Scalable Electromagnetic Simulation Environment

Raju R. Namburu[1], **Eric R. Mark, and Jerry A. Clarke**

Abstract: Computational electromagnetic (CEM) simulations of full-range military vehicles play a critical role in enhancing the survivability and target recognition of combat systems. Modeling of full-range military systems subjected to high frequencies may involve generating large-scale meshes, solving equations, visualization, and analysis of results in the range of billions of unknowns or grid points. Hence, the overall objective of this research is to develop and demonstrate a scalable CEM software environment to address accurate prediction of radar cross sections (RCS) for full-range armored vehicles with realistic material treatments and complex geometric configurations. A software environment consisting of scalable preprocessing, postprocessing, and an accurate CEM algorithmic approach is needed to achieve a significant reduction in overall simulation time for practical military applications. In addition to RCS, this high-fidelity scalable software environment can be easily extended to address wideband communications applications. This paper presents a scalable computational environment or framework for large-scale computational electromagnetics and acoustics (CEA) applications consisting of (a) scalable grid generation based on implicit surfaces and voxel methods, (b) scalable finite difference time domain method, (c) eXtensible Data Model and Format, and (d) parallel visualization utilizing a network distributed global memory approach. Two different applications are presented to illustrate the capabilities of the proposed approach. The first application demonstrates the scalability and validity of the proposed CEM software environment. The second application demonstrates the capability of the proposed approach to model and analyze a very large-scale application, namely, a full-scale combat vehicle simulation consisting of 2.56 billion cells.

keyword: finite difference time domain, scalable computing environment, computational electromagnetics, voxel grid, implicit surfaces, parallel preprocessing, parallel visualization, radar cross section, large-scale military application.

[1] U.S. Army Research Laboratory, Aberdeen Proving Ground, MD 21005.

1 Introduction

A low observable and controllable signature is the best survivability technique for a combat system. Next generation combat systems are envisioned to be of light weight, with tailored materials and a very short development cycle time. To take advantage of these lightweight new material systems, a better understanding of the signatures for the combat systems in a realistic battlefield environment is key in achieving dominance on the battlefield. For this to occur and to meet short combat system development cycles, validated high-performance computational electromagnetic (CEM) methods are needed. Scalable CEM simulations enable dramatic reductions in the cost and time associated with accurate predictions of radar cross section (RCS) for full-size armored vehicles with realistic material treatments and details/components, including cavities, thin edges, and embedded antennas.

Numerical approaches for solving electromagnetic scattering problems either in the time or frequency domain may be classified broadly into the following four categories: (1) differential equation methods, (2) integral equation methods, (3) high-frequency asymptotic methods, and (4) hybrid methods. Each numerical technique has its own advantages and trade-offs for the analysis of a particular type of application. For example, traditional asymptotic analyses are well suited for modeling the scattering properties of electrically large complex shapes (Chew et al., 2001); such analyses have difficulty treating nonmetallic material composition and volumetric complexity of the structure. Significant progress has been made in addressing the solution of the large systems of equations generated by frequency domain integral equation methods; still it is very difficult to incorporate material and device nonlinearities. Direct time-domain differential equations are usually solved with grid-based approaches. These time-domain approaches don't have the limitations of the frequency domain and high-frequency asymptotic methods. Geometric complexity and algorithmic stability, dependent on the physics of the application, dictate the grid size. For example, the size of the computational grid increases tremendously (to the order of billions of cells), with an increase in the size of electromagnetic domains and frequencies. Generating large grids and solving a large number of equations are the main drawbacks of these approaches. A scalable computing environment is needed to overcome these problems.

The field of CEMs is a highly interdisciplinary activity consisting of different areas, namely, physics, engineering, mathematics, and computer science. Development of computational algorithms incorporating underlying mathematical principles and improved physics are the cornerstones for the tremendous growth in CEMs. The advent of parallel computers and associated developments in computational algorithms to take advantage of these computers further enhanced this growth.

Finite difference time domain (FDTD) is one of the most commonly used time-domain methods for solving the Maxwell equations (Taflove, 1998). The FDTD method is an explicit time difference scheme which uses central difference on a staggered Cartesian grid (Yee grid) and is second-order accurate in both time and space. Since the finite difference scheme is based on a Cartesian grid, inaccuracies may be introduced at the model boundries, particularly with curved geometries. Use of unstructured mesh-based approaches will circumvent this problem with additional computational cost. To minimize this drawback, one can resort to enhancements to the FDTD method (higher-order methods), unstructured/structured hybrid approaches, and higher-order finite element approximations (Castillo et al. 2004). Explicit formulations are ideally suited for parallel implementation, and the basic scheme is easy to implement on parallel computers. Implementation of essential and natural boundary conditions in parallel is relatively difficult compared to the basic scheme.

Typically, solving large-scale practical electromagnetic applications consists of the following three main steps: (1) preprocessing or problem setup, (2) computational approach or application software, and (3) postprocessing or visualization and analyses. All three steps must exploit scalable computers in order to address large-scale practical Army applications. Compared to the growth and improvements in scalable computational algorithms for electromagnetic application software, there has been limited growth in the development of scalable pre- and post-processing approaches. For example, a single processor memory is not enough to generate the input file or visualize massive output from practical large-scale application. Hence, a computing environment with the capabilities for parallel preprocessing, scalable domain decomposition, scalable electromagnetic software, and parallel or run-time visualization is needed. Recently, Hassan et al. (2004) developed finite element based framework to address scalable pre- and post-processing approaches for three-dimensional time domain electromagnetic scattering applications using tetrahedral meshes. As opposed to the above tetrahedral finite element approach, the overall objective of this effort is to develop and implement scalable preprocessing and postprocessing algorithms consistent with scalable FDTD CEM algorithms for modeling very large-scale complex Department of Defense applications. The environment is developed in modular form so that it works with other grid- based CEM software.

