NASA-CR-199212

THE OHIO STATE UNIVERSITY

X-Antenna — A Graphical Interface for Antenna Analysis Codes

B.L. Goldstein, E.H. Newman and H.T. Shamansky

The Ohio State University

ElectroScience Laboratory

Department of Electrical Engineering
Columbus, Ohio 43212

Technical Report 722792-8
Grant No. NAG-1-1058
January 1995

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665-5225

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO.	2.	3. Recipient's Accession No.

4. Title and Subtitle

X-Antenna — A Graphical Interface for Antenna Analysis Codes

5. Report Date
January 1995

6.

7. Author(s)
B.L. Goldstein, E.H. Newman and H.T. Shamansky

8. Performing Org. Rept. No.
722792-8

9. Performing Organization Name and Address
The Ohio State University
ElectroScience Laboratory
1320 Kinnear Road
Columbus, OH 43212

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.
(C)
(G) NAG-1-1058

12. Sponsoring Organization Name and Address
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665-5225

13. Report Type/Period Covered
Technical Report

14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

This report will serve as the user's manual for the **X-Antenna** code. **X-Antenna** is intended to simplify the analysis of antennas by giving the user graphical interfaces in which to enter all relevant antenna and analysis code data. Essentially, **X-Antenna** *creates* a Motif interface to the user's antenna analysis codes. A command-file allows new antennas and codes to be added to the application. The menu system and graphical interface screens are created dynamically to conform to the data in the command-file. Antenna data can be saved and retrieved from disk. **X-Antenna** checks all antenna and code values to ensure they are of the correct type, writes an output file, and runs the appropriate antenna analysis code. Volumetric pattern data may be viewed in 3-D space with an external viewer run directly from the application. Currently, **X-Antenna** includes analysis codes for thin wire antennas (dipoles, loops and helices), rectangular microstrip antennas and thin slot antennas.

17. Document Analysis a. Descriptors

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
A. Approved for public release; Distribution is unlimited.	Unclassified	105
	20. Security Class (This Page) Unclassified	22. Price

(See ANSI-Z39.18) *See Instructions on Reverse* OPTIONAL FORM 272 (4-77)
Department of Commerce

Contents

List of Figures

Chapter 1

Introduction

This report will serve as the users manual for the **X-Antenna** code. **X-Antenna** is intended to simplify the analysis of antennas by their respective analysis codes. Typically, when using a text-based analysis program, the user must create and modify raw data files which an analysis code reads in. **X-Antenna** provides a graphical interface for entering this data and allows the user to easily modify data values and run the analysis codes directly from the graphical interface. **X-Antenna** does this by letting the user define antennas and codes through a command-file which is read in when **X-Antenna** is executed. Once antennas and codes are defined, a menu system is created dynamically for the set of antennas and codes. The user may then select an antenna from the menu and enter the parameters which are used to define it. Parameters specific to a particular analysis code may also be selected through the menu in a similar manner. When the user selects either a pattern or frequency sweep run, the code and antenna data-values are checked to see that they conform to the data-type specified in the command-file, a data-file is written with the appropriate antenna, pattern, and code information, and the analysis code for the given antenna is run. Volumetric patterns may be viewed and manipulated in 3-D space using an external volumetric pattern viewer which may also be run from the **X-Antenna** interface. Currently, **X-Antenna** supports three antenna analysis codes. The **MATWRS** [1] code for thin wire antennas, the **Microstrip** code for rectangular microstrip antennas and the **Slot** code for thin slot antennas.

Chapter 2 provides a general overview of the **X-Antenna** application and its operation. The user unfamiliar with this code should read this chapter first. Chapter 3 outlines the operation of all the menu functions. This includes the file menu for saving and loading antenna data, the antenna menu for selecting antenna types for analysis, the code menu for selecting code values, and the view menu for viewing volumetric patterns and text files. This chapter also covers volumetric pattern and frequency sweep runs, the error checking performed when a run is selected, and provides an example pattern run. Chapter 4 details the use of the command-file and the format of the command directives used to define antennas and codes. Chapter 5 describes the format of the output file and how to use conversion routines if necessary to convert the **X-Antenna** file to a different format. Chapter 6 covers the operation of the volumetric pattern viewer from both the command line and from within the **X-Antenna** application. The appendices provide listings of a sample command file, conversion routines used by existing antenna analysis codes, and a description of the use of analysis codes for thin slot and rectangular microstrip antennas.

This application is written entirely in the C language. The graphical interfaces were produced with the Motif toolkit, version 1.0 [2, 3] running under the X Windows System, version 11R5 [4]. This code should be compiled with versions of Motif and X-Windows of these versions or higher.

Chapter 2

Application Overview

This chapter will provide the user with a general overview of the **X-Antenna** application and its operation. Detailed descriptions of each menu function and the specifics of operation will be given in Chapter 3. Throughout this manual a typeface convention will be adhered to for the various categories of text. This convention is listed in Table 2.1.

2.1 Overview

The purpose of **X-Antenna** is to provide the user with graphical interfaces for the entry of all data relevant to the analysis of an antenna by an associated computer code. Specifically, **X-Antenna** provides the user with graphical interfaces for defining:

- the antenna geometry and other antenna data.
- volumetric pattern angles or frequency sweep values.
- other data which an analysis code might need.

X-Antenna is actually a collection of the following computer codes:

- the graphical interface code which allows the user to enter all necessary data.
- several antenna analysis codes, i.e., one for each class of antenna supported by **X-Antenna**. Currently, the supported analysis codes are the **MATWRS** [1] code for wire antennas, the **Slot** code for thin slots (see Appendix C), and the **Microstrip** code for rectangular microstrip antennas (see Appendix D).

Table 2.1: Typeface convention for this manual.

Text	Format
Menu and Sub-Menu Names	**Bold**
Push Button Labels	**Bold**
Parameter Data-Field Labels	*Italics*
Indicator Selection Labels	*Italics*
Executable Program Names	*Italics*
Source Code File names	*Italics*
Typed Text (on the command-line)	`Typewriter`
Computer Code Listings	`Typewriter`
Subroutine Names	`Typewriter`
Command-File Directives	CAPITAL

- auxiliary codes which perform functionally separate tasks such as graphical displaying of volumetric pattern data.

X-Antenna is intended to be generic so that any (reasonably simple) antenna may be analyzed with its associated analysis code. To accomplish this, the parameters necessary to describe an antenna and the data which its associated code might require are defined in a *command-file* which is read in when **X-Antenna** is executed. These parameters are then incorporated into the graphical interfaces for the respective antennas and codes. In this way, **X-Antenna** can be viewed as a code which *creates* a Motif interface to several antenna analysis codes defined via the command file. The command-file is discussed in Section 2.3 and covered in more detail in Chapter 4.

Typically, the following is done (in any order) prior to performing a pattern or frequency sweep run:

- Create a new antenna by selecting one from the **Antenna** menu, or load a previously defined antenna from disk using the **Load** option in the **File** menu.
- Enter new antenna parameters, or change existing parameters if necessary.

- Enter the appropriate pattern angle or frequency sweep values.
- Change the code parameters if necessary. (if any are defined for the analysis code used for the particular antenna)

Once this information is entered and the user selects either a pattern or frequency sweep run, **X-Antenna** performs these tasks:

- Check the antenna parameters to ensure they are of the correct data type (as defined in the command-file).
- Check the frequency sweep or pattern angle sweep values to ensure they are consistent.
- Write an output file containing the necessary antenna data, pattern or frequency sweep data, and code data. This file is then used as the input file for the antenna analysis code.
- Execute the antenna analysis code specified for that antenna after allowing the user to add command line options.

Ultimately, **X-Antenna** simply organizes all the necessary data and writes an output file to be used as the input file for the antenna analysis code. The advantage to using **X-Antenna** is that the compilation of the antenna data is done through graphical interfaces. This allows the user to easily change the geometry of an antenna, the pattern angle or frequency sweep values, and any code information without having to modify a raw data file.

2.2 The X-Antenna output file

The antenna analysis codes must be capable of using the output file from **X-Antenna**. If a new code is being written, it can be designed to conform to the **X-Antenna** output file format. (This format is outlined in Chapter 5). It is unlikely, however, that any existing program will be able to directly read in this output file. The user must then either convert the **X-Antenna** output file to a format that can be

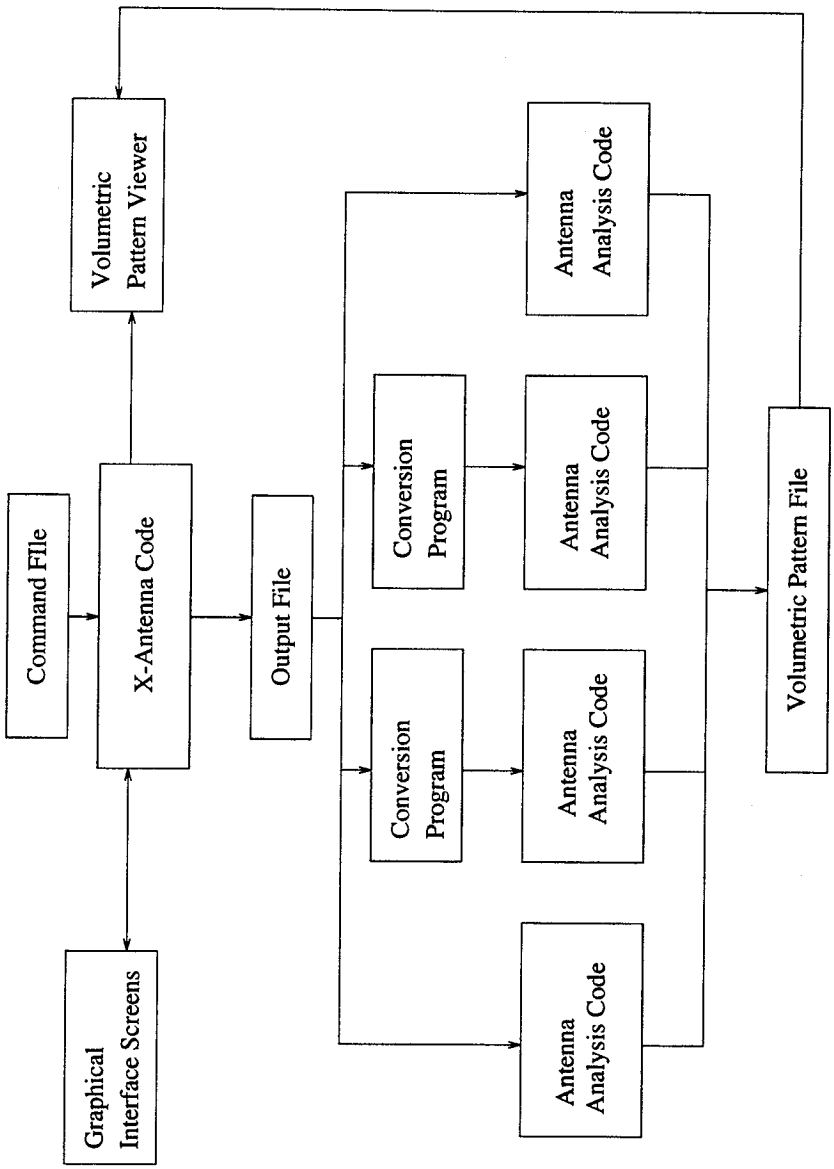


Figure 2.1: Block diagram of the **X-Antenna** application.

interpreted, or alter their antenna analysis program to read in the file directly. In most cases, it is easier to convert the **X-Antenna** output file to conform to the format of the existing antenna analysis code. This was done in the case of the **MATWRS** [1] code for the analysis of thin wire antennas. To aid the user in converting the output data, two generic routines have been provided, one in FORTRAN and one in C (*convert.f* and *convert.c*), which will read in any **X-Antenna** output file. These routines can be modified to produce the input file for an existing antenna analysis code and then execute that code, or to call another subroutine directly. Additionally, the routines used for the **MATWRS**, **Slot**, and **Microstrip** codes are provided as specific examples of data-conversion routines. A description of these routines is provided in Chapter B.

2.3 The Command File

The user specifies the parameters needed to define each antenna in a *command-file* which is read in when **X-Antenna** is executed. The user defines the data-type of each parameter and an optional default value. Each parameter for a given antenna is then represented by a data-field in the graphical interface screen brought up when that antenna is selected. Additionally, the name specified for each parameter is used to label the respective data-fields in the antenna screen. The command-file format is detailed in Chapter 4. To allow flexibility with analysis codes, the user may also specify parameters to be used by each analysis code in the command-file. These are referred to as *code-options* and are also covered in Chapter 4.

2.4 Graphical Display for Antennas

X-Antenna allows a bitmap to be defined for each antenna. This bitmap is made visible in a separate window when the user selects the **Geometry** button available on each antenna screen. The bitmap is contained in a file which conforms to the X-bitmap (xbm) format. The name of this bitmap file is specified in the command-file for each antenna. If a bitmap file is not specified for a particular antenna, the

Geometry button will be grayed-out in the corresponding antenna screen. This feature may be used to provide a graphical description of the antenna geometry. The creation and use of bitmap files is covered in more detail in Chapter 3.

2.5 Dynamic Menu and Screen Creation

The menu system of **X-Antenna** is dynamically created to conform to the antennas and codes defined in the command-file. When an antenna or code is selected from the menu, a screen containing the antenna or code data is created with the correct data-fields. In this way, there is never a need to recompile **X-Antenna** no matter how many different antennas or analysis codes are used. Also, several command files may be created, each containing different sets of antenna and code information since the command-file is specified on the command-line when **X-Antenna** is executed.

2.6 Viewing Volumetric Data

Volumetric pattern data from the antenna analysis codes can be viewed using the **Volumetric Pattern Viewer** directly from **X-Antenna**. Operation of this program is discussed in Chapter 6. This program utilizes graphics routines specific to Silicon Graphics machines and will therefore only operate on a Silicon Graphics terminal. Volumetric patterns may be viewed in one of two ways. The first is as a sphere with the magnitude of the gain represented by color-shading. The second is similar, except that the radius of the pattern also varies with the magnitude of the gain. This shows the lobes of the pattern in 3-D space. Both representations may be viewed with either color or gray-scale indexing. Both θ and ϕ-polarized fields can be viewed, and the pattern can be translated, scaled, and rotated using the mouse as a "virtual trackball" directly on the screen. Additionally, an RGB-format screen-dump of the image can be obtained.

2.7 The Application Resource File

This application runs under the X Windows System [4]. Within this system, resources which control the appearance and behavior of applications may be specified through a separate file known as an *application-defaults file* [2, 3]. It is suggested that a good understanding of resource specifications be obtained prior to altering this file.

2.8 Reporting Bugs and Asking Questions

If any bugs are found with **X-Antenna** or if you have any question about its operation or would like to comment on how it may be improved, contact:

Ed Newman
The Ohio State University
ElectroScience Lab
(614) 292-4999
ehn@lenz.eng.ohio-state.edu

Chapter 3

Using X-Antenna

This chapter will outline the content and use of the different graphical interface screens in **X-Antenna**. These include:

- the main screen, containing volumetric pattern angle and frequency sweep information
- the antenna screen, containing the antenna data
- the code-options screen, containing optional code parameters.

It is assumed at this time that there are antenna(s) and code(s) defined within the application. Details of how to specify antennas and codes through a command-file will be given in Chapter 4. For the beginning user, it is recommended that the current chapter be read first in order to understand how these parameters are actually realized in the application.

3.1 Running X-Antenna from the Command Line

To execute **X-Antenna** change to the directory where the **X-Antenna** code is located (typically called Xant) and at the command prompt enter:

```
xant commandfile
```

xant = name of the **X-Antenna** executable code.

commandfile = name of the command-file to use.

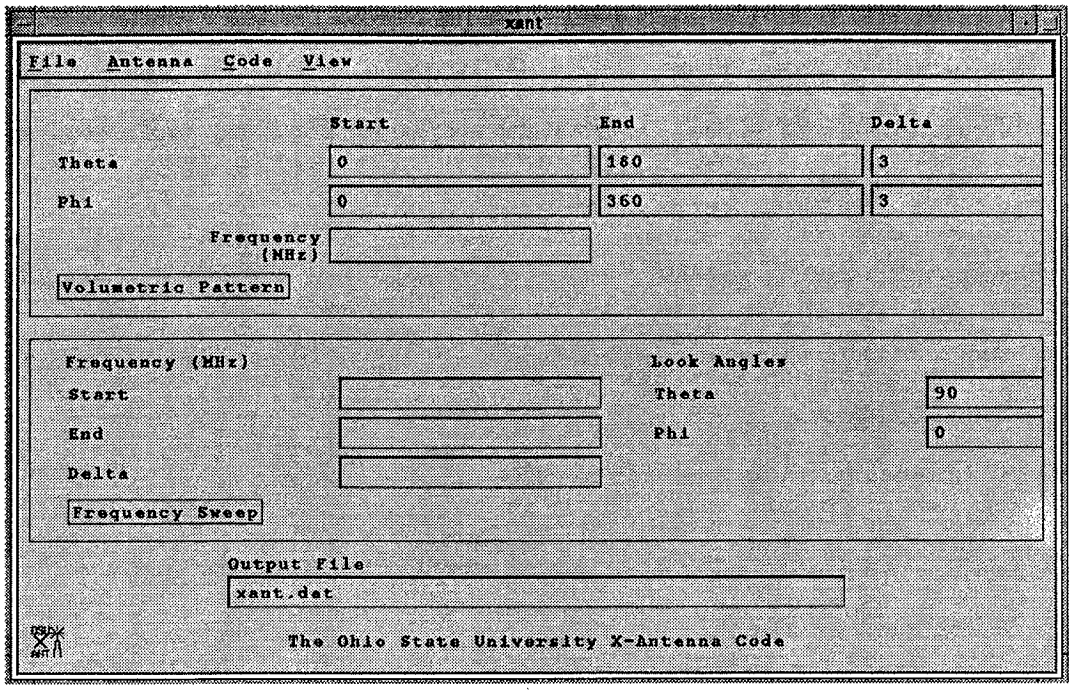


Figure 3.1: The **X-Antenna** application main screen.

X-Antenna will then read in the command-file and display the main screen shown in Figure 3.1.

