United States Department of Commerce
National Institute of Standards and Technology

NISTIR 3970

PERSONAL COMPUTER CODES FOR ANALYSIS OF PLANAR NEAR FIELDS

Lorant A. Muth
Richard L. Lewis

NISTIR 3970

PERSONAL COMPUTER CODES FOR ANALYSIS OF PLANAR NEAR FIELDS

Lorant A. Muth
Richard L. Lewis

Electromagnetic Fields Division
Electronics and Electrical Engineering Laboratory
National Institute of Standards and Technology
Boulder, Colorado 80303-3328

June 1991

Sponsored, in part, by
Air Force Guidance and Metrology Center
Newark Air Force Base, Ohio 43057

U.S. DEPARTMENT OF COMMERCE, Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, John W. Lyons, Director

CONTENTS

Personal Computer Codes for Analysis of Planar Near Fields

Lorant A. Muth and Richard L. Lewis

Electromagnetic Fields Division
National Institute of Standards and Technology
U. S. Department of Commerce
Boulder, Colorado

We have developed Fortran codes for analysis of planar near-field data. We describe some of the inner workings of the codes, the data management schemes, and the structure of the input/output sections to enable scientists and programmers to use these codes effectively as a research tool in antenna metrology. The open structure of the codes allows a user to incorporate into the package new applications for future use with relative ease. The subroutines currently in existence are briefly described, and a table showing the interdependence among these subroutines is constructed. Some basic research problems, such as transformation of a near field to the far field and correction of probe position errors, are carried out from start to finish to illustrate use and effectiveness of these codes. Sample outputs are shown. The advantage of a high degree of modularization is demonstrated by the use of DOS batch files to execute Fortran modules in a desired sequence.

Key words: antenna metrology; computer codes; data management; planar near fields; far fields; research tool; subroutines

1. Introduction

Most research problems in antenna metrology are computationally intensive, and program development makes up a substantial part of the research effort. Hence, isolating frequent computational themes in this research area and developing *independent modules* that can perform any of these computational themes in any order independently of any previous computational step are very desirable. Improvements in both the quality and quantity of research can be a by-product of such a computational tool. Ideally, such a tool should be an open-ended system; that is, new modules can be added painlessly to increase the versatility of the package. It should also be easy to use and learn, and therefore adaptable to new areas of research. With cooperative effort such a software package could evolve into a comprehensive research tool over a short period of time.

With these thoughts in mind, we have taken the first steps to accomplish the goal of creating a comprehensive software package suitable for conducting state-of-the-art research on a personal computer. We have achieved a very high level of modularity by creating a large number of Fortran subroutines that can be used in many different contexts, because the subroutines emphasize structure rather than content of small computational problems. By the same effort, we have made it relatively simple to create higher-level subroutines, because such routines can rely heavily on the existing low-level subroutines of general applicability. These higher-level routines accomplish more complicated and complete computational tasks than the low-level subroutines. In turn, they can be combined to form independent modules, which are the selected subtasks of a particular research effort. These subtasks will be usually subtasks in other research areas, too. Hence, the effort expended in creating them will be saved many times over in future endeavors.

Particular attention has been given to the way information flows to and from the modules and between modules. We have automated much of the data management needed to provide a smooth transition as one module finishes its task and another is executed to accomplish the next step of the research. Many small modules, playing a supportive role in data management, have been created to allow manipulation of datasets according to the needs of the current phase of the research project. For example, an existing dataset merely has to be *activated* to make it accessible to a module about to be executed. Thus, both the modules chosen to be executed and the datasets to be used can be controlled interactively by the scientist. This makes for a very flexible computational procedure, freeing one's time and energy to think about research procedure rather than computational detail.

Because most research problems in antenna metrology are computationally intensive and usually require large amounts of memory, we recommend as a minimum that a personal computer equipped with the fastest available CPU and floating-point processor be used, and that a minimum of 4 megabytes of RAM be made available.

In the next section, concentrating on the main features, we outline the structure of the computational package named Planar Near-Field Codes (PNFC); in the subsequent sections we present essential details of the main features. It is our intention that researchers, programmers, or scientists be able to use these codes effectively after familiarizing themselves with the contents of this report.

<u>Revisions</u>

This report is a revision of a previous publication on the same software package [1]. This revision was written to improve the exposition in some sections, to update the tables and appendices to include new modules and subroutines not in the previous publication, and to add a new appendix showing the subroutine dependencies of the research modules.

2. General Features

The complete PNFC is structured into *modules.* To be able to determine the function of a module we merely have to decipher the acronym that was constructed to name the module. Once the acronym is deciphered, the full function of the module should be self-evident. In Table 1 we have compiled the symbols used to construct module names along with the definition of each symbol. In Tables 2 and 3 we list the modules used to conduct research, and those used to manage data access during the course of research, respectively. A brief description of each module's function is also included.

All research modules listed in Table 2 manipulate some existing dataset; that is, they either numerically transform the dataset, or perform some I/O operation on it. Datasets created subsequent to the original dataset are stored as binary files and are given filenames *fort.xx*, where the file extension *xx* is a *unit number* that is automatically assigned to a specific dataset. The only information a research module needs in order to access an existing dataset is the unit number assigned to that dataset. Each module was designed to perform a single computational task that is an important aspect of research in antenna metrology. Some of the modules are more specific to antenna metrology than others. For example, the module URDNFFF (Utility, ReaD a Near Field and transform it to the Far Field) is an ever-present computational step specific to this research area, but UPRNCBD (Utility, PRiNt a Complex Binary Dataset) is obviously of more general applicability. How to execute these modules is demonstrated in Section 4.