2 Computational Formulations

2.1 Mathematical Preliminaries

Electromagnetic fields are modeled by four vector equations and put forward by the physicist James Clerk Maxwell in the middle of the 19th century. Using vector calculus, the equations can be written in concise form as

$$\nabla \cdot \mathbf{B} = 0, \tag{1}$$

$$\nabla \cdot \mathbf{D} = \rho, \tag{2}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}, \quad \text{and} \tag{3}$$

$$\frac{\partial \mathbf{D}}{\partial t} = \nabla \times \mathbf{H} - \mathbf{J}_e, \tag{4}$$

where $\mathbf{E}$(x,t) is the electric field, $\mathbf{D}$(x,t) is the electric flux density, $\mathbf{H}$(x,t) is the magnetic field, $\mathbf{B}$(x,t) is the magnetic flux density, $\mathbf{J}_e$(x,t) is the electric current density, and ρ(x,t) is the charge density. Maxwell's equations are complemented by the following continuity equation:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{J}_e. \tag{5}$$

For linear, isotropic, nondispersive materials, and nondispersive electric losses that attenuate **E** fields via conversion to heat energy, we have

$$\mathbf{B} = \mu\mathbf{H}, \quad \mathbf{D} = \varepsilon\mathbf{E}, \text{ and } J_e = \sigma\mathbf{E}, \tag{6}$$

where ε(x) is the electric permeability, μ(x) is the magnetic permeability, and σ(x) electric conductivity. Inserting Eq. (6) into Eqs. (1-4) yields the following:

$$\nabla \cdot (\mu\mathbf{H}) = 0, \tag{7}$$

$$\nabla \cdot (\varepsilon\mathbf{E}) = \rho, \tag{8}$$

$$\frac{\partial\mathbf{H}}{\partial t} = -\frac{1}{\mu}\nabla\times\mathbf{E}, \text{and} \tag{9}$$

$$\frac{\partial\mathbf{E}}{\partial t} = -\frac{1}{\varepsilon}\nabla\times\mathbf{H} - \frac{1}{\varepsilon}\sigma\mathbf{E} \tag{10}$$

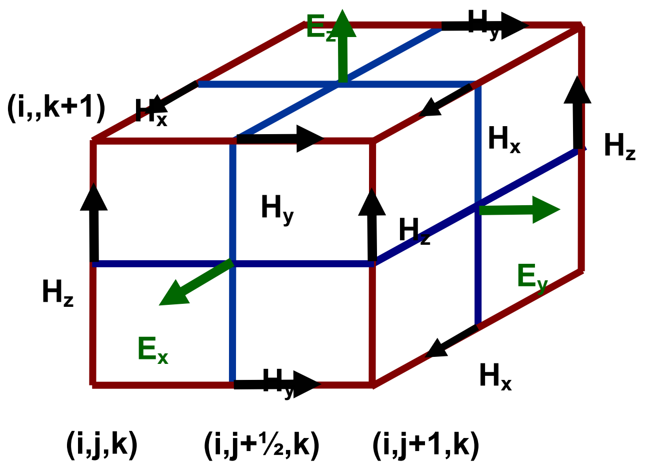


Figure 1 : basic finite difference cell (Yee cell).

2.2 FDTD Method

The FDTD approach uses standard central-difference approximations to evaluate the space and time derivatives of the partial differential equations on a staggered grid. “Staggered” here indicates that the different electromagnetic components are not located at the same place and the fields are not represented at the same time levels. The basic finite difference space grid (also known as a Yee cell (Yee, 1997) is given in Figure 1. The magnetic field components are represented on the cell’s faces and the electric field components are represented on the cell’s edges. This scheme is second-order accurate in both space and time and yields an explicit solver, which is amenable to parallel implementation.

Examples of a typical finite difference stencil of a magnetic and electric field in one direction, namely E$_x$ and H$_{x,}$ are given in Eqs. (11-12), where the subscripts (n n+1/2) represent time levels and superscripts (i,j+1/2,k+1/2) represent space. Δx, Δy, Δz, are the space increments in the x, y, and z Cartesian coordinate system, and Δt is the time increment or time step to march in time. These equations are quite straightforward to implement. Since this is a second- order accurate explicit time integration scheme, there is a limit on the time step size to ensure stability of the time integration scheme, governed by Courant-Friedrichs-Lewy (CFL) criteria. In simple terms, the time step to be used is proportional to the characteristic size of the cell.