3.2 The Main Screen

The main screen, shown in Figure 3.1, houses a menu bar at the top containing pull-down menus for file operations, antenna selection, code selection and viewing data. In the main area are data-fields for the volumetric pattern and frequency sweep values, push-buttons used to initiate pattern and frequency sweep runs, and the name of the output to be written when a run is selected.

The data-fields for a volumetric pattern are:

Start Theta, Stop Theta = Initial and final angles, respectively, (in degrees) for the pattern in the θ-direction. Default: (Start Theta, Stop Theta) = (0,180).

Start Phi, Stop Phi = Initial and final angles, respectively, (in degrees) for the pattern in the ϕ-direction. Default: (Start Phi, Stop Phi) = (0,360).

Delta Theta, Delta Phi = Increment angle (in degrees) in the θ and ϕ-directions, respectively. Default: (Delta Theta, Delta Phi) = (3,3). Caution: the present volumetric pattern viewer can produce erratic results if the angle increments exceed 4 degrees.

Frequency = Frequency (in MHz) at which the pattern is performed.

The default values for the pattern defines a full volumetric pattern with an increment angle of 3 degrees.

The data-fields for a frequency sweep are:

Start Freq, Stop Freq = Initial and final frequencies, respectively, (in MHz) for the frequency sweep.

Delta Freq = Increment frequency (in MHz) for a frequency sweep.

Theta, Phi = Look angles (in degrees) in the θ and ϕ-directions, respectively.

Data-fields common to both pattern and frequency-sweeps:

Output File = Name of the **X-Antenna** output file to be used as the input file for the antenna analysis code. Default: `xant.dat`.

All pattern and frequency sweep computations are far-field. Figure 3.2 shows the spherical (r, θ, ϕ) coordinates used for all volumetric pattern cuts and frequency sweep look angles.

3.3 The Antenna Menu

Unless a previously defined antenna is loaded from disk, one must be created before a run may be performed. This is done by selecting an antenna from the **Antenna** menu. As with most menu-bars, a selection is made by holding the left mouse-button down over the intended menu-bar item and dragging the mouse down over the intended selection. Alternately, if the button is clicked once over the menu-bar, the pull-down menu will stay visible and the selection may then be made. Figure 3.3 shows the

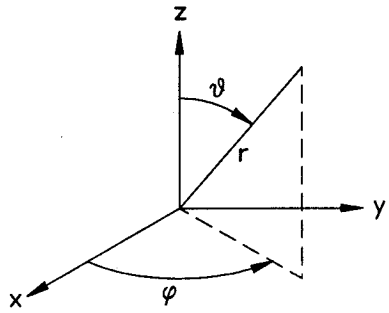


Figure 3.2: The (x, y, z) rectangular and (r, θ, ϕ) spherical coordinate systems used for all pattern angles and frequency sweep look angles.

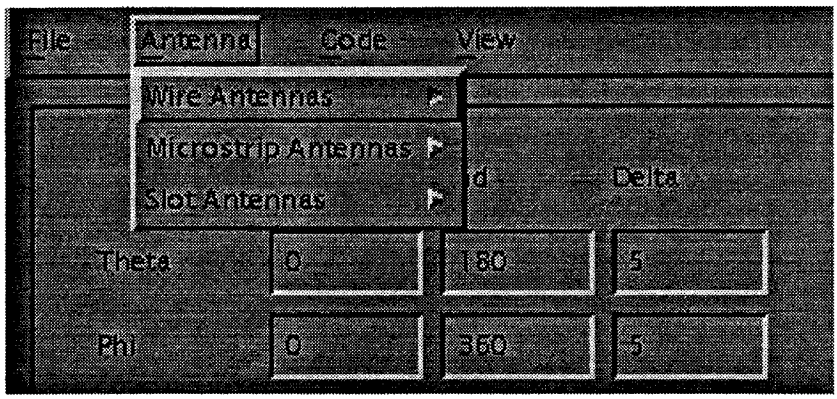


Figure 3.3: Antenna pull-down menu showing the antenna category sub-menus.

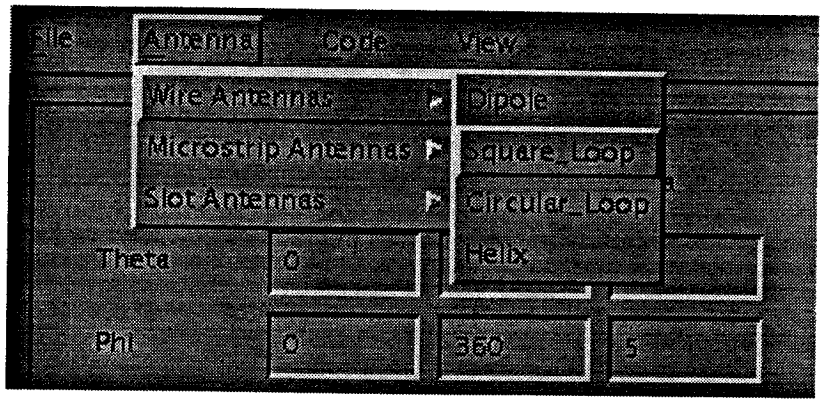


Figure 3.4: Wire Antennas sub-menu showing available wire antenna types.

categories of antennas available under the **Antenna** menu. Each of these categories in turn is a sub-menu, each containing one or more antennas which may be selected. Figure 3.4 shows the antennas available under the **Wire Antennas** category. In order to organize the menu system, antennas always belong to a specific category. There is nothing inappropriate, however, with having only one category if that is the logical arrangement. The actual organization of antennas into categories is done through the command-file which is covered in Chapter 4.

3.3.1 The Antenna Screen

Once an antenna is selected, a graphical screen will appear that contains the data-fields for defining the antenna parameters. An example of this is the dipole antenna screen shown in Figure 3.5. The top bar of the window contains the name of the antenna (the same name as in the **Antenna** menu). The first two rows of the screen list the current filename of the antenna for saving it to disk and the analysis code used for the antenna. When a new antenna is defined, the antenna file name is constructed from the name of the antenna with the extension, **antd** (**antenna data**), appended after a period. Figure 3.5 shows the default filename of the dipole antenna to be `Dipole.antd`.

Until now, the term "parameter" has been used to refer to the data which defines the antenna. There are actually two distinct data forms which may be utilized when describing an antenna:

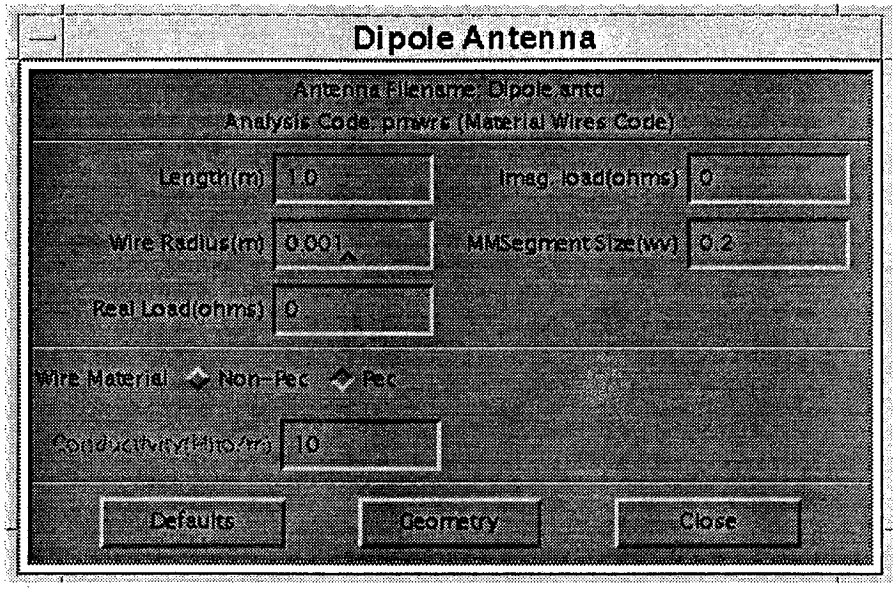


Figure 3.5: The dipole antenna screen showing two data-lists, one with parameters only, the other with a parameter and an indicator. Note that the label, *Conductivity(Mho/m)*, is grayed-out because the indicator in the same data-list is set to its **FALSE** value (see text).

- Parameters, which may be integers, real numbers or character strings.
- Indicators, which represent logical variables. i.e., true or false.

As shown in Figure 3.5, this dipole antenna is described by six parameters, *Length, Wire Radius, Real Load, Imag. Load, MM Segment Size* and *Conductivity*, and one indicator, *Wire Material.* It is very important to note the groupings of the parameters and indicators. A horizontal separator distinguishes groups of parameters and indicators which represent related data and are therefore grouped together. Chapter 4 describes in detail how to define these categories in the command-file. These groupings will be referred to as *data-lists* from here on.

3.4 Data-Lists

Data-lists are composed of an arbitrary number of parameters and possibly one (optional) indicator. However, it is only required that at least one parameter or an indicator be present in a data-list. In this way, data-lists may be composed of:

- One or more parameters.

- A single indicator with a true and false value.
- A single indicator with a true and false value plus one or more parameters which are relevant only if the indicator is true (see Section 3.4.1).

A helix antenna containing data-lists configured in these three ways is shown in Figure 3.6.

3.4.1 Indicators with Associated Parameters

If an indicator is defined in the same data-list as one or more parameters, the indicator is intended to reflect the relevance of the parameters associated with it in the same data-list. In Figure 3.5, the indicator for *Wire Material* is associated with the parameter that specifies the wire conductivity. If the left most button, in this case *Non-PEC* is selected, the associated parameter is considered relevant. However, if *PEC* was selected it is not. This arrangement allows the user to define one or more parameters which may be utilized only if a particular condition is specified. In this example, the user may select the wire material to be a PEC (Perfect Electric Conductor), in which case the parameter *Conductivity* need not be defined since a PEC has infinite conductivity. The helix antenna in Figure 3.6 shows two data-lists that contain both parameters and an indicator. In this case the benefit of not having to enter several parameters is more obvious. If the user selects *No* for the *Matching Cup* indicator, the eight parameters defining the matching-cup in the corresponding data-list need not be entered.

The left-hand button of an indicator always represents the condition when the parameter(s) associated with that indicator are relevant. The right-hand button always represents the opposite case. For this reason, the left-hand button is said to represent the **TRUE** value of the indicator, while the right-hand button represents the **FALSE** value. These terms will be used from now on. If the **FALSE** value of an indicator is selected, the labels of any parameters in an associated data-list are grayed-out. The user in this case is also prevented from altering the values of the parameters, or from entering them if they have not already been entered. This

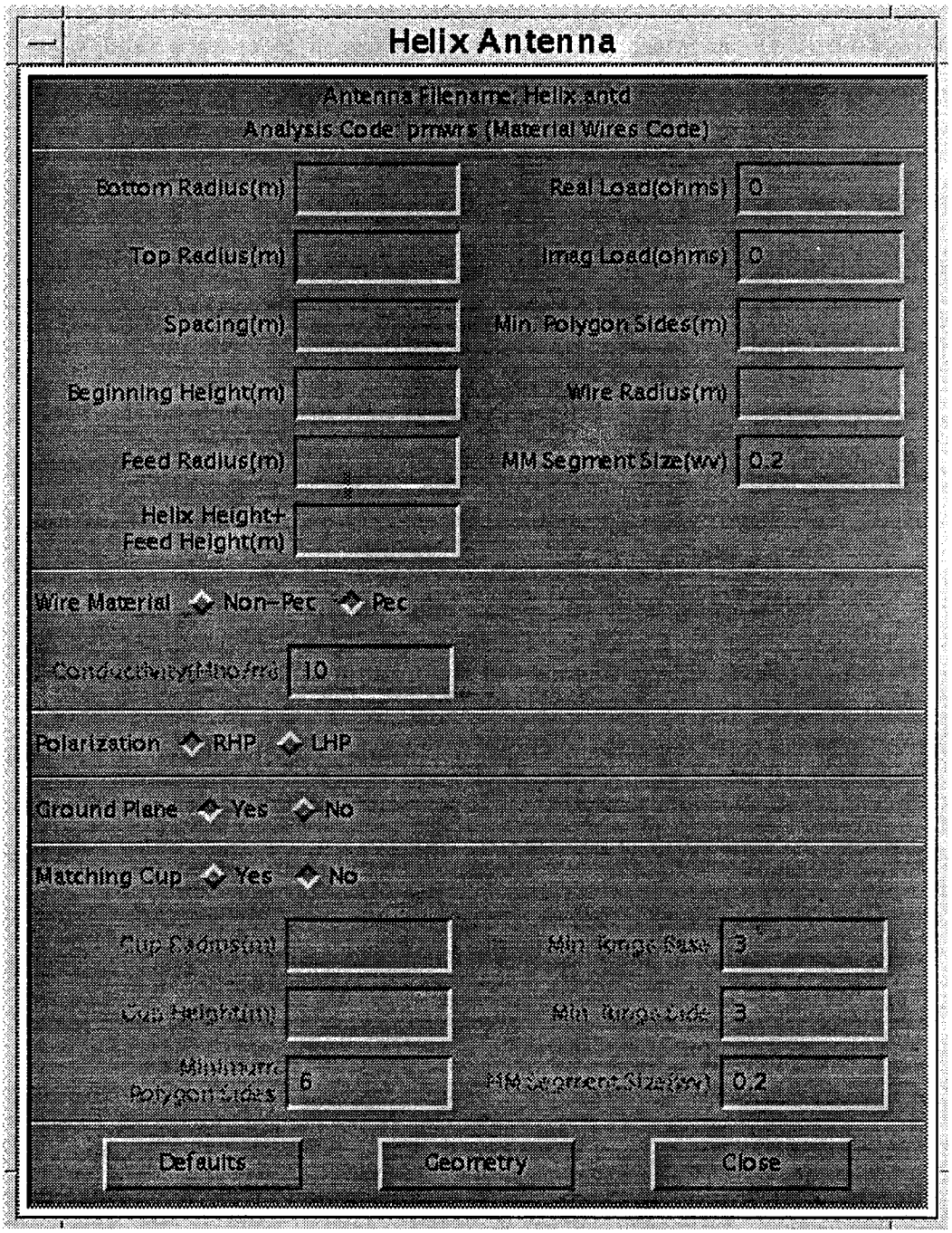


Figure 3.6: Helix antenna screen showing data-lists with only parameters, only an indicator, and both parameters and an indicator.

emphasizes the fact that they are not relevant for a given configuration. It should also be noted that no error-checking will be performed on these parameters when a run is selected. In this way, a value does not even need to be entered if the indicator is set to **FALSE**. Otherwise, the user would be forced to enter a valid value for a parameter which is not even being used for a particular configuration. Error checking and code execution are discussed more fully in Section 3.8. Even if the indicator in a data-list with parameters is set to its **FALSE** value, the parameters will still be written to the output file. If the data-field for a parameter is blank, a dummy value will be written. This is to maintain the constancy of the output file format. The output file is covered in more detail in Chapter 5.

The user specifies the name of the indicator as well as the labels for the **TRUE** and **FALSE** valued buttons in the command-file. It is the users responsibility to select appropriate names to represent the **TRUE** and **FALSE** conditions of the indicator.

3.4.2 Action Buttons

Three buttons are available to the user at the bottom of every antenna screen:

Close - Closes the antenna screen. Once this is done, all information in the antenna screen is lost. It is identical to the **Close** option available in the **File** menu. The user must select **Save** under the **File** menu if it is desired to save the antenna information to disk before it is closed. The antenna screen must remain open while that antenna is being analyzed. Only one antenna screen may be open at a time.

Geometry - Brings up a screen with graphical information (or other information) about the antenna. The contents of the screen are obtained from an X-bitmap file which the user must create. The name of this file (including path) is specified in the command-file. The geometry screen and bitmap files are covered in Section 3.4.3

Defaults - Resets all parameter data-fields and indicators in the antenna screen to the default values specified in the command-file. If no default was specified in

the command-file for a parameter, the corresponding data-field is cleared or left blank. If no default was specified for an indicator, it will be set to its **FALSE** value.

3.4.3 Geometry Screen

When the **Geometry** button is selected from an antenna screen, a separate screen with geometry (or other information) about the antenna is made visible. Subsequent selection of the **Geometry** button toggles the visibility of this geometry screen. The contents of this screen are obtained from a separate file which the user must create. The name of this file is specified in the command-file. The file must be in the X-bitmap (xbm) format.

The X-Bitmap format was chosen because of its simplicity. The user must have a means of creating a X-bitmap file with the geometric (or other) information desired. This is usually done with a drawing application which can output X-bitmap files. The author used the application *xfig* [5] to create the bitmaps for the geometry screens in this manual. This application has the capability to read and write several image formats including xbm. Figures 3.7 and 3.8 show the **Geometry** screens for the dipole and helix antennas, respectively.

If the X-bitmap files cannot be created, or if none are desired, the specification of the bitmap file is left out of the command-file. The **Geometry** button in this case will be grayed-out.

3.5 The Code Menu

Many codes have run control or code-option parameters which control what the code does and how it does it, and are not associated with antenna geometry. Examples are parameters which control the accuracy of numerical integrations, what data is printed to the output file, and even what method of analysis is to be used. Since **X-Antenna** is intended to be used with arbitrary analysis codes, a method of defining these code-option parameters used by a particular analysis code is available. The code-options are specified as part of the command file. When the user executes

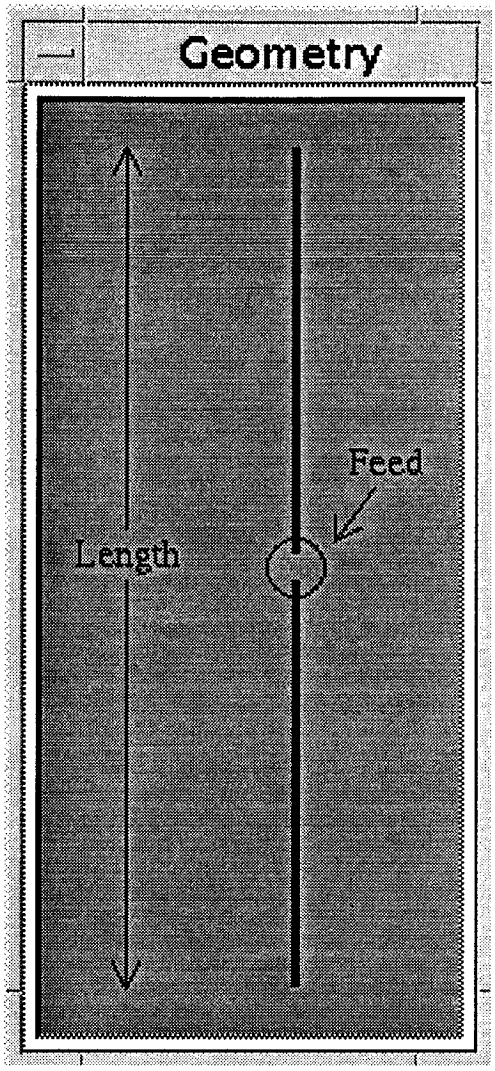


Figure 3.7: Geometry screen for a dipole antenna.