The modules listed in Table 3 perform simple data management. For example, USWTOFF (Utility, SWitch TO Far Fields) activates far-field datasets that have been previously created and recorded within the data management part of the system. After USWTOFF has been executed any subsequent executions of modules that can use either far-field or near-field data will access the far-field datasets, unless this switch is overridden by a nonzero *active* dataset switch. How to *activate* a specific dataset to make it the dataset that any module will use will be covered in Section 3.

The research modules are constructed from a large set of independent subroutines that perform specific computational or I/O subtasks. They are used repeatedly in various sequences to produce the specific results of the module. These subroutines are compiled into a library, which is linked to a module at compilation time. All existing subroutines are listed in Table 7, along with a brief description of their function.

All research modules access file DABD.IOF, which contains the filename of the research project's parameter file. This file gives the relevant input parameters for the research project and the filename of the original dataset. The original datasets are recorded as direct-access binary files, so that specific records within them can be accessed or modified at will. How to create the original direct-access datasets from some ASCII file that was created on some other computer or data acquisition system is explained in Appendix A. The first seven records in these datasets contain the essential parameters of the dataset. All modules access the original direct-access dataset to input these essential parameters, although only a subset of these might actually be needed by the specific module in use. This procedure assures that the same parameter set will be used by all modules using a specific dataset. A list of the essential dataset parameters is given in Appendix A.

Each module might also access a parameter (.PAR) file that is specific to it. For example, UMAKEDZ (Utility, MAKE DZ), which creates a probe displacement error function, reads the parameter file PERDZ.PAR if periodic error functions are requested, and UTSZ (Utility, Taylor Series in Z) reads the parameter file SCALE.DZ to input the amplitude of the error function requested for the current execution. The parameter files currently in existence and the research modules that access them are listed in Table 4. The parameter files and data management files accessed by the data management modules are tabulated in Table 5.

All necessary I/O procedures are handled within each module, but some specific modules prepare the data and create ASCII files that can be further processed for graphical output. Two such modules are UCBDGRD (Utility, Complex Binary Dataset to .GRD file) and UCBDDAT (Utility, Complex Binary Dataset to .DAT file), which create ASCII datasets to.be used for plotting 3-D and simple linear plots, respectively. These modules also rely on specific parameter files to perform their function as desired.

Finally, all modules have very similar structures and differ significantly only in their computational sections. The common structure is as follows:

a. Read all relevant switch settings and determine the unit numbers of existing datasets. Check to see whether any new unit numbers can be allocated and assign the new unit number.
b. Read all relevant parameters needed by the module.
c. Read all parameters describing the dataset to be used.
d. Read all datasets needed by the module.
e. Prepare for computations.
f. Perform the computations.
g. Output the results to the preassigned units.
h. Set the relevant switches and update the unit numbers of the new datasets.
i. Output a limited log file to record essential parameters and I/O activity.
j. Update the history file to show which modules were executed.
k. Stop execution of the module with *'Successful termination'* message.

This structure seems to be very successful, in that modules that are truly independent of each other have been constructed, which, therefore, can be executed in any order as long as the relevant datasets have been created. Under these conditions a research project can be implemented with relative ease, either interactively, or with the use of DOS batch files. (The use of DOS batch files to enhance research efficiency is discussed in Section 6.)

3. Data Management

In this section we present the details of unit or dataset management built into the system as a whole. Specific modules make use of this procedure according to their requirements. Here the terms *data management* and *unit management* have the same meaning, because datasets generated by the PNFC reside on files with filenames *fort.xx*, where *xx* is some integer refering to a Fortran *unit number* assigned internally by the module being executed. (The filename *fort* is automatically assigned when a Fortran binary-write is executed.)

a. Initialization of the system.

The system has to be initialized before starting any research project with a new dataset. Both the system parameters and the unit numbers where different datasets will reside are initialized in this procedure. Here we will describe how the unit numbers are set and manipulated at the start of the research project. In Appendix B the output of the initialization module is shown and an explanation of features not covered in this section is presented.

When the UINITUN (Utility, INITialize Unit Numbers) module is executed, the initial unit numbers for the far-field and the near-field datasets are read from a parameter file (INIT.IUN) and entered into the unit number files named FF.IUN and NF.IUN. After initialization the modules URDFFNF or URDNFFF can be executed to read in the existing direct-access complex binary dataset containing the original data to be analyzed. (Subsequently, the same modules will access datasets according to the unit management switch settings. See Section 3b below.) Both modules output both far-field and near-field datasets to *fort.xx* files; the filename extensions *xx* are obtained from the files FF.IUN and NF.IUN.

All far-fields datasets created after initialization will be assigned unit numbers one less than the previously assigned far-field unit number, and all near-field datasets created after initialization will be assigned unit numbers one higher than the previously assigned near-field unit number. Hence, the far-field and near-field unit numbers will converge toward each other as datasets are created by executing module after module. Before any module proceeds with execution of its task it checks to see whether there is enough of a difference between the last far-field and the last near-field unit numbers to allow the creation of additional datasets. If the far-field and near-field unit numbers are adjacent to each other, no module that creates a new dataset is allowed to proceed, and an appropriate error message to that effect is displayed. In this manner, disk overload is prevented, because new datasets cannot be created indefinitely.

b. The Complex Binary Dataset (CBD) files.