Absorbing or radiating boundary conditions (ABC) will permit one to address electromagnetic wave interactions in unbounded regions (i.e. an outer lattice to simulate the extension of the lattice to infinity). The basic limitation with this boundary condition is that this absorbing layer is matched only to a normally incident wave. To overcome this problem, a perfectly matched layer boundary condition (Berenger, 1994) was implemented.

$$\begin{aligned} E_x\Big|^{n+1/2}_{i,j+1/2,k+1/2} &= \left(\frac{1-\frac{\sigma_{i,j+1/2.k+1/2}\Delta t}{2\varepsilon_{i.j+1/2,k+1/2}}}{1+\frac{\sigma_{i,j+1/2,k+1/2}\Delta t}{2\varepsilon_{i,j+1/2,k+1/2}}}\right) E_x\Big|^{n-1/2}_{i,j+1/2,k+1/2} \\ &+ \left(\frac{\frac{\Delta t}{\varepsilon_{i.j+1/2,k+1/2}}}{1+\frac{\sigma_{i,j+1/2,k+1/2}\Delta t}{2\varepsilon_{i,j+1/2,k+1/2}}}\right) \left(\begin{array}{l} \frac{H_z\big|^{n}_{i,j+1,k+1/2} - H_z\big|^{n}_{i,j,k+1/2}}{\Delta y} \\ -\frac{H_y\big|^{n}_{i,j+1/2,k+1} - H_y\big|^{n}_{i,j+1/2,k}}{\Delta z} \\ -J_e\Big|^{n}_{i,j+1/2,k+1/2} \end{array}\right). \end{aligned} \tag{11}$$

$$\begin{aligned} H_x\Big|^{n+1}_{i-1/2,j+1,k+1} &= \left(\frac{1-\frac{\sigma^{\Delta}_{i-1/2,j+1,k+1}t}{2\mu_{i-1/2,j+1/2,k+1}}}{1+\frac{\sigma^{\Delta}_{i-1/2,j+1,k+1}t}{2\mu_{i-1/2,j+1,k+1}}}\right) H_x\Big|^{n}_{i-1/2,j+1,k+1} \\ &+ \left(\frac{\frac{\Delta t}{\mu_{i-1/2.j+1,k+1}}}{1+\frac{\sigma^{\Delta}_{i-1/2,j+1,k+1}t}{2\mu_{i-1/2,j+1,k+1}}}\right) \left(\begin{array}{l} \frac{E_y\big|^{n+1/2}_{i-1/2,j+1,k+3/2} - E_y\big|^{n+1/2}_{i-1/2,j+1,k+1/2}}{\Delta z} \\ -\frac{E_z\big|^{n+1/2}_{i-1/2,j+3/2,k+1} - E_z\big|^{n+1/2}_{i-1/2,j+1/2,k+1}}{\Delta y} \\ -M_{source}\Big|^{n+1/2}_{i-1/2,j+1,k+1} \end{array}\right). \end{aligned} \tag{12}$$

The main advantages of FDTD-based techniques for solving electromagnetic problems are simplicity and the ability to handle complex geometries (in principle, using highly refined grids) and multimaterials, including nonmetallic. As opposed to the traditional tetrahedral explicit finite element method, this staggered formulation substantially reduces the required memory for solving the problem. However, the major disadvantage of FDTD is that Cartesian grids conform poorly to curved geometry, thus introducing so-called stair- stepping errors. Further, a huge grid is required when the dimensions of the scattering object are large compared to the input wavelength. This drawback can be addressed either by resorting to hybrid approaches or using refined higher order cells. However, the focus of this current research is to provide a capability for solving very large-scale grids and highly refined grid applications. Next, the scalable implementation of the software is discussed briefly.

2.3 Parallel Implementation of FDTD

The parallel FDTD solver is implemented using the message passing interface (MPI) allowing the flexibility to port the software to different parallel computing architectures. For more details on the parallel implementation and features, refer to Nehrbass (2001), Guiffent and Mahdjoubi (2001), Chew and Fusco (1995), and Namburu et al. (2002). Because it is based on a structured mesh, the FDTD method is an excellent candidate for parallelization using domain-decomposition techniques (Karypis and Kumar, 1998). The domain decomposition process partitions the FDTD solution domain into subspaces, and each subspace is attributed to a processor. The objective of domain decomposition is to balance computational loads and optimize communications between "N" processors such that computational time required to solve the problem is reduced by a factor of "N" (theoretically). Scalability is one of the measures to assess this speedup of the solution using multiple-processors. Perfect scalability (when the problem size is increased linearly with the number of processors, the execution time is constant) is obtained for this FDTD software on SGI O3K and IBM SP computing platforms. Note that the SGIO3K is a shared-memory parallel computer and the IBM SP is a distributed parallel computer.

3 Scalable Computing Environment

The Scalable Computing Environment developed in this work is based on a common data model and format. This data model and format takes advantage of the commonly used e**X**tensible **M**arkup **L**anguage (***XML***), hence we coined the term e**X**tensible **D**ata **M**odel and **F**ormat (***XDMF***) (Clarke and Namburu, 2001). XDMF efficiently consolidates large quantities of data for preprocessing, postprocessing, and run-time visualization on distributed parallel processors. This common data model and format or XDMF is used in conjunction with the FDTD computational simulations in order to alleviate the complexities of communicating data between different computational topologies, preprocessing, partitioning software, postprocessing, and applications software. For example, data can be written to the XDMF utilizing one partitioning scheme, and read back with a totally different scheme. Instead of imposing a new programming paradigm on electromagnetic software on parallel computers, XDMF uses the existing concept of file I/O for distributed coordination to address large-scale applications, utilizing Network Distributed Global Memory (***NDGM***), Hierarchical Data Format version 5 (***HDF5***), and eXtensible Markup Language (***XML).***