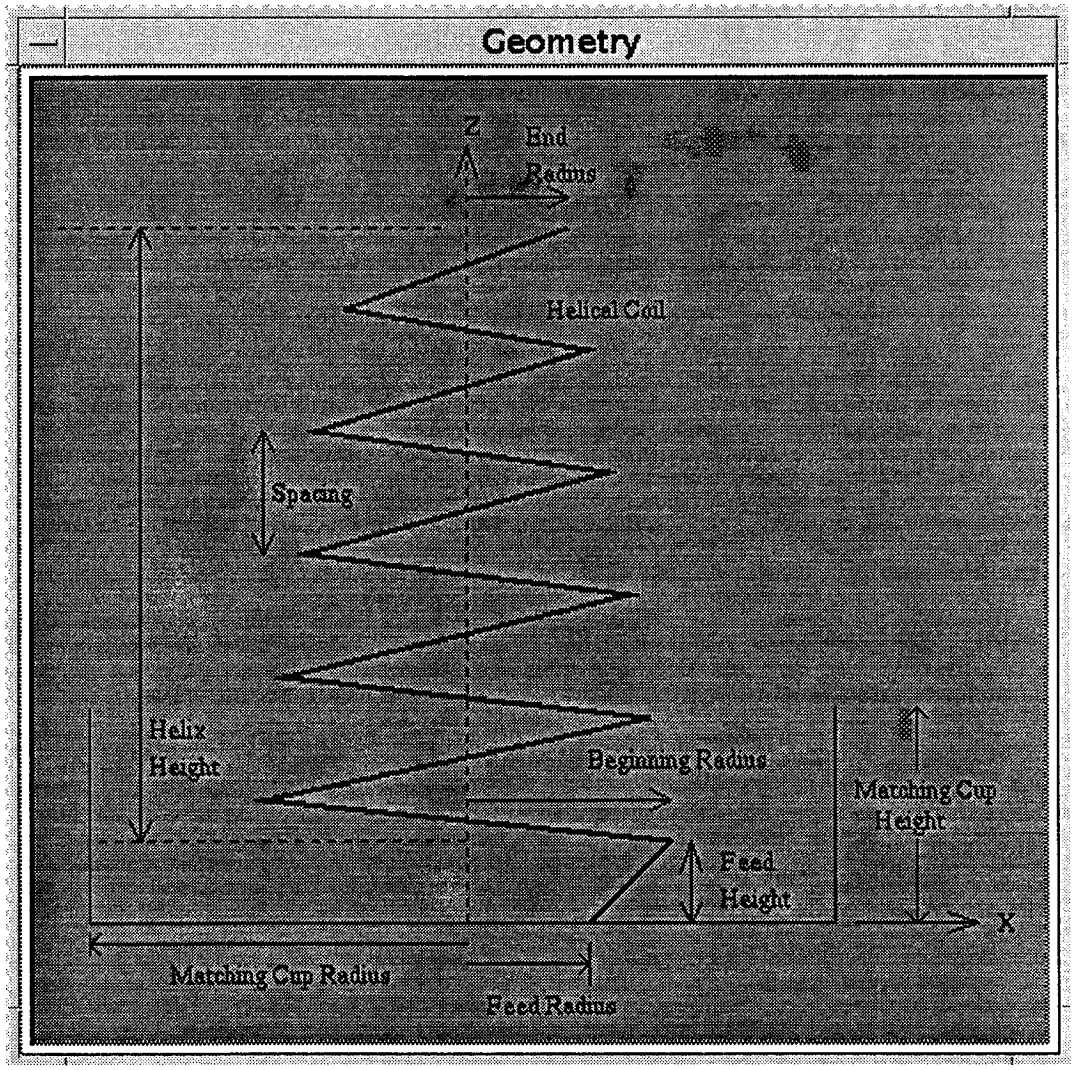


Figure 3.8: Geometry screen for a helix antenna.

a run, any code-options associated with the code specified for the antenna are also written to the output file. In this way, the user may output data associated with code execution along with the actual antenna data. Whatever code reads in the output file from **X-Antenna** will need to interpret this data accordingly. (The format of the **X-Antenna** output file is covered in Chapter 4.)

The **Code** menu contains a selection for each code which has code-options specified. The specification of the code-options is done in the command-file. If no code-options are specified for a given analysis code, there will be no option to select it under the **Code** menu.

3.5.1 The Code Options Screen

When the user selects a particular code under the **Code** menu, a screen is presented, similar the the antenna screen, which contains all the code-option parameters and indicators defined in the command-file for that code. Just as antennas are described by parameters and indicators, so are code-options. There is one important difference though. There is no grouping of indicators and/or parameters into data-lists. That is, an arbitrary number of indicator(s) and/or parameter(s) may be present as long as there is at least one of either. Since there is no possible association between indicators and parameters, there is no greying-out of parameter labels. Additionally, one must have all values specified in the command file so that the user need not even select this menu if the values do not need to be changed. The code-option screen for the **MATWRS** [1] code is shown in Figure 3.9.

3.5.2 Action Buttons

Two buttons are available to the user at the bottom of every code-options screen:

Accept - Accept the current values for the code-options. All parameters are checked to ensure they are of the correct data-type as specified in the command-file. If a parameter is not of the correct type, the data-field is cleared and the user is prompted to enter a new value.

Cancel - Discard any changes made on this screen.

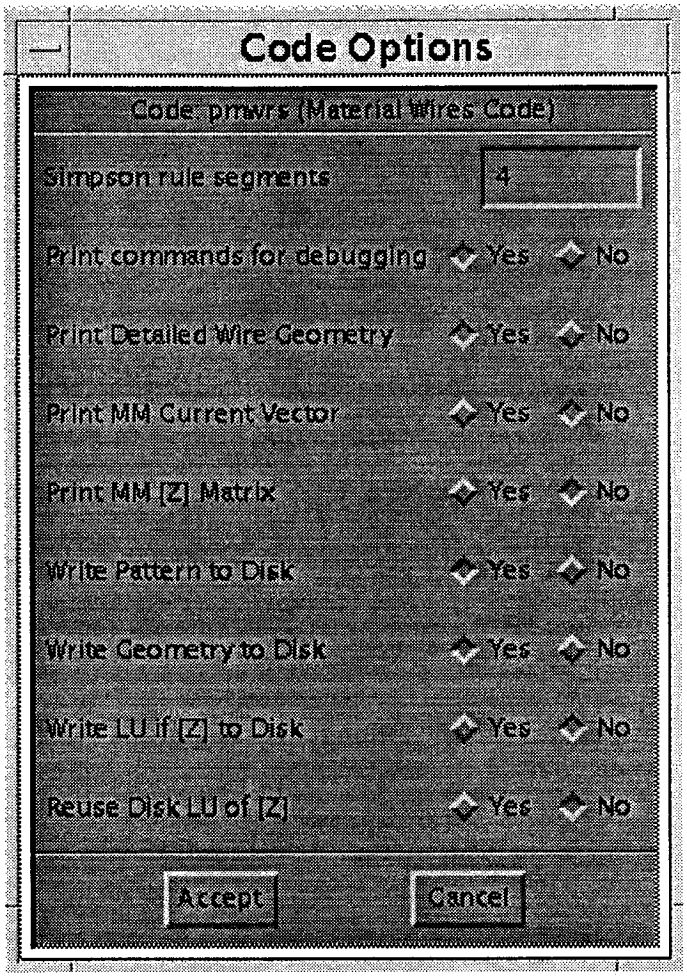


Figure 3.9: Code-Options screen for the **MATWRS** code.

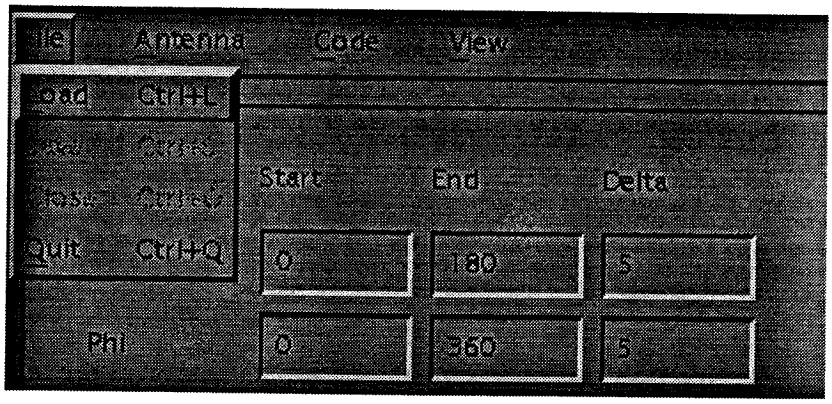


Figure 3.10: File menu showing available options.

3.6 The File Menu

The file menu, shown in Figure 3.10, allows the user to save and load antenna configurations, close an antenna screen, and quit the application. When **Save** is selected, a file-selection screen appears. This screen contains a directory listing of files with the default file-extension (`antd`) used by **X-Antenna**. The default filename given to an antenna is always the antenna name with the default extension appended after a period. This name appears in the *Selection* region of the screen. As an example, Figure 3.11 shows the screen when **Save** is selected for the dipole antenna of Figure 3.5.

The user may of course select a different file name from the default given. If a file already exits, it is overwritten, so be certain not to use the name of an existing file if you do not want it to be overwritten. Selecting **OK** will save the file. Selecting **Filter** will scan the directory using the current filter. Selecting **Cancel** will cancel the save operation.

When **Load** is selected, a similar file selection screen, shown in Figure 3.12, is made available. Since an antenna is being loaded, no default filename is given. However, the file listing will show any files in the current directory with the default extension (`antd`). The user may select one of these files or chose another if it is a valid **X-Antenna** antenna file. An antenna loaded from disk must contain an antenna type currently defined by **X-Antenna**. This may be a problem if there are several command files

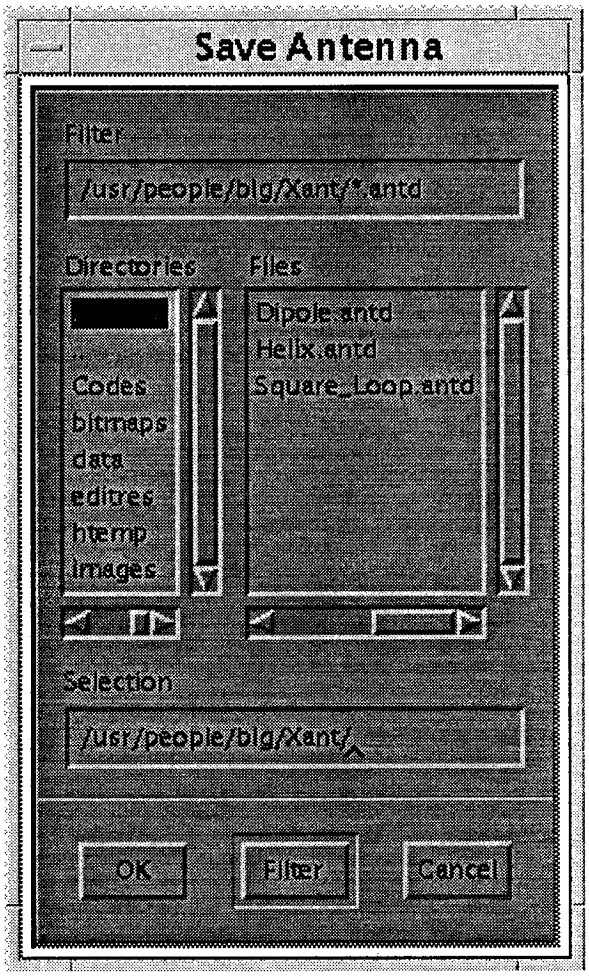


Figure 3.11: File Selection Screen when saving a dipole antenna.

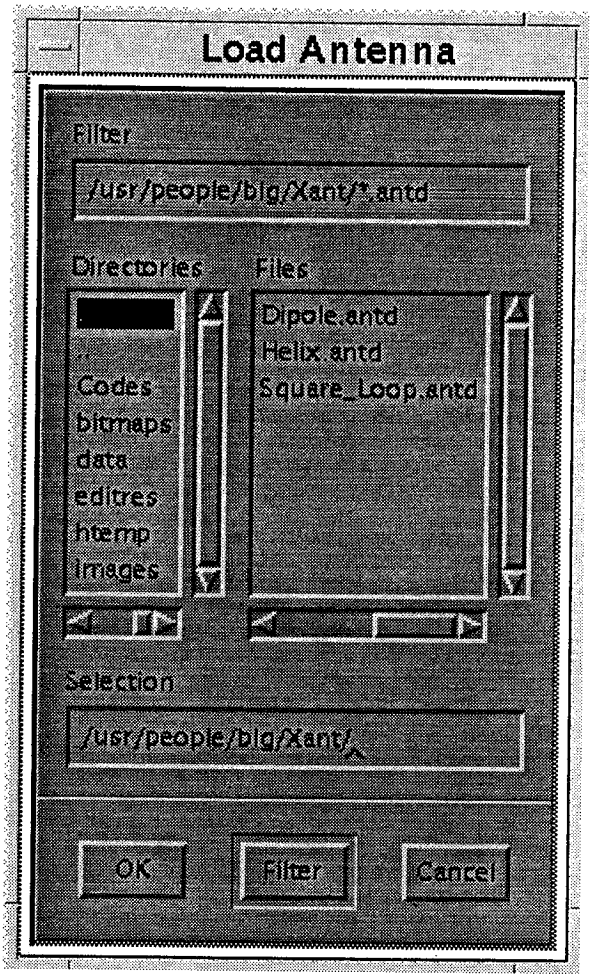


Figure 3.12: File Selection Screen when loading an antenna from disk.

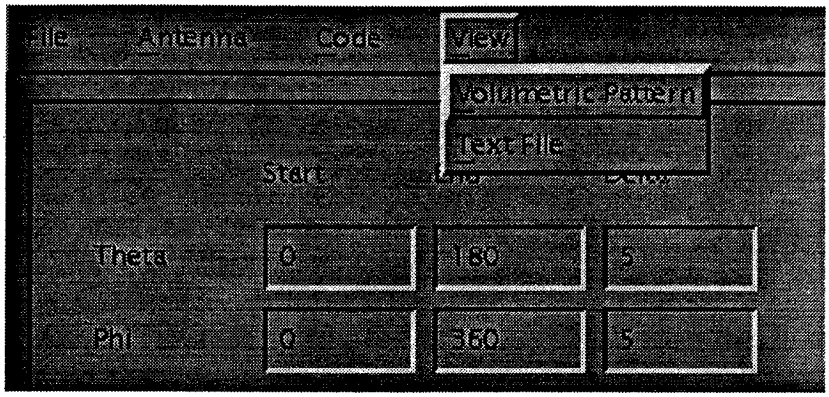


Figure 3.13: View menu showing available options.

containing different sets of antenna definitions. The **OK**, **Filter** and **Cancel** buttons perform the same functions here as when saving.

Selecting **Close** from the **File** menu will close the current antenna. Only one antenna may be defined at a time, so the antenna screen must be closed before another antenna can be selected. The user must select **Save** from the **File** menu to save the antenna information to disk before the antenna screen is closed. This function is duplicated on the antenna screen with the **Close** button. (See Section 3.4.2).

3.7 The View Menu

The **View** menu is show in Figure 3.13. Two options are available under this menu:

View Pattern - View a volumetric pattern using the external **Volumetric Pattern Viewer** program. The default is the vlpv code described in Chapter 6, however, the user may enter a different code name. The default volumetric pattern data file is volpat.dat, and can also be changed by the user. The **Volumetric Pattern Viewer** is then run as a separate program.

View File - View a text file. When selected, the user is prompted to enter the filename of the file to view. A separate window will appear containing the file. The window is read only and does not provide any editing capabilities. There is no limit to the number of file-viewing windows which may be opened at the same time.

3.8 Performing a Run

When a run is initiated by selecting either the **Volumetric Pattern** or **Frequency Sweep** buttons from the main screen of Figure 3.1, several events occur:

- **X-Antenna** determines if an antenna has been defined. If not, the user is prompted to define one.

- The values entered in the data-fields of the antenna screen are checked to ensure they match the type defined for the corresponding parameter. This type was selected in the command-file when the parameter was defined. If they are not of the correct type, the user is prompted to enter the appropriate type and the corresponding data-field is cleared. If a set of parameters are in a data-list which has an indicator, and that indicator is set to its **FALSE** value, no error-checking will be performed on those parameters. Note: Indicators are either **TRUE** or **FALSE**, so no checking of them is necessary.

- The values entered in the pattern or frequency sweep data-fields (for a pattern or frequency sweep run respectively) are checked. The manner of error-checking is identical for each case and consists of two parts. First, the values must be of the correct type, in this case, a valid real number. Next, it is ensured that the increment value divides an integer number of times into its corresponding range. As an example, consider the values V_1, V_2 and ΔVrepresenting the starting, ending and increment values, respectively, of either θ, ϕ or frequency. If ΔV does not divide evenly into $V_2 - V_1$, but instead divides N times with a remainder, ΔV will be modified such that it divides $N + 1$ times (unless the remainder is sufficiently small ($\leq 10^{-3}$) in which case it will not be changed). If $V_2 - V_1 = 0$, or more precisely, if $V_2 - V_1 \leq 10^{-3}$, then it is assumed that only a single point is selected. In this case ΔV is set to 0 and the user is prevented from altering the value. The user's analysis code must take into account the case where $\Delta V = 0$ so as not to produce an error when an increment value equals zero.

