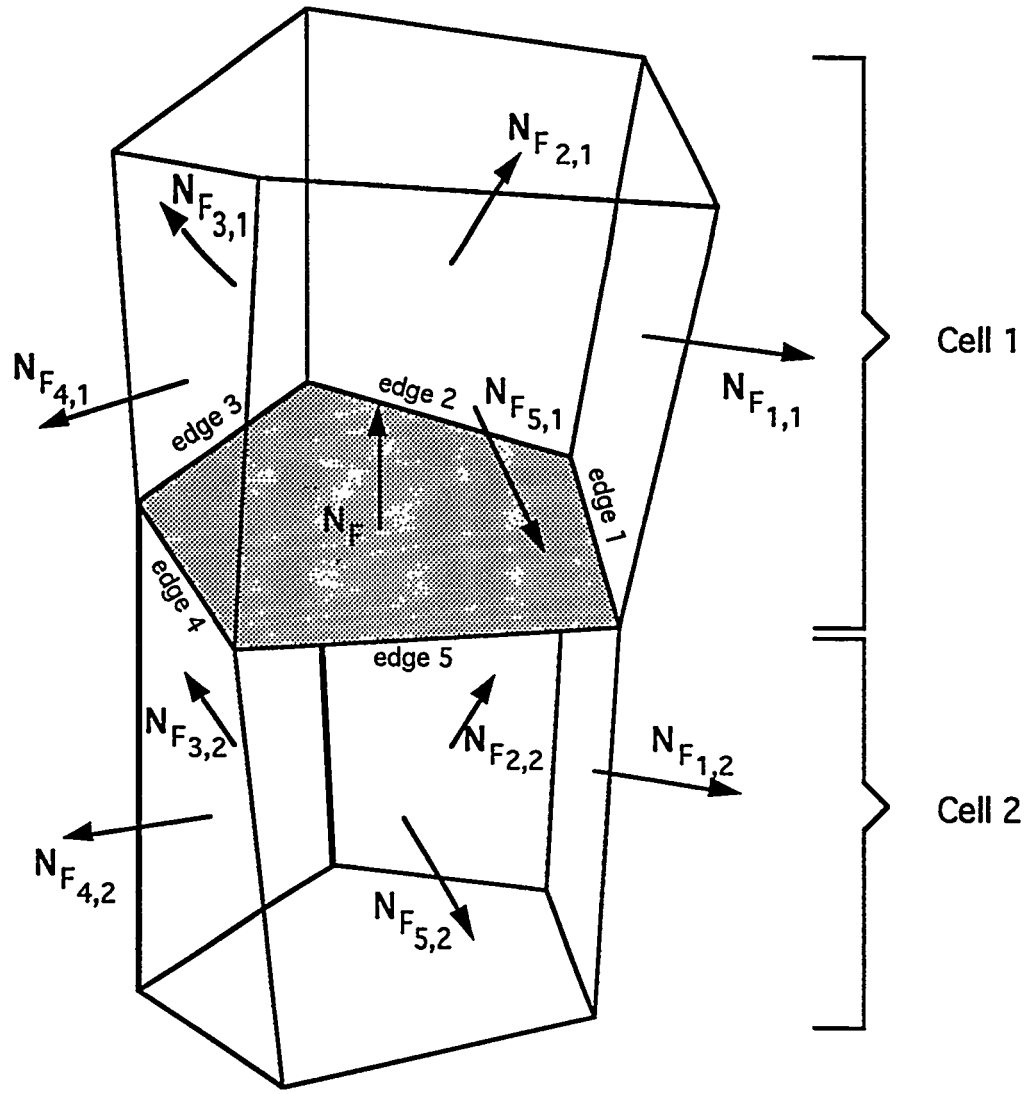
piecewise planar approximating surfaces. Using (2) for the primary grid face, F, we define the time derivative of the normal component of the magnetic field to be:

$$\frac{d\mathbf{B}^k}{dt}\cdot\mathbf{N}_F \equiv \int_F\left(\frac{\partial\mathbf{B}^k}{\partial t}\cdot\mathbf{n}\right)dS = -\int_F\left(\nabla\times\mathbf{E}^k\right)\cdot\mathbf{n}\,dS$$

$$= \oint_F \mathbf{E}^k\cdot d\mathbf{l} \qquad (3)$$

Equation (3) allows us to obtain the time derivative of the normal component of the magnetic field on a primary face, F, from the electric field projections onto the perimeter edges of that face. The last integral in (3) is easily computed numerically by summing these edge projections. This can be done for each primary cell face.

The next step in the algorithm is to use these time derivatives of the normal components of the magnetic field to compute a full vector value of $d\mathbf{B}_F^k/dt$ for the primary cell face, F. We will assume that the face F is defined by P primary edges and nodes, with the i^{th} node being located at the intersection of the consecutive edges i and $m=(i \bmod P)+1$. Also, we assume that the face F is shared by N_c primary grid cells, where by construction: $N_c=1$ for boundary faces and $N_c=2$ for interior faces. We will denote by $F_{i,j}$, the face of cell j (other than F) which shares edge i. The following figure depicts these associations for a dual edge associated with a primary face defined by five primary edges.