Except for the original datasets, which are stored as direct-access binary files, the modules read and write complex binary datasets (CBD) during execution to store intermediate results in the course of the research project. These datasets are recorded with the filename *fort* and with integer unit numbers for extensions. The unit numbers are automatically assigned, as described in the previous section. For example, *fort.40* would contain the initial near-field data, while *fort.60* would contain the initial far-field data. To maximize disk storage, all datasets are stored as unformatted binary files.

Because all modules read and/or write one or more CBD files, we must keep track of these files and must be able to access a desired dateset with relative ease. For this purpose a support system to manage unit numbers has been constructed. This works as follows:

An *existing* dataset is identified by its *unit number*, which is the extension of the *fort* file. An *existing* unit number is any unit number that has been created since initialization. An existing unit number, in general, has no special status and is not automatically accessed by any module until it is made *active*, *additional*, or *current*. A unit number is *active* if its value is recorded in the ACTIVE.IUN file, whereas a unit number is *additional* if its value is recorded in the ADD.IUN file. The *current* unit numbers are the last unit numbers recorded in the files FF.IUN and NF.IUN. In general, these are the unit numbers created by the most recently executed module, but can be altered according to the user's needs. A general purpose module will access either the *current near-field* unit number or the *current far-field* unit number, depending on the setting of the variable FFNF recorded in the file FFORNF.IUN. The variable FFNF can have the values 'ff' or 'nf'.

When modules access datasets a precedence rule is followed: the ACT*ive* file gets accessed first, and the ADD*itional* file gets accessed if the module requires two datasets. The *current* file gets accessed only if the ACT*ive* file is set to zero, and any *existing* file can be accessed only if it is made ACT*ive*, ADD*itional* or *current*. To access a desired *current* file with modules that process either far-field or near-field datasets the 'FFORNF' switch has to be set to tell the system whether far-field or near-field unit numbers are of interest.

Several utilities have been written to define these file types easily. These utilities are listed in Table 3. To view the existing unit numbers we execute USHOWUN (Utility, SHOW Unit Numbers), which summarizes the existing files according to their type (as defined in FFORNF.IUN) and status (ACT, ADD, *current*, existing). USHOWUN will also identify the unit numbers of special datasets, such as the TS (Taylor Series) file, EC (error corrected) and DS (direct sum) files. To *activate* a dataset, execute one of the special utilities listed in Table 3. Similarly, we can *add* a dataset. To make a dataset *current*, we can execute the decrementing or incrementing modules (UDECFF, UDECNF, UINCFF, UINCNF) repeatedly until the desired unit number is the last unit number shown by USHOWUN. Two examples of the output of USHOWUN are given in Appendix C with explanations.

c. Output files.

Most modules read and write CBD files according to the unit management scheme built into every module. In addition, some of the modules create special ASCII files to be used as input to graphics programs. The module UCBDGRD, for example, reads the ACT*ive* or *current* CBD file, with filename *fort* and an extension defined by the *active* or *current* unit number. It then outputs ASCII files, whose filenames are obtained by concatenating the setting of the switch FFORNF with the descriptors AMP or PHASE, and appending a filename extension .GRD. The structure of these files is determined by the requirement of the graphics package in use. Similary, the module UCBDDAT creates ASCII files for simple xy-plots, with filenames obtained the same way as for .GRD files, but using .DAT as the filename extension. This module outputs a set of x-values and one, two, or three y-values. The actual number of data columns output by UCBDDAT is determined by the ACT*ive*, ADD*itional* and *current* switch settings. The rules are as follows: to write only a single column of y-values, the *active* file must be nonzero and the *additional* file must be zero. To write two sets of y-values, the *additional* file must also be nonzero. To write three sets of y-values, both the *active* and *additional* unit numbers must be zero, in which case the *current* unit number will be used to create the first column, and the next two adjacent *existing* unit numbers will be used to create columns two and three in the .DAT file. A simple module UACTADD0 (Utility, set ACTive and ADDitional to zero) will reinitialize the unit numbers so that up to three columns of data might be written.

All research modules create output files that contain information about the execution flow of the module. These files have filenames identical to the module names and .OUT file extension. Parameters used and the unit numbers accessed or created are listed in these files, so that an orderly cross-referencing can be conducted if some of the results are brought into question. In addition, these modules record their activity in a history file (.HST) so that the sequence of executions can be checked at a later time.

4. Research Modules

In Table 2 we list the currently existing modules. These modules were designed in the course of a research project where the goal was to understand the propagation of errors in near-field data to the far field data, and to develop techniques to remove the effects of these errors from the far-field data. Thus, some of these modules are very specific to this research projects; others, however, have more general applicability.

To illustrate the use of these modules in research, we provide first a simple, then a more elaborate, example of a computational sequence that delivers results required by two representative research problems.

A simple research problem.

Given a near-field dataset, obtain perspective plots of the near field and of the computed far field.

Using 'x' to mean 'execute' a module, this simple task would be accomplished by entering the following batch commands at the DOS prompt:

```
x uinitun
x urdnfff
x ucbdgrd
plt ff
x uswtonf
x ucbdgrd
plt nf
```

Here *plt* is a DOS batch file that calls on the system plot package to process the graphical data files output by UCBDGRD. The details of this procedure would vary from system to system, depending on the graphics package used.

From Table 2 we can easily ascertain that the above sequence of computational steps will deliver the results required. First, by executing UINITUN we initialize the system variables and unit numbers. As a result, all previous settings will be lost. Next, we read in the original near-field dataset and transform it to the far field. At this point, the data management system sets the *ffornf* variable to *ff*, because the last field created was a far field. Then, UCBDGRD will access the far-field dataset to create a perspective plot file. To create a plot file using the *current* near-field dataset, we must set the system variable *ffornf* to *nf*. Hence, we execute USWTONF, and then UCBDGRD will access the near-field dataset to create a perspective plot file for the near field.

