

2

AD-A240 410



A Fast Algorithm for Plotting Antenna and Scattering Patterns in Three Dimensions

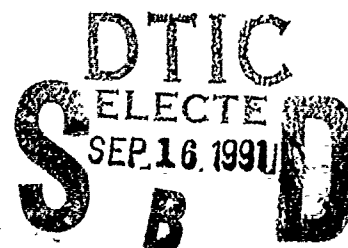
Prepared by

T. J. PETERS
Communications Systems Subdivision

31 August 1991

Prepared for

SPACE SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
Los Angeles Air Force Base
P. O. Box 92960
Los Angeles, CA 90009-2960



Engineering and Technology Group

91-10511



THE AEROSPACE CORPORATION
El Segundo, California



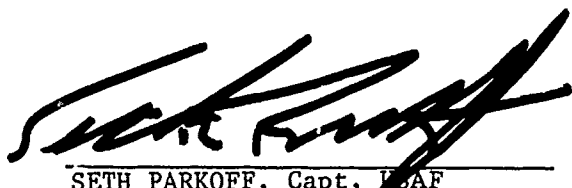
APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

91 9 12 123

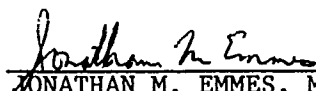
This report was submitted by The Aerospace Corporation, El Segundo, CA 90245-4691, under Contract No. F04701-88-C-0089 with the Space Systems Division, P. O. Box 92960, Los Angeles, CA 90009-2960. It was reviewed and approved for The Aerospace Corporation by J. M. Straus, Principal Director, Communications Systems Subdivision.

Seth Parkoff, Capt, USAF, was the project officer for the Mission-Oriented Investigation and Experimentation (MOIE) program. This report has been reviewed by the Public Affairs Office (PAS) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.



SETH PARKOFF, Capt, USAF
MOIE Project Officer
SSD/MHE



JONATHAN M. EMMES, Maj, USAF
MOIE Program Manager
PL/WCO OL-AH

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				
1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR-0091(6925-05)-6		5. MONITORING ORGANIZATION REPORT NUMBER(S) SSD-TR-91-28		
6a. NAME OF PERFORMING ORGANIZATION The Aerospace Corporation Communications Systems Subdivision		6b. OFFICE SYMBOL (If applicable)	7. NAME OF MONITORING ORGANIZATION Space Systems Division	
6c. ADDRESS (City, State, and ZIP Code) P. O. Box 92957 Los Angeles, CA 90009-2957		7b. ADDRESS (City, State, and ZIP Code) Los Angeles Air Force Base P. O. Box 92960 Los Angeles, CA 90009-2960		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F04701-88-C-0089	
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
				WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) A Fast Algorithm for Plotting Antenna and Scattering Patterns in Three Dimensions				
12. PERSONAL AUTHOR(S) Peters, T. J.				
13a. TYPE OF REPORT		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) 31 August 1991
15. PAGE COUNT 59				
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) An algorithm is presented for plotting antenna and scattering patterns in three dimensions on video displays or laser printers. The algorithm exploits the property of single-valued surfaces to allow the implicit removal of hidden lines with virtually no extra computation. This reduces the computation time significantly over that required by more general surface-representation methods. The algorithm is flexible enough to implement on most graphic systems. Simple language-independent pseudo-code is presented and tested for functions in rectangular, cylindrical, and spherical coordinates.				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL		22b. TELEPHONE (Include Area Code)		22c. OFFICE SYMBOL

Contents

I.	Introduction	7
II.	Coordinate Systems	9
III.	Scale, Rotation, and Projection	11
IV.	Algorithm Description	15
V.	Rectangular Coordinates	17
VI.	Cylindrical Coordinates	21
VII.	Spherical Coordinates	29
VIII.	Conclusion	35
	References	37
	Appendix: Fortran 77 Programs	39



Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Figures

1. Drawing flow pattern for rectangular coordinates	18
2. Power pattern of uniformly excited square aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$	19
3. Drawing flow pattern for cylindrical coordinates	22
4. Power pattern of uniformly excited square aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$ and plotted in dB, with $\nu = -40$ dB and $\eta = -25$ dB.	23
5. Magnitude of far-field electric field of a uniformly excited circular aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$ and plotted in dB, with $N_s = 14$, $\nu = -40$ dB and $\eta = -25$ dB	24
6. Circular waveguide and magnitude of surface current induced on inside wall for TM_{21} mode viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$	26
7. Elevation dependence drawing pattern for spherical coordinates.	30
8. Power pattern of uniformly excited circular aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$	31
9. Power pattern of uniformly excited circular aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$	32

Tables

1. Variable definitions9
2. Conversions to rectangular coordinates9

I. Introduction

The representation of antenna and scattering patterns in three dimensions provides a useful tool for analyzing power flow or field strength, especially when used in conjunction with field line contour plots (Ref. 1). Geometrically, these patterns represent surfaces. A comprehensive bibliography of surface-representation algorithms is given by Griffiths (Ref. 2). Most of these algorithms are geared towards representing the complex shape of a physical object. Specific algorithms for plotting mathematical functions of two variables in rectangular coordinates generally follow the method proposed by Wright (Ref. 3). Advanced line drawing algorithms suitable for cylindrical and spherical coordinates have been developed by Scott (Ref. 4). Unfortunately, all these methods require a significant amount of additional computation to be able to plot the surface with the hidden lines removed. The algorithm developed and presented in this paper avoids this extra computation by exploiting known properties of the surfaces being plotted.

In general, line drawing algorithms do not remove hidden lines, but rather, simply do not draw them. An alternative way to remove the hidden lines of a surface is to paint over the hidden part with the same color as the background. This is exactly the technique an artist would use to paint a landscape. The image is placed on the viewing surface from background to foreground and hidden lines are painted over. Of course, this requires the graphics system to be able to fill or erase a polygonal region. Therefore, the algorithm proposed in this paper is not suitable for representing surfaces by means of mechanical pen plotters. However, it is ideally suited for video displays and laser printers.

The algorithm is based on the following postulate. If a function $f(u, v)$, where u and v are two coordinates of an orthogonal system, generates a single-valued surface in the variables u and v , then, there exists a systematic, although not unique, ordered sequence in which to draw the surface from

back to front. This sequence is known, a priori, once the observation angles are specified. Therefore, no hidden line removal is necessary. Thus, plotting a function with the hidden lines removed takes the same amount of time as plotting without removing the hidden lines. A single-valued surface is defined as a surface in which there is a one to one correspondence between each pair of coordinates (u, v) and a point on the surface. Aperture distributions, antenna patterns, and scattering patterns are known to generate surfaces that are single-valued with respect to a particular coordinate system.

II. Coordinate Systems

The functions of interest in this study are plotted in rectangular, cylindrical, and spherical coordinates. The standard variables that describe these coordinate systems are defined by

$$r = \sqrt{x^2 + y^2} \quad (1)$$

$$R = \sqrt{r^2 + z^2} \quad (2)$$

$$\phi = \tan^{-1}(y/x) \quad (3)$$

$$\theta = \tan^{-1}(r/z). \quad (4)$$

In order to organize the algorithm inputs consistently, let $f(u, v)$ describe a function of two variables from one of these coordinate systems. Table 1 shows the convention adopted for the relationship between u and v and the standard variables. Since the final plot is placed on a two-dimensional surface, it is convenient to perform this step in rectangular coordinates. Therefore, all plotting points will be converted to rectangular coordinates prior to any drawing. The conversion conventions are shown in Table 2. Once the points on the surface are converted to rectangular coordinates, the graphical operations of scale, rotation, and projection may be applied.

Table 1. Variable Definitions

	rectangular	cylindrical		spherical
u	x	ϕ	ϕ	ϕ
v	y	r	z	θ

Table 2. Conversions to Rectangular Coordinates

	rectangular	cylindrical		spherical
x	u	$v \cos u$	$f(u, v) \cos u$	$f(u, v) \sin v \cos u$
y	v	$v \sin u$	$f(u, v) \sin u$	$f(u, v) \sin v \sin u$
z	$f(u, v)$	$f(u, v)$	v	$f(u, v) \cos v$

III. Scale, Rotation, and Projection

The surface defined by $f(u, v)$ is represented by the graphical transformations of scale, rotation and projection. Each graphical operation must be applied to individual coordinates of the surface. Let each point of the surface be defined by the vector w such that

$$w = \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (5)$$

The graphical operations may now be defined as matrices that operate on this vector.

In order to enhance some visual attributes, it may be desirable to scale each point before plotting. A scaling matrix S is defined as

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \quad (6)$$

where each component is some specified constant. Note that to preserve the linearity of the scale, the condition $S_x = S_y$ must be satisfied in cylindrical coordinates and $S_x = S_y = S_z$ must be satisfied in spherical coordinates. The coordinate points are then scaled by forming the matrix vector product Sw . Another type of scaling, which is quite commonly used in plotting antenna and scattering patterns, involves converting the function to decibels. This allows the viewer to observe more detail of the sidelobe behavior. Let f be normalized to the range $0 \leq f \leq 1$. Then define the zero reference in dB as ν . Next, define a plot floor level in dB as η , such that $\eta \geq \nu$. This plot floor is the level to which the function is set to for any value below the floor. This avoids a cluttered graph that results from too many low-level sidelobes. The function f can then be converted to a dB scale such that $0 \leq f_{dB} \leq 1$ by the nonlinear transformation

$$f_{dB} = \begin{cases} \frac{1}{|\nu|}(\eta + |\nu|) & f \leq 10^{(\eta/10)} \\ \frac{1}{|\nu|}(10 \log(f) + |\nu|) & f > 10^{(\eta/10)}. \end{cases} \quad (7)$$

The observer is assumed to be stationary, so the surface of the function must be rotated to the correct view. Viewing any finite three-dimensional object requires a minimum of one rotation axis. However, it is usually necessary to have rotation around two axes. These axes should be perpendicular to allow the widest range of viewing angles. The convention adopted in this study is to allow an azimuthal rotation around the z axis and an elevation rotation in the yz plane. This is convenient for plotting antenna patterns, aperture distributions and scattering patterns. The visual result is a graph that appears to spin in azimuth around the z axis and is tipped in elevation toward or away from the viewer.