The primary grid faces used to time advance a magnetic field vector.

At each of the P nodes of face F, we will unfold N_c vector values $d\mathbf{B}_{ij}^{k}/dt$ by solving the 3×3 system of equations:

$$\begin{aligned}
\frac{d\mathbf{B}_{ij}^{k}}{dt}\cdot\mathbf{N}_{F} &= -\oint_{F}\mathbf{E}^{\mathbf{k}}\cdot d\mathbf{l} \\
\frac{d\mathbf{B}_{ij}^{k}}{dt}\cdot\mathbf{N}_{F_{i,j}} &= -\oint_{F_{i,j}}\mathbf{E}^{\mathbf{k}}\cdot d\mathbf{l} \\
\frac{d\mathbf{B}_{ij}^{k}}{dt}\cdot\mathbf{N}_{F_{m,j}} &= -\oint_{F_{m,j}}\mathbf{E}^{\mathbf{k}}\cdot d\mathbf{l}
\end{aligned} \tag{4}$$

where $i = 1,...,P$, $j = 1,...,N_c$, and $m = (i \bmod P) + 1$. For this primary face, F, which is shared by N_c primary cells, there are PN_c different values of $d\mathbf{B}_{ij}^k/dt$, which will now be averaged or interpolated to form a single $d\mathbf{B}_F^k/dt$ vector for the face. The particular averaging or interpolation we use is given by:

$$\frac{d\mathbf{B}_F^k}{dt} = \frac{\sum_{j=1}^{N_c}\sum_{i=1}^{P}|w_{ij}|\frac{d\mathbf{B}_{ij}^k}{dt}}{\sum_{j=1}^{N_c}\sum_{i=1}^{P}|w_{ij}|} \tag{5}$$

where the weight

$$w_{ij} = N_F \cdot \left(N_{F_{i,j}} \times N_{F_{i+1,j}}\right),$$

represents the volume of the j^{th} local coordinate system at node i of face F. We note also that the weight w_{ij} is the determinant of the system of equations (4). We also comment that equation (5) has been derived assuming the vector orientations shown in the preceding figure. For other orientations, appropriate modifications of the signs of the various terms must be properly accounted for.

The full **B** vector for each primary face, F, may now be advanced in time using the time-centered leapfrog algorithm:

$$\mathbf{B}^{k+\frac{1}{2}} = \mathbf{B}^{k-\frac{1}{2}} + \Delta t \frac{d\mathbf{B}^k}{\mathbf{dt}}, \tag{6}$$

where Δt is the specified time step size. Finally, a time advanced value of the projection of the magnetic field onto the dual edge, $\mathbf{s}^*$, which penetrates the primary cell face, F, is easily obtained using:

$$\left(\mathbf{H} \cdot \mathbf{s}^*\right)^{k+\frac{1}{2}} = \frac{\mathbf{B}^{k+\frac{1}{2}} \cdot \mathbf{s}^*}{\mu}, \tag{7}$$

where μ is an appropriate permeability value. If the permeability is discontinuous then a value can be determined by an appropriate average as is done for FDTD algorithms.

It is important to note that the first equation in (4) (namely, the equation coming from the given face F) is common to all of the sets of 3×3 equations being used in the time advance process for the face. This implies that the averaged value (5) of the time derivative of the magnetic field vector for F also satisfies this equation. It is from this aspect of the numerical algorithm that divergence or charge conservation can be demonstrated. If we integrate the divergence of (6) over a primary cell, C, we obtain

$$\int_C \left(\nabla \cdot B^{k+\frac{1}{2}}\right) dV = \int_C \left(\nabla \cdot B^{k-\frac{1}{2}}\right) dV + \Delta t \int_C \left(\nabla \cdot \frac{d\mathbf{B}^k}{\mathbf{dt}}\right) dV. \tag{8}$$

Considering the last integral term in (8) we have

$$\int_C \left(\nabla \cdot \frac{d\mathbf{B}^k}{\mathrm{dt}} \right) dV = \int_{\partial C} \left(\frac{d\mathbf{B}^k}{\mathrm{dt}} \cdot \mathbf{n} \right) dS = -\sum_i \oint_{F_i} \mathbf{E}^k \cdot d\mathbf{l} = 0, \tag{9}$$

where the sum runs over all the faces Fi of the closed cell C. The last sum is zero because each edge of C will be traversed twice, once in each direction. Thus we see that the local divergence of the time derivative of the magnetic field vector is zero and so if the initial fields have zero local divergence, then (8) and (9) prove that zero divergence of the time advanced fields will be conserved.