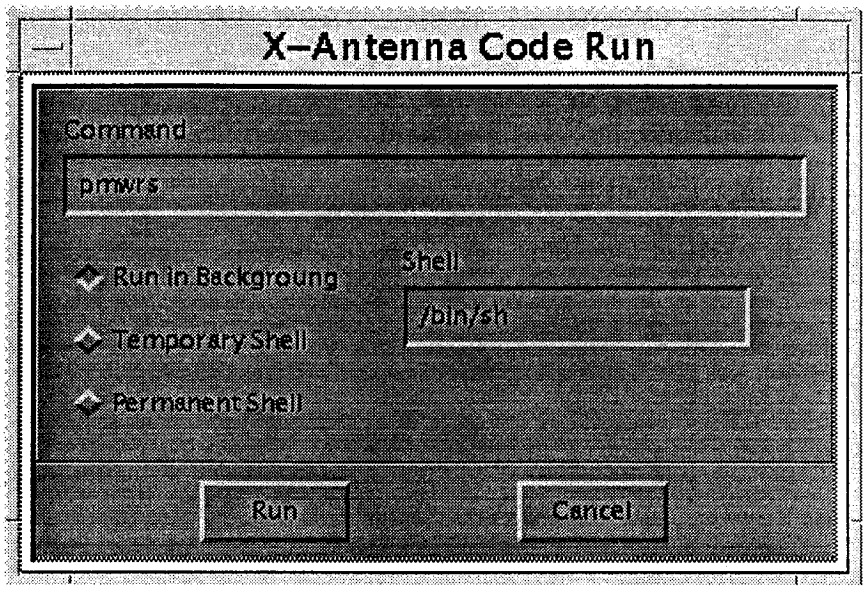


Figure 3.14: Code-Run screen.

- The **X-Antenna** output file is written to disk. The default name of this file is defined in the *Output File* data-field in the main screen. The default name is xant.dat. The format of the output file is covered in Chapter 5

- The code-run screen of Figure 3.14 is presented. The user may enter command-line options for the analysis code in the *Command* data-field. This is the exact command that will be sent to the shell for execution and initially contains only the name of the analysis code. In addition, the user may either run the code in the same window from which **X-Antenna** was executed, or run the code in a separate window by selecting the appropriate toggle button:

  Background = execute the program in the background in the window from which **X-Antenna** was executed.

  Temporary Shell = open a separate window to execute the program. When the program is finished, the window will close when the user presses a key. This allows any output data sent to the screen to be viewed before the window is closed.

  Permanent Shell = same as the previous option except that the window will stay open until the user explicitly closes it.

When a temporary or permanent shell is selected, the shell program specified in the *Shell* data-field is run in the opened window. The user may choose to run a different shell program than the default, */bin/sh*. This may make a difference when a permanent shell is selected since the user may have configuration files for a specific shell other than the default (currently, the author uses the *bash* shell).

Two buttons are available at the bottom of this screen:

Run = run the command specified in the *Command* data-field in the appropriate window.

Cancel = cancel the operation.

Any changes made to the *Command* data-field will remain in effect until the current antenna is closed. Any changes made to the toggle buttons and the *Shell* data-field remain in effect until explicitly changed.

3.9 Example Run

This section will step through the use **X-Antenna** to create an output file and run an antenna analysis program.

1. Execute **X-Antenna** from the command line by entering:

   `xant commandfile`

   where `commandfile` is the name of the command-file. This will cause the main screen of Figure 3.1 to appear.

2. Select an antenna from the **Antenna** menu. As an example, select **Dipole** from the **Wire Antennas** category.

3. Enter the parameters and select the indicator values. Figure 3.15 shows some example values for a dipole antenna.

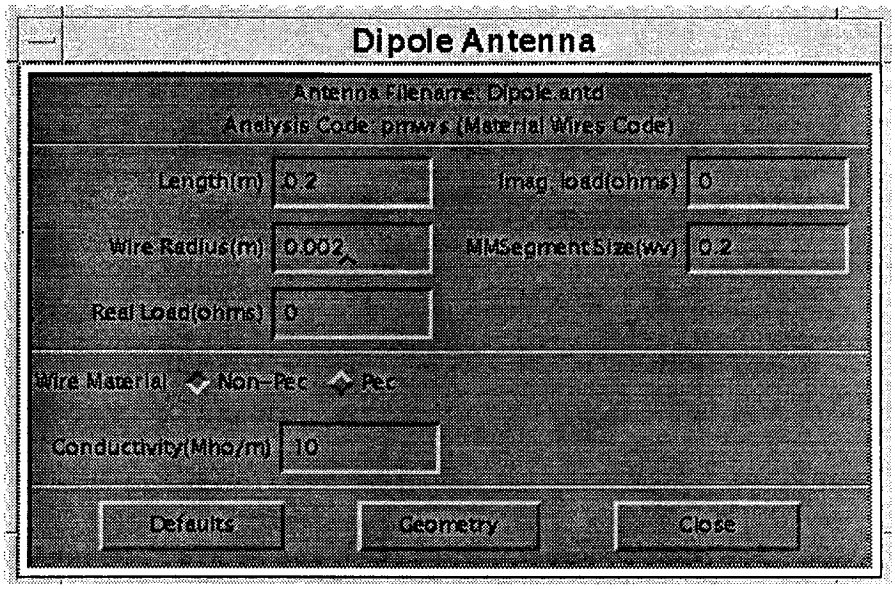


Figure 3.15: Example values for a dipole antenna.

4. View the code-options. In this case, the dipole antenna uses the **MATWRS** [1] code. Select **mwrs** from the **Code** menu. Change any values if desired. Figure 3.16 shows the code-options selected for this run.

5. Enter the pattern or frequency sweep values on the main screen. In this example we will perform a pattern. Figure 3.17 shows some example volumetric pattern values. In this case the pattern is full volumetric.

6. Initiate the run by pressing the **Pattern Sweep** button on the main screen. The code-run screen of Figure 3.14 will appear. Selecting **Run** on this screen will execute the code. Any errors will appear in the window from which the code was run.

This is the **X-Antenna** output file created when the dipole pattern run was initiated:

```
Dipole
1
7
0.200000
0.00200000
10.0000
0.00000
```

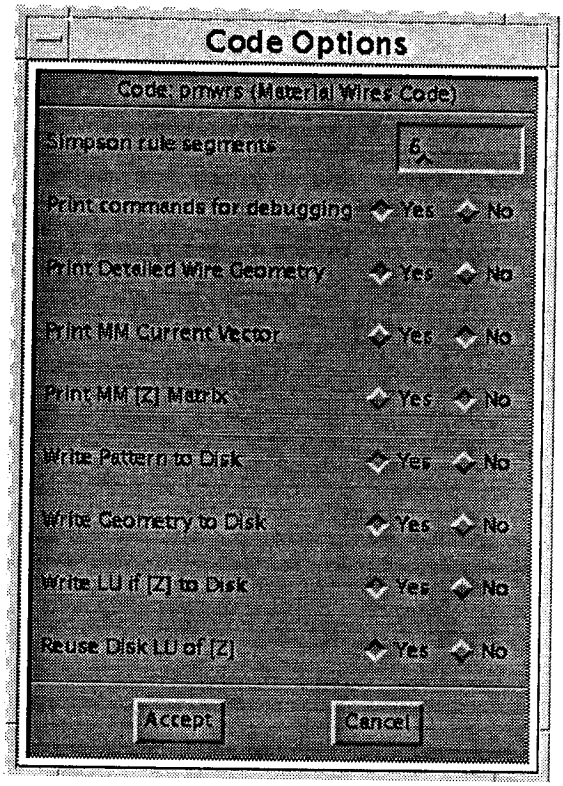


Figure 3.16: Example code-option values for the **MATWRS** code.

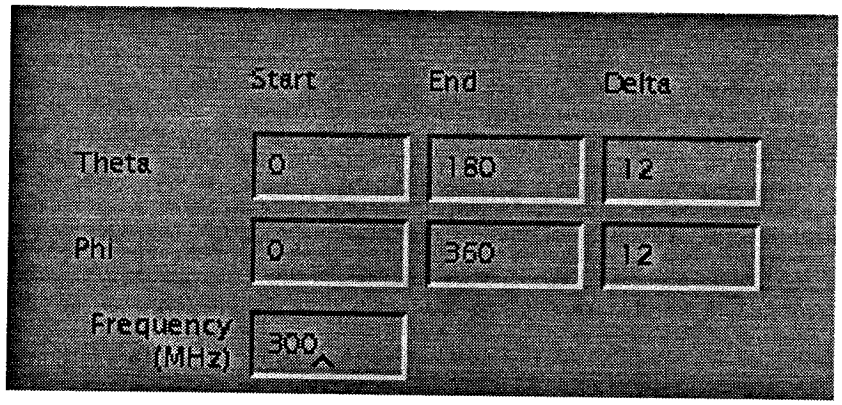


Figure 3.17: Example of a full volumetric pattern.

```
0.200000
1
10.0000
1
0.00000
180.000
3.00000
0.00000
360.000
3.00000
300.000
mwrs
4
1
1
0
0
1
1
0
0
```

Since the **MATWRS** code was executed, an output file `outmwr.dat` should have been created along with any other data files selected by the code-options. This file will contain information for viewing including any errors produced. See the **MATWRS** User's Manual [1] for more details.

Chapter 4

The Command-File

4.1 Introduction

The command-file allows the user to customize **X-Antenna** so that any (reasonably simple) antenna type and any (reasonably simple) analysis code may be used. When **X-Antenna** is executed, the command-file specified on the command line is read. If any semantic errors are found, the program exits and prints an error message. This does not mean all possible errors will be found, but if the command-file format is adhered to properly, it should be fairly simple to isolate where the error originates.

When the command-file is parsed, certain keywords are expected. These keywords will be referred to as command directives. Data usually follows these directives and is expected in a specific format. It is important to follow the format outlined here to obtain consistent results. Internally, the information in the command-file is stored in a large data structure. As command directives and additional data are given, this data structure is appended.

The command directives are divided into two categories: antenna directives and code-option directives. The antenna directives define the parameters and indicators for each antenna (such as parameters and indicators for defining geometry, material properties, etc.), and related information such as the name of the code used to analyze the antenna. The code-option directives control the defining of parameters and indicators for data which a particular analysis code might need. If an antenna uses an analysis code that has code-options defined, those code-options will be written to the output file (see Section 5.1.3) along with the appropriate antenna and pattern or

frequency sweep data. The command-file used to define all antennas and codes used in this manual is listed in Appendix A.

The specification of antenna directives and code-option directives may be intermixed, but the command-file will be easier to understand if the respective commands are grouped together. It does not matter which group of directives are listed first. Sections 4.3 and 4.4 describe the command directives.

4.2 File Parsing

The command-file is parsed on a line-by-line basis. This, along with rules for defining the antenna types and the code-options, leads to the following restrictions for specifying information in the command-file:

- All blank lines are ignored.
- All command directives are capitalized.
- Only one command directive is allowed per line and should be the only data present on that line.
- Some command directives may only appear after other directives have been present in the command-file. The specific ordering depends on the directives and will be outlined in the following sections.
- Data may only appear on a line following the directive, and must conform to the format specified for that directive.
- Character strings may contain blank spaces between the first and last non-whitespace characters which appear on a single line.
- Any trailing or leading spaces in any data, including character strings, are ignored.
- Character strings always appear as the last (or only) data on a line.

Special attention should be paid to the parsing of character strings. To allow more flexibility with category names, antenna names, etc., character strings may consist of several words, each separated by one or more blank spaces. As noted above, any blank spaces before the first or after the last non-blank character are ignored. In this way, a character string defining an antenna name may be written as "Square Loop" instead of the one-word alternate "Square_Loop" which contains no blanks. Care should be taken, however, with this practice since in the case of antenna names, the default file name used for saving the antenna to disk has the antenna name in it. This means a file name might have embedded blanks. While there is no problem with the programs functionality with regard to this, the user does have to remember to quote file names from the UNIX-prompt so the blanks will be interpreted correctly. The author has chosen to use underscores to separate the words in all antenna names to avoid this complication.

To aid the readability of the command-file, it is suggested that data associated with a command directive be placed several spaces or a tab stop over from the command directive itself. Since leading spaces are ignored, this practice will never affect the parsing of the data.

4.2.1 Comments in the Command-File

Any line with either a '!', '%', or a '#' in the first column is ignored and may be used to introduce a comment into the command-file.

4.2.2 Newlines in Parameter and Indicator Labels

To prevent the labels of parameters and indicators in the antenna and code-options screens from becoming too long, newlines may be embedded within a character string with the '\' (backslash) character. There is no means to have this character printed in an label. Newlines should not be embedded in any character strings except labels.

4.3 Antenna Command Directives

The commands in this section define the antennas which will be available from the **Antenna** menu. The commands which define the antennas are entered in a specific order. As outlined in the Chapter 3, an antenna category contains one or more antenna types. Each antenna in turn is defined by one or more data-lists, and each of these data-lists contains parameters, an indicator, or both. When an antenna category is defined, it becomes the currently active category for any antenna types which follow. When an antenna type is defined, it becomes the currently active antenna type for any data-lists which follow. Similarly, when a data-list is defined, it becomes the currently active data-list for any parameters or indicators which follow. In this way, the operation is like a state-machine. This eliminates the need for commands which signify the "end" of a particular definition.

4.3.1 CATEGORY: The Category Command

This command begins a new antenna category. It is used only to organize the antenna types in the **Antenna** menu.

Form of the command:

CATEGORY
Category_Name

Category_Name = name of the antenna category. This name will appear as a sub-menu category under the **Antenna** pull-down menu. Any antennas defined after this command will be included in this category and so will appear under the corresponding sub-menu.

4.3.2 ANTENNA: The Antenna Command

This directive begins a new antenna type.

Form of the command:

ANTENNA

Antenna_Name

Antenna_Number

Antenna_Name = name of the antenna type. This name will appear in the antenna sub-menu category defined with the last CATEGORY command to allow selection of this antenna type. It will also be the default base name used to save the antenna data to disk (see Section 3.3.1). It is suggested that no blank spaces are embedded in this name because of this. This name will also be written to the **X-Antenna** output file (see Section 5.1.1) as part of the antenna data. All data-lists defined after this command will be included in this antenna type.

Antenna_Number = integer number associated with this antenna. This data is only for the users convenience. It will be written the **X-Antenna** output file (see Section 5.1.1) with the other antenna data and may be used to identify the particular antenna type. This number need not be unique.

4.3.3 DATALIST: The Data-List Command

This directive begins a new data-list in the currently active antenna defined by the most recent ANTENNA command. When this directive is specified, the named data-list becomes the currently active data-list for any PARAMETER or INDICATOR commands which follow. This command has no other data associated with it since it only organizes the parameter(s) and/or indicator which may follow.

Form of the command:

DATALIST

4.3.4 PARAMETER: The Parameter Command

This directive defines a parameter to be included in the currently active data-list defined by the most recent DATALIST command. Form of the command:

PARAMETER

Parameter_Name

Data_Type (Default_Value)

Parameter_Name = name of the parameter. This name will appear in the antenna screen to the left of the data-field for the parameter. As described previously, The '\' character may be used to embed new-lines in the character string to prevent it from becoming too long and unnecessarily widening the appearance of the antenna screen.

Data_Type = the data-type of the parameter specified by the word 'int', 'float', or 'char' (uppercase is also acceptable), for an integer, real number, or character string, respectively. Any other word is unrecognized and will produce an error. This parameter will be checked to ensure it matches this data-type (see Section 3.8) when a pattern or frequency sweep run is initiated.

Default_Value (optional) = the default value of the parameter which appears when the antenna screen is first brought up. If no default value is specified, the corresponding data-field in the antenna screen will be left blank. This value is always read in as a character string, so no checking is done to ensure that it conforms to the data-type specified. If a default value does not conform the specified data-type, a new value must be entered on the antenna screen before a run will proceed.

4.3.5 INDICATOR: The Indicator Command

This directive defines the indicator for the currently active data-list defined with the most recent DATALIST command. Only one indicator is allowed per data-list. If any parameters are defined in the same data-list, the indicator will be associated with them (See Section 3.4.1).

Form of the command:

INDICATOR
Indicator_Name
True_Label
False_Label
True_Value False_Value (Default_Value)

Indicator_Name = label for the indicator in the antenna screen. This label is placed to the left of both indicator buttons and their respective labels.

True_Label = label for the **TRUE** valued button for the indicator. This button is the left-most button.

False_Label = label for the **FALSE** valued button for the indicator. This button is the right-most button.

True_Value = integer value associated with the **TRUE** valued button. This value will be printed in the output file if the **TRUE** button is selected.

False_Value = integer value associated with the **FALSE** valued button. This value will be printed in the output file if the **FALSE** button is selected.

Default_Value = default value of the indicator (optional). It is specified either by the word 'TRUE' or the word 'FALSE' for **TRUE** and **FALSE** respectively. If not present, the indicator will default to **FALSE**.

4.3.6 CODENAME: The Code-Name Command

This command specifies the name of the analysis code used to analyze the active antenna-type specified by the ANTENNA command. Only one code-name specification is allowed per antenna type. Form of the command:

CODENAME
Code_Filename

Code_Filename = exact name of the executable program used to analyze the antenna.

It is best to specify the full path-name of the program to ensure it will always be accessible to the program. If only a file-name is specified, it will be searched for in the directory from which **X-Antenna** is run. While blank spaces may be included in this name, the user is strongly advised to refrain from this practice.

4.3.7 BITMAP: The Bitmap Command

This command specifies the file which contains a bitmap file to be used as a descriptive help screen for the active antenna defined be the most recent ANTENNA directive. This help-screen is made visible by the **Geometry** button on the antenna screen. If no bitmap file is specified, the button will be grey-out.

Form of the command:

BITMAP
Bitmap_Filename

Bitmap_Filename = exact name of the the X-bitmap file.

Again, it is best to use the full path-name and not just the name relative to the directory from which **X-Antenna** is run.

4.4 Code-Option Command Directives

The commands in this section define parameters which a particular analysis code might require. When an antenna is selected from the **Antenna** menu, the code-option parameters are searched to see if a match exists between the name of the analysis code used for that antenna and any of the analysis codes for which there are code-options. If a match is found, those code-options are written to the output file along with the antenna and pattern data. Thus, it is very important to use exactly the same name when defining the code here as when specifying an analysis code for an antenna with the CODENAME directive. Note: this is the executable name only, not the entire path which may be specified with the CODENAME directive (see Section 4.4.1).