XDMF is both a data model and format and allows for a self-describing method of storing large data structures and the information necessary to tell how the data is to be used. This makes the development of reusable pre- and postprocessing tools possible. "Data model" refers to the intended use of the data. For example, a three-dimensional (3-D) array of floating point values may be the X, Y, and Z geometry for a grid or calculated vector values. Without a data model, it is impossible to tell the difference. Since the data model only describes the data, it is purely light data and thus stored using XML. The data model is targeted at scientific simulation data concentrating on scalars, vectors, and tensors defined on some type of computational grid. Both structured and unstructured grids can be described via their topology and geometry. Calculated, time-varying data values are described as "attributes" of the grid. The actual values for the grid geometry, connectivity, and attributes are contained in the data format. This separation of data format and model allows FDTD software to efficiently produce and store values in a convenient manner without being encumbered by large-scale data, which may be different from their internal arrangement. Utilizing the common data model and format, FDTD can produce and consume data just as writing and reading any other data file. These data “files,” however, can exist in a network distributed shared memory system (called NDGM), which has barriers and semaphores to help coordinate parallel activity. This is what makes XDMF more than just another file format. In addition, a C++ class library is provided mainly as a convenience layer. Preprocessing, domain decomposition, FDTD, and postprocessing use this layer from C++, C, or FORTRAN to easily access any XDMF functionality. In addition, this layer has been “wrapped” for access from Tcl, Python, and Java.

The FDTD solver was incorporated into the scalable computing environment in a modular form. That is, in addition to be able to use various libraries in this environment, the compatible I/O files required for the solver are easily generated with XDMF. Integrating into this environment does not require any modifications to the solver. First, the solver is compiled for the IBM-SP3 and the SGI Origin 2000 and 3000 platforms and linked with the rest of scalable computing environment. Since XDMF is part of this environment, all the required input files, domain decomposition info, communication files, and output files were provided to the solver very easily.

4 Model Preparations and Scalable Visualization

The major difficulty in model preparation or preprocessing is converting the available geometric data for complex configurations into a suitable format for simulations. For example, complex configurations are available in computer-aided design (CAD) format either as volume representations and/or surface representations. In some cases, only surface representation from scanned image file formats are available. CAD geometries are ideally suited for generating unstructured meshes of a desired refinement (theoretically) for high-frequency CEM simulations. However, facet files and some forms of scanned images are not well suited for generating appropriate volume meshes efficiently. Unfortunately, most of the geometries available for military applications fall into this category. Hence, our following discussions are focused towards working with facet file representation to generate volumetric data suitable for a finite- difference, time-domain computational approach.

The next difficulty in representing the geometry of the solution domain is the quality of mesh as dictated by the numerical approach, material properties, boundary conditions, and source conditions. It is well known that this is one of the most time-consuming activities and is usually a major bottleneck in complex simulations.

4.1 Grid Generation

Traditionally mesh generation is broadly divided into two distinct classes– structured mesh generation and unstructured mesh generation. Creation of structured meshes based on transformations and mappings usually proceeds by creating a structured mesh of the application domain with regular polyhedron and then mapping the topology of the meshed object into a structured mesh domain. The structured mesh typically represents a volume. Generation of the volume from the available unstructured topology of the meshed object or surface representation of the object is challenging for complex geometries. Similar challenges exist in creating a volumetric representation of the object or solution domain for unstructured meshes starting from surface mesh of the object. Unstructured mesh generation techniques are typically based on tetrahedral elements (Said et al., 1999) for 3-D volume configurations. With the advent of fully automatic mesh generators, unstructured mesh methods have proven to be particularly attractive for addressing preprocessing of complex 3-D configurations. In unstructured meshing starting from CAD geometries, advancing front techniques (Chan and Anastasiou, 1997), Delaunay triangulation (Krysl and Ortiz, 2001), and the combination of the two (Borochaki et al., 2000 and Radovitzky and Ortiz, 2000) are quite common. A major issue in triangulation techniques is ensuring that individual elements have high-quality shapes. Despite the good properties of triangulation techniques, they still require much user preprocessing both to determine material boundaries and to adjust the mesh size. To suit the needs of the application and adaptive computations, refinement of the mesh is needed. Degeneration of element shape and aspect ratio under repetitive local mesh refinements (Cougny and Shepherd, 1999) is one major issue and is still under active research.

Electromagnetic simulations at higher frequencies of battlefield configurations demand generation of high- resolution meshes on the order of billions of mesh entities. Hence, mesh generators need to utilize parallel computers to address generation of large meshes because single-processor computers cannot handle this. When the generation of large grids is attempted with the current parallel unstructured mesh generators, it is found that these approaches are too slow and the memory requirements needed for the preprocessing requires much more than solving the problem. Hence, there is a clear need for fast, efficient parallel mesh generators.

When compared to tetrahedral elements, hexahedral elements (e.g., the staggered solution procedure with Yee cells) require less memory, have better stablilty per time step, and require fewer elements to model the entire domain. Hence, the use of hexahedral elements is more attractive for electromagnetic simulations. However, the major drawback of hexahedral elements is that they cannot model curved surfaces and complex geometric entities accurately. It results in so-called stair stepping at the boundaries. Tetrahedral elements are more attractive for modeling curved geometries and complex geometric entities. Hence, a combination of hexahedral elements with a modest number of tetrahedral elements is the desired approach, (i.e., a hybrid approach combining both structured and unstructured mesh generation into one analysis is ideal).