To time advance the electric field vectors which are associated with each dual cell face, we proceed in a manner which is exactly "dual" to the magnetic field procedure described above. For any dual cell face, we define the dual area-normal vector to be $\mathbf{N}^* = \int \mathbf{n}^* dS^*$, where $\mathbf{n}^*$ is a unit dual surface normal defined by the right-hand rule in relation to a specified circulation around the perimeter of the dual cell face. Using (1) for a given dual grid face F^*, we define the time derivative of the normal component of the electric displacement vector to be:

$$\frac{d\mathbf{D}^{k+\frac{1}{2}}}{\mathrm{dt}} \cdot \mathbf{N}^*_{F^*} \equiv \int_{F^*} \left(\frac{\partial \mathbf{D}^{k+\frac{1}{2}}}{\partial t} \cdot \mathbf{n}^* \right) dS^* = \int_{F^*} \left(\nabla \times \mathbf{H}^{k+\frac{1}{2}} \right) \cdot \mathbf{n}^* dS^* = \oint_{F^*} \mathbf{H}^{k+\frac{1}{2}} \cdot d\mathbf{l}^* \tag{10}$$

Again, the last integral in (10) is easily computed by summing the projections of the magnetic field on the edges defining the dual cell face. This can be done for each dual cell face.

The next step in the algorithm is to use these time derivatives of the normal components of the electric field to compute a full vector value of $d\mathbf{D}^k_{F^*}/dt$ for the dual cell face, F^*. We will assume that the face F^* is defined by P^* dual edges and nodes, with the i^{th} dual node being located at the intersection of dual edges i and $m = (i \bmod P^*) + 1$. Also, we assume that the dual face F^* is shared by N^*_c dual cells. We will denote by $F^*_{i,j}$, the dual face of dual cell j (other than F^*) which shares dual edge i. The above figure with all quantities replaced by their appropriate duals would depict these associations. At each of the P^* dual nodes of dual face F^*, we will unfold N^*_c vector values $d\mathbf{D}^{k+\frac{1}{2}}_{ij}/dt$ by solving the 3×3 system of equations:

$$\begin{aligned}
\frac{d\mathbf{D}^{k+\frac{1}{2}}_{ij}}{dt} \cdot \mathbf{N}^*_{F^*} &= \oint_{F^*} \mathbf{H}^{k+\frac{1}{2}} \cdot d\mathbf{l}^* \\
\frac{d\mathbf{D}^{k+\frac{1}{2}}_{ij}}{dt} \cdot \mathbf{N}^*_{F^*_{i,j}} &= \oint_{F^*_{i,j}} \mathbf{H}^{k+\frac{1}{2}} \cdot d\mathbf{l}^* \\
\frac{d\mathbf{D}^{k+\frac{1}{2}}_{ij}}{dt} \cdot \mathbf{N}^*_{F^*_{m,j}} &= \oint_{F^*_{m,j}} \mathbf{H}^{k+\frac{1}{2}} \cdot d\mathbf{l}^*,
\end{aligned} \tag{11}$$

where $i = 1,...,P^*$, $j = 1,...,N_c^*$ and $m = \left(i \bmod P^*\right) + 1$. For this dual face, F^*, which is shared by N_c^* dual cells, there are $P^* N_c^*$ different values of $d\mathbf{D}_{ij}^{k+\frac{1}{2}}/dt$, which will now be averaged or interpolated to form a single $d\mathbf{D}_{F^*}^{k+\frac{1}{2}}/dt$ vector for the dual face. The averaging methods (5) used for the magnetic fields are also used for the electric fields.

The full **D** vector for the dual face, F^*, may now be advanced in time using the time-centered leapfrog algorithm:

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \Delta t \frac{d\mathbf{D}^{k+\frac{1}{2}}}{\mathbf{dt}}, \tag{12}$$

where Δt is the specified time step size. Finally, a time advanced value of the projection of the electric field onto the primary edge, **s**, which penetrates the dual cell face, F^*, is easily obtained using:

$$(\mathbf{E} \cdot \mathbf{s})^{k+1} = \frac{\mathbf{D}^{k+1} \cdot \mathbf{s}}{\varepsilon}, \tag{13}$$

where ε is an appropriate permittivity value. The local conservation of divergence for the electric field can be easily shown in a manner similar to that described above for the magnetic fields.

Equations (3)-(7) for the magnetic field quantities, equations (10)-(13) for the electric field quantities and a linear averaging or interpolation method (5), constitute the new divergence conserving DSI approximation methods.

We note that significant simplifications occur when a primary edge is orthogonal to its dual face, or "dually", when a dual edge is orthogonal to its primary face. When this occurs, the vectors **s** and $\mathbf{N}_{F^*}^*$ (or $\mathbf{s}^*$ and $\mathbf{N}_F$) are aligned. The time advance of the edge projected field value may be performed directly using (3) (or (10)) and the averaging (5) may be completely bypassed. Stated another way, when the above orthogonality conditions exist, the time advance of $\mathbf{E} \cdot \mathbf{s}$ (or $\mathbf{H} \cdot \mathbf{s}^*$) may be accomplished using a single line integral around the perimeter of the face F^* (or F). It is this fact that demonstrates that the DSI methods are completely equivalent to the canonical FDTD methods when orthogonal hexahedral based grids are used. Therefore, the DSI methods are direct generalizations of the orthogonal grid FDTD methods for unstructured nonorthogonal grids. These orthogonality conditions may occur locally for only a relatively few edges or may occur globally for the entire grid. They occur globally when structured grids composed of orthogonal hexahedral cells are used. They also occur globally when grids are used which are three dimensional analogs of two-dimensional grids formed from Delaunay triangles and their dual Voronoi polygons.

