The Development of a Graphical Pre- and Post Processor with Ray Tracing for the NEC-BSC Code Using A GUI

Don Davis

A Thesis

in

the Department of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science

at Concordia University

Montréal, Québec, Canada

April 1995

ISBN 0-612-01347-2

ABSTRACT

The Development of a Graphical Pre- and Post- Processor with Ray Tracing for the NEC Basic Scattering Code using A GUI

Don Davis

The growing concern about the behavior of electromagnetic fields has prompted the development of computational electromagnetic codes used in the analysis of fields and electromagnetic compatibility studies. The focus has been on the computational part of the software design rather than the processing of the raw data into a more palatable form. An example of such a situation is the NEC Basic Scattering Code. The code user must use a cumbersome interface and receives the data as a sequence of numbers. To address this issue of assisting the code user in the processing and presenting the data generated by the code, a companion code to the NEC Basic Scattering Code was developed.

The companion code, the Basic Scattering Code Viewer, BSCV is designed to assist the user in the development and analysis of different antenna platforms. The companion code uses the MATLAB environment and graphics engine to produce images of the geometry under study, the ray paths associated with the fields generated, and radiation patterns from the data generated from the NEC Basic Scattering Code.

The companion code offers a versatile package that can generate presentation quality graphics as well as assist the code user in model development and analysis.

ACKNOWLEDGEMENTS

I would like to express my thanks to Dr. Robert Paknys whose guidance and encouragement provided the inspiration for this work, and whose trust in my abilities let me believe that I could accomplish this. I would also like to thank Dr. Stanley Kubina and Dr. Chris Trueman. Without their patience, advice, and willingness to share their valuable time I would have never accomplished this.

I must thank many other people who have in no small way contributed to the work that is presented here these people are both fellow students and the ece staff that has been doing such a wonderful job. Thank you very much Ken, Ted, Quan Lu, Nicky, Dave, Guy, Terese, Mike, Greg, Ed, and my friends for putting up with me while I was doing the work on this thesis.

I would like to finally thank my family, without them there is nothing.

Don Davis, Feb. 1995

CONTENTS

List of Figures

1.0 Introduction

The growing concern about the behavior of electromagnetic fields has prompted the development of many computer codes to assist in modeling the effects different structures have on electromagnetic fields. In the area of electromagnetic compatibility, the study of the behavior of electronic devices in the presence of electromagnetic fields, the use of computers has allowed for cost efficient studies to be performed.

There have been many codes written for the purpose of analyzing the behavior of electromagnetic fields in the presence of scattering structures. At present, the need for integrated analysis packages has become more pressing. The bulk of the analysis is being done by persons who are end users of the code rather than developers of codes. Due to this fact, the people using the code must be able to perform their tasks with ease. The analysis is often incorporated into presentations and data sets. The use of an integrated package to assist the person performing the analysis as well as preparing a standard data set of the analysis is a topic that has received attention from many areas of electromagnetic field analysis [1].

The main requirements of the integrated package should be discussed. The ease of use is of great importance. This requires the building of a graphic user interface, GUI, to allow quick performance of tasks without need for memorization of a large instruction set. The standardization of the data sets so that the package is independent of the system is also important. This requires the package to be developed within a programming environment that is not operating system dependent. The package will have to produce graphical output to allow the end user to see with a one to one correspondence the input geometries as well as the output field quantities computed by the code. These requirements are summarized by the chart in Figure 1.1

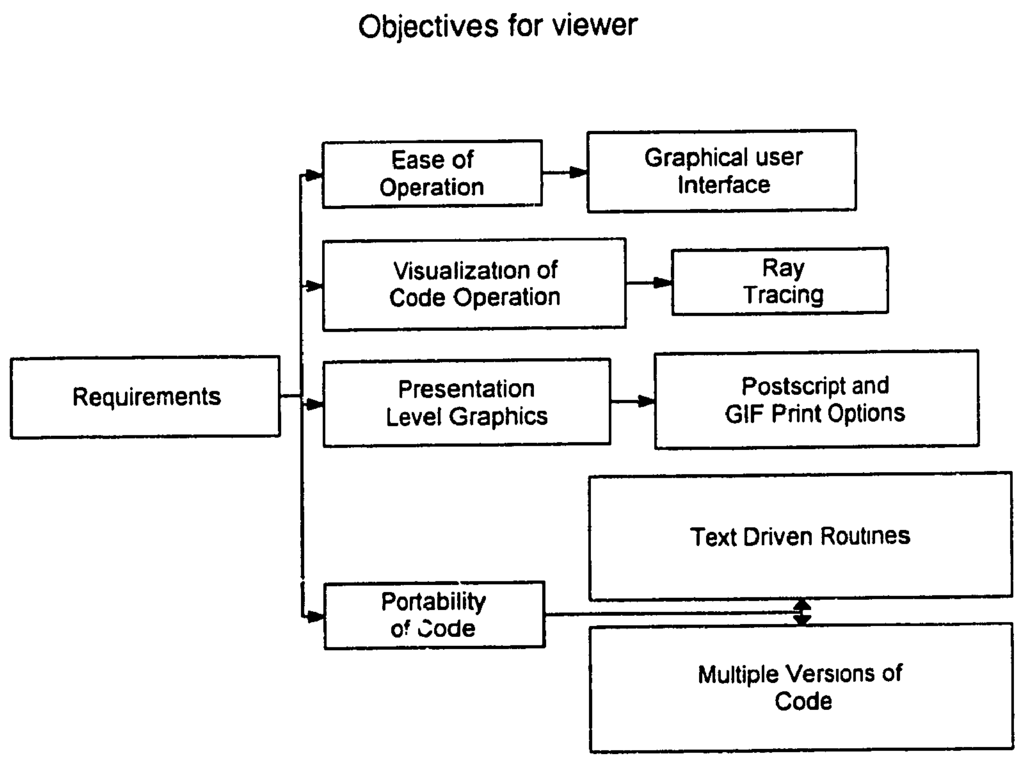


Figure 1.1 Requirements for the Viewer

This thesis will discuss one such integrated package, the BSC Viewer [2]. The BSC Viewer, BSCV, is a code developed to assist end users of the Numerical Electromagnetic

Code Basic Scattering Code [3], NECBSC, in analysis and model development. The BSCV code will allow the user to preview the input and post process the data derived from the runs. These capabilities will allow the code user to evaluate the performance of the code and analyze models under study.

The NECBSC code uses the Uniform Geometrical Theory of Diffraction to analyze the behavior of the electromagnetic fields in the presence of scattering surfaces [4]. This Uniform Geometric Theory of Diffraction, often called UTD or GTD, is a high frequency method using ray tracing and asymptotic analysis to determine the effects different structures have on an electromagnetic field. The UTD theory and the expressions used to model an electromagnetic field using UTD will be presented in Chapter 2.

The NECBSC code is composed of over fourteen thousand lines of FORTRAN code. The code is subdivided into different subroutines which perform various tasks associated with UTD analysis. The breakdown of the code and the modifications performed upon it will be discussed in Chapter 3.

The Basic Scattering Code Viewer, BSCV, is a set of programs. Some of the programs are in FORTRAN while others are routines written in MATLAB code [5]. The BSCV code performs the functions of both a pre and post processor. The BSCV environment is contained within MATLAB. The entire analysis and modeling procedure is controlled by a menu system using a graphical user interface, GUI. The design of the graphics will allow for a "what you see is what you get" system. The output of the NECBSC code is extracted by the BSCV code so that the entire modeling process is transparent to the user. In particular, the ray tracing feature allows the user to observe the UTD mechanisms that were used to calculate the electromagnetic field. This is done by plotting the ray picture from data extracted from the NECBSC code as it runs. This allows the user to develop greater insight into the behavior of the NECBSC code and the UTD analysis that it performs. The breakdown of the BSCV code as well as particulars about the operation of the code will be presented in Chapter 4.

Chapter 5 will cover examples of the BSCV output and examples of how to use the BSCV as a tool in a computerized UTD analysis. The examples will show the use of the BSCV code as a tool in analysis as well as an aide to preparing the presentation graphics used as part of a final report. The capabilities of the BSCV code as both an integrated platform for UTD analysis of systems as well as an electromagnetic compatibility study tool will be presented. The use of the BSCV code in evaluating the behavior and assessing the accuracy of the analysis of the NECBSC code will also be discussed in this section.

Chapter 6 will present the summary of the topic material as well as suggestions for further study. This section will highlight the potential for future development of the BSCV code into a complete toolbox for electromagnetic compatibility, EMC, testing.

The features listed in Section 4 allow this package to perform the functions of an integrated electromagnetic compatibility package with full graphics capability and complete portability of the code. The portability is due to the fact that the MATLAB code can run within any system that supports MATLAB. The MATLAB code is simply copied to any machine that must run the viewer. The FORTRAN components of the viewer can be compiled on both PC and UNIX platforms with no modifications.

The viewer has certain advantages over the other packages available at this time. The EAM-BSC code, Electromagnetic Antenna Modeling: BSC Analysis, is a commercially available code that runs under Windows [6]. The EAM-BSC code has no hidden line removal and does not draw the model as a solid. The model is drawn as a wire grid form. The EAM-BSC code has no ray tracing capability and may only plot the radiation patterns in a linear plot. The EAM-BSC package does not use nested menus to make the smooth transition between features. The EAM-BSC code does not allow for user specified options to be implemented within the menu system. The EAM-BSC code does have a drawing feature to produce the input file it does not give a true surface representation of the object. Thus, the EAM-BSC code lacks the versatility and range of applications that BSCV offers.

The code AAPG is a UTD based code that is older than the NECBSC2 code [7]. The AAPG code is limited to surface mounted antennas and as this is the case would be used for different models. The code ALDAS is another UTD based code that may be purchased [8]. However the source code is not available and as such cannot be modified to suit the user needs. The code PROMETHEE is yet another UTD based code that may be purchased [9]. As with ALDAS, the source code is not available which makes modifications of the code impossible.

The other option is the use of drawing programs commercially available. These codes are not always convenient to interface with codes such as the NECBSC code. The possibility of having an integrated package becomes remote. The commercial codes are also not usually menu driven and are hard to modify to suit particular user needs. The requirements of many of the drawing programs do not allow for the smooth running of the analysis codes.

One such example is the program *movie.byu* [10]. This package allows for the drawing and animation of objects in three dimensional space. However the program must be run in the dominant mode, this means that it must be run in the foreground and the user may not use other programs while this is running. The program also lacks the versatile menu systems that allow for easy use and implementation of an integrated system.

The second option is *gnuplot* [11]. This package is freeware offered by the gnu group. This package is free and has the capability of being called by other programs. The data may be sent with ease to this program. However, while this program does have hidden line removal it can only plot functions, not true surfaces, and it must be modified to draw three dimensional surfaces. This makes the ray tracing and previewing features difficult to produce without rewriting the program. The program also lacks a menu system of its own and the X-Window tool kit must be used to create a menu system for the program.

From the above listing, the ease of use and the versatility of the viewer make it an excellent choice for an integrated EMC package. The menu systems are available and can be modified to suit the user needs. The features may be modified with a text editor and there is no need to compile and debug service routines. The platform, MATLAB, has an email site for questions and problems. This allows for a very useful and versatile package to be available to the user.

2.0 The Uniform Geometrical Theory of Diffraction

To better understand the Uniform Geometrical Theory of Diffraction, also called UTD, some of the historical background and developments that helped form the theoretical base of UTD will be presented in section 2.1. Section 2.2 will give the reader an overview of the UTD theory.

2.1 Historical Background

The first people on record who were involved in the subject of optics were Pythagoras (circa 580-500 BC) and Empedocles (circa 490-430 BC). Pythagoras contributed the mathematical foundation for geometry and Empedocles continued the enlargement of the work of Pythagoras [12]. Euclid of Alexandria (circa 330-275 BC) first stated the law of reflection: that when light is reflected from a smooth surface, the angle of incidence is equal to the angle of reflection [13]. Al Hazen (circa 980 AD), presented the work that would form the basis of Fermat's principle [14]. In the year 1621 Willebrord Snell experimentally determined the law of refraction: for a ray impinging upon the interface of two media, the ratio of the sines of the angles of incidence and refraction equals the ratio of the refractive indices of the two media [15]. Fermat's principle was completed in 1654 by the French mathematician of the same name. William Hamilton produced the mathematical base for the Fermat principle from 1824 to 1844 [16]. The above works constituted the mathematical foundation for what has been called *classical geometrical optics* [17].

In addition, there was work done by physicists in the area of wave theory. Christian Huygens (1629-1695) proposed the wave theory of light and presented the first theoretical derivation of the law of refraction [18]. Robert Hooke (1635-1703) observed diffraction and interference. James Clerk Maxwell (1831-1879) produced a set of equations that allowed a mathematical expression to model the behavior of light and other forms of electromagnetic waves [19].

There was interest to join classical geometrical optic ray tracing techniques with the powerful wave theory mathematics to produce a useful general approach to modeling electromagnetic wave behavior. The work in producing this hybrid of the two schools of thought is attributed mainly to R.M. Luneberg [20] and M. Kline [21]. Their work created what is often called modern Geometrical Optics, also referred to as 'GO'.

The effects of wave diffraction still had to be accounted for. Joseph B. Keller produced a paper in June of 1953 which allowed for the modeling of the effect of diffraction from an edge [22]. This early paper created a mathematical form which could enhance the usefulness of the GO techniques. This addition to the theoretical base created a new area of study called the Geometrical Theory of Diffraction [23]. Joseph

Keller wrote additional papers describing other diffraction mechanisms such as creeping waves [24].

Further work was done by Kouyoumjian, Pathak and others to improve the mathematical model used for diffraction effects [25]. The effect of this work was the Uniform Geometrical Theory of Diffraction which was first presented November 1974 [26]. The Uniform Geometrical Theory of Diffraction, UTD, is now a respected analysis technique which is used in the modeling and analysis of the behavior of electromagnetic fields in the presence of scattering geometries. The UTD theory and applications are still being refined and developed. Examples of the applications of UTD analysis can be found in a number of papers on the UTD analysis of horn antennas and radar cross sections, RCS, of different objects [27].

2.2 Uniform Geometrical Theory of Diffraction Mechanisms

This section will discuss the various features of GTD/UTD modeling and the theoretical backbone of the techniques. The first part 2.2.1 will discuss the ray optical field formulation. The following sections will present the different mechanisms used in the construction of a UTD field. The reflected ray will be discussed in section 2.2.2. The edge diffracted ray will be discussed in section 2.2.3. The topic of corner diffracted rays will be the focus of section 2.2.4. The creeping wave or surface diffracted wave will form the basis of the discussion of section 2.2.5. Higher order rays will be presented in section 2.2.6.

2.2.1 Ray Optical Fields

M. Kline presented a paper "An Asymptotic Solution of Maxwell's Equations" in 1951 which, in combination with the text *Electromagnetic Theory and Geometrical Optics* published in 1965 by Kline and I. Kay, described the formulation of the ray optical expression for an electromagnetic field [28],[29]. The ansatz form of the expression of a high frequency electromagnetic field in a lossless, isotropic medium with constitutive parameters ε and $\mu=\mu_o$ is:

$$E(r,\omega) \sim e^{-jk\Psi(r)} \sum_{n=0}^{\infty} \frac{E_n(r)}{(j\omega)^n} \tag{2.1}$$

$$H(r,\omega) \sim e^{-jk\Psi(r)} \sum_{n=0}^{\infty} \frac{H_n(r)}{(j\omega)^n} \tag{2.2}$$

Here E and H are written as scalars that can represent any component of the vector field that is transverse to the direction of propagation. The term ansatz describes a proposed solution whose validity is demonstrated by showing that it satisfies a given set of mathematical conditions, e.g. boundary conditions. The above expansions are referred to as the Luneberg-Kline asymptotic expansions [30]. The value of $k^2=\omega^2\mu_o\varepsilon$ and $\Psi(r)$ is the phase function which will be discussed later in this section. The "~" symbol implies equal to in the asymptotic sense. A more complete description of asymptotic analysis can be obtained from the 1965 paper that Robert G. Kouyoumjian presented to the IEEE entitled "Asymptotic High-Frequency Methods" [31]. The assumption of high frequency, where ω would be large, would allow for an approximate form of the expansion to be used. This expression would only have the first term of the series, namely E_0 for the electric field and H_0 for the magnetic field. This gives the following expression for the electric field [32].

$$E(r) \underset{\omega\to\infty}{\sim} E_0(r)e^{-jk\Psi(r)} \tag{2.3}$$

The magnetic field has the following form.

$$H(r) \underset{\omega\to\infty}{\sim} H_0(r)e^{-jk\Psi(r)} \tag{2.4}$$

These expressions are the plane wave form of the fields. If the waves are not planar in nature the spread factor "*A(r)*" must be added so as to allow for different intensities along the ray path. This term will be discussed later in the chapter. The field expressions with the spread factor are shown below [33].

The electric field expression with the spread factor is:

$$E(r) \underset{\omega \to \infty}{\sim} E_o(r) A(r) e^{-jk\Psi(r)} \tag{2.5}$$

The magnetic field expression with the spread factor is:

$$H(r) \underset{\omega \to \infty}{\sim} H_o(r) A(r) e^{-jk\Psi(r)} \tag{2.6}$$

The term $\Psi(r)$ represents the phase variation along the path chosen for that particular ray. The ray optical field expression can be represented by the equations 2.5 and 2.6. This expression describes an electromagnetic field propagating along a ray path in a homogenous medium.

The spread factor must now be discussed. The expression of the spread factor is as follows:

$$A(r) = \sqrt{\frac{\varrho_1 \varrho_2}{(\varrho_1 + r)(\varrho_2 + r)}} \tag{2.7}$$

This expression uses the terms ϱ_1, ϱ_2 and r. The first term ϱ_1, refers to the first principal radius of curvature for the constant phase surface. The second term ϱ_2, refers to the second radius of curvature of the constant phase surface. The last term r refers to the position along the path of propagation, the axial ray of Figure 2.1, that is under consideration. The square root of the ratio of the gaussian curvatures of the two surfaces r=0 and r=r gives the spread factor [34]. Figure 2.1 will show the geometry used for this expression. ϱ_2

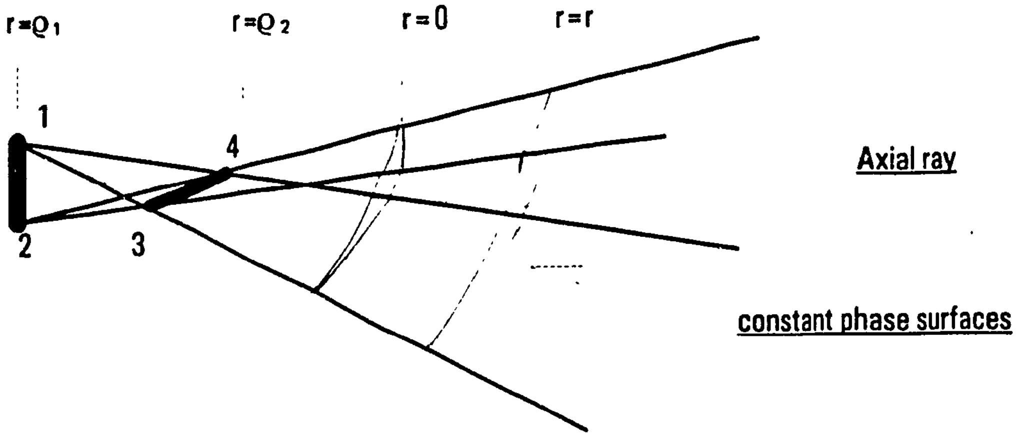


Figure 2.1 Infinitesimally narrow diverging astigmatic ray tube

The spread factor allows for the modeling of different types of waves. The expression for the spread factor takes on different forms according to the type of wave being considered. In the case of a plane wave $\varrho_1 \to \infty$ and $\varrho_2 \to \infty$. This produces a spread factor of $A(r)=1$. In the case of cylindrical waves one of the radii of curvature is finite while the other is infinite. For example, let $\varrho_1 \to \infty$ and $\varrho_2 = \varrho_2$. This makes a spread factor of the following form:

$$A(r)=\sqrt{\left|\frac{\varrho}{\varrho+r}\right|} \tag{2.8}$$

The last case to be considered is that of the spherical wave tube. In this case $\rho_1 = \rho_2 = \rho$, and this leads to the following expression for the spread factor [35]:

$$A(r) = \frac{\rho}{\rho + r} \tag{2.9}$$

Other forms can be derived from the combination of the three forms mentioned above. Thus a more complex expression can be decomposed into a expression with these three basis forms. In the case of antennas or antenna arrays the pattern factor and the array factor must be multiplied to the expression for the field. For example, the electric field for an antenna array has the following expression:

$$E(r) \underset{\omega \to \infty}{\sim} E_o(r) G(\theta,\phi) F(\theta,\phi) A(r) e^{-jk\Psi(r)} \tag{2.10}$$

Where G(θ,ϕ) and F(θ,ϕ) represent the pattern factor and the array factor respectively [36]. The expressions written above are true in vector form as well. The polarization information would be contained in the expression. In the case of a wave with polarization $\bar{\mathbf{P}}$, the expression for the electric field is of the form:

$$E(r)\bar{\mathbf{P}} \underset{\omega \to \infty}{\sim} E_o(r) \bar{\mathbf{P}} A(r) e^{-jk\Psi(r)} \tag{2.11}$$

This concludes the discussion on the ray optical field. The later parts of this chapter will discuss the various geometrical optics, GO, and UTD terms that will be used to construct a model of an electromagnetic field in the presence of scattering geometries. The properties discussed above will apply to these fields.

2.2.2 Reflected Fields

When a wave strikes a scattering surface the field will reflect in accordance to Snell's law of reflection. The law can be stated as follows: The angle of reflection equals the angle of incidence [37].

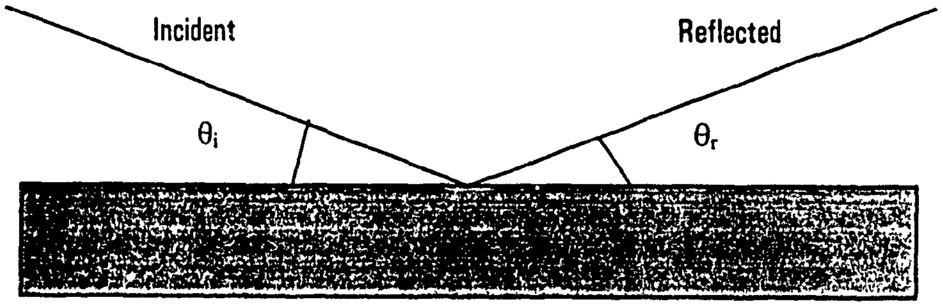


Figure 2.2 Graphical Representation of Snell's Law of Reflection

That is, $\theta_r = \theta_i$ is the assertion of Snell's law. The magnitude of the reflected field is determined by the reflection coefficient Γ. The use of perfect conductors will simplify the expression to $\Gamma = -1$ in the case of an electric field tangential to the reflecting surface and $\Gamma = 1$ in the case of an electric field parallel to the plane of incidence [38].

The two components mentioned up to this point are the direct ray, a ray that travels without striking any scattering surface, and the reflected ray. These two mechanisms constitute the geometrical optics field components. The technique used in GO modeling is ray tracing. The electromagnetic ray emanates from a source and propagates along the ray path until it either strikes a scattering object or is caught by the receiver. If the ray strikes a scattering object, it may reflect and continue to the receiver or it may refract and continue to the receiver or it may be unable to continue to the receiver. If it is unable to continue to the receiver the receiver is said to be in the shadow region for that ray.

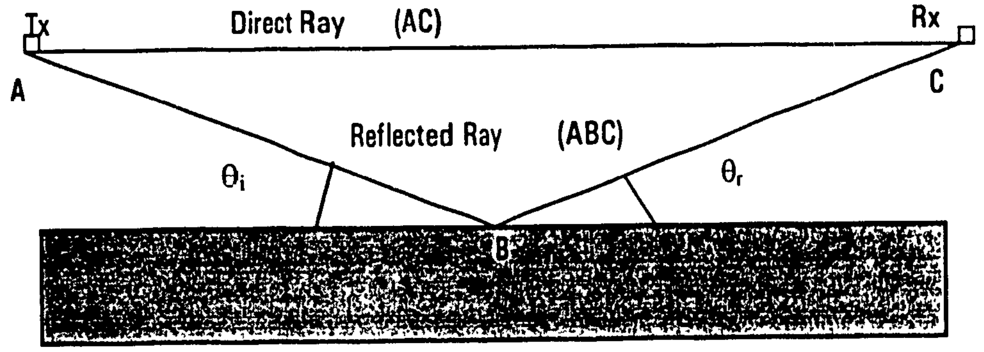


Figure 2.3 Geometrical Optics Field Mechanisms

The term Tx stands for the transmitter or the source of the field. The term Rx stands for the receiver or the observer. The rays emanate from the source and travel towards the receiver.

The GO field for the above diagram is the addition of the two rays that are present, the direct ray and the reflected ray. The two rays will travel different paths to reach the receiver. The direct ray travels from A to C. The reflected ray travels from A to B and then from B to C. These path lengths will allow for constructive or destructive interference according to the relative phases of the two rays.

$$\bar{E}_t = \bar{E}(C)e^{-jkR_1} + \Gamma\bar{E}(B)e^{-jkR_2} \tag{2.12}$$

The value of R_1 is the path length from A to C. The value of R_2 is the total path of the reflected ray which is the sum of the path length from A to B and then from B to C. The value of $\bar{E}(C)$ is the value of the field magnitude at the point in space C. The reflected field magnitude depends upon the magnitude of the field incident upon the reflection point B. The value of the magnitude may change with position in space due to the spread factor, *A(r)*, which was included in the expression of $\bar{E}(r)$. The value of the reflection coefficient Γ will depend upon the polarization of the wave and the properties of the scattering surface. The value of k will depend upon the medium that the wave is propagating in. In the case of a plane wave, the value of the field magnitude will be constant,$\bar{E}(r) = \bar{E}_o$.

The next factor to consider is the shadowing effect. The ray may not be able to reach the receiver in all cases. Figure 2.4 below illustrates some of the shadowing possibilities.

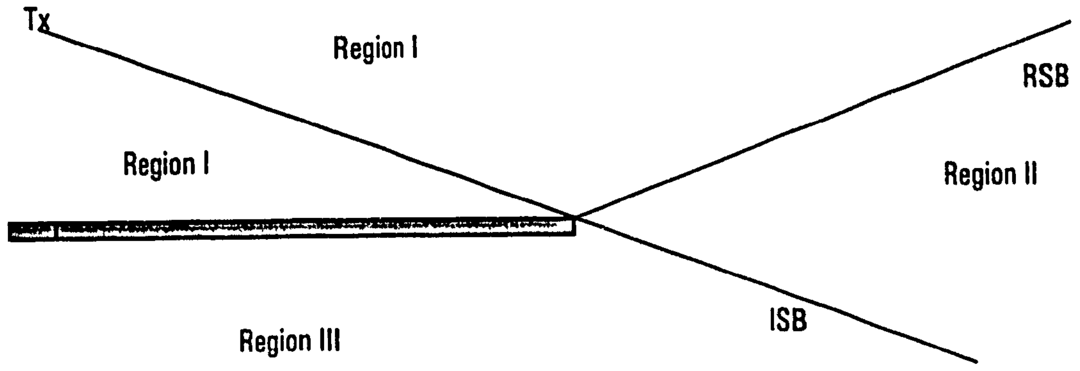


Figure 2.4 Possible GO shadowing for flat plate

In region I the direct and the reflected ray may both contact the receiver. In region II the reflection shadow boundary, RSB, has been crossed. This boundary represents the border between regions where a path exists that allows the reflected ray to reach the receiver and a region where there is no path for the reflected ray. In region III the direct ray cannot reach the receiver, nor can a reflected ray reach a receiver. The term ISB refers to the incident shadow boundary. This represents the boundary between regions where a path for a direct ray from source to receiver exists and regions where such a path does not exist. The region 1 is referred to as the deep lit region. The region 3 is referred to as the deep shadow region [39]. The GO fields for the three regions are as follows:

Region 1: $\vec{E}_t = \vec{E}_i + \vec{E}_r$ (2.13)

Region 2: $\vec{E}_t = \vec{E}_i$ (2.14)

Region 3: $\vec{E}_t = \vec{0}$ (2.15)

where $\bar{E}_t$ is the total GO field, $\bar{E}_i$ is the direct ray contribution and $\bar{E}_r$ is the reflected field contribution. It should be noted that the above example is for a single plate. In the case of multiple plates the principles are the same. The only difference is that there are more potential ray paths to consider. A ray can reflect any number of times off of multiple scattering surfaces before striking the receiver. For an accurate model, all the possible rays should be considered. The GO model will become more complex as the number and intricacy of the scattering objects increases.

There are limitations of the GO model. The GO model works well in the deep lit region. The GO model is useless in the deep shadow region as the total field is zero for the GO expression. This is not the case for the actual field behavior in the real world. Therefore the GO terms are not always enough to accurately model the behavior of electromagnetic fields in the presence of scattering bodies. The inaccuracy of the model would result in a discontinuous radiation pattern.

This leads to the next sections which will discuss the various GTD/UTD mechanisms used to model field behavior in the transition regions and the deep shadow region.

2.2.3 Edge Diffracted Rays

Joseph B. Keller offered the following statement on diffraction. "*Diffraction is the process whereby light propagation differs from the predictions of Geometrical Optics.*" [40]. This description of diffraction highlights an important factor. The creation of edge diffracted rays was spurred by the deficiencies in the GO model. The work by Joseph B.

Keller in his paper first introduced an expression for the prediction or modeling of edge diffracted rays as a way of compensating for the inaccurate expressions GO will produce when outside the deep lit region.

The edge diffraction term is added to the expression to create a more complete model of the field behavior. Edge diffraction will compensate for the loss of the reflected ray, E_r, or incident ray, E_i, due to shadowing from an edge. This third term will be called E_d for the edge diffracted ray. The expressions for the GO field can be modified by the addition of this mechanism. The example given in 2.2.2 for the GO field, equations 2.13-2.15 in the Figure 2.4 can be modified for the UTD as follows:

$$\text{Region 1: } \vec{E}_t = \vec{E}_i + \vec{E}_r + \vec{E}_d \tag{2.16}$$

$$\text{Region 2:} \vec{E}_t = \vec{E}_i + \vec{E}_d \tag{2.17}$$

$$\text{Region 3: } \vec{E}_t = \vec{E}_d \tag{2.18}$$

The term E_d is the term for the edge diffracted field. The expression for the diffracted ray may be expressed in the following form [41]:

$$\vec{E}_d = \vec{E}_i \cdot \mathbf{D} \sqrt{\frac{\rho}{r(r+\rho)}} e^{-jkr} \tag{2.19}$$

$$\mathbf{D} = -\vec{\beta}'_o \vec{\beta}_o D_s - \vec{\phi}' \vec{\phi} D_h \tag{2.20}$$

$$D_{s,h} = \left(L', L^{ro}, L^{rn}, \phi, \phi', \beta_o, n\right) = D_1 + D_2 \mp \left(D_3 + D_4\right) \tag{2.21}$$

$$D_1 = \frac{-e^{-j\pi/4}}{2n\sqrt{2\pi k}\sin\beta_o}\cot\left[\frac{\pi + (\phi - \phi')}{2n}\right]F\left[kL'a^{+}(\phi - \phi')\right] \tag{2.22}$$

$$D_2 = \frac{-e^{-j\pi/4}}{2n\sqrt{2\pi k}\sin\beta_o}\cot\left[\frac{\pi - (\phi - \phi')}{2n}\right]F\left[kL'a^{-}(\phi - \phi')\right] \tag{2.23}$$

$$D_3 = \frac{-e^{-j\pi/4}}{2n\sqrt{2\pi k}\sin\beta_o}\cot\left[\frac{\pi + (\phi + \phi')}{2n}\right]F\left[kL^{rn}a^{+}(\phi + \phi')\right] \tag{2.24}$$

$$D_4 = \frac{-e^{-j\pi/4}}{2n\sqrt{2\pi k}\sin\beta_o}\cot\left[\frac{\pi - (\phi + \phi')}{2n}\right]F\left[kL^{ro}a^{-}(\phi + \phi')\right] \tag{2.25}$$

$$F(x) = 2j\sqrt{x}e^{jx}\int_{\sqrt{x}}^{\infty} e^{-ju^2}\,du \tag{2.26}$$

$$L' = \frac{r(\varrho_e' + r)\varrho_1'\varrho_2'}{\varrho_e'(\varrho_1' + r)(\varrho_2' + r)}\sin^2\beta_o \tag{2.27}$$

$$L^{ro,n} = \frac{r(\varrho_e^{ro,n} + r)\varrho_1^{ro,n}\varrho_2^{ro,n}}{\varrho_e^{ro,n}(\varrho_1^{ro,n} + r)(\varrho_2^{ro,n} + r)}\sin^2\beta_o \tag{2.28}$$

$$\frac{1}{\varrho_1^r}=\frac{1}{\varrho_1^i}+\frac{2}{a_1\cos\theta'} \tag{2.29}$$

$$\frac{1}{\varrho_2^r}=\frac{1}{\varrho_2^i}+\frac{2\cos\theta'}{a_2} \tag{2.30}$$

$$\frac{1}{\varrho_e^{ro,n}}=\frac{1}{\varrho_e^i}-\frac{2\left(\vec{n}_e\cdot\vec{n}_{o,n}\right)\left(\vec{r}^{\,i}\cdot\vec{n}_{o,n}\right)}{|a_e|\sin^2\beta_o} \tag{2.31}$$

$$\vec{E}^i=E^i_{\beta_o^i}\,\vec{\beta}_o^{\,i}+E^i_{\phi^i}\,\vec{\phi}^{\,i} \tag{2.32}$$

The term $\vec{E}_i$ refers to the field incident upon the diffraction point. The term p refers to the edge caustic, a caustic is a confluence of the ray paths, distance. $\vec{\beta}_o$ and $\vec{\beta}_o^{\,i}$ are used with reference to the edge fixed coordinate system. The variables $\vec{\phi}$ and $\vec{\phi}^{\,i}$ are also used with respect to the edge coordinate system. The $D_{s,h}$ represents the edge diffraction coefficient for the soft, Dirichlet condition, and hard, Neumann condition, cases respectively. The terms $\hat{e}$,$\hat{s}$ and $\hat{s}^{i}$ are used to represent the vector expressions for the edge of the wedge, the direction of propagation for the diffracted ray and the direction of propagation of the ray incident upon the edge. The term ϕ' represents the angle between the direction of propagation vector $\hat{s}$, for the ray incident upon the edge, and the edge vector $\hat{e}$. The term ϕ represents the angle between the propagation vector $\hat{s}$, for the diffracted ray, and the edge vector $\vec{e}$. The ray propagates in the direction $\vec{s}'$. The ray causes a diffracted ray to emanate from the edge of the wedge which propagates in the direction $\vec{s}$. The edge is defined by the vector$\vec{e}$. The vectors will all be referenced to the edge of the wedge. The mathematical format for the vectors is as follows:

$$\vec{\phi}' = \frac{-\vec{e} \times \vec{s}'}{|\vec{e} \times \vec{s}'|} \tag{2.33}$$

$$\vec{\beta}_o' = \vec{\varphi}' \times \vec{s}' \tag{2.34}$$

$$\vec{\phi} = \frac{\vec{e} \times \vec{s}}{|\vec{e} \times \vec{s}|} \tag{2.35}$$

$$\vec{\beta}_o = \vec{\phi} \times \vec{s} \tag{2.36}$$

Figure 2.5 represents the geometry being discussed above. The o face corresponds to the face of the wedge that is illuminated. The n face is the face that is shadowed. The choice of the o and n face is arbitrary. The notation used in the diagram is one of the most common forms of diffraction notation [42]. The diffracted ray s will emanate from the edge of the wedge in such a way that a cone of diffracted rays can be determined. This cone is defined by the angle that the incident ray makes with the edge of the wedge. This angle will control the shape of the cone. Figure 2.6 presents an example of the form of the diffraction cone. The cone is also defined by the law of edge diffraction [43], which states that $\beta'_o = \beta_o$

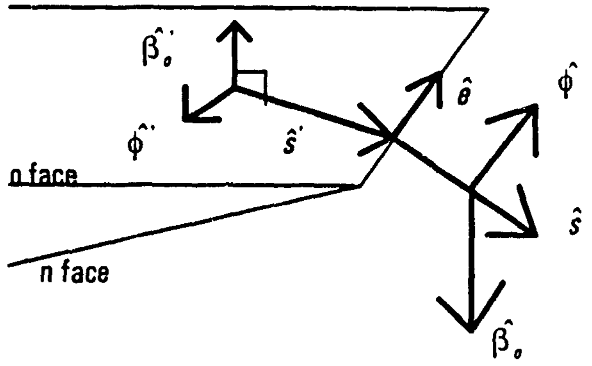


Figure 2.5 Edge Fixed Coordinate system

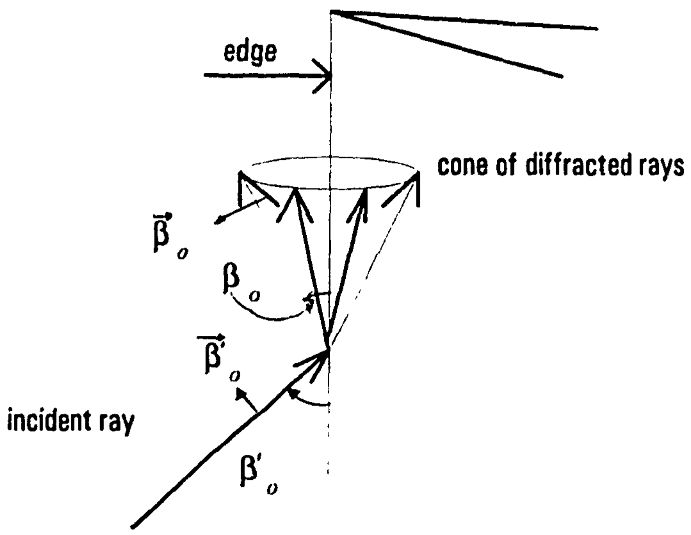


Figure 2.6 Cone of diffracted rays produced by ray incident on edge

The terms $\varrho_1^{r o,n}$ and $\varrho_2^{r o,n}$ represent the principal radii of curvature of the reflected wave front from the o- and n- faces respectively. The distances $\varrho_e^{r o,n}$ are the radii of curvature of the reflected wave front in the plane containing the reflected ray and the edge. The symbol $a_{1,2}$ represents the radius of curvature of the surface at the point of diffraction Q_e with the subscript 1 pertaining to the plane of incidence and the subscript 2 pertaining to the plane transverse to the plane of incidence. The unit vectors $\hat{n}_{o,n}$ are the unit vectors normal to the o- and n- faces at Q_e respectively. The unit vector $\hat{n}_e$ is normal to the edge at the point Q_e. The point Q_e is the position on the edge where the diffracted ray emanates from [44].