4.4.1 CODEDEF: The Code Definition Command

This command defines a new code for which there will be code-options. Each code defined with the CODEDEF command may be selected from the **Code** menu. When selected, a code-options screen will appear allowing the user to view (and change if

desired) the code-option values. This screen will contain the information from any CODEPARAM and CODEINDICATOR commands that follow.

Form of the command:

CODEDEF
Code_Name
Code_Description

Code_Name = name of the analysis for which code-options should be defined. This name should exactly match the name of the analysis code defined with the **CODENAME** command. Note: this is the name of the executable code only, NOT the entire path that may be listed under the CODENAME directive. That is, if the full path of an analysis code listed under the CODENAME directive for an antenna is

"/usr/people/blg/analysis/mycode",

then the name used under the CODEDEF command should be

"mycode".

Code_Description = description of the analysis code. Since the name of executable programs are generally cryptic, option allows a description of the code to be stored with the code-name. This description is only used in the antenna and code-option screens and does not affect the operation of the application.

4.4.2 CODEPARAM: The Code Parameter Command

This command defines a parameter for the currently active code defined by the most recent CODEDEF command. This command is similar to the PARAMETER command used for antenna data. The difference is that the data for code-options is not sectioned into separate data-lists. As such, there is no association between indicators and parameters. This simplifies the defining of parameters.

Form of the command:

CODEPARAM

Parameter_Name

Data_Type Default_Value

Parameter_Name = name of the parameter. This name will appear in the code-options screen to the left of the data-field for the parameter. As described previously, the '\' character may be used to imbed new-lines in the character string to prevent it from becoming to long and unnecessarily widening the appearance of the code-options screen.

Data_Type = the data-type of the parameter specified by the word 'int', 'float', or 'char' (uppercase is also acceptable), for an integer, real number, or character string, respectively. Any other word is unrecognized and will produce an error.

Default_Value = the default value of the parameter which appears when the code-options screen is first brought up. This value is not optional as with the default value used with the PARAMETER command for antenna data. This is because code-options should be completely specified when the application runs so the user need not change the values. The default value is always read in as a character string from the command-file, so no checking will have been performed to ensure that it conforms to the data-type specified unless the user has selected the **Accept** button from the **Code-Options** screen (see Section 3.5.2). If the default value does not conform the specified data-type, an error may result when an analysis code attempts to read the data.

4.4.3 CODEINDICATOR: The Code Indicator Command

This command defines an indicator for the currently active code defined by the most recent CODEDEF command. This command is similar to the INDICATOR command used for antenna data. The difference is that the data for code-options is not sectioned into separate data-lists. As such, there is no association between indicators and parameters. Because of this, there is no limit to the number of indicators which may be present.

Form of the command:

CODEINDICATOR

Indicator_Name

True_Label

False_Label

True_Value False_Value Default_Value

Indicator_Name = label for the indicator in the code-options screen. This label is placed to the left of both indicator buttons and their respective labels.

True_Label = label for the **TRUE** valued button for the indicator. This button is always placed on the left-side.

False_Label = label for the **FALSE** valued button for the indicator. This button is always placed on the right-side.

True_Value = integer value associated with the **TRUE** valued button. This value will be printed in the output file if the **TRUE** button is selected.

False_Value = integer value associated with the **FALSE** valued button. This value will be printed in the output file if the **FALSE** button is selected.

Default_Value = default value of the indicator. It is specified either by the word 'TRUE' or the word 'FALSE' for **TRUE** and **FALSE** respectively. As opposed to the default value used with the INDICATOR command, the default value here is not optional. This is because code-options must be completely specified so the user need not bring up a code-options screen before running an antenna analysis code. Note that this default value specifies whether the indicator defaults to **TRUE** or **FALSE** as opposed to specifying the default integer value itself.

Chapter 5

The Output File

This chapter describes the format of the **X-Antenna** output file (default name xant.dat). This file is intended to be read in by an analysis code of the user's choosing. The data in the file consists of the antenna data, the pattern or frequency sweep data, and the code-option data (if applicable) in that order. If a new analysis code is being developed, it can be designed to read in this file directly. An existing code may be altered such that it can read the file, or a pre-processing routine may be written to convert the file to the format of the existing code.

5.1 Output File Format

The output file consists of data in the following three categories written in this order:

- The antenna data.
- The pattern or frequency sweep data.
- The code-option data (if any exist for the particular analysis code for the given antenna).

All data is always written one data-item per line. This is meant to simplify the reading of the output file by the antenna analysis code. When using FORTRAN it can be tedious to read in arbitrary numbers of parameters on a data line since the *read* function always advances to the next line. In C this is not a problem since all whitespace is skipped when looking for data with the *sscanf* function.

5.1.1 Antenna Data

Format of the antenna data:

Antenna_Name
Antenna_Number
Num_Data
(Indicator(1))
Parameter(1,1)
Parameter(1,2)
.....
Parameter(1,N)

.....
.....

(Indicator(M))
Parameter(M,1)
Parameter(M,2)
.....
Parameter(M,N)

Antenna_Name = the name of the antenna defined in the command file using the ANTENNA command. If this name was defined with embedded blank spaces they will be present here.

Antenna_Number = integer number associated with the antenna. It is also defined in the command-file using the ANTENNA command. This number need not be unique. Its purpose is to allow the user another means of identifying the antenna besides the antenna name.

Num_Data = Number of data values (one per line) to follow. Each parameter and indicator are counted as a data value. This is intended to simplify the process

of reading in the antenna data. It eliminates the need to know in advance how many parameters an antenna might have. This is useful for codes which might analyze many antenna types.

Indicator(i) = the value of the indicator in the ith data-list ($i = 1, \ldots, M$) where M is the number of data-lists for the antenna as defined in the command-file. The integer value is either the **TRUE** value or the **FALSE** value defined in the command file with the INDICATOR command and is determined by the user's selection of the indicator in the antenna screen. If no indicator was defined for the ith data-list, this value will not be present in the output file.

Parameter(i, j) = Value of the jth parameter ($j = 1, \ldots, N$) in the ith data-list where N is the number of parameters in the ith data-list as defined in the command-file. This parameter may be an integer, a real number, or a character string. The value of this parameter is determined by the user's input to the corresponding data-field in the antenna screen. If an indicator is present in the data-list in which this parameter is defined, and the indicator is set to its **FALSE** value, this parameter will still be written to the output file. This is meant to preserve the consistency of the output file format and eliminate the need for the user to selectively read in parameters conditional upon particular indicator values. If a data-field for a parameter in a data-list with a **FALSE** valued indicator is blank, a dummy value will be written to the output file in its place. Also, no error checking will be performed on this parameter if a value is present in the data-field.

The order of the antenna data follows the logical order of data which appears in the antenna screen. This order in turn was defined by the order of the PARAMETER and DATALIST commands in the command file for the given antenna type. Thus, there is a one-to-one correspondence between the order of the antenna data specified in the command file, and the order of the antenna data written to the **X-Antenna** output file. The one exception is that an indicator value is always written before any parameter values in a data-list if both are present. Indicators are also physically

placed above parameters in the data-lists in the antenna screen. (See Chapter 3). This preserves the notion that indicators validate any parameters in the same data-list which follow.

5.1.2 Pattern and Frequency Sweep Data

The data in this section of the output file will depend on whether **Volumetric Pattern** or **Frequency Sweep** is selected from the main screen. Note that the parameter *Pattern_Type* is present in both formats and serves to indicate which type of data follows.

Format of pattern data:

Pattern_Type (= 1)
Theta1
Theta2
Delta_Theta
Phi1
Phi2
Delta_Phi
Freq

Pattern_Type = indicator (1 = pattern, 2 = frequency sweep). Here it will be equal to 1 since this is a pattern run.

Theta1, Theta2 = start and end angles (in degrees), respectively, in the θ-direction. Theta1 may equal Theta2 and implies an elevation-plane pattern. If (Theta2 - Theta1) $\leq 10^{-3}$, then Delta_Theta = 0.

Delta_Theta = increment angle (in degrees) in the theta-direction. **X-Antenna** ensures this number is an integral divisor of (Theta2 - Theta1). If Delta_Theta is not an integer, a small round-off error may be present. It is the users responsibility to account for this. Delta_Theta may be negative.

Phi1, Phi2 = start and end angles (in degrees), respectively, in the the phi-direction. Phi1 may equal Phi2 and implies an azimuth-plane pattern. If (Phi2 - Phi1) $\leq 10^{-3}$, then Delta_Phi = 0.

Delta_Phi = increment angle (in degrees) in the ϕ-direction. **X-Antenna** ensures this number is an integral divisor of (Phi2 - Phi1). If Delta_Phi is not an integer, a small round-off error may be present. It is the users responsibility to account for this. Delta_Phi may be negative.

Freq = Frequency (in Hz) of pattern. Note that this value was entered in MHz in the main screen.

Format of frequency sweep data:

Pattern_Type (= 2)
Freq1
Freq2
Delta_Freq
Theta
Phi

Pattern_Type = indicator (1 = pattern sweep, 2 = frequency sweep). Here it will be equal to 2 since this is a frequency sweep.

Freq1, Freq2 = start and end frequencies, respectively, in Hz. Freq1 may equal Freq2. Note that this value was entered in MHz in the main screen.

Delta_Freq = increment frequency in Hz. **X-Antenna** ensures this number is an integral divisor of (Freq2 - Freq1). If Delta_Freq is not an integer, a small round-off error may be present. It is the users responsibility to account for this. Delta_Freq may be negative. If (Freq2 - Freq1 $\leq 10^{-3}$), then Delta_Freq = 0. Note that this value was entered in MHz in the main screen.

Theta, Phi = look angle (in degrees) in the θ and ϕ directions respectively.

5.1.3 Code-Options Data

Code-Option data will be present in the output file if the analysis code specified for the antenna selected has code-option data defined. The defining of this data would have been done in the command file. The order of the data is the same as that in the code-option screen. This order in turn was determined by the order in which it was defined in the command file by the CODEPARAM and CODEINDICATOR commands.

Format of code-option data:

Code_Name
Parameter(1)
.....
.....
.....
Parameter(N)

Indicator(1)
.....
.....
.....
Indicator(M)

Code_Name = name of the analysis code used. This name is defined in the command file using the CODEDEF command.

Parameter(i) = value of ith parameter ($i = 1, \ldots, N$) where N is the number of parameters defined for the named analysis code in the command-file. This may be an integer, real number, or character string.

Indicator(j) = value of jth indicator ($j = 1, \ldots, M$), where M is the number of indicators defined for the named analysis code in the command-file. If the indicator was set to true, this will be the **TRUE** valued integer as set in the command file. If it was set to false, it will be the **FALSE** valued integer.

5.2 Conversion of the Output File

It is unlikely that any existing program will be able to directly read in the output file from **X-Antenna**. The user must then either convert this data to a format that can be interpreted, or alter their program to read in the file directly. In most cases, it is easier to alter the output data to conform to a different format. To aid the user in converting the output data, two generic routines have been written, one in FORTRAN and one in C, which will read in any **X-Antenna** output file. These routines can be modified to write an output file and then run another program, or to call a subroutine directly with the appropriate parameters. The listings for these routines are provided in Appendix B.

The **Microstrip** and **Slot** codes were written as subroutines so that they could be called from any other program. Main calling routines were then written to read in a data file and call the appropriate subroutine. This allowed a main routines to be written specifically for reading **X-Antenna** output files. The listings for these main file-reading routines are provided in Appendix B as an example of this method.

The **MATWRS** [1] code requires an input file with a specific format and cannot be called directly as a subroutine. A conversion program was written which reads in the **X-Antenna** output file and then writes the correctly formatted **MATWRS** input file. The program then calls a UNIX routine to run the **MATWRS** code. If the latter was not done, the user would then have to run the **MATWRS** code from the command line since **X-Antenna** would have only run the conversion program which outputs the correctly formatted output file. A listing of this conversion routine is provided in Appendix B.

If an analysis code can be run as a single subroutine, it is probably easier to take the approach used for the **Slot** and **Microstrip** codes and write a top-level program to read in the **X-Antenna** output file and call the subroutine(s) directly. If the analysis code requires a complicated input file and cannot be called as a subroutine, it is most likely easier to write a file-conversion program as in the case of the **MATWRS**

code. Whether or not this file-conversion program then runs the antenna analysis code directly using UNIX routines is the user's prerogative.

Chapter 6

Using the Volumetric Pattern Viewer

6.1 Introduction

The **Volumetric Pattern Viewer** is a stand alone viewing program for volumetric patterns which can be run directly from **X-Antenna**. The **Volumetric Pattern Viewer** reads in a data file containing the volumetric pattern data and renders a 3-D image of the pattern. To be used in conjunction with **X-Antenna**, the antenna analysis code(s) run by **X-Antenna** must output the appropriate volumetric data file.

The program utilizes the IRIS GL graphics routines available on Silicon Graphics machines and will only run on machines with such capability. Several options are available within the program including a screen dump feature allowing the user to print a hard-copy of the pattern. The view can be rotated through 3-D space, translated and scaled using the mouse as a "virtual trackball" directly on the rendered pattern.

6.2 Running the Volumetric Pattern Viewer from X-Antenna

When **Volumetric Pattern** is selected from the **View** menu, the screen of Figure 6.1 will be presented. This screen has three data-fields:

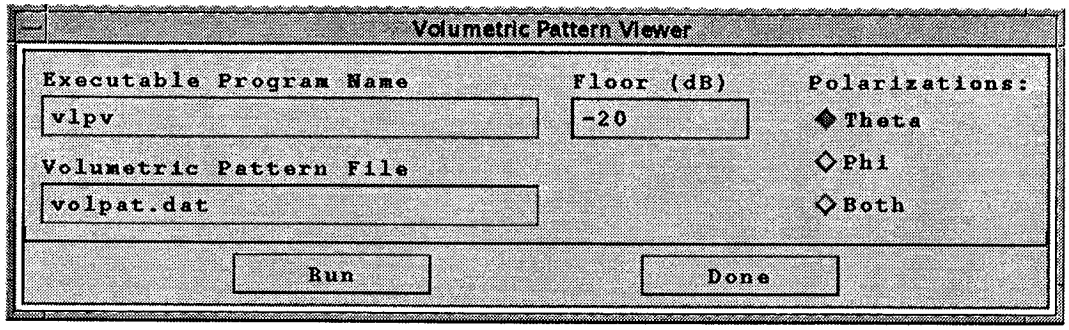


Figure 6.1: Screen for executing the **Volumetric Pattern Viewer** from **X-Antenna.**

Executable Program Name = name of the **Volumetric Pattern Viewer** executable code. The default is *vlpv.*

Volumetric Pattern File = name of the volumetric pattern file to read in. The default is volpat.dat. This file must conform to format outlined in Section 6.6.

Floor = the minimum magnitude (in dB) to render. All values lower than this floor will be set equal to the floor. This allows the user to scale the data to a different range. The minimum value is -99 dB, and we employ a default value of -20 dB.

The user must select what polarizations to render from the toggle buttons **Theta, Phi** and **Both** which will render θ, ϕ and both θ and ϕ polarizations, respectively.

There are two buttons available at the bottom of the screen:

Run = execute the **Volumetric Pattern Viewer** with the values specified.

Done = close the screen.

When **Run** is selected, **X-Antenna** sends a formatted command to the UNIX shell to run the **Volumetric Pattern Viewer.** If any errors occur in trying to run the program, the user will not be notified other than any error messages produced by the shell.

6.3 Running the Volumetric Pattern Viewer from the Command Line

Command line usage of the **Volumetric Pattern Viewer:**

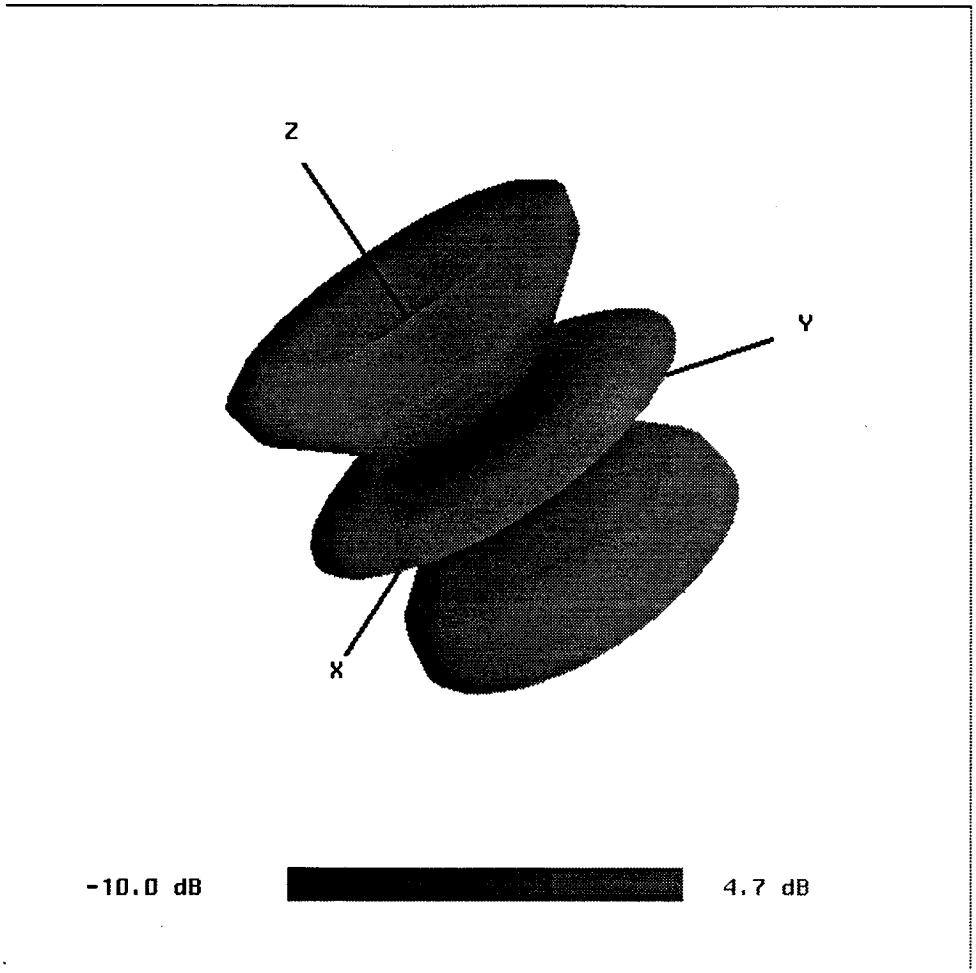


Figure 6.2: Lobe-type pattern for a dipole antenna.

```
vlpv datafile <-theta | -phi | -both>
```

datafile = the volumetric pattern file to be read in.