Boundary Conditions

Perfect Electric Conductor (PEC) boundary conditions are some of the most commonly used boundary conditions in computing RCS values for metallic objects. PEC conditions require that the total tangential electric field be set to zero on the PEC surface. This is easily accomplished because the tangential electric field components, $\mathbf{E} \cdot \mathbf{s}$, live on the boundary and we simply set $\mathbf{E} \cdot \mathbf{s} = E_{tan} = 0$ for the total field. If the grid is composed of orthogonal hexahedral cells, this is all that is required. For nonorthogonal grids, another subtlety arises in updating primary cell edge values for edges which have an endpoint on the problem boundary surface and the other endpoint in the problem interior. As has been depicted earlier, the update of these edges requires the use of the two dual cells which surround the endpoints of the edge. The dual cell which surrounds the interior endpoint of the edge is a typical dual cell and is treated in the usual manner. However, the dual cell associated with the edge endpoint which lies on the problem boundary is special in that most of its dual faces are half the normal size as they are clipped by the boundary. For these "clipped" dual faces (which are associated with boundary primary edges) special care must be taken to guarantee that the values of the path integrals around these faces are compatible with the PEC boundary condition. Two items need to be considered for compatibility. First, the dual face area normal vector for the "clipped" dual face must be tangent to the boundary surface. This is readily accomplished by carefully considering what the full "unclipped" dual face would be if the problem space were extended beyond the actual boundary. Second, the path integral of the total magnetic field around the dual face must be zero as it is equal to the time derivative of the total tangential electric field which is zero from the definition of the PEC. In the scattered field formulation, this means that the path integral of the scattered magnetic field must be set proportional to the time derivative of the incident tangential electric field at the boundary in the direction of the dual face area normal vector.

Perfect Magnetic Conductor (PMC) symmetry plane boundary conditions are another useful boundary condition for certain types of problems. They frequently allow a problem to be solved using a grid that is half as big as the full grid for the problem. These conditions require that the total tangential magnetic field be set to zero on the specified surface. We desire that these conditions be applied on the surface of the primary grid which is usually where tangential electric field information is specified. We accomplish this by using reflection principles and images which are inherent with this type of planar boundary. Specifically, the tangential magnetic fields and normal electric fields have odd symmetry across the plane, and the normal magnetic fields and tangential electric fields are of even symmetry across the plane. Using these symmetries, appropriate values of the full path integrals can be computed using only the half path integrals which are in the computational domain. With these appropriate path integrals, values for the tangential electric field value which lie in the symmetry plane can be updated on a time step by time step basis.

Radiation Boundary Conditions (RBC) are an absolute necessity for computing RCS values in open region problems. These conditions allow one to truncate the grid at a reasonable distance from the scatterer and thus avoid having to discretize extremely large volumes of space. These conditions are designed to allow scattered EM wave to leave the computational region without significant reflections. Achieving good RBC conditions is usually a very nontrivial process and much research has been performed looking for better and better RBC algorithms. No definitively "best" RBC algorithm has been identified - even for orthogonal grid FDTD algorithms. The situation for unstructured nonorthogonal grid codes is much more difficult and the state of RBC algorithms for these codes is still quite primitive. We have tested the standard Mur[6] conditions and Liao[5] conditions for the situation where we terminate the exterior of the computational grid with several layers of structured and orthogonal hexahedral cells. These conditions work reasonably well but can impose considerable mesh generation difficulties in that it can be quite nontrivial to effect the transition from a boundary conforming unstructured and nonorthogonal grid to the structured and orthogonal portion of the grid where the RBC is applied. We have successfully implemented unstructured grid versions of several first order RBCs such as Mur, Liao, and Higdon[4-6]. Each of these has been parameterized so that it will minimize the backward reflection at a user-specified angle of incidence. We have had difficulty adapting the second and higher order conditions to the unstructured grid regime. Attempts thus far have resulted in instabilities.

The Farfield Transform for RCS

Far field potentials are well known in the frequency domain; we use the time domain forms obtained by Kunz and Luebbers [2]

$$\underset{\sim}{W}(\hat{r}R,t)=\frac{1}{4\pi Rc}\frac{\partial}{\partial t}\left\{\int_S \hat{n}\times\underset{\sim}{H}\left(t+\frac{\underset{\sim}{r'}\bullet\hat{r}-R}{c}\right)da\right\}$$

$$\underset{\sim}{U}(\hat{r}R,t)=\frac{1}{4\pi Rc}\frac{\partial}{\partial t}\left\{\int_S \underset{\sim}{E}\left(t+\frac{\underset{\sim}{r'}\bullet\hat{r}-R}{c}\right)\times\hat{n}da\right\} \quad ,$$

where $\hat{n}$ is the normal vector on S, the integration surface completely enclosing the scatterer, $\underset{\sim}{r'}$ is the point on the integration surface where the integrand is evaluated, and $\hat{r}R$ is the position of the far field evalution point.

