AUTOMATIC AND EFFICIENT SURFACE FDTD MESH GENERATION FOR ANALYSIS OF EM SCATTERING AND RADIATION*

Weimin Sun, Mike P. Purchine*, Jian Peng, Constantine A. Balanis, and George C. Barber**
Telecommunications Research Center, Arizona State University
Tempe, AZ 85287-7206
* Communications Semiconductor Products Division, Motorola
Phoenix, AZ 85062-2953
** JRPO/CECOM/NASA, NASA Langley Research Center
Hampton, VA 23681-0001

Abstract

The finite difference time domain (FDTD) method has found very wide applications in the analysis of electromagnetic scattering and radiation. The first and most important issue of the FDTD modeling is to decompose a computation space, containing a given geometry, into FDTD unit cells. There are a few dedicated mesh generators which could discretize the space into cells for general analysis. However, among FDTD applications, a large portion of them deal with objects structured primarily with conducting and/or thin-dielectric plates, such as a conducting sphere or cube, a cavity, an airplane, etc.. The mesh data necessary to input are those of node indices and material parameters on the object surface. Consequently geometry modeling is essentially to generate FDTD cells on the surfaces. For this purpose, a simple and effective algorithm capable of on-surface FDTD mesh generation is introduced based on a ray-tracing method. The algorithm presumes that the input geometry is described in polygons and lines which are often approximations of smooth surfaces and thin wires. In output, the algorithm decomposes automatically the polygons and lines into on-surface cells compatible with, and readable by, an FDTD solver. The algorithm has been coded in programs allowing effective and automatic generation of surface cells on various high- or low-end computer platforms.

*This work was sponsored by the Advanced Helicopter Electromagnetics (AHE) Program and NASA Grant NAG-1-1082.

1 Introduction

The finite difference time domain (FDTD) method has been demonstrated to be both powerful and versatile for modeling complex electromagnetic problems [1]. However, a complex geometry has to be discretized first to apply the method. While there are many mesh generators developed primarily for a finite element method, relatively few automatic mesh generators are dedicated to the FDTD method. Though an FDTD mesh is simpler and easier to generate, it has unique lattice-like features and needs some special consideration. Especially when the surface of a geometry cannot be represented by simple analytic formulas, an automatic mesh generator is necessary and desirable.

In most publications reporting FDTD applications, Yee's offset cube cell model [2] has been overwhelmingly used. Based on the model, a computation domain is decomposed into a lattice of cube cells in Cartesian coordinates. As shown in Figure 1, for each unit cell, the electric and magnetic field components are positioned so that not only central differences in space can be applied, but also the fundamental physical laws are naturally obeyed. Geometry modeling or mesh generation of a complex object is thus to establish an FDTD lattice of Yee's cells and provide the position and medium parameters for each cell. At the same time, the object volume is best body-fitted with unit cells.

In an FDTD lattice, cells filling non-occupied space always have medium parameters of free space; only their cell positions are important. Moreover, if a cell in the lattice is indexed by three indices which

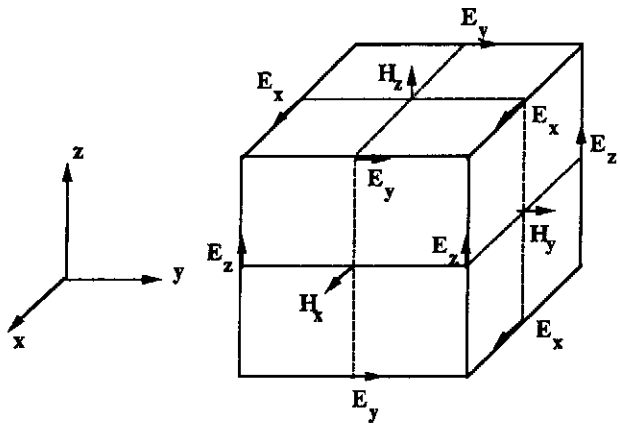


Figure 1: FDTD unit cell – Yee's model

are proportional to coordinates of the cell, its position is then determined solely by the origin and the lattice constant (cell size). The burden of geometry modeling lies in the provision of cell indices and medium parameters of cells occupying object volume. In general, the cell data are generated from a 3D volume space and need large storage space.