It is important to distinguish between the observation angles and the rotation angles. The viewer-supplied elevation observation angle is defined as θ_0 , and the azimuth observation angle is defined as ϕ_0 . These angles will be restricted to the ranges $0 \leq \theta_0 \leq \pi$ and $0 \leq \phi_0 \leq 2\pi$, respectively. The use of these angles provides viewers a familiar frame of reference. In order to view the surface from these angles, it is necessary to define an azimuth rotation angle α and an elevation rotation angle β . These angles are dependent on the observation angles and the orientation of the viewer to the rectangular coordinate system. It is assumed that the observer will look in from the positive z axis onto the xy plane. The x axis increases from left to right and the y axis increases from bottom to top. Once this convention is established and the observation angles are specified, the rotation angles can be calculated as

$$\alpha = -\frac{\pi}{2} - \phi_0 \quad (8)$$

$$\beta = \theta_0. \quad (9)$$

The order of rotation is not commutative. The plots must first be spun in azimuth by the angle α and then tipped in elevation by the angle β . Reversing the order would allow the plot to tip from side to side, which presents an awkward picture. The azimuth rotation in the xy plane is represented by the

matrix vector product $R_\alpha \mathbf{w}$, where

$$R_\alpha = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (10)$$

and the elevation rotation in the yz plane by the product $R_\beta \mathbf{w}$, where

$$R_\beta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \beta & \sin \beta \\ 0 & -\sin \beta & \cos \beta \end{bmatrix}. \quad (11)$$

The complete transformation can be represented by the matrix T , defined by

$$T = R_\beta R_\alpha S. \quad (12)$$

If $\mathbf{w}'$ represents the transformed coordinates then

$$\mathbf{w}' = T\mathbf{w}. \quad (13)$$

Viewing a three-dimensional plot on a two-dimensional surface requires a projection of the three-dimensional coordinate points onto a two-dimensional viewing plane. Perspective projection implies that the further a point is away from the viewer, the smaller it appears. This type of projection is appropriate for images that evoke a strong depth cue such as a building or a landscape. However, mathematical functions do not require a strong depth cue since there is no physical object being represented. Therefore, it is sufficient to project each point along a parallel line until it intercepts the viewing plane. This type of projection is called parallel projection. Using parallel projection implies that only the x' and y' components of the vector $\mathbf{w}$ are needed to represent the surface. The z' coordinate represents the depth of each point and is not used.

These three graphical transformations are independent of the type of function being plotted. They represent the transformation of a point on the function surface to a point on the viewing surface. The order in which points are operated on by the transformation is determined by the sequencing algorithm discussed in the next section.

IV. Algorithm Description

The proposed algorithm is based on the generic "painter's algorithm." This concept means that the surface of the plot is rendered by building up the image from back to front. Since the function has two variables, the surface is most naturally described by a collection of quadrilaterals. Each quadrilateral is placed on the viewing surface and is painted with the background color of the surface. A line around the perimeter is then drawn. The key, of course, is to know the order in which to place the quadrilaterals on the viewing surface. A previous method, developed by the author (Ref. 5), used the transformed z coordinate of the centroid of each quadrilateral and sorted by depth. This method was general enough to plot any surface. However, the time required to compute the depth of each centroid and to sort the values led to a significant time delay. The new method proposed avoids this delay by drawing in a prescribed order based on the observation angles ϕ_0 and θ_0 .

The variables u and v are discretized such that $u = u_m$ for $m = 1 \dots N_u$ and $v = v_n$ for $n = 1 \dots N_v$. By convention, the values sequence from the minimum to the maximum values. The function $f(u, v)$ is then sampled at the discrete points $f_{m,n} = f(u_m, v_n)$. Each quadrilateral has a reference corner that has index values (m, n) . The other three corners are dependent on these indices and are given by $(m+1, n)$, $(m+1, n+1)$, and $(m, n+1)$. It should be noted that the definition of a quadrilateral is extended to allow any number of corners to have the same location. Therefore, a point, a line, and a triangle, can also be represented by a quadrilateral. The crux of the algorithm is to find a systematic method of sequencing through the indices in order to approximate a back-to-front ordering. The sequencing orders presented for the three coordinate systems analyzed were chosen based on ease of programming.

V. Rectangular Coordinates

Rectangular coordinates are the most common way of representing patterns in three dimensions. The coordinates x and y are replaced by ϕ and θ . Assume for simplicity that the coordinates are translated so that the origin is in the interior of the range. This does not affect the generality of the algorithm, but merely the presentation. Since the function is plotted on a rectangular base, it may be surmised that one corner of the surface will always be nearest to the observer and the opposite corner will be the farthest away. Because of the rotation conventions used, this is dependent only on the azimuth observation angle ϕ_0 and independent of the elevation observation angle θ_0 . Therefore, the first part of the algorithm determines which corner is nearest to the observer. Once the orientation is established, the quadrilaterals are simply drawn from the back corner to the front corner by rows that alternate in direction. Alternating the rows helps to move forward in a more uniform manner. Figure 1 shows the drawing directions as the quadrilaterals are placed on the viewing surface from back to front. Note that at the angles 0° , 90° , 180° , 270° and 360° there are two back corners equidistant from the viewer. In this case, it does not matter which corner is chosen as long as it is chosen consistently. Figure 2 shows the far-field power pattern of a uniformly excited square aperture plotted at observation angles $\phi_0 = 30^\circ$ and $\theta_0 = 60^\circ$. The plotting algorithm used is given in pseudo-code as follows:

```

M = minimum of ( $N_u, N_v$ )
if  $0 \leq \phi_0 < \pi/2$  then {find nearest corner}
     $i = 1, j = 1$ 
else if  $\pi/2 \leq \phi_0 < \pi$  then
     $i = -1, j = 1$ 
else if  $\pi \leq \phi_0 < 3\pi/2$  then
     $i = -1, j = -1$ 
else if  $3\pi/2 \leq \phi_0 \leq 2\pi$  then
     $i = 1, j = -1$ 
end if
loop from  $l = 1$  to  $M - 1$ 
    if  $0 \leq \phi_0 < \pi/2$  then {find nearest corner}

```

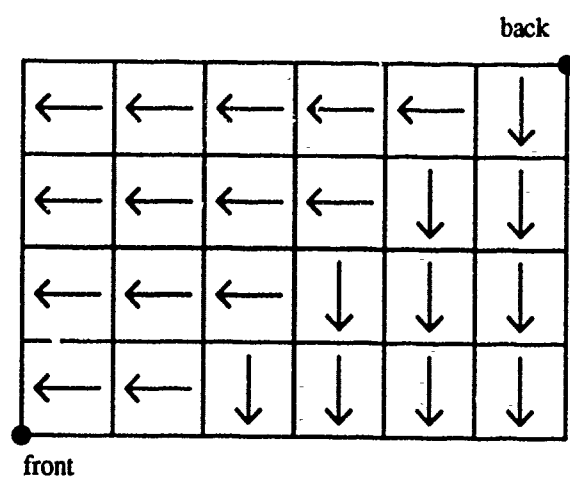


Fig. 1. Drawing flow pattern for rectangular coordinates

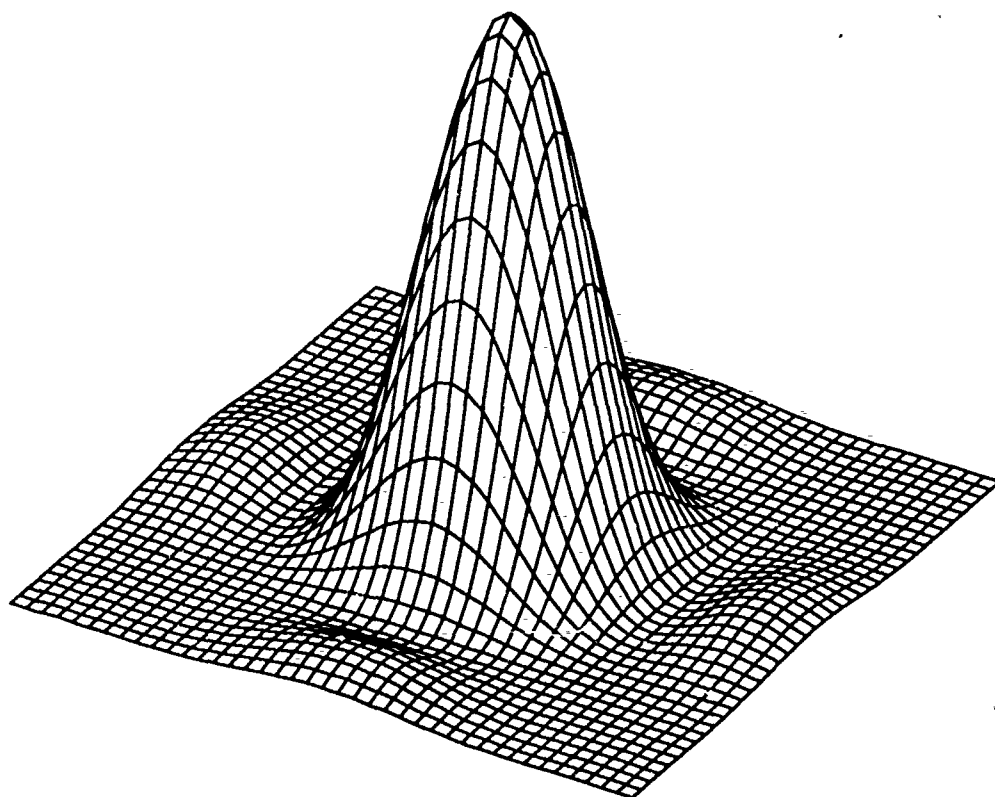


Fig. 2. Power pattern of uniformly excited square aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$

```

     $m_0 = l - 1, \quad n_0 = l$ 
     $m_1 = l, \quad n_1 = l$ 
  else if  $\pi/2 \leq \phi_0 < \pi$  then
     $m_0 = N_u - l + 1, \quad n_0 = l$ 
     $m_1 = N_u - l, \quad n_1 = l$ 
  else if  $\pi \leq \phi_0 < 3\pi/2$  then
     $m_0 = N_u - l + 1, \quad n_0 = N_v - l$ 
     $m_1 = N_u - l, \quad n_1 = N_v - l$ 
  else if  $3\pi/2 \leq \phi_0 \leq 2\pi$  then
     $m_0 = l - 1, \quad n_0 = N_v - l$ 
     $m_1 = l, \quad n_1 = N_v - l$ 
  end if
  loop from  $k = 1$  to  $N_u - l - 1$ 
     $m = m_0 + ik, \quad n = n_0$ 
    { get  $(u_m, v_n)$  and other 3 corners }
    { convert to rectangular coordinates - Table 2 }
    { scale, rotate, and project using Eqn. (13)}
    { fill quadrilateral, then draw perimeter }
  continue  $k$  loop
  loop from  $k = 1$  to  $N_v - l$ 
     $m = m_1, \quad n = n_1 + jk$ 
    { get  $(u_m, v_n)$  and other 3 corners.}
    { convert to rectangular coordinates - Table 2 }
    { scale, rotate, and project using Eqn. (13)}
    { fill quadrilateral, then draw perimeter }
  continue  $k$  loop
continue  $l$  loop

```

VI. Cylindrical Coordinates

Cylindrical coordinate functions of the form $z = f(\phi, r)$ and $r = f(\phi, z)$ are commonly encountered in antenna and scattering analysis. Far-field patterns can be plotted in the coordinates $f(\phi, r)$ by letting $r = \theta$. This type of plot is good for observing the finer details of the sidelobe structure. It is also a natural way to plot circular aperture distributions. Alternatively, functions of the form $r = f(\phi, z)$ are useful for observing near-field patterns or surface currents on structures such as a cylinder or a body of revolution.