From the potentials, $\underset{\sim}{U}$ and $\underset{\sim}{W}$, the fields can be found accurate to $\frac{1}{R}$ as

$$E_\theta(\hat{r}R,t)=-\eta W_\theta(\hat{r}R,t)-U_\phi(\hat{r}R,t)$$

These far fields can be computed by accumulating time histories of $\underset{\sim}{U}$ and $\underset{\sim}{W}$ at each far field point, or by saving the time histories of $\underset{\sim}{E}$ and $\underset{\sim}{H}$ at all edges on the integration surface. In order to save in data transfer requirements in the parallel implementation, and in intermediate file storage, we chose the former solution.

Because a long time history of these vector potentials is required for wide-band RCS results, a buffering mechanism has been developed to keep storage requirements to a minimum.

A post-processor reads the vector potentials from binary files and computes the RCS.

The integration surface is chosen to be a set of faces on the primary grid that form a closed surface that completely encloses the scatter. For each face, $\underset{\sim}{A} \equiv \hat{n}da$ is approximated by the area normals that are available in DSI3D. Also needed in the calculation are $\underset{\sim}{H}$ and $\underset{\sim}{E}$. $\underset{\sim}{H}$ is available at the centroid of each face because there is a dual edge associated with each face in the primary grid. $\underset{\sim}{E}$ is found at the midpoints of the edges of each face; these values need to be interpolated to the centroid of the face.

Because these field values on the near field integration surface are different from those obtained by the standard DSI solver (vectors instead of scalars), another set of solver coefficients must be generated on each edge that contributes vector field values to the integration over faces on the far field integration surface.

Because the retarded time of the fields on the integration surface is not an integral multiple of the problem time step (Δt), the time derivative of the integrand on a face of the integration surface must be linearly interpolated. For the $\underset{\sim}{U}$ equation at the m^{th} time step and the j^{th} face, this leads to:

$$\underset{\sim l}{U} = \underset{\sim l}{U} + \frac{\alpha_j}{4\pi Rc(\Delta t)} \underset{\sim j}{E^m} \times \underset{\sim j}{A}$$

$$\underset{\sim l+1}{U} = \underset{\sim l+1}{U} + \frac{1-2\alpha_j}{4\pi Rc(\Delta t)} \underset{\sim j}{E^m} \times \underset{\sim j}{A}$$

$$\underset{\sim l+2}{U} = \underset{\sim l+2}{U} + \frac{\alpha_j - 1}{4\pi Rc(\Delta t)} \underset{\sim j}{E^m} \times \underset{\sim j}{A}$$

$$\text{where: } l = l(j,m) = \left\lfloor m + \gamma_j + \frac{1}{2} \right\rfloor - \left\lfloor m_0 + \min_{k \in S} \gamma_k + \frac{1}{2} \right\rfloor + 1$$

$$\alpha_j = \left\lfloor \gamma_j + \frac{1}{2} \right\rfloor - \left\lfloor \gamma_j - \frac{1}{2} \right\rfloor$$

$$\text{and } \gamma_j = \frac{R - \hat{r} \bullet \underset{\sim}{r}_j}{c(\Delta t)}$$

Due to the leapfrog time integration strategy for Maxwell's equations, $\underset{\sim}{H}$ lags $\underset{\sim}{E}$ by half of a time step. For the $\underset{\sim}{W}$ equation at the m^{th} time step and the j^{th} face, this leads to:

$$\underset{\sim}{W}_k = \underset{\sim}{W}_k + \frac{\beta_j}{4\pi R c(\Delta t)} \underset{\sim}{A}_j \times \underset{\sim}{H}_j^{m-1/2}$$

$$\underset{\sim}{W}_{k+1} = \underset{\sim}{W}_{k+1} + \frac{1 - 2\beta_j}{4\pi R c(\Delta t)} \underset{\sim}{A}_j \times \underset{\sim}{H}_j^{m-1/2}$$

$$\underset{\sim}{W}_{k+2} = \underset{\sim}{W}_{k+2} + \frac{\beta_j - 1}{4\pi R c(\Delta t)} \underset{\sim}{A}_j \times \underset{\sim}{H}_j^{m-1/2}$$

$$\text{where: } k = k(j,m) = \left\lfloor m + \gamma_j \right\rfloor - \left\lfloor m_0 + \min_{i \in S} \gamma_i \right\rfloor + 1$$

$$\text{and } \beta_j = \left\lfloor \gamma_j \right\rfloor - \gamma_j + 1$$

The final step before computing the radar cross section (RCS) is to Fourier transform the incident pulse and the far field. Then, we can compute the RCS as

$$\sigma_{\alpha\beta} = \lim_{R \to \infty} 4\pi R \frac{\left\| \underset{\sim}{\beta} \bullet \underset{\sim}{E}(\hat{r}R, \omega) \right\|^2}{\left\| \underset{\sim}{\alpha} \bullet \underset{\sim}{E}^{inc}(\omega) \right\|^2}$$

where $\underset{\sim}{\alpha}$ is the polarization of the transmitting radar antenna, $\underset{\sim}{\beta}$ is the polarization of the receiving radar unit, and $\underset{\sim}{E}^{inc}$ is the E field of the incident Gaussian pulse.