A more complicated research problem.

Given a near-field dataset and a known probe-position error function, use the Taylor series expansion to generate error-contaminated near-field values. Then, remove these errors from the data using a well defined error-correction technique, and compare the error-free, error-contaminated and error-corrected near and far fields by looking at the respective complex ratios of field values at each data point. Present the results in perspective plots and/or linear plots, showing amplitude ratios and phase differences.

Using the existing set of research modules, this relatively involved research task can be brought to conclusion as follows:

```
x uinitun
x umakedz
x urdnfff
x uswtonf
x utsz
x uecz4
```

Executing this sequence, we have accomplished the first part of the research. Again, we started by initializing the system parameters and unit numbers. Then, a probe-displacement error field is created by executing UMAKEDZ, which reads relevant parameter files as shown in Table 4 to obtain the desired error function's specifications. This routine also creates a .GRD file for obtaining a perspective plot of the error function. Next, the original near-field dataset is read in and the corresponding far-field dataset is calculated. We execute USWTONF so that the *current* near field will be read by module UTSZ. Then errors are introduced into the original near-field dataset by executing module UTSZ, which carries out a Taylor series expansion with respect to the Z coordinates. The errors that have been introduced are then removed by executing UECZ4, which removes probe-position errors in the Z coordinate up to the fourth order. A discussion of this error-correction technique is given elsewhere by the authors [2].

At this point each dataset has been recorded on the disk in *complex binary data* files with filenames *fort* and file extensions *.xx*, where *xx* is some unit number automatically assigned by the data management section of the system. We can now obtain the far field corresponding to each near field that has been created. We proceed as follows:

x udecnf
x urdnfff
x uincnf
x urdnfff

All far-field datasets of interest have now been created. By executing UDECNF, the *current* near field unit number has been decremented by one (assuming that the unit increment/decrement parameter is one, the default), thereby making the near field obtained prior to the last near field *current*. Then executing URDNFFF transforms this near-field dataset into a far-field dataset, which is stored as a *fort.xx* file with the next available filename extension *xx* having been obtained from FF.IUN. Next, UINCNF increments the near-field unit number to increase the *current* unit number by one, which, in this case, is the last near field created. Again executing URDNFFF creates the corresponding far-field dataset. This procedure has relied on using the *current* near-field unit number to specify which near-field dataset is to be read in and transformed into a far-field dataset. An entirely equivalent procedure, which would make use of *active* unit numbers to accomplish the same task, proceeds as follows:

x uacttsz
x urdnfff
x uactecz
x urdnfff

Only plotting and comparing the various near fields and far fields is left. The module UDIVCBD can be used to form the complex ratio of two near-field or far-field datasets. As discussed in the data-management section, the desired datasets may be loaded by defining an *active* and an *additional* unit number, or if these are set to zero, then the two most recently created fields (near or far) will be used, depending on the setting recorded in file FFORNF.IUN. Thus, to take the ratio of the error-contaminated near field to the original near field, we execute the following:

x uswtonf
x uacttsz
x uaddnf0
x udivcbd

Similarly, to take the ratio of the error-corrected near field and of the original near field we execute the following:

x uactecz
x uaddnf0

x udivcbd

In both of the above sequences of operations complex ratio fields are created, which are recorded sequentially using near-field unit numbers, after USWTONF was executed at the beginning of the sequence. The second execution of UADDNF0 is really redundant, because the first execution of this module is still in effect.

To create far-field ratios the procedure is somewhat different, since far fields have not been labeled by special identifiers, such as *ts* and *ec*. Any far field can be made *current* by incrementing or decrementing the far-field unit numbers an appropriate number of times, and can be selected by executing one of the modules UACTFF or UADDFF. Thus, to form all ratios we execute the following sequence:

x uswtoff
x uaddff0
x uincff
x uactff
x udecff
x udivcbd
x uincff
x uactff
x udecff
x udivcbd

All far-field ratios of interest have now been created and recorded on far-field unit numbers. This was accomplished by first switching to the far fields (USWTOFF), then making the original far field the *additional* field (UADDFF0), followed by making the far field created before the last one the *active* field (UINCFF, UACTFF and UDECFF) and taking the ratio (UDIVCBD). After the ratio was taken the *current* far-field unit number was automatically decreased. Next, the previously created far field was made *current* (UINCFF) and *active* (UACTFF), the *current* unit number reincremented (UDECFF) and then the ratio (UDIVCBD) was taken. Each ratio field was automatically assigned the next available far-field unit number.

At this point we can obtain a system status report, so that any problem with the sequence of operations could be detected. For this purpose we execute the module USHOWUN, whose output is presented in the second table in Appendix C, with a detailed discussion.

After examining the output of USHOWUN and ascertaining that no errors were made, we can plot any of the existing fields (*fort.xx* files). First, an ASCII plot file (.GRD) needs to be created using the module UCBDGRD, after which plots can be created using the plot package. The module UCBDGRD will read the *current* far- or near-field dataset, depending on the setting of the switch *ffornf*. This setting can be selected by executing USWTOFF or USWTONF. The chosen *current* file will then be accessed unless the *active* file is nonzero. A desired unit number can be made *active* by executing one of the modules that have the phrase ACT in their name followed by the appropriate .IUN filename designator.