Cylindrical functions of the form $z = f(\phi, r)$ are plotted on a circular base with a specified foreground angle ϕ_0 and a background angle $\phi_a = \phi_0 \pm \pi$, chosen such that $0 \leq \phi_a \leq 2\pi$. Therefore, the ϕ dependency of the quadrilaterals should be drawn starting with the background angle ϕ_a and proceeding toward the foreground observation angle ϕ_0 . The method employed is to find the index m of the u_m nearest to ϕ_a and then alternately increase and decrease the value to draw from back to front. For simplicity, assume that $u_{min} = 0$ and $u_{max} = 2\pi$. Since ϕ is periodic, the index m must be periodic with period $N_u - 1$. Drawing the radial dependence from back to front is dependent on the value of ϕ . Observation of the drawing flow pattern in Fig. 3 shows that the radial dependence should be drawn from r_{max} to r_{min} when the angle u_m is greater than 90° from ϕ_0 , and from r_{min} to r_{max} when the angle is less than 90° from ϕ_0 . Figure 4 shows the power pattern of a uniformly excited square aperture plotted in dB, with $\nu = -40$ dB and $\eta = -25$ dB. In some plotting situations it is desirable to remove an angular sector, to allow better viewing of the surface. This can be easily accomplished because of the way the ϕ values are alternated when plotting. Defining N_s as the number of segments to remove, the total number of ϕ values is just reduced by that amount. Figure 5 illustrates the use of angular cuts in a plot of the magnitude of the far-field electric field of a uniformly excited circular aperture. The following pseudo-code implements the algorithm:

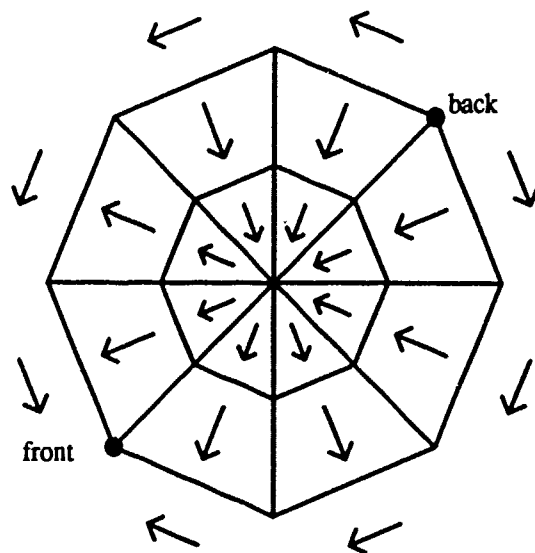


Fig. 3. Drawing flow pattern for cylindrical coordinates

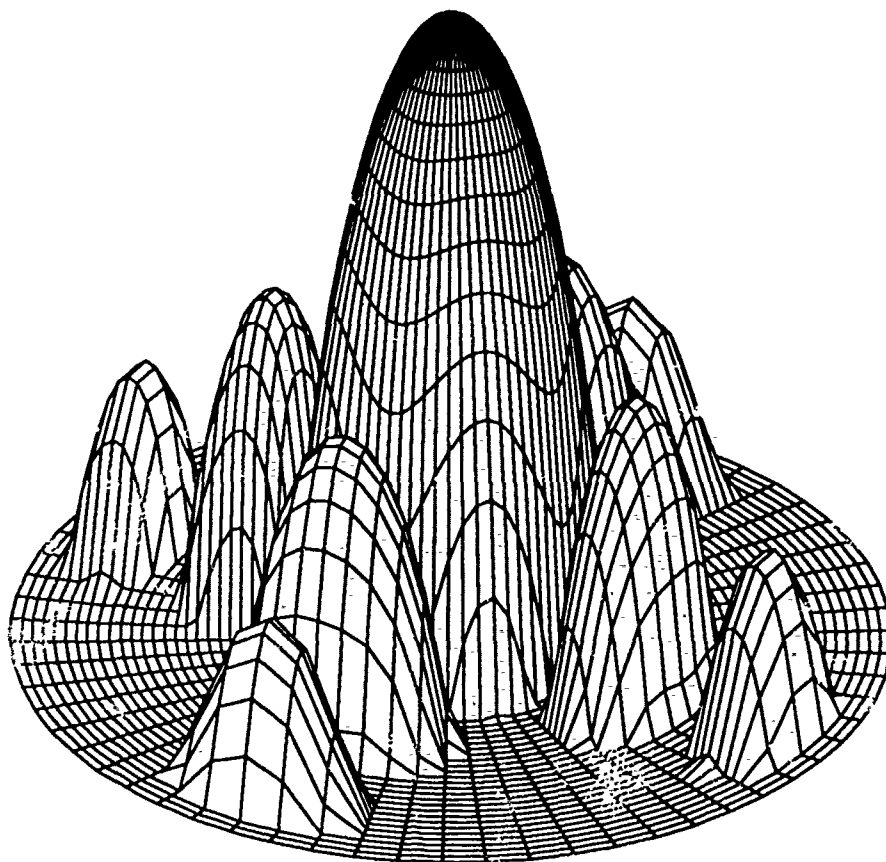


Fig. 4. Power pattern of uniformly excited square aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$ and plotted in dB, with $\nu = -40$ dB and $\eta = -25$ dB

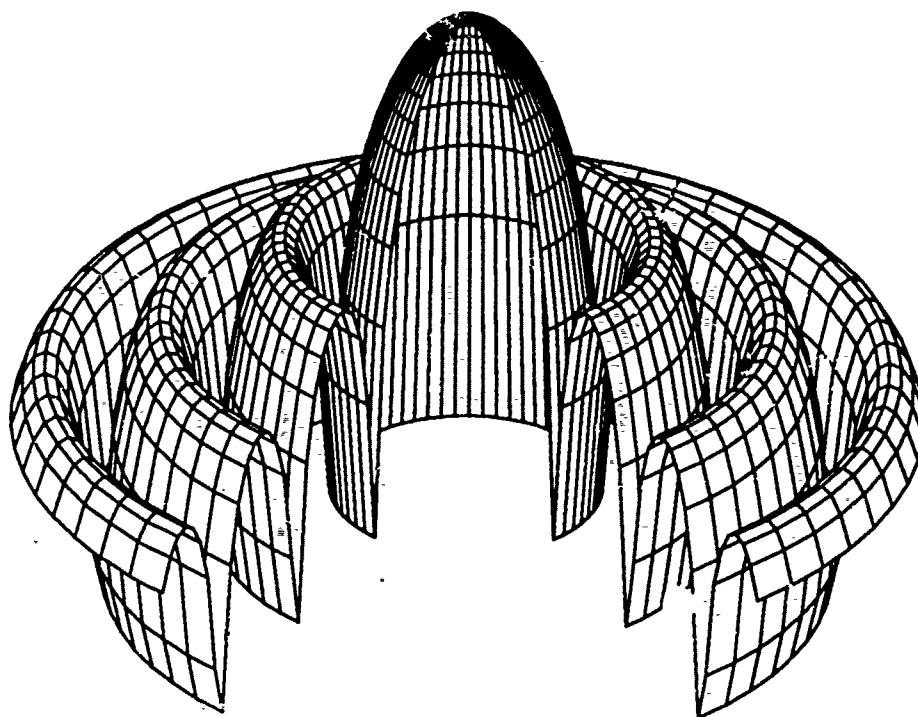


Fig. 5. Magnitude of far-field electric field of a uniformly excited circular aperture viewed from $\theta_0 = 50^\circ$ and $\phi_0 = 30^\circ$ and plotted in dB, with $N_r = 14$, $\nu = -40$ dB and $\eta = -25$ dB

```

loop from  $k = 1$  to  $N_u - 1$  {find index of angle  $\phi_a$ }
  if  $\phi_a \geq u_k$  and  $\phi_a \leq u_{k+1}$  then
     $m = k$ 
  end if
continue  $k$  loop
loop from  $l = 1$  to  $N_u - N_s - 1$ 
   $m = m - (-1)^l(l - 1)$ 
  if  $(m < 1)$  then {force index to be periodic}
     $m = m + N_u - 1$ 
  else if  $(m > N_u - 1)$  then
     $m = m - (N_u - 1)$ 
  end if
  if  $|u_m - \phi_0| < \pi/2$  or  $|u_m - \phi_0| > 3\pi/2$  then
     $n_0 = 0, \quad i = 1$  {draw from  $r_{min}$  to  $r_{max}$ }
  else
     $n_0 = N_v, \quad i = -1$  {draw from  $r_{max}$  to  $r_{min}$ }
  end if
  loop from  $k = 1$  to  $N_v - 1$ 
     $n = n_0 + ik$ 
    {get  $(u_m, v_n)$  and other 3 corners }
    {convert to rectangular coordinates - Table 2 }
    {scale, rotate, and project using Eqn. (13)}
    {fill quadrilateral, then draw perimeter }
  continue  $k$  loop
continue  $l$  loop

```

The algorithm for plotting functions of the form $r = f(\phi, z)$ uses the same procedure for handling the ϕ variation. The z variation is handled very easily by simply noting that if the observation angle θ_0 is less than or equal to 90° , then draw from z_{min} to z_{max} . Otherwise, if θ_0 is greater than 90° , then draw from z_{max} to z_{min} . Figure 6 shows a section of circular waveguide and the magnitude of the longitudinal surface current induced on the inner walls for the TM_{21} mode. The pseudo-code for the algorithm is given as follows:

```

loop from  $k = 1$  to  $N_u - 1$  {find index of angle  $\phi_a$ }
  if  $\phi_a \geq u_k$  and  $\phi_a \leq u_{k+1}$  then
     $m_a = k$ 
  end if
continue  $k$  loop
loop from  $k = 1$  to  $N_v - 1$ 
  if  $\theta_0 \leq \pi/2$  then
     $n = k$  {draw from  $z_{min}$  to  $z_{max}$ }
  else
     $n = N_v - k$  {draw from  $z_{max}$  to  $z_{min}$ }
  end if
   $m = m_a$ 