Note that

$$\underline{\alpha} = \cos(\alpha_{inc})\hat{\theta} + \sin(\alpha_{inc})\hat{\phi}$$

$$\underline{\beta} = \cos(\alpha_{far})\hat{\theta} + \sin(\alpha_{far})\hat{\phi}$$

By convention, q=0 is along the + z axis, and u=0 is on the + x axis.

Parallel Processing and Problem Partitioning

To efficiently use multiple processors on a parallel computer to solve any one problem, it is required to split or partition the problem among the available processors. With the problem partitioned, each processor can produce the solution for its part of the problem. In order for this process to be efficient, there are to fundamental goals any partitioning process.

- Each processor should have an equal workload.
- The amount of inter-processor communication required should be minimized.

For DSI3D, this partitioning ultimately divides up the primary and dual edge variables $\mathbf{E}\cdot\mathbf{s}$ and $\mathbf{H}\cdot\mathbf{s}^*$. For some structured grids it is intuitively obvious how to partition the problem so that the above stated goals are achieved. For unstructured grids and particularly for tetrahedral based grids adequate partitioning schemes are very non-intuitive and difficult to construct. Fortunately, recent work by H. Simon[3] has produced some automatic approaches for solving this problem.

The new approach is termed a Recursive Spectral Bisection method (RSB). The details of the RSB approach as well as comparisons with other approaches may be found in the above references. For DSI3D we have chosen to use the grid cells as the basic quantities to be partitioned. This is primarily done because there are significantly fewer cells than edges and faces in the primary and dual grids. This results in a smaller and more efficient RSB process.

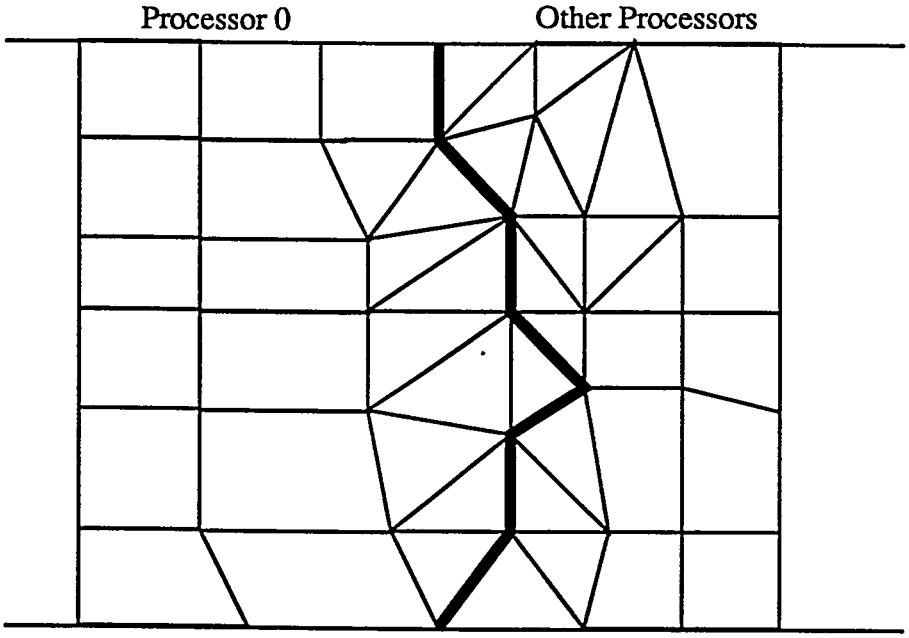
First, a graph of the connectivity of the cells is constructed. We choose to define that two cells are connected as meaning that they share a face (other possibilities are that they share a node or an edge). From the graph the so-called "Laplacian Matrix" is formed. This matrix has a row and column for each cells in the grid. For a given row, negative ones are placed in the columns of the other cells which are connected to it. The diagonal entry is found be summing the non-diagonal entries and placing the absolute value of this sum as the diagonal entry.

This matrix is always singular (has a zero eigenvalue). This is easily seen as a vector consisting of all "ones" multiplied by the Laplacian matrix gives zero. All of the other non-zero eigenvalues are positive. Using a Lanczos algorithm the next smallest eigenvalue is computed and an eigenvector corresponding to this eigenvalue is found. The median entry of the eigenvector is found and the problem is split in half by assigning cells which correspond to eigenvector entries greater than the median value to one processor and those which correspond to entries less than the median value to the other processor. One significant drawback to the RSB method is that it produces partitions for numbers of processors which are a power of 2 only.

This simple sounding approach actually works quite well in practice. There is mathematical theory that states that if the original grid is connected that each of the produced sub-partitions will also be connected. Stated another way, each processor will be working on a single sub-piece of the grid rather than a few cells here and a few more from someplace distant in the grid. There is also theory that shows that the partitioning is close to optimal as far as the communication requirements go. It also is guaranteed to produce the same number of cells (± 1) for each processor to process.