The diffracted field can be modeled by the use of expressions 2.19-2.32. This modeling of the field requires information about the field incident upon the edge of the wedge, 2.32, as well as information about the geometry of the wedge, 2.27-2.31, and the position of the receiver, 2.22-2.25. The diffracted field may then be calculated and used in the expression for the total field, equations 2.16-2.18.

The diffraction term will appear for each edge that has a ray incident upon it. This means that there is a possible edge diffraction contribution for each edge in the scattering geometry under consideration. There is a region where the diffracted rays are not ray optical in nature. These regions are called the transition regions. The transition regions are shaped like cones and centered around the shadow boundaries of the wedge. The mathematical expression would be based upon the region where $F(kLa^{\pm})<1$ where F is the function defined in 2.26. If the receiver is within this region the diffracted field contribution cannot be used as it is not properly defined, not a ray optical field [45].

The edge diffracted rays will occur as long as there is an edge illuminated by a ray incident upon the edge. The rays will arrive at the receiver unless shadowed by some other structure or if the diffraction point is located on a point off of a finite edge [46].

2.2.4 Corner Diffracted Rays

Corner diffraction compensates for the loss of an edge diffracted ray when the diffraction point is off of a finite edge. The effect of the corner diffraction is the removal of a discontinuity caused by the abrupt termination of an edge. The formulation is shown below [47]:

$$\begin{bmatrix} E^c_{\beta_e} \\ E^c_{\phi} \end{bmatrix} = \begin{bmatrix} I & Z \\ M & Y \end{bmatrix} \frac{\sqrt{\sin\beta_c \sin\beta_{oc}}}{\cos\beta_{oc} - \cos\beta_c} F\left(kL_c a\left[\pi + \beta_{oc} - \beta_c\right]\right) \frac{e^{-jkr}}{4\pi r} \tag{2.37}$$

Corner diffraction was derived for a flat plate with equivalent currents I and M and with Z and Y being the free space impedance and admittance respectively.

$$\begin{bmatrix} I \\ M \end{bmatrix} = -\begin{bmatrix} E^i_{\beta_o} \\ E^i_{\phi} \end{bmatrix} \begin{bmatrix} C_s(a_o)Y \\ C_h(a_o)Z \end{bmatrix} \sqrt{\frac{8\pi}{k}} e^{-j\pi/4} \tag{2.38}$$

$$C_{s,h} = \frac{-e^{-i\pi/4}}{2\sqrt{2\pi k}\sin\beta_o}\left\{ \frac{F\left[kLa(\phi-\phi')\right]}{\cos\left(\frac{\phi-\phi'}{2}\right)} \left| F\left(\frac{\left[La^{(\phi-\phi')}/\lambda\right]}{kL_c a(\pi+\beta_{oc}-\beta_c)} \right) \right| \right.$$

$$\left. \mp \frac{F\left[kLa(\phi+\phi')\right]}{\cos\left(\frac{\phi+\phi'}{2}\right)} \left| F\left(\frac{\left[La^{(\phi-\phi')}/\lambda\right]}{kL_c a(\pi+\beta_{oc}-\beta_c)} \right) \right| \right\} \quad (2.39)$$

$$L = \frac{r'r''}{r'+r''}\sin^2\beta_o \quad (2.40)$$

$$L_c = \frac{r_c r}{r_c + r} \quad (2.41)$$

$$a(\phi \mp \phi') = 2\cos^2\left(\frac{\phi \mp \phi'}{2}\right) \quad (2.42)$$

In the case of a general wedge, where n≠2, the following expression can be used:

$$C_{s,h} = \frac{-e^{-i\pi/4}}{2n\sqrt{2\pi k}\sin\beta_o}\left[\cot\left(\frac{\pi-\beta^-}{2n}\right) F\left[kL^i a(\beta^-)\right.\right.$$

$$\cdot \left| F\left\{ \frac{\left[L^{i}\, a(\beta^{-}) \big/ \lambda \right]}{kL_{c}a\left(\pi + \beta_{oc} - \beta_{c}\right)} \right\} \right| + \cot\left(\frac{\pi - \beta^{-}}{2n} \right) F\left[kL'a(\beta^{-}) \right]$$

$$\cdot \left| F\left\{ \frac{\left[L^{i}\, a(\beta^{-}) \big/ \lambda \right]}{kL_{c}a\left(\pi + \beta_{oc} - \beta_{c}\right)} \right\} \right| \mp \cot\left(\frac{\pi - \beta^{+}}{2n} \right) F\left[kL^{m}a(\beta^{+}) \right]$$

$$\cdot \left| F\left\{ \frac{\left[L^{m}\, a(\beta^{+}) \big/ \lambda \right]}{kL_{c}a\left(\pi + \beta_{oc} - \beta_{c}\right)} \right\} \right| + \cot\left(\frac{\pi - \beta^{+}}{2n} \right) F\left[kL^{ro}a(\beta^{+}) \right]$$

$$\cdot \left| F\left\{ \frac{\left[L^{ro}{}_{0}\, a(\beta^{+}) \big/ \lambda \right]}{kL_{c}a\left(\pi + \beta_{oc} - \beta_{c}\right)} \right\} \right| \Bigg] \qquad (2.43)$$

It should be noted that the term $\beta^{\mp} = \phi \mp \phi'$. The geometry of the diffraction problem is presented in Figure 2.7.

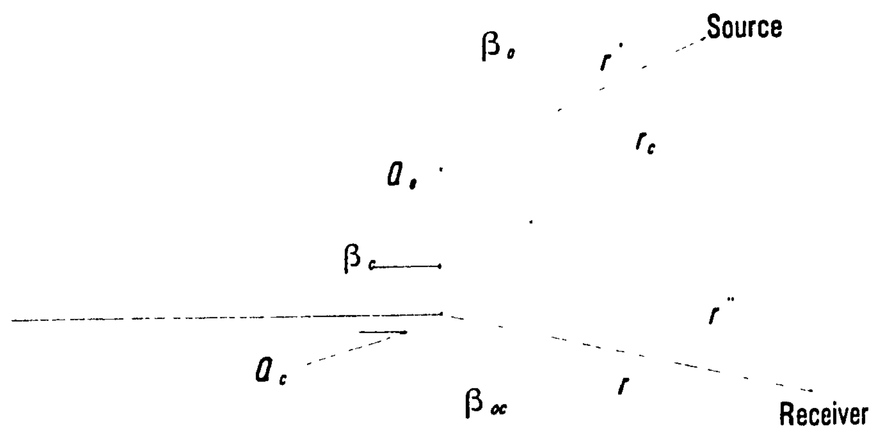


Figure 2.7 Geometry for Corner Diffraction

The term Q_e represents the diffraction point for the edge diffracted ray. The term Q_c represents the corner at which the corner diffracted ray will emanate. The term β_c symbolizes the angle measured from the edge to the ray emanating from the source and striking the corner. The term β_{oc} refers to the angle measured from the edge to the ray emanating from the corner and propagating towards the receiver. The corner diffracted field term is added to the expression for the total field. There will be one corner diffraction term per corner.

The corner diffracted fields have a spherical phase front while the edge diffracted field has a cylindrical phase front. These terms represent the UTD mechanisms for the effects of wedges on the behavior of electromagnetic fields. The next factor to be considered is the effect of a smooth surface on the behavior of electromagnetic waves.

2.2.5 Surface Diffraction

The effect of a smooth surface shadowing an area will now be considered. When a ray strikes a smooth conducting surface, the surface may allow the ray to diffract along the surface and then leave the surface at a different point. The expressions used to generate the model for this phenomenon was taken from the paper written by Prabhakar Pathak, Walter Burnside, and Ronald Marhefka. The paper entitled *A Uniform GTD Analysis of the Diffraction of Electromagnetic Waves by a Smooth Convex Surface* presented the following expressions for the modeling of the surface diffracted fields [48].

$$\bar{E}(P_L) \sim \bar{E}^i(P_L) + \bar{E}^i(Q_R)\cdot\bar{\bar{R}}\sqrt{\frac{\rho_1^r\,\rho_2^r}{(\rho_1^r+s^r)(\rho_2^r+s^r)}}\,e^{-jks^r} \tag{2.44}$$

The term P_L represents a point in the lit region. If the shadow region is to be considered at a point P_S, where P_S is within the shadow region.

$$\bar{E}(P_S) \sim \bar{E}^i(Q_1)\cdot\bar{\bar{T}}\sqrt{\frac{\rho_2^d}{s^d\,(\rho_1^d+s^d)}}\,e^{-jks^d} \tag{2.45}$$

The dyadics, a shorthand matrix expression, have the following forms.

$$\bar{\bar{R}} = R_s\hat{e}_\perp\hat{e}_\perp + R_h\hat{e}^i_\parallel\hat{e}^r_\parallel \tag{2.46}$$

The unit vector $\bar{e}_{\perp}$ is the unit vector perpendicular to the plane of incidence of the reflecting surface. The unit vectors $e'_{\|}$ and $e'_{\|}$ represent the unit vectors parallel to the reflecting surface for the incident and reflected rays respectively.

$$\bar{\bar{T}} = T_s \vec{b}_1 \vec{b}_2 + T_h \vec{n}_1 \vec{n}_2 \tag{2.47}$$

$$R_{s,h} = -\left[\sqrt{\frac{-4}{\xi^{L}}}\, e^{-i (\xi^{L})^3 / 12} \left\{ \frac{e^{-i(\pi/4)}}{2\sqrt{\pi}\,\xi^{L}} \left[1 - F(X^{L})\right] + \hat{P}_{s,h}(\xi^{L}) \right\}\right] \tag{2.48}$$

$$T_{s,h} = -\left[\sqrt{m(a_1)m(a_2)}\sqrt{\frac{2}{k}} \left\{ \frac{e^{-i(\pi/4)}}{2\sqrt{\pi}\,\xi^{d}} \left[1 - F(X^{L})\right] + P_{s,h}(\xi^{d}) \right\}\right] \sqrt{\frac{d\eta(a_1)}{d\eta(a_2)}}\, e^{-jkt} \tag{2.49}$$

$$P_s(x) = p^{*}(x) e^{-i\pi/4} - \frac{e^{-i\pi/4}}{2x\sqrt{\pi}} \tag{2.50}$$

$$P_h(x) = q^{*}(x) e^{-i\pi/4} - \frac{e^{-i\pi/4}}{2x\sqrt{\pi}} \tag{2.51}$$

$$p^{*}(x) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{V(\lambda) e^{-jx\lambda}}{W_2(\lambda)} d\lambda \tag{2.52}$$

$$q^{*}(x) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \frac{V'(\lambda) e^{-jx\lambda}}{W_2'(\lambda)} d\lambda \tag{2.53}$$

$$V(\lambda) = \frac{W_1(\lambda) - W_2(\lambda)}{2j} \tag{2.54}$$

$$V'(\lambda) = \frac{dV(\lambda)}{d\lambda} \tag{2.55}$$

$$W_1(\lambda) = \frac{1}{\sqrt{\pi}} \int_{C_1} e^{-z^3/3} e^{\lambda z} dz \tag{2.56}$$

$$W_2(\lambda) = \frac{1}{\sqrt{\pi}} \int_{C_2} e^{-z^3/3} e^{\lambda z} dz \tag{2.57}$$

The terms p,q are the Pekeris functions and the terms W_1, W_2 are the Fock type Airy functions. Descriptions of these functions can be found in the book by McNamara, Pistorius and Malherbe, *Introduction to the Uniform Geometrical Theory of Diffraction* [49]. The parameters C_1 and C_2, in equations 2.56 and 2.57 respectively, refer to the contour followed by the integrals. The form of the path C_1 is shown in Figure 2.8.

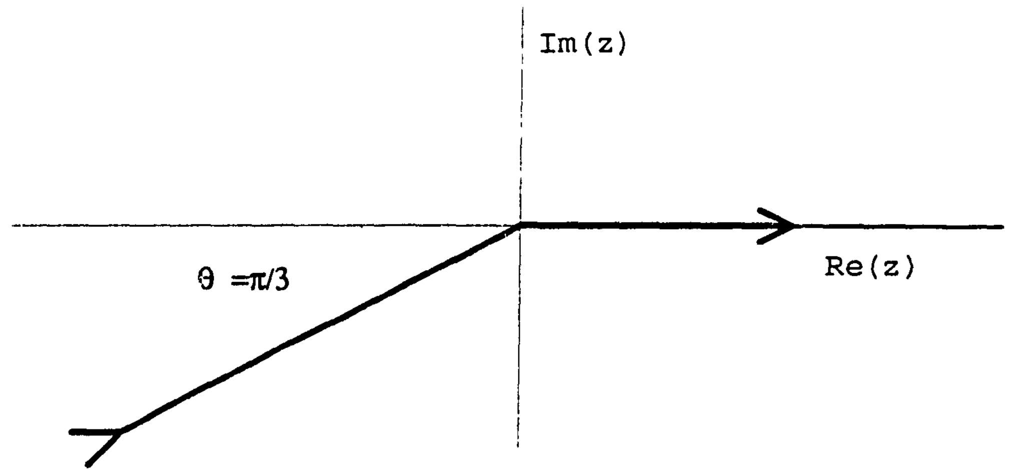


Figure 2.8 Path C_1 used for W_1 calculation

The path C_1 can be seen to come from ∞ to 0 using the path arg=$-2\pi/3$, measured from the real axis positive side. The path will go from 0 to ∞ using the real axis as the path.

The path for the second contour C_2 is shown below in the Figure 2.9. From the diagram it can be seen that the path will go from ∞ to 0 using the path arg=$2\pi/3$. The second part of the path will be that traveling from 0 to ∞ using the real axis. As before the argument is measured from the positive real axis.

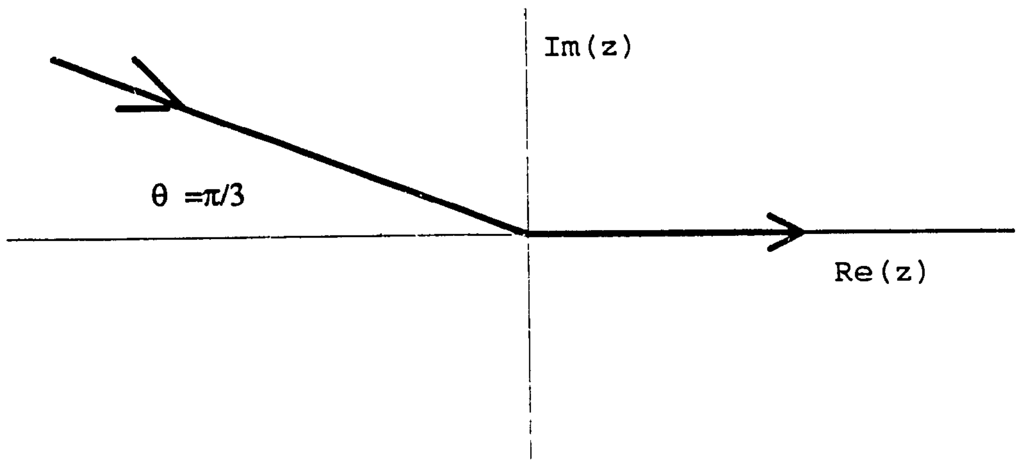


Figure 2.9 Path C_2 used for W_2 calculation

The vectors defined in the expressions 2.46 and 2.47 are determined from the geometry below. The point Q_1 is the point on the surface that the incident ray strikes. The point Q_2 is the point on the surface that the surface diffracted ray is shed. The surface diffracted ray will propagate until it is either shadowed or it reaches the receiver. The vectors presented in Figure 2.10 below are either normal, $\vec{n}_{1,2}$, or tangential, $\vec{t}_{1,2}$, to the surface at the point in question Q_1 or Q_2.

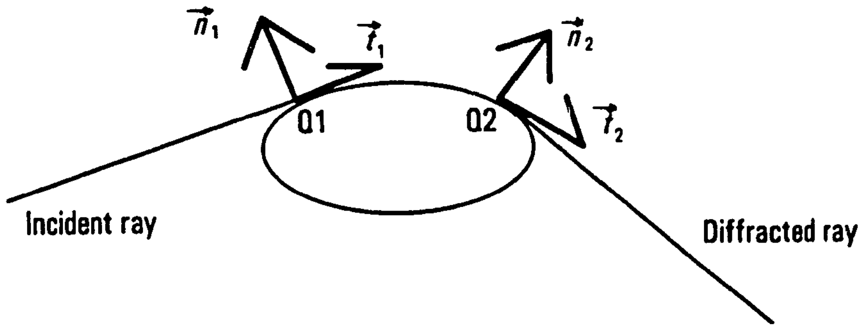


Figure 2.10 Determination of Vectors for Surface Diffraction

The vector quantities, shown in Figure 2.10 have can be expressed in the formulation given by equation 2.58 shown below,

$$\vec{t}_{1,2} \times \vec{n}_{1,2} = \vec{b}_{1,2} \tag{2.58}$$

while the parameters of the expressions are given by the following expressions,

$$\xi^{L} = -2m(a_R)\left[f(a_R)\right]^{1/3} \cos\theta' \tag{2.59}$$

$$f(a_R) = 1 + \left[\frac{\varrho_g^2(a_R)\cos^2\theta'}{\varrho_g(a_R)\varrho_t(a_R)} - \frac{\varrho_g^2(a_R)\cos^2\theta^i}{R_1(a_R)R_2(a_R)} \right] \approx 1 \tag{2.60}$$

$$\xi^{d} = \int_{a1}^{a2} dt' \frac{m(t')}{p_g(t')} \tag{2.61}$$

$$m(a) = \left[\frac{kp_g(a)}{2} \right]^{1/3} \tag{2.62}$$

$$t = \int_{a1}^{a2} dt' \tag{2.63}$$

$$X^{i} = 2kL^{i}\cos^{2}\theta^{i} \tag{2.64}$$

$$X^{d} = \frac{kL^{d}(\xi^{d})^{2}}{2m(Q_{1})m(Q_{2})} \tag{2.65}$$

$$L^{i} = l \quad . \tag{2.66}$$

The term l is the length of the ray path from the source to the reflection point Q_R on the surface where the ray will strike.

$$L^{d} = s_{o} \tag{2.67}$$

The term s_0 is the length of the path from the source to the point where the surface diffracted ray strikes the surface. This point is named Q_1.

$$\sqrt{\frac{d\eta(Q1)}{d\eta(Q2)}} = \sqrt{\frac{s_{o}}{s_{o}+t}} \tag{2.68}$$

$$\varrho_{1}^{r} = \left[\frac{1}{l} + \frac{2\sin^{2}\theta_{o}}{\varrho_{t}\cos\theta^{i}}\right]^{-1} \tag{2.69}$$

The term ρ_t represents the principal radius of curvature in the circumferential direction at the point Q_R.

$$\rho_2^r = l \tag{2.70}$$

$$\rho_2^d = s_o + t \tag{2.71}$$

The term ρ_g refers to the radius of curvature of the surface at the point Q_R. The term t refers to the length of the path taken by the surface diffracted ray. The ray may travel both clockwise and counter clockwise around the surface until it sheds or strikes a discontinuity. In Figure 2.11 a sample of the geometry is given. The ray is incident upon the surface at the point Q_1. The ray travels along the surface until it sheds from the point Q_2. The length of the path is t. In this case the shadowing surface is a elliptical cylinder. The creeping wave is actually not on the surface of the cylinder but just above the surface. The wave will lose a significant portion of the strength it began with as it travels in this fashion.

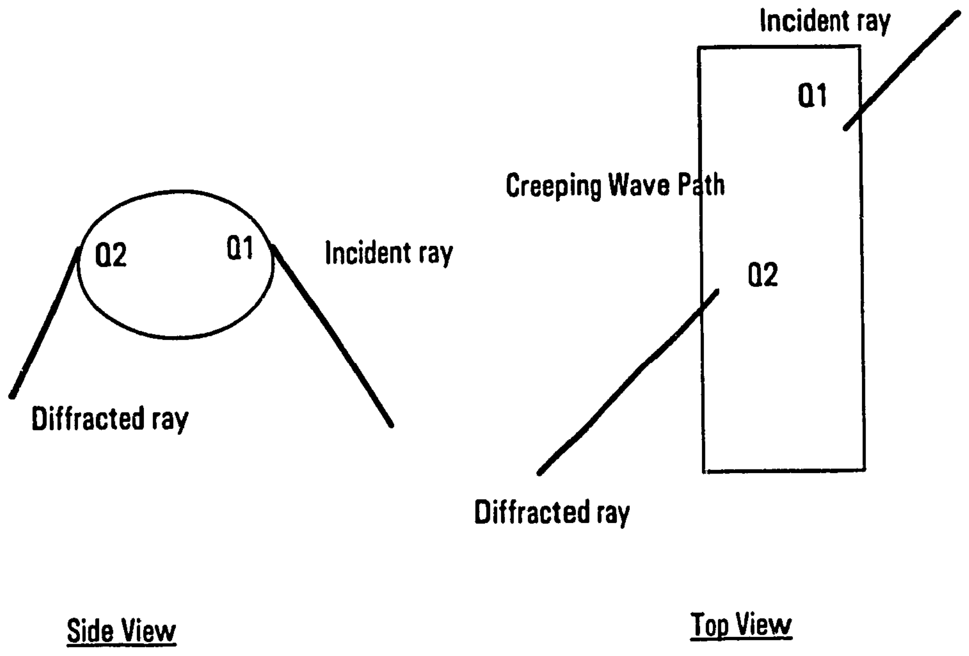


Figure 2.11 View of Surface Diffracted Ray Path

The fundamental UTD mechanisms have now been covered. The effects of edges, corners, and surface shadowing have been considered. The next step is to consider the combination of the mechanisms.

2.2.6 Higher Order Rays

There are two elements that will be considered in this section. The first is slope diffracted rays. The second is higher order rays due to complex geometries.

The slope diffracted ray is proportional to the derivative of the incident field at the diffraction point. The slope diffracted field can be expressed as:

$$U^{d}(s)=\frac{1}{jk}\frac{\partial D_{s,h}}{\partial \phi'}\frac{\partial U}{\partial N}\bigg|_{Q_e}\frac{e^{-jks}}{\sqrt{s}} \tag{2.72}$$

Where U represents the field of interest and the diffraction term can be represented by the derivative of equation 2.20. The slope diffraction mechanism is used to compensate for the vanishing of the incident field due to the source field pattern.

The other higher order rays to be considered are those that are formed by multiple UTD mechanisms. A ray may be reflected off of one surface, and then strike another surface before it strikes the receiver. Thus a ray may be reflected and then reflected again, reflected and then diffracted, or diffracted and then reflected, or diffracted and then diffracted again. The method is as follows: the first UTD mechanism's resulting ray is used as the incident ray for the next mechanism. There is no maximum number of UTD mechanisms that can be used to generate a ray path from source to receiver. The only limitations are those of the transition region and of caustics. The field in the transition region is not ray optical. Therefore if the next UTD mechanism is within the transition region of the previous one, the incident ray cannot be used as it will violate the initial ray optical assumption. At a caustic there are an infinite number of rays, such as at the focus of a parabola. Since it is impossible to sum up the contributions of an infinite number of rays, the UTD method fails and a different method should be used. Physical optics or the equivalent current method are possible alternatives in such a situation.

The total GTD field is expressed as a sum of the possible GTD mechanisms that have ray paths striking the observation point. This can be written as the expression in equation 2.73.

$$\vec{E}_t = \vec{E}_i + \vec{E}_r + \vec{E}_d + \vec{E}_c + \vec{E}_s + \vec{E}_{sp} + \vec{E}_{ho} \tag{2.73}$$

The term $\vec{E}_t$describes the total field at the observation point. The term $\vec{E}_i$ is the direct ray contribution to the total field. The term $\vec{E}_r$ represents the contribution of the reflected ray to the total field. The term $\vec{E}_d$ is the contribution of the edge diffracted rays to the total field. The corner diffracted ray contribution is represented by $\vec{E}_c$. The field contribution from surface diffracted rays is represented by $\vec{E}_s$. The slope diffracted field contribution is represented by $\vec{E}_{sp}$. The last term to be considered is the contribution by the higher order rays, expressed as $\vec{E}_{ho}$.

These mechanisms serve to give a complete and smooth radiation pattern. The more complex the geometry under study, the larger number of GTD terms are required to give an accurate representation of the field. Therefore not all GTD terms are required to give a good representation of the field in the case of a simple geometry. If a GTD term is required but not used, a discontinuity of the radiation pattern will occur. This discontinuity of the radiation pattern is an indication of a missing GTD term. A missing GTD term would cause an inaccuracy in the calculated field.

There are other UTD mechanisms which are not mentioned in this thesis. For a more complete listing of the various UTD terms, the book *Introduction to the Uniform Geometrical Theory of Diffraction* by D.A. McNamara is a good text [13].

The next section is concerned with the NECBSC code. This code will use the above mentioned mechanisms to evaluate UTD solutions for different geometries. The analysis techniques used by the code are based upon the formulations used in this chapter. The three dimensional nature of the geometries is taken into account by the code and thus the full three dimensional formulation of the mechanisms were presented.

3.0 NEC Basic Scattering Code

The Numerical Electromagnetic Code-Basic Scattering Code version 2, NECBSC, is a computer code for the electromagnetic analysis of the radiation from antennas in the presence of complex structures at UHF and above [50]. The code uses the UTD theory to perform the analysis of the structures. The restriction of the UTD theory is that the object must be at least a wavelength in size. This implies that it is a high frequency technique. This could be expected in light of the background and theoretical base presented in chapter 2.

The code is written in FORTRAN and uses a series of subroutines governed by a main program. The input and output of the code are written as text files. The input file is a read only file for the NECBSC code. The output file is a write only file for the NECBSC code.

In the first section, 3.1, the operation of the code will be discussed. Section 3.2 will discuss the code structure and the subroutine functions. Section 3.3 will discuss the limitations and restrictions of the code.

3.1 Operation of the Code

The NECBSC code will first determine what geometries are present. This information can be obtained from the input file. The geometry of the structure will determine what UTD mechanisms are required. The code allows for combinations of elliptic cylinders and polygonal flat plates. The presence of flat plates will require the use of both edge and corner diffraction. The plates will also use the reflected ray in the solution. The direct ray is present for all geometries. The cylinder will require the surface diffraction mechanism as well as the reflected ray and the edge diffraction terms, for the endcaps of the cylinder.

The presence of complex geometries will increase the number of terms required for a complete analysis of the field. Multiple plates or cylinders will generate higher order rays due to their interaction. Plates and cylinders may interact as well, however this version of the code does not take this effect into consideration so this thesis will not dwell on this topic.

The interaction of the multiple plates will be called upon if the structure has more than one flat plate. In the same manner the cylinder interaction will be called upon if the structure warrants the use of that UTD mechanism. These higher order terms are used in addition to the lower order UTD that were mentioned earlier.

The code will calculate the UTD field in the following manner. The main program will initially cycle through the geometry to determine what UTD mechanisms are applicable. The first step is to find the terminal point of the rays. This is done by determining the receiver location, which will be the terminal point for the rays emanating from the source. The next step is to cycle through the UTD mechanisms for each part of the structure.

As the code looks at each possible UTD mechanism it will determine if the mechanism under consideration can reach the receiver. If the mechanism is shadowed and it cannot reach the receiver, it will not be included into the expression for the total field at the receiver. The code will cycle through each possible mechanism for each part of the structure to sum up all the contributions to the total field.

The polarization of the field is described in terms of the reference coordinate system defined by the user. The code will store all the field quantities according to the required coordinate system. It should be noted that while the calculations of the different mechanisms may require the use of different coordinate systems, the field is always returned to the user defined coordinate system.

The total field as well as the total power are broken into the three reference coordinate systems for the output. The magnetic field is determined from the plane wave relationship between electric and magnetic fields. E_v is the electric field polarized in some arbitrary direction v and H_u is the magnetic field polarized in some arbitrary direction u. The field propagates in the direction r. The directions: v,u,r are mutually orthogonal and form the reference coordinate system.

$$\frac{E_v}{H_u}=\eta \tag{3.1}$$

The term η represents the intrinsic impedance of the medium. In most cases this will be free space. The intrinsic impedance of free space is $120\pi(\Omega)$. This relationship is used to determine the magnetic field strength as well as the power.

The code will send all the calculated field values to an output file. This file will be in text format and will contain the magnitude, phase, and polarization information as determined from the UTD field model.

3.2 Code Structure

The NECBSC code uses the following UTD mechanisms: direct rays, reflected rays, edge diffracted rays, corner diffracted rays, surface diffracted rays, and rays of the second order. The second order rays considered are those resulting from two reflections, a diffraction then a reflection, or a reflection and then a diffraction. Rays resulting from double diffraction are not considered by this code. The reason for this is that the code does not check for the transition region. The presence of a diffraction point within the transition region would cause invalid results. Therefore the code avoids the doubly diffracted rays.

The subroutines are responsible for the evaluation of the different UTD mechanisms. The main program will call the routines as the mechanisms are needed. The discussion of the routines will be done according to the UTD mechanism it is responsible for. Section 3.2.1 will discuss those routines responsible for the direct field. Section 3.2.2 will consider the mechanisms used for flat plates The section 3.2.3 will cover the routines used when elliptical cylinders are considered. The section after the surface diffraction routines, section 3.2.4, will discuss the service routines and the plate-cylinder interaction routines.

3.2.1 Direct Ray Routines

The direct ray is the ray that does not encounter any shadowing geometries. This ray will propagate directly from the source to the receiver. The routine involved in the direct field calculations is INCFLD. The routine was modified to allow the extraction of the ray path data as well as the ray magnitude. The ray path data will be placed in a file to be used later by the BSCV code.

3.2.2 Plate Field Routines

The UTD components available for the analysis of a flat plate are as follows: direct rays when the plate does not shadow the receiver, reflected rays from the plate surface, edge diffracted rays, and corner diffracted rays. For multiple plates the options also include: the field reflected from one plate and then reflected off a second plate and then reaching the receiver, the field being reflected off of one plate and then diffracted from another and then reaching the receiver, and finally a ray being diffracted from one plate and then being reflected from a second plate and then the ray will arrive at the receiver. The routine that handles the field reflected off of a plate is called REFPLA, it has been modified so that the ray path data and the information about the ray magnitude are extracted from it each time it is called. The routine that determines the diffracted fields from a plate is called DIFPLT. The routine DIFPLT has been modified so that both the ray path data and magnitude of the ray are extracted and placed within a file for later use. The routine that handles the field contribution due to doubly reflected rays is named RPLRPL. This routine has been modified like those above so that the ray magnitude and the ray path data are extracted. The routine DPLRPL handles the field contributions from rays diffracted from one plate and then reflected from a second plate. This routine underwent the same modification as those above in which the ray path data and the ray magnitude information was extracted and put into a file. The last of the routines for the flat plates is the one that handles the field contributions from rays that are reflected from one plate and then diffracted from a second plate. This routine is called RPLDPL and was modified so that the ray path data and the field magnitude data are extracted. The direct rays are determined from the output of the routine INCFLD which was mentioned in the section 3.2.1. The next section will discuss the routines that determine the UTD field contributions for the elliptical cylinders that are allowed by the code.

3.2.3 Cylinder Field Routines

The cylinder has the following UTD mechanisms: direct ray contributions, creeping waves caused by surface diffraction, field contributions due to endcap rim diffraction, fields reflected from an endcap, field contributions due to the reflection of the ray from the surface of the cylinder. If there are multiple cylinders the following UTD mechanisms are also included: field contributions due to the ray reflected from one cylinder and then scattered by a second cylinder and field contributions due to cylinder diffracted fields that are then reflected from a second cylinder.

The routines that are responsible for these UTD terms are now described. The UTD field due to shadowing from a cylinder surface is controlled by the routine SCTCYL. This routine has been modified so that the field magnitude and ray path data are extracted and written to a file. The field contributions due to endcap reflection is calculated by the routine REFCAP. This routine has been modified in the same way as SCTCYL. The next routine to be discussed is the one that computes the diffraction from the endcap of the cylinder. This routine is called ENDIF and has been modified in the same manner as the routines mentioned above. The routine that handles the effect of fields first reflected from one cylinder and then diffracted from a second cylinder is called RSCYL. This routine has been modified so that the ray path data and the ray magnitude is extracted and put into a file. The last routine to be mentioned is called DSCYL, this routine governs the determination of the contribution of the rays diffracted from one cylinder and then reflected from a second cylinder.

3.2.4 Service Routines

There are over forty service routines in version 2 of the NECBSC code. The bulk of them determine the position of diffraction points, reflection points or perform field calculations for the routines mentioned above. The diffraction coefficients and the effects of dielectric coatings are determined in the service routines. The mechanics for transforms of the coordinates and the repositioning of the geometries are also located within the service routines. These routines were essentially unaffected by the modifications of the NECBSC code and can be examined by consulting the NECBSC code manual.

The interaction of the plates and cylinders is not considered in the version 2 of the NECBSC code. This implies that the routines governing the effects of plate fields upon cylinders and cylinder fields upon the plates are not active. This removal of the higher order rays associated with such interactions may cause some discontinuities in the radiation pattern. These are caused by the missing mechanisms. The next chapter will discuss the BSCV viewer code that was written to supplement the NECBSC code [2].

4.0 The Basic Scattering Code Viewer

The Basic Scattering Code Viewer, BSCV, is a series of nested codes written in both MATLAB and FORTRAN languages. The viewer was designed to assist the users of the NECBSC code. The viewer has geometry preview features as well as ray tracing and radiation pattern plotting routines. The intent was to create an integrated package which would allow for a complete electromagnetic compatibility study of a NECBSC model.

The viewer runs from within the MATLAB environment. The code is independent of the operating system. The specifics of the viewer and the functions it performs will be discussed in detail in the following sections. Section 4.1 will discuss the routines that form the code and the operations that these routines perform. Section 4.2 will discuss the operation of the code and go through some examples of the output of the code. Section 4.3 will discuss future improvements of the code and possible additional features.

4.1 Structure of Basic Scattering Code Viewer

A flowchart highlighting the structure of the viewer is shown in Figure 4.1. The viewer is composed of two main components. The first component is the input geometry viewer which requires information from the input file of the NECBSC code. The second part of the viewer is the post processor. The post processor requires both the

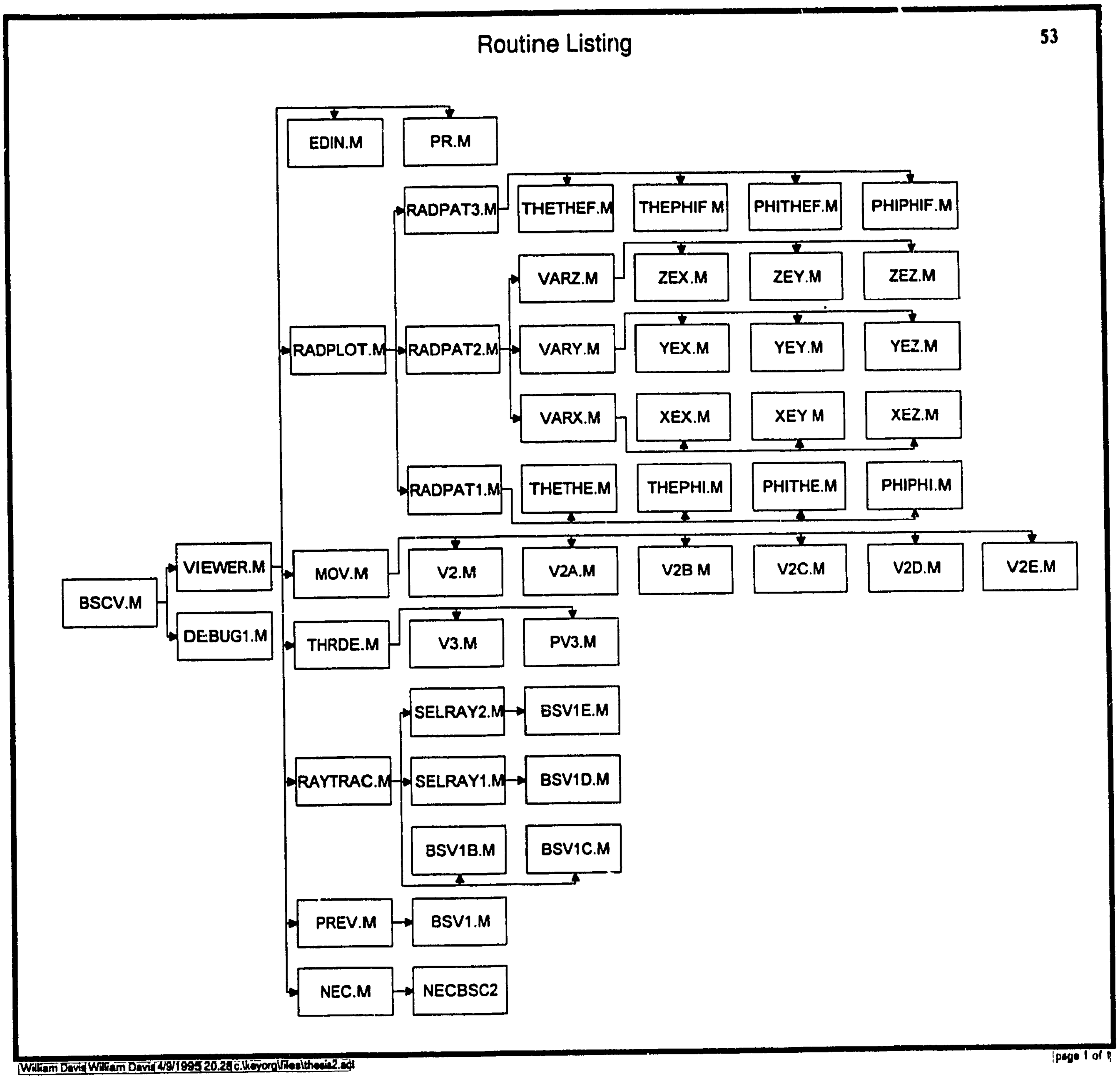


Figure 4.1 Flowchart for Routine Links

input files for NECBSC and ray data written out by from the modified NECBSC code. The post processor displays the ray tracing and allows radiation pattern plotting.