```

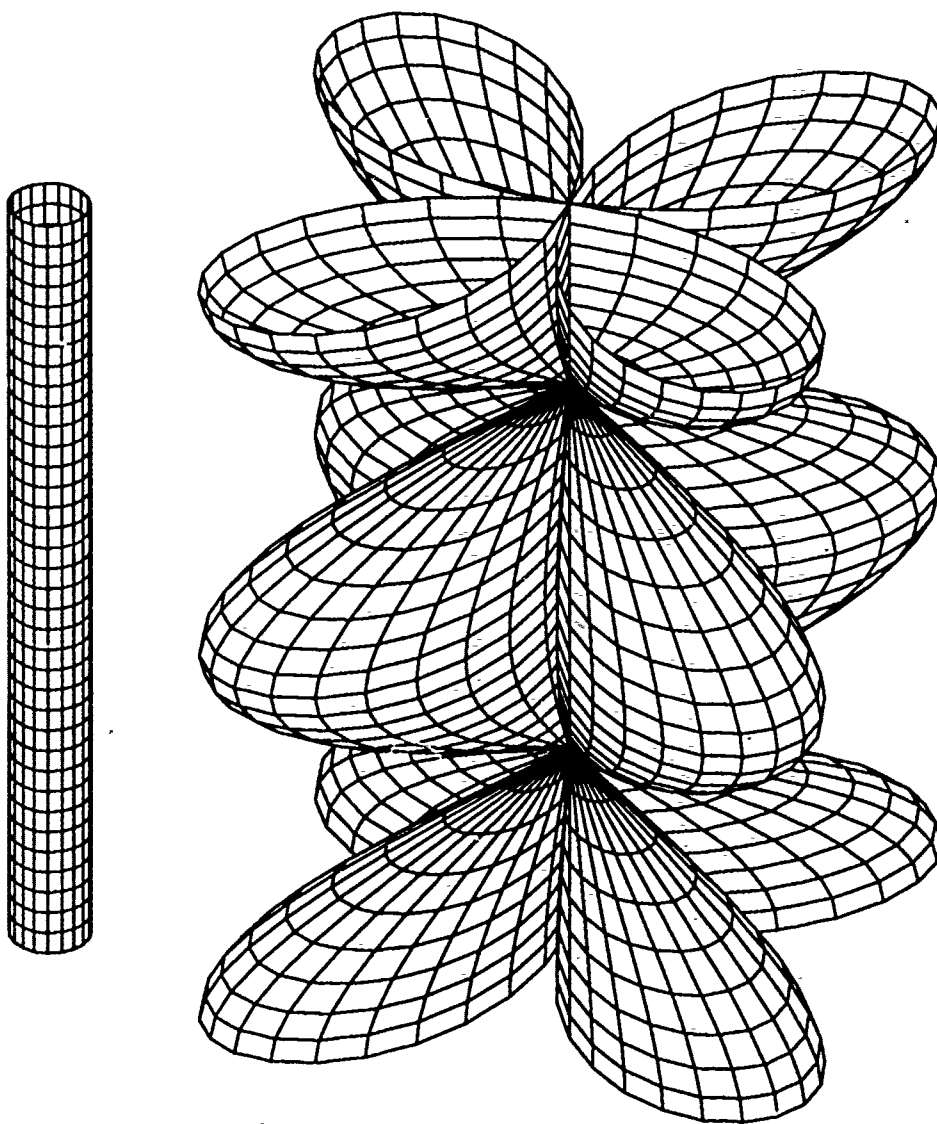


Fig. 6. Circular waveguide and magnitude of surface current induced on inside wall

```

loop from  $l = 1$  to  $N_u - N_s - 1$ 
   $m = m - (-1)^l(l - 1)$ 
  if  $(m < 1)$  then {force index to be periodic}
     $m = m + N_u - 1$ 
  else if  $(m > N_u - 1)$  then
     $m = m - (N_u - 1)$ 
  end if
  {get  $(u_m, v_n)$  and other 3 corners }
  {convert to rectangular coordinates - Table 2 }
  {scale, rotate, and project using Eqn. (13)}
  {fill quadrilateral, then draw perimeter }
continue  $l$  loop
continue  $k$  loop

```

VII. Spherical Coordinates

The flow of power in far-field antenna and scattering patterns is best described by using plots in spherical coordinates. Establishing the front and back orientation requires knowledge of both observation angles. The azimuth background angle ϕ_a is defined the same as in cylindrical coordinates. Therefore, the ϕ variation is drawn in exactly the same way as for cylindrical plots. The elevation background angle is defined as $\theta_a = \pi - \theta_0$. As shown in Fig. 7, the elevation dependence is drawn in opposite directions away from θ_a when u_m is greater than 90° from ϕ_0 . When u_m is less than 90° from ϕ_0 , the elevation dependence is drawn from the minimum and maximum limits to θ_0 . Figure 8 shows a power pattern of a circular aperture plotted in dB with $\nu = \eta = -60$ dB. As with cylindrical coordinates, it is sometimes convenient to cut away an angular piece of the plot so that the detail of the sidelobe pattern may be observed. However, since the lobes of spherical plots are generally closed surfaces, it looks better if the angular cut is filled with the background color. This is illustrated in Fig. 9. Filling this cut requires making a polygon using all the θ values at a constant ϕ . The following pseudo-code implements the algorithm:

```

loop from  $k = 1$  to  $N_u - 1$  {find index of angles  $\phi_0, \phi_a$ }
  if  $\phi_0 \geq u_k$  and  $\phi_0 \leq u_{k+1}$  then
     $m_0 = k$ 
  end if
  if  $\phi_a \geq u_k$  and  $\phi_a \leq u_{k+1}$  then
     $m_a = k$ 
  end if
continue  $k$  loop
 $n_0 = N_v$     $n_a = N_v$ 
loop from  $k = 1$  to  $N_v - 1$  {find index of angles  $\theta_0, \theta_a$ }
  if  $\theta_0 \geq v_k$  and  $\theta_0 \leq v_{k+1}$  then
     $n_0 = k$ 
  end if
  if  $\theta_a \geq v_k$  and  $\theta_a \leq v_{k+1}$  then
     $n_a = k$ 
  end if
continue  $k$  loop
 $m = m_a$ 

```

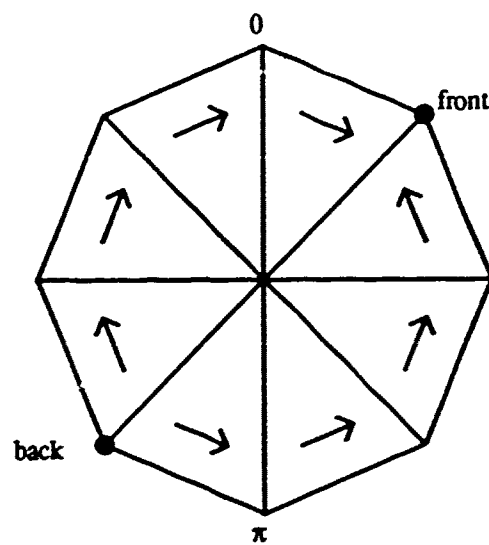


Fig. 7. Elevation dependence drawing pattern for spherical coordinates

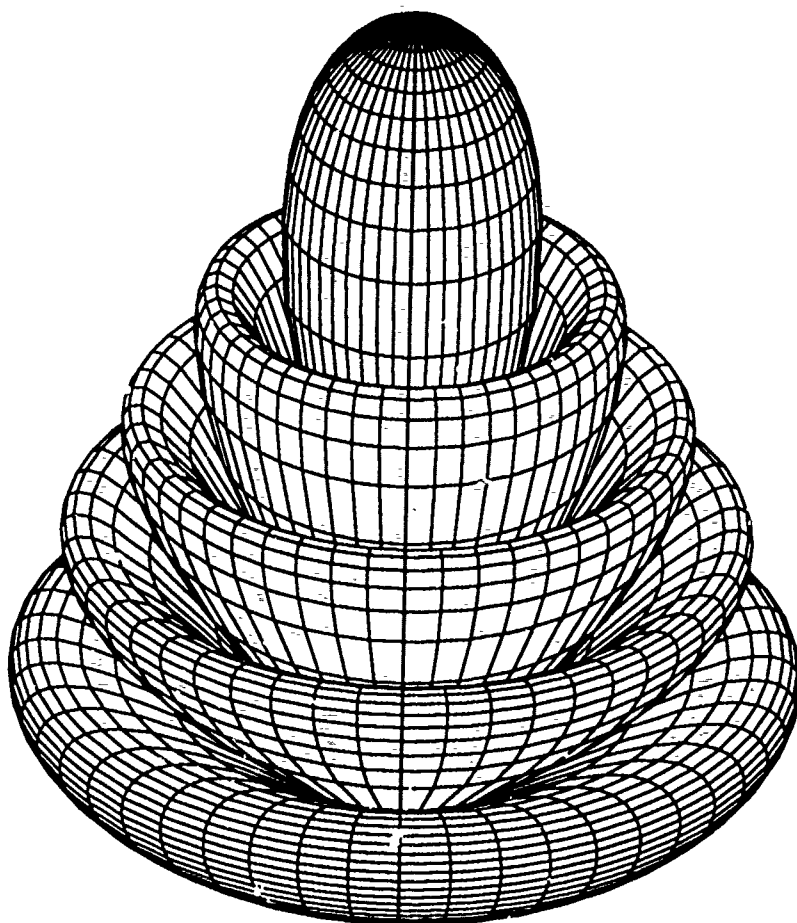


Fig. 8. Power pattern of uniformly excited circular aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$