Communication Strategy

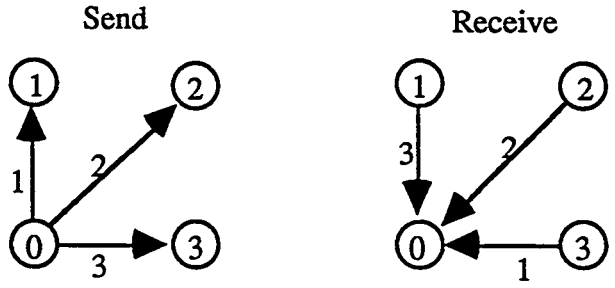
Once each processor on the parallel computer has its sub-piece of the global grid defined, the processor will be responsible for computing and/or updating all of the edge-based variables which are associated with its piece of the grid. We note that there will be some duplication due to the fact that faces and edges which exist on a partition boundary will necessarily be shared with the other processor(s) which "owns" the cell(s) on the other side of the partition boundary. In order for a processor to be able to update all of its variables it will need have access to all of the other cells which share nodes on the partition boundary. The next figure shows a 2D grid with a partition boundary depicted as the heavy solid line.



Grid with processor partition depicted with heavy solid line

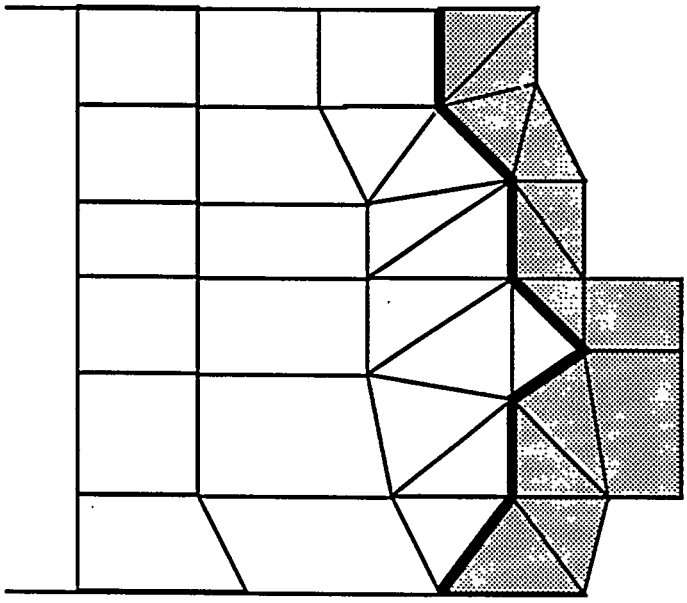
As each processor does not know in the beginning what cells or processors it may be surrounded by, each processor must discover these relationships by communicating with the other processors. The first step in this process is that each processor identifies all of its own boundary nodes and elements. These nodes may be nodes which lie on the real physical problem boundary or the partition boundary or both. This boundary-element data is then successively sent to every other processor. At the same time as a processor sends its boundary data it also receives the boundary data from another processor. A processor then compares its own boundary data with that obtained from another processor looking for matches between the boundary node sets. When a match occurs, then data is

stored indicating which elements need to be sent to the other processor and also which elements need to be received from the other processor. This process is repeated n-1 times (where n is the number of processors being used) as if the processors were connected in a "ring-like" manner. After completion of the boundary identification stage, each processor then actually sends all of its elements to the individual processors who need to have them. Again at the same time, a processor is receiving from the other processors all of the elements it requires.



Ringlike send / receive communication order for processor 0 to discover neighboring cells belonging

At the conclusion of this boundary element exchange stage, each processor will now have all of its original elements plus a layer of new elements which share its grid partition boundaries. At this point each processor has all of the geometry data which will be required to perform time step updates of all of its original edge and dual edge values. As a final step in setting up the communication tables which will be required in the time stepping update process, each processor must create a final communication table which lists all of the actual edge and dual edge field values it must send/receive to/from other processors. This is readily generated from the previously generated boundary element exchange data.



Extra cells (shaded) obtained by processor 0 from neighboring processors.

There is a somewhat subtle issue relating to obtaining the same results each time the code is run for the same problem which must be addressed. Because more than one processor will update edge values which lie on the grid partition boundaries, slightly different values for the same edge may be generated by the different processors sharing the edge. This possibility arises because even though they have identical geometry information (node coordinates and elements definitions), they may use a different ordering of the arithmetic operations to compute a new edge value. Round-off errors will then cause slightly different values to be produced. When these different values are sent to other processors through the message passing communication system, it is usually not feasible to control the exact order in which messages arrive at another processor. As a result of this asynchronous behavior, the same code can produce different answers each time it is run even for the same problem.

This potential randomness can be overcome by developing a unique "ownership" strategy for all of the problem variables. We do this in a simple but effective way by decreeing that an edge value is "owned" exclusively by the lowest numbered processor which shares the edge. This processor's value will then be the only value that will be used by any other processor. This strategy insures that with a constant partition of the grid and a constant number of processor being utilized for the solution process, identical results will be obtained each time the problem is run. Changing the grid partitioning or the number of processors being used may still give slightly different results.