The actual code structure will be the next point to focus on. In the following sections the features and routines that compose the viewer will be highlighted. The MATLAB environment will be briefly discussed and the interface between the viewer and the NECBSC code will be explained.

4.1.1 BSCV Previewing Routines

The BSCV code may be broken down into those routines which are used by the previewer and those used by the post processor. The previewer is formed of a pair of codes. The pair consists of a FORTRAN code named bsv.f and a MATLAB code named bsv.m. The FORTRAN code **bsv.f** is used to process the input file of the NECBSC code into the proper format to be used by **bsvl.m.** The bsv1.m code uses the MATLAB environment to draw the geometry described in the input file for the NECBSC code. The input file has the name **inbsc.dat.** The **bsv.f** code generates a set of data files used in the drawing of the input geometry. These files will be discussed below as they are shared by both the previewer and the post processor.

4.1.2 BSCV Data Files

The post processor is a more complex section of the viewer. It requires both the input file **inbsc.dat** as well as information taken from the computations of the NECBSC code during the run. These files are as follows: **trace.dat**, **op1.dat**, **op2.dat**, **ray.dat**, **geommat.dat**, **coeff.dat**, **parts.dat**, **raymag.dat**, **raytyp.dat**, **ray3.dat**, **heading.dat** and **cylnmat.dat**. The description of what the files are used for will begin below.

The file **geommat.dat** contains the information used to generate the plots of the flat plates used in the model. The file **cylnmat.dat** contains the information required to draw the elliptical cylinders used in the model. The file **parts.dat** will indicate to the drawing program if there are cylinders, plates or both cylinders and plates in the model. The file **heading.dat** contains the information required for the placement and drawing of the sources used in the model. The file **ray.dat** contains the information about the paths that the different rays choose this information is a series of points in three dimensional space that allow for the tracing of the ray paths. The file **raytyp.dat** contains the information about the type of ray that is following the paths described in **ray.dat**. The file **raymag.dat** contains the information regarding the magnitude of the rays. The file **op1.dat** contains the information about the electric field as determined by the NECBSC code. The file **op2.dat** contains the information about the magnetic field as determined from the NECBSC code. The file **coeff.dat** will be used to convert from different units.

The standard input for this code is in meters, this will be adjustable later by setting coeff to the proper conversion factor.

4.1.3 BSCV Ray Tracing Routines

The routines are the next thing to be discussed. The viewer is controlled by a nested menu system. The routines are run without the user specifically naming them. The order of the description will be as follows. The first named will be those routines used to ray trace the different UTD mechanisms. The second set of routines will be those used to plot the radiation patterns. The third set of routines will be the service routines used for menus and other utility functions.

The routine that covers ray tracing with the ray type color coded is called **bsv1b.m.** This routine will generate an image of the model under study. Then the rays will be drawn in color according to their UTD mechanism type. The direct rays are drawn in yellow. The reflected rays are drawn in cyan. The edge diffracted rays are drawn in magenta. The corner diffracted rays are drawn in white. It should be noted that due to the method used in producing the color plots, the white rays were turned into black rays. This is a result of the reverse video mode used in printing with MATLAB, this mode will change the black areas to white and the white areas to black. This is only due to the nature of the printing option used and the rays will appear white on the computer screen.

The surface diffracted rays are drawn in white. The higher order rays are drawn in red. To allow for easy differentiation the cylinders are drawn in blue and the flat plates are drawn in green.

The second routine to be mentioned is the **bsv1d.m** routine. This routine will draw *selected* rays in the same format as **bsv1b.m** does. The user selects the UTD mechanism which will be drawn. The routine then proceeds to draw the rays of the selected UTD mechanism. Only the selected rays will be drawn so that a complicated ray picture can be broken down into less complex components.

The next two routines are those used to draw color coded rays according to magnitude. The first routine is the **bsv1c.m**. This routine will draw the rays with colors ranging from red, the highest energy ray, to violet, lowest energy ray. The spectrum will be displayed on the bottom right hand corner of the plot so that the user may identify the color of the ray with the respective magnitude of the ray. The geometry is drawn entirely in white so as to achieve the maximum contrast with the rays.

The second routine is the **bsv1e.m** routine. This will draw selected rays that are color coded according to magnitude. The picture format is the same as the routine above with the exception that only the selected UTD mechanism rays will be present.

The first set of routines allow the user to identify the UTD mechanisms contributing to the field. The second set of routines allow the user to determine the extent of the contribution from each ray. In combination this set of routines will allow the user to see the effect of each part of the model has on the total field calculated from the NECBSC code.

4.1.4 BSCV Plotting Routines

The next set of routines are those that control the plotting of the radiation patterns. These are broken down into those that handle the near and far field. The near and far field routines are further broken down into those with Cartesian and those with spherical coordinates.

The routines governing the far field radiation pattern plotting from the electric field are as follows: **thethef.m**, **thephif.m**, **phiphif.m**, **phithef.m**. The routine **thethef.m** control the plotting of the theta polarized electric field magnitude with respect to variations in theta. The routine **thephif.m** has control of the plotting of the theta polarized electric far field magnitude with respect to variations in phi. The routine **phiphif.m** allows the program to plot the phi polarized electric far field with respect to the variations in phi. The last routine **phithef.m** plots the phi polarized electric far field magnitude with respect to the variations in theta.

The near field electric field plotting can be done in either Cartesian or spherical coordinates. The Cartesian coordinate field plotting routines are: **xex.m**, **xey.m**, **xez.m**, **yex.m**, **yey.m**, **yez.m**, **zex.m**, **zey.m**, **zez.m**.

The routine **xex.m** generates the radiation pattern for the x polarized field as x varies. The routine **xey.m** controls the plotting of the x polarized electric field as y varies. The routine **xez.m** controls the plotting of the x polarized electric field as z varies. The routine **yex.m** controls the plotting of the y polarized electric field as x varies. In a similar way **yey.m** controls the plotting of the y polarized electric field as y varies. The routine **yez.m** will plot the y polarized electric field as z varies. The plotting of the z polarized field as x varies is controlled by the routine **zex.m**. The plotting of the z polarized electric field as y varies is the responsibility of the routine **zey.m**. The last routine is the one that controls the plotting of the z polarized electric field as z varies. The routine is called **zez.m**.

The last of the plotting routines are those that are used in the plotting of the near field in spherical coordinates. These are: **thethe.m**, **thephi.m**, **phiphi.m**, **phithe.m**.

The routine that controls the theta polarized electric field plotting with respect to variations in theta is called **thethe.m**. The plotting of the theta polarized electric field as phi varies is the responsibility of the **thephi.m**. The plotting of the phi polarized electric field as phi varies is controlled by the routine **phiphi.m**. The last routine, **phithe.m**, is the one that controls the plotting of phi polarized electric fields as theta varies.

4.1.5 Menu and Utility Routines

The next set of routines are those that control the menus and the utilities. The routine **bscv.m** starts the program and called up the first menu. The Main Menu is controlled by the routine called **viewer.m**. The routine that edits the input file is called **edin.m**. The routine that will print the graphics is **pr.m**, note that this routine allows the user to specify special printing options such as encapsulated postscript printing. The routine **prev.m** controls the previewer menu. The routine **ptrace.m** prints out data from the subroutine trace from the trace utility. The routine **pv3.m** prints out the images shown in the three dimensional view routine. The routine **radpat1.m** controls the menu for the near field spherical coordinates. The routine **radpat2.m** controls the menu for the near field Cartesian coordinates. The routine **radpat3.m** controls the menu for the far field plotting routines. The routine **radplot.m** controls the menu selecting which radiation pattern type, spherical or rectangular coordinates used to generate the pattern, is required. The routine **raytrac.m** controls the menu for the ray tracing selection. The routine **selray1.m** controls the menu for plotting selected rays color coded according to UTD mechanism. The routine **selray2.m** controls the menu for plotting selected rays color coded according to magnitude. the routine **thrde.m** controls the menu that allows for the three main views of the geometry. These views are the top view, side view and the front view. The routine **vtrace.m** allows the user to view the trace file. The trace file traces the subroutines used in the NECBSC code and what output those subroutines produced. The routines **v2.m**, **v2a.m**, **...**, **v2e.m** are all routines that generate a movie via animated views, by rotating the geometry. They are called by the routine **mov.m** which is the menu routine for the program. The routine **v4.m** is the main viewer routine. This routine generates the slider menus which allow the user to rotate and zoom within the image generated. This routine is the controller for the submenus used in the other routines. The last routine to be mentioned is the nec.m. This routine will execute the NECBSC code so that the viewer may perform the post processing functions for the code user.

The flowchart in Figure 4.1 represents the links between the routines mentioned in the preceding sections. The listing shows the routine name and the routine or routines that is calls. Any file that has the format*filename.m* is a MATLAB file.

4.2 Operation of the Viewer

The viewer uses a series of menus which allow the user of the code to navigate through the available features. A flowchart showing the organization of the viewer is shown in Figure 4.3. The examples in this section are based on a single flat plate and a dipole source.

The user of the code must be within the MATLAB environment to start the operation of the code. The following programs must be present and compiled: the modified version of the necbsc.f, prepro.f, trace.f and bsv.f. The compilation must be done with FORTRAN. The files mentioned in section 4.1.1 to 4.1.5 must also be present.

The user will start the viewer from within MATLAB by typing the following command: "bscv". Then hit the enter key. The following menu, shown in Figure 4.2, should appear along with a Figure box that will begin empty.

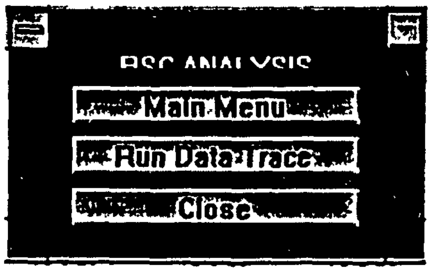


Figure 4.2 Starting Menu

BSCV Viewer Control Diagram

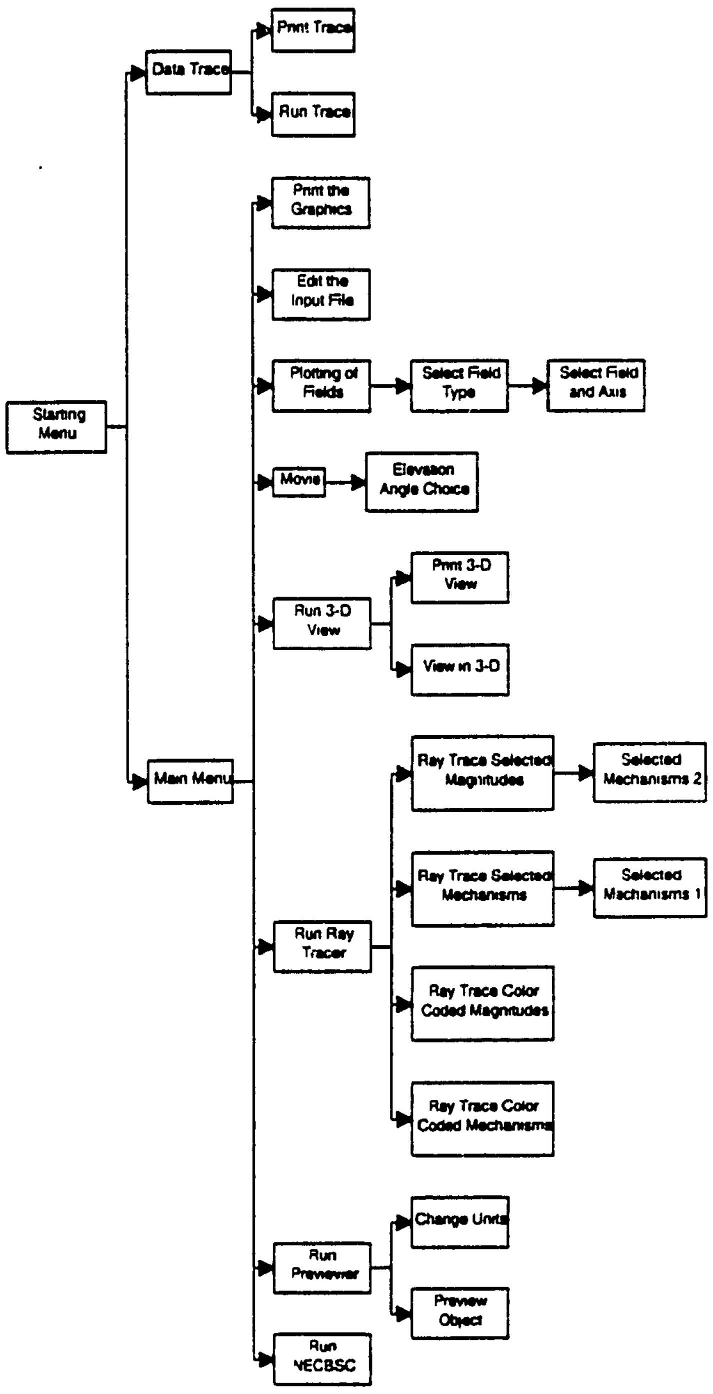


Figure 4.3 BSCV Control Diagram Flowchart

The person using the code would then have the choice of selecting the previewer and post processor or the data trace. The previewer is used for the model development, the trace is used for debugging the code or testing for problems with the diffraction mechanisms. The next menu that the user would encounter if the previewer button was selected is shown in Figure 4.4, below.

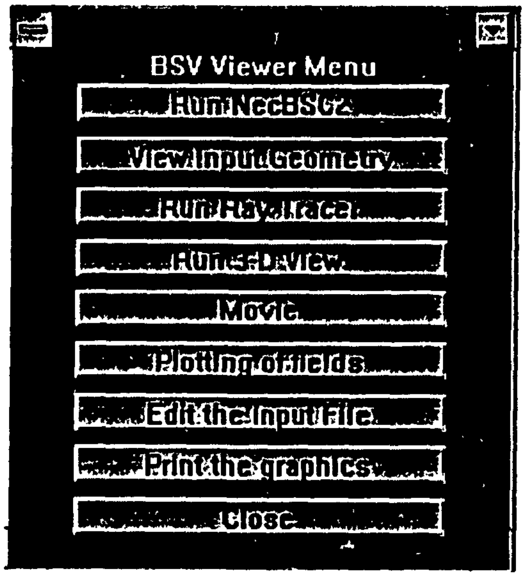


Figure 4.4 Main Menu for BSCV

This menu lists the features available for the code user. The code user will be able to cycle through the options by clicking the appropriate buttons. We shall discuss the features as we progress down the menu.

The first feature is the Run NECBSC. This will cause the NECBSC code to execute a computation of the fields present for the geometry specified within the input file. This command exits the MATLAB environment to execute the instruction and then returns control to the user of the code after the NECBSC code has completed the run.

The next option is the previewer option. This will generate a three dimensional image of the geometry under study. This option is very helpful for checking the model before the user runs the NECBSC code. The user may rotate the image or zoom in and out. This allows the user to view the model from all angles and as close as the user desires. The image may be printed as part of a presentation or as a hard copy for reference. The menu for the previewer option is shown in the Figure 4.5. This menu will appear when the run previewer option is selected from the main menu. The option to change units is used to rescale the image axis without changing the geometry. When used it will allow the user to set the scale in inches, meters, or wavelengths. The default is meters.

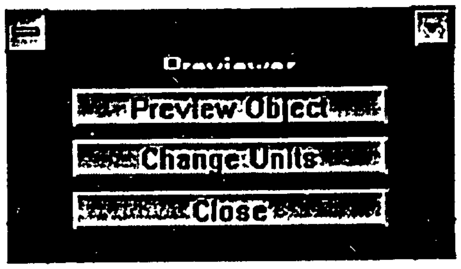


Figure 4.5 Previewer Menu

The previewer option will allow the user of the code to view the image of the model on the screen. The image will be in three dimensions. The parts of the model that are flat plates will be colored green. The elliptical cylinders will be colored blue.

The next option will be that of the 3-d view, the menu shown in Figure 4.6. The option will draw the top view, the side view and the front view of the image under study. When selected the following menu will appear. The user will be prompted to select either to have the three views printed or displayed on the monitor. The output from such a command was shown in Figures 4.8 and 4.9.

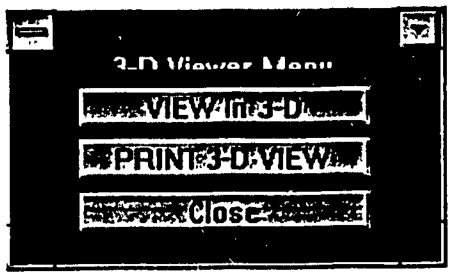


Figure 4.6 3-D View Submenu

The image shown in Figure 4.7 is the model of flat rectangular plate with a dipole above the plate, this model will be used for most of the examples of this section. The plate is a 1 meter by 1 meter square flat plate. The plate is located on the X-Y axis. The dipole is a z directed dipole of length 0.3 meters. The wavelength is 1 meter at the frequency that this model was analyzed at.

The images of Figures 4.7, 4.8 and 4.9 were created using the previewer section and 3-d view section. The NECBSC code was not used in the generation of this image. The plates are shown in green. The color coding is arbitrary and may be changed if the user so desires. The view may also be changed to three fixed views. The top view, the side view, and the front view. These are chosen from the menu option run 3-D view from the main menu. The Figures 4.8 and 4.9 represent two of these views. The next three pages will contain the preview images mentioned above. While the view in Figure 4.7 corresponds to the oblique view, the other three views are formed from viewing the object from one of the three Cartesian axis. That implies that the top view is the image seen from the positive z-axis looking down.

An important feature of the modeling using the NECBSC code is that the structures must be constructed using elliptic cylinders and flat plates. The structures are treated as infinitesimally thin structures in the case of the flat plates. This means that the plate has no thickness. The plate is not visible when looking at it edge on. This property can be noticed in the Figure 4.9 where the side view shows only the antenna. The plate is not visible from that view.

This implies that care must be taken when choosing the views since certain elements of the geometry will be difficult to view from certain angles, looking at plates edge on for example.

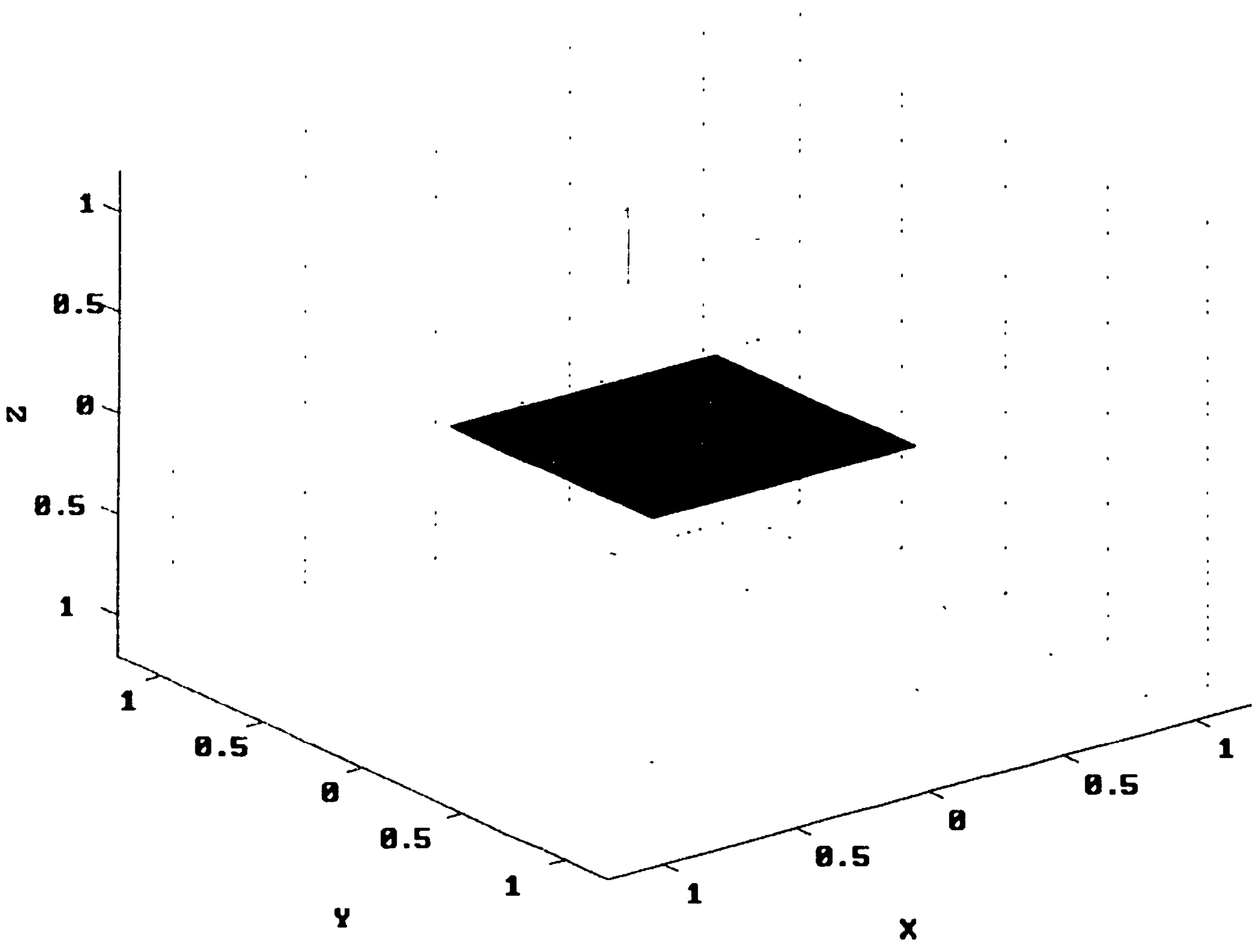


Figure 4.7 Image generated from preview option

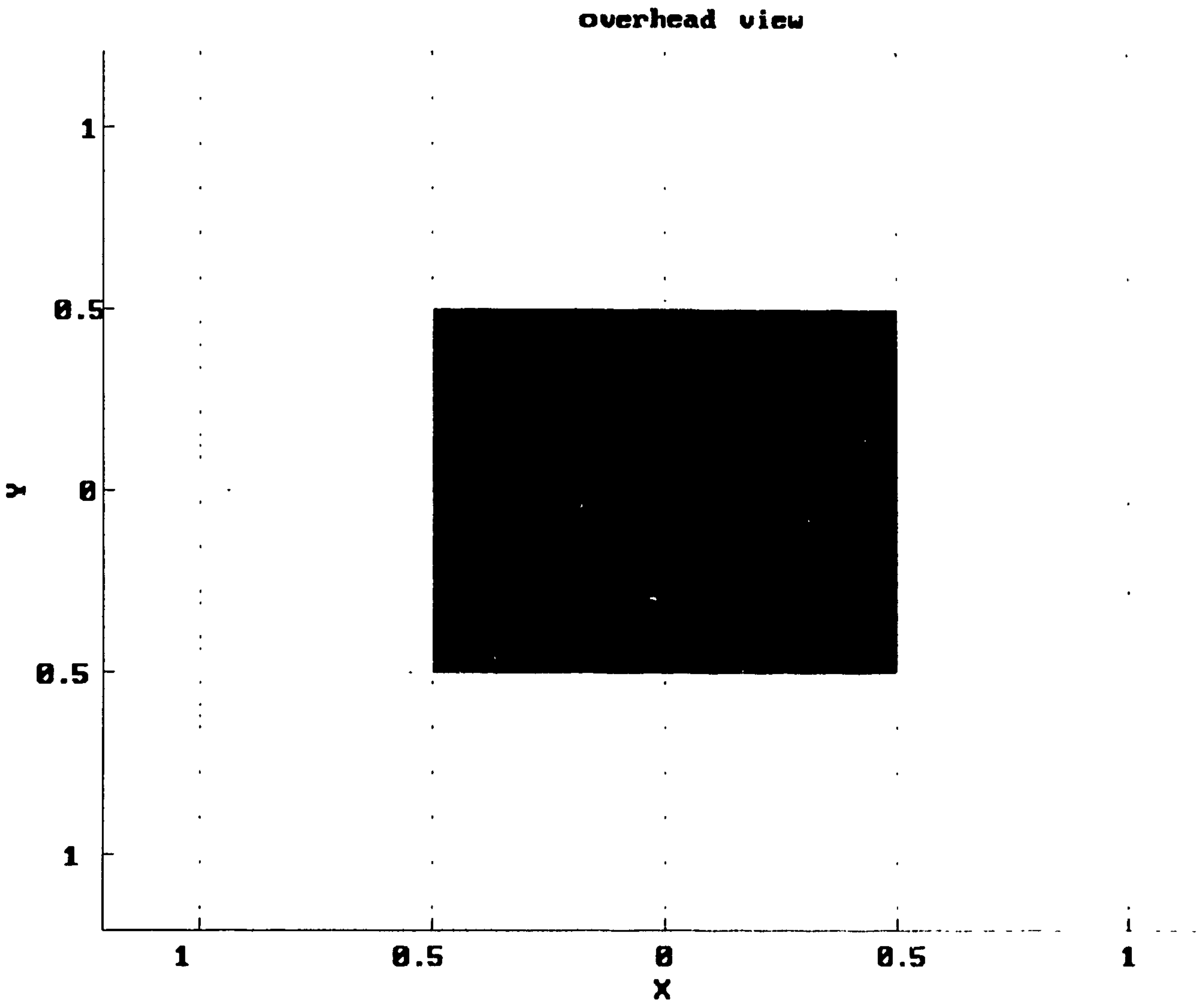


Figure 4.8 Image generated from 3-D view option using overhead view

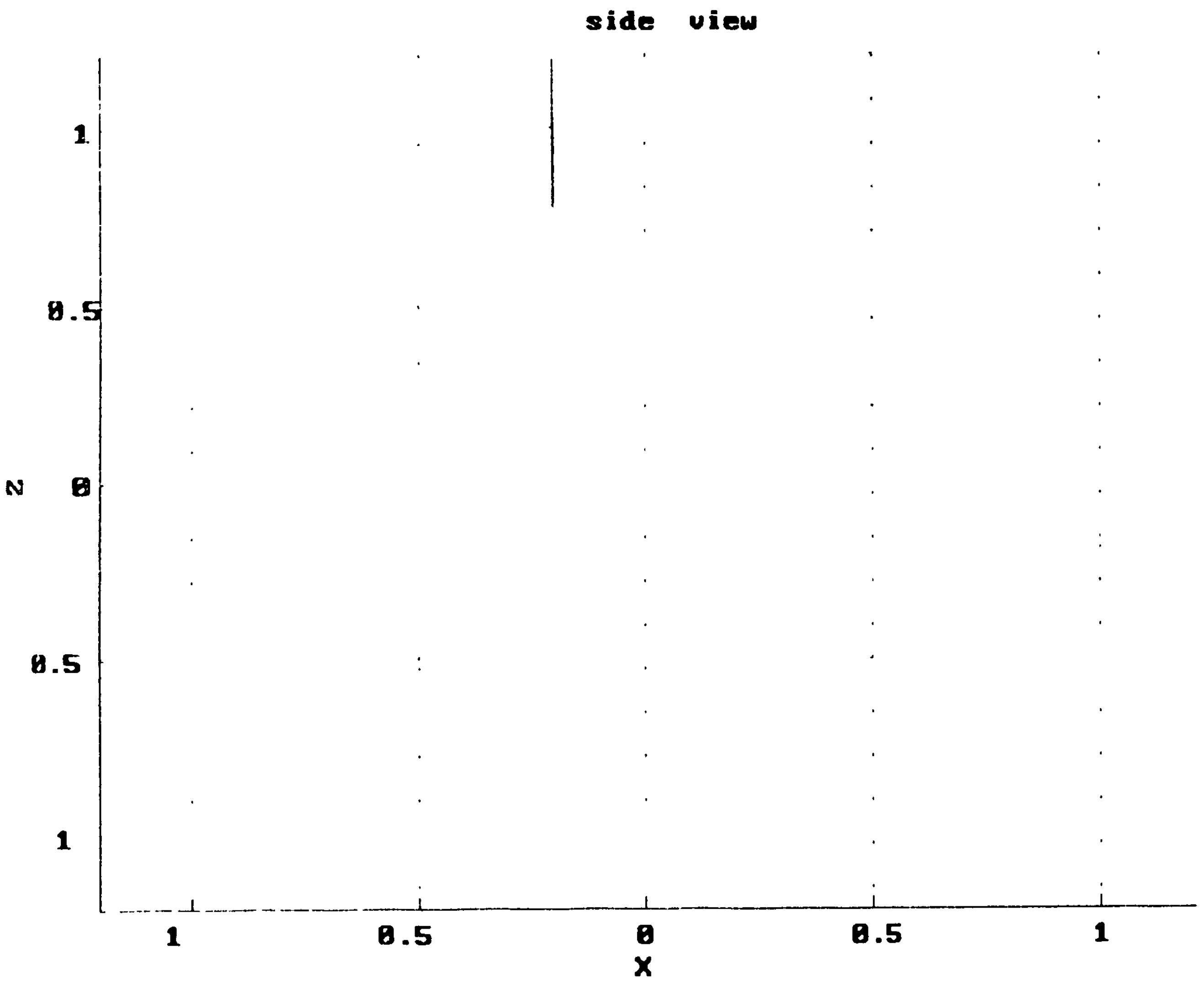


Figure 4.9 Image generated from 3-D view side view option

The next feature to be discussed is the run ray tracer option. This option requires information taken from the output of the NECBSC code. Therefore this option must be selected only after the run NECBSC option has been chosen. Note that once the run NECBSC option has been selected, the data files will remain until a new NECBSC run is called. Therefore the run NECBSC option is called only once for a particular observation point. The selection of the ray tracing menu option will present the code user with another menu, shown in Figure 4.10. This is the main menu of the ray tracing section of the code.

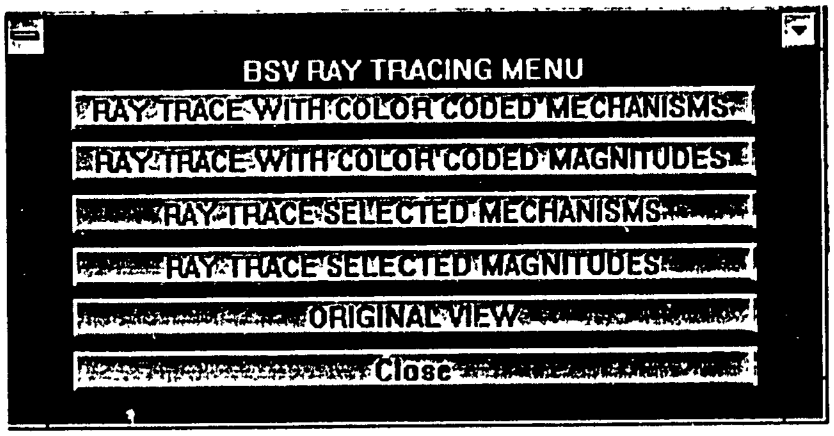

BSV RAY TRACING MENU
RAY TRACE WITH COLOR CODED MECHANISMS
RAY TRACE WITH COLOR CODED MAGNITUDES
RAY TRACE SELECTED MECHANISMS
RAY TRACE SELECTED MAGNITUDES
ORIGINAL VIEW
Close

Figure 4.10 Ray Tracing Main Menu

From the menu in Figure 4.10, the user can choose to trace the rays with the ray colors coded to the UTD ray mechanism or to the magnitude of the ray. The user may choose to draw all the rays or only selected rays. The first option is that of drawing rays color coded according to mechanism. Figure 4.11 shows the image of the flat plate and source with the observation point above and in front of the of the plate which is determined by the Cartesian coordinates (1, 1, 1.5). The rays present are as follows: The direct ray is drawn in yellow. The edge diffracted rays are drawn in magenta. The reflected ray is drawn in cyan. The corner diffracted rays are drawn in white on a crt, but in black on a hardcopy. The following rays are not present, creeping waves and higher order rays. If they were present their color coding would be as follows. The creeping waves are drawn in white. The higher order rays are drawn in red.

If the second option is chosen then the image in Figure 4.12 will be presented to the viewer. In this case the rays are drawn according to their magnitude. The rays with the higher strength will be on the red side of the spectrum. The weaker rays will move along the color scale, shown on the bottom right hand corner, until violet is reached. The strengths are based upon relative magnitudes. The magnitude of the ray is measured in decibels with the reference level taken as 1 Volt/meter field strength.

The model under consideration is the same as in Figure 4.11. The strength of the rays and the GTD mechanism of the rays maybe identified by using the two forms of ray tracing. For example, the direct ray is identified in the Figure 4.11 as the yellow ray emanating from the source and propagating towards the observation point. The strength of the ray can be determined from the image in Figure 4.12. The direct ray is seen to be deep red, identifying it as the strong ray of magnitude 23.07 dB. The scale is divided into fifty segments of equal size. The range is given on the top of the bar, from this the magnitude of each ray may be determined. The color coded magnitude image shows that certain rays are dominant in this particular geometry. The yellow rays are much stronger than the blue rays as can be seen from the magnitude bar. The color coded mechanism plot allows the user of the code to single out the mechanism of interest.

The user of the viewer is also given the option of drawing of selected rays from the menu in Figure 4.10. The option to draw selected mechanisms allows the code user to select a single UTD mechanism and draw only the rays associated with that mechanism. When the option to draw color coded mechanisms is selected, the user will be presented with the set of options listed in Figure 4.13. The user may then select which one of the UTD mechanisms will be drawn. For this particular feature, all of the rays will be drawn in red.

The user also has the option to draw rays from a selected UTD ray type according to magnitude. This option will draw the selected ray type with color coding according to the criterion set up in the option number two. This will allow the user to see which ray type is contributing the most to the total field. The menu for this option is shown in Figure 4.14.

The menu with the title *Selected Mechanisms 1* from Figure 4.13 will draw the selected ray mechanism, reflected rays for example, in red. Only the selected rays will be drawn. The other menu, seen in Figure 4.14, will draw the rays which are selected from the list on the menu. These rays will be color coded according to the relative magnitude of the rays.

The output for these options is given in the Figures 4.15 and 4.16 for the color coded mechanisms and color coded ray magnitudes respectively. The observation point is in the near field of the plate. In the Figure 4.15, the only GTD mechanism present is the direct ray. The direct ray from the source to the observation point is drawn in red. The antenna that is transmitting is in the same position as in the previous examples. The image in Figure 4.15 is focusing on the direct ray emanating from the source and striking the receiver. The image of Figure 4.16 shows the direct ray, path and magnitude, from source to receiver.

The menu shown in Figure 4.10 was used to generate the images seen in Figures 4.11 and 4.12. The menus shown in Figures 4.13 and 4.14 were used to generate the plots in Figures 4.15 and 4.16 respectively.

These images present the ray tracing and field strength information that was generated from the NECBSC2 run. The information presented in these pictures is extracted from the NECBSC2 runs without any changes to the operation of the code. This means that the field computations done by the NECBSC2 code are not interrupted by these features. It also means that the images give a direct view of the operation of the code.