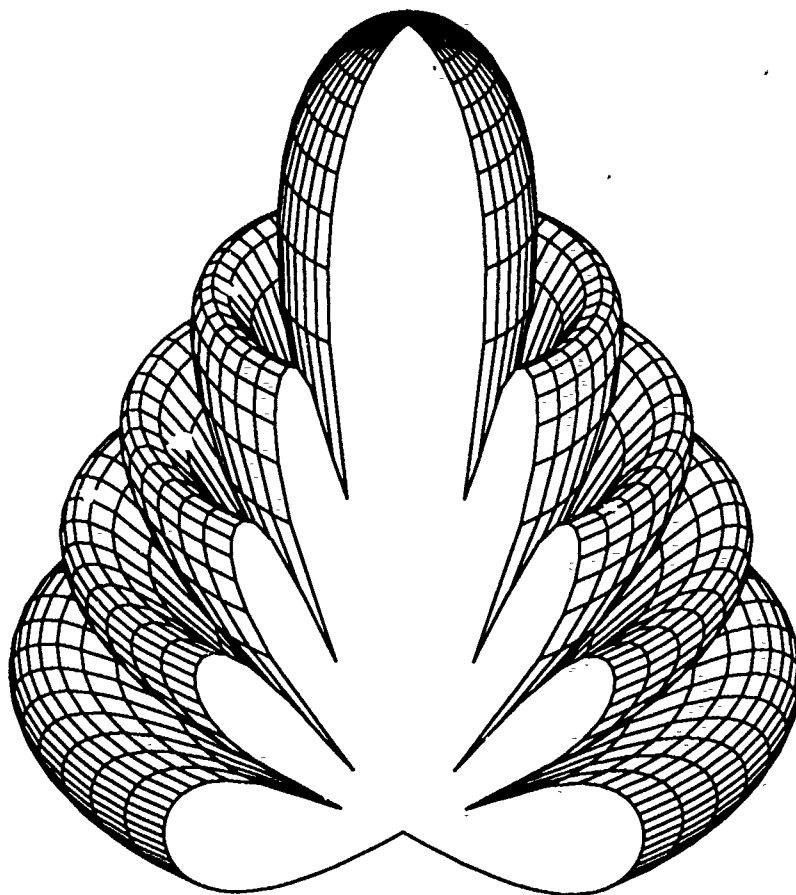


Fig. 9. Power pattern of uniformly excited circular aperture viewed from $\theta_0 = 60^\circ$ and $\phi_0 = 30^\circ$

```

loop from  $l = 1$  to  $N_u - N_s - 1$ 
   $m = m - (-1)^l(l - 1)$ 
  if  $(m < 1)$  then {force index to be periodic}
     $m = m + N_u - 1$ 
  else if  $(m > N_u - 1)$  then
     $m = m - (N_u - 1)$ 
  end if
  if  $|u_m - \phi_0| < \pi/2$  or  $|u_m - \phi_0| > 3\pi/2$  then
     $i = N_v, j = 0, L_1 = N_v - n_0, L_2 = n_0 - 1$ 
  else
     $i = M_a, j = n_a - 1, L_1 = n_a - 1, L_2 = N_v - n_a$ 
  end if
  loop from  $k = 1$  to  $L_1$ 
     $n = i - k$ 
    { get  $(u_m, v_n)$  and other 3 corners }
    { convert to rectangular coordinates - Table 2 }
    { scale, rotate, and project using Eqn. (13)}
    { fill quadrilateral, then draw perimeter }
  continue  $k$  loop
  loop from  $k = 1$  to  $L_2$ 
     $n = j + k$ 
    { get  $(u_m, v_n)$  and other 3 corners }
    { convert to rectangular coordinates - Table 2 }
    { scale, rotate, and project using Eqn. (13)}
    { fill quadrilateral, then draw perimeter }
  continue  $k$  loop
  if  $l > N_u - N_s - 3$  and  $N_s > 0$  then
    if  $u_m \geq \phi_0$  then
      if  $(u_m \geq \phi_0$  and  $u_m \leq \phi_a)$  then
         $p = m$ 
      else
         $p = m + 1$ 
      end if
    else
      if  $(u_m \geq \phi_a$  and  $u_m \leq \phi_0)$  then
         $p = m + 1$ 
      else
         $p = m$ 
      end if
    loop from  $q = 1$  to  $N_v$ 
      { form polygon with vertices  $(u_p, v_q)$  }
    continue  $q$  loop
    { fill polygon, then draw perimeter }
  end if
continue  $l$  loop

```

VIII. Conclusion

An algorithm has been presented that allows the rapid plotting of antenna and scattering patterns in three dimensions. Because of the special properties of the types of functions considered, the plotting speed is essentially the same as if no hidden lines were removed. Any graphics system that allows a polygon fill operation may implement the algorithm. Although not suitable for mechanical pen plotters, the algorithm is ideal for video displays and laser printers. The specific variations in rectangular, cylindrical, and spherical coordinates have been developed and tested yielding excellent results.

References

1. K. W. Kark and R. Dill, "A General Theory on the Graphical Representation of Antenna-Radiation Fields," IEEE Trans. Antennas Propagat., vol. AP-38, no. 2, pp.160-165, Feb. 1990.
2. J. G. Griffiths, "A Bibliography of Hidden-Line and Hidden Surface Algorithms," Computer Aided Design, 10(3), May 1978, pp. 203-206.
3. T. J. Wright, "A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables," IEEE Trans. Comput., vol. C-19, pp.28-33., January 1973.
4. W. R. Scott, Jr., "A General Program for Plotting Three-Dimensional Antenna Patterns," in 1987 IEEE Antennas Propagat. Soc. Symp. Dig., vol. 1, June 1988, pp. 330-333.
5. T. J. Peters, "Computation of the Scattering by Planar and Non-Planar Plates Using a Conjugate Gradient FFT Method," Ph.D. dissertation, Radiation Laboratory, The University of Michigan, 1988, pp. 206-221.

Appendix: Fortran 77 Programs

This appendix contains the Fortran 77 programs RECT3D, CYLA3D, CYLR3D, and SPHR3D as well as all associated subroutines. The required inputs to each program are listed in the comments found in the source code. These programs were used to generate all the results presented in this report. The output of each program is a PostScript file. Information on programming in the PostScript language is available from most bookstores.

```

1      PROGRAM RECT3D
2 C    *****
3 C    * THIS PROGRAM PRODUCES A POSTSCRIPT FILE REPRESENTING A THREE *
4 C    * DIMENSIONAL PLOT OF A FUNCTION IN RECTANGULAR COORDINATES. *
5 C    *****
6 C    * TIMOTHY J. PETERS                                LAST UPDATED *
7 C    * THE AEROSPACE CORPORATION                        3/1/91      *
8 C    * 2350 EAST EL SEGUNDO BOULEVARD.                  *
9 C    * EL SEGUNDO, CA 90245                             *
10 C   *****
11 C   * INPUTS:                                           *
12 C   *                                                 *
13 C   * NU   - NUMBER OF U POINTS.                       *
14 C   * NV   - NUMBER OF V POINTS.                       *
15 C   * AX   - X DIRECTION SCALE FACTOR.                 *
16 C   * AY   - Y DIRECTION SCALE FACTOR.                 *
17 C   * AZ   - Z DIRECTION SCALE FACTOR.                 *
18 C   * PO   - PHI OBSERVATION ANGLE IN RANGE 0<=PO<=2PI. *
19 C   * TO   - THETA OBSERVATION ANGLE IN RANGE 0<=PO<=PI. *
20 C   * IC   - IF IC=1 THEN CONVERT THE FUNCTION VALUES TO DB. *
21 C   *      NOTE THAT IF IC=1 THEN F MUST BE IN THE RANGE 0<=F<=1. *
22 C   * ETA  - PLOT FLOOR IN DB.                         *
23 C   * NUU  - EFFECTIVE ZERO IN DB.                     *
24 C   * U(MAX) - U COORDINATE ARRAY.                     *
25 C   * V(MAX) - V COORDINATE ARRAY.                     *
26 C   * F(MAX,MAX) - FUNCTION VALUE MATRIX.             *
27 C   *                                                 *
28 C   * OUTPUT:                                           *
29 C   *                                                 *
30 C   * RECT.PS - POSTSCRIPT FILE REPRESENTING THE PLOT. *
31 C   *                                                 *
32 C   *****
33     PARAMETER (MAX=100)
34     REAL*4 U(MAX),V(MAX),F(MAX,MAX),NUU
35     OPEN(UNIT=2,FILE='RECT.PS')
36     REWIND(2)
37     RAD=.17453293E-01
38     PI=.3141593E+01
39 C   *****
40 C   * READ THE INPUTS. *
41 C   *****
42     OPEN(UNIT=1,FILE='DATA')
43     READ(1,*) NU,NV,AX,AY,AZ,PO,TO,IC,ETA,NUU
44     READ(1,*) (U(M),M=1,NU)
45     READ(1,*) (V(N),N=1,NV)
46     READ(1,*) ((F(M,N),M=1,NU),N=1,NV)
47 C   *****
48 C   * DEFINE SOME MACROS IN POSTSCRIPT. *
49 C   *****

```

```

50      WRITE(2,100) 'initgraphics erasepage letter'
51      WRITE(2,100) '/m {moveto} def /l {lineto} def /s {stroke} def'
52      WRITE(2,100) '/sg {setgray} def /c {closepath 1 sg fill s} def'
53      WRITE(2,100) '/w {closepath 0 sg s} def /t {translate} def'
54 C    *****
55 C    * SET THE LINE CHARACTERISTICS. *
56 C    *****
57      WRITE(2,100) '0.5 setlinewidth 1 setlinecap 1 setlinejoin'
58 C    *****
59 C    * TRANSLATE THE ORIGIN TO THE GEOMETRIC CENTER OF THE PAPER. *
60 C    *****
61      XOFFSET=306.0
62      YOFFSET=396.0
63      WRITE(2,101) XOFFSET,YOFFSET,' t'
64 C    *****
65 C    * CONVERT THE OBSERVATION ANGLES TO RADIAN. *
66 C    *****
67      POBS=RAD*PO
68      TOBS=RAD*TO
69 C    *****
70 C    * COMPUTE THE ROTATION ANGLES WHICH YIELD THE DESIRED OBSERVATION *
71 C    * ANGLES. *
72 C    *****
73      RP=-POBS-PI/2.0
74      RT=TOBS
75      CP=COS(RP)
76      SP=SIN(RP)
77      CT=COS(RT)
78      ST=SIN(RT)
79 C    *****
80 C    * IF REQUESTED CONVERT THE DATA TO DB SCALE. *
81 C    *****
82      IF (IC .EQ. 1) THEN
83          TR=10.0**(0.1*ETA)
84          DO 1 M=1,NU
85              DO 2 N=1,NV
86                  IF(F(M,N) .LE. TR) THEN
87                      F(M,N)=(ETA+ABS(NUU))/ABS(NUU)
88                  ELSE
89                      F(M,N)=(10.0*ALOG10(F(M,N))+ABS(NUU))/ABS(NUU)
90                  END IF
91      2      CONTINUE
92      1      CONTINUE
93      ELSE
94      END IF
95 C    *****
96 C    * SET SOME CONSTANTS. *
97 C    *****
98      IF ((POBS .GE. 0.0) .AND. (POBS .LT. PI/2.0)) THEN

```

```

99      IS1=1
100     IS2=1
101     ELSE IF ((POBS .GE. PI/2.0) .AND. (POBS .LT. PI)) THEN
102         IS1=-1
103         IS2=1
104     ELSE IF ((POBS .GE. PI) .AND. (POBS .LT. 3.0*PI/2.0)) THEN
105         IS1=-1
106         IS2=-1
107     ELSE IF ((POBS .GE. 3.0*PI/2.0) .AND. (POBS .LE. 2.0*PI)) THEN
108         IS1=1
109         IS2=-1
110     ELSE
111     END IF
112 C *****
113 C * BEGIN SEQUENCE. *
114 C *****
115     DO 3 L=1,NV-1
116         IF ((POBS .GE. 0.0) .AND. (POBS .LT. PI/2.0)) THEN
117             MO=L-1
118             NO=L
119             M1=L
120             N1=L
121         ELSE IF ((POBS .GE. PI/2.0) .AND. (POBS .LT. PI)) THEN
122             MO=NU-L+1
123             NO=L
124             M1=NU-L
125             N1=L
126         ELSE IF ((POBS .GE. PI) .AND. (POBS .LT. 3.0*PI/2.0)) THEN
127             MO=NU-L+1
128             NO=NV-L
129             M1=NU-L
130             N1=NV-L
131         ELSE IF ((POBS .GE. 3.0*PI/2.0) .AND. (POBS .LE. 2.0*PI)) THEN
132             MO=L-1
133             NO=NV-L
134             M1=L
135             N1=NV-L
136         ELSE
137         END IF
138 C *****
139 C * LOOP THROUGH THE U VALUES WITH V CONSTANT. *
140 C *****
141     DO 4 K=1,NU-L
142         M=MO+IS1*K
143         N=NO
144 C *****
145 C * COMPUTE THE 4 VERTICES OF THE QUADRILATERAL. *
146 C *****
147     IS1=U(M)