Lacking fast and robust scalable unstructured mesh generators for generating very large-scale meshes and efficient structured mesh generators for surface representations, we explored voxel-based structured meshing approaches. Voxel-based structured meshing is widely used in the bioengineering and engineering community (Hollister and Kikuchi, 1994). Most of these approaches are based on 3-D volume scanned images. To address large-scale grid generation of complex geometries (starting from CAD geometries and surfaces based on image representation), we developed scalable methodologies using voxel-based concepts in conjunction with an implicit surface approach.

4.2 *Scalable Grid Generator Based on Voxel/Implicit Surfaces*

The steps involved in grid generation discussed in this paper are as follows:

(a) Facet file representation from scanned images.

(b) Covert the facet file format into XDMF using simple XML scripts.

(c) Check normals on the facets.

(d) Map the facets onto a coarse Cartesian grid and decompose the domain.

(e) Generate desired thickness using implicit surfaces concept.

(f) Generate desired refined voxels and identify voxels that overlap the extruded polygons on each processor.

(g) Generate the desired communication information of the generated voxel grid for the parallel FDTD

The first step is to covert the given coarse facet file representation into the XDMF. Several conversion utilities were incorporated into the scalable computing environment, which utilizes simple XML tags to convert various formats of facet/mesh file representations into XDMF. In addition, a simple file viewer is implemented for visually confirming the correct conversion of the model, including representation of facet normal directions.

With the model in XDMF format, we now need to give appropriate volume to the facet file to generate the thickness for the model. One easy way to approach this problem is to use a nodal approach, which provides appropriate thickness to the facet files. However, it is very time consuming for complex geometries. To address this problem, we employed an implicit surface modeling technique. This technique is widely used in the computational graphics community (Bloomenthal, 1988; Chadwick et al., 1989). Traditional implicit surface approaches are based on analytical equations. Recent developments in this research area demonstrate the generation of fast, efficient, and accurate layered models using a skeleton concept. Skeletons are simple geometric primitives such as points, curves, triangles, and volumes. Skeletons can be manipulated to produce implicit isosurfaces such as “distance surfaces.”

Voxel mesh generation and domain decomposition are handled in one step. An MPI-based utility was developed that places the coarse surface geometry representation into a coarse voxel domain of a specified block size and then decomposes the coarse voxel domain along with the coarse facet representation. Next, the extrusion depth or layer thickness is specified to create a volume representation that depends on the required thickness or material configuration of the object. This results in a model with the same external dimension with thickness. The thickness can vary for each skeleton. Simple homogeneous extrusion may cause some internal geometry to overlap, particularly inside convex outer surfaces. The desired number of volumetric cells can be generated utilizing these polygons. Using this method, a polygon mesh can be generated very quickly from a faceted or surface representation. However there are several potential pitfalls. As with many model conversion schemes, the direction of the original vertex normals is critical. A second chance for error arises if the extrusion depth exceeds the thickness of any of the convex surfaces of the model. This will cause the geometry to extend back through the model surface. Some of these errors can easily be corrected with simple file viewer checks.

The next step involves identifying voxels, which overlap with the polygon mesh. The standard approach is to use ray-counting and winding number methods (Jin-Fa Lee and Nehrbass, 2001). The ray-counting algorithm determines whether or not a point “P” lies within any object ”O.”, The winding number method determines whether or not the intersection of a ray and a surface lies within a surface and is therefore counted as a valid intersect. A combined ray-counting and winding-number approach will give more accurate results. However, this approach is very expensive for a large-scale voxel mesh. Hence, we adopted a different approach based on simple checks for each of the voxels.

The grid generation is then handled in parallel on a user-specified number of processors. Each processor is given a single computational block to grid the entire block, searching for the cells that overlap the polygon. Cells are assigned a material value if the model is found to be at that cell location; otherwise, it is assumed to be free space. (Figure 2). When a given processor completes its task, it is given another block until the mesh is complete. The desired number of volume cells can be generated through the thickness of polygons (i.e., properties for various material layers can be identified directly for the corresponding the volume cell).

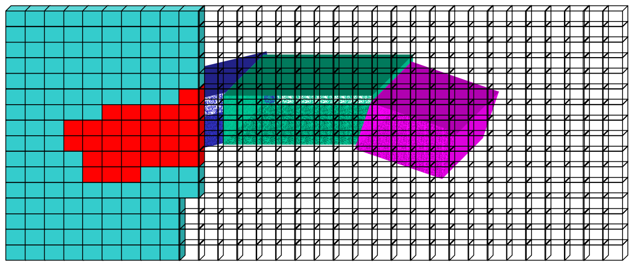


Figure 2 : Implicit modeler marches through its segment of the domain generating cell material values based on location of geometry.

Upon completion of the mesh, an input template file is created in the format required by the solver. Included in the input file is the distribution of blocks to the requested number of processors. The FDTD solver utilizes a perfectly matched layer (PML) of absorbing boundary condition around the computational domain, providing an absorber region that prevents computational reflection back into the domain. The FDTD solver implicitly accounts for this region; therefore, it is unnecessary to physically define these blocks. However, they are accounted for during the decomposition process. Load distribution is accomplished by dividing the number of computational blocks by the number of requested processors. Based on this, each computational block, or group of blocks, is distributed to a processor during execution time. Any adjoining absorber blocks are assigned to that same processor to reduce interprocessor communication. Since the grid is generated from the implicit mesh information and number of blocks, the required grid can be generated using MPI on any number of processors. For example, the grid can be generated on “M” processors if the desire is to execute the job on “M” processors or the grid can be generated on “N” number of processors ($N << M$, with sufficient memory to generate grid) and can later be spread onto “M” processors. The input template file is edited to specify the desired frequency range, number of time steps, values of PML conductivity, desired volumetric output, and any other desired options.