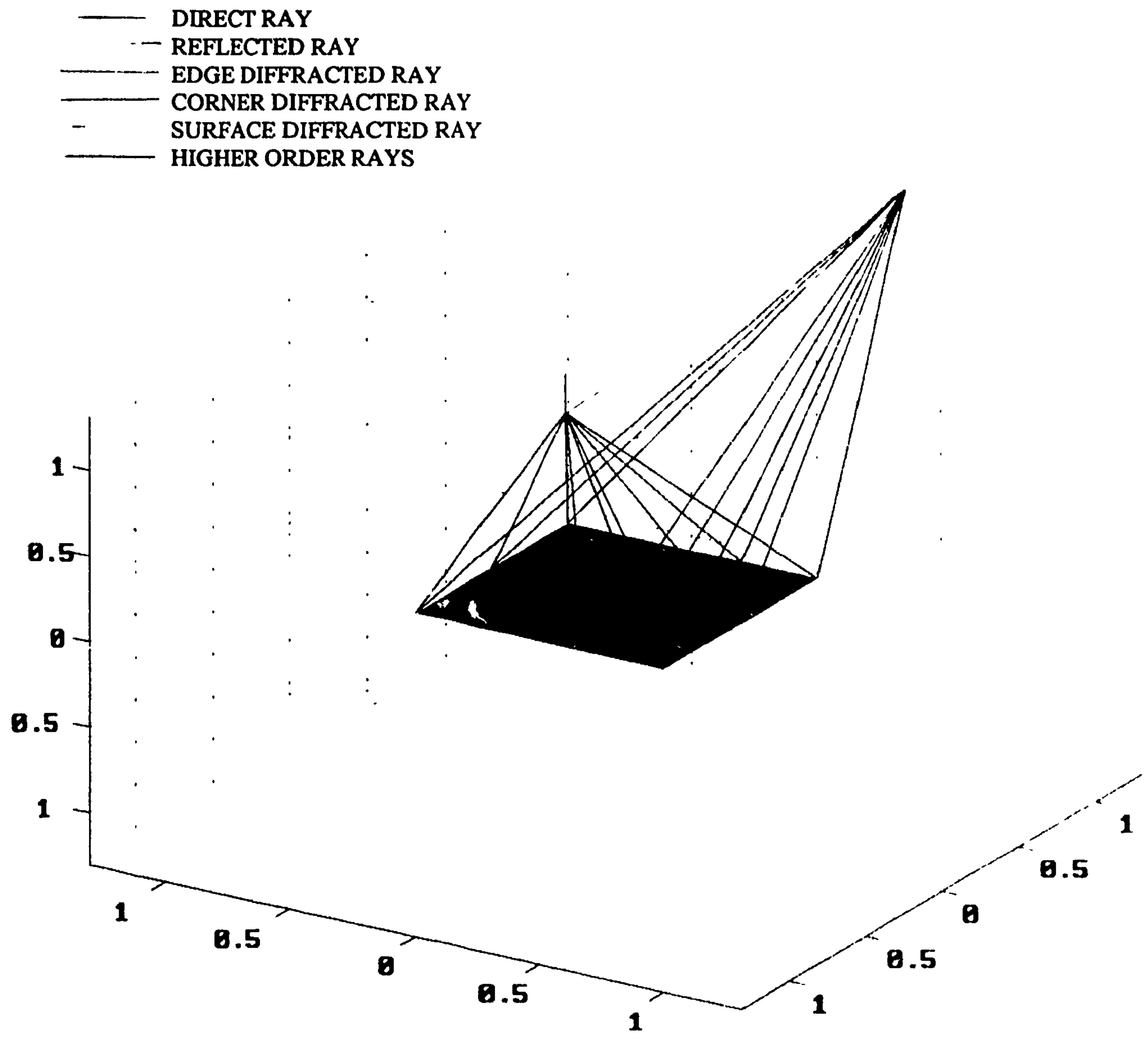


Figure 4.11 BSCV ray tracing output for color coded mechanisms

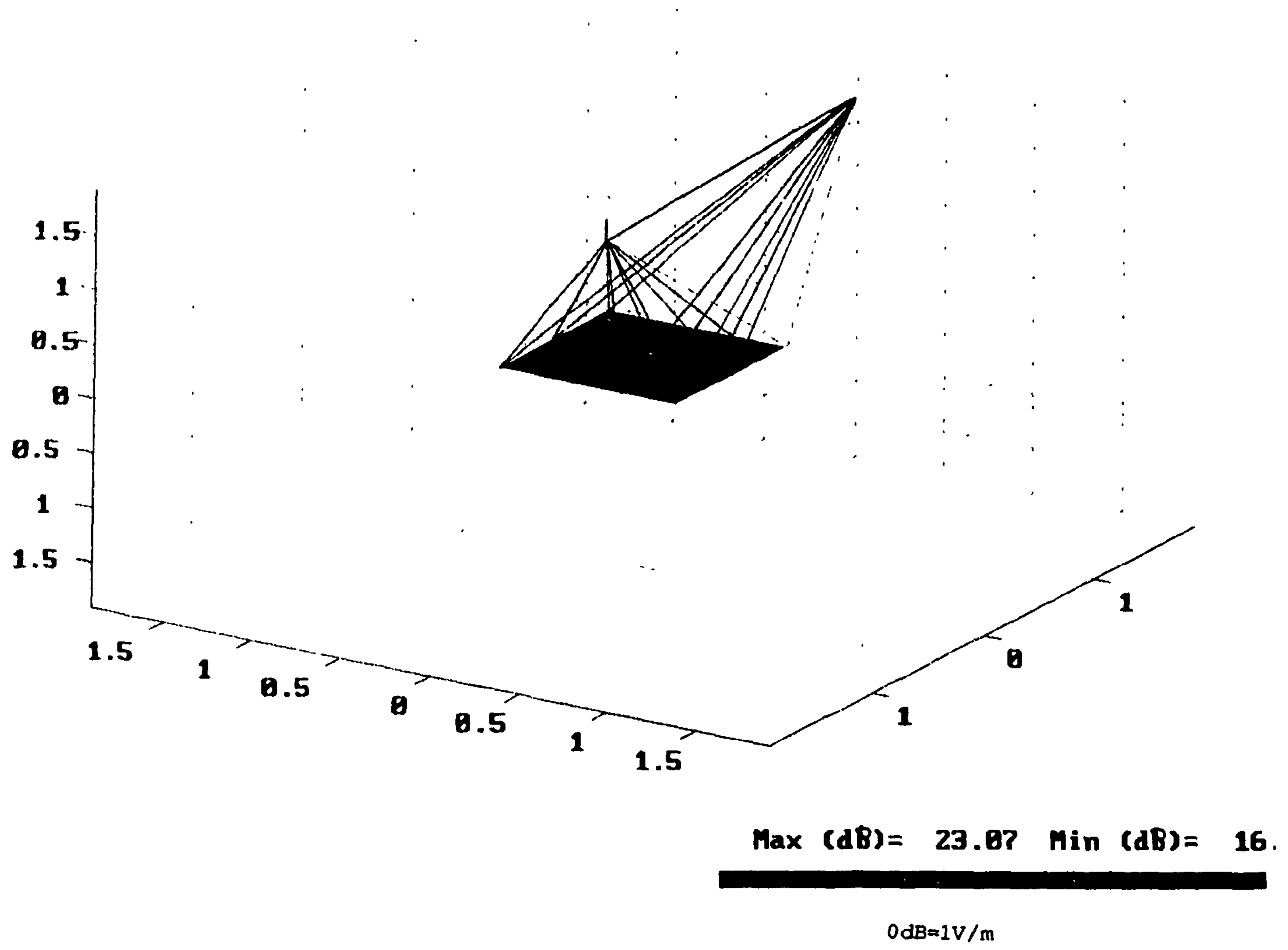


Figure 4.12 BSCV ray tracing output for color coded magnitudes

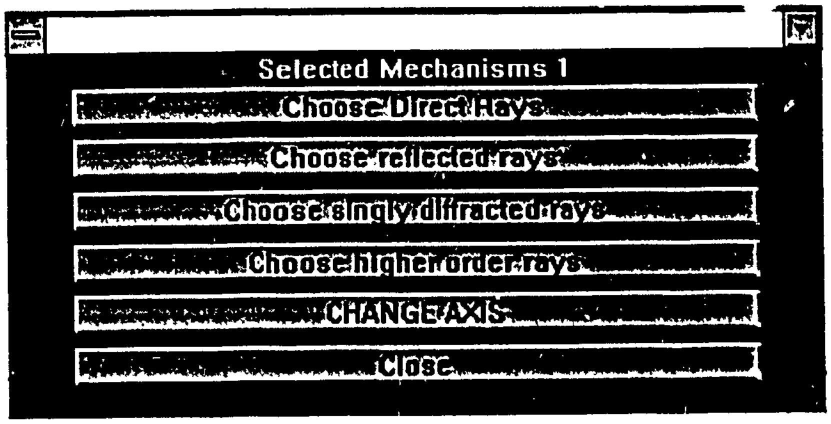


Figure 4.13 Color Coded UTD Mechanism Menu 2

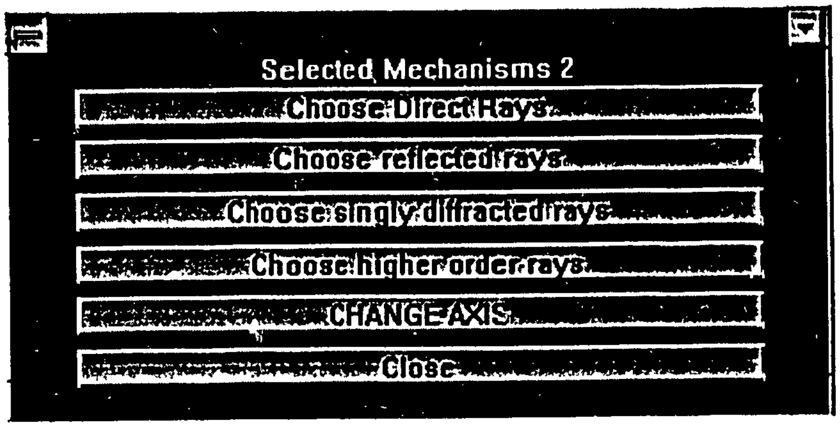


Figure 4.14 Magnitude Color Coded Menu 2

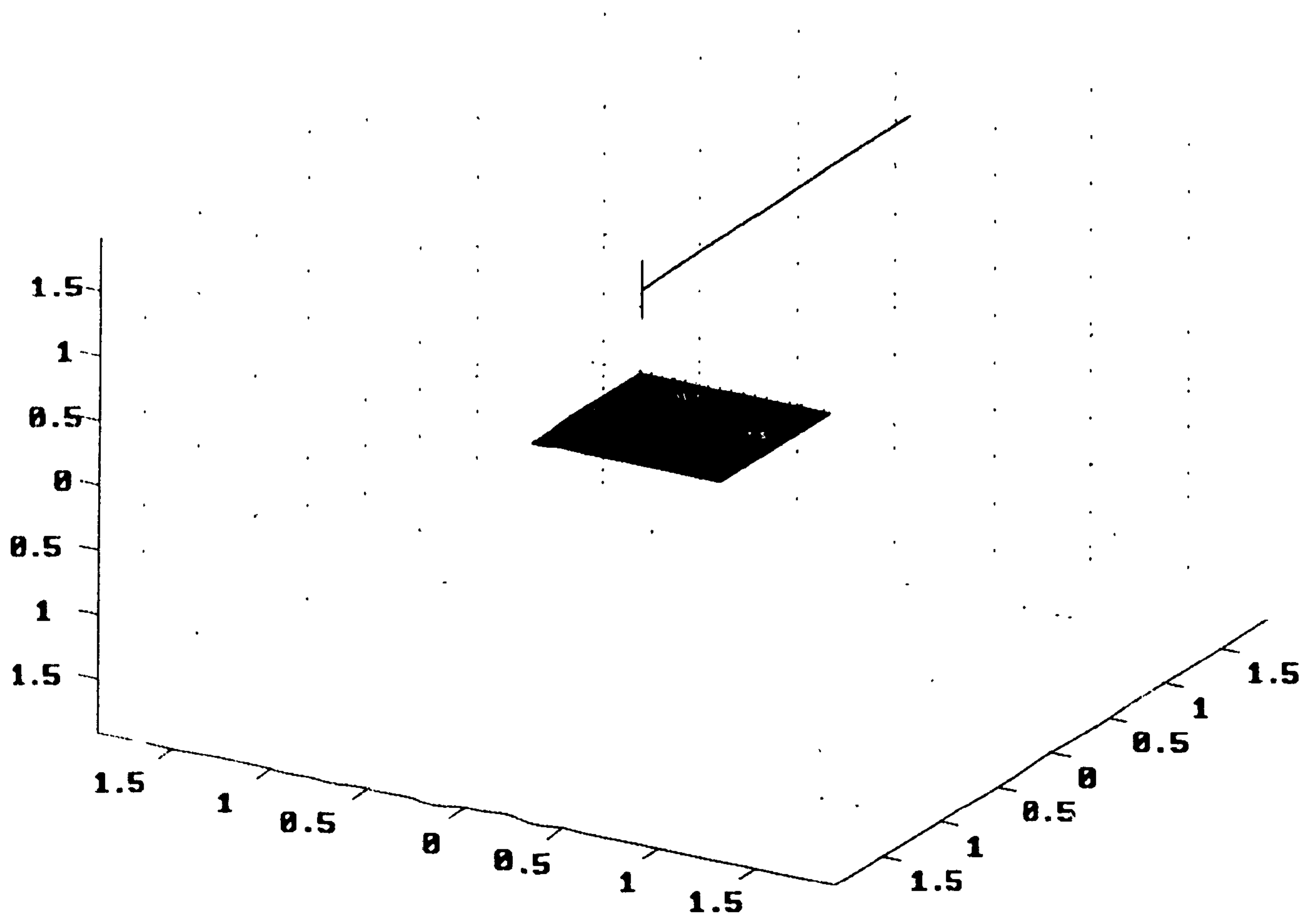


Figure 4.15 BSCV ray tracing output for selected ray mechanisms color coded mech.

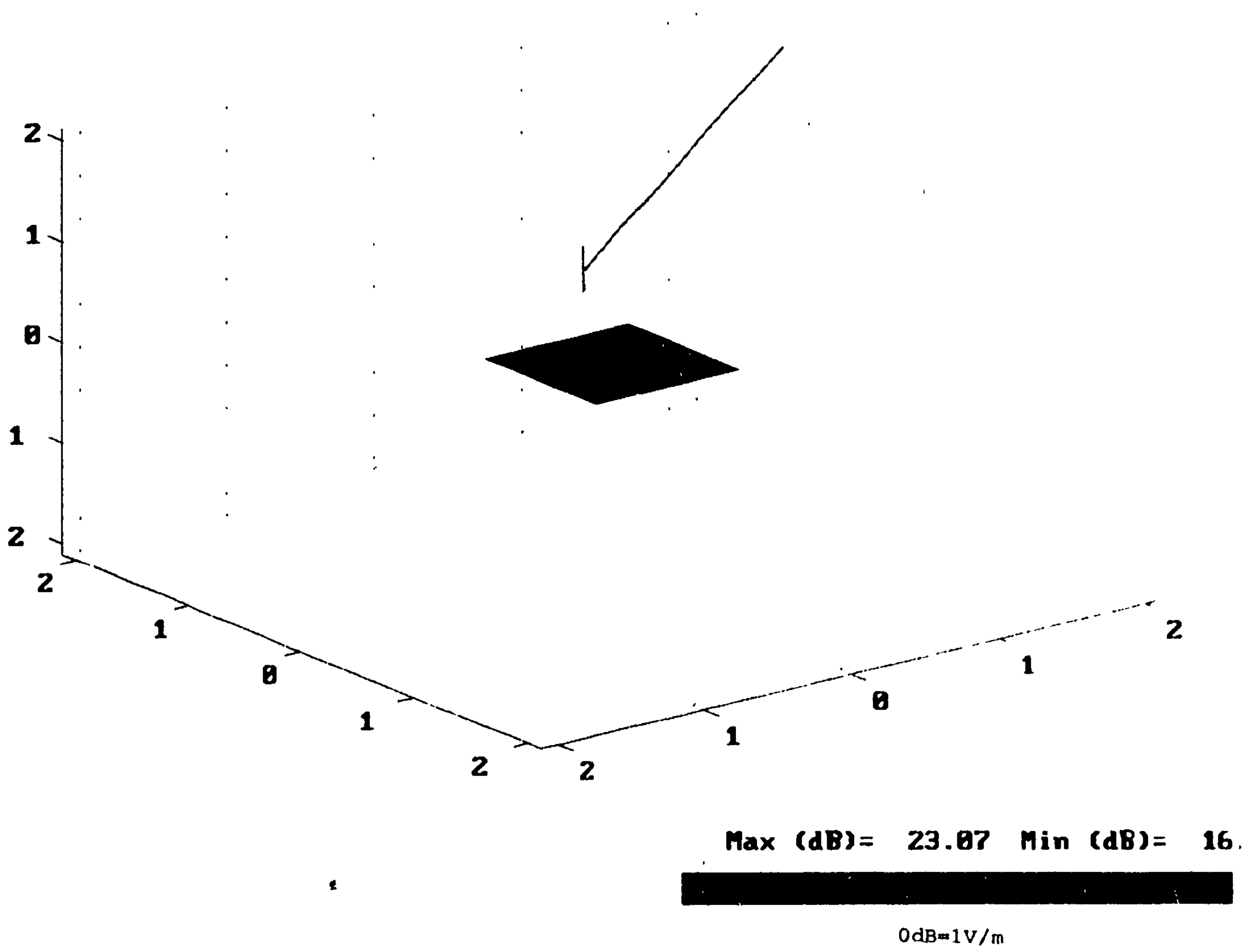


Figure 4.16 BSCV ray tracing output for selected rays color coded magnitudes

Another useful way of obtaining 3-D views is the Movie option. This will present an animated series of images that will allow the user of the code to view the object from any angle. The animation feature can be called on when one of the other ray tracing or previewing options has been called previously. The menu for the movie feature is shown in the Figure 4.17.

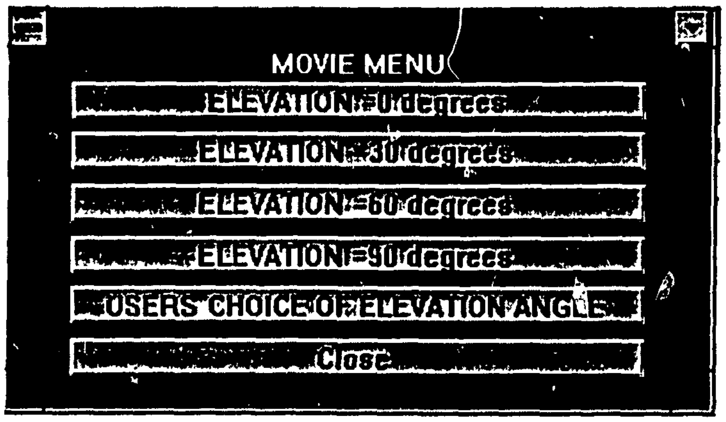


Figure 4.17 The Movie SubMenu

4.3 Examples of Viewer Operation

In this section a few sample geometries will be analyzed using some of the features discussed in section 4.2. The first example will be the flat plate seen earlier in this chapter. The second example will be an elliptic cylinder with a flat plate attached to the side of the cylinder.

4.3.1 Flat Plate Radiation Pattern

The next feature to be considered is the plotting of fields. This option is called from the main menu. When it is called the user is given a choice of what field type to view. The first submenu is shown in the Figure 4.18. It should be noted that these options are for the plotting of the electric field. There are a parallel series of menus added for the plotting of the magnetic field radiation patterns.

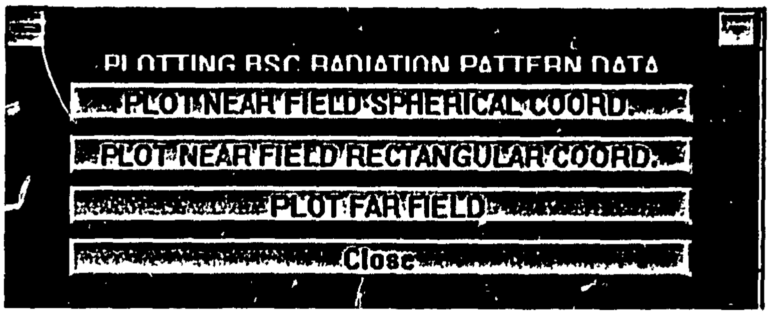

PLOTTING BSC RADIATION PATTERN DATA
PLOT NEAR FIELD SPHERICAL COORD.
PLOT NEAR FIELD RECTANGULAR COORD.
PLOT FAR FIELD
Close

Figure 4.18 Radiation Pattern Plotting Submenu 1

The options are for the three types of fields allowed by the NECBSC code. The next series of submenus govern the different possibilities for plotting the field. To illustrate the procedure a plot of the theta polarized electric field versus variations in the elevation angle theta will be made. Since this is a near field plot in spherical coordinates, the first choice will be selected from the menu above. The user of the code will be then given a further set of choices from the second level submenu shown in Figure 4.19. The user of the viewer should note that the plotting feature requires the user to generate the data from the NECBSC run before selecting this feature. Therefore if the user of the code wants to plot near field with theta varying, the input file of the NECBSC code must contain the correct instructions for the code to generate the required data.

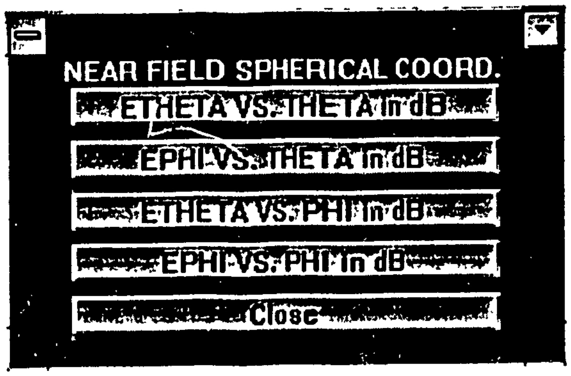


Figure 4.19 Radiation Pattern Plotting SubMenu 2

Since the field pattern of interest is the theta variation versus the theta polarized electric field. The selection will be theta versus etheta. When this choice is made the plot will be sent to the screen. An example of the radiation pattern plot can be seen in Figure 4.20.

The flat plate seen earlier in this chapter is the focus of the first example. The NECBSC2 code was used to calculaıe the radiation pattern in the near field about the plate shown in Figure 4.5. The pattern was for the theta polarized electric field with theta varying. The pattern was calculated for the GO field from equations 2.13 -2.15, the field using the GO with edge diffraction terms added as in equations 2.16-2.18, and with the full range of possible UTD mechanisms used by the code as in equation 2.73.

The radiation patterns are shown in the Figure 4.20. The GO field has abrupt discontinuities caused by the disappearance of the reflected ray when theta is around 42 degrees and the disappearance of the direct ray as theta approaches 138 degrees. The pattern is somewhat smoothed out put the use of the edge diffraction compensating for the loss of the GO rays. The discontinuities in the GO + edge diffraction curve are caused by the appearance and disappearance of two of the edge diffracted rays. The images shown in Figures 4.21, 4.22 and 4.23 show the code user what is occurring as the observer approaches and then passes the incident shadow boundary of the plate. It should be noted that for the images shown in 4.21-4.23 the NECBSC2 code used only GO and edge diffracted terms. Since corner diffraction was not used in the calculation, there will be no corner diffraction in the viewer images. This was done to better illustrate the effects of edge diffraction upon the loss of the direct ray.

The image shown in Figure 4.21 shows the direct ray in red, being the strongest ray, and the two edge diffracted rays in blue and magenta. This image has the observer positioned at a point before the incident shadow boundary is crossed.

The next image, Figure 4.22, shows the observation point closer to the incident shadow boundary, the direct ray is still present but the edge diffracted ray from the closest edge has increased in intensity, changing from blue in Figure 4.21 to green in Figure 4.22. This shows how the edge diffracted ray will compensate for the loss of the direct ray by taking over when it disappears.

In Figure 4.23 the observation point is past the incident shadow boundary. Here the edge diffracted ray that was increasing in strength in Figures 4.21 and 4.22 has now become the dominant ray. This image also shows the appearance of two new edge diffracted rays. The appearance of these rays caused the discontinuity for the GO + edge diffraction plot shown in the Figure 4.20. The corner diffracted rays, which were used in the plot represented with the dotted line, compensate for the sudden appearance or disappearance of the edge diffracted rays and smoothed out the plot for the regions where the discontinuity was caused by this effect. These regions were at theta equals 40 degrees and theta equals 140 degrees.

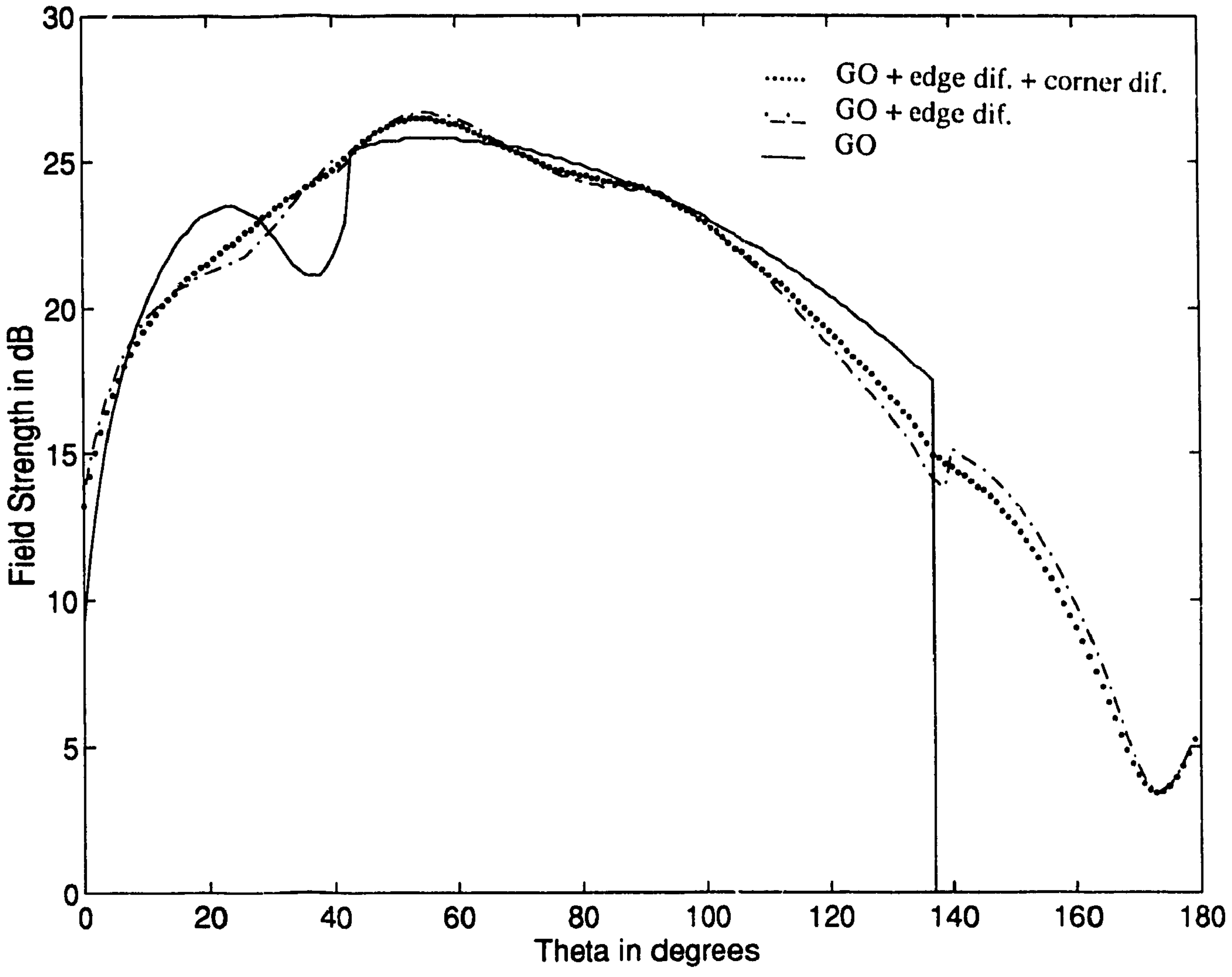


Figure 4.20 Radiation Pattern Comparison for GO and GO + UTD

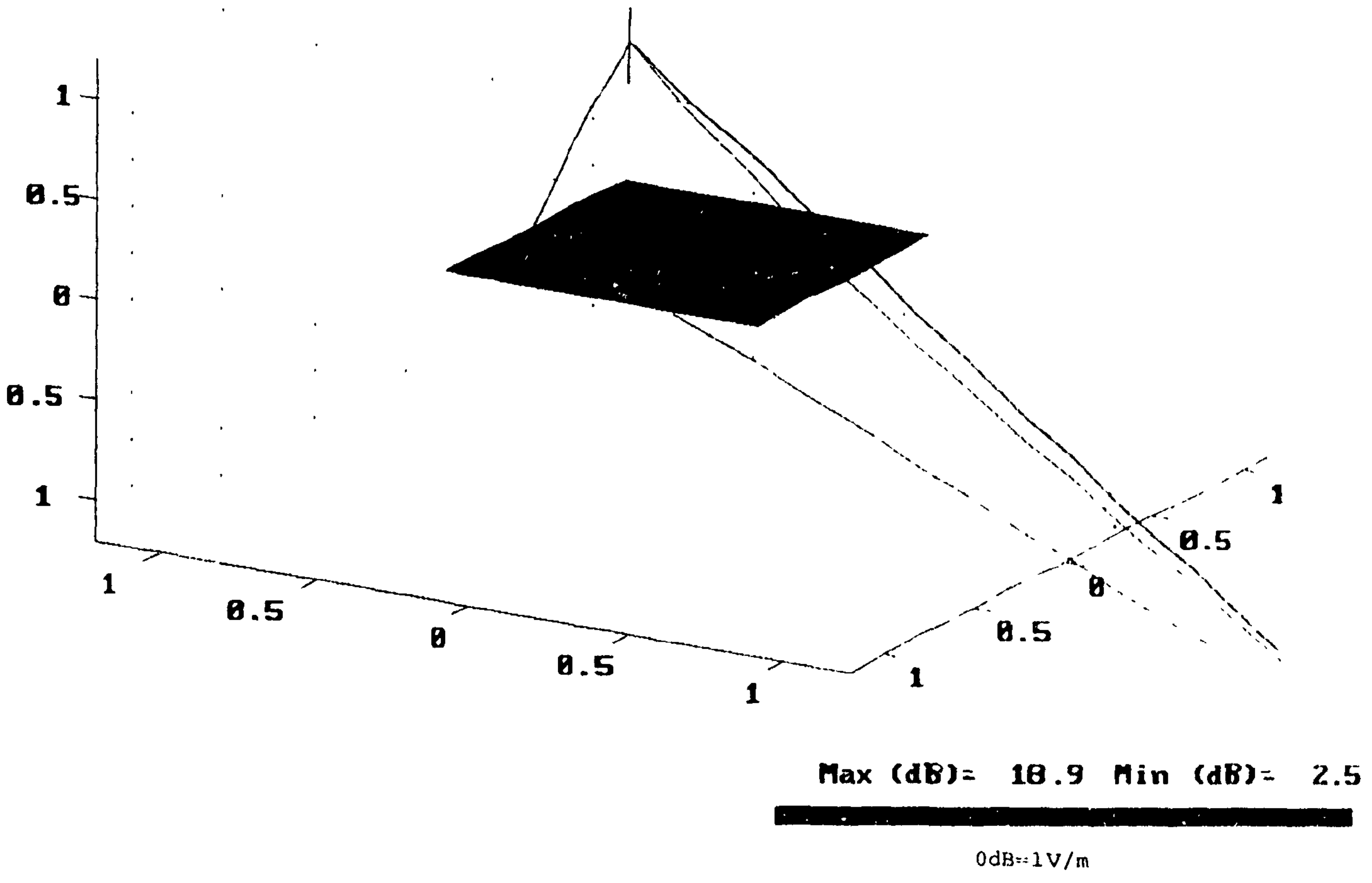


Figure 4.21 GO + edge diffracted rays well before ISB (0 dB = 1 V/m)

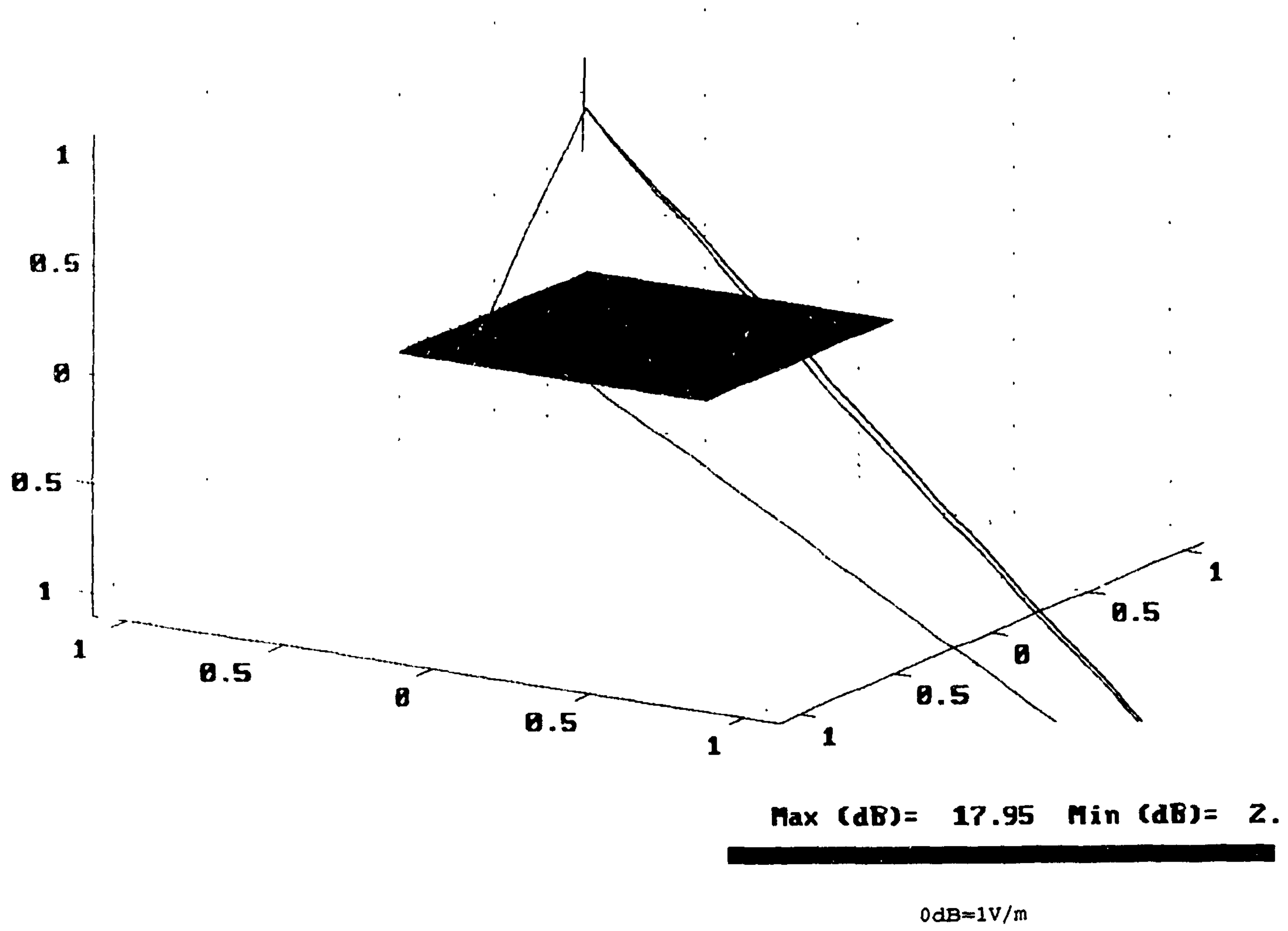


Figure 4.22 GO + edge diffracted rays immediately before ISB

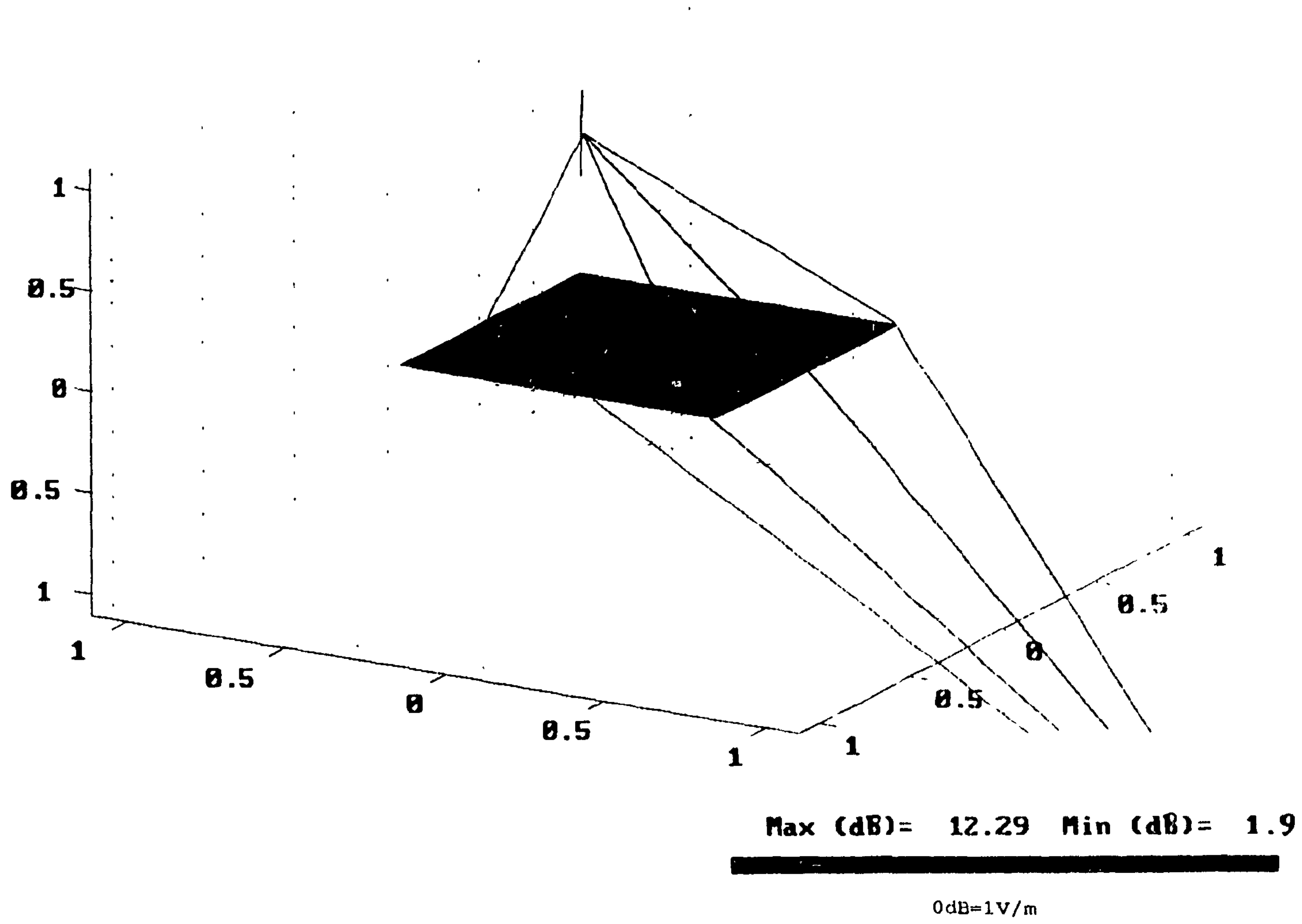


Figure 4.23 GO + edge diffracted rays after ISB

4.3.2 Flat Plate and Elliptic Cylinder

A second example will show a problem area of the code that was brought to light by the viewer. The plate-cylinder interactions that exist in reality are not handled by this version of the code. The viewer was given an input file that contained a cylinder with a flat plate attached to it. The code then performed the UTD analysis for a single observation point located below the cylinder. The plate was checked and seen to be inserted within the cylinder.

The results of the run can be seen in the Figures 4.24 and 4.25. One of the creeping waves from the cylinder can be seen to pass though the flat plate. In the Figure 4.24 we see the geometry from an overhead view. In this case we should look at the cyan ray, creeping wave, which is heading though the plate. To confirm that the creeping wave does indeed pass through the plate, a second image was used. Figure 4.25 shows that the creeping waves are both unimpeded in their progress.

This implies that the code does not accurately model the interactions between plates and cylinders. This would also indicate that the viewer can assist in locating and identifying trouble with either the model or how the code analyzes the model.

In the first example, given in section 4.3.1, the viewer was used to identify the specific UTD terms are used in the construction of the total field at the observation point. The viewer was also used to show how the individual mechanisms contribute to the total field. This allows the code user to associate the UTD terms with both the geometry and the field pattern.

The second example, given in section 4.3.2, the viewer was used to illustrate the lack of plate-cylinder interactions that is a failing of this version of the code. This shows how the viewer can be used as a tool for the analysis of both the model as well as the performance and operation of the NECBSC2 code. This will also allow the user to determine if the model and modeling is valid for the study in question.

The next chapter will discuss the use of the BSCV code in an electromagnetic compatibility study. The study detected an anomaly in the radiation pattern which was not a physical phenomenon. This implied that the code was in error at some point in the radiation pattern. The BSCV code viewer was developed in part to determine if it was an error that could be detected from ray tracing the paths that the code used.