```

```

148      YS1=V(N)
149      ZS1=F(M,N)
150      XS2=U(M+1)
151      YS2=V(N)
152      ZS2=F(M+1,N)
153      XS3=U(M+1)
154      YS3=V(N+1)
155      ZS3=F(M+1,N+1)
156      XS4=U(M)
157      YS4=V(N+1)
158      ZS4=F(M,N+1)
159 C      *****
160 C      * ROTATE THE 4 VERTICES OF THE QUADRILATERAL. *
161 C      *****
162      CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,RP,RT)
163      CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,RP,RT)
164      CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,RP,RT)
165      CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,RP,RT)
166 C      *****
167 C      * FILL THE QUADRILATERAL. *
168 C      *****
169      WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
170 C      *****
171 C      * DRAW PERIMETER OF THE QUADRILATERAL. *
172 C      *****
173      WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
174 4    CONTINUE
175 C      *****
176 C      * LOOP THROUGH THE V VALUES WITH U CONSTANT. *
177 C      *****
178      DO 5 K=1,NV-L-1
179          M=M1
180          N=N1+IS2*K
181 C      *****
182 C      * COMPUTE THE 4 VERTICES OF THE QUADRILATERAL. *
183 C      *****
184      XS1=U(M)
185      YS1=V(N)
186      ZS1=F(M,N)
187      XS2=U(M+1)
188      YS2=V(N)
189      ZS2=F(M+1,N)
190      XS3=U(M+1)
191      YS3=V(N+1)
192      ZS3=F(M+1,N+1)
193      XS4=U(M)
194      YS4=V(N+1)
195      ZS4=F(M,N+1)
196 C      *****

```

```

187 C      * ROTATE THE 4 VERTICES OF THE QUADRILATERAL.      *
188 C      *****
189          CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,RP,RT)
190          CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,RP,RT)
191          CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,RP,RT)
192          CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,RP,RT)
193 C      *****
194 C      * FILL THE QUADRILATERAL.      *
195 C      *****
196          WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
197 C      *****
198 C      * DRAW PERIMETER OF THE QUADRILATERAL.      *
199 C      *****
200          WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
201
202 5      CONTINUE
203 3      CONTINUE
204 C      *****
205 C      * SHOW THE PAGE.      *
206 C      *****
207          WRITE(2,100) 'showpage'
208 C      *****
209 C      * FORMATS.      *
210 C      *****
211 100     FORMAT(A72)
212 101     FORMAT(F7.2,1X,F7.2,A58)
213 200     FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
214          &F7.2,' l ',F7.2,1X,F7.2,' l c')
215 201     FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
216          &F7.2,' l ',F7.2,1X,F7.2,' l w')
217
218     END
219 C
220
221     SUBROUTINE ROTATE(XA,YA,ZA,X,Y,AX,AY,AZ,RP,RT)
222     X=AX*COS(RP)*XA-AY*SIN(RP)*YA
223     Y=COS(RT)*(AX*SIN(RP)*XA+AY*COS(RP)*YA)+AZ*SIN(RT)*ZA
224     RETURN
225
226     END

```

```

1      PROGRAM CYLA3D
2 C    *****
3 C    * THIS PROGRAM PRODUCES A POSTSCRIPT FILE REPRESENTING A THREE *
4 C    * DIMENSIONAL PLOT OF A FUNCTION IN CYLINDRICAL COORDINATES. *
5 C    *****
6 C    * TIMOTHY J. PETERS                                     LAST UPDATED *
7 C    * THE AEROSPACE CORPORATION                             3/1/91      *
8 C    * 2350 EAST EL SEGUNDO BOULEVARD.                      *
9 C    * EL SEGUNDO, CA 90245                                  *
10 C   *****
11 C   * INPUTS:                                                *
12 C   *                                                        *
13 C   * NU - NUMBER OF U POINTS.                               *
14 C   * NV - NUMBER OF V POINTS.                               *
15 C   * AX - X DIRECTION SCALE FACTOR.                         *
16 C   * AY - Y DIRECTION SCALE FACTOR.                         *
17 C   * AZ - Z DIRECTION SCALE FACTOR.                         *
18 C   * PO - PHI OBSERVATION ANGLE IN RANGE  $0 \leq PO \leq 2\pi$ . *
19 C   * TO - THETA OBSERVATION ANGLE IN RANGE  $0 \leq PO \leq \pi$ . *
20 C   * IC - IF IC=1 THEN CONVERT THE FUNCTION VALUES TO DB. *
21 C   *      NOTE THAT IF IC=1 THEN F MUST BE IN THE RANGE  $0 \leq F \leq 1$ . *
22 C   * ETA - PLOT FLOOR IN DB.                                *
23 C   * NUU - EFFECTIVE ZERO IN DB.                             *
24 C   * NS - NUMBER OF SEGMENTS TO REMOVE.                     *
25 C   * U(MAX) - U COORDINATE ARRAY.                           *
26 C   * V(MAX) - V COORDINATE ARRAY.                           *
27 C   * F(MAX,MAX) - FUNCTION VALUE MATRIX.                   *
28 C   *                                                        *
29 C   * OUTPUT:                                                *
30 C   *                                                        *
31 C   * CYLA.PS - POSTSCRIPT FILE REPRESENTING THE PLOT.      *
32 C   *                                                        *
33 C   *****
34     PARAMETER (MAX=100)
35     REAL*4 U(MAX),V(MAX),F(MAX,MAX),NUU
36     OPEN(UNIT=2,FILE='CYLA.PS')
37     REWIND(2)
38     RAD=.17453293E-01
39     PI=.3141593E+01
40 C   *****
41 C   * READ THE INPUTS.                                        *
42 C   *****
43     OPEN(UNIT=1,FILE='DATA')
44     READ(1,*) NU,NV,AX,AY,AZ,PO,TO,IC,ETA,NUU,NS
45     READ(1,*) (U(M),M=1,NU)
46     READ(1,*) (V(N),N=1,NV)
47     READ(1,*) ((F(M,N),M=1,NU),N=1,NV)
48 C   *****
49 C   * DEFINE SOME MACROS IN POSTSCRIPT.                      *

```

```

50 C *****
51 WRITE(2,100) 'initgraphics erasepage letter'
52 WRITE(2,100) '/m {moveto} def /l {lineto} def /s {stroke} def'
53 WRITE(2,100) '/sg {setgray} def /c {closepath 1 sg fill s} def'
54 WRITE(2,100) '/w {closepath 0 sg s} def /t {translate} def'
55 C *****
56 C * .SET THE LINE CHARACTERISTICS. *
57 C *****
58 WRITE(2,100) '0.5 setlinewidth 1 setlinecap 1 setlinejoin'
59 C *****
60 C * TRANSLATE THE ORIGIN TO THE GEOMETRIC CENTER OF THE PAGE. *
61 C *****
62 XOFFSET=306.0
63 YOFFSET=396.0
64 WRITE(2,101) XOFFSET,YOFFSET,' t'
65 C *****
66 C * CONVERT THE OBSERVATION ANGLES TO RADIANS. *
67 C *****
68 POBS=RAD*P0
69 TOBS=RAD*TO
70 C *****
71 C * COMPUTE THE ROTATION ANGLES WHICH YIELD THE DESIRED OBSERVATION *
72 C * ANGLES. *
73 C *****
74 RP=-POBS-PI/2.0
75 RT=TOBS
76 CP=COS(RP)
77 SP=SIN(RP)
78 CT=COS(RT)
79 ST=SIN(RT)
80 C *****
81 C * IF REQUESTED CONVERT THE DATA TO DB SCALE. *
82 C *****
83 IF (IC .EQ. 1) THEN
84   TR=10.0**(0.1*ETA)
85   DO 1 M=1,NU
86     DO 2 N=1,NV
87       IF(F(M,N) .LE. TR) THEN
88         F(M,N)=(ETA+ABS(NUU))/ABS(NUU)
89       ELSE
90         F(M,N)=(10.0*ALOG10(F(M,N))+ABS(NUU))/ABS(NUU)
91       END IF
92   2 CONTINUE
93   1 CONTINUE
94   ELSE
95   END IF
96 C *****
97 C * DETERMINE THE VALUE WHICH IS JUST BELOW P0+180 DEGREES. *
98 C *****

```

```

99      IF (POBS .LT. PI) THEN
100      PA=POBS+PI
101      ELSE
102      PA=POBS-PI
103      END IF
104      DO 3 M=1,NU-1
105      IF ((PA .GE. U(M)) .AND. (PA .LT. U(M+1))) THEN
106      IREF=M
107      ELSE
108      END IF
109 3     CONTINUE
110 C     *****
111 C     * SEQUENCE THROUGH THE INDICES. *
112 C     *****
113      M=IREF
114      DO 4 L=1,NU-MS-1
115      IS=(-1)**L
116      M=M-IS*(L-1)
117      IF (M .LT. 1) THEN
118      M=M+NU-1
119      ELSE IF (M .GT. NU-1) THEN
120      M=M-(NU-1)
121      ELSE
122      END IF
123      IF ((ABS(U(M))-POBS) .LT. PI/2.0)
124      &      .OR. (ABS(U(M)-POBS) .GT. 3.0*PI/2.0)) THEN
125      IS=1
126      NO=0
127      ELSE
128      IS=-1
129      NO=NV
130      END IF
131      DO 5 K=1,NV-1
132      N=NO+IS*K
133 C     *****
134 C     * GENERATE THE RECTANGULAR POINTS. *
135 C     *****
136      CPI=COS(U(M))
137      SPI=SIN(U(M))
138      CPI1=COS(U(M+1))
139      SPI1=SIN(U(M+1))
140      XS1=V(N)*CPI
141      YS1=V(N)*SPI
142      ZS1=F(M,N)
143      XS2=V(N)*CPI1
144      YS2=V(N)*SPI1
145      ZS2=F(M+1,N)
146      XS3=V(N+1)*CPI1
147      YS3=V(N+1)*SPI1