Sample plotting procedures would be as follows:

```
x uact0
x uswtonf
x unorm1
x ucbdgrd
plt nf
```

and

```
x unorm0
x uacttsz
x ucbdgrd
plt nf
```

and

```
x uact0
x uswtoff
x unorm1
x ucbdgrd
plt ff
```

In all three examples we first specify the type of fields we want to access. Thus, in the first example, we first set the *active* file to zero and then execute USWTONF so that the *current* near field is accessed. Then, UNORM1 sets the normalization constant to one, since we wish to plot a ratio field, which should not be renormalized when it is converted to decibels. Next, the plot file is created by UCBDGRD. In the second example, the normalization constants are restored to their proper values (UNORM0), the error-contaminated near field that was created using the Taylor series is *activated* (UACTTSZ), and then UCBDGRD creates a plot file of the error-contaminated near field. In the third example, we again plot a ratio field since the *current* far field is accessed. All three cases use the DOS batch command *plt* to plot either the far field (*ff*) or the near field (*nf*).

5. Output Files.

All research modules have been constructed to write an output file where the parameters and data files used during execution are clearly listed. This way the settings of input/output parameters can be cross-referenced, and the correctness of the computational sequence and numerical inputs can be ascertained. These output files have the name of the modules as their filenames and .OUT for the file extension.

Certain modules write ASCII datasets to be used by the graphics package on the system. The module UCBDGRD creates two-dimensional ASCII datasets for perspective and contour plots, and the module UCBDDAT creates ASCII datasets (.DAT) for simple *xy*-plots. The module URMSCBD creates a .DAT file to plot the rms distribution of the power radiated in a far field. These .GRD and .DAT ASCII files may also be used to examine the data for any features we might be interested in.

The module UPRNCBD creates an ASCII file that contains a printout of the absolute amplitude and the phase of the rows and/or columns of any far- or near-field CBD (Complex Binary Data) file, which is chosen according to the switch setting of *ffornf* and the settings of the *current* and the *active* unit numbers. Thus, if the *active* unit number is zero, then the *current* file will be printed. The particular rows and/or columns to be printed over a specific data range are specified in the parameter file SUB.PRN. The module UPRDBCBD converts all the amplitudes to dB before creating a similar table.

6. DOS Batch Files

DOS batch files can be used to advantage to save time and effort when performing step-by-step computations to obtain a result. We can write batch files merely as abbreviations of longer commands, or to collect a set of executable steps that will be used many times over. The complexity of the batch files and their usefulness are limited only by the programmer's knowledge of the DOS operating system and the programmer's imagination.

The use of the *plt.bat* file has been illustrated in the previous section a number of times. Another example of a batch file is the abbreviation of the execution of the first simple research problem discussed above. Thus, the batch file *pltnfff* would look like this:

```
call x uinitun
call x urdnfff
call x ucbdgrd
call x uswtonf
call x ucbdgrd
call plt nf
call plt ff
```

Simply typing *pltnfff* at the DOS prompt would execute all the steps in this batch file. We now have a very easily usable, high-level program that will produce plots of the current near- and far-field datasets. The DOS expression *call* is used here to continue execution within the batch file to the last line. Without *call* execution would not return to the next step, but exit to the DOS prompt.

The second research problem is the implementation of an error-correction technique after an error-contaminated near-field dataset has been created using the Taylor series expansion with a predefined probe-position error function. What might change from one implementation to the next is the original dataset to be used, and the form and magnitude of the error function. These are all inputs to the complete procedure; that is, the program execution steps are the same, independent of these parameters. Therefore, a DOS batch file is appropriate for recording the steps of this relatively complicated research project. This batch file could be appropriately called *errcor.bat* (error correction), and would look like this:

```
call x uinitun
call x umakedz
call x urdnfff
call x uswtonf
call x utsz
call x uecz4
call x uacttsz
call x urdnfff
call x uactecz
call x urdnfff
call x uswtonf
call x uacttsz
call x uaddnf0
call x udivcbd
call x uactecz
call x udivcbd
call x uswtoff
call x uaddff0
call x uincff
call x uactff
call x udecff
call x udivcbd
call x uincff
call x uactff
call x udecff
call x udivcbd
```

This batch file goes as far as creating all the required near and far fields of the research project, as well as the ratio fields. It stops short of plotting any of the existing fields. A separate batch file would be appropriate for creating a desired set of plots.

Batch files using executable modules of the PNFC allows us to create and save complicated research procedures in a straightforward and efficient manner.

A collection of such batch files can greatly enhance the computational scope and efficiency of any research project.

7. Symbol Definitions

Table 6 lists descriptors used in naming the subroutines of the PNFC. This table should make reading the source codes easier. We hope that authors of new code will use existing symbols as far as possible to contribute to the coherence of the full package.

8. Subroutine Descriptors

Table 7 lists the available subroutines along with brief descriptions of their functions. This can be helpful when creating new modules or when planning to write new subroutines to perform computational tasks not yet addressed in the package.

9. Table of Dependencies

Appendix D is a table of dependencies for the research modules, listing in the order called the first occurence of each distinct subroutine call for each module listed in table 2. Similarly, Appendix E is a table of dependencies for the subroutines, showing the interrelationships between the various subroutines. This also serves as an index of subroutines, because all existing subroutines are included alphabetically in the leftmost column. In each case, the subroutines called by the routines on the left are listed in the order in which they are called. In this manner we can get an overview of both the contents and structure of the complete code. These files can be used to advantage when developing new code, or when improving the existing code is contemplated.