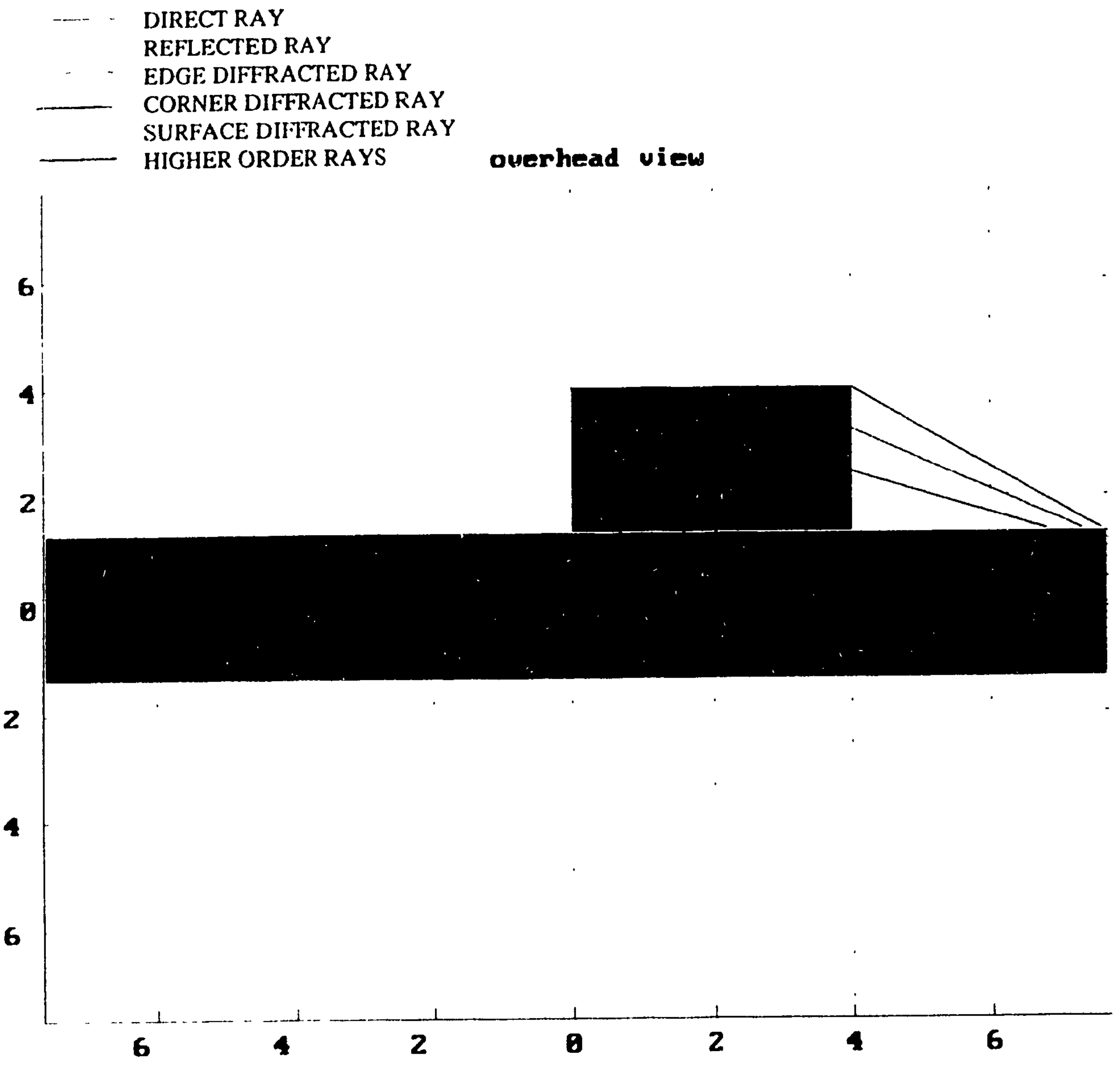


Figure 4.24 Plate-cylinder ray picture overhead view

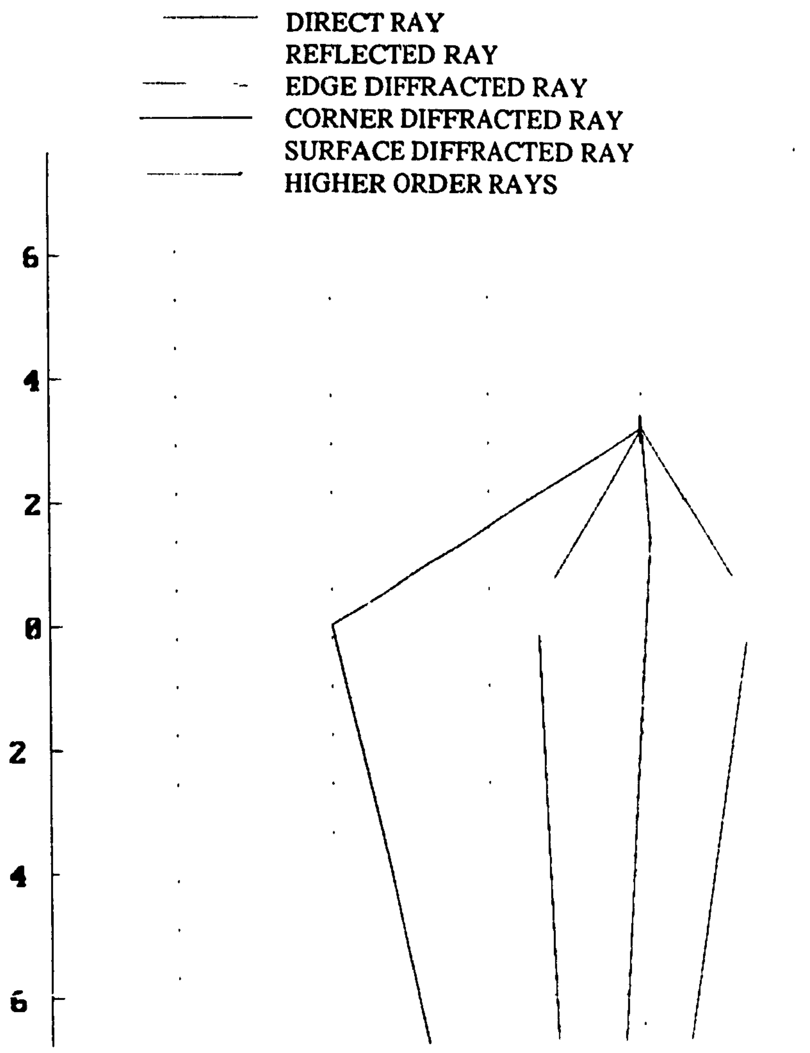


Figure 4.25 Plate-cylinder ray picture front view

5.0 Analysis Using Viewer

The viewer is used as a companion code to the NECBSC code. The initial stage for any study is to develop a model for that study. The actual procedure to develop and analyze a model will require an understanding of the control flow of the viewer. The control flow can be represented in a flow chart as seen in Figure 4.2. This was shown in chapter 4.

This study will focus on the Challenger Jet. This aircraft is a 19.7 meter long aircraft with a notch antenna mounted on the tail assembly. A drawing of the Challenger Jet is shown in Figure 5.1. The notch antenna is shown in close up in Figure 5.2. These models were taken from the thesis of Quan Cuong Luu [51]. The Figures 5.14 and 5.15 represent the axis used to determine the values of theta and phi. A point in three dimensional space can be described by using a distance from the origin and two angles, theta and phi. Theta is the angle measured from the z axis towards the radial vector, in the case of Figure 5.3 theta is shown. Phi is the angle measured from the x axis to the projection of the radial vector on the x-y plane. The Figure 5.4 shows phi. The relationship for the angles used in the viewer is shown in Figure B.1 in appendix B. The model of the Challenger Jet is given for reference and to illustrate the relationship between the views and the NECBSC2 angles. The first stage of the control is the starting menu. This will allow for the choice of using either the data trace or progressing to the BSCV main menu. The main menu has the option to edit the input file.

The edit input file option will allow the user to create a model of the geometry to be studied. The position of the receiver is also determined from the input file. When all of the information is input, the user exits the editor and will return to the main menu.

The aircraft was modeled as a circular cylinder for the fuselage with flat plates for the wing and tail fin assembly. The engines were modeled as flat plates. The fuselage has a diameter of 2.7 meters and a length of 15 meters. The flat plates for the wing and tail fin assembly match the shape and size of the actual elements of the plane. The notch antenna was modeled as a dipole antenna. There are two reasons for this choice of antenna. The first and most compelling is that the NECBSC2 code uses either a dipole antenna or an aperture antenna. The structure of the notch antenna as seen in the Figure 5.2 would make the dipole antenna a more likely choice due to symmetry in form. A more compelling argument for the use of the dipole as the model for the source is found in Quan Cuong Luu's thesis work where the notch antenna was modeled as a wire segment over the tail fin assembly leading edge [52]. In this case the form and current distribution would make the choice of a dipole seem to be a reasonable first approximation to the true notch antenna.

A more accurate representation of the effect of the notch antenna would require a hybrid technique which would use the Method of Moments to model the antenna performance near the conducting edge of the airframe and then resolve the field pattern about the aircraft and determine any other quantities that are required.

The next step would be to preview the input geometry. This is always advisable since an error in the model would invalidate the results of the study and would waste computer time. To preview the model, the user selects the Run Previewer option from the main menu screen. This selection would next allow the user to choose preview the object. Examples of the output from the previewer are shown in Figures 5.5-.5.7

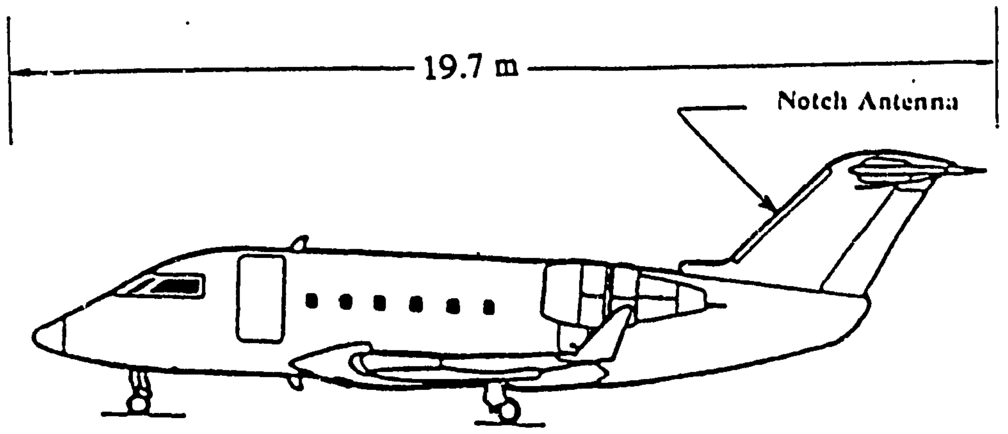


Fig. 5.1 The CL-600/CHALLENGER aircraft with its notch antenna.

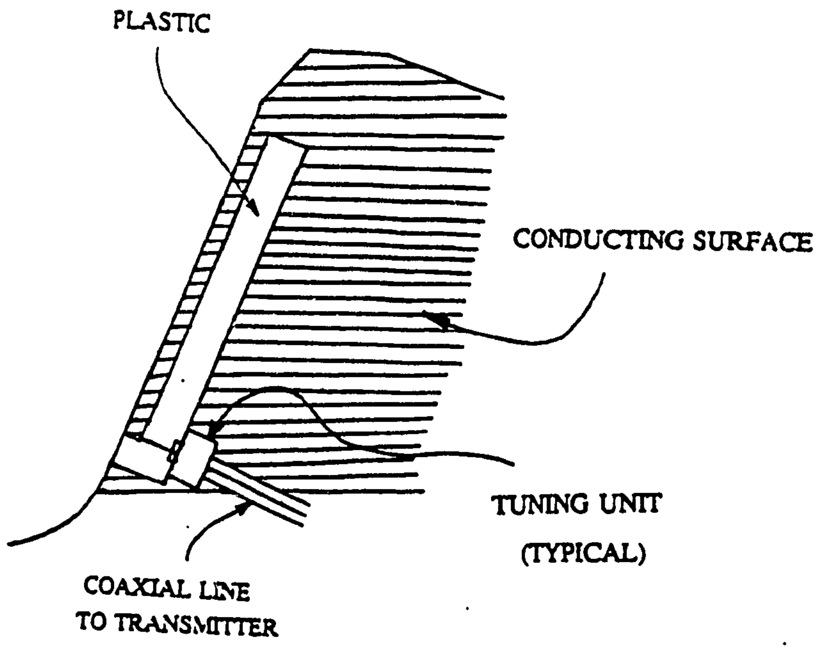


Fig. 5.2 The construction of the notch antenna on the CL-600.

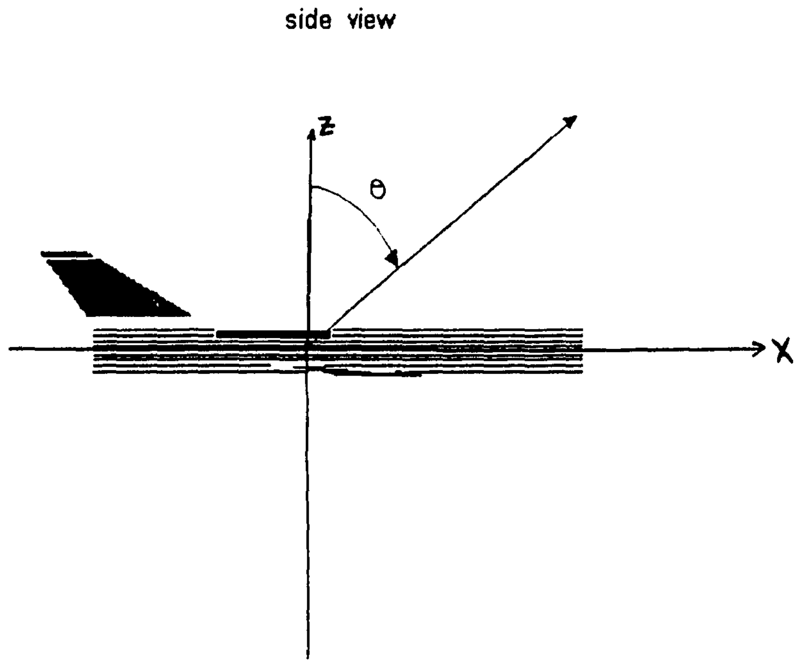


Figure 5.3 Determination of theta for NECBSC2

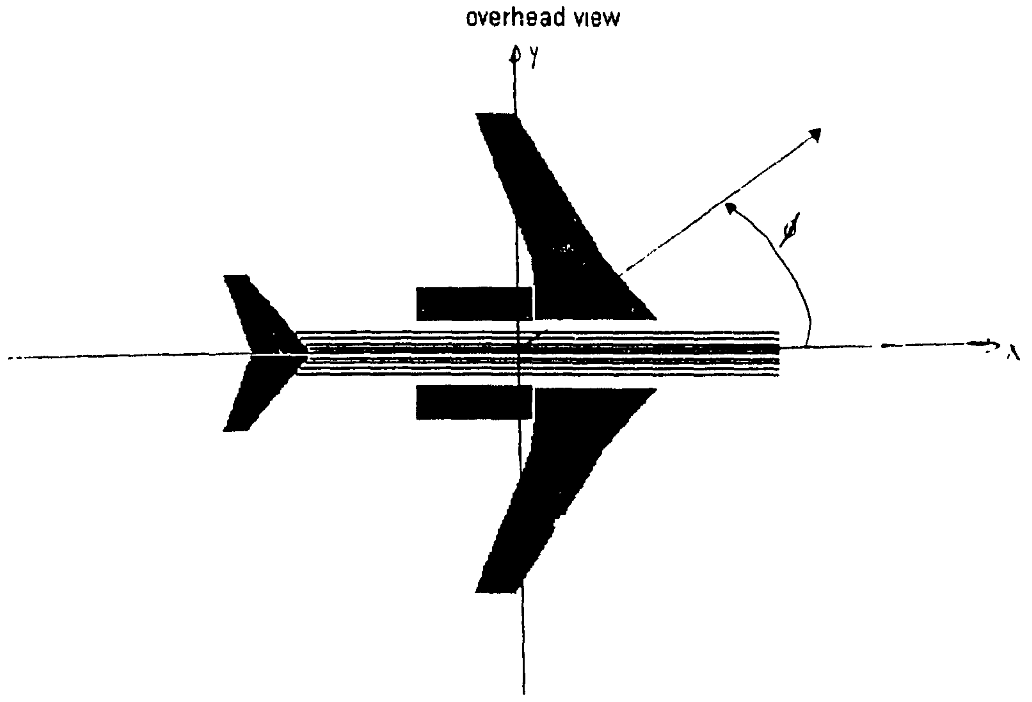


Figure 5.4 Determination of phi

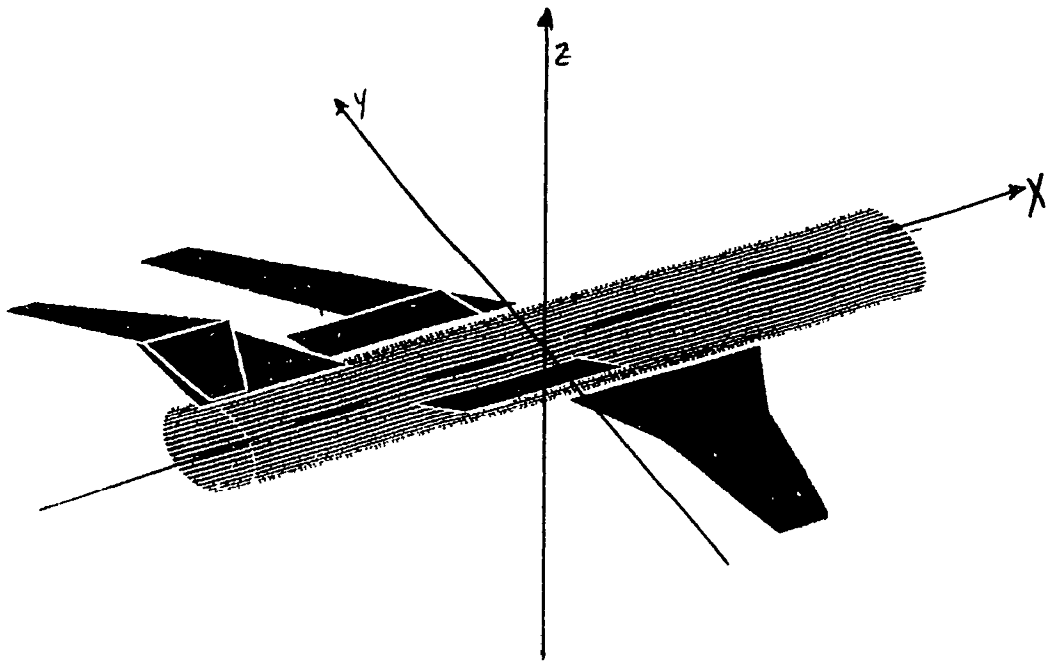


Figure 5.5 Preview of Challenger Jet model

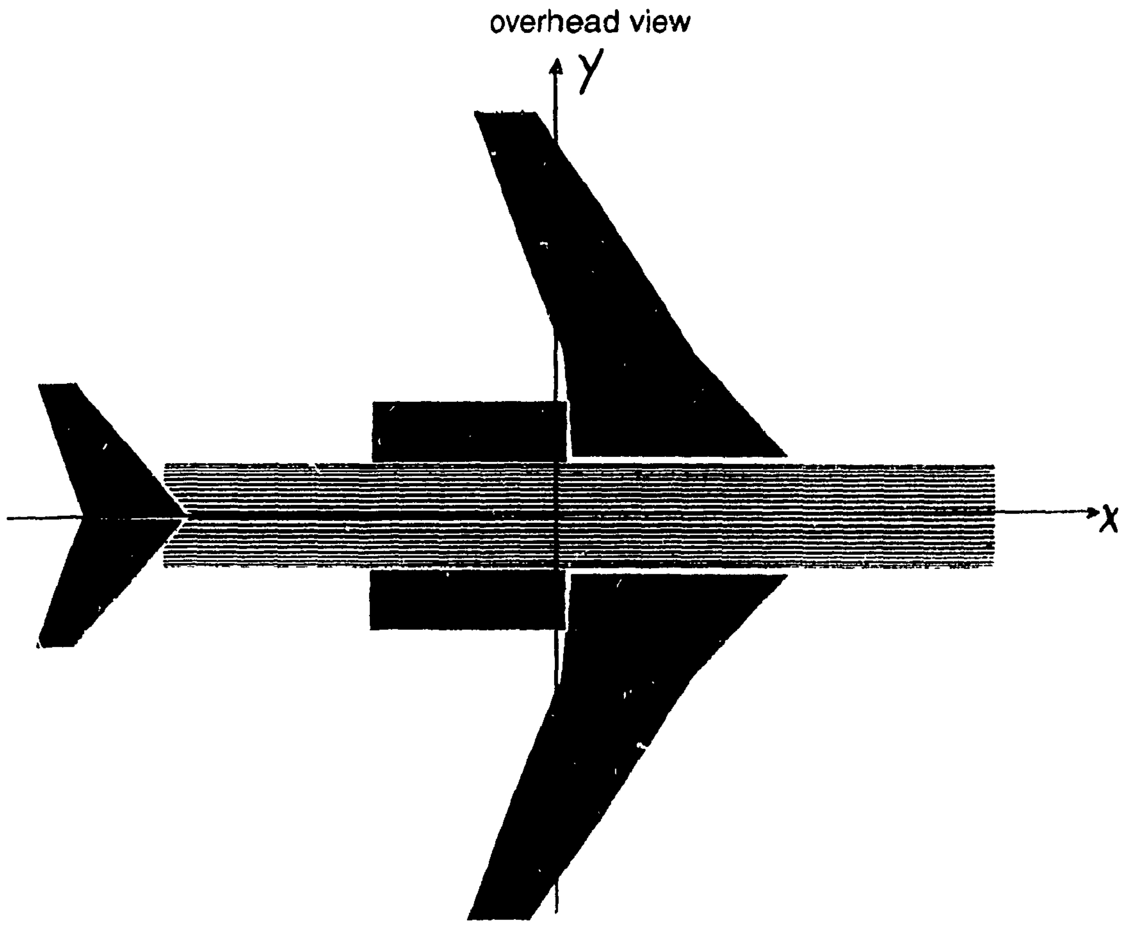


Figure 5.6 Overhead view of Challenger Jet model

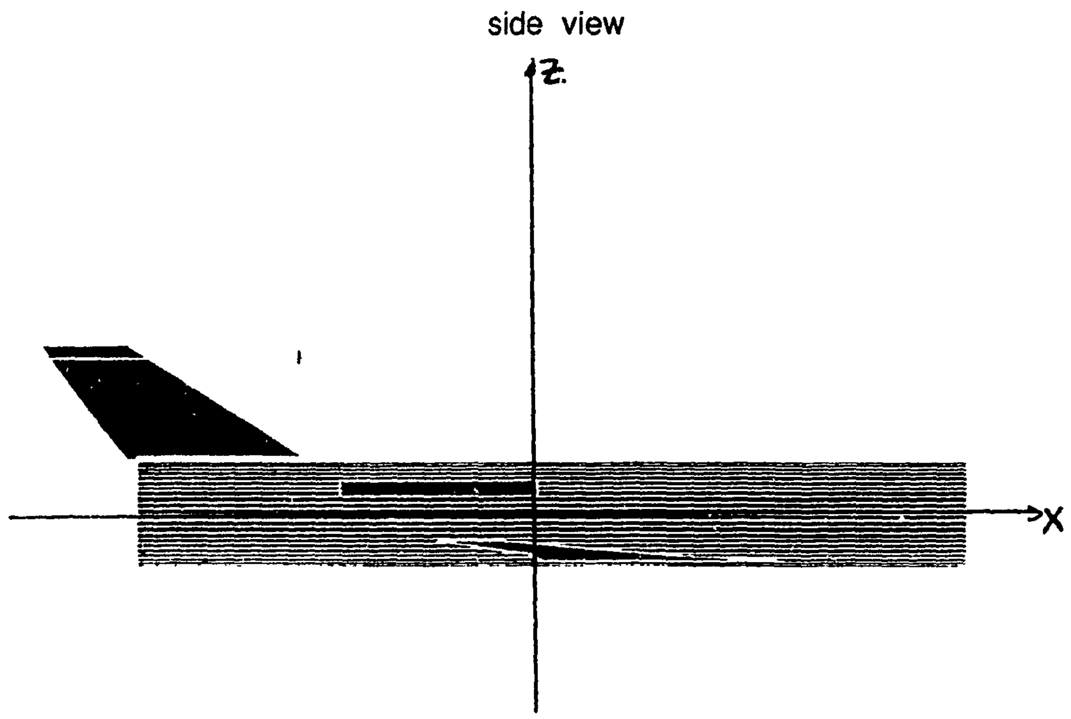


Figure 5.7 Side view of Challenger Jet model

The next stage in the analysis is to choose the option run NECBSC. This will activate the NECBSC code and feed it the input file that was written up in the first stage. The NECBSC code will take over control of the BSCV and the viewer will be inactive until the NECBSC code has finished the run. This is to prevent the user from accessing files that are being used by the analysis programs.

After the NECBSC code has completed the run, the field and ray path data are available to be accessed. If the ray path data is to be accessed, the Run Ray Tracer option is chosen from the main menu. This will then offer the options for what type of ray tracing the user requires. Examples of the type of output that can be generated from these ray tracing routines are shown in Figures 5.8-5.11. Figure 5.8 shows the output if the user selects the color coded ray mechanisms option. While the number of rays is large and the image may be a bit cluttered, the diffraction from the front endcap of the cylinder can be seen clearly in magenta. The corner diffraction from the wingtips shown in black are also visible. Figure 5.9 shows the output if the color coded magnitude option is selected. From this image the intensity of the ray may be identified. The stronger red and orange rays can be seen to be emanating from the front end of the aircraft while the weaker rays are seen to emanate mainly from the tail fin assembly. The option to choose only selected rays is available and samples of the output for this option can be seen in Figures 5.10 and 5.11. Figure 5.10 shows the direct ray path as the selected mechanism. The image in Figure 5.11 presents the user with the higher order ray paths that occurred around the tail fin assembly.

—— - DIRECT RAY
REFLECTED RAY
EDGE DIFFRACTED RAY
—— · CORNER DIFFRACTED RAY
SURFACE DIFFRACTED RAY
— -- HIGHER ORDER RAYS

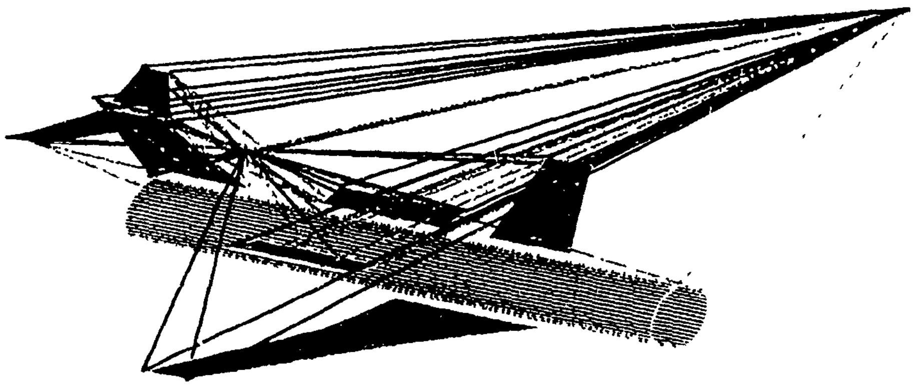
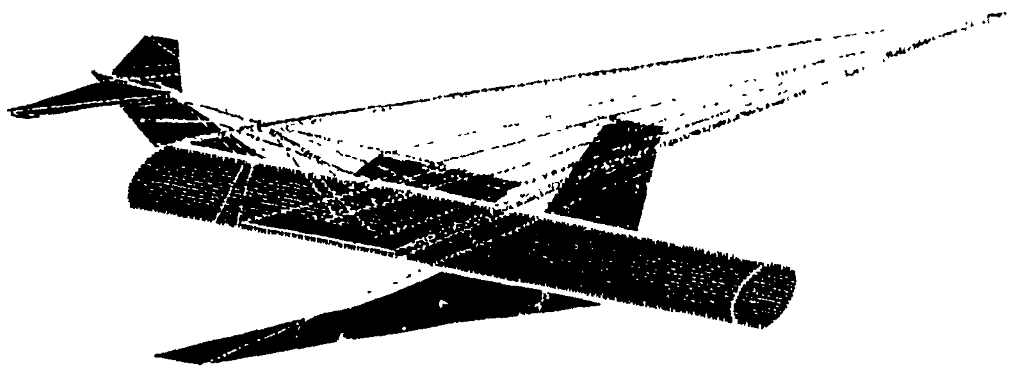


Figure 5.8 BSCV ray tracing output for color coded mechanisms



Max (dB)= -3.07   Min (dB)= -13.27

0dB=1V/m

Figure 5.9 BSCV plot for color coded magnitudes

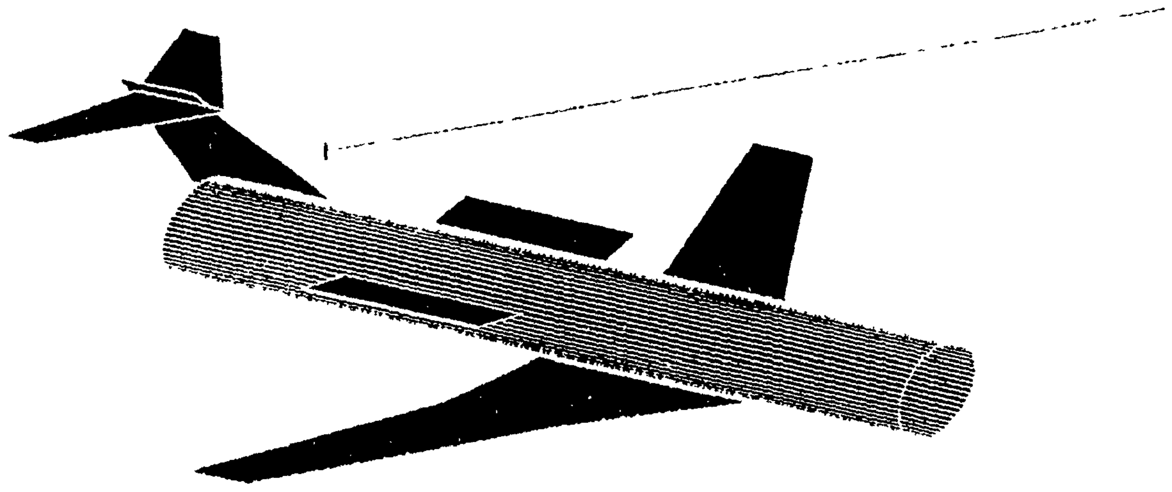
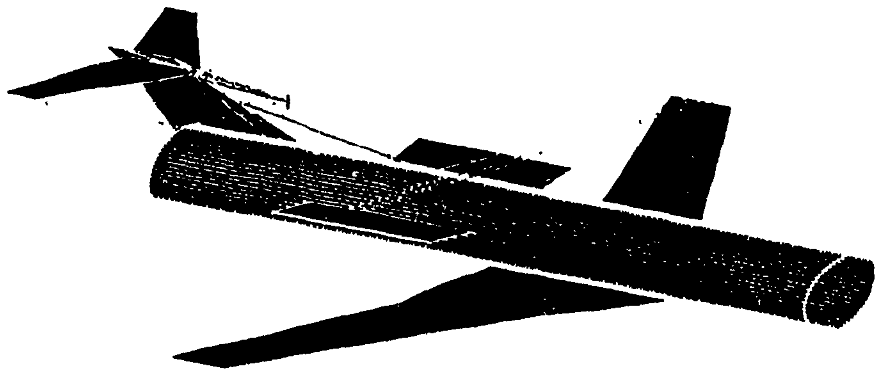


Figure 5.10 BSCV plot for selected mechanisms



Max (dB)= -3.07   Min (dB)= -13.27

Figure 5.11 BSCV plot for selected magnitudes

If the user requires the radiation pattern plotting instead, then the Plotting of Fields option i، selected and the user will then select the radiation pattern type that is required. It should be noted that if the user wants to ray trace, a small number of points should be chosen so that the image is not cluttered and the viewer will not spend a lot of CPU time creating the image of the ray paths.

The other option is the determination of coupling between antennas and radar cross section. To address the coupling issue, the study done by the Concordia University Electromagnetic Compatibility Lab on antenna coupling will be used. The electromagnetic compatibility lab was contracted to perform a study of the coupling between the antennas on the Challenger Jet. The study was part of the contract on low frequency antenna coupling for Defence Research Establishment Ottawa [53].

The study also determined the radiation patterns for the transmit antenna on the tail in of the aircraft. The coupling between antennas can be computed from the following series of relations. The coupling between two antennas with no scattering object in the path of wave propagation is given by equation 5.1 and is called the Friis transmission equation [54].

$$\frac{P_r}{P_t} = e_{cdt} e_{cdr} \left(1 - |\Gamma_t|^2\right)\left(1 - |\Gamma_r|^2\right)\left(\frac{\lambda}{4\pi R}\right)^2 D_{gt}(\theta_t, \phi_t) D_{gr}(\theta_r, \phi_r)\ |\hat{\rho}_t \cdot \hat{\rho}_r|^2 \qquad (5.1)$$

The above expression shows the ratio of the power received to the power transmitted. The terms $e_{cdr,t}$ represent the conductor-dielectric losses in the receive and transmit antennas respectively. The term $\Gamma_{r,t}$ represents the reflection loss due to a mismatch in the impedance of the receive and transmit antennas. The term D_{gr} is used to model the directivity of the receive while the transmit antenna has a directivity of D_{gt}. The unit vectors representing the polarization of the receive and the transmit antennas are symbolized by $\hat{\rho}_{r,t}$. The wavelength is represented by λ and the separation of the two antennas is denoted by the term R. The system of the two antennas in free space can be modeled as in the Figure 5.12 below. The antennas are marked as Tx for the transmit antenna and Rx for the receiving antenna.

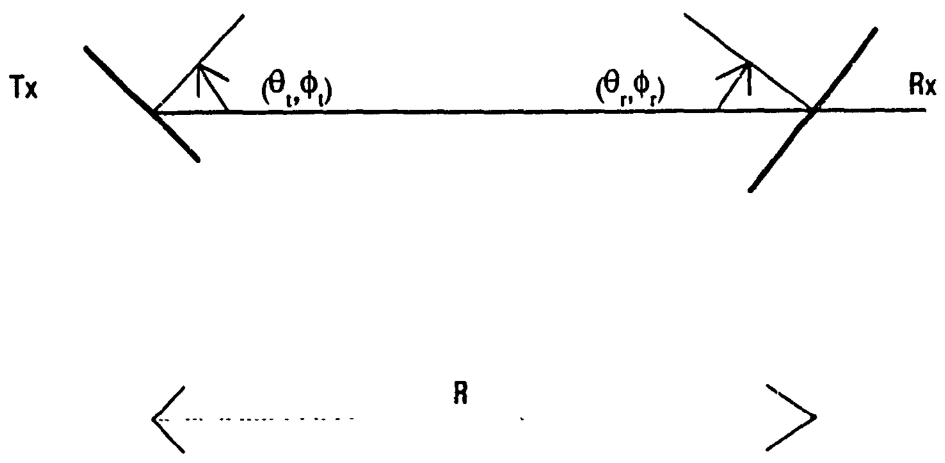


Figure 5.12 Antenna Coupling Diagram

The NECBSC code uses a modified form of the Friis equation. The expression is shown in equation 5.2.

$$\frac{P_o}{P_i} = \frac{\left|0.5 I_m V_{oc}\right|^2}{P_{tt} P_{rr}} \left[\frac{R_r R_t}{\left|Z_r + Z_t\right|^2} \right] \tag{5.2}$$

The variables represent the following quantities: P_o and P_i represent the power received and the power transmitted respectively. The terms P_{tt} and P_{rr} represent the power radiated by the transmitter and the power radiated by the receiver as if it were acting as an transmitter. The term I_m represents the peak current value on the antenna. The term V_{oc} represents the open circuit voltage across the terminals of the antenna. The terms R_r and R_t represent the receiving antenna radiation resistance and the transmit antenna radiation resistance. The terms Z_r and Z_t represent the impedance of the receive and transmit antennas respectively [55].

The applications of coupling and radar cross section may be added to a future version of the viewer as automated features, by generating the MATLAB code and incorporating the new features into the existing menu system.

The near field for the Challenger Jet was calculated by the NECBSC code. The radiation patterns were plotted to see how the fields were behaving around the aircraft. It was at this time that a discontinuity in the near field was noticed.

Figures 5.13 and 5.14 represent two instances of the discontinuity of the field calculated by the NECBSC code, while Figure 5.15 represents the roll plane pattern which is free of discontinuities. The coordinate system used in the analysis of the aircraft is shown in Figures 5.3 and 5.4. The plots shown in Figures 5.13-5.15 were made by the program *pplot* which was developed by the Concordia University EMC Laboratory [56]. The plot in Figure 5.13 shows the radiation pattern for an elevation plane cut, which implies theta varying, with phi equal to 5°. It can be seen that there is a discontinuity in the radiation pattern when theta approaches 180°. A second plot, Figure 5.14, taken with phi at 45° shows the same type of discontinuity at theta approaching 180°. The Figure 5.15 shows the last cut was taken with phi at 90°, there is no discontinuity in this pattern.

The viewer was used to see if there was a UTD mechanism that was missing or if there was a incorrect ray arriving at the receiver antenna. This was done by using the ray tracing features. The study showed that there was no discontinuity in the ray picture. The number and type of rays was unchanged around the region of the discontinuity. This would imply that either the field computation section of the code was in error or there was a missing high order UTD term. If the error is within the field calculation part of the code or the section that generates the receiver position, modification of the code would be necessary. If the error is due to a missing higher order mechanism, then a routine would need to be added. The ray pictures for the case of phi equaling 5° and theta of 179°, 180° and 181° can be seen in the Figures 5.16, 5.17 and 5.18 respectively. It should be noted that the ray picture shows no new rays. This implies that a sudden appearance or disappearance of a ray is not the cause of the discontinuity of the radiation pattern.

The viewer has narrowed down the options to the two listed above. Since the discontinuity was away from the location of any of the antenna locations in this study, the discontinuity was, therefore, not significant to the coupling study.