Vectorization Considerations

When using nonorthogonal unstructured grids, a heavy penalty in computational efficiency can occur if sufficient care is not taken in structuring the computations and data. This will occur primarily because the inner-most loops involve a great deal of indirect addressing and the data may be accessed in rather random patterns. The basic DSI3D time stepping loops consist of dot products of vector coefficients with a vector of solution variables. These dot products can and will vary in length and involve indirect addressing. These conditions usually defeat most vectorizing compilers.

We have found that most of this difficulty can be easily overcome by reordering and regrouping the data so that much of it lies consecutively in memory. We will rearrange the data so that all of the data for dot products of the same length lie consecutively in memory. The next diagram shows the original inner loop for updating the electric field edge projected values. Next to it is shown the new inner loop restructured to gain more efficiency.

The short, variable length inner loops that perform the dot products limit performance on vector computers

```
do i = 1, hLen
   hdot = 0.
   do j = ipth(i), ipth(i+1) - 1
      hdot = hdot + coefh(j)
           *eds(npth(j))
   enddo
   hds(i) = hds(i) - dt * hdot
enddo
```

Sorting the variable update order by loop length allows do-loop orders to be exchanged, leaving very long inner loops

```
do L = 1, maxLLen
   i1 = firstHofLen(L)
   i2 = firstHofLen(L+1) - 1
   j1 = firstHCofLen(L) - i1
   j2 = firstHCofLen(L+1) - i1 - 1
   do j = j1, j2, HLengthCount(L)
      do i = i1, i2
         hds(i) = hds(i) + coefh(i+j)
                * eds(npth(i+j))
      enddo
   enddo
enddo
```

The performance improvement using this reordering and rearranging technique is dramatic. On a Cray C-90 single cpu, the performance on the original loop was about 15-20 megaflops. The rearranged data loop now performs at about 307 megaflops. On the Meiko CS-2 parallel processing computer the performance improves from about 5 megaflops to about 35 megaflops on each processor. We have even observed that the data restructuring also improves (to a lesser extent) the performance on scalar CPUs for various other machines.

Post-Processing

Post-processing for DSI3D is somewhat different than for most other codes. This is in part due to the fact that full electric and magnetic field vectors are almost never computed in the code as a part of the basic solution algorithm. Rather, projected field values onto primary grid and dual grid edges are the basic quantities computed and used in DSI3D. Also, since DSI3D has been designed as a parallel computer code, the data does not reside all in one place in that it is distributed across many processors. Another fact is that there is very little in the way of parallel graphics software available on almost any of the parallel computers.

The philosophy that we have adopted is that we will do all graphics post-processing not on the parallel computer but rather on workstations. This has several important implications. First, since the performance and capacity on most workstations is rather limited as compared to the parallel computer, it may not be possible to visualize all of the data together in one place at one time. Second, we will be required to develop software which will run on the parallel computer which will form and extract the data we desire to visualize.

We have developed a parallel post-processing tool called DSI3DIO which performs the required tasks. When DSI3D is run on a parallel computer, it will produce restart dump files which contain the edge and dual edge data. These files will exist on the separate processors and/or their local disk space. When DSI3DIO runs, it will read the original grid files and extract sup-pieces of the original grid as specified by the user. These sub-pieces may be planar cuts through the grid or particular material types. For the grid sub-pieces, DSI3DIO then uses the data in the restart files to build or interpolate full vector field data which will exist at the nodes of the grid sub-pieces. The code will then produce a reduced ascii grid file (which contains only the boundary cells of the region specified by the user) and ascii files containing the vector field data for this reduced grid. Currently, one can form electric field vectors, magnetic field vectors and Poynting vector field data. This data which will be much smaller than the entire problem data is then moved to a workstation and can be visualized with many different graphics software packages.

References

1. N.K. Madsen, "Divergence Preserving Discrete Surface Integral Methods for Maxwell's Curl Equations Using Non-orthogonal Unstructured Grids," *J. Comp. Phys.*, Vol. 119, pp. 34-45, 1995.

2. K.S. Kunz and R.J. Luebbers, *The Finite Difference Time Domain Method for Electromagnetics*, CRC Press, 1993.

3. H. Simon, "Partitioning of Unstructured Problems for Parallel Processing", *Computing Systems in Engineering*, Vol. 2, No. 2/3, pp. 135-148, 1991.

4. R.L. Higdon, "Numerical Absorbing Boundary Conditions for the Wave Equations", *Math. of Comput.*, Vol. 49, pp. 65-91, July 1987.

5. Z. Liao, H.L. Wong, B. Yang and Y. Yuan, "A Transmitting Boundary for Transient Wave Analysis", *Scientia Sinica (Series A)*, Vol. 27, pp. 1063-1076, 1984.

6. G. Mur, "Absorbing Boundary Conditions for the Finite-Difference Approximation of the Time-Domain Electromagnetic Field Equations", *IEEE Trans. Electromagn. Compat.*, Vol. 23, pp. 377-382, 1981.