10. Conclusion

In this report we have outlined the computational structrure of a newly created software package named Planar Near-Field Codes (PNFC) for personal computers. This package supports the computational effort needed to solve research problems in antenna metrology.

The PNFC package can be used to address diverse research problems because of its highly modular structure. The modules have been constructed to provide the computational procedure for recurring research themes in antenna metrology as well as for research problems that arise in connection with the specific task of correcting for probe position errors in planar near-field data. We have implemented a data management procedure that automatically keeps track of the various datasets being created and stored during the course of research. Because of the highly modular nature of the PNFC new research modules can be easily constructed and incorporated into the total system. A large number of independent subroutines are available to support new efforts, and new subroutines can be added without any difficulty.

Streamlining computations along the lines presented in this software package can reduce significantly the time needed to obtain answers to complicated research problems. Adding to the current version of the package will in time result in a truly comprehensive software package capable of dealing with most computational needs of antenna metrology. For this reason all users are encouraged to add to the effort as they see appropriate.

References

[1] L. A. Muth and R. L. Lewis, *Planar Near-Field Codes for Personal Computers*, National Institute of Standards & Technology Internal Report, NISTIR 89-3929, Oct. 1989.

[2] L. A. Muth and R. L. Lewis, *A general technique to correct probe position errors in planar near-field measurements to arbitrary accuracy*, IEEE Trans. Antennas Propagat., vol AP-38, no. 12, pp 1925-1932, Dec. 1990.

Table 1

Definition of Symbols Used in Naming Modules

SYMBOL	MEANING
0	initial, set to 0
1	version 1
2	squared quantity
act	active, activate
act0	set ACTive switch to 0
add	additional
add0	set ADDitional switch to 0
amp	amplitude
ap	amplitude, phase
cbd	complex binary dataset
cor	correct, corrected, correction
db	in decibels
dif	difference
div	divide, divided (ratio)
drv	derivative
ds	direct sum
dacb	direct access complex binary (file)
dat	.DAT (file)
dbp	dB, phase complex storage
dc	decrement
dec	decrement
deriv	derivative
dif2	difference between squared amplitudes
difa	difference in amplitude
dz	function dz
ec	error correction
err	error
ff	far field
ff0	original far field
grd	.GRD (DOS file extension)
hst	history
inc	increment
init	initialize
laplcn	Laplacian
make	make
nf	near field
nf0	original near field
nc	increment
norm0	normalization of original datasets
norm1	normalization with 1

op	operator
prn	print
rbd	real binary dataset
rd	read
rms	root mean square
show	show
sw	switch
to_	to
ts	Taylor series
u	utility
un	unit number

Table 2

List of Modules That Perform Basic Computational Tasks

UAMP2CBD	read a near-field or a far-field dataset and write its squared amplitude to a complex binary data file
UAPDACB	read an amplitude, phase ASCII file and write a direct-access complex binary file
UCBDDAT	read a complex binary data file and create a .DAT file for x-y plots
UCBDGRD	read a complex binary data file and create a two-dimensional .GRD file for contour or surface plotting
UDBPDACB	read a dB,phase ASCII file and write a direct-access complex binary file
UDERIV	read a near-field dataset and write the derivative of some specified order
UDIF2CBD	read two far-field or near-field datasets and write the difference of the squared amplitudes to a CBD file
UDIFACBD	read two far-field or near-field datasets and write the difference of the amplitudes to a CBD file
UDIFCBD	read two far-field or near-field datasets and write the complex difference to a CBD file
UDIFDB	read two far-field or near-field datasets and write the difference of amplitudes in dBs and the phase difference to a CBD file
UDIVCBD	read two far-field or near-field datasets and write the complex ratio to a CBD file
UDIVRBD	read two real binary data files and write the ratio to a RBD file
UDSX	create a near-field dataset containing x-axis position errors using the direct-sum algorithm
UDSXY	create a near-field dataset containing both x-axis and y-axis position errors using the direct-sum algorithm
UDSXYZ	create a near-field dataset containing position errors along all three coordinate axes using the direct-sum algorithm
UDSY	create a near-field dataset containing y-axis position errors using the direct-sum algorithm
UDSZ	create a near-field dataset containing z-axis position errors using the direct-sum algorithm
UECX4	read a near-field dataset containing x-axis position errors and perform a fourth-order error correction
UECY4	read a near-field dataset containing y-axis position errors and perform a fourth-order error correction
UECZ2	read a near-field dataset containing z-axis position errors and perform a second-order error correction
UECZ3	read a near-field dataset containing z-axis position errors and perform a third-order error correction