The discontinuity is seen at the elevation angle equal to 180 degrees. This corresponds to the underside of the airframe. The model was tested and this discontinuity occurred with only the fuselage as well as with the full model of the aircraft. The two examples are at phi equals 5 degrees and phi equals 45 degrees. Therefore the discontinuity is present for a large range of angles.

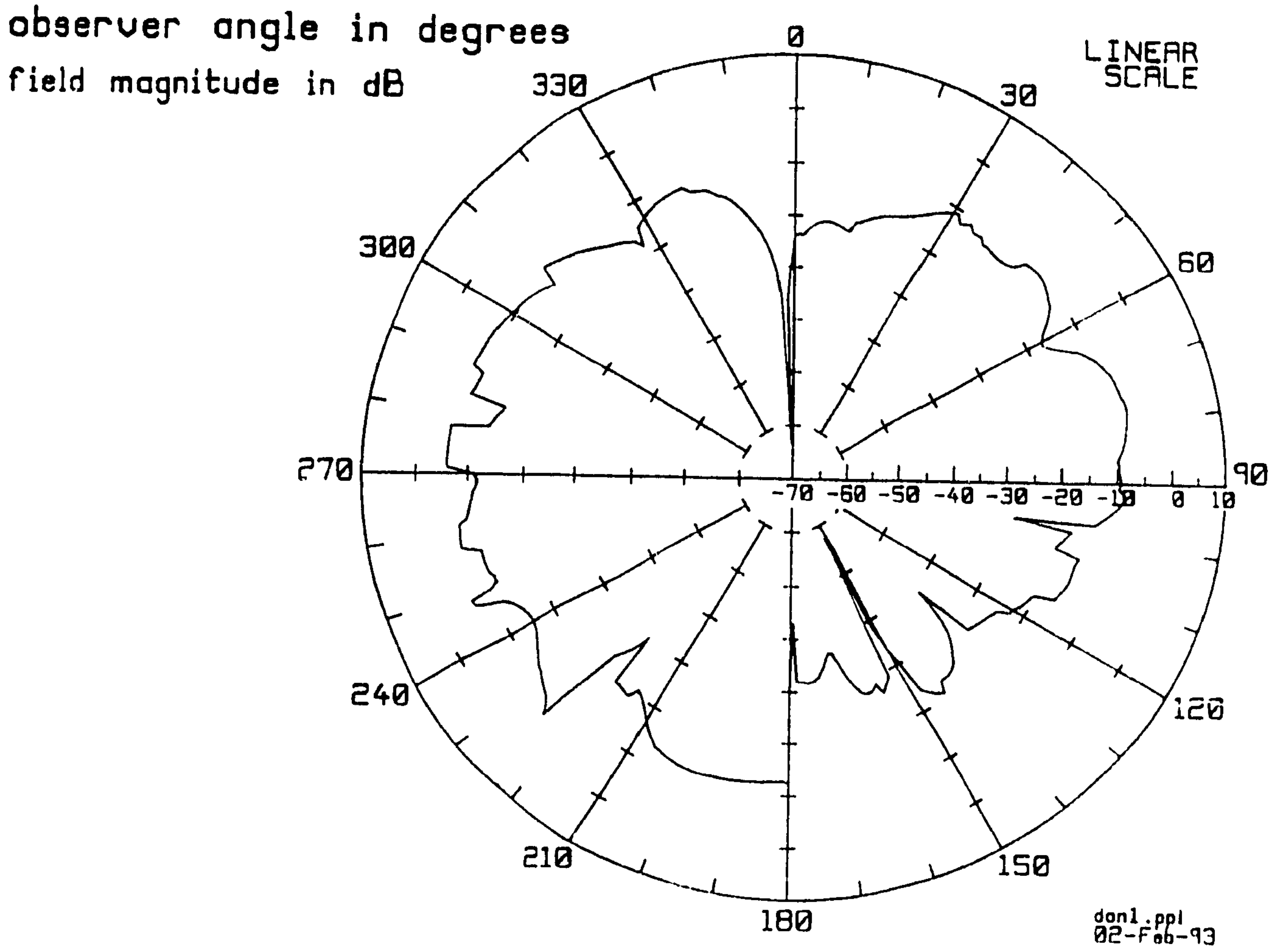


Figure 5.13 Elevation plane cut of electric field ϕ at 5 degrees

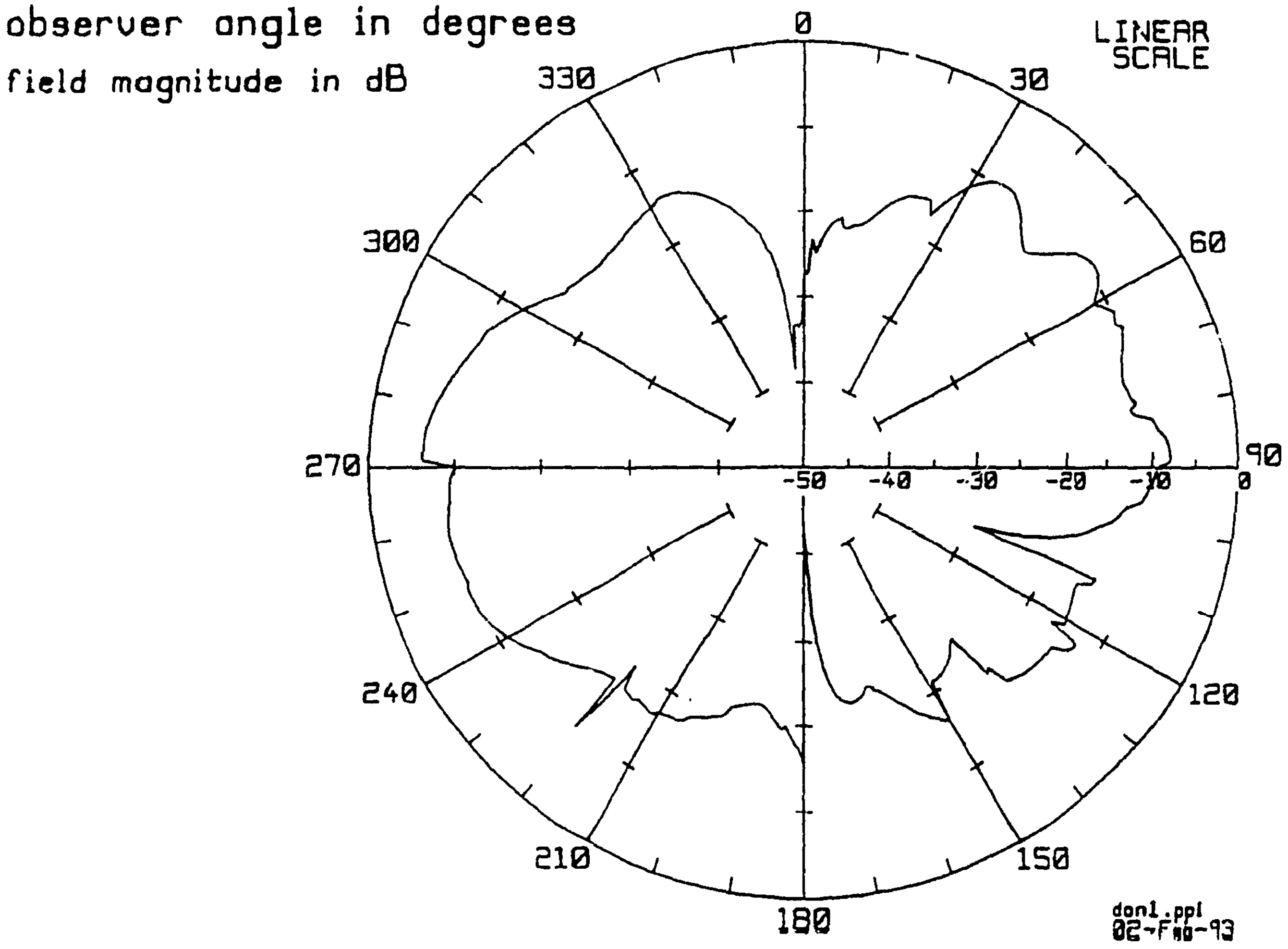


Figure 5.14 Elevation plane cut of electric field at ϕ equal 45 degrees

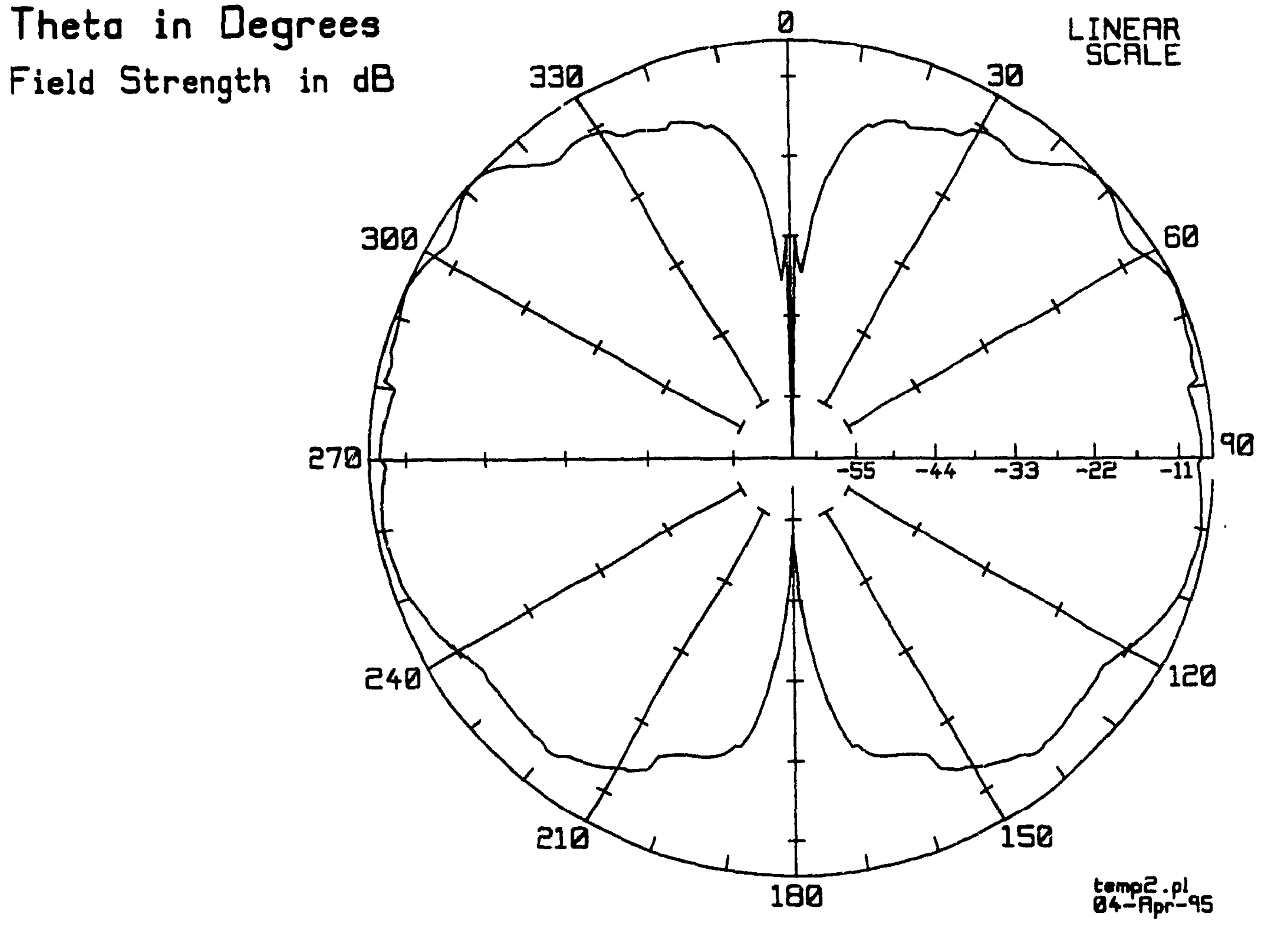


Figure 5.15 roll plane cut of electric field ϕ at 90 degrees

DIRECT RAY
— REFLECTED RAY
— EDGE DIFFRACTED RAY
— CORNER DIFFRACTED RAY
SURFACE DIFFRACTED RAY
— HIGHER ORDER RAYS

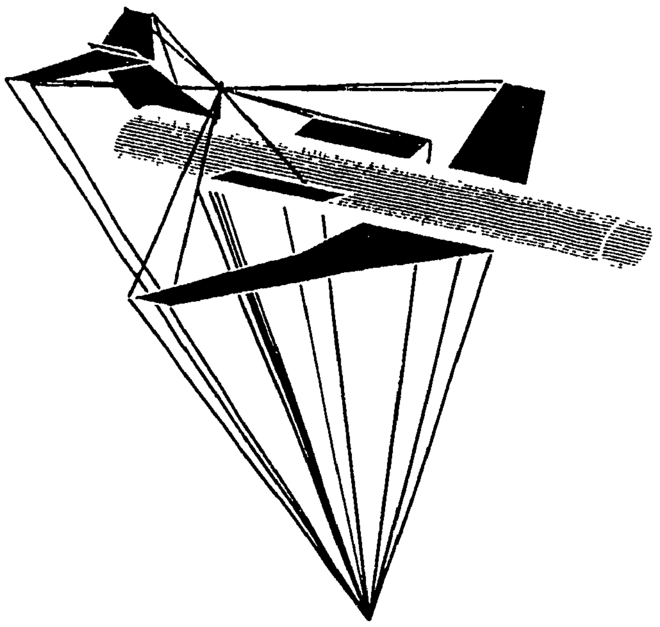


Figure 5.16 Ray Picture ϕ=5°, θ=179°

— DIRECT RAY
REFLECTED RAY
-·-·- EDGE DIFFRACTED RAY
——— CORNER DIFFRACTED RAY
SURFACE DIFFRACTED RAY
——— HIGHER ORDER RAYS

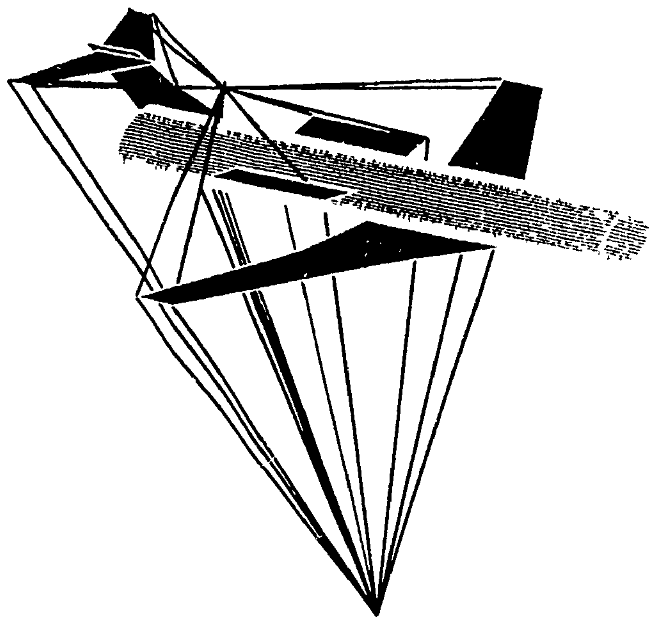


Figure 5.17 Ray Picture ϕ=5°, θ=180°

DIRECT RAY
REFLECTED RAY
——— EDGE DIFFRACTED RAY
——— CORNER DIFFRACTED RAY
SURFACE DIFFRACTED RAY
——— HIGHER ORDER RAYS

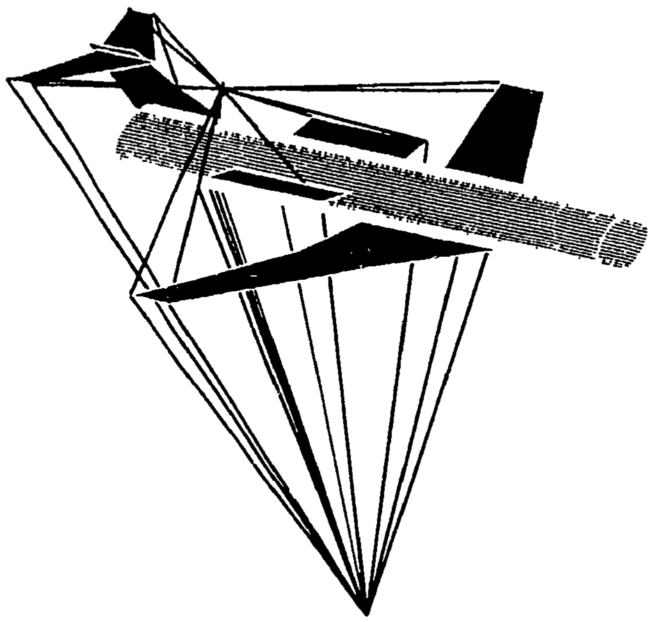


Figure 5.18 Ray Picture $\phi=5°, \theta=181°$

From the Figures 5.16-5.18 an important item should be mentioned. There is a surface diffracted ray which passes through the plate used for the engine. The behavior is the same as that discussed in example 4.3.2. This behavior which was identified by the viewer would lead the user of the code to question the accuracy of results for those observation points.

The viewer was also used to generate the images of the Challenger Jet used in this thesis. By plotting the radiation patterns and linking the pattern to a specific ray picture, the user of the viewer may determine what elements of the geometry have the most effect upon the field at a particular point in space.

This linking of the ray picture to the field pattern allows a better understanding of the ray optical fields that are used to construct the total field. From this understanding the code user may be able to design and modify the object under study so as to optimize the electromagnetic compatibility of the object. The use of color coded magnitudes for the rays allows the user to determine the significance of the rays and the geometry that caused them. This allows the user to decide if the UTD term is of consequence in the model under study. The viewer also identified the presence of incorrect rays. This will allow the code user and code developers to identify and modify problem areas in the NECBSC2 code to better model the actual behavior of electromagnetic fields using UTD theory and techniques.

The viewer is a complete package that will allow the user to perform a study of how an object may affect the UTD field of an antenna or to determine the radar cross section of a certain geometry. The plotting of the fields and the ray tracing are linked controlled from the same program. The menu driven system may be completely specified so that a user may customize the package for a specific need.

The next section contains the conclusion and suggestions for further research.

6.0 Conclusions

An integrated package which allows for easy studies on the effects of scattering geometries on electromagnetic fields has been developed in this thesis. The code is easy to use, requiring intuitive controls through a GUI. Standardized files are used so that the program is device independent. The package produces graphics for presentation quality output and allows the user of the code to visually determine the behavior of the fields that are calculated by the code.

The menu system of the BSCV code is composed of a set of nested option lists. The user may always move through the menus either forward or backward through the use of the mouse. The three dimensional graphics may be viewed from any distance and from any angle by using slider menus that will give the user a full range of positions. The entire three dimensional space may be chosen by the use of the slider menus. This intuitive form of control through the use of nested menus and GUI's allows the user to interactively use the viewer as a tool to help with the analysis of the model. This removes the need for the code user to memorize an extensive list of commands.

The FORTRAN and MATLAB files that comprise the viewer are independent of the platform used. The text files for the source code of the viewer may be ported to any machine and then compiled to create the viewer. The MATLAB files and the data files generated after the run of the NECBSC code are identical for all machines that will use the viewer. Since the data sets and the source code are identical, the viewer can be used in either stand alone or networked form. This, and the ease with which the code may be expanded by the grafting of additional features to the viewer allows the package to be very portable and versatile.

The ability to produce presentation quality graphics can be addressed by the quality of the images shown in this thesis. All of the geometry and ray images were generated as well as some of the radiation pattern plots were generated by the viewer. The viewer is able to support color postscript printing as well as grayscale printing. The images may be sent directly to a printer or saved as either a color or grayscale postscript file or GIF file. The menus are from the PC version of the program but the differences in the two menu systems are insignificant. The capacity to produce quality images of the ray paths and of the previewed object are proof of the viewer's ability to produce presentation quality graphics.

From the above discussion it becomes apparent that the viewer has fulfilled the requirements for the integrated package used as a tool in electromagnetic compatibility studies. The limits of the viewer have not yet been reached. The development of the BSCV package is the introduction to the potential of such packages.

This leads to the next topic, future research. The addition of error checking routines would improve the usefulness of the code. The plates of the model should be defined such that the corners are numbered in accordance with the right hand rule with the thumb pointed towards the illumination. A visual warning from the previewer that could indicate when the plates are not defined correctly would be of assistance for detecting and correcting these errors [54].

The attachment of plates to other plates and cylinders requires that the plate be inserted a small amount within the surface 10^{-5} wavelengths. The previewer could have the feature of highlighting the components of the geometry that are not attached deeply enough. This would allow the code user to reposition the plates such that they are either fully attached or a safe distance away from the other parts of the geometry.

The last feature that could be added is a CAD interface. This would allow the code user to produce a model without editing the text of the input file. The user would generate the image on the screen and this image would be directly translated into the input file for the NECBSC code. This would greatly reduce the amount of time in model development as the user would be continuously previewing the object.

The EAM-BSC code has the same base analysis technique as the BSCV code. The implementation of the two codes varies. The EAM-BSC code offers a CAD interface, the BSCV code does not. The BSCV code offers ray tracing, EAM-BSC does not. The

EAM-BSC code is limited to the Microsoft Windows operating platform, BSCV is platform independent as well as operating system independent. The EAM-BSC code is written in C++ code which makes upgrades necessary though recompiled code, BSCV upgrades and new features are in the form of text files that may be freely distributed through email. The BSCV code requires MATLAB as the command interpreter, this is not a restriction since even the relatively inexpensive student edition of MATLAB is sufficient for our application. MATLAB is available on all of the platforms that are commonly used in both private and public institutions. The BSCV code offers a versatility that the EAM-BSC code does not. The advantage of a CAD interface is offset by the lack of ray tracing and the more complicated method of upgrading the code.

It is hoped that in the near future codes with such as the viewer will become a standard tool for all of those who are interested in the area of electromagnetic compatibility.

References

[1] T. Hubing, "Using commercial CAD packages to interface with 3-d modelling codes", *ACES 8th annual review of progress in applied computational electromangetics*, pp.607-610, (Monterey California), March 1992.

[2] D. Davis et al.,"A ray tracer for the NEC basic scattering code", *ACES 10th annual review of progress in applied computational electromagnetics*, Vol. II, pp. 388-395, (Monterey California), March 1994.

[3] R.J. Marhefka and W.D. Burnside, *Numerical Electromagnetic Code: Basic Scattering Code Nec-BSC (version 2) Part I: User's manual* The Ohio State University ElectroScience Laboratory technical report 712242-14.

[4] R. G. Kouyoumjian and P. H. Pathak, " A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface", *Geometric Theory of Diffraction IEEE press*; *Proc. IEEE*, vol 62, pp. 1448-1461, Nov. 1974.

[5] The Math Works Inc., *MATLAB High-Performance Numeric Computation and Visualization Software user's guide*. The Math Works inc., 1992.

[6] SAIC Inc., *Communication Engineering Laboratory Products and Services Product Sheets*, Science Applications International Corporation, 300 Nickerson Road, Marlborough, Massachusetts 01752.

[7] H. Widmer, *A Guide to Using the AAPG Program (version 07)*, Consulting Report ECAC-CR-87-024, DoD ECAC, Annapolis, MD, July 1987.

[8] F. A. Molinet, "GTD/UTD: Brief history of successive development of theory and recent advances-applications to antennas on ships and aircraft", *AGARD Lecture Series No. 165*, (London, United Kingdon), 26-27 October.

[9] P. R. Foster, "Geodesics on convex surfaces for a GTD/UTD Program", *ACES 8th Annual Review of Progress in Applied Computational Electromagnetics*, pp. 404-411, (Monterey, California), March 16-20, 1992.

[10] M.Stephenson and H. Christiansen, *Movie.byu training Text 1985 edition*, Community Press, Provo Utah, 1985.

[11] R. H. Landau and P. J. Fink Jr., *A Scientest's and engineer's guide to workstations and supercomputers*, John Wiley & Sons 1993.

[12] *Webster's New World Encyclopedia*, Prentice Hall, 1992., pp. 295-296.

[13] D.A. McNamara , C.W.I. Pistorius, J.A.G. Malherbe, *Introduction to the Uniform Geometrical Theory of Diffraction*, University of Praetoria, Artech House: London, 1990,pp. xiii-xiv.

[14] Sedgwick and Tyler, A Short History of Science, McMillan, New York, 1976.

[15] D. Cheng, *Field and Wave Electromagnetics 2nd edition*, Addison Wesley Publishing Company, New York, 1989.

[16] McNamara et al., p.3.

[17] McNamara et al., p.7.

[18] *Webster's New World Encyclopedia*, pp. 465-466

[19] W. H. Hayt Jr., *Engineering Electromagnetics 4th edition*, McGraw-Hill Book Company, New York, 1981.

[20] R.M. Luneberg, *Mathematical Theory of Optics*, Brown University Press, Providence, RI, 1944

[21] M. Kline and I. Kay , *Electromagnetic Theory and Geometrical Optics*, Interscience New York, 1965

[22] J. B. Keller, "Geometrical theory of diffraction", *Journal Optical Society of America*, vol. 52, no. 2, Feb. 1962, pp. 116-130.

[23] C. Balanis, *Advanced Engineering Electromagnetics*, John Wiley & Sons, New York, 1989.

[24] J. B. Keller , " Diffraction by a convex cylinder", *IRE Transactions on Antennas and Propagation*, AP-4, 1956, pp. 312-321.

[25] R. G. Kouyoumjian,"Asymptotic high frequency methods", *Proceedings IEEE*, vol. 53, Aug. 1965, pp. 864-876.

[26] R. G. Kouyoumjian and P.H. Pathak, "A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface", *Proceedings IEEE*, vol. 62, Nov. 1974, pp. 1448-1461.

[27] R. C. Hansen" Applications" , *Geometric Theory of Diffraction IEEE press* pages vi-vii

[28] M. Kline "An asymptotic solution of maxwell's equations", *Comm. Pure Appl. Math*, Vol. 4.1951 pp. 225-262.

[29] M. Kline and I. Kay, *Electromagnetic Theory and Geometrical Optics*, Intercience , New York, 1965

[30] McNamara et al., pp. 10-11.

[31] R. G. Kouyoumjian " Asymptotic high-frequency methods " *IEEE proceedings vol. 53*, Aug 1965, pp. 864-876.

[32] McNamra et al., p. 17.

[33] McNamara et al., p. 31.

[34] McNamara et al., pp. 34-36.

[35] McNamara et al., p. 77.

[36] W.Stutzman and G.Thiele, *Antenna Theory and Design*, Wiley & Sons, New York, 1981.

[37] D. Cheng, pp. 406-408.

[38] McNamara et al., pp. 94-104.

[39] McNamara et al., pp. 268-271.

[40] J. B. Keller, "One hundred years of diffraction theory", *IEEE transactions on Antennas and Propagation*, VOL. AP-33, No. 2, february 1985.

[41] McNamara et al., pp.266-267.

[42] McNamra et al., pp. 270-271.

[43] J. B. Keller, "Geometrical theory of diffraction" *Geometrical Theory of Diffraction IEEE Press*, Feb. 1962, pp 7-21.

[44] McNamara et al., pp. 197-199.

[45] McNamra et al., pp. 288-291.

[46] McNamara et al., pp. 364-366.

[47] McNamara et al., pp. 361-363.

[48] P. H. Pathak, W.D. Burnside, R.J. Marhefka."A uniform GTD analysis of the diffraction of electromagnetic waves by a smooth convex surface", *Geometric Theorey of Diffraction, IEEE press* pp155-166.

[49] McNamara et al., pp. 240-244.

[50] R. J. Marhefka, *Numerical Electromagnetic Code - Basic Scattering Code, NEC-BSC (version 2), Part II: Code Manual*, Ohio State University Electroscience Laboratory, Technical Report 712242-15, Department of Electrical Engineering, Columbus, Ohio 43212.

[51] Q. C. Luu, *Numerical Techniques for the Study of HF Coupling Modes on Large Aircraft*, Master's Thesis, Concordia University Department of Elec. and Comp. Engineering, Montreal Quebec, H4B 1R6, March 1994, p. 13, p. 213

[52] Q. C. Luu, pp. 221-224.

[53] S. J. Kubina and C. W. Trueman, *Low Frequency Antenna Coupling*, Final Report Contract No. W7714-2-9646/01-ST, for Department of National Defense, by Electromagnetic Compatability Laboratory, Concordia University Department of Elec. and Comp. Engineering, Montreal Quebec, H4B 1R6.

[54] C. Balanis, *Antenna Theory Analysis and Design*, John Wiley & Sons, New York, 1982, pp 63-65.

[55] R. J. Marhefka, pp. 6.9-6.11.

[56] C.W. Trueman, *PPLOT*, Electromagnetic Compatibility Laboratory, Concordia University Department of Elec. and Comp. Engineering, Montreal Quebec, H4B 1R6.

Appendix A: Source Codes

BSV.F FORTRAN source code listing

```
         program prepro
c     .
************************************************************************
*
c *      This program will preprocess the input file of bsc. This
procedure
c *      will create a matrix with 5 columns and a variable number of
rows.
c *      Each command will be given a separate section. Comment cards
CE and
c *      CM will be ignored. Pattern and other non geometry commands
will be
c *      ignored. The program will open four files. "inbsc.dat",
"geommat.dat"
c *      and finally "heading.dat". The first file >> inbsc.dat << will
be
c *      read only. No modifications should be done to inbsc as it is
used
c *      as the input file to run bsc. The next file "geommat.dat" will
c *      contain the information about the geometry of the structure
being
c *      studied. This file will contain the plate data. The next file
c *      "cylnmat.dat" will contain the data required to draw the
cylinder
c *      in matlab. The last data file, "heading.dat" holds any other
data
c *      such as the source locations.
c *
c
************************************************************************
**
c
c
c
========================================================================
==
c =
c =                      Defining the variables
c =
c =      xpos= the x position in three space of a feature
c =
c =      ypos=     y
c =
c =      zpos=     z
c =
c =      thetazp= the theta angle rotation used to alter the z axis
```

```
c =      position with respect to the reference axis. (cylinder)
c =
c =      phizp= the phi angle rotation used to alter the z axis postion
c =      with repect to the reference axis. (cylinder)
c =
c =      thetaxp= the theta angle rotation used to alter the x axis
c =      postion with repect to the reference axis. (cylinder)
c =
c =      phixp= the phi angle rotation used to alter the x axis
position
c =      with repect to the reference axis. (cylinder)
c =
c =      radx= the x axis radius of the cylinder. Assuming the cylinder
c =      is originally defined as a z oriented cylinder and then
rotated
c =      to the appropriate postion.
c =
c =      rady= the y axis radius of the cylinder.
c =
c =      firstz= the most negative point of a flat endcap cylinder.
c =
c =      lastz= the most positive point of a flat endcap cylinder.
c =
c =      vrt= the matrix which will be used to rotate the shape
c =      according to the thetazp et al.
c =
c =      ang1= the angle of the cut of the endcap of one side of the
c =      cylinder.
c =
c =      ang2= the angle of the cut of the endcap of the other side.
c =
c =      temp= all temps are temporary storage and calculation
variables.
c =
c =      x= the x position in three space of a corner of the plate.
c =
c =      y=     y
c =
c =      z=     z
c =
c =      len= the length of the cylinder.
c =
c =      rtd= radius to degrees conversion factor.
c =
c =      rotaz1= the theta value of the rotation of the z axis of
c =      object with respect to the reference axis.
c =
c =      rotaz2= the phi value of the rotation of the z axis.
c =
c =      rotax1= the theta value of the rotation of the x axis
c =
c =      rotax2= the phi value of the rotation of the x axis
c =
c =      pi= the value of pi ==>3.14......
c =
c =      dot= the value of the dot product of two vectors.
c =
c =      vrt2= another rotation matrix, (RT: command uses this one).
c =
c =      tx = the translation of the coordinate axis for the x position
c =      used with the RT: command.
c =
c =      ty = y axis translation.
c =
c =      tz = z axis translation.
c =
c =      numsid= the number of sides of the plate.
c =
c =      cnt= a dummy variable used to control looping.
c =
c =      instr= a character string that will be used to look into
c =      the input file for the relevant data.
c =
c =      filnm= the name of the input file used to get the data.
c =
c =      thetsz= theta orientation of source z axis
c =
c =      phisz= phi orientaion of source z axis
c =
c =      thetsx= theta orientaion of source x axis
c =
c =      phisx= phi orientation of source x axis
c =
c =      plat= flag to tell viewer if plates are present (0 means no)
c =
c =      cylin= flag to tell viewer if cylinders are present (0=>no)
c =
c
=====================================================================
=
c
c
c ----------------------------defining the variables--------------------
----
         real xpos,ypos,zpos,vrt(3,3),thetzp,phizp,thetxp,phixp
         real radx,rady,firstz,lastz,ang1,ang2,temp,x,y,z,len,rtd
         real temp1,temp2,temp3,rotaz1,rotaz2,rotax1,rotax2,pi,dot
         real vrt2(3,3),tx,ty,tz,thetsz,phisz,thetsx,phisx,cylin
         real plat,jmx(4),coef,vellt,wlth
         integer numside,cnt,j,k,jk,unt
         character instr*2
         character *32 filnm
         logical lslope,ltest,lout,lwarn,ldebug,lcornr,lsor,lkj(4,6)
         pi=4.*atan(1.)
         rtd=pi/180.
         coef=1.
```

```
        jmx(1)=1
        jmx(2)=6
        jmx(3)=5
        jmx(4)=4
c
c -----------------------getting the input file-----------------------
---
c
c       If you want variable names for the input file uncomment this
section
c
c        write(6,900)
c900     format('   Filename= ???   ',$)
c        read(5,910)filnm
c910     format(a)
c

c -----------------------opening the input and scratch files ---------
----
c
c    geommat.dat contains all the plate data.
c    cylnmat.dat contains all the cylinder data
c    heading.dat contains source data
c
c ---------------------------------------------------------------------
---
c        open (unit=7,status='unknown',file=filnm)
        open (unit=7,status='unknown',file='inbsc.dat')
        open (unit=8,status='unknown',file='geommat.dat')
        open (unit=9,status='unknown',file='heading.dat')
        open (unit=10,status='unknown',file='cylnmat.dat')
        open (unit=12,status='unknown',file='parts.dat')
        open (unit=13,status='unknown',file='coeff.dat')
        cylin=0.
        plat=0.
        vellt=300000000.
c
c ---initializing the rotated angles and translate coordinates---------
---
c
        tx=0.
        ty=0.
        tz=0.
        thetzp=0.
        phizp=0.
        thetxp=90.
        phixp=0.
        goto 95
c
c ----------------start looking into file----------------------------
---
c
30      read (7,11) instr
```

```
11       format (36A2)
c
c -----------determine what to do with that command-------------------
---
c
         if (instr.eq.'CM'.or.instr.eq.'CE') goto 30
         if (instr.eq.'UN') goto 35
         if (instr.eq.'FR') goto 40
         if (instr.eq.'UN') goto 40
         if (instr.eq.'US') goto 40
         if (instr.eq.'SG') goto 50
         if (instr.eq.'CG') goto 60
         if (instr.eq.'PG') goto 50
         if (instr.eq.'RT') goto 90
c         if (instr.eq.'PN') goto 70
c         if (instr.eq.'RD') goto 70
         if (instr.eq.'EN') goto 20
         goto 30
c
c ------------------------units-------------------------------------
-
c
35      read (7,*) unt
        if (unt.eq.1) coef=1.
        if (unt.eq.2) coef=12.*2.54/100.
        if (unt.eq.3) coef=2.54/100.
        write(13,*) coef
        goto 30
c ------------------------frequency---------------------------------
-
c
40       read (7,*) temp
c         write(9,*) instr,temp
         wlth=1000000.*temp/vellt
         goto 30
c
c ------------------------source------------------------------------
c
50      if (instr.eq.'SG') goto 55
         goto 56
55       read(7,*) xpos,ypos,zpos
         xpos=xpos+tx
         ypos=ypos+ty
         zpos=zpos+tz
         x=xpos*vrt2(1,1)+ypos*vrt2(1,2)+zpos*vrt2(1,3)
         y=xpos*vrt2(2,1)+ypos*vrt2(2,2)+zpos*vrt2(2,3)
         z=xpos*vrt2(3,1)+ypos*vrt2(3,2)+zpos*vrt2(3,3)
         read(7,*) thetsz,phisz,thetsx,phisx

         read(7,*) temp1,temp2,temp3
         len=temp2
         read(7,*) temp1,temp2
         write(9,*) 1,x*coef,y*coef,z*coef,len*coef
        goto 30
c
c --------------------plate--------------------------------------
c
56      read(7,*) numside,temp1
        plat=1.
        do 57 cnt=1,numside

        read(7,*) xpos,ypos,zpos
        xpos=xpos+tx
        ypos=ypos+ty
        zpos=zpos+tz
      if (cnt.eq.1) xi=xpos*vrt2(1,1)+ypos*vrt2(1,2)+zpos*vrt2(1,3)
      if (cnt.eq.1) yi=xpos*vrt2(2,1)+ypos*vrt2(2,2)+zpos*vrt2(2,3)
      if (cnt.eq.1) zi=xpos*vrt2(3,1)+ypos*vrt2(2,3)+zpos*vrt2(3,3)
        x=xpos*vrt2(1,1)+ypos*vrt2(1,2)+zpos*vrt2(1,3)
        y=xpos*vrt2(2,1)+ypos*vrt2(2,2)+zpos*vrt2(2,3)
        z=xpos*vrt2(3,1)+ypos*vrt2(3,2)+zpos*vrt2(3,3)
        write(8,*) -numside,x*coef,y*coef,z*coef,cnt
57      continue
        write(8,*) -numside,xi*coef,yi*coef,zi*coef,cnt
        goto 30
c
c ---------------------cylinder-----------------------------------
c
60      read(7,*) xpos,ypos,zpos
        cylin=1.
        xpos=xpos+tx
        ypos=ypos+ty
        zpos=zpos+tz
        read(7,*) rotaz1,rotaz2,rotax1,rotax2
        read(7,*) radx,rady
        read(7,*) firstz,angend1,lastz,angend2
        write(10,*) 2,xpos*coef,ypos*coef,zpos*coef,rotaz1
        write(10,*) 3,rotaz2,rotax1,rotax2,radx*coef
        write(10,*) 4,rady*coef,firstz*coef,ang1,lastz*coef
        write(10,*) 5,ang2,0,0,0
        goto 30
90      read(7,*) tx,ty,tz
        read(7,*) thetzp,phizp,thetxp,phixp
C
C  generating the rotation matrix
C
95      vrt2(1,1)=sin(thetxp*rtd)*cos(phixp*rtd)
        vrt2(1,2)=sin(thetxp*rtd)*sin(phixp*rtd)
        vrt2(1,3)=cos(thetxp*rtd)
        vrt2(3,1)=sin(thetzp*rtd)*cos(phizp*rtd)
        vrt2(3,2)=sin(thetzp*rtd)*sin(phizp*rtd)
        vrt2(3,3)=cos(thetzp*rtd)
        dot=vrt2(1,1)**2+vrt2(1,2)**2+vrt2(1,3)**2
        dot=sqrt(dot)
        vrt2(1,1)=vrt2(1,1)/dot
        vrt2(1,2)=vrt2(1,2)/dot
```

```
          vrt2(1,3)=vrt2(1,3)/dot
          dot=vrt2(3,1)**2+vrt2(3,2)**2+vrt2(3,3)**2
          dot=sqrt(dot)
          vrt2(3,1)=vrt2(3,1)/dot
          vrt2(3,2)=vrt2(3,2)/dot
          vrt2(3,3)=vrt2(3,3)/dot
          vrt2(2,1)=(vrt2(3,2)*vrt2(1,3)-vrt2(3,3)*vrt2(1,2))
          vrt2(2,2)=(vrt2(3,3)*vrt2(1,1)-vrt2(3,1)*vrt2(1,3))
          vrt2(2,3)=(vrt2(3,1)*vrt2(1,2)-vrt2(3,2)*vrt2(1,1))
c
c transpose of the matrix. This will equal the inverse in this case.
c
          temp=vrt2(1,2)
          vrt2(1,2)=vrt2(2,1)
          vrt2(2,1)=temp
          temp=vrt(1,3)
          vrt2(1,3)=vrt2(3,1)
          vrt2(3,1)=temp
          temp=vrt2(2,3)
          vrt2(2,3)=vrt2(3,2)
          vrt2(3,2)=temp


c              write(6,*) vrt2(1,1),vrt2(1,2),vrt2(1,3)
c              write(6,*) vrt2(2,1),vrt2(2,2),vrt2(2,3)
c              write(6,*) vrt2(3,1),vrt2(2,3),vrt2(3,3)
          goto 30
c
c ---------------------Test Options--------------------------
c
c
120       read(7,*) ldebug,ltest,lout,lwarn
          read(7,*) lslope,lcornr,lsor
          do 121 k=1,4
             jk=jmx(k)
             read(7,*) (lkj(k,j),j=1,jk)
121       continue
          goto 30
20        write(12,*) plat,cylin
          end
```