In a large portion of electromagnetic scattering and radiation problems a complex object is structured with shells of conducting and/or thin-dielectric materials. Now the necessary input is the indices and medium parameters of the cells distributed on the object surface. Thus it is appropriate to generate only on-surface mesh cells. Consequently, very efficient algorithms can be developed and implemented even on a very low-end personal computer.

This paper presents a program developed at Arizona State University to generate automatically three-dimensional surface FDTD meshes for complex objects. The program is based on an efficient ray-tracing method [3], and has been tested in various numerical simulations [4].

2 Algorithm Development

As a starting point, the algorithm assumes that a given surface geometry can be approximated by a cluster of polygons and lines. More accurate geometry modeling usually requires larger amount of polygons. Their mesh generation is actually to embed these polygons and lines into an FDTD lattice of Yee's cells. The word "embed" means that a polygon is approximated into stair-cased unit squares, each of which is aligned with a cell face. These squares are best fit to the original polygon. To do so, we project respectively the polygon onto three Cartesian coordinate planes. In each projection, the unit squares shaded by the projected polygon are identified. The distance of each shaded unit square to the polygon in the projection direction is evaluated and then the unit square is displaced the same distance in the projection direction. How to judiciously identify a unit square, which is covered by the projection of a polygon, is the focus of the algorithm, which can resort to a ray-tracing method and the rules set in the following sections.

Ray tracing method

Since any polygon with more than three vertices can always be tessellated into triangles, it is sufficient only to implement ray tracing on a triangle. Ray tracing is a well developed method to determine whether a ray intersects with a triangle in the 3D space. As shown in Figure 2, a ray originates from point s on the yz plane with coordinates of (0,j,k). A given triangle has three vertices represented by the three vectors o, p, and q. Its normal direction is given by

$$\mathbf{n} = (\mathbf{o} - \mathbf{p}) \times (\mathbf{q} - \mathbf{p}) \tag{1}$$

If an intersect point is represented by vector r, then vector r must satisfy

$$\mathbf{n} \cdot \mathbf{r} + c = 0 \tag{2}$$

where c is the distance of the triangle to the origin which can be obtained from $-\mathbf{n} \cdot \mathbf{p}$. In Figure 2, one can see that

$$\mathbf{r} - \mathbf{p} = \alpha(\mathbf{q} - \mathbf{p}) + \beta(\mathbf{o} - \mathbf{p}) \tag{3}$$

and the intersection distance d is given by

$$d = -(c + \mathbf{n} \cdot \mathbf{s})/\mathbf{n} \cdot \hat{d} \tag{4}$$

with $\hat{d}$ being a unit vector in the ray direction.

The sole condition which guarantees that the intersect point r is inside the triangle is [3]

$$\alpha \geq 0, \quad \beta \geq 0, \quad and \ \alpha + \beta \leq 1 \tag{5}$$

Thus a ray that intersects with a triangle can be identified from the solution of α and β in (3). To

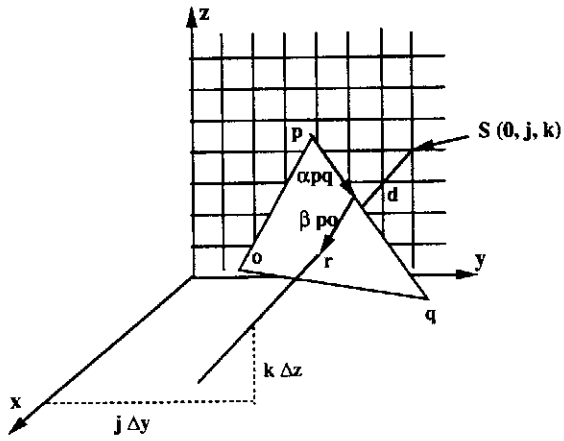


Figure 2: Illustration of a ray-trace algorithm

solve α and β from (3), only two component equations are required. By referring to Figure 2, the y and z components of (3) can be utilized as

$$r_z - p_z = \alpha(q_z - p_z) + \beta(o_z - p_z) \quad (6)$$

$$r_y - p_y = \alpha(q_y - p_y) + \beta(o_y - p_y) \quad (7)$$

All vector component coordinates in (6) and (7) are known, therefore α and β can be directly solved. Subsequently it can be determined whether the ray, originating from (0, j, k), intersects with the triangle.