-theta, -phi, -both = options to render θ or ϕ-polarizations or both, respectively. If none is specified, both polarizations will be rendered.

6.4 Operation

Figure 6.2 shows the application with a dipole gain-pattern rendered and Figure 6.3 shows the option-menu. This menu is activated by depressing the 3rd mouse button anywhere on the screen. The menu options are:

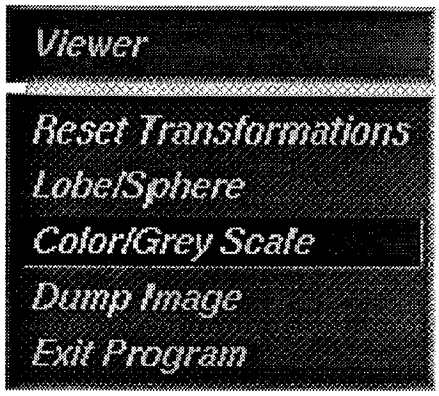


Figure 6.3: **Volumetric Pattern Viewer** option menu.

Lobe/Sphere = toggle between rendering the pattern as a constant-radius sphere or as a lobe-type pattern. Figure 6.2 shows the lobe-type pattern while Figure 6.4 shows the sphere-type pattern for the same dipole antenna.

Color/Grayscale = toggle between color and grey-scale indexing. Color indexing is more visually pleasing while grey-scale is more suited for producing hard-copies.

Screen Dump = dump a RGB file of the screen to disk. While there are utilities that allow selected areas of a screen to be scanned and dumped to disk, caution must be used since these utilities might use the default colormap of the screen. The **Volumetric Pattern Viewer** re-maps the colormap and the screen dump feature takes into account. The name of the RGB file is `viewer.rgb`.

Exit = exit the program.

6.5 Rotating, Translating and Scaling the Pattern

The rendered pattern can be rotated, moved and scaled through 3-D space using the mouse directly on the rendered object. The first and second mouse buttons control the behavior as follows:

- Middle mouse button down - Rotate the rendered object. The behavior of the rotation is similar to that of a trackball being moved with the palm of the hand.

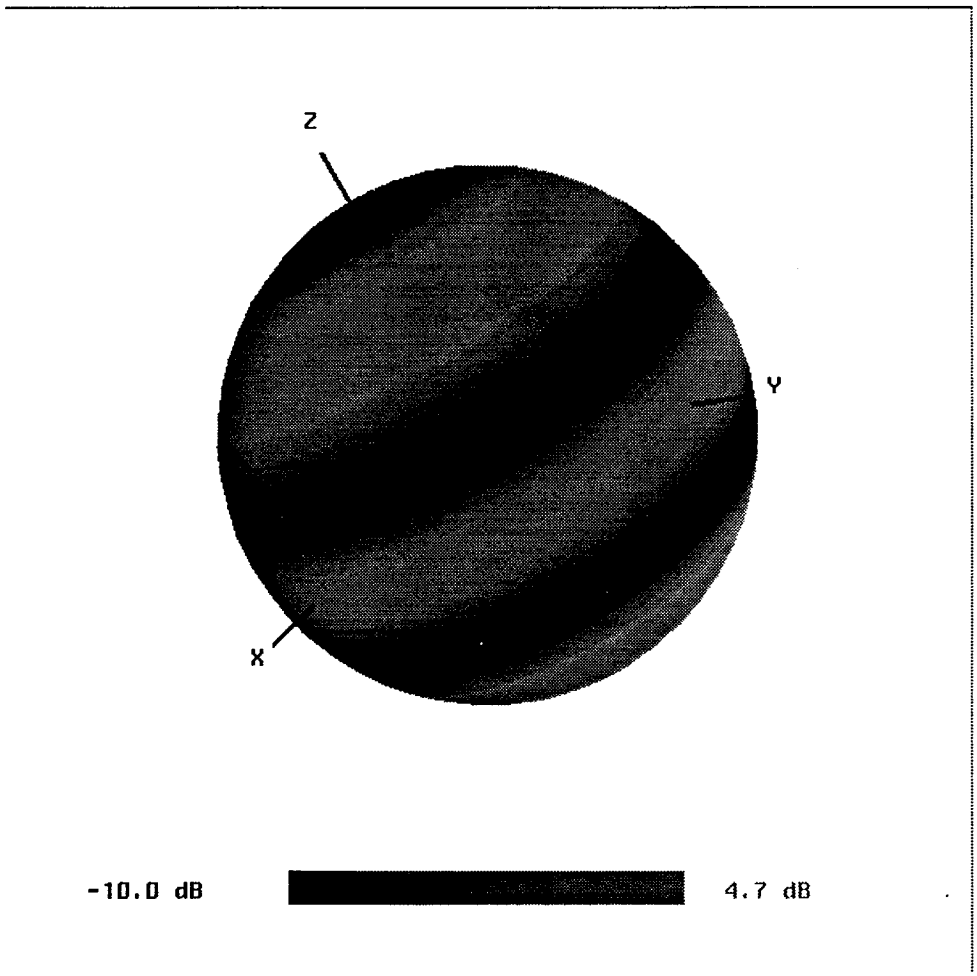


Figure 6.4: Sphere-type pattern for a dipole antenna.

- Left mouse button down - Move the rendered object across the screen.
- Both left and middle mouse buttons down - Scale the rendered object. In general, moving the mouse to the top of the screen will enlarge the object, while moving the mouse to the bottom of the screen will reduce the object.
- Right mouse button down - bring up options menu.

6.6 The Volumetric Pattern Viewer Input File

The **Volumetric Pattern Viewer** reads in a file containing the volumetric pattern information to construct the 3-D pattern. This file contains a header followed by the pattern data. It is important to understand how the data points are ordered to achieve correct results.

An azimuth-cut is a pattern in which θ is fixed and ϕ varies. An elevation-cut is a pattern in which ϕ is fixed and θ varies. The data file organizes points into groups of azimuth-cuts. Each azimuth cut must contain the same number of data points.

Form of the volumetric pattern input file:

type freq rad
npts_th th1 th2 dth
npts_ph ph1 ph2 dph

theta(1,1) phi(1,1) gain_th phase_th gain_ph phase_ph
.....
theta(1,M) phi(1,M) gain_th phase_th gain_ph phase_ph

theta(2,1) phi(2,1) gain_th phase_th gain_ph phase_ph
.....
theta(2,M) phi(2,M) gain_th phase_th gain_ph phase_ph

.....

.....

theta(N,1) phi(N,1) gain_th phase_th gain_ph phase_ph

.....

theta(N,M) phi(N,M) gain_th phase_th gain_ph phase_ph

type = indicator of sweep type (1 = angle sweep (i.e. patterns), 2 = frequency sweep). Any value except 1 will cause the program to exit. This is to ensure only pattern data is used with this program.

freq = frequency of pattern. This included only for compatibility with other programs. Any value may be placed here as a dummy value.

rad = radius of pattern (in meters) field points. This value ignored but used for compatibility with other programs.

npts_th = Number of points in the pattern in the θ-direction. This is the number of azimuth-cut.

npts_ph = Number of points in the pattern in the ϕ-direction. This is the number of points in each azimuth-cut in the pattern.

th1, th2 = start and end values (in degrees) of pattern in the θ-direction. Th1 must equal 0 and th2 must equal 180 for a full volumetric pattern. Currently these values are ignored since only full volumetric patterns are allowed.

ph1, ph2 = start and end location (in degrees) of pattern in the ϕ-direction. Phi1 must equal 0 and phi2 must equal 360 for a full volumetric pattern. Currently these values are ignored since only full volumetric patterns are allowed.

dth, dph = increment (in degrees) in the θ and ϕ-directions respectively. Currently these values are ignored but are left in for compatibility with other programs.

theta(i, j) = value of θ (in degrees) for the jth data point ($j = 1, \ldots, N$) where $N =$ npts_th in the ith azimuth-cut ($i = 1, \ldots, M$) where $M =$ npts_ph. This value is ignored but left in for compatability with other programs.

phi(i,j) = value of ϕ (in degrees) for the jth data point ($j = 1,\ldots,N$) where N = npts_th in the ith azimuth-cut ($i = 1,\ldots,M$) where M = npts_ph. This value is ignored but left in for compatability with other programs.

gain_theta, gain_phi = gain of θ and ϕ-polarized fields, respectively, (in dB) at corresponding (θ,ϕ) location.

phase_theta, phase_phi = phase (in degrees) of θ and ϕ-polarized fields, respectively, at corresponding (θ,ϕ) location. Currently these values are ignored but are left in for compatibility with other programs.

An output file of this format may be produced by using the FORTRAN subroutine `vmpattern` listed in Appendix E.

Chapter 7

Conclusion

This report describes the **X-Antenna** code which provides graphical interfaces which simplify the entering of data for antenna analysis codes. Data which describes the antenna geometry, the pattern or frequency sweep to be run, and various analysis code options are entered via a series of graphical interface windows. These windows are defined, and can easily be changed, via a command file. It is emphasized that **X-Antenna** contains no electromagnetics, but rather serves as a graphical or Motif interface to the users antenna analysis codes. It is our experience that **X-Antenna** allows the user to quickly and easily define and modify antenna parameters, and then to view the results. This should make **X-Antenna** a useful design or educational tool. Finally, we comment that in the future it is hoped that the volumetric pattern files generated by **X-Antenna** can be incorporated into the "Helicopter Antenna Radiation Prediction Code" (**HARP**) [8] to determine the radiation pattern of the antennas on a complicated platform such as a helicopter or an aircraft.

Appendix A

Sample Command File

```
# Wire Antenna Category
CATEGORY
Wire Antennas

# Dipole Antenna
ANTENNA
Dipole
1
PIXMAP
bitmaps/dipole.xbm
CODENAME
pmwrs

# The first data-list of the antenna
DATALIST
PARAM
Length(m)
float
PARAM
Wire Radius(m)
float
PARAM
Real Load(ohms)
float 0
PARAM
Imag. load(ohms)
float 0
PARAM
MMSegment Size(wv)
float 0.2

# This data-list has an indicator
DATALIST
INDICATOR
Wire Material
Non-Pec
Pec
1 0 false
PARAM
```

```
Conductivity(MMho/m)
float 10


ANTENNA
Square_Loop
2
PIXMAP
bitmaps/sq_loop.xbm
CODENAME
pmwrs
DATALIST
PARAM
Side Length(m)
float
PARAM
Wire Radius(m)
float
PARAM
Real Load(ohms)
float 0
PARAM
Imag Load(ohms)
float 0
PARAM
MM Segment Size(wv)
float 0.2
DATALIST
INDICATOR
Wire Material
Non-Pec
Pec
1 0 false
PARAM
Conductivity(MMho/m)
float 10

ANTENNA
Circular_Loop
3
PIXMAP
bitmaps/circ_loop.xbm
CODENAME
pmwrs
DATALIST
PARAM
Loop Raduis(m)
float
PARAM
Wire Radius(m)
float
PARAM
Real Load(ohms)
float 0
```

```
PARAM
Imag Load(ohms)
float 0
PARAM
MM Segment Size(wv)
float 0.2
PARAM
Min. Polygon(wv)
int  8
DATALIST
INDICATOR
Wire Material
Non-Pec
Pec
1 0 false
PARAM
Conductivity(MMho/m)
float 10


ANTENNA
Helix
4
PIXMAP
bitmaps/helix.xbm
CODENAME
pmwrs
DATALIST
PARAM
Bottom Radius(m)
float
PARAM
Top Radius(m)
float
PARAM
Spacing(m)
float
PARAM
Beginning Height(m)
float
PARAM
Feed Radius(m)
float
PARAM
Helix Height+\Feed Height(m)
float
PARAM
Real Load(ohms)
float 0
PARAM
Imag Load(ohms)
float 0
PARAM
Min. Polygon Sides(m)
```

```
int
PARAM
Wire Radius(m)
float
PARAM
MM Segment Size(wv)
float 0.2
DATALIST
INDICATOR
Wire Material
Non-Pec
Pec
1 0 false
PARAM
Conductivity(MMho/m)
float 10
DATALIST
INDICATOR
Polarization
RHP
LHP
1 2 True
DATALIST
INDICATOR
Ground Plane
Yes
No
1 0 True
DATALIST
INDICATOR
Matching Cup
Yes
No
1 0 False
PARAM
Cup Radius(m)
float
PARAM
Cup Height(m)
float
PARAM
Minimum\ Polygon Sides
int 6
PARAM
Min. Rings Base
int 3
PARAM
Min. Rings Side
int 3
PARAM
MM Segment Size(wv)
float 0.2
```

```
CATEGORY
Microstrip Antennas
ANTENNA
Rectangular_Microstrip
6
PIXMAP
bitmaps/mstrip.xbm

CODENAME
mstripx
DATALIST
PARAM
Width(m)
float
PARAM
Length(m)
float
PARAM
Feed Distance
float 0
PARAM
Substrate Thickness
float
PARAM
Relative Epsilon
float 1.0
PARAM
Loss Tangent
float


CATEGORY
Slot Antennas
ANTENNA
Thin_Slot
7
PIXMAP
bitmaps/slot.xbm
CODENAME
slotx
DATALIST
PARAM
Width(m)
float
PARAM
Length(m)
float

# Code Definitions
CODEDEF
pmwrs
Material Wires Code
CODEPARAM
Simpson rule segments
```

```
int 4
CODEINDICATOR
Print commands for debugging
Yes
No
1 0 True
CODEINDICATOR
Print Detailed Wire Geometry
Yes
No
1 0 True
CODEINDICATOR
Print MM Current Vector
Yes
No
1 0 False
CODEINDICATOR
Print MM [Z] Matrix
Yes
No
1 0 False
CODEINDICATOR
Write Pattern to Disk
Yes
No
1 0 True
CODEINDICATOR
Write Geometry to Disk
Yes
No
1 0 True
CODEINDICATOR
Write LU if [Z] to Disk
Yes
No
1 0 False
CODEINDICATOR
Reuse Disk LU of [Z]
Yes
No
1 0 False
```

Appendix B

Listings for Output File Conversion Routines

This appendix contains listings for several **X-Antenna** output file conversion routines including:

- Generic FORTRAN and C routines to read in an arbitrary **X-Antenna** output file.
- The main routines which read in **X-Antenna** output files and call the subroutines for the **Microstrip** and **Slot** codes.
- The file conversion program used by the **MATWRS** [1] code which reads in an **X-Antenna** output file, writes out a **MATWRS** input file and runs the **MATWRS** code.

It is assumed that the user is familiar with the C and FORTRAN programming languages in order to interpret the listing accordingly.

B.1 Generic FORTRAN Routine: *convert.f*

This routine reads in an arbitrary **X-Antenna** output file. The antenna data is read in as character string. If a subroutine is called, these character strings must be converted to the appropriate data type using FORTRAN routines. If an input file to another program is being created, there is no need to convert the data as it can be written out as a character string.

```
      program convert

      implicit none

      character*50  code_name
      character*50  ant_name
      character*50  params[100]
      integer       ant_type
      integer       num_params

      integer    isweep
      real       th1,th2,dth
      real       ph1,ph2,dph
      real       ps_freq
      real       fr1,fr2,dfr
      real       fs_theta,fs_phi

c     open the X-Antenna output file
      open(unit=10,file='xant.dat',status='old')

c     open the input file for the analysis code if needed
      open(unit=10,file='convert.dat',status='unknown')

      read(10,200)ant_name
 200  format(a)

      read(10,*)ant_type
      read(10,*)num_params

      do i=1,num_params
         read(10,*)param(i)
      end

      read(10,*)isweep

      if(isweep .eq. 1) then
         read(10,*)th1,th2,dth
         read(10,*)ph1,ph2,dph
         read(10,*)ps_freq
      elseif (isweep .eq. 2) then
         read(10,*)fr1,fr2,dfr
         read(10,*)fs_theta,fs_phi
      else
         print*,'Error: Must select either pattern or frequency sweep.'
         stop
      end if

      read(10,100)code_name
 100  format(a)

c
c     read in code data here if necessary
c


c
c     write out data or call subroutine here.
c
      end
```

B.2 Generic C Routine: *convert.c*

This routine reads in an arbitrary **X-Antenna** output file. The antenna data is read in as a character string. If a subroutine is called, these character strings must be converted to the appropriate data type using C library routines. If an input file to another program is being created, there is no need to convert the data as it can be written out as a character string.

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>

#define NAMEMAX 100

int main(int argc, char **argv)
{
  int   i,numParams;
  char  antName[NAMEMAX];
  char  codeName[NAMEMAX];
  char  antParams[50][20];
  char  codeData[50][20];
  float patParam[7],freqParam[5];
  int   patType,antNumber;
  FILE  *fp1,*fp2;

/* open the X-antenna data file */

    if((fp1 = fopen("xant.dat", "r"))==NULL)
      {
fprintf(stderr,"Cannot open X-Antenna output file\n");
exit(1);
      }

/* open the input file for the analysis program if needed */

    if((fp2 = fopen("convert.dat", "w"))==NULL)
       {
 fprintf(stderr,"Cannot open the analysis code input file\n");
 exit(1);
       }

/* read in antenna data */

  fscanf(fp1,"%s %i",antName,&antNumber);
  fscanf(fp1,"%i",&numParams);

  for(i=0;i<numParams;i++)
    fscanf(fp1,"%s",&antParams[i]);

/* read in pattern data */

  fscanf(fp1,"%i",&patType);

  if(patType == 1)
    for(i=0;i<7;i++)
      fscanf(fp1,"%f\n",&patParam[i]);
  else if(patType == 2)
    for(i=0;i<5;i++)
      fscanf(fp1,"%f\n",&freqParam[i]);
  else
```

```
    fprintf(stderr,"Error: must select either pattern or frequency sweep");

  fscanf(fp1,"%s",codeName);

  /*
   * read in code data here if necessary
   */

  /*
   * write out data or call subroutine here.
   */

  /* Close the files */

  fclose(infile);
  fclose(outfile);
}
```

B.3 Conversion Routine for the Slot Code

This routine reads in an **X-Antenna** output file and calls the `slot` subroutine directly. This eliminates the need to create an additional file to be read in by the **Slot** code. Because this routine is specifically for the **Slot** code, all data is read in as the appropriate data type, i.e. integer, real number or character string. The `slot` subroutine may now be called using the data read from the file. Note that in this case there are no code-options parameters as the **Slot** code does not require any.