4.3 *Visualization of Large Data*

Visualization of the very large data generated from the scalable simulations is a major problem for the analyses of results. For example, one of the numerical test cases discussed in this paper used 384 processors to simulate the problem and generated about 2 Tbytes of output for one simulation. Gathering all the information onto one processor to perform visualization is impossible. Hence, in our work, we adopted the scalable computing environment discussed in this paper for parallel visualization.

Parallel FDTD software generates RCS and volumetric data for each grid associated to the computational block on each processor. To visualize volumetric data of the entire domain, the block must be reassembled. The scalable computing environment uses NDGM, along with XDMF for this purpose. NDGM is a user level, heterogeneous shared memory system that creates and manages a potentially distributed shared memory buffer, which can be accessed. Based on a user request, requested data from each processor will be communicated back to the user. The user request will be updated and/or synchronized for each gathering. Directly accessing the data from memory substantially improves the performance. This facilitates accessing of the previous time step data in near real time. That is, visualization can be performed while simulations are going on.

5 Numerical Test Cases

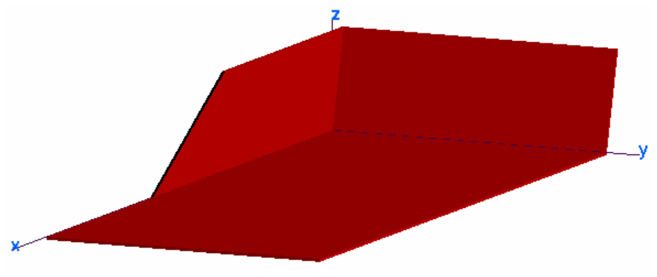
5.1 T5M3 Left Front Trihedron

To validate the scalable FDTD approach, a voxel-based model was generated using the available facet representation of the T5M3 left front trihedron (Figure 3). This is one of the standard benchmark problems developed at the National Ground Intelligence Center (NGIC). Figure 3 shows the shape and dimensions of the trihedron (Spurgeon et al., 2003). For this problem, RCS simulations of the parallel finite-difference time-domain method were compared with the available experimental results. Simulations at 10 GHz

were run for various depression angles. Results at a 10° depression angle are shown in Figures 4 and 5. As can be seen for this benchmark problem, FDTD compares very well with the experimental results. To assess the parallel performance of the FDTD solver, a small series of calculations were run using a trihedron model as a test case (Figure 3). A 1037 x 468 x 184 (89 million cells) mesh was generated and decomposed into 360 computational blocks. Table 1 lists the comparison for each calculation run on the ARL IBM-SP (512 - 375 MHz Power3 PEs). The computational blocks were divided evenly across the requested number of processors. No attempt was made to distribute in a manner that reduced interprocessor communication.

Number of processors	Wall Clock Time
32	8.5
64	4.9
128	2.8

Table 1 : Parallel performance of simulations.



(1.4 m x 0.6 m x 0.2 m)

Figure 3 : T5M3 trihedron geometric model.

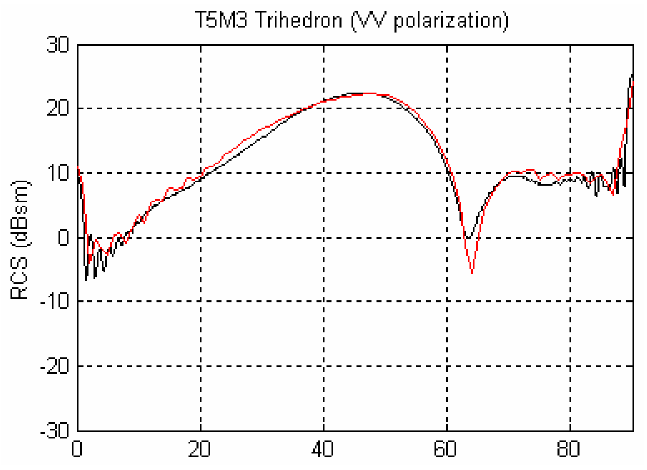


Figure 4 : Comparison with experimental results for VV polarization.

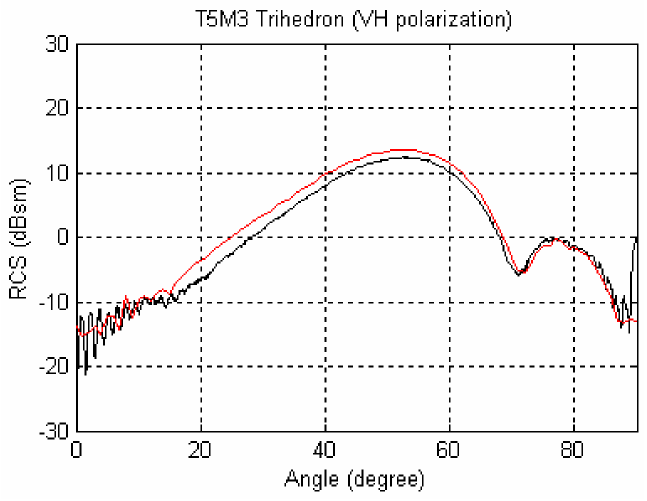


Figure 5 : Comparison with experimental results for VH polarization.