TRACE.F FORTRAN source code listing

```
program trace
real temp1,temp2,temp3,temp4,temp5,temp6
integer cnt1,cnt2
character sub1*36
open (unit=7,file='ray.dat',status='unknown')
open (unit=8,file='raytyp.dat',status='unknown')
open (unit=9,file='trace.dat',status='unknown')
open (unit=10,file='trace2.dat',status='unknown')
```

```
30   read(9,31) sub1
31   format(12A6)
     if (sub1.eq.'                        ') goto 40


50   write(10,*) sub1
     goto 30
40   end
```

BSCV.M MATLAB source code listing

```
header='BSC ANALYSIS';
labels=str2mat(...
    'Run Previewer ',...
    'Run Data Trace');
callbacks=str2mat('viewer','debug1');
choices('MAIN1',header,labels,callbacks);
```

BSV1.M MATLAB source code listing

```
% NECBSC2 previewer. This file will take any file in the standard bsc
% format and convert it to a 3-D graphic that can be printed or viewed
% as you choose.
%
%       This previewer was written by Don Davis June 12 1993
%
%     This previewer version was revised October 14 1993 by:
%
%                   Don Davis
%             Concordia University
%
% ===================================================================
% geommat is a matrix that will contain all the plate data required
% to draw a plate for this previewer.
%
% cylnmat is a matrix that will contain all the cylinder data required
% to draw a cylinder for this previewer.
%
% heading will contain the source data used to place the sources.
%
% ray.dat will hold the ray tracing data.
%
% raytyp.dat will hold the data on what ray mechansim it is ,eg.
relected.
%
% parts will contain the information on which parts are present. These
% can be either cylinders or plates.
% ==================================================================
%
```

```
% This file will be used to generate pictures of the inbsc.dat format
% file. The input file will be chosen by the user. This data will be in
a
% matrix form. To run this invoke the preprocessor by typing bsv.
% The preprocessor should be compiled as follows. f77 bsv.f -o bsv.
%
% ==================================================================
%clear
%
% now to call the preprocessor which is a fortran program
%
!bsv
%
% clear any previous graphics
%
clg
%
%set some constants
load coeff.dat
rtd=pi/180;
ref=1;
adj=0;
hold
%
% check for what parts are present
%
load parts.dat
plat=parts(:,1);
cylin=parts(:,2);
%
%
% are plates present ?
if plat>0
%
% Load plate data in matrix form.
load geommat.dat
flag=geommat(:,1);
x=geommat(:,2);
y=geommat(:,3);
z=geommat(:,4);
fin=geommat(:,5);
cnt1=size(flag,1);
%
% rtd=radians to degrees conversion
%
rtd=pi/180;
ref=1;
adj=0;


%-------------------------------------------------------------------
%  This is the section of the program which will use the data for the
%  plates to draw the shapes in three space using fill3 command.
%  the rotation of the plates will have been done by the preprocesor
%  the case of the plates.
%  flag contains the number of corners that the plate has. The loop
%  will cycle through each of the points to make a closed contour of
%  the shape and use fill to color in the polygon. Cnt1 is the number
%  of plates.
%----------------------------------------------------------------------
--
%
%
%  cnt: a dummy variable used to cycle through the plates.
%  chngplt: controls the change from one plate to the next one.
%  cnt3: a dummy variable used to count through the points on the
plate.
%
%
=======================================================================
=
%

     for cnt=1:cnt1,
       chngplt=flag(cnt)+fin(cnt);
       if chngplt== 1
          cnt2=cnt2+1;
          cnt3=flag(cnt)+cnt;

          fill3(x(cnt3:cnt),y(cnt3:cnt),z(cnt3:cnt),'g')
       end
     end
    end

%     pause




     end
%   end of the flat plate section.
end
%
%
% are cylinders present?
if cylin>0
% ---------------------------------------------------------------------
--
%   cylnmat.dat is a NX5 matrix containing data on the cylnders.
%   There are N cylinders. Each cylinder is formed from a set of
rectangles
%   that are parallel to the main axis. These rectangles are made from
%   chords on the base of the cylinder. The base is elliptic or
circular.
```

```
%   The rotation is done here in the case of the cylinder.
% ---------------------------------------------------------------------
-
% clear
%
% Load cylinder data in matrix form
%
%
%
%   cnt2: an index variable used to move through the cylinder data.
%   numcyl: the number of cylinders being considered.
%   x2: cylinder data
%   y2: cylinder data
%   z2: cylinder data
%   fin2: cylinder data
%   xpos,ypos,zpos: x,y,z position of center of cylinder.
%   radx,rady: major & minor axis. not necessarlily in that order.
%   ang*: the angles that define the cylinder coord. system.
%
======================================================================
==
%
%
load cylnmat.dat
flag2=cylnmat(:,1);
cnt2=size(flag2,1);
numcyl=cnt2/4;
%
% numcyl is the number of cylinders that will be drawn
%
x2=cylnmat(:,2);
y2=cylnmat(:,3);
z2=cylnmat(:,4);
fin2=cylnmat(:,5);
     for cnt2=1:numcyl
       phi=0;
       phi2=0;
       cylncnt=(cnt2-1)*4;
       xpos=x2(1+cylncnt);
       ypos=y2(1+cylncnt);
       zpos=z2(1+cylncnt);
       angz1=fin2(1+cylncnt);
       angz2=x2(2+cylncnt);
       angx1=y2(2+cylncnt);
       angx2=z2(2+cylncnt);
       radx=fin2(2+cylncnt);
       rady=x2(3+cylncnt);
       firstz=y2(3+cylncnt);
       ang1=z2(3+cylncnt);
       lastz=fin2(3+cylncnt);
       ang2=x2(4+cylncnt);
       a=radx;
       b=rady;
       a2=radx^2;
       b2=rady^2;
%       delz=abs(lastz-firstz);
       cnt2;
%
% The vrt matrix is a transformation matrix that can be used to rotate
the cylinder
% about the initial set of axis to a new postion in the xyz plane used
by the plates.
% The cylinder is initially a z directed cylinder. This means that the
axis of the
% cylinder was parallel to the z axis.
%
       vrt(3,1)=sin(angz1*rtd)*cos(angz2*rtd);
       vrt(3,2)=sin(angz1*rtd)*sin(angz2*rtd);
       vrt(3,3)=cos(angz1*rtd);
       temp(1)=vrt(3,1);
       temp(2)=vrt(3,2);
       temp(3)=vrt(3,3);
       dot=temp*temp';
       mag1=abs(dot);
       dot=sqrt(mag1);
       vrt(3,1)=temp(1)/dot;
       vrt(3,2)=temp(2)/dot;
       vrt(3,3)=temp(3)/dot;
       vrt(1,1)=sin(angx1*rtd)*cos(angx2*rtd);
       vrt(1,2)=sin(angx1*rtd)*sin(angx2*rtd);
       vrt(1,3)=cos(angx1*rtd);
       temp(1)=vrt(1,1);
       temp(2)=vrt(1,2);
       temp(3)=vrt(1,3);
       dot=temp*temp';
       mag1=abs(dot);
       dot=sqrt(mag1);
       vrt(1,1)=temp(1)/dot;
       vrt(1,2)=temp(2)/dot;
       vrt(1,3)=temp(3)/dot;
       vrt(2,1)=(vrt(3,2)*vrt(1,3)-vrt(3,3)*vrt(1,2));
       vrt(2,2)=(vrt(3,3)*vrt(1,1)-vrt(3,1)*vrt(1,3));
       vrt(2,3)=(vrt(3,1)*vrt(1,2)-vrt(3,2)*vrt(1,1));
       vrt2=inv(vrt);
%
%  generating cylinder loop
%
       for cnt4=1:60
         delta1=2*pi/60;
         temp1=(sin(phi))^2;
         temp2=(cos(phi))^2;
         phi2=phi2+delta1;
         temp3=(sin(phi2))^2;
         temp4=(cos(phi2))^2;
%
```

```
%  starting the generation of the rectangle approximation of cylinder
surface.
%
         r1=a*b/sqrt(a2*temp1+b2*temp2);
         r2=a*b/sqrt(a2*temp3+b2*temp4);
         x1b=0.998*r1*cos(phi);
         y1b=0.998*r1*sin(phi);
         x2b=0.998*r2*cos(phi2);
         y2b=0.998*r2*sin(phi2);
% testing
      donx1(cnt4)=x1b;
        dony1(cnt4)=y1b;
      donx2(cnt4)=x2b;
        dony2(cnt4)=y2b;
         xo(1)=xpos+x1b;
         yo(1)=ypos+y1b;
         zo(1)=zpos+firstz;

         xo(2)=xo(1);
         yo(2)=yo(1);
         zo(2)=zpos+lastz;
         xo(3)=xpos+x2b;
         yo(3)=ypos+y2b;
         zo(3)=zo(2);
         xo(4)=xo(3);
         yo(4)=yo(3);
         zo(4)=zo(1);
         xo(5)=xo(1);
         yo(5)=yo(1);
         zo(5)=zo(1);
%
% Here the rotation will take place.
%
         for cnt5=1:5
           temp(1)=xo(cnt5);
           temp(2)=yo(cnt5);
           temp(3)=zo(cnt5);
           newcoord=vrt2*temp';
           xo(cnt5)=newcoord(1);
           yo(cnt5)=newcoord(2);
           zo(cnt5)=newcoord(3);
         end
%
% rotation ends
%
         fill3(xo,yo,zo,'b')
         phi=phi+delta1;
      end
     end
end
%
%-----------------------------------------------------------------
%  Generating the source. This section will create a Figure for the
%  excitation dipole(s). The input will be from the file heading.dat
%-----------------------------------------------------------------
%
 load heading.dat
 flag=heading(:,1);
 cnt5=size(flag,1);
 xs=heading(:,2);
 ys=heading(:,3);
 zs=heading(:,4);
 srclen=heading(:,5);
%
% this will draw a yellow line of the same length as the length of the
dipole
%
% uncomment this section if you want this form for the source.
%
 for cnt6=1:cnt5
   zs1(1)=zs(cnt6)-srclen/2;
   zs1(2)=zs(cnt6)+srclen/2;
   xs1(1)=xs(cnt6);
   xs1(2)=xs(cnt6);
   ys1(1)=ys(cnt6);
   ys1(2)=ys(cnt6);
   plot3(xs1,ys1,zs1,'y')
 end
%
%
%
*********************************************************************
% The following routine will generate a solid  to represent the source
% as a phase center. This is a simple modification of the sphere
routine
% used with matlab. Right now the size of the source is 0.5 meters. It
% can be changed by altering sizsrc to some other value.
%
*********************************************************************
%
%  sizsrc=0.5;
%  colormap(hot)
%  brighten(1)
%  for cnt6=1:cnt5
%   n=4;
%   xp=xs(cnt6)*ones(n+1,n+1);
%   yp=ys(cnt6)*ones(n+1,n+1);
%   zp=zs(cnt6)*ones(n+1,n+1);
%   theta3=(-n:2:n)/n*pi;
%   phi3=(-n:2:n)'/n*pi/2;
%   cosphi=cos(phi3);
%   cosphi(1)=0;
%   cosphi(n+1)=0;
%   sintheta=sin(theta3);
%   sintheta(1)=0;
%   sintheta(n+1)=0;
%   xs1=sizsrc*cosphi*cos(theta3)+xp;
%   ys1=sizsrc*cosphi*sintheta+yp;
%   zs1=sizsrc*sin(phi3)*ones(1,n+1)+zp;
%  surf(xs1,ys1,zs1)

% end
 %colormap(hsv)
 xlabel('X')
 ylabel('Y')
 zlabel('Z')
 view(3)
 grid

%
% now to delete the scratch files.
%
%!rm -f geommat.dat cylnmat.dat heading.dat
%
% free up the variables so that there will be no overlap.
%
%clear
%
% this version was completed october 4 1993
%
axis(coeff*[-20,20,-20,20,-20,20])
% if ray tracing is required uncomment the following section of
% code
%
% now call the axis sizing routine
%
%v1
% if you have the v2.m file and want to show movies of the Figure.
uncomment
% the v2 command or call it from matlab.
%v2
%
% if you have the v3.m file and want to show the three views of the
Figure.
% uncomment the v3 command or call it from matlab
% v3
view(3)
v4
```

BSV1B.M MATLAB source code

```
% NECBSC2 previewer. This file will take any file in the standard bsc
% format and convert it to a 3-D graphic that can be printed or viewed
% as you choose.
%
%       This previewer was written by Don Davis June 12 1993
%
%     This previewer version was revised October 14 1993 by:
```

```
%
%                  Don Davis
%            Concordia University
%
% ======================================================================
% geommat is a matrix that will contain all the plate data required
% to draw a plate for this previewer.
%
% cylnmat is a matrix that will contain all the cylinder data required
% to draw a cylinder for this previewer.
%
% heading will contain the source data used to place the sources.
%
% ray.dat will hold the ray tracing data.
%
% raytyp.dat will hold the data on what ray mechansim it is ,eg.
relected.
%
% parts will contain the information on which parts are present. These
% can be either cylinders or plates.
% =====================================================================
%
% This file will be used to generate pictures of the inbsc.dat format
% file. The input file will be chosen by the user. This data will be in
a
% matrix form. To run this invoke the preprocessor by typing bsv.
% The preprocessor should be compiled as follows. f77 bsv.f -o bsv.
%
% ====================================================================
%clear
%
% now to call the preprocessor which is a fortran program
%
!bsv
%
% clear any previous graphics
%
clg
%
%set some constants
rtd=pi/180;
ref=1;
adj=0;
load coeff.dat
hold
%
% check for what parts are present
%
load parts.dat
plat=parts(:,1);
cylin=parts(:,2);
%
%
```

```
% are plates present ?
if plat>0
%
% Load plate data in matrix form.
load geommat.dat
flag=geommat(:,1);
x=geommat(:,2);
y=geommat(:,3);
z=geommat(:,4);
fin=geommat(:,5);
cnt1=size(flag,1);
%
% rtd=radians to degrees conversion
%
rtd=pi/180;
ref=1;
adj=0;


%-----------------------------------------------------------------------
%  This is the section of the program which will use the data for the
%  plates to draw the shapes in three space using fill3 command.
%  the rotation of the plates will have been done by the preprocesor
%  the case of the plates.
%  flag contains the number of corners that the plate has. The loop
%  will cycle through each of the points to make a closed contour of
%  the shape and use fill to color in the polygon. Cnt1 is the number
%  of plates.
%-----------------------------------------------------------------------
--
%
%
%  cnt: a dummy variable used to cycle through the plates.
%  chngplt: controls the change from one plate to the next one.
%  cnt3: a dummy variable used to count through the points on the
plate.
%
%
=======================================================================
=
%

     for cnt=1:cnt1,
       chngplt=flag(cnt)+fin(cnt);
       if chngplt== 1
          cnt2=cnt2+1;
          cnt3=flag(cnt)+cnt;

          fill3(x(cnt3:cnt),y(cnt3:cnt),z(cnt3:cnt),'g')
       end
     end
   end
```

```
%     pause




     end
%   end of the flat plate section.
end
%
%
% are cylinders present?
if cylin>0
% -------------------------------------------------------------------
--
%   cylnmat.dat is a NX5 matrix containing data on the cylnders.
%   There are N cylinders. Each cylinder is formed from a set of
rectangles
%   that are parallel to the main axis. These rectangles are made from
%   chords on the base of the cylinder. The base is elliptic or
circular.
%   The rotation is done here in the case of the cylinder.
% -------------------------------------------------------------------
-
% clear
%
% Load cylinder data in matrix form
%
%
%
%   cnt2: an index variable used to move through the cylinder data.
%   numcyl: the number of cylinders being considered.
%   x2: cylinder data
%   y2: cylinder data
%   z2: cylinder data
%   fin2: cylinder data
%   xpos,ypos,zpos: x,y,z position of center of cylinder.
%   radx,rady: major & minor axis. not necessarlily in that order.
%   ang*: the angles that define the cylinder coord. system.
%
=====================================================================
==
%
%
load cylnmat.dat
flag2=cylnmat(:,1);
cnt2=size(flag2,1);
numcyl=cnt2/4;
%
% numcyl is the number of cylinders that will be drawn
%
x2=cylnmat(:,2);
```

```
y2=cylnmat(:,3);
z2=cylnmat(:,4);
fin2=cylnmat(:,5);
     for cnt2=1:numcyl
       phi=0;
       phi2=0;
       cylncnt=(cnt2-1)*4;
       xpos=x2(1+cylncnt);
       ypos=y2(1+cylncnt);
       zpos=z2(1+cylncnt);
       angz1=fin2(1+cylncnt);
       angz2=x2(2+cylncnt);
       angx1=y2(2+cylncnt);
       angx2=z2(2+cylncnt);
       radx=fin2(2+cylncnt);
       rady=x2(3+cylncnt);
       firstz=y2(3+cylncnt);
       ang1=z2(3+cylncnt);
       lastz=fin2(3+cylncnt);
       ang2=x2(4+cylncnt);
       a=radx;
       b=rady;
       a2=radx^2;
       b2=rady^2;
%       delz=abs(lastz-firstz);
       cnt2;
%
% The vrt matrix is a transformation matrix that can be used to rotate
the cylinder
% about the initial set of axis to a new postion in the xyz plane used
by the plates.
% The cylinder is initially a z directed cylinder. This means that the
axis of the
% cylinder was parallel to the z axis.
%
       vrt(3,1)=sin(angz1*rtd)*cos(angz2*rtd);
       vrt(3,2)=sin(angz1*rtd)*sin(angz2*rtd);
       vrt(3,3)=cos(angz1*rtd);
       temp(1)=vrt(3,1);
       temp(2)=vrt(3,2);
       temp(3)=vrt(3,3);
       dot=temp*temp';
       mag1=abs(dot);
       dot=sqrt(mag1);
       vrt(3,1)=temp(1)/dot;
       vrt(3,2)=temp(2)/dot;
       vrt(3,3)=temp(3)/dot;
       vrt(1,1)=sin(angx1*rtd)*cos(angx2*rtd);
       vrt(1,2)=sin(angx1*rtd)*sin(angx2*rtd);
       vrt(1,3)=cos(angx1*rtd);
       temp(1)=vrt(1,1);
       temp(2)=vrt(1,2);
       temp(3)=vrt(1,3);
       dot=temp*temp';
       mag1=abs(dot);
       dot=sqrt(mag1);
       vrt(1,1)=temp(1)/dot;
       vrt(1,2)=temp(2)/dot;
       vrt(1,3)=temp(3)/dot;
       vrt(2,1)=(vrt(3,2)*vrt(1,3)-vrt(3,3)*vrt(1,2));
       vrt(2,2)=(vrt(3,3)*vrt(1,1)-vrt(3,1)*vrt(1,3));
       vrt(2,3)=(vrt(3,1)*vrt(1,2)-vrt(3,2)*vrt(1,1));
       vrt2=inv(vrt);
%
%  generating cylinder loop
%
       for cnt4=1:60
         delta1=2*pi/60;
         temp1=(sin(phi))^2;
         temp2=(cos(phi))^2;
         phi2=phi2+delta1;
         temp3=(sin(phi2))^2;
         temp4=(cos(phi2))^2;
%
%  starting the generation of the rectangle approximation of cylinder
surface.
%
         r1=a*b/sqrt(a2*temp1+b2*temp2);
         r2=a*b/sqrt(a2*temp3+b2*temp4);
         x1b=0.998*r1*cos(phi);
         y1b=0.998*r1*sin(phi);
         x2b=0.998*r2*cos(phi2);
         y2b=0.998*r2*sin(phi2);
% testing
      donx1(cnt4)=x1b;
        dony1(cnt4)=y1b;
      donx2(cnt4)=x2b;
        dony2(cnt4)=y2b;
         xo(1)=xpos+x1b;
         yo(1)=ypos+y1b;
         zo(1)=zpos+firstz;

         xo(2)=xo(1);
         yo(2)=yo(1);
         zo(2)=zpos+lastz;
         xo(3)=xpos+x2b;
         yo(3)=ypos+y2b;
         zo(3)=zo(2);
         xo(4)=xo(3);
         yo(4)=yo(3);
         zo(4)=zo(1);
         xo(5)=xo(1);
         yo(5)=yo(1);
         zo(5)=zo(1);
%
% Here the rotation will take place.
```

```
%
         for cnt5=1:5
           temp(1)=xo(cnt5);
           temp(2)=yo(cnt5);
           temp(3)=zo(cnt5);
           newcoord=vrt2*temp';
           xo(cnt5)=newcoord(1);
           yo(cnt5)=newcoord(2);
           zo(cnt5)=newcoord(3);
         end
%
% rotation ends
%
         fill3(xo,yo,zo,'b')
         phi=phi+delta1;
      end
     end
end
%
%-----------------------------------------------------------------
%  Generating the source. This section will create a Figure for the
%  excitation dipole(s). The input will be from the file heading.dat
%-----------------------------------------------------------------
%
 load heading.dat
 flag=heading(:,1);
 cnt5=size(flag,1);
 xs=heading(:,2);
 ys=heading(:,3);
 zs=heading(:,4);
 srclen=heading(:,5);
%
% this will draw a yellow line of the same length as the length of the
dipole
%
% uncomment this section if you want this form for the source.
%
for cnt6=1:cnt5
   zs1(1)=zs(cnt6)-srclen/2;
   zs1(2)=zs(cnt6)+srclen/2;
   xs1(1)=xs(cnt6);
   xs1(2)=xs(cnt6);
   ys1(1)=ys(cnt6);
   ys1(2)=ys(cnt6);
   plot3(xs1,ys1,zs1,'y')
 end
 grid
%
%
%
% now to delete the scratch files.
%
%!rm -f geommat.dat cylnmat.dat heading.dat
```

```
%
% free up the variables so that there will be no overlap.
%
%clear
%
% this version was completed october 4 1993
%
axis(coeff*[-20,20,-20,20,-20,20])
% if ray tracing is required uncomment the following section of
% code
load ray.dat
cnt7=size(ray,1);
rayx1=ray(:,1);
rayy1=ray(:,2);
rayz1=ray(:,3);
rayx2=ray(:,4);
rayy2=ray(:,5);
rayz2=ray(:,6);
load raytyp.dat
%
% draw one ray at a time
%
for cnt8=1:cnt7
    rx(1)=rayx1(cnt8);
    ry(1)=rayy1(cnt8);
    rz(1)=rayz1(cnt8);
    rx(2)=rayx2(cnt8);
    ry(2)=rayy2(cnt8);
    rz(2)=rayz2(cnt8);
%
% color code the rays according to mechanism.
%
%
% direct ray
%
   if raytyp(cnt8)==1  plot3(rx,ry,rz,'y')
   end
%
% reflected ray
%
   if raytyp(cnt8)==2  plot3(rx,ry,rz,'c')
   end
%
% diffracted ray
%
   if raytyp(cnt8)==3  plot3(rx,ry,rz,'m')
   end
%
% diff-ref or ref-diff
%
   if raytyp(cnt8)==4  plot3(rx,ry,rz,'r')
   end
%
```

```
% creeping wave
%
   if raytyp(cnt8)==5  plot3(rx,ry,rz,'w')
   end
end
hold
% delete the ray.dat file.
%!rm -f ray.dat raytyp.dat
% end of the ray tracing section
%
% now call the axis sizing routine
%
%v1
% if you have the v2.m file and want to show movies of the Figure.
uncomment
% the v2 command or call it from matlab.
%v2
%
% if you have the v3.m file and want to show the three views of the
Figure.
% uncomment the v3 command or call it from matlab
% v3
view(3)
v4
```

BSV1C.M MATLAB source code

```
% NECBSC2 previewer. This file will take any file in the standard bsc
% format and convert it to a 3-D graphic that can be printed or viewed
% as you choose.
%
%       This previewer was written by Don Davis June 12 1993
%
%     This previewer version was revised October 14 1993 by:
%
%                   Don Davis
%             Concordia University
%
% ====================================================================
% geommat is a matrix that will contain all the plate data required
% to draw a plate for this previewer.
%
% cylnmat is a matrix that will contain all the cylinder data required
% to draw a cylinder for this previewer.
%
% heading will contain the source data used to place the sources.
%
% ray.dat will hold the ray tracing data.
%
% raytyp.dat will hold the data on what ray mechansim it is ,eg.
relected.
%
% parts will contain the information on which parts are present. These
% can be either cylinders or plates.
% ====================================================================
%
% This file will be used to generate pictures of the inbsc.dat format
% file. The input file will be chosen by the user. This data will be in
a
% matrix form. To run this invoke the preprocessor by typing bsv.
% The preprocessor should be compiled as follows. f77 bsv.f -o bsv.
%
% ====================================================================
%clear
%
% now to call the preprocessor which is a fortran program
%
!bsv
%
% clear any previous graphics
%
clg
axes('position',[0.55 0.05 .4 .03])
pcolor([1:50;1:50])
axis('off')
title('Magnitude')
ylabel('Max')
axis([0 48 0 2])
axes('position',[.1 .2 .8 .75])
%
%set some constants
rtd=pi/180;
ref=1;
adj=0;
load coeff.dat
hold
%
% check for what parts are present
%
load parts.dat
plat=parts(:,1);
cylin=parts(:,2);
%
%
% are plates present ?
if plat>0
%
% Load plate data in matrix form.
load geommat.dat
flag=geommat(:,1);
x=geommat(:,2);
y=geommat(:,3);
z=geommat(:,4);
fin=geommat(:,5);
cnt1=size(flag,1);
%
```

```
% rtd=radians to degrees conversion
%
rtd=pi/180;
ref=1;
adj=0;

%------------------------------------------------------------------
%  This is the section of the program which will use the data for the
%  plates to draw the shapes in three space using fill3 command.
%  the rotation of the plates will have been done by the preprocesor
%  the case of the plates.
%  flag contains the number of corners that the plate has. The loop
%  will cycle through each of the points to make a closed contour of
%  the shape and use fill to color in the polygon. Cnt1 is the number
%  of plates.
%------------------------------------------------------------------
--
%
%
%  cnt: a dummy variable used to cycle through the plates.
%  chngplt: controls the change from one plate to the next one.
%  cnt3: a dummy variable used to count through the points on the
plate.
%
%
==================================================================
=
%

     for cnt=1:cnt1,
       chngplt=flag(cnt)+fin(cnt);
       if chngplt== 1
          cnt2=cnt2+1;
          cnt3=flag(cnt)+cnt;

          fill3(x(cnt3:cnt),y(cnt3:cnt),z(cnt3:cnt),'w')
       end
     end
    end

%     pause




     end
%   end of the flat plate section.
end
%
%
% are cylinders present?
```

```
if cylin>0
% -----------------------------------------------------------------
--
%   cylnmat.dat is a NX5 matrix containing data on the cylnders.
%   There are N cylinders. Each cylinder is formed from a set of
rectangles
%   that are parallel to the main axis. These rectangles are made from
%   chords on the base of the cylinder. The base is elliptic or
circular.
%   The rotation is done here in the case of the cylinder.
% -----------------------------------------------------------------
-
% clear
%
% Load cylinder data in matrix form
%
%
%
%   cnt2: an index variable used to move through the cylinder data.
%   numcyl: the number of cylinders being considered.
%   x2: cylinder data
%   y2: cylinder data
%   z2: cylinder data
%   fin2: cylinder data
%   xpos,ypos,zpos: x,y,z position of center of cylinder.
%   radx,rady: major & minor axis. not necessarlily in that order.
%   ang*: the angles that define the cylinder coord. system.
%
==================================================================
==
%
%
load cylnmat.dat
flag2=cylnmat(:,1);
cnt2=size(flag2,1);
numcyl=cnt2/4;
%
% numcyl is the number of cylinders that will be drawn
%
x2=cylnmat(:,2);
y2=cylnmat(:,3);
z2=cylnmat(:,4);
fin2=cylnmat(:,5);
     for cnt2=1:numcyl
       phi=0;
       phi2=0;
       cylncnt=(cnt2-1)*4;
       xpos=x2(1+cylncnt);
       ypos=y2(1+cylncnt);
       zpos=z2(1+cylncnt);
       angz1=fin2(1+cylncnt);
       angz2=x2(2+cylncnt);
       angx1=y2(2+cylncnt);
```

```
        angx2=z2(2+cylncnt);
        radx=fin2(2+cylncnt);
        rady=x2(3+cylncnt);
        firstz=y2(3+cylncnt);
        ang1=z2(3+cylncnt);
        lastz=fin2(3+cylncnt);
        ang2=x2(4+cylncnt);
        a=radx;
        b=rady;
        a2=radx^2;
        b2=rady^2;
%        delz=abs(lastz-firstz);
        cnt2;
%
% The vrt matrix is a transformation matrix that can be used to rotate
the cylinder
% about the initial set of axis to a new postion in the xyz plane used
by the plates.
% The cylinder is initially a z directed cylinder. This means that the
axis of the
% cylinder was parallel to the z axis.
%
        vrt(3,1)=sin(angz1*rtd)*cos(angz2*rtd);
        vrt(3,2)=sin(angz1*rtd)*sin(angz2*rtd);
        vrt(3,3)=cos(angz1*rtd);
        temp(1)=vrt(3,1);
        temp(2)=vrt(3,2);
        temp(3)=vrt(3,3);
        dot=temp*temp';
        mag1=abs(dot);
        dot=sqrt(mag1);
        vrt(3,1)=temp(1)/dot;
        vrt(3,2)=temp(2)/dot;
        vrt(3,3)=temp(3)/dot;
        vrt(1,1)=sin(angx1*rtd)*cos(angx2*rtd);
        vrt(1,2)=sin(angx1*rtd)*sin(angx2*rtd);
        vrt(1,3)=cos(angx1*rtd);
        temp(1)=vrt(1,1);
        temp(2)=vrt(1,2);
        temp(3)=vrt(1,3);
        dot=temp*temp';
        mag1=abs(dot);
        dot=sqrt(mag1);
        vrt(1,1)=temp(1)/dot;
        vrt(1,2)=temp(2)/dot;
        vrt(1,3)=temp(3)/dot;
        vrt(2,1)=(vrt(3,2)*vrt(1,3)-vrt(3,3)*vrt(1,2));
        vrt(2,2)=(vrt(3,3)*vrt(1,1)-vrt(3,1)*vrt(1,3));
        vrt(2,3)=(vrt(3,1)*vrt(1,2)-vrt(3,2)*vrt(1,1));
        vrt2=inv(vrt);
%
%  generating cylinder loop
%
```

```
        for cnt4=1:60
          delta1=2*pi/60;
          temp1=(sin(phi))^2;
          temp2=(cos(phi))^2;
          phi2=phi2+delta1;
          temp3=(sin(phi2))^2;
          temp4=(cos(phi2))^2;
%
%  starting the generation of the rectangle approximation of cylinder
surface.
%
          r1=a*b/sqrt(a2*temp1+b2*temp2);
          r2=a*b/sqrt(a2*temp3+b2*temp4);
          x1b=0.998*r1*cos(phi);
          y1b=0.998*r1*sin(phi);
          x2b=0.998*r2*cos(phi2);
          y2b=0.998*r2*sin(phi2);
% tasting
      donx1(cnt4)=x1b;
        dony1(cnt4)=y1b;
      donx2(cnt4)=x2b;
        dony2(cnt4)=y2b;
          xo(1)=xpos+x1b;
          yo(1)=ypos+y1b;
          zo(1)=zpos+firstz;

          xo(2)=xo(1);
          yo(2)=yo(1);
          zo(2)=zpos+lastz;
          xo(3)=xpos+x2b;
          yo(3)=ypos+y2b;
          zo(3)=zo(2);
          xo(4)=xo(3);
          yo(4)=yo(3);
          zo(4)=zo(1);
          xo(5)=xo(1);
          yo(5)=yo(1);
          zo(5)=zo(1);
%
% Here the rotation will take place.
%
          for cnt5=1:5
            temp(1)=xo(cnt5);
            temp(2)=yo(cnt5);
            temp(3)=zo(cnt5);
            newcoord=vrt2*temp';
            xo(cnt5)=newcoord(1);
            yo(cnt5)=newcoord(2);
            zo(cnt5)=newcoord(3);
          end
%
% rotation ends
%
```

```
            fill3(xo,yo,zo,'w')
            phi=phi+delta1;
         end
       end
end
%
%----------------------------------------------------------------------
%  Generating the source. This section will create a Figure for the
%  excitation dipole(s). The input will be from the file heading.dat
%----------------------------------------------------------------------
%
 load heading.dat
 flag=heading(:,1);
 cnt5=size(flag,1);
 xs=heading(:,2);
 ys=heading(:,3);
 zs=heading(:,4);
 srclen=heading(:,5);
%
% this will draw a yellow line of the same length as the length of the
dipole
%
% uncomment this section if you want this form for the source.
%
for cnt6=1:cnt5
   zs1(1)=zs(cnt6)-srclen/2;
   zs1(2)=zs(cnt6)+srclen/2;
   xs1(1)=xs(cnt6);
   xs1(2)=xs(cnt6);
   ys1(1)=ys(cnt6);
   ys1(2)=ys(cnt6);
   plot3(xs1,ys1,zs1,'y')
 end
 grid
%
%
%
% now to delete the scratch files.
%
%!rm -f geommat.dat cylnmat.dat heading.dat
%
% free up the variables so that there will be no overlap.
%
%clear
%
% this version was completed october 4 1993
%
axis(coeff*[-20,20,-20,20,-20,20])
% if ray tracing is required uncomment the following section of
% code
load ray.dat
cnt7=size(ray,1);
rayx1=ray(:,1);
```

```
rayy1=ray(:,2);
rayz1=ray(:,3);
rayx2=ray(:,4);
rayy2=ray(:,5);
rayz2=ray(:,6);
%
%
%  magnitude information for color coding
%
%
load raymag.dat
for cnt12=1:cnt7
  if raymag(cnt12)== 0
     raymag(cnt12)=0.0001;
  end
end
raymag1=log10(raymag);
m1=max(raymag1);
m2=min(raymag1);
colormap=(hsv);
x1=hsv;
delmag=(m1-m2)/52;

%
% draw one ray at a time
%
cnt12=0;
for cnt8=1:cnt7
      cnt12=cnt12+1;
    rx(1)=rayx1(cnt8);
    ry(1)=rayy1(cnt8);
    rz(1)=rayz1(cnt8);
    rx(2)=rayx2(cnt8);
    ry(2)=rayy2(cnt8);
    rz(2)=rayz2(cnt8);
%
% color code the rays according to magnitude
%
%
    if raymag1(cnt8) >= 0
       posmag=0;
       normag=raymag1(cnt8)/m1;
    end
    if raymag1(cnt8) < 0
       posmag=52;
       normag=raymag1(cnt8)/m2;
    end
    maxmag=1;
    cnt12=0;
    delnorm=1/52;
    diffmag=1;
    while diffmag > 0
        cnt12=cnt12+1;
        diffmag=maxmag-normag;
        maxmag=maxmag-delnorm;
    end
    plot3(rx,ry,rz,'Color',x1(abs(posmag-cnt12),:))



% ----------------------------old spectrum-----------------
%     n1=0;
%    n1=2*normag1(cnt8)*(1-normag1(cnt8));
%    if raymag1(cnt8) >= 0
%      normag=raymag1(cnt8)/m1;
%      n1=abs(normag);
%      n1=3*n1*(1-n1)*(n1+(1-n1));
%      h=plot3(rx,ry,rz);
%      set(h,'Color',[normag,n1,1-normag1)
%    end
%    if raymag1(cnt8) < 0
%      normag=raymag1(cnt8)/m2;
%      n1=abs(normag);
%      n1=3*n1*(1-n1)*(n1+(1-n1));
%      h=plot3(rx,ry,rz);
%      set(h,'Color',[1-normag,n1,normag])
%    end
% ------------------------------------------------------------
end
hold
% delete the ray.dat file.
%!rm -f ray.dat raytyp.dat
% end of the ray tracing section
%
% now call the axis sizing routine
%
%v1
% if you have the v2.m file and want to show movies of the Figure.
uncomment
% the v2 command or call it from matlab.
%v2
%
% if you have the v3.m file and want to show the three views of the
Figure.
% uncomment the v3 command or call it from matlab
% v3
view(3)
v4
```