Rounding off a triangle

From the established ray tracing method, it can be determined whether a ray hits a triangle, or in other words, whether the point s is covered by the projection of the triangle on the yz plane, as shown in Figure 2. To approximate a triangle by stair-cased square cell faces, we need to find all squares, which are displaced cell faces in the projection direction, covered by the projection of the triangle. However, to determine whether a unit square is covered by a triangle, some rules need to be established. The rules used in this algorithm include:

- if the center of the square is covered by the projection, then the square is displaced. Otherwise,
- if a center of one of the edges of the square is hit by the triangle, the edge (short line segment) is displaced. The edge will be used to force tangential boundary condition in the FDTD solution procedure. Otherwise,
- no squares and edges are displaced

The output of a square is different from that of a short segment. A square output will inform an FDTD solver to force boundary conditions at all four edges, while a short segment will inform the solver to force boundary conditions only at the segment.

The second problem in rounding off a triangle to FDTD cells is to round off the triangle vertices to the nearest cell nodes. It is considered necessary for the rules established above to be implemented without complication. Otherwise, more possibilities, concerning how a unit square is covered by a triangle, need to be considered in the algorithm. It should be noted that before rounding off its vertices, the triangle has to be displaced to yield a least rounding-off error. This fact is best illustrated in Figure 3 where a triangle is rounded off to unit squares and edge segments. Figure 3(a) is an original triangle. Its vertices are rounded off to nearest nodes in Figure 3(b), and applying the above rules results in an FDTD lattice compatible triangle as shown in Figure 3(c). However, if the triangle in Figure 3(a) is displaced first as shown in Figure 3(d) and then rounded off as in Figure 3(e), the resulting approximate triangle in Figure 3(f) is substantially different from that in Figure 3(c).

A complex structure consists of a cluster of polygons and lines. Any displacement should be applied to all polygons and lines. The criterion governing the displacement is to minimize the overall vertex rounding-off error. Minimization of the global vertex rounding-off error is a least square problem the solution of which could be very time consuming. When one is concerned about development of an efficient algorithm, it is only necessary to minimize the global rounding-off error in subdomains. For example, the absolute global error in rounding vertices to their nearest cell nodes can be considered in one dimension at a time and defined as

$$\epsilon_x = \min_j \sum_{i=1}^{N} |r_{ix} - d_x^j - Round[r_{ix} - d_x^j]| \quad (8)$$

where N is the total number of vertices of all polygons, r_{ix} is the x coordinate of the ith vertex, and d_x^j is a displacement in x direction which is the x-distance needed to round off the jth vertex to its nearest node. The *Round* operator in (8) rounds an