Note that the scalability numbers given in Table 1 are for the entire simulation starting from finite difference time-domain software to run-time visualization. Next, scalability studies for grid generation alone for this trihedron example are evaluated. The time required to generate an 81-million cell grid, including the appropriate input file with communications information on 8 processors, is 3 minutes and 41 seconds. Memory required for this example is about 150 Mbytes for each processor.

5.2 *Full-Scale Ground Vehicle at 10 GHz*

The objective of this test case is to advance first principle electromagnetic calculations to evaluate full-scale vehicles at X-band. As a test data set, we have chosen a full-scale ground vehicle for testing, the ZSU23-4 antiaircraft vehicle. This vehicle was selected because it has been thoroughly analyzed with traditional signature assessment tools and has been the subject of extensive range measurements. A high-resolution geometric facet model was obtained and translated into XDMF format. To evaluate the vehicle at 10 GHz, the maximum cell size was determined to be 3 mm by assuming the least acceptable resolution to be approximately 9-10 cells per wavelength. Based on a desired 3 mm cell size, each facet in the XDMF model was given a thickness of 1.6 cm (approximately 5 x 3 mm) by extruding as previously described. The final mesh was then generated using the computing environment tools. The result was a 1984 x 1043 x 1188 grid (2.46-billion cells) with a cell thickness of 3.2 mm. The mesh was divided into 384 actual blocks with additional 416 implied absorber blocks surrounding the problem space. The calculation was run to 17000 time steps on the Army Research Laboratory (ARL) Major Shared Resource Center (MSRC) IBM-SP3 on 384 processors (PEs). Total wall clock run time was approximately 32.6 hours for a single incident/azimuth angle. This calculation generated approximately 2.0 TB of volumetric output. An example of the volumetric output at two different time steps is shown in Figures 6 and 7.

This effort has successfully demonstrated the ability to efficiently generate and decompose very large-scale meshes from existing facet geometry utilizing parallel techniques.

Parallel visualization techniques were developed to interact with the large amounts of volumetric data generated by the solver during run time.

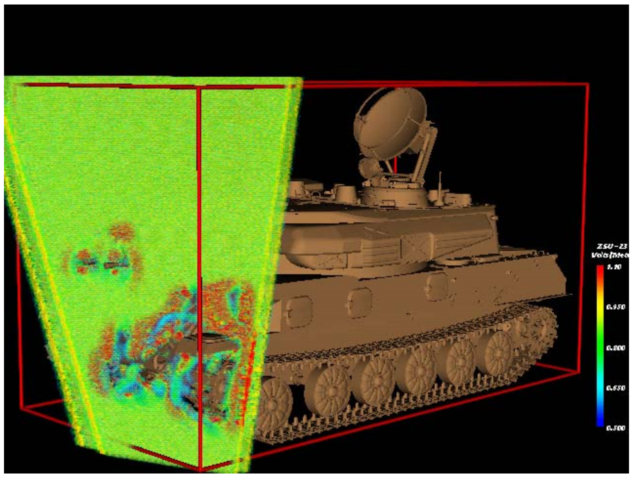


Figure 6 : An incident pulse (centered around 10 GHz) impinging the armored vehicle – magnitude of E field.

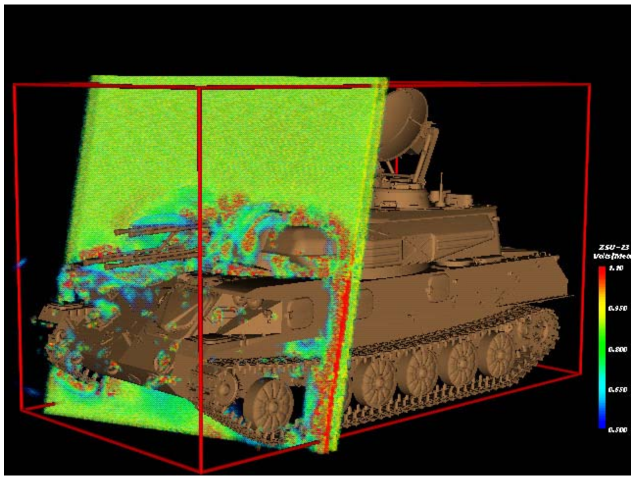


Figure 7 : An incident pulse (centered around 10 GHz) impinging the armored vehicle – magnitude of E field at a later time.

6 Concluding Remarks

This effort successfully demonstrated the ability to efficiently generate very large-scale rectilinear meshes from existing CAD or facet geometry utilizing parallel techniques within the scalable computing environment. The Parallel FDTD solver has been validated with the available experimental data. Scalability of the application within the computing environment starting from preprocessing, finite difference time-domain software and run-time visualization were also demonstrated. Parallel visualization techniques have also been developed to interact with the large amounts of volumetric data generated by the Parallel FDTD solver. The Scalable Computing Environment, with the addition of the tools previously described, has proven to be an effective means of generating input in utilizing multiple processors and post-processing for very large-scale electromagnetic calculations. This capability, along with the high-performance computing assets, allows the solution of a new class of electromagnetic simulations.

In this study, we focused our efforts towards developing a scalable computing environment to address practical Army applications utilizing scalable computers. This basic framework can be used to enhance overall modeling efforts in CEM. Some of the areas that we are planning to address in future work include adding multiple material support to the implicit modeler during mesh generation, improving load balancing, modifying the Parallel FDTD solver to add binary data transfer, streamlining the visualization process, improving the accuracy of the numerical approaches (higher order methods), developing hybrid methods (e.g., finite-difference/finite element, finite-difference/method of moments, etc.), and developing scalable structured-unstructured hybrid meshing algorithms.