UECZ4	read a near-field dataset containing z-axis position errors and perform a fourth-order error correction
UKCORR	read a near-field dataset containing z-axis position errors and multiply by the phase-correction factor $e^{-ik\delta z}$ to obtain a zeroth-order error correction
ULAPLCN	read a near-field dataset and form the Laplacian and check that it satisfies the scalar wave equation
UMAKEDX	create an array DX using a specified error function and write a GRD file to plot the error function
UMAKEDY	create an array DY using a specified error function and write a GRD file to plot the error function
UMAKEDZ	create an array DZ using a specified error function and write a GRD file to plot the error function
UOPNORM	calculate the norm of the error operator
UPRDBCBD	print the amplitude in decibels and the phase in degrees of specified rows and columns of a complex binary data file
UPRNCBD	print the magnitude and phase of specified rows and columns of a complex binary data file
UPRRICBD	print the real and imaginary values of specified rows and columns from a complex binary data file
URBDDAT	read a real binary data file and create a .DAT file for plotting a specified row and column as x-y plots
URBDGRD	read a real binary dataset and create a two-dimensional .GRD file for plotting contour or surface plots
URDFFNF	read a far-field dataset and transform it to a near-field dataset
URDNFFF	read a near-field dataset and transform it to a far-field dataset
URMSCBD	sum the rms values at grid points of a CBD file, and create a .DAT file for plotting
USUBGRD	convert a specified part (SUB) of a CBD file to a GRD file for plotting
UTSNFAP	create two additional near-field data sets by mixing amplitudes and phases from two existing near-field data sets
UTSTZ	create an error-field dataset containing z-coordinate position errors using the Taylor series method
UTSX	introduce x-coordinate position errors into a near-field dataset using the Taylor series method
UTSXY	introduce both x- and y-coordinate position errors into a near-field dataset using the Taylor series method
UTSY	introduce y-coordinate position errors into a near-field dataset using the Taylor series method
UTSZ	introduce z-coordinate position errors into a near-field dataset using the Taylor series method

Table 3

List of Modules That Perform Basic Data Management Functions

UACT0	set the *active* unit number to 0
UACTADD0	set the *active* and *additional* unit numbers to 0
UACTDDB	set the *active* unit number to the value in *difdb.iun*
UACTDIF	set the *active* unit number to the value in *dif.iun*
UACTDIV	set the *active* unit number to the value in *div.iun*
UACTDRV	set the *active* unit number to the value in *drv.iun*
UACTDS	set the *active* unit number to the value 0 and write error message
UACTECX	set the *active* unit number to the value in *ecx.iun*
UACTECY	set the *active* unit number to the value in *ecy.iun*
UACTECZ	set the *active* unit number to the value in *ecz.iun*
UACTFF	set the *active* unit number to the final value in *ff.iun*
UACTFF0	set the *active* unit number to the initial value in *ff.iun*
UACTKEC	set the *active* unit number to the value in *kec.iun*
UACTNF	set the *active* unit number to the final value in *nf.iun*
UACTNF0	set the *active* unit number to the initial value in *nf.iun*
UACTTSX	set the *active* unit number to the value in *tsx.iun*
UACTTSXY	set the *active* unit number to the value in *tsxy.iun*
UACTTSY	set the *active* unit number to the value in *tsy.iun*
UACTTSZ	set the *active* unit number to the value in *tsz.iun*
UADD0	set the *additional* unit number 0
UADDDRV	set the *additional* unit number to the value in *drv.iun*
UADDDS	set the *additional* unit number to 0 and write error message
UADDECX	set the *additional* unit number to the value in *ecx.iun*
UADDECY	set the *additional* unit number to the value in *ecy.iun*
UADDECZ	set the *additional* unit number to the value in *ecz.iun*
UADDFF	set the *additional* unit number to the final value in *ff.iun*
UADDFF0	set the *additional* unit number to the initial value in *ff.iun*
UADDNF	set the *additional* unit number to the final value in *nf.iun*
UADDNF0	set the *additional* unit number to the initial value in *nf.iun*
UADDTSX	set the *additional* unit number to the value in *tsx.iun*
UADDTSXY	set the *additional* unit number to the value in *tsxy.iun*
UADDTSY	set the *additional* unit number to the value in *tsy.iun*
UADDTSZ	set the *additional* unit number to the value in *tsz.iun*
UDCDFDV	decrement the unit number recorded in *difdiv.iun*
UDECFF	decrement the value of the *current* far-field unit number
UDECNF	decrement the value of the *current* near-field unit number
UINCFF	increment the unit number recorded in *ff.iun*
UINCNF	increment the unit number recorded in *nf.iun*
UINITUN	initialize the system parameters and unit numbers
UNCDFDV	increment the unit number recorded in *difdiv.iun*
UNORM0	set the far-field and near-field normalization constants to their initial values

UNORM1	set the far-field and near-field normalization constants to unity
URESTFF	restore the unit numbers in *ff.iun* to the values saved in *save.ffs*
URESTNF	restore the unit numbers in *nf.iun* to the values saved in *save.nfs*
USAVEFF	save the unit numbers recorded in *ff.iun* in *save.ffs*
USAVENF	save the unit numbers recorded in *nf.iun* in *save.nfs*
USCLDX1	set the value of *scale.dx* to the first number in *epsxyz.set*
USCLDX2	set the value of *scale.dx* to the second number in *epsxyz.set*
USCLDX3	set the value of *scale.dx* to the third number in *epsxyz.set*
USCLDY1	set the value of *scale.dy* to the first number in *epsxyz.set*
USCLDY2	set the value of *scale.dy* to the second number in *epsxyz.set*
USCLDY3	set the value of *scale.dy* to the third number in *epsxyz.set*
USCLDZ1	set the value of *scale.dx* to the first number in *epsxyz.set*
USCLDZ2	set the value of *scale.dx* to the second number in *epsxyz.set*
USCLDZ3	set the value of *scale.dx* to the third number in *epsxyz.set*
USETDB1	copy the first set of values of *dbctoff,dbfloor* in *dbmins.set* to *dbmin.db*
USETDB2	copy the second set of values of *dbctoff,dbfloor* in *dbmins.set* to *dbmin.db*
USETDB3	copy the third set of values of *dbctoff,dbfloor* in *dbmins.set* to *dbmin.db*
USHOWUN	display on the screen the current system parameter settings and the current unit settings
USWBOTH	record the value *ffnf* into *ffornf.iun*
USWFFNF	toggle the value recorded in *ffornf.iun* between *ff* and *nf*
USWTOAMP	record the value *amp* in *ampordb.grd*
USWTODB	record the value *dB* in *ampordb.grd*
USWTOFF	record the value *ff* in *ffornf.iun*
USWTONF	record the value *nf* in *ffornf.iun*
USWTONON	record the value *none* in *ampordb.grd*