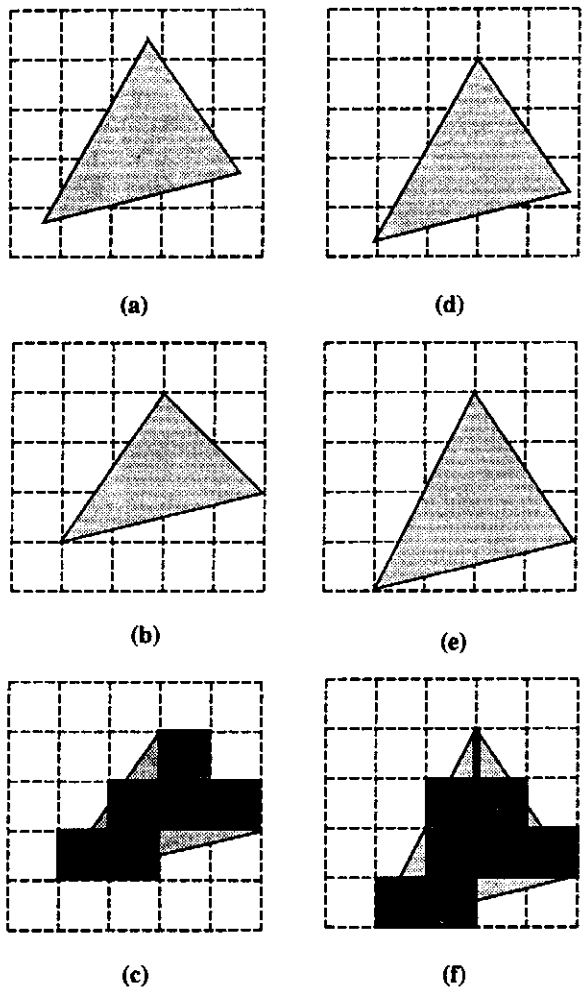


Figure 3: Algorithm illustration for rounding off a triangle. a) A triangle projected onto a coordinate plane; b) Round off the vertices of the triangle; c) Round off the triangle (in light shadow) to FDTD cell compatible squares (in dark shadow); d) Displace the triangle first; e) Then round off the vertices of the triangle; f) After that round off the triangle.

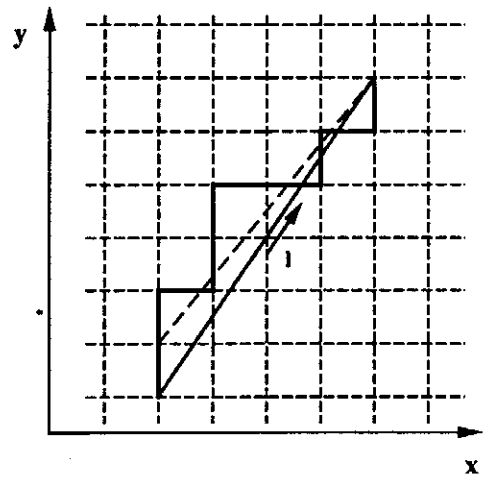


Figure 4: Digitizing a line in 2D space.

non-integer number into an integer. By choosing d_x^j to be the x distance of the jth vertex to its nearest node each time with a j for N times, the minimal global error ϵ_x and thus the d_x can be identified. This procedure is then repeated for both the $\hat{y}$ direction and $\hat{z}$ direction. The resulting d_x, d_y, and d_z define the global displacement of the object.

Digitizing a line

At low frequencies, thin wire antennas are very often employed. The geometry of a thin wire can be modeled by a line. For tangential boundary conditions to be applied on the wires, their representative lines have to be compatible with the FDTD lattice in which they are embedded. A line in an FDTD lattice is only allowed to run through cell edges due to the fact that electric field components in a cell are positioned at, and aligned with, cell edges.

Figure 4 illustrates the rule used to digitize a line in a two-dimensional space. A line is in the direction $\mathbf{l}$. If the inner product of $\mathbf{l}\cdot\hat{x}$ is greater than $\mathbf{l}\cdot\hat{y}$, a unit line segment in $\hat{x}$ direction is chopped off. The line is terminated at a new end. Repeating the procedure leads to a zig-zag line which is a digitized line with all its segments aligned with either the $\hat{x}$ or $\hat{y}$ direction. The same line digitizing rule has been applied to the three-dimensional space and implemented in the algorithm.

Algorithm

To summarize, the algorithm proposed for an on-surface FDTD mesh generation can be presented as:

1. Divide the surface of a given object into a cluster of polygons and use different colors to identify different media. The shape and the number of utilized polygons determine the accuracy of geometry modeling.

2. Round off the vertices of each triangle to nearest mesh nodes (represented in integer indices) with a minimal global rounding-off error respectively in $\hat{x}$, $\hat{y}$ and $\hat{z}$ directions.

3. Tessellate each polygon into triangles. If the vertices of the polygon are numbered clockwise from 1 to N, then the tessellation is done by connecting vertices i and j with

$$|\mathbf{r}_i - \mathbf{r}_j| < |\mathbf{r}_i - \mathbf{r}_k|$$

for $k, j \in (1, 2, ..., N)$ and $k, j \neq i-1, i, i+1$.

4. Project each triangle, respectively, onto three principal coordinate planes and scan the triangle by using the ray-tracing method, as shown in Figure 2. When a ray in a normal direction originates from a node and hits the triangle, the node indices **i** and **j** are collected and the distance **d** between the ray intersection point and the node is evaluated. Whether a ray hits the triangle can be determined via the criterion of (5).

5. The intersection distance **d** is rounded off to the nearest integer. The three integer indices (two from the coordinate plane and one from the distance), and the color of the intersection point are output as a surface cell or a short segment.