Acknowledgement: This work was supported by the ARL Director's initiative on signature modeling. The authors wish to thank Calvin Le, ARL, and John Nehrbass, Ohio Supercomputer Center, for their guidance and assistance. Authors also wish to acknowledge the ARL-MSRC's high-performance computing assets and services.

References

Berenger, J. P. (1994): A Perfectly Matched Layer for the Absorption of Electromagnetic Waves, *Journal of Computational Physics*, 114(1), pp. 185-200.

Bloomenthal, J. (1988): Polygonization of Implicit Surfaces, *Computer Aided Geometric Design*, 5, pp. 341-355.

Borochaki, H.; Lang P.; George, P. L. (2000): Parametric Surface Meshing Using a Combined Advancing Front Generalized Delaunay Approach, *International Journal for Numerical Methods in Engineering*, 49, pp. 233-259.

Castillo, P.; Koning, J.; Rieben, R.; White, D. (2004): A Discrete Differential Forms Framework for Computational Electromagnetism. *CMES: Computer Modeling in Engineering & Sciences*, vol. 5, no. 4, pp. 331-346.

Chadwick, J. E., et al. (1989): Layered Construction for Deformable Animated Characters, *Computer Graphics*, 23(3), pp. 243-252.

Chan, C. T.; Anastasiou, K. (1997): An Automatic Tetrahedral Mesh Generating Scheme by the Advancing Front Method, *Communications in Numerical Methods in Engineering*, 13 pp. 33-46.

Chew, K. C.; Fusco, V. F. (1995): A Parallel Implementation of the Finite Difference Time Domain Algorithm, *International Journal of Numerical Modeling Electronic Networks, Devices and Fields*, 8, pp. 293-299.

Chew, W. C.; Jin, J.; Michielssen, E.; Song, J. (2000): *Fast and Efficient Algorithms in Computational Electromagnetics*, Artech House.

Clarke J.; Namburu, R. (2001): The eXtensible Data Model and Format: a High performance Data Hub for Connecting Parallel Codes and Tools, *ARL-TR-2552*, July.

Cougny, H. L.; Shepherd, M. S. (1999): Parallel refinement and coarsening of tetrahedral meshes, *International Journal for Numerical Methods in Engineering*, 46, pp. 1101-1125.

Guiffaut, C.; Mahdjoubi, K. (2001): A Parallel FDTD Algorithm Using The MPI Library, *IEEE Antennas and Propagation Magazine*, 43, (2), pp. 94-102.

Hassan, O.; Morgan, K., Jones, J.; Larwood, B.; Weatherhill, N. P. (2004): Parallel 3D Time Domain Electromagnetic Scattering Simulations on Unstructured Meshes. *CMES: Computer Modeling in Engineering & Sciences*, vol. 5, no. 5, pp. 383-394.

Hollister, S. J.; Kikuchi, N. (1994): Homogenization Theory and Digital Imaging: A Basis for Studying the Mechanics and Design Principles of Bone Tissue, *Biotechnology and Bioengineering*, 43(7), pp. 586-596.

Karypis, G.; Kumar, V. (1998): Multilevel k-way Partitioning Scheme for Irregular Graphs, *Journal of Parallel and Distributed Computing*, 48(1), pp. 96-129.

Krysl, P.; Ortiz, M. (2001): Variational Delaunay Approach to the Generation of Tetrahedral Finite Element Meshes, *International Journal for Numerical Methods in Engineering*, 50(7), pp. 1681-1700.

Lee, J-F.; Nehrbass, J. (2001): Automatic Grid Generations for FDTD Computations, Electromagnetics Code Consortium Conference, Kaui, Hawaii.

Namburu, R.; Mark, E.; Clarke, J. (2002): Computational Electromagnetic Methodologies for Combat System Applications, proceedings of 23^{rd} US Army Science Conference, Orlando, FL.

Nehrbass, J. (2001): Computational Electromagnetics Workshop Notes, held at U.S. Army Research Laboratory, Adelphi, MD.

Radovitzky, R.; Ortiz, M. (2001): Tetrahedral Mesh Generation Based on Node Insertion in Crystal Lattice Arrangements and Advancing Front-Delaunay Triangulation, *Computer Methods in Applied Mechanics and Engineering*, 187, pp. 543-569.

Said, R.; Weatherhill, N. P.; Morgan, K.; Verhoeven, N. A. (1999): Distributed Parallel Delaunay Mesh Generation, *Computer Methods in Applied Mechanics and Engineering*, 177, pp. 109-125.

Spurgeon, W. A.; Stratton, S. R.; Tan, R. J. (2003): Outdoor Radar Range Cross-Section Measurements on the T5M3 Trihedron," U.S. Army Research Laboratory. Submitted for publication as an ARL technical report.

Taflove, A. (1998): Advance in Computational Electrodynamics: The Finite Difference Time Domain Method, Artech House.

Yee K. S.; Chen, J. S. (1997): The Finite-Difference Time-Domain (FDTD) and the Finite-Volume Time-Domain (FVTD) Methods in Solving Maxwell's Equations, *IEEE Transactions On Antennas and Propagation*, 45, (3), pp. 354-363.