```
c*******************************************************************************
c
c     slotx -  Program to run the slot subroutine from an X-Antenna data File.
c
c     Written By:
c                     Brent Goldstein
c                     ElectroScience Laboratory
c                     The Ohio State University
c                     1320 Kinnear Rd.
c                     Columbus, OH 43212
c                     (614) 292-7981
c                     blg@lenz.eng.ohio-state.edu
c
c*******************************************************************************
      program slotx

      implicit none

      character  code_name*50
      character  ant_name*50
      integer    ant_type,num_params
      integer    scale

      real       width
      real       length
      integer    isweep
      real       th1,th2,dth
      real       ph1,ph2,dph
      real       ps_freq
```

```
      real       fr1,fr2,dfr
      real       fs_theta,fs_phi

      open(unit=10,file='xant.dat',status='old')

      read(10,200)ant_name
  200 format(a)

      read(10,*)ant_type
      read(10,*)num_params
      read(10,*)width,length
      read(10,*)isweep

      scale = 1e6

      if(isweep .eq. 1) then
         read(10,*)th1,th2,dth
         read(10,*)ph1,ph2,dph
         read(10,*)ps_freq
         ps_freq = ps_freq/1e6          !convert to MHz
      elseif (isweep .eq. 2) then
         read(10,*)fr1,fr2,dfr
         read(10,*)fs_theta,fs_phi
         fr1 = fr1/scale                !convert to MHz
         fr2 = fr2/scale
         dfr = dfr/scale
      else
         print*,'Error: Must select either pattern or frequency sweep.'
         stop
      end if

      call slot(
     &   width,length,
     &   isweep,
     &   ps_freq,th1,th2,dth,ph1,ph2,dph,
     &   fr1,fr2,dfr,fs_theta,fs_phi)

      end
```

B.4 Conversion Routine for the Microstrip Code

This routine reads in an **X-Antenna** output file and calls the **mstrip** subroutine directly. This eliminates the need to create an additional file to be read in by the **Microstrip** code. Because this routine is specifically for the **Microstrip** code, all data is read in as the appropriate data type, i.e. integer, real number or character string. The **mstrip** subroutine may now be called using the data read from the file. Note that in this case there are no code-options parameters as the **Microstrip** code does not require any.

```
c*******************************************************************************
c
c     mstripx -  Program to run the mstrip subroutine from an X-Antenna data File.
c
c     Written By:
```

```
c                   Brent Goldstein
c                   ElectroScience Laboratory
c                   The Ohio State University
c                   1320 Kinnear Rd.
c                   Columbus, OH 43212
c                   (614) 292-7981
c                   blg@lenz.eng.ohio-state.edu
c
c*******************************************************************************
      program mstripx

      implicit none

      character  code_name*50
      character  ant_name*50
      integer    ant_type,num_params
      integer    scale

      real       width,length,feed,T
      real       eps_rel,tand
      integer    isweep
      real       th1,th2,dth
      real       ph1,ph2,dph
      real       ps_freq
      real       fr1,fr2,dfr
      real       fs_theta,fs_phi

      open(unit=10,file='xant.dat',status='old')

      read(10,200)ant_name
 200  format(a)

      read(10,*)ant_type
      read(10,*)num_params
      read(10,*)width,length,feed,T,eps_rel,tand
      read(10,*)isweep

      scale = 1e6

      if(isweep .eq. 1) then
         read(10,*)th1,th2,dth
         read(10,*)ph1,ph2,dph
         read(10,*)ps_freq
         ps_freq = ps_freq/1e6          !convert to MHz
      elseif (isweep .eq. 2) then
         read(10,*)fr1,fr2,dfr
         read(10,*)fs_theta,fs_phi
         fr1 = fr1/scale                !convert to MHz
         fr2 = fr2/scale
         dfr = dfr/scale
      else
         print*,'Error: Must select either pattern or frequency sweep.'
         stop
      end if

      call mstrip(
     &   width,length,T,feed,eps_rel,tand,
     &   isweep,
     &   ps_freq,th1,th2,dth,ph1,ph2,dph,
     &   fr1,fr2,dfr,fs_theta,fs_phi)

      end
```

B.5 Conversion program for the MATWRS Code

This routine, (pmwrs), creates a **MATWRS** input file from an **X-Antenna** output file. The routine then runs the **MATWRS** code directly eliminating the need to run it from the command line after **X-Antenna** runs the conversion program. Also listed is an **X-Antenna** output file for a dipole antenna and the corresponding **MATWRS** input file created by the conversion program.

B.5.1 MATWRS Conversion Program Listing

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>

# define LIMIT 100

int main(void)
{
/* Run Control Commands */
  char *END = "END: End of Commands\n";
  char *FMZ = "FMZ: Frequency\n";
  char *INT = "INT: Integration Parameter\n";
  char *NWG = "NWG: Number of Wire Geometries\n";
  char *PRC = "PRC: Print Commands\n";
  char *RUN = "RUN: Run Data\n";
  char *WRI = "WRI: Write Indicators\n";
  char *ZIO = "ZIO: [Z] Matrix I/O\n";
  char *MES = "MES: Message\n";
  char *REM = "REM: Remark\n";

/* Wire Geometry Commands */
  char *CSG = "CSG: Conductor Segments\n";
  char *MSG = "MSG: Material Segments\n";
  char *SLV = "SLV: Dielectric Sleeves\n";
  char *XYZ = "XYZ: XYZ Point Coordinates\n";
  char *FPT = "FPT: Feed Points\n";
  char *WGM = "WGM: Subroutine WGEOMM\n";

/* Desired Pattern Commands */
  char *VMR = "VMR: Volumetric Radiation Pattern\n";
  char *ELR = "ELR: Elevation Plane Radiation Pattern\n";
  char *AZR = "AZR: Azimuth Plane Radiation Pattern\n";
  char *ELS = "ELS: Elevation Plane Scattering Pattern\n";
  char *AZS = "AZS: Azimuth Plane Scattering Pattern\n";
  char *BSA = "BSA: Bistatic Angle\n";
  char *NZP = "NZP: Near Zone Pattern\n";
  char *IMG = "IMG: Image Wave\n";

/* Other Commands */
  char *FSP = "FSP: Frequency Sweep Computation\n";
  char *CPL = "CPL: Mutual Coupling Computation\n";

  int   i,numParams;
  char  codeName[LIMIT],antName[LIMIT];
  float MMSeg;
  int   simpInt;
  int   codeInd[10];
  float patParam[7];
  int   patType,antType;
```

```
  char  antParams[50][20];
  FILE  *fp,*infp;

  fprintf(stderr,"Enter mwrs pre-process\n");

/*
 * open the MATWRS input file
 */
    if((fp = fopen("inmwr.dat", "w"))==NULL)
       {
 fprintf(stderr,"Cannot open mwrs input file\n");
 exit(1);
       }

/*
 * open the X-antenna output file
 */

  if((infp = fopen("xant.dat", "r"))==NULL)
      {
fprintf(stderr,"Cannot open X-Antenna output file\n");
exit(1);
      }

/*
 * read in antenna data
 */

  fscanf(infp,"%s %i",antName,&antType);
  fscanf(infp,"%i",&numParams);

  for(i=0;i<numParams;i++)
    fscanf(infp,"%s",&antParams[i]);

/*
 * read in pattern data
 */

  fscanf(infp,"%i",&patType);
  for(i=0;i<7;i++)
    fscanf(infp,"%f\n",&patParam[i]);


/*
 * read in code data
 */

  fscanf(infp,"%s",codeName);
  fscanf(infp,"%i",&simpInt);
  for(i=0;i<8;i++)
    {
      fscanf(infp,"%i",&codeInd[i]);
    }

/* Comments at the top */
  fprintf(fp, "REM: Input File for MATWRS Created by X-Antenna Code\n");

/* Print Commands */
  if(codeInd[0])
    fprintf(fp, "%s", PRC);

/* Message Command */
  fprintf(fp, "%s", MES);
  fprintf(fp, "Input File for MATWRS Created by X-Antenna Code\n");

/* Run Command */
  fprintf(fp, "%s", RUN);
```

```
/* Frequency in MHz */
  fprintf(fp, "%s", FMZ);
  fprintf(fp, "%f\n", patParam[6]);

/* Integration Parameter */
  fprintf(fp, "%s", INT);
  fprintf(fp, "%i\n", simpInt);

/* Write Indicators */
  fprintf(fp, "%s", WRI);
  fprintf(fp, "%i  %i  %i  %i  %i\n",codeInd[1],codeInd[2],codeInd[3]
  ,codeInd[4],codeInd[5]);

/* [Z] Matrix I/O */
  fprintf(fp, "%s", ZIO);
  fprintf(fp, "%i  %i\n", codeInd[6], codeInd[7]);

/* Wire geometry to be generated in subroutine WGEOMM */
  fprintf(fp, "%s", WGM);

/* Volumetric Radiation Patterns */
  fprintf(fp,"%s",VMR);
  fprintf(fp,"%f %f %f\n",patParam[0],patParam[1],patParam[2]);
  fprintf(fp,"%f %f %f\n",patParam[3],patParam[4],patParam[5]);

/* End of Commands */
  fprintf(fp, "%s", END);

/* Antenna Parameters */
  fprintf(fp, "%i  %i  (%s Antenna, No. of Parameters)\n",
  antType, numParams, antName);
  for(i=0; i<numParams; i++)
    fprintf(fp,"%s\n",antParams[i]);

/* Close the files */
  fclose(fp);
  fclose(infp);

/* overlay this program with MATWRS code */

  fprintf(stderr,"Overlay with mwrs\n");
  execl("./mwrs","mwrs",(char *)0);
}
```

B.5.2 Example Conversion of X-Antenna Output File to MATWRS Input File

The following is a listing of an **X-Antenna** output file for a dipole antenna:

```
Dipole
1
7
0.200000
0.00100000
10.0000
0.00000
0.200000
1
```

```
10.0000
1
0.00000
180.000
5.00000
0.00000
360.000
5.00000
300.000
mwrs
4
1
1
0
0
1
1
0
0
```

This is the **MATWRS** input file, `inmwr.dat`, created by the conversion program:

```
REM: Input File for MATWRS Created by X-Antenna Code
PRC: Print Commands
MES: Message
Input File for MATWRS Created by X-Antenna Code
RUN: Run Data
FMZ: Frequency
300.000000
INT: Integration Parameter
4
WRI: Write Indicators
1  0  0  1  1
ZIO: [Z] Matrix I/O
0  0
WGM: Subroutine WGEOMM
VMR: Volumetric Radiation Pattern
0.000000 180.000000 5.000000
0.000000 360.000000 5.000000
END: End of Commands
1  7  (Dipole Antenna, No. of Parameters)
0.200000
0.00100000
10.0000
0.00000
0.200000
```

```
1
10.0000
```

Appendix C

The Slot Code

C.1 Introduction

The **Slot** code analyzes thin-slot antennas and performs either pattern or frequency sweep analysis of the antenna gain [7]. The pattern sweep is defined in terms of a volumetric pattern (partial or full) and the frequency. The pattern, if properly specified, can be equivalent to an azimuth or elevation cut. The frequency sweep is defined in terms of the frequency range and the far-field angle (θ,ϕ). Both the pattern and frequency sweep analysis are assumed to be in the far-field. The code is in FORTRAN.

C.2 Running the Slot code

The **Slot** code may be run from the main routine, `slotf`, which reads in a data file containing the antenna data and both pattern angle and frequency sweep information. This routine then calls the `slot` subroutine which performs all the computations. The `slot` subroutine may be included in another program and called directly with the appropriate parameters. Figure C.1 shows the geometry of the slot. It is assumed that the slots are rectangular with a length and width no longer than one-half wavelength. Figure C.1 shows the geometry of the slot antenna.

Form of the data file read in by `slot_main`:

width length

isweep

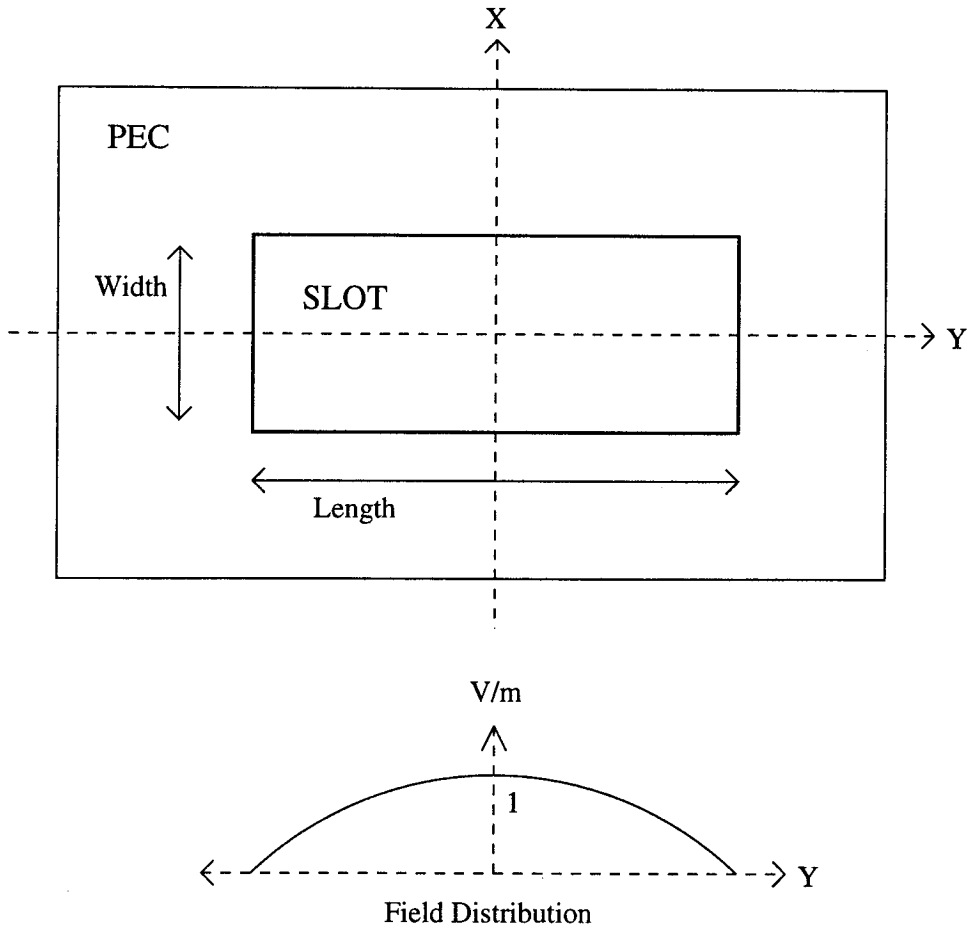


Figure C.1: Geometry for the thin-slot antenna.

th1 th2 dth

ph1 ph2 dph

ps_freq

fr1 fr2 dfr

fs_theta fs_phi

width, length = width and length respectively of the slot in meters. Both width and length must be one-half wavelength or smaller.

isweep = indicator for pattern or frequency sweep analysis. (1 = pattern, 2 = frequency).

th1, th2 = starting and ending angles, respectively, (in degrees) for the pattern in the θ-direction.

dth = increment angle (in degrees) for the pattern sweep in the θ-direction. If dth does not divide evenly into (th2 - th1), but instead divides N times with a remainder greater than 10^{-3}, dth will be modified to divide $N + 1$ times into (th2 - th1). If dth is not an integer, there may be small roundoff error.

ph1, ph2 = starting and ending angles, respectively, (in degrees) for the pattern in the ϕ-direction.

dph = increment angle (in degrees) for the pattern in the ϕ-direction. The value of dph will be altered in an identical manner as dth.

ps_freq = frequency in MHz for pattern.

fr1, fr2 = starting and ending frequencies (in MHz) of the frequency sweep.

dfr = increment frequency (in MHz) of frequency sweep. The value of dfr will be altered in an identical manner as dth and dph.

fs_theta, fs_phi = look angles (in degrees) for the frequency sweep in the θ and ϕ-directions respectively.

Form of the subroutine call to `slot`:

```
 call slot(
&   width,length,
&   isweep,
&   ps_freq,th1,th2,dth,ph1,ph2,dph,
&   fr1,fr2,dfr,fs_theta,fs_phi)
```

The subroutine parameters are identical to those listed above for the input file.

C.3 Slot Code Output Files

When performing a pattern, the following output files are written to disk:

`vmpattern.dat` = volumetric pattern data output file. Suitable for use by the **Volumetric Pattern Viewer** (see Chapter 6).

When performing a frequency sweep, the following output files are written to disk:

`fsgain.dat` = frequency sweep data of antenna gain. It is intended for plotting gain versus frequency (see Section C.5).

The following files are output for both frequency and pattern sweep analysis:

`view.dat` = file containing all slot parameters and output data produced by the analysis run. It is intended for viewing. Any errors produced during the analysis run will be printed to this file.

C.4 Running the Slot Code from X-Antenna

Another main routine, `slotx`, will read in the **X-Antenna** output file and call the `slot` routine directly with the appropriate parameters. In this case, the program assumes the name of the **X-Antenna** output file to be the default name for the output file, `xant.dat`. This simplifies the running of the program since no command line parameters are needed. The name of this routine can then be used as the name of the analysis code for the slot antenna in the **X-Antenna** command-file (See Chapter 5). A listing of the `slotx` program is provided in Appendix B.

C.5 Subroutines for Producing Plotting Data

The **Slot** code makes use of a subroutine, `vmpattern`, to output volumetric pattern data in a standardized format. This format is identical to the input file format used by the **Volumetric Pattern Viewer** described in Chapter 6. This means that the output data from the **Slot** code can be read in directly by the **Volumetric Pattern Viewer**. The `vmpattern` subroutine may incorporated into a code the user is developing in order to output volumetric pattern data for use by the **Volumetric Pattern Viewer**. A listing and description of this subroutine is provided in Appendix E.