6. Repeat the above procedure for every triangle and every polygon.

7. Digitize a line into short FDTD cell compatible segments, and output the indices and direction of each individual segment.

3 Numerical Example

The above algorithm has been coded in both C and FORTRAN, and tested in various numerical simulations. For brevity, we only illustrate a few examples here.

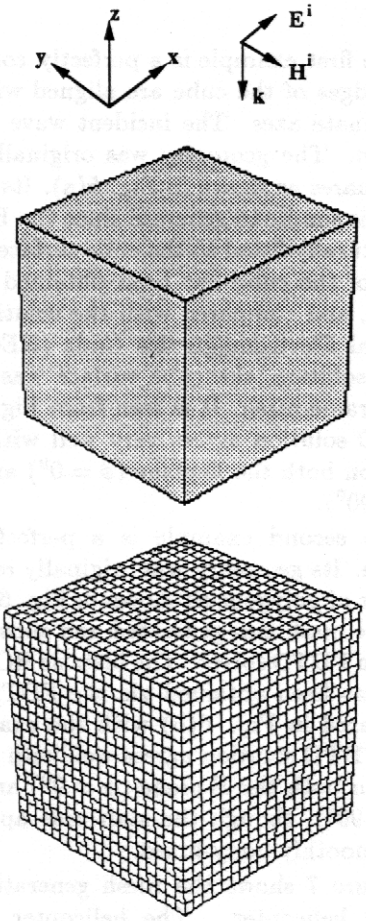
The first example is a perfectly conducting cube. The edges of the cube are aligned with rectangular coordinate axes. The incident wave is in the -$\hat{z}$ direction. The geometry was originally presented in six squares as shown in Fig. 5(a). Its FDTD surface mesh is simple to generate since the FDTD cells can be naturally fitted to the cube surfaces. The bistatic RCS of the cube has been obtained via an FDTD solver, and compared with the solution by the Numerical Electromagnetics Code (NEC) [5]. In the NEC solution, the cube surface was modeled by a wire frame mesh. It is seen from Fig. 5(b) that the FDTD solution agrees very well with the NEC results on both the E-plane ($\phi = 0^0$) and the H-plane ($\phi = 90^0$).

The second example is a perfectly conducting sphere. Its geometry was originally represented as a cluster of polygons as shown in Fig. 6(a). Its FDTD surface mesh was generated and subsequently input into an FDTD solver. The bistatic RCS of the sphere has also been obtained via the FDTD method and compared in Fig. 6(b) with the exact Mie theory. The FDTD solution agrees well with the exact solution on both the E-plane ($\phi = 0^0$) and the H-plane ($\phi = 90^0$), though the stair-case approximation is not smoothly body-fitted.

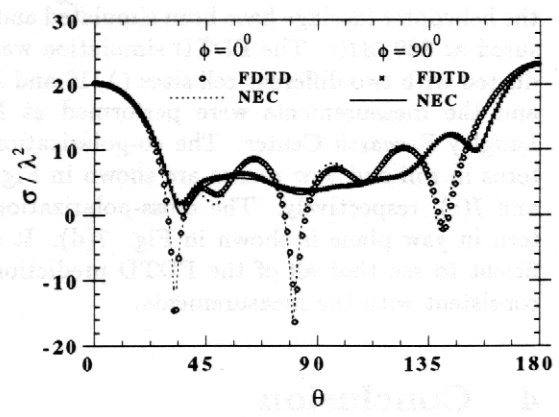
Figure 7 shows the mesh generation for a scale model helicopter. The helicopter geometry was modeled in polygons, and then discretized into a surface mesh, as illustrated in Fig. 7(a). The radiation patterns of a dipole mounted at the bottom center of the helicopter fuselage have been simulated and measured at 880 MHz. The FDTD simulation was conducted with two different cell sizes ($\lambda/16$ and $\lambda/20$), and the measurements were performed at NASA Langley Research Center. The co-polarization patterns in roll and yaw planes are shown in Fig. 7(b) and 7(c), respectively. The cross-polarization pattern in yaw plane is shown in Fig. 7(d). It is sufficient to see that all of the FDTD predictions are consistent with the measurements.