```

```

148      ZS3=F(M+1,N+1)
149      XS4=V(N+1)*CPI
150      YS4=V(N+1)*SPI
151      ZS4=F(M,N+1)
152 C      *****!*****
153 C      * ROTATE THE 4 VERTICES OF THE QUADRILATERAL. *
154 C      *****
155      CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,RP,RT)
156      CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,RP,RT)
157      CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,RP,RT)
158      CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,RP,RT)
159 C      *****
160 C      * FILL THE QUADRILATERAL. *
161 C      *****
162      WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
163 C      *****
164 C      * DRAW PERIMETER OF THE QUADRILATERAL. *
165 C      *****
166      WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
167      5      CONTINUE
168      4      CONTINUE
169 C      *****
170 C      * SHOW THE PAGE. *
171 C      *****
172      WRITE(2,100) 'showpage'
173 C      *****
174 C      * FORMATS. *
175 C      *****
176 100      FORMAT(A72)
177 101      FORMAT(F7.2,1X,F7.2,A58)
178 200      FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
179      &F7.2,' l ',F7.2,1X,F7.2,' l c')
180 201      FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
181      &F7.2,' l ',F7.2,1X,F7.2,' l w')
182      END
183 C
184      SUBROUTINE ROTATE(XA,YA,ZA,X,Y,AX,AY,AZ,RP,RT)
185      X=AX*COS(RP)*XA-AY*SIN(RP)*YA
186      Y=COS(RT)*(AX*SIN(RP)*XA+AY*COS(RP)*YA)+AZ*SIN(RT)*ZA
187      RETURN
188      END

```



```

50 C *****
51 WRITE(2,100) 'initgraphics erase page letter'
52 WRITE(2,100) '/m {moveto} def /l {lineto} def /s {stroke} def'
53 WRITE(2,100) '/sg {setgray} def /c {closepath 1 sg fill s} def'
54 WRITE(2,100) '/w {closepath 0 sg s} def /t {translate} def'
55 C *****
56 C * SET THE LINE CHARACTERISTICS. *
57 C *****
58 WRITE(2,100) '0.5 setlinewidth 1 setlinecap 1 setlinejoin'
59 C *****
60 C * TRANSLATE THE ORIGIN TO THE GEOMETRIC CENTER OF THE PAGE. *
61 C *****
62 XOFFSET=306.0
63 YOFFSET=396.0
64 WRITE(2,101) XOFFSET,YOFFSET,' t'
65 C *****
66 C * CONVERT THE OBSERVATION ANGLES TO RADIAN. *
67 C *****
68 POBS=RAD*PO
69 TOBS=RAD*TO
70 C *****
71 C * COMPUTE THE ROTATION ANGLES WHICH YIELD THE DESIRED OBSERVATION *
72 C * ANGLES. *
73 C *****
74 RP=-POBS-PI/2.0
75 RT=TOBS
76 CP=COS(RP)
77 SP=SIN(RP)
78 CT=COS(RT)
79 ST=SIN(RT)
80 C *****
81 C * IF REQUESTED CONVERT THE DATA TO DB SCALE. *
82 C *****
83 IF (IC .EQ. 1) THEN
84   TR=10.0**(0.1*ETA)
85   DO 1 M=1,NU
86     DO 2 N=1,NV
87       IF(F(M,N) .LE. TR) THEN
88         F(M,N)=(ETA+ABS(NUU))/ABS(NUU)
89       ELSE
90         F(M,N)=(10.0*ALOG10(F(M,N))+ABS(NUU))/ABS(NUU)
91       END IF
92   2 CONTINUE
93   1 CONTINUE
94 ELSE
95   END IF
96 C *****
97 C * DETERMINE THE VALUE WHICH IS JUST BELOW PO+180 DEGREES. *
98 C *****

```

```

99      IF (POBS .LT. PI) THEN
100      PA=POBS+PI
101      ELSE
102      PA=POBS-PI
103      END IF
104      DO 3 M=1,NU-1
105      IF ((PA .GE. U(M)) .AND. (PA .LT. U(M+1))) THEN
106      IREF=M
107      ELSE
108      END IF
109 3     CONTINUE
110 C     *****
111 C     * SEQUENCE THROUGH THE INDICES. *
112 C     *****
113      DO 4 K=1,NV-1
114      N=K
115      M=IREF
116      DO 5 L=1,NU-NS-1
117      IS=(-1)**L
118      M=M-IS*(L-1)
119      IF (M .LT. 1) THEN
120      M=M+NU-1
121      ELSE IF (M .GT. NU-1) THEN
122      M=M-(NU-1)
123      ELSE
124      END IF
125 C     *****
126 C     * GENERATE THE RECTANGULAR POINTS. *
127 C     *****
128      CPI=COS(U(M))
129      SPI=SIN(U(M))
130      CPI1=COS(U(M+1))
131      SPI1=SIN(U(M+1))
132      XS1=F(M,N)*CPI
133      YS1=F(M,N)*SPI
134      ZS1=V(N)
135      XS2=F(M+1,N)*CPI1
136      YS2=F(M+1,N)*SPI1
137      ZS2=V(N)
138      XS3=F(M+1,N+1)*CPI1
139      YS3=F(M+1,N+1)*SPI1
140      ZS3=V(N+1)
141      XS4=F(M,N+1)*CPI
142      YS4=F(M,N+1)*SPI
143      ZS4=V(N+1)
144 C     *****
145 C     * ROTATE THE POINTS. *
146 C     *****
147      CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,RP,RT)

```

```

148      CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,RP,RT)
149      CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,RP,RT)
150      CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,RP,RT)
151 C      *****
152 C      * FILL THE QUADRILATERAL. *
153 C      *****
154      WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
155 C      *****
156 C      * DRAW PERIMETER OF THE QUADRILATERAL. *
157 C      *****
158      WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
159 5      CONTINUE
160 4      CONTINUE
161 C      *****
162 C      * SHOW THE PAGE. *
163 C      *****
164      WRITE(2,100) 'showpage'
165 C      *****
166 C      * FORMATS *
167 C      *****
168 100      FORMAT(A72)
169 101      FORMAT(F7.2,1X,F7.2,A68)
170 200      FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
171      &F7.2,' l ',F7.2,1X,F7.2,' l c')
172 201      FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' l ',F7.2,1X,
173      &F7.2,' l ',F7.2,1X,F7.2,' l w')
174      END
175 C
176      SUBROUTINE ROTATE(XA,YA,ZA,X,Y,AX,AY,AZ,RP,RT)
177      X=AX*COS(RP)*XA-AY*SIN(RP)*YA
178      Y=COS(RT)*(AX*SIN(RP)*XA+AY*COS(RP)*YA)+AZ*SIN(RT)*ZA
179      RETURN
180      END

```

```

1      PROGRAM SPHR3D
2 C    *****
3 C    * THIS PROGRAM PRODUCES A POSTSCRIPT FILE REPRESENTING A THREE *
4 C    * DIMENSIONAL PLOT OF A FUNCTION IN SPHERICAL COORDINATES.   *
5 C    *****
6 C    * TIMOTHY J. PETERS                                           LAST UPDATED *
7 C    * THE AEROSPACE CORPORATION                                3/1/91      *
8 C    * 2350 EAST EL SEGUNDO BOULEVARD.                          *
9 C    * EL SEGUNDO, CA 90245                                       *
10 C   *****
11 C   * INPUTS:                                                     *
12 C   *                                                         *
13 C   * NU    - NUMBER OF U POINTS.                                *
14 C   * NV    - NUMBER OF V POINTS.                                *
15 C   * U(MAX) - U COORDINATE ARRAY.                               *
16 C   * V(MAX) - V COORDINATE ARRAY.                               *
17 C   * F(MAX,MAX) - FUNCTION VALUE MATRIX.                       *
18 C   * PO    - PHI OBSERVATION ANGLE IN RANGE  $0 \leq PO \leq 2\pi$ .    *
19 C   * TO    - THETA OBSERVATION ANGLE IN RANGE  $0 \leq PO \leq \pi$ .  *
20 C   * IC    - IF IC=1 THEN CONVERT THE FUNCTION VALUES TO DB.  *
21 C   *       NOTE THAT IF IC=1 THEN F MUST BE IN THE RANGE  $0 \leq F \leq 1$ . *
22 C   * ETA   - PLOT FLOOR IN DB.                                  *
23 C   * NUU   - EFFECTIVE ZERO IN DB.                              *
24 C   * NS    - NUMBER OF SEGMENTS TO REMOVE.                     *
25 C   * AX    - X DIRECTION SCALE FACTOR.                          *
26 C   * AY    - Y DIRECTION SCALE FACTOR.                          *
27 C   * AZ    - Z DIRECTION SCALE FACTOR.                          *
28 C   *                                                         *
29 C   * OUTPUT:                                                     *
30 C   *                                                         *
31 C   * SPHR.PS - POSTSCRIPT FILE REPRESENTING THE PLOT.          *
32 C   *                                                         *
33 C   *****
34     PARAMETER (MAX=100)
35     REAL*4 U(MAX),V(MAX),F(MAX,MAX),NUU
36     INTEGER P,Q
37     OPEN(UNIT=2,FILE='SPH.PS')
38     REWIND(2)
39     RAD=.17453293E-01
40     PI=.3141593E+01
41 C   *****
42 C   * READ THE INPUTS.                                           *
43 C   *****
44     OPEN(UNIT=1,FILE='DATA')
45     READ(1,*) NU,NV,AX,AY,AZ,PO,TO,IC,ETA,NUU,NS
46     READ(1,*) (U(M),M=1,NU)
47     READ(1,*) (V(N),N=1,NV)
48     READ(1,*) ((F(M,N),M=1,NU),N=1,NV)
49 C   *****

```

```

50 C      * DEFINE SOME MACROS IN POSTSCRIPT.
51 C      *****
52      WRITE(2,100) 'initgraphics erasepage letter'
53      WRITE(2,100) '/m {moveto} def /l {lineto} def /s {stroke} def'
54      WRITE(2,100) '/sg {setgray} def /c {closepath 1 sg fill s} def'
55      WRITE(2,100) '/w {closepath 0 sg s} def /t {translate} def'
56 C      *****
57 C      * SET THE LINE CHARACTERISTICS.
58 C      *****
59      WRITE(2,100) '0.5 setlinewidth 1 setlinecap 1 setlinejoin'
60 C      *****
61 C      * TRANSLATE THE ORIGIN TO THE GEOMETRIC CENTER OF THE PAPER.
62 C      *****
63      XOFFSET=306.0
64      YOFFSET=306.0
65      WRITE(2,101) XOFFSET,YOFFSET,' t'
66 C      *****
67 C      * CONVERT THE OBSERVATION ANGLES TO RADIAN.
68 C      *****
69      POBS=RAD*UO
70      TOBS=RAD*VO
71 C      *****
72 C      * COMPUTE THE ROTATION ANGLES WHICH YIELD THE DESIRED OBSERVATION
73 C      * ANGLES.
74 C      *****
75      ALPHA=-PI/2.0-POBS
76      BETA=TOBS
77 C      *****
78 C      * IF REQUESTED CONVERT THE DATA TO DB SCALE.
79 C      *****
80      IF (IC .EQ. 1) THEN
81          TR=10.0*(0.1*ETA)
82          DO 1 N=1,NU
83              DO 2 N=1,NV
84                  IF(F(M,N) .LE. TR) THEN
85                      F(M,N)=(ETA+ABS(NUU))/ABS(NUU)
86                  ELSE
87                      F(M,N)=(10.0*ALOG10(F(M,N))+ABS(NUU))/ABS(NUU)
88                  END IF
89              2 CONTINUE
90          1 CONTINUE
91      ELSE
92      END IF
93 C      *****
94 C      * DETERMINE THE INDEX OF THE ANGLES PHI_0 AND PHI_A
95 C      *****
96      IF (POBS .LT. PI) THEN
97          PA=POBS+PI
98      ELSE