The **Slot** code makes use of another subroutine, `fsgain`, which will produce an output file for a frequency sweep of the antenna gain. A description and listing of this subroutines is provided in Appendix F.

Appendix D

The Microstrip Code

D.1 Introduction

The **Microstrip** code analyzes microstrip antennas and performs either pattern or frequency sweep analysis of the antenna gain [6]. Additionally, a frequency sweep will also compute the input impedance of the antenna. The pattern sweep is defined in terms of a volumetric pattern (partial or full) and the frequency. The pattern, if properly specified, can be equivalent to an azimuth or elevation cut. The frequency sweep is defined in terms of the frequency range and the field angle (θ,ϕ). Both the pattern and frequency sweep analysis are assumed to be in the far-field. The code is in FORTRAN.

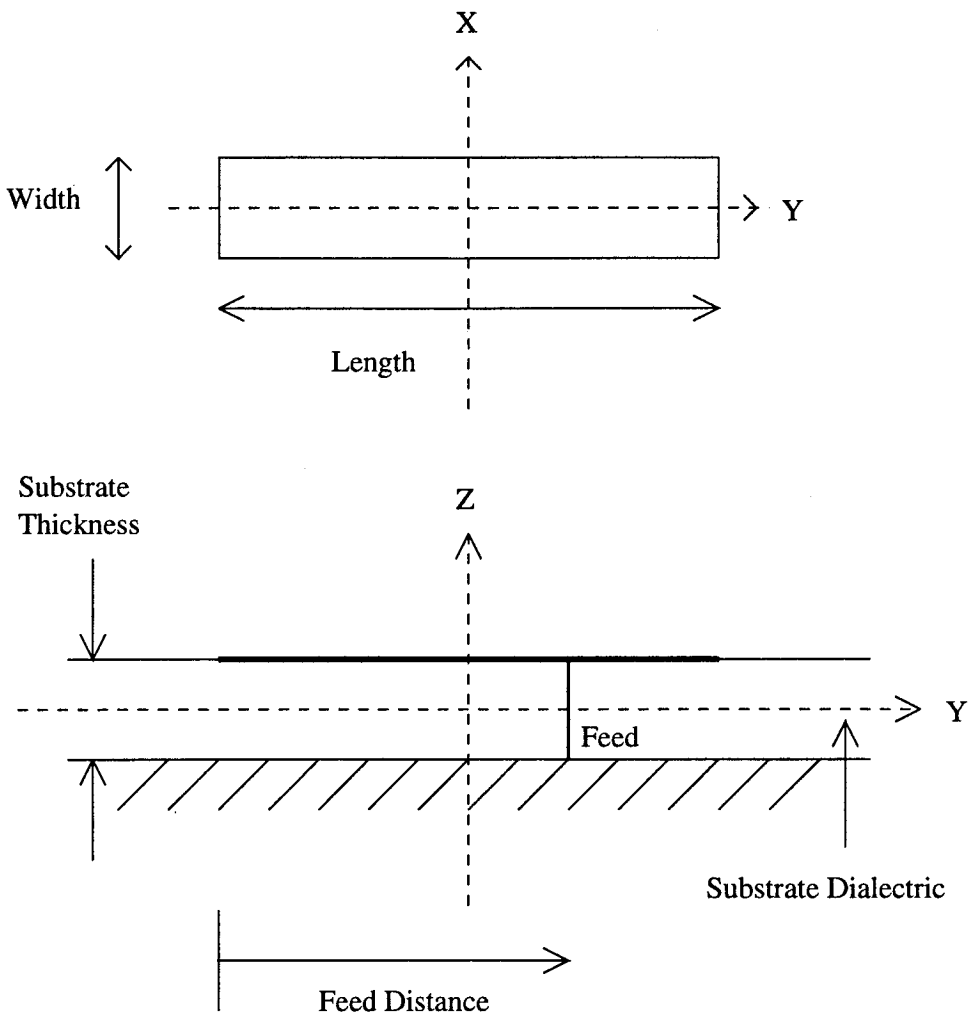
D.2 Running the Microstrip code

The **Microstrip** code may be run from the main program, `mstripf`, which reads in a data file containing the antenna data and both pattern angle and frequency sweep information. This routine then calls the main subroutine `mstrip` which performs the computations. This subroutine may be included in another program and called directly with the appropriate parameters. Figure D.1 shows the geometry of the microstrip antenna.

Form of the data file read in by `mstripf`:

width length feed T eps_rel tand

isweep



Note: Feed distance may be measured from either end.

Figure D.1: Geometry for the microstrip antenna.

th1 th2 dth

ph1 ph2 dph

ps_freq

fr1 fr2 dfr

fs_theta fs_phi

width, length = width and length (in meters) of the microstrip.

feed = distance (in meters) from one end of the microstrip to the location of the feed. This number may be greater or less than than half the length of the microstrip.

eps_rel = relative dielectric constant of the substrate.

tand = loss tangent of the substrate.

isweep = indicator for pattern or frequency sweep analysis. (1 = pattern, 2 = frequency).

th1, th2 = starting and ending angles in degrees, respectively, for the pattern in the θ-direction.

dth = increment angle (in degrees) for the pattern in the θ-direction. If dth does not divide evenly into (th2 - th1), but instead divides N times with a remainder greater than 10^{-3}, dth will be modified to divide $N + 1$ times into (th2 - th1). If dth is not an integer, there may be small roundoff error.

ph1, ph2 = starting and ending angles in degrees, respectively, for the pattern in the ϕ-direction.

dph = increment angle (in degrees) for the pattern in the ϕ-direction. The value of dph will be altered in an identical manner as dth.

ps_freq = frequency in MHz for the pattern.

fr1, fr2 = starting and ending frequencies (in MHz),respectively, of the frequency sweep.

dfr = increment frequency (in MHz) of frequency sweep. The value of dfr will be altered in an identical manner as dth and dph.

fs_theta, fs_phi = look angles (in degrees) for the frequency sweep in the θ and ϕ-directions respectively.

Form of the subroutine call to `mstrip`:

```
 call mstrip(
&   width,length,T,feed,eps_rel,tand,
&   isweep,
&   ps_freq,th1,th2,dth,ph1,ph2,dph,
&   fr1,fr2,dfr,fs_theta,fs_phi)
```

The subroutine parameters are identical to those listed above for the input file.

D.3 Microstrip Code Output Files

When performing a pattern, the following output files are written to disk:

`vmpattern.dat` = volumetric pattern data output file. Suitable for use by the **Volumetric Pattern Viewer** (See Chapter 6).

When performing a frequency sweep, the following output files are written to disk:

`fsgain.dat` = frequency sweep data of antenna gain. It is intended for plotting gain versus frequency (see Section D.5).

`fsimp.dat` = frequency sweep data of antenna input impedance. It is intended for plotting input impedance versus frequency (see Section D.5).

The following files output for both frequency and pattern analysis:

`view.dat` = file containing all microstrip parameters and output data produced by the analysis run. Intended for viewing. Any errors produced during the analysis run will be printed to this file.

D.4 Running the Microstrip Code from X-Antenna

Another routine, `mstripx`, will read in the **X-Antenna** output file and call the `mstrip` routine with the appropriate parameters. In this case, the program assumes the name of the **X-Antenna** output file to be the default name for the output file, `xant.dat`. This simplifies the running of the program since no command line parameters are needed. The name of this routine can then be used as the name of the analysis code for the microstrip antenna in the **X-Antenna** command-file (See Chapter 5). A listing of the `mstripx` program is provided in Appendix B.

D.5 Subroutines for Producing Plotting Data

The **Microstrip** code makes use of a subroutine, `vmpattern`, to output volumetric pattern data in a standardized format. This format is identical to the input file format used by the **Volumetric Pattern Viewer** described in Chapter 6. This means that the output data from the **Microstrip** code can be read in directly by the **Volumetric Pattern Viewer**. The `vmpattern` subroutine may incorporated into a code the user is developing in order to output volumetric pattern data for use by the **Volumetric Pattern Viewer**. A listing and description of this subroutine is provided in Appendix E.

The **Microstrip** code also makes use of two other subroutines, `fsimp` and `fsgain` which produce frequency sweep output files for the antenna input impedance and gain respectively. A description and listing of these subroutines is provided in Appendix F.

Appendix E

Subroutine to Output Standardized Volumetric Pattern Files

E.1 Subroutine Listing

```
      subroutine vmpattern(gain_th,gain_ph,phase_th,phase_ph,
     &   npts_th,th1,th2,dth,npts_ph,ph1,ph2,dph,
     &   max,plot_file,freq)

      implicit none

      real     gain_th(max,max),gain_ph(max,max)
      real     phase_th(max,max),phase_ph(max,max)
      integer  npts_th,npts_ph
      real     th1,th2,dth
      real     ph1,ph2,dph
      integer  max
      integer  plot_file
      real     freq

      integer i,j
      real    theta,phi
      real    fMHz

      fMHz = freq/1e6

      write(plot_File,*)1,fMHz,-1.0
      write(plot_File,*)npts_th,th1,th2,dth
      write(plot_File,*)npts_ph,ph1,ph2,dph

c     Output as azimuth patterns. ehn 11/26/94
```

```
         do j=1,npts_th
            theta = th1 + (j-1)*dth

      do i=1,npts_ph
         phi = ph1 + (i-1)*dph

            write(plot_File,300)theta,phi,gain_th(i,j),phase_th(i,j),
     &         gain_ph(i,j),phase_ph(i,j)

 300        format(2(f6.2,2x),2(2(f8.2,2x),2x))

         end do
      end do


      end



      subroutine fsgain(gain_th,gain_ph,phase_th,phase_ph,
     &   npts_fr,fr1,fr2,dfr,plot_file,
     &   look_th,look_ph)

      implicit none

      real     gain_th(*),gain_ph(*)
      real     phase_th(*),phase_ph(*)
      integer  npts_fr
      real     fr1,fr2,dfr
      integer  plot_file
      real     look_th,look_ph

      integer  i
      real     freq
      real     radial

c
c     Note: radial used to indicate radial distance (radial=-1.0 indicates
c           far-zone pattern).  Included for consistency with other data
c           files that have similar parameters for indication of far-zone
c           patterns.
c
      radial=-1.0

      write(plot_file,*)npts_fr,fr1,fr2,dfr
      write(plot_file,*)look_th,look_ph,radial
      write(plot_file,*)
```

```
      do i=1,npts_fr
         freq = fr1 + (i-1)*dfr

         write(plot_file,100)freq,gain_th(i),phase_th(i),
     &      gain_ph(i),phase_ph(i)

 100     format(f8.2,2x,2(2(f8.2,2x),2x))

      end do

      end



      subroutine fsimp(imp,npts_fr,fr1,fr2,dfr,
     &   plot_file)

      implicit none

      complex  imp(*)
      integer  npts_fr
      real     fr1,fr2
      real     dfr
      integer  plot_file

      integer  i
      real     freq

      write(plot_file,*)npts_fr,fr1,fr2,dfr
      write(plot_file,*)

         do i=1,npts_fr
            freq = fr1 + (i-1)*dfr

            write(plot_file,100)freq,imp(i)
 100        format(f8.2,5x,e10.3,2x,e10.3)

         end do

         end
```

gain_th, gain_ph = two dimensional array containing the gain of the θ and ϕ-polarized fields respectively. The data points are organized in groups of azimuth-cuts. See below for explanation.

phase_th, phase_ph = two dimensional array containing the phase of the θ and ϕ-polarized fields respectively. The data points are organized in groups of azimuth-cuts. See below for explanation.

npts_th = number of data points in the θ-direction. This is the number of data points in each azimuth-cut.

npts_ph = number of data points in the ϕ-direction. This is the number of azimuth-cuts making up the volumetric pattern.

th1, th2 = starting and ending angles of the pattern in the θ-direction.

dth = increment angle in the θ-direction.

ph1, ph2 = starting and ending angles of the pattern in the ϕ-direction.

dph = increment angle in the ϕ-direction.

max = dimension of arrays *gain_th, gain_ph, phase_th, phase_ph.*

plot_file = name of the output file to be written.

freq = frequency of the pattern sweep.

Note that there are no units for the various parameters. This is because this routine simply outputs data. It is the user's responsibility to ensure the data has the appropriate units.

E.2 Organization of Data Points for a Volumetric Pattern

An azimuth-cut is a pattern in which θ is fixed and ϕ varies. An elevation-cut is a pattern in which ϕ is fixed and θ varies. This subroutine organizes data points in the two-dimensional arrays *gain_th, gain_ph, phase_th, phase_ph* into groups of azimuth-cuts beginning with the azimuth cut at θ = th1 when $j = 1$, and ending with the azimuth-cut at θ = th2 when j = npts_th. Each azimuth cut must contain the same

number of data points where ϕ varies from $\phi =$ ph1 when $i = 1$ to $\phi =$ ph2 when $i =$ npts_ph.

Appendix F

Subroutine to Output Standardized Frequency Sweep Files

F.1 Frequency Sweep of Gain

```
      subroutine fsgain(gain_th,gain_ph,phase_th,phase_ph,
     &   npts_fr,fr1,fr2,dfr,plot_file,
     &   look_th,look_ph)

      implicit none

      real     gain_th(*),gain_ph(*)
      real     phase_th(*),phase_ph(*)
      integer  npts_fr
      real     fr1,fr2,dfr
      integer  max
      integer  plot_file
      real     look_th,look_ph

      integer  i
      real     freq
      real     radial

c
c     Note: radial used to indicate radial distance (radial=-1.0 indicates
c           far-zone pattern).  Included for consistency with other data
c           files that have similar parameters for indication of far-zone
c           patterns.
c
      radial=-1.0
```

```
      write(plot_file,*)npts_fr,fr1,fr2,dfr
      write(plot_file,*)look_th,look_ph,radial
      write(plot_file,*)

      do i=1,npts_fr
         freq = fr1 + (i-1)*dfr

         write(plot_file,100)freq,gain_th(i),phase_th(i),
     &       gain_ph(i),phase_ph(i)

 100     format(f8.2,2x,2(2(f8.2,2x),2x))

      end do
```

The parameters for this subroutine are:

gain_th, gain_ph = one dimensional array containing the gain of the θ and ϕ-polarized fields, respectively, for all frequency values in order from smallest frequency to largest.

phase_th, phase_ph = one dimensional array containing the phase of the θ and ϕ-polarized fields, respectively, for all frequency values in order from smallest frequency to largest.

npts_fr = number of frequency data points.

fr1, fr2 = starting and ending frequency values.

dfr = increment frequency value such that (fr2 − fr1) = npts_fr×dfr.

plot_file = name of output file to be written.

look_th, look_ph = look angles in the θ and ϕ-directions respectively.

The output file has the following format:

numpts_fr fr1 fr2 dfr

look_th look_ph rad

freq(1) gain_th phase_th gain_ph phase_ph

freq(2) gain_th phase_th gain_ph phase_ph

.....

.....

.....

freq(N) gain_th phase_th gain_ph phase_ph

npts_fr = Number of points in the frequency sweep.

fr1, fr2 = start and end frequencies in MHz.

dfr = increment frequency.

look_th, look_ph = look angles in the θ and ϕ-directions, respectively.

rad = radius of pattern.

freq(i) = frequency of ith data point ($i = 1, \ldots, N$) where N = npts_fr.

gain_theta, gain_phi = gain of θ and ϕ-polarized fields, respectively, for corresponding frequency.

phase_theta,phase_phi = gain of θ and ϕ-polarized fields, respectively, for corresponding frequency.

F.2 Frequency Sweep of Input Impedance

```
subroutine fsimp(imp,npts_fr,fr1,fr2,dfr,plot_file)

implicit none

complex  imp(*)
integer  npts_fr
real     fr1,fr2
real     dfr
integer  max
```

```
      integer  plot_file


      integer  i
      real     freq

      write(plot_file,*)npts_fr,fr1,fr2,dfr
      write(plot_file,*)

         do i=1,npts_fr
            freq = fr1 + (i-1)*dfr

            write(plot_file,100)freq,imp(i)
100         format(f8.2,5x,e10.3,2x,e10.3)

         end do

         end
```

imp = one dimensional array containing the complex input impedance for all frequency values in order from smallest frequency to the largest.

npts_fr = number of frequency data points.

fr1, fr2 = starting and ending frequency values.

dfr = increment frequency value such that (fr2 − fr1) = npts_fr×dfr.

plot_file = name of output file to be written.

The output file has the following format.

numpts_fr fr1 fr2 dfr

freq(1) imp_re imp_imag

freq(2) imp_re imp_imag

.....

.....

.....

freq(N) imp_re imp_imag

npts_fr = Number of points in the frequency sweep.

fr1, fr2 = start and end frequencies in MHz.

dfr = increment frequency.

freq(i) = frequency of ith data point ($i = 1, \ldots, N$) where N = npts_fr.

imp_re, imp_imag = real and imaginary components, respectively, of the complex input impedance for the corresponding frequency.

Bibliography

[1] Edward H. Newman, "A User's Manual for the Material Wires Code", The Ohio State University, ElectroScience Lab Report 722767–1, Nov. 1990.

[2] Adrian Nye and Tim O'Reilly, *X Toolkit Intrinsics Programming Manual*, O'Reilly & Associates, Inc., California, 1990.

[3] Dan Heller, *Motif Programming Manual*, O'Reilly & Associates, Inc., California, 1990.

[4] X Windows System, Version 11. A network transparent window system developed at MIT which runs on a wide range of computing and graphics machines.

[5] *xfig - Facility for Interactive Generation of figures under X11*, Original author: Supoj Sutanthavibul, University of Texas at Austin.

[6] K.R. Carver and J.W. Mink, "Microstrip Antenna Technology," IEEE Trans. on Antennas and Prop., Vol. AP-29, pp 2-24, Jan. 1981.

[7] Richard C. Johnson and Henry Jasik, *Antenna Engineering Handbook*, McGraw-Hill, New York, 1961.

[8] E.W. Braeden, F.T. Klevenow, E.H. Newman, R.G. Rojas, K.S. Sampath, J.T. Scheik, and H.T. Shamansky, "Operation of the Helicopter Antenna Radiation Prediction Code," Ohio State University ElectroScience Lab. Report 722792-4, prepared under Grant No. NAG-1-1058 with the National Aeronautics and Space Administration, Langley Research Center, June 1993.