4 Conclusion

Based on a ray-tracing method, an efficient surface FDTD mesh generator has been developed. The mesh generator can be used for a complex object enclosed by conducting and thin-dielectric surfaces. The high efficiency of the algorithm allows the mesh generation to be performed on a personal computer.

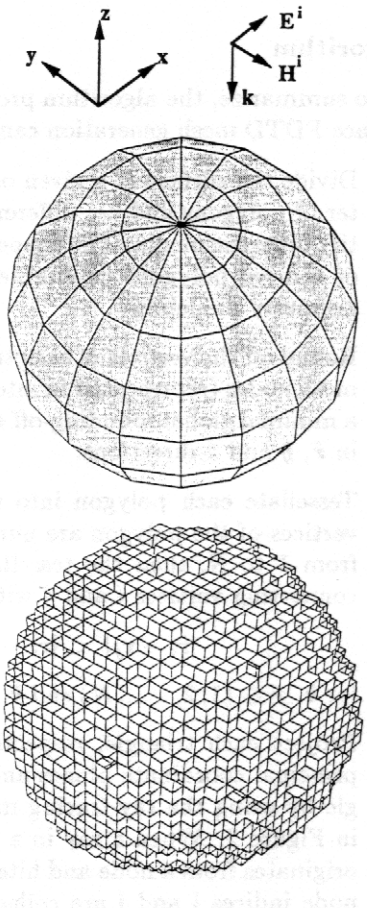


(a)

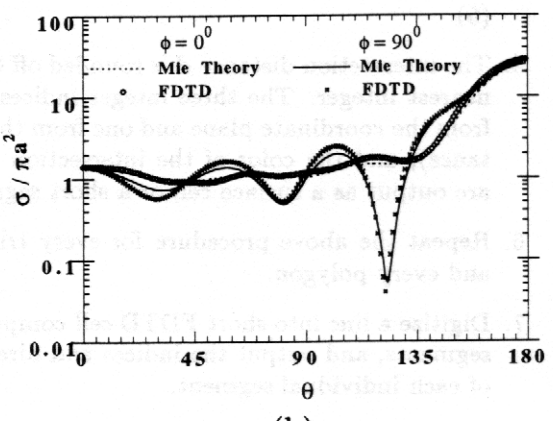


(b)

Figure 5: Mesh generation for a conducting cube and its bistatic RCS solution. (a) A cube modeled in six squares and its FDTD mesh; b) Bistatic RCS of the cube in the E-plane ($\phi = 0^0$) and the H-plane ($\phi = 90^0$).

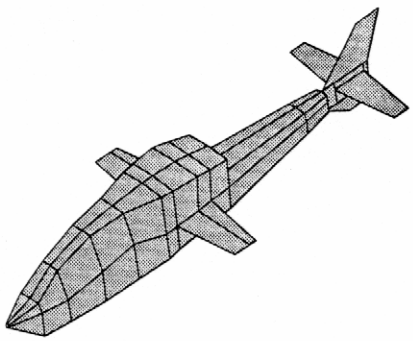


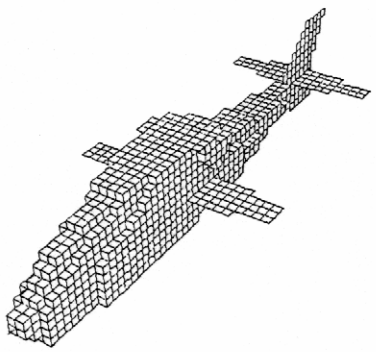
(a)



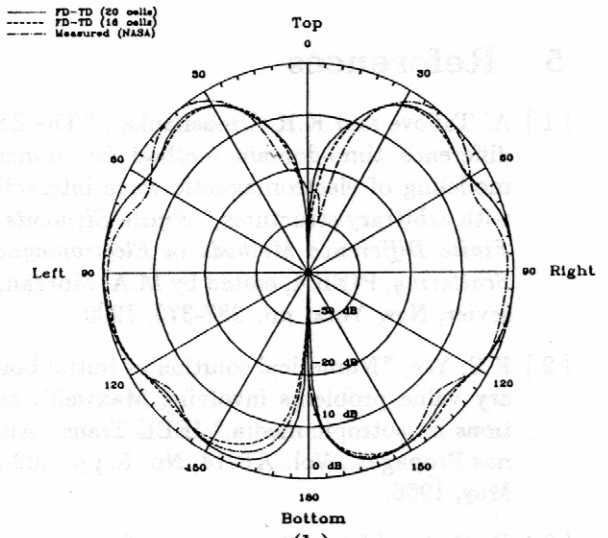
(b)