```

```

99      PA=POBS-PI
100     END IF
101     DO 3 K=1,NU-1
102         IF ((POBS .GE. U(K)) .AND. (POBS .LE. U(K+1))) THEN
103             MO=K
104         ELSE
105             END IF
106         IF ((PA .GE. U(K)) .AND. (PA .LT. U(K+1))) THEN
107             MA=K
108         ELSE
109             END IF
110     3  CONTINUE
111 C     *****
112 C     * DETERMINE THE INDEX OF THE ANGLE TOBS. *
113 C     *****
114     TA=PI-TOBS
115     NO=Nv
116     NA=Nv
117     DO 4 K=1,Nv-1
118         IF ((TOBS .GE. V(K)) .AND. (TOBS .LT. V(K+1))) THEN
119             NO=K
120         ELSE
121             END IF
122         IF ((TA .GE. V(K)) .AND. (TA .LT. V(K+1))) THEN
123             NA=K
124         ELSE
125             END IF
126     4  CONTINUE
127 C     *****
128 C     * BEGIN MAIN LOOP. *
129 C     *****
130     M=MA
131     DO 5 L=1,NU-NS-1
132         M=M-((-1)**L)*(L-1)
133         IF (M .LT. 1) THEN
134             M=M+NU-1
135         ELSE IF (M .GT. NU-1) THEN
136             M=M-(NU-1)
137         ELSE
138             END IF
139 C     *****
140 C     * SET THE CONSTANTS FOR DRAWING V. *
141 C     *****
142     DP=U(M)-POBS
143     IF ((ABS(DP) .LT. PI/2.0) .OR. (ABS(DP) .GT. 3.0*PI/2.0)) THEN
144         I=Nv
145         J=0
146         L1=Nv-NO
147         L2=NO-1

```

```

148      ELSE
149          I=NA
150          J=NA-1
151          L1=NA-1
152          L2=NV-NA
153      END IF
154 C      *****
155 C      * DRAW THE V DEPENDENCY IN THE FIRST DIRECTION. *
156 C      *****
157      DO 6 K=1,L1
158          N=I-K
159 C      *****
160 C      * COMPUTE THE 4 VERTICES OF THE FIRST QUADRILATERAL. *
161 C      *****
162          CPA=COS(U(M))
163          SPA=SIN(U(M))
164          CPB=COS(U(M+1))
165          SPB=SIN(U(M+1))
166          CQA=COS(V(N))
167          SQA=SIN(V(N))
168          CQB=COS(V(N+1))
169          SQB=SIN(V(N+1))
170 C      *****
171 C      * COMPUTE THE 4 VERTICES OF THE QUADRILATERAL. *
172 C      *****
173          C=F(M,N)*SQA
174          XS1=C*CPA
175          YS1=C*SPA
176          ZS1=F(M,N)*CQA
177          C=F(M+1,N)*SQA
178          XS2=C*CPB
179          YS2=C*SPB
180          ZS2=F(M+1,N)*CQA
181          C=F(M+1,N+1)*SQB
182          XS3=C*CPB
183          YS3=C*SPB
184          ZS3=F(M+1,N+1)*CQB
185          C=F(M,N+1)*SQB
186          XS4=C*CPA
187          YS4=C*SPA
188          ZS4=F(M,N+1)*CQB
189 C      *****
190 C      * ROTATE THE 4 VERTICES OF THE QUADRILATERAL. *
191 C      *****
192          CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,ALPHA,BETA)
193          CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,ALPHA,BETA)
194          CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,ALPHA,BETA)
195          CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,ALPHA,BETA)
196 C      *****

```

```

197 C      * FILL THE QUADRILATERAL.                                     *
198 C      *****
199      WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
200 C      *****
201 C      * DRAW PERIMETER OF THE QUADRILATERAL.                       *
202 C      *****
203      WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
204 6      CONTINUE
205 C      *****
206 C      * DRAW THE V DEPENDENCY IN THE SECOND DIRECTION.             *
207 C      *****
208      DO 7 K=1,L2
209          N=J+K
210 C      *****
211 C      * COMPUTE THE 4 VERTICES OF THE FIRST QUADRILATERAL.         *
212 C      *****
213          CPA=COS(U(M))
214          SPA=SIN(U(M))
215          CPB=COS(U(M+1))
216          SPB=SIN(U(M+1))
217          CQA=COS(V(N))
218          SQA=SIN(V(N))
219          CQB=COS(V(N+1))
220          SQB=SIN(V(N+1))
221 C      *****
222 C      * COMPUTE THE 4 VERTICES OF THE QUADRILATERAL.               *
223 C      *****
224          C=F(M,N)*SQA
225          XS1=C*CPA
226          YS1=C*SPA
227          ZS1=F(M,N)*CQA
228          C=F(M+1,N)*SQA
229          XS2=C*CPB
230          YS2=C*SPB
231          ZS2=F(M+1,N)*CQA
232          C=F(M+1,N+1)*SQB
233          XS3=C*CPB
234          YS3=C*SPB
235          ZS3=F(M+1,N+1)*CQB
236          C=F(M,N+1)*SQB
237          XS4=C*CPA
238          YS4=C*SPA
239          ZS4=F(M,N+1)*CQB
240 C      *****
241 C      * ROTATE THE 4 VERTICES OF THE QUADRILATERAL.                 *
242 C      *****
243          CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,ALPHA,BETA)
244          CALL ROTATE(XS2,YS2,ZS2,X2,Y2,AX,AY,AZ,ALPHA,BETA)
245          CALL ROTATE(XS3,YS3,ZS3,X3,Y3,AX,AY,AZ,ALPHA,BETA)

```

```

246      CALL ROTATE(XS4,YS4,ZS4,X4,Y4,AX,AY,AZ,ALPHA,BETA)
247 C      *****
248 C      * FILL THE QUADRILATERAL. *
249 C      *****
250      WRITE(2,200) X1,Y1,X2,Y2,X3,Y3,X4,Y4
251 C      *****
252 C      * DRAW PERIMETER OF THE QUADRILATERAL. *
253 C      *****
254      WRITE(2,201) X1,Y1,X2,Y2,X3,Y3,X4,Y4
255 7      CONTINUE
256 C      *****
257 C      * CHECK AND SEE IF A CUT IS REQUESTED. *
258 C      *****
259      IF ((NS .GT. 0) .AND. (L .GT. NU-NS-3)) THEN
260 C      *****
261 C      * GENERATE A POLYGON IN THETA AT A CONSTANT PHI. *
262 C      *****
263      IF (POBS .LE. PI) THEN
264          IF ((U(M) .GE. POBS) .AND. (U(M) .LE. PA)) THEN
265              P=M
266          ELSE
267              P=M+1
268          END IF
269      ELSE
270          IF ((U(M) .GE. PA) .AND. (U(M) .LE. POBS)) THEN
271              P=M+1
272          ELSE
273              P=M
274          END IF
275      END IF
276      CPA=COS(U(P))
277      SPA=SIN(U(P))
278      CQA=COS(V(1))
279      SQA=SIN(V(1))
280      C=F(P,1)*SQA
281      XS1=C*CPA
282      YS1=C*SPA
283      ZS1=F(P,1)*CQA
284      CALL ROTATE(XS1,YS1,ZS1,X1,Y1,AX,AY,AZ,ALPHA,BETA)
285      WRITE(2,300) X1,Y1
286      DO 8 Q=1,NV
287          CQA=COS(V(Q))
288          SQA=SIN(V(Q))
289          C=F(P,Q)*SQA
290          XS1=C*CPA
291          YS1=C*SPA
292          ZS1=F(P,Q)*CQA
293          CALL ROTATE(XS1,YS1,ZS1,XQ,YQ,AX,AY,AZ,ALPHA,BETA)
294          WRITE(2,301) XQ,YQ

```

```

295 8      CONTINUE
296      WRITE(2,100) ' c '
297 C      *****
298 C      * DRAW A LINE AROUND THE PERIMETER OF THE POLYGON.      *
299 C      *****
300      WRITE(2,100) ' newpath '
301      WRITE(2,100) ' 0 sg '
302      WRITE(2,300) X1,Y1
303      DO 9 Q=1,NV
304          CQA=COS(V(Q))
305          SQA=SIN(V(Q))
306          C=F(P,Q)*SQA
307          XS1=C*CPA
308          YS1=C*SPA
309          ZS1=F(P,Q)*CQA
310          CALL ROTATE(XS1,YS1,ZS1,XQ,YQ,AX,AY,AZ,ALPHA,BETA)
311          WRITE(2,301) XQ,YQ
312 9      CONTINUE
313      WRITE(2,100) ' B '
314      ELSE
315      END IF
316 5      CONTINUE
317 C      *****
318 C      * SHOW THE PAGE.      *
319 C      *****
320      WRITE(2,100) 'showpage'
321 C      *****
322 C      * FORMATS.      *
323 C      *****
324 100     FORMAT(A72)
325 101     FORMAT(F7.2,1X,F7.2,A58)
326 200     FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' 1 ',F7.2,1X,
327          &F7.2,' 1 ',F7.2,1X,F7.2,' 1 c')
328 201     FORMAT(F7.2,1X,F7.2,' m ',F7.2,1X,F7.2,' 1 ',F7.2,1X,
329          &F7.2,' 1 ',F7.2,1X,F7.2,' 1 w')
330 300     FORMAT(F7.2,1X,F7.2,' m')
331 301     FORMAT(F7.2,1X,F7.2,' 1')
332      END
333 C
334      SUBROUTINE ROTATE(XA,YA,ZA,X,Y,AX,AY,AZ,RP,RT)
335      X=AX*COS(RP)*XA-AY*SIN(RP)*YA
336      Y=COS(RT)*(AX*SIN(RP)*XA+AY*COS(RP)*YA)+AZ*SIN(RT)*ZA
337      RETURN
338      END

```