Table 4

List of Parameter Files Used by the Research Modules and the Data Files They Create

MODULE	PARAMETER FILES	DATA FILES
UAMP2CBD	dabd.iof, difdiv.iun	uamp2cbd.out, amp2.iun, fort.*xx*[1]
UAPDACB	adab.iof	[adab.iof][2]
UCBDDAT	dabd.iof, ampordb.grd ampnorn.nf, ampnorm.ff difdiv.iun, dbmin.db dbloss.grd, iregion.nf iregion.ff, plot.dir	nfyamp.dat, nfyphase.dat nfxamp.dat, nfxphase.dat ffyamp.dat, ffyphase.dat ffxamp.dat, ffxphase.dat ucbddat.out
UCBDGRD	ampordb.grd, ampnorm.nf ampnorm.ff, dbloss.grd dbmin.db , dabd.iof, plot.dir	nfamp.grd, nfphase.grd ffamp.grd, ffphase.grd ucbdgrd.out
UDBPDACB	adab.iof	[adab.iof][2]
UDERIV	order.drv, dabd.iof, sub.grd dbmin.db, plot.dir	order.drv, uderiv.out, fort.*xx*[1] drv.iun, drvamp.grd, drvphase.grd
UDIF2CBD	difdiv.iun, dabd.iof	udif2cbd.out, dif2.iun, fort.*xx*[1]
UDIFACBD	difdiv.iun, dabd.iof	udifacbd.out, dif2.iun, fort.*xx*[1]
UDIFCBD	difdiv.iun, dabd.iof	udifcbd.out, dif.iun, fort.*xx*[1]
UDIFDB	ampnorm.nf, ampnorm.ff difdiv.iun, dabd.iof, dbmin.db	ddbamp.grd, ddbphase.grd udifdb.out, difdb.iun, fort.*xx*[1]
UDIVCBD	difdiv.iun, dabd.iof iregion.ff, iregion.nf	udivcbd.out, div.iun fort.*xx*[1]
UDIVRBD	difdiv.iun, dabd.iof iregion.ff, iregion.nf	udivrbd.out, rdiv.iun fort.*xx*[1]
UDSX	sub.ds, filter.ff, scale.dx sub.prn, dabd.iof, fun.dx dx.iun	ds.iun, uds_out.*xx* dsnf.*xx*[3]
UDSXY	sub.ds, filter.ff, scale.dx scale.dy, sub.prn, dabd.iof fun.dx, fun.dy, dx.iun, dy.iun	ds.iun, uds_out.*xx* dsnf.*xx*[3]
UDSXYZ	sub.ds, filter.ff, scale.dx scale.dy, scale.dz, sub.prn dabd.iof, fun.dx, fun.dy fun.dz, dx.iun, dy.iun, dz.iun	ds.iun, uds_out.*xx* dsnf.*xx*[3]
UDSY	filter.ff, scale.dy, dabd.iof sub.ds, sub.prn, fun.dy, dy.iun	ds.iun, uds_out.*xx* dsnf.*xx*[3]
UDSZ	filter.ff, scale.dz, dabd.iof sub.ds, sub.prn, fun.dz, dz.iun	ds.iun, uds_out.*xx* dsnf.*xx*[3]
UECX4	tsx.iun, dx.iun, fun.dx scale.dx, dabd.iof	uecx4.out, ecx.iun, fort.*xx*[1]
UECY4	tsy.iun, dy.iun, fun.dy scale.dy, dabd.iof	uecy4.out, ecy.iun, fort.*xx*[1]

UECZ2	tsz.iun, dz.iun, fun.dz scale.dz, dabd.iof	uecz2.out, ecz.iun, fort.xx[1]
UECZ3	tsz.iun, dz.iun, fun.dz scale.dz, dabd.iof	uecz3.out, ecz.iun, fort.xx[1]
UECZ4	tsz.iun, dz.iun, fun.dz scale.dz, dabd.iof	uecz4.out, ecz.iun, fort.xx[1]
UKCORR	dabd.iof, tsz.iun, dz.iun fun.dz, scale.dz	ukcorr.out, kec.iun fort.xx[1]
ULAPLCN	dabd.iof, plot.dir	lnamp.grd, amp0.grd, ulaplcn.out lnphase.grd, phase0.grd
UMAKEDX	dabd.iof, fun.dx, polydx.par perdx.par, randx.par	iregion.ff, iregion.nf, [dx.grd][2] dx.iun, umakedx.out, fort.xx[1]
UMAKEDY	dabd.iof, fun.dy, polydy.par perdy.par, randy.par	iregion.ff, iregion.nf, [dy.grd][2] dy.iun, umakedy.out, fort.xx[1]
UMAKEDZ	dabd.iof, fun.dz, polydz.par perdz.par, randz.par	iregion.ff, iregion.nf, [dz.grd][2] dz.iun, umakedz.out, fort.xx[1]
UOPNORM	dabd.iof, scale.dz, iregion.ff difdiv.iun, iregion.nf, fun.dz	uopnorm.out
UPRDBCBD	ampnorm.nf, ampnorm.ff dabd.iof, dbloss.grd dbmin.db, sub.prn	uprdbcbd.out