Figure 6: Mesh generation for a sphere and its bistatic RCS solution. (a) A sphere modeled in polygons and its subsequent FDTD mesh; b) Bistatic RCS of the sphere in the E-plane ($\phi = 0^0$) and the H-plane ($\phi = 90^0$).

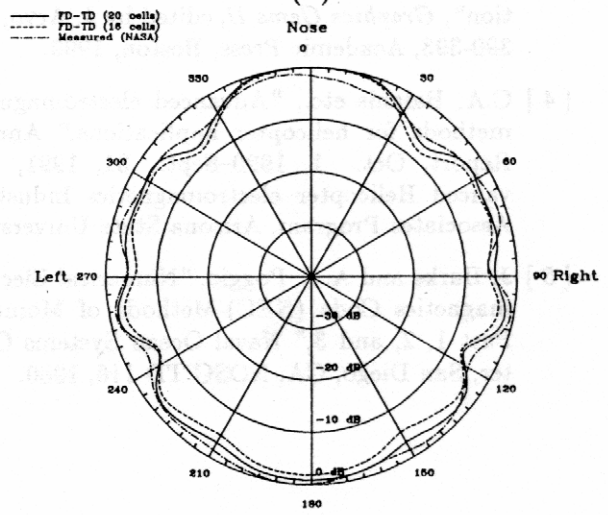




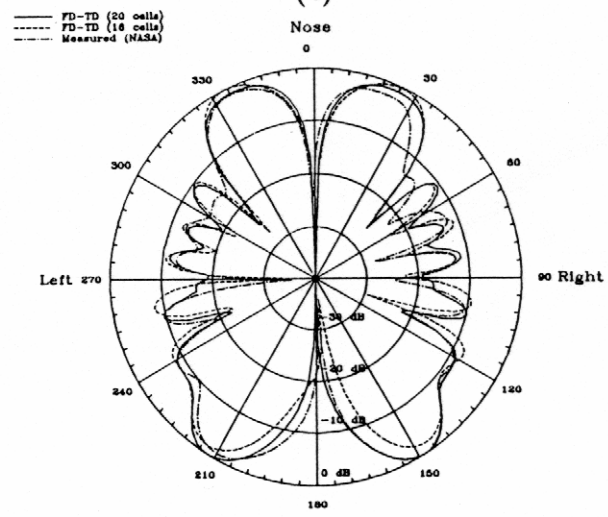
(a)



(b)



(c)



(d)

Figure 7: Mesh generation for a scale helicopter. a) The scale helicopter modeled in polygons and Its FDTD mesh; b) A dipole co-polarization radiation pattern in roll plane; c) The dipole co-polarization radiation pattern in yaw plane; d) The dipole cross-polarization radiation pattern in yaw plane.

5 References

[1] A. Taflove and K.R. Umashankar, "The finite-difference time-domain method for numerical modeling of electromagnetic wave interactions with arbitrary structures," *Finite Elements and Finite Difference Methods in Electromagnetics Scattering*, PIER 4, edited by M.A. Morgan, Elsevier, New York, pp. 287-373, 1990.

[2] K.S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," IEEE Trans., Antennas Propagat., Vol. AP-14, No. 5, pp. 302-307, May, 1966.

[3] D. Badouel,"An efficient ray-polygon intersection", *Graphics Gems II*, edited by J. Arvo, pp. 390-393, Academic Press, Boston, 1990.

[4] C.A. Balanis etc., "Advanced electromagnetic methods for helicopter applications," Annual Report, Oct. 1, 1990–Sept. 31, 1991, Advanced Helicopter electromagnetics Industrial Associates Program, Arizona State University.

[5] J. Burke and A. J. Poggio, "Numerical Electromagnetics Code (NEC)-Methods of Moments Part 1, 2, and 3," Naval Ocean Systems Center, San Diego, CA, NOSC TD 116, 1980.