BSV1D.M MATLAB source code

```
% NECBSC2 previewer. This file will take any file in the standard bsc
% format and convert it to a 3-D graphic that can be printed or viewed
% as you choose.
%
```

```
%       This previewer was written by Don Davis June 12 1993
%
%     This previewer version was revised October 14 1993 by:
%
%                  Don Davis
%             Concordia University
%
% ===================================================================
% geommat is a matrix that will contain all the plate data required
% to draw a plate for this previewer.
%
% cylnmat is a matrix that will contain all the cylinder data required
% to draw a cylinder for this previewer.
%
% heading will contain the source data used to place the sources.
%
% ray.dat will hold the ray tracing data.
%
% raytyp.dat will hold the data on what ray mechansim it is ,eg.
relected.
%
% parts will contain the information on which parts are present. These
% can be either cylinders or plates.
% ===================================================================
%
% This file will be used to generate pictures of the inbsc.dat format
% file. The input file will be chosen by the user. This data will be in
a
% matrix form. To run this invoke the preprocessor by typing bsv.
% The preprocessor should be compiled as follows. f77 bsv.f -o bsv.
%
% ===================================================================
%clear
%
% now to call the preprocessor which is a fortran program
%
!bsv
%
% clear any previous graphics
%
clg
%
%set some constants
rtd=pi/180;
ref=1;
adj=0;
load coeff.dat
hold
%
% check for what parts are present
%
load parts.dat
plat=parts(:,1);
```

```
cylin=parts(:,2);
%
%
% are plates present ?
if plat>0
%
% Load plate data in matrix form.
load geommat.dat
flag=geommat(:,1);
x=geommat(:,2);
y=geommat(:,3);
z=geommat(:,4);
fin=geommat(:,5);
cnt1=size(flag,1);
%
% rtd=radians to degrees conversion
%
rtd=pi/180;
ref=1;
adj=0;


%-----------------------------------------------------------------------
% This is the section of the program which will use the data for the
% plates to draw the shapes in three space using fill3 command.
% the rotation of the plates will have been done by the preprocesor
% the case of the plates.
% flag contains the number of corners that the plate has. The loop
% will cycle through each of the points to make a closed contour of
% the shape and use fill to color in the polygon. Cnt1 is the number
% of plates.
%-----------------------------------------------------------------------
--
%
%
% cnt: a dummy variable used to cycle through the plates.
% chngplt: controls the change from one plate to the next one.
% cnt3: a dummy variable used to count through the points on the
plate.
%
%
=======================================================================
=
%

     for cnt=1:cnt1,
       chngplt=flag(cnt)+fin(cnt);
       if chngplt== 1
          cnt2=cnt2+1;
          cnt3=flag(cnt)+cnt;

          fill3(x(cnt3:cnt),y(cnt3:cnt),z(cnt3:cnt),'g')
       end
     end
    end

%     pause







     end
%   end of the flat plate section.
end
%
%
% are cylinders present?
if cylin>0
% -------------------------------------------------------------------
--
%   cylnmat.dat is a NX5 matrix containing data on the cylnders.
%   There are N cylinders. Each cylinder is formed from a set of
rectangles
%   that are parallel to the main axis. These rectangles are made from
%   chords on the base of the cylinder. The base is elliptic or
circular.
%   The rotation is done here in the case of the cylinder.
% -------------------------------------------------------------------
-
% clear
%
% Load cylinder data in matrix form
%
%
%
%   cnt2: an index variable used to move through the cylinder data.
%   numcyl: the number of cylinders being considered.
%   x2: cylinder data
%   y2: cylinder data
%   z2: cylinder data
%   fin2: cylinder data
%   xpos,ypos,zpos: x,y,z position of center of cylinder.
%   radx,rady: major & minor axis. not necessarlily in that order.
%   ang*: the angles that define the cylinder coord. system.
%
====================================================================
==
%
%
load cylnmat.dat
flag2=cylnmat(:,1);
cnt2=size(flag2,1);
numcyl=cnt2/4;
%
```

```
% numcyl is the number of cylinders that will be drawn
%
x2=cylnmat(:,2);
y2=cylnmat(:,3);
z2=cylnmat(:,4);
fin2=cylnmat(:,5);
     for cnt2=1:numcyl
       phi=0;
       phi2=0;
       cylncnt=(cnt2-1)*4;
       xpos=x2(1+cylncnt);
       ypos=y2(1+cylncnt);
       zpos=z2(1+cylncnt);
       angz1=fin2(1+cylncnt);
       angz2=x2(2+cylncnt);
       angx1=y2(2+cylncnt);
       angx2=z2(2+cylncnt);
       radx=fin2(2+cylncnt);
       rady=x2(3+cylncnt);
       firstz=y2(3+cylncnt);
       ang1=z2(3+cylncnt);
       lastz=fin2(3+cylncnt);
       ang2=x2(4+cylncnt);
       a=radx;
       b=rady;
       a2=radx^2;
       b2=rady^2;
%       delz=abs(lastz-firstz);
       cnt2;
%
% The vrt matrix is a transformation matrix that can be used to rotate
the cylinder
% about the initial set of axis to a new postion in the xyz plane used
by the plates.
% The cylinder is initially a z directed cylinder. This means that the
axis of the
% cylinder was parallel to the z axis.
%
       vrt(3,1)=sin(angz1*rtd)*cos(angz2*rtd);
       vrt(3,2)=sin(angz1*rtd)*sin(angz2*rtd);
       vrt(3,3)=cos(angz1*rtd);
       temp(1)=vrt(3,1);
       temp(2)=vrt(3,2);
       temp(3)=vrt(3,3);
       dot=temp*temp';
       mag1=abs(dot);
       dot=sqrt(mag1);
       vrt(3,1)=temp(1)/dot;
       vrt(3,2)=temp(2)/dot;
       vrt(3,3)=temp(3)/dot;
       vrt(1,1)=sin(angx1*rtd)*cos(angx2*rtd);
       vrt(1,2)=sin(angx1*rtd)*sin(angx2*rtd);
       vrt(1,3)=cos(angx1*rtd);
```

```
        temp(1)=vrt(1,1);
        temp(2)=vrt(1,2);
        temp(3)=vrt(1,3);
        dot=temp*temp';
        mag1=abs(dot);
        dot=sqrt(mag1);
        vrt(1,1)=temp(1)/dot;
        vrt(1,2)=temp(2)/dot;
        vrt(1,3)=temp(3)/dot;
        vrt(2,1)=(vrt(3,2)*vrt(1,3)-vrt(3,3)*vrt(1,2));
        vrt(2,2)=(vrt(3,3)*vrt(1,1)-vrt(3,1)*vrt(1,3));
        vrt(2,3)=(vrt(3,1)*vrt(1,2)-vrt(3,2)*vrt(1,1));
        vrt2=inv(vrt);
%
% generating cylinder loop
%
        for cnt4=1:60
          delta1=2*pi/60;
          temp1=(sin(phi))^2;
          temp2=(cos(phi))^2;
          phi2=phi2+delta1;
          temp3=(sin(phi2))^2;
          temp4=(cos(phi2))^2;
%
% starting the generation of the rectangle approximation of cylinder
surface.
%
          r1=a*b/sqrt(a2*temp1+b2*temp2);
          r2=a*b/sqrt(a2*temp3+b2*temp4);
          x1b=0.998*r1*cos(phi);
          y1b=0.998*r1*sin(phi);
          x2b=0.998*r2*cos(phi2);
          y2b=0.998*r2*sin(phi2);
% testing
       donx1(cnt4)=x1b;
         dony1 (cnt4)=y1b;
       donx2(cnt4)=x2b;
         dony2 (cnt4)=y2b;
          xo(1)=xpos+x1b;
          yo(1)=ypos+y1b;
          zo(1)=zpos+firstz;

          xo(2)=xo(1);
          yo(2)=yo(1);
          zo(2)=zpos+lastz;
          xo(3)=xpos+x2b;
          yo(3)=ypos+y2b;
          zo(3)=zo(2);
          xo(4)=xo(3);
          yo(4)=yo(3);
          zo(4)=zo(1);
          xo(5)=xo(1);
          yo(5)=yo(1);
         zo(5)=zo(1);
%
% Here the rotation will take place.
%
         for cnt5=1:5
           temp(1)=xo(cnt5);
           temp(2)=yo(cnt5);
           temp(3)=zo(cnt5);
           newcoord=vrt2*temp';
           xo(cnt5)=newcoord(1);
           yo(cnt5)=newcoord(2);
           zo(cnt5)=newcoord(3);
         end
%
% rotation ends
%
         fill3(xo,yo,zo,'b')
         phi=phi+delta1;
      end
     end
end
%
%-------------------------------------------------------------------
%  Generating the source. This section will create a Figure for the
%  excitation dipole(s). The input will be from the file heading.dat
%-------------------------------------------------------------------
%
 load heading.dat
 flag=heading(:,1);
 cnt5=size(flag,1);
 xs=heading(:,2);
 ys=heading(:,3);
 zs=heading(:,4);
 srclen=heading(:,5);
%
% this will draw a yellow line of the same length as the length of the
dipole
%
% uncomment this section if you want this form for the source.
%
for cnt6=1:cnt5
   zs1(1)=zs(cnt6)-srclen/2;
   zs1(2)=zs(cnt6)+srclen/2;
   xs1(1)=xs(cnt6);
   xs1(2)=xs(cnt6);
   ys1(1)=ys(cnt6);
   ys1(2)=ys(cnt6);
   plot3(xs1,ys1,zs1,'y')
 end
 grid
%
%
%
```

```
% now to delete the scratch files.
%
%!rm -f geommat.dat cylnmat.dat heading.dat
%
% free up the variables so that there will be no overlap.
%
%clear
%
% this version was completed october 4 1993
%
axis(coeff*[-20,20,-20,20,-20,20])
% if ray tracing is required uncomment the following section of
% code
load ray.dat
cnt7=size(ray,1);
rayx1=ray(:,1);
rayy1=ray(:,2);
rayz1=ray(:,3);
rayx2=ray(:,4);
rayy2=ray(:,5);
rayz2=ray(:,6);
load raytyp.dat
%
% draw one ray at a time
%
for cnt8=1:cnt7
    rx(1)=rayx1(cnt8);
    ry(1)=rayy1(cnt8);
    rz(1)=rayz1(cnt8);
    rx(2)=rayx2(cnt8);
    ry(2)=rayy2(cnt8);
    rz(2)=rayz2(cnt8);
%
% color code the rays according to mechanism.
%
%
% direct ray
%
   if sel1==3
      if raytyp(cnt8)==5
      plot3(rx,ry,rz,'r')
      end
   end
   if raytyp(cnt8)==sel1  plot3(rx,ry,rz,'r')
   end
end
hold
% delete the ray.dat file.
%!rm -f ray.dat raytyp.dat
% end of the ray tracing section
%
% now call the axis sizing routine
%
```

```
%v1
% if you have the v2.m file and want to show movies of the Figure.
uncomment
% the v2 command or call it from matlab.
%v2
%
% if you have the v3.m file and want to show the three views of the
Figure.
% uncomment the v3 command or call it from matlab
% v3
view(3)
v4
```

BSV1E.M MATLAB source code

```
% NECBSC2 previewer. This file will take any file in the standard bsc
% format and convert it to a 3-D graphic that can be printed or viewed
% as you choose.
%
%       This previewer was written by Don Davis June 12 1993
%
%     This previewer version was revised April 16 1994 by:
%
%                    Don Davis
%             Concordia University
%
% ==================================================================
% geommat is a matrix that will contain all the plate data required
% to draw a plate for this previewer.
%
% cylnmat is a matrix that will contain all the cylinder data required
% to draw a cylinder for this previewer.
%
% heading will contain the source data used to place the sources.
%
% ray.dat will hold the ray tracing data.
%
% raytyp.dat will hold the data on what ray mechansim it is ,eg.
relected.
%
% parts will contain the information on which parts are present. These
% can be either cylinders or plates.
% ================================================================
%
% This file will be used to generate pictures of the inbsc.dat format
% file. The input file will be chosen by the user. This data will be in
a
% matrix form. To run this invoke the preprocessor by typing bsv.
% The preprocessor should be compiled as follows. f77 bsv.f -o bsv.
%
% ================================================================
%clear
```

```
%
% now to call the preprocessor which is a fortran program
%
!bsv
%
% clear any previous graphics
%
clg
axes('position',[0.55 0.05 .4 0.03])
pcolor([1:50;1:50])
axis('off')
title('Magnitude')
ylabel('Max')
axes('position',[.1 .2 .8 .75])
%
%set some constants
rtd=pi/180;
ref=1;
adj=0;
load coeff.dat
hold
%
% check for what parts are present
%
load parts.dat
plat=parts(:,1);
cylin=parts(:,2);
%
%
% are plates present ?
if plat>0
%
% Load plate data in matrix form.
load geommat.dat
flag=geommat(:,1);
x=geommat(:,2);
y=geommat(:,3);
z=geommat(:,4);
fin=geommat(:,5);
cnt1=size(flag,1);
%
% rtd=radians to degrees conversion
%
rtd=pi/180;
ref=1;
adj=0;


%----------------------------------------------------------------------
%  This is the section of the program which will use the data for the
%  plates to draw the shapes in three space using fill3 command.
%  the rotation of the plates will have been done by the preprocesor
%  the case of the plates.
%  flag contains the number of corners that the plate has. The loop
%  will cycle through each of the points to make a closed contour of
%  the shape and use fill to color in the polygon. Cnt1 is the number
%  of plates.
%-----------------------------------------------------------------
--
%
%
%  cnt: a dummy variable used to cycle through the plates.
%  chngplt: controls the change from one plate to the next one.
%  cnt3: a dummy variable used to count through the points on the
plate.
%
%
==================================================================
=
%

     for cnt=1:cnt1,
       chngplt=flag(cnt)+fin(cnt);
       if chngplt== 1
          cnt2=cnt2+1;
          cnt3=flag(cnt)+cnt;

          fill3(x(cnt3:cnt),y(cnt3:cnt),z(cnt3:cnt),'w')
       end
     end
    end

%      pause




     end
%   end of the flat plate section.
end
%
%
% are cylinders present?
if cylin>0
% ---------------------------------------------------------------
--
%   cylnmat.dat is a NX5 matrix containing data on the cylnders.
%   There are N cylinders. Each cylinder is formed from a set of
rectangles
%   that are parallel to the main axis. These rectangles are made from
%   chords on the base of the cylinder. The base is elliptic or
circular.
%   The rotation is done here in the case of the cylinder.
% ---------------------------------------------------------------
-
% clear
%
% Load cylinder data in matrix form
%
%
%
%   cnt2: an index variable used to move through the cylinder data.
%   numcyl: the number of cylinders being considered.
%   x2: cylinder data
%   y2: cylinder data
%   z2: cylinder data
%   fin2: cylinder data
%   xpos,ypos,zpos: x,y,z position of center of cylinder.
%   radx,rady: major & minor axis. not necessarlily in that order.
%   ang*: the angles that define the cylinder coord. system.
%
================================================================
==
%
%
load cylnmat.dat
flag2=cylnmat(:,1);
cnt2=size(flag2,1);
numcyl=cnt2/4;
%
% numcyl is the number of cylinders that will be drawn
%
x2=cylnmat(:,2);
y2=cylnmat(:,3);
z2=cylnmat(:,4);
fin2=cylnmat(:,5);
     for cnt2=1:numcyl
       phi=0;
       phi2=0;
       cylncnt=(cnt2-1)*4;
       xpos=x2(1+cylncnt);
       ypos=y2(1+cylncnt);
       zpos=z2(1+cylncnt);
       angz1=fin2(1+cylncnt);
       angz2=x2(2+cylncnt);
       angx1=y2(2+cylncnt);
       angx2=z2(2+cylncnt);
       radx=fin2(2+cylncnt);
       rady=x2(3+cylncnt);
       firstz=y2(3+cylncnt);
       ang1=z2(3+cylncnt);
       lastz=fin2(3+cylncnt);
       ang2=x2(4+cylncnt);
       a=radx·
       b=rady;
       a2=radx^2;
       b2=rady^2;
%       delz=abs(lastz-firstz);
       cnt2;
%
% The vrt matrix is a transformation matrix that can be used to rotate
the cylinder
% about the initial set of axis to a new postion in the xyz plane used
by the plates.
% The cylinder is initially a z directed cylinder. This means that the
axis of the
% cylinder was parallel to the z axis.
%
       vrt(3,1)=sin(angz1*rtd)*cos(angz2*rtd);
       vrt(3,2)=sin(angz1*rtd)*sin(angz2*rtd);
       vrt(3,3)=cos(angz1*rtd);
       temp(1)=vrt(3,1);
       temp(2)=vrt(3,2);
       temp(3)=vrt(3,3);
       dot=temp*temp';
       mag1=abs(dot);
       dot=sqrt(mag1);
       vrt(3,1)=temp(1)/dot;
       vrt(3,2)=temp(2)/dot;
       vrt(3,3)=temp(3)/dot;
       vrt(1,1)=sin(angx1*rtd)*cos(angx2*rtd);
       vrt(1,2)=sin(angx1*rtd)*sin(angx2*rtd);
       vrt(1,3)=cos(angx1*rtd);
       temp(1)=vrt(1,1);
       temp(2)=vrt(1,2);
       temp(3)=vrt(1,3);
       dot=temp*temp';
       mag1=abs(dot);
       dot=sqrt(mag1);
       vrt(1,1)=temp(1)/dot;
       vrt(1,2)=temp(2)/dot;
       vrt(1,3)=temp(3)/dot;
       vrt(2,1)=(vrt(3,2)*vrt(1,3)-vrt(3,3)*vrt(1,2));
       vrt(2,2)=(vrt(3,3)*vrt(1,1)-vrt(3,1)*vrt(1,3));
       vrt(2,3)=(vrt(3,1)*vrt(1,2)-vrt(3,2)*vrt(1,1));
       vrt2=inv(vrt);
%
%  generating cylinder loop
%
       for cnt4=1:60
         delta1=2*pi/60;
         temp1=(sin(phi))^2;
         temp2=(cos(phi))^2;
         phi2=phi2+delta1;
         temp3=(sin(phi2))^2;
         temp4=(cos(phi2))^2;
%
%  starting the generation of the rectangle approximation of cylinder
surface.
```

```
%
        r1=a*b/sqrt(a2*temp1+b2*temp2);
        r2=a*b/sqrt(a2*temp3+b2*temp4);
        x1b=0.998*r1*cos(phi);
        y1b=0.998*r1*sin(phi);
        x2b=0.998*r2*cos(phi2);
        y2b=0.998*r2*sin(phi2);
% testing
      donx1(cnt4)=x1b;
        dony1(cnt4)=y1b;
      donx2(cnt4)=x2b;
        dony2(cnt4)=y2b;
        xo(1)=xpos+x1b;
        yo(1)=ypos+y1b;
        zo(1)=zpos+firstz;

        xo(2)=xo(1);
        yo(2)=yo(1);
        zo(2)=zpos+lastz;
        xo(3)=xpos+x2b;
        yo(3)=ypos+y2b;
        zo(3)=zo(2);
        xo(4)=xo(3);
        yo(4)=yo(3);
        zo(4)=zo(1);
        xo(5)=xo(1);
        yo(5)=yo(1);
        zo(5)=zo(1);
%
% Here the rotation will take place.
%
        for cnt5=1:5
          temp(1)=xo(cnt5);
          temp(2)=yo(cnt5);
          temp(3)=zo(cnt5);
          newcoord=vrt2*temp';
          xo(cnt5)=newcoord(1);
          yo(cnt5)=newcoord(2);
          zo(cnt5)=newcoord(3);
        end
%
% rotation ends
%
        fill3(xo,yo,zo,'w')
        phi=phi+delta1;
      end
     end
end
%
%-------------------------------------------------------------------
%  Generating the source. This section will create a Figure for the
%  excitation dipole(s). The input will be from the file heading.dat
%-------------------------------------------------------------------
```

```
%
 load heading.dat
 flag=heading(:,1);
 cnt5=size(flag,1);
 xs=heading(:,2);
 ys=heading(:,3);
 zs=heading(:,4);
 srclen=heading(:,5);
%
% this will draw a yellow line of the same length as the length of the
dipole
%
% uncomment this section if you want this form for the source.
%
for cnt6=1:cnt5
   zs1(1)=zs(cnt6)-srclen/2;
   zs1(2)=zs(cnt6)+srclen/2;
   xs1(1)=xs(cnt6);
   xs1(2)=xs(cnt6);
   ys1(1)=ys(cnt6);
   ys1(2)=ys(cnt6);
   plot3(xs1,ys1,zs1,'y')
 end
 grid
%
%
%
% now to delete the scratch files.
%
%!rm -f geommat.dat cylnmat.dat heading.dat
%
% free up the variables so that there will be no overlap.
%
%clear
%
% this version was completed october 4 1993
%
axis(coeff*[-20,20,-20,20,-20,20])
% if ray tracing is required uncomment the following section of
% code
load ray.dat
cnt7=size(ray,1);
rayx1=ray(:,1);
rayy1=ray(:,2);
rayz1=ray(:,3);
rayx2=ray(:,4);
rayy2=ray(:,5);
rayz2=ray(:,6);
load raymag.dat
for cnt12=1:cnt7
  if raymag(cnt12)==0
     raymag(cnt12)=0.001;
  end
```

```
end
raymag1=log10(raymag);
m1=max(raymag1);
m2=min(raymag1);
colormap=(hsv(128));
x1=hsv;
load raytyp.dat

%
% draw one ray at a time
%
cnt12=0;
for cnt8=1:cnt7
      cnt12=cnt12+1;
    rx(1)=rayx1(cnt8);
    ry(1)=rayy1(cnt8);
    rz(1)=rayz1(cnt8);
    rx(2)=rayx2(cnt8);
    ry(2)=rayy2(cnt8);
    rz(2)=rayz2(cnt8);
%
% color code the rays according to magnitude
%
%


    if raymag1(cnt8) >= 0
       posmag=0;
       normag=raymag1(cnt8)/m1;
    end
    if raymag1(cnt8) < 0
       posmag=52;
       normag=raymag1(cnt8)/m2;
    end
    maxmag=1;
    cnt12=0;
    delnorm=1/52;
    diffmag=1;
    while diffmag > 0
        cnt12=cnt12+1;
        diffmag=maxmag-normag;
        maxmag=maxmag-delnorm;
    end
    if sel2==3
       if raytyp(cnt8)==5
          plot3(rx,ry,rz,'Color',x1(abs(posmag-cnt12),:))
       end
    end
    if raytyp(cnt8)==sel2
       plot3(rx,ry,rz,'Color',x1(abs(posmag-cnt12),:))
    end
```

```
% --------------------------old spectrum------------------------
%   n1=0;
%   if sel2==3
%      if raytyp(cnt8)==5
%         if raymag1(cnt8) >= 0
%            normag=raymag1(cnt8)/m1;
%            n1=abs(normag);
%            n1=3*n1*(1-n1)*(n1+(1-n1));
%            h=plot3(rx,ry,rz);
%            set(h,'Color',[normag,n1,1-normag])
%         end
%         if raymag1(cnt8) < 0
%            normag=raymag1(cnt8)/m2;
%            n1=abs(normag);
%            n1=3*n1*(1-n1)*(n1+(1-n1));
%            h=plot3(rx,ry,rz);
%            set(h,'Color',[1-normag,n1,normag])
%         end
%      end
%   end
%if raytyp(cnt8)==sel2
%    if raymag1(cnt8) >= 0
%       normag=raymag1(cnt8)/m1;
%       h=plot3(rx,ry,rz);
%       set(h,'Color',[normag,n1,1-normag])
%    end
%    if raymag1(cnt8) <0
%       normag=raymag1(cnt8)/m2;
%       h=plot3(rx,ry,rz);
%       set(h,'Color',[1-normag,n1,normag])
%    end
end
end
hold
% delete the ray.dat file.
%!rm -f ray.dat raytyp.dat
% end of the ray tracing section
%
% now call the axis sizing routine
%
%v1
% if you have the v2.m file and want to show movies of the Figure.
uncomment
% the v2 command or call it from matlab.
%v2
%
% if you have the v3.m file and want to show the three views of the
Figure.
% uncomment the v3 command or call it from matlab
% v3
view(3)
v4
```

THETHEF.M MATLAB Source Code

```
clg
load fort.8
index1=size(fort);
thepat=fort(:,1);
thetadb=fort(:,5);
plot(thepat,thetadb)
```

THEPHIF.M MATLAB Source Code

```
clg
load fort.8
index1=size(fort);
thepat=fort(:,1);
thetadb=fort(:,9);
plot(thepat,thetadb)
```

PHIPHIF.M MATLAB Source Code

```
clg
load fort.8
index1=size(fort);
thepat=fort(:,2);
thetadb=fort(:,9);
plot(thepat,thetadb)
```

PHITHEF.M MATLAB Source Code

```
clg
load fort.8
index1=size(fort);
thepat=fort(:,2);
thetadb=fort(:,5);
plot(thepat,thetadb)
```

XEX.M MATLAB Source Code

```
clg
```

```
load fort.8
rpat=fort(:,1);
rdb=fort(:,6);
plot(rpat,rdb)
```

XEY.M MATLAB Source Code

```
clg
load fort.8
rpat=fort(:,1);
rdb=fort(:,9);
plot(rpat,rdb)
```

XEZ.M MATLAB Source Code

```
clg
load fort.8
rpat=fort(:,1);
rdb=fort(:,12);
plot(rpat,rdb)
```

YEX.M MATLAB Source Code

```
clg
load fort.8
rpat=fort(:,2);
rdb=fort(:,6);
plot(rpat,rdb)
```

YEY.M MATLAB Source Code

```
clg
load fort.8
rpat=fort(:,2);
rdb=fort(:,9);
plot(rpat,rdb)
```

YEZ.M MATLAB Source Code

```
clg
load fort.8
rpat=fort(:,2);
rdb=fort(:,12);
plot(rpat,rdb)
```

ZEX.M MATLAB Source Code

```
clg
load fort.8
rpat=fort(:,3);
rdb=fort(:,6);
plot(rpat,rdb)
```

ZEY.M MATLAB Source Code

```
clg
load fort.8
rpat=fort(:,3);
rdb=fort(:,9);
plot(rpat,rdb)
```

ZEZ.M MATLAB Source Code

```
clg
load fort.8
rpat=fort(:,3);
rdb=fort(:,12);
plot(rpat,rdb)
```

THETHE.M MATLAB Source Code

```
clg
load fort.8
index1=size(fort);
thepat=fort(:,2);
thetadb=fort(:,9);
plot(thepat,thetadb,'b')
xlabel('Elevation angle in degrees')
```

```
ylabel('E theta in dB')
```

THEPHI.M MATLAB Source Code

```
clg
load fort.8
index1=size(fort);
thepat=fort(:,2);
thetadb=fort(:,12);
plot(thepat,thetadb)
```

PHIPHI.M MATLAB Source Code

```
clg
load fort.8
index1=size(fort);
thepat=fort(:,3);
thetadb=fort(:,12);
plot(thepat,thetadb)
```

PHITHE.M MATLAB Source Code

```
clg
load fort.8
index1=size(fort);
thepat=fort(:,3);
thetadb=fort(:,9);
plot(thepat,thetadb)
```

EDIN.M MATLAB Source Code

```
!emacs inbsc.dat
```

PR.M MATLAB Source Code

```
print
```

PREV.M MATLAB Source Code

```
header='Previewer';
labels=str2mat(...
   'Preview Object ',....
   'Change Units');
callbacks=str2mat('bsv1','scale2');
choices('Preview',header,labels,callbacks);
```

PTRACE.M MATLAB Source Code

```
!lpr trace.dat
```

RADPAT1.M MATLAB Source Code

```
header='NEAR FIELD SPHERICAL COORD.';
labels=str2mat(...
   'THETA VS. ETHETA in dB',...
   'THETA VS. EPHI in dB',...
   'PHI VS. ETHETA in dB',...
   'PHI VS. EPHI in dB');
callbacks=str2mat('thethe','thephi','phithe','phiphi');
choices('PLOTS2',header,labels,callbacks);
```

RADPAT2.M MATLAB Source Code

```
header='NEAR FIELD RECTANGUALR COORD.';
labels=str2mat(...
   ' VARY X',...
   ' VARY Y',...
   ' VARY Z');
callbacks=str2mat('varx','vary','varz');
choices('PLOTS3',header,labels,callbacks);
```

RADPAT3.M MATLAB Source Code

```
header='FAR FIELD PATTERNS';
labels=str2mat(...
    'THETA VS. ETHETA in dB',...
    'THETA VS. EPHI in dB',...
    'PHI VS. ETHETA in dB',...
    'PHI VS. EPHI in dB');
callbacks=str2mat('thethef','thephif','phithef','phiphif');
choices('PLOTS3',header,labels,callbacks);
```

RADPLOT.M MATLAB Source Code

```
header='PLOTTING BSC RADIATION PATTERN DATA';
labels=str2mat(...
    'PLOT NEAR FIELD SPHERICAL COORD.',...
    'PLOT NEAR FIELD RECTANGULAR COORD.',...
    'PLOT FAR FIELD');
callbacks=str2mat('radpat1','radpat2','radpat3');
choices('PLOTS',header,labels,callbacks);
```

RAYTRAC.M MATLAB Source Code

```
header='BSV RAY TRACING MENU';
labels=str2mat(...
    'RAY TRACE WITH COLOR CODED MECHANISMS',...
    'RAY TRACE WITH COLOR CODED MAGNITUDES',...
    'RAY TRACE SELECTED MECHANISMS        ',...
    'RAY TRACE SELECTED MAGNITUDES        ',...
    'ORIGINAL VIEW');
callbacks=str2mat('bsv1b','bsv1c','selray1','selray2','view(3)');
choices('RAYTRACER',header,labels,callbacks);
```

SELRAY1.M MATLAB Source Code

```
header='Selected Mechanisms 1';
labels=str2mat(...
   'Choose Direct Rays             ',...
   'Choose reflected rays          ',...
   'Choose singly diffracted rays  ',...
   'Choose higher order rays       ',...
   'CHANGE AXIS');
callbacks=str2mat('sel1=1;bsv1d','sel1=2;bsv1d',...
       'sel1=3;bsv1d','sel1=4;bsv1d','v4');
choices('SELECTOR1',header,labels,callbacks);
```

SELRAY2.M MATLAB Source Code

```
header='Selected Mechanisms 2';
labels=str2mat(...
   'Choose Direct Rays             ',...
   'Choose reflected rays          ',...
   'Choose singly diffracted rays  ',...
   'Choose higher order rays       ',...
   'CHANGE AXIS');
callbacks=str2mat('sel2=1;bsv1e','sel2=2;bsv1e',...
       'sel2=3;bsv1e','sel2=4;bsv1e','v4');
choices('SELECTOR2',header,labels,callbacks);
```

THRDE.M MATLAB Source Code

```
header='3-D Viewer Menu';
labels=str2mat(...
   'VIEW in 3-D',...
   'PRINT 3-D VIEW');
callbacks=str2mat('v3','pv3');
choices('THREEDEE',header,labels,callbacks);
```

VTRACE.M MATLAB Source Code

```
!emacs trace.dat
```

PV3.M MATLAB Source Code

```
view(90,0)
```

```
title(' front view ')
print
view(0,0)
title(' side  view ')
print
view(0,90)
title(' overhead view')
print
```

V2.M MATLAB Source Code

```
for i=1:41
ang1(i)=0;
ang2(i)=9*(i-1);
end
M=moviein(41);
for j=1:41
view(ang2(j),ang1(j));
M(:,j)=getframe;
end
movie(M)
clear
```

V2A.M MATLAB Source Code

```
for i=1:41
ang1(i)=0;
ang2(i)=9*(i-1);
end
M=moviein(41);
for j=1:41
view(ang2(j),ang1(j));
M(:,j)=getframe;
end
movie(M)
clear
```

V2B.M MATLAB Source Code

```
for i=1:41
ang1(i)=30;
```

```
ang2(i)=9*(i-1);
end
M=moviein(41);
for j=1:41
view(ang2(j),ang1(j));
M(:,j)=getframe;
end
movie(M)
clear
```

V2C.M MATLAB Source Code

```
for i=1:41
ang1(i)=60;
ang2(i)=9*(i-1);
end
M=moviein(41);
for j=1:41
view(ang2(j),ang1(j));
M(:,j)=getframe;
end
movie(M)
clear
```

V2D.M MATLAB Source Code

```
for i=1:41
ang1(i)=90;
ang2(i)=9*(i-1);
end
M=moviein(41);
for j=1:41
view(ang2(j),ang1(j));
M(:,j)=getframe;
end
movie(M)
clear
```

V2E.M MATLAB Source Code

```
ang=input('please type in the elevation angle in degrees: ')
for i=1:41
```

```
angl(i)=ang;
ang2(i)=9*(i-1);
end
M=moviein(41);
for j=1:41
view(ang2(j),angl(j));
M(:,j)=getframe;
end
movie(M)
clear
```

MOV.M MATLAB Source Code

```
header='MOVIE MENU';
labels=str2mat(...
   'ELEVATION =0 degrees',...
   'ELEVATION =30 degrees',...
   'ELEVATION =60 degrees',...
   'ELEVATION =90 degrees',...
   'USERS CHOICE OF ELEVATION ANGLE');
callbacks=str2mat('v2a','v2b','v2c','v2d','v2e');
choices('MOVIEMEN',header,labels,callbacks);
```

V4.M MATLAB Source Code

```
fig=gcf
%clf
sli_azm=uicontrol(fig,...
   'Style','slider',...
   'Position',[50 50 120 20],...
   'Min',-90,'Max',90,'Value',30,...
   'CallBack',[...
   'set(azm_cur,"String",',...
   'num2str(get(sli_azm,"Val"))),',...
   'set(gca,"View",',...
   '[get(sli_azm,"Val"),get(sli_elv,"Val")])']);

sli_elv=uicontrol(fig,...
   'Style','slider',...
   'Position',[240 50 120 20],...
   'Min',-90,'Max',90,'Value',30,...
   'CallBack',[...
```

```
    'set(elv_cur,"String",',...
    'num2str(get(sli_elv,"Val"))),',...
    'set(gca,"View",',...
    '[get(sli_azm,"Val"),get(sli_elv,"Val")])']);

 azm_min=uicontrol(fig,...
    'Style','text',...
    'Pos',[20 50 30 20],...
    'String',num2str(get(sli_azm,'Min')));

 elv_min=uicontrol(fig,...
    'Style','text',...
    'Pos',[210 50 30 20],...
    'String',num2str(get(sli_elv,'Min')));

 azm_max=uicontrol(fig,...
    'Style','text',...
    'Pos',[170 50 30 20],...
    'String',num2str(get(sli_azm,'Max')),...
    'Horiz','right');

 elv_max=uicontrol(fig,...
    'Style','text',...
    'Pos',[360 50 30 20],...
    'String',num2str(get(sli_elv,'Max')),...
    'Horiz','right');

 azm_cur=uicontrol(fig,...
    'Style','text',...
    'Pos',[120 80 50 20],...
    'String',num2str(get(sli_azm,'Value')));

 elv_cur=uicontrol(fig,...
    'Style','text',...
    'Pos',[310 80 50 20],...
    'String',num2str(get(sli_elv,'Value')));

% now trying the zooming of axis
n=20*coeff;
sli_zom=uicontrol('style','slider','pos',[430 50 120 20],...
      'min',.1*coeff,'max',100*coeff,'val',n,...
      'call','n=get(sli_zom,"value");axis(scale1*[-n n -n n -n n])');

txt_zom=uicontrol('style','text','pos',[430 80 65 20],...
        'string','zoom');

txt_elv=uicontrol('style','text','pos',[240 80 65 20],...
        'string','elv.');

txt_azm=uicontrol('style','text','pos',[50 80 65 20],...
        'string','azm.');
```

Appendix B: MATLAB Information

The MATLAB program is a matrix oriented programming environment. The input data is in either matrix or vector form. There is a limited capacity to handle strings and it is most often useful to handle strings with a program written in a different language such as C or FORTRAN or Pascal. The MATLAB environment allows the code user to shell out into the operating system. Therefore the use of other programs is not difficult. The files input to MATLAB should be in matrix or vector form.

The graphics library of MATLAB can handle hidden line removal as well as three dimensional plotting. The graphics library can be set up to be controlled by a GUI. The GUI interface is similar to X-Window controls. The user must specify the control position, type, and the variables affected by the GUI.

The observation point for the three dimensional view is controlled by the elevation and azimuth angles. The definition of these angles is particular to MATALB and are shown in Figure B.1. Since these angles are particular to MATLAB, care should be exercised in interfacing the graphics library to other codes with different coordinate systems.

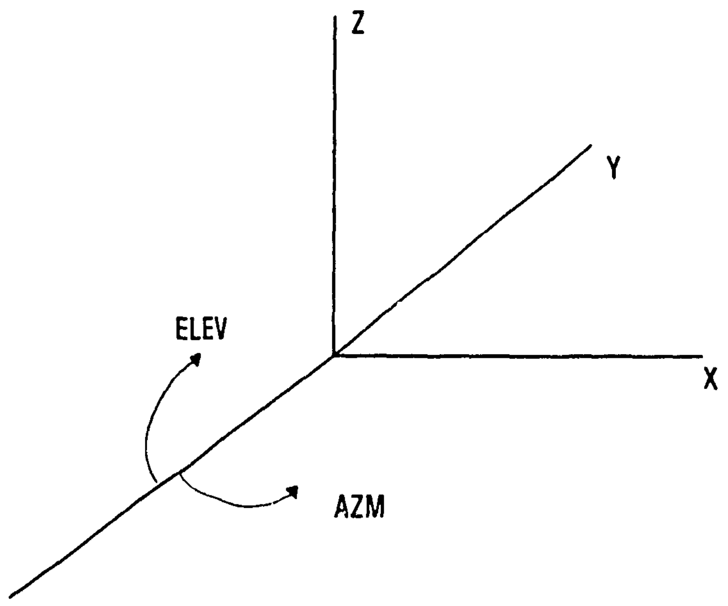


Figure B.1 Definition of azmiuth and elevation angles

The placement of data onto the plots as text should be done by first using the str2mat command. Then the code user may place the text as desired. Remember, that since this is a matrix it may have to be inverted to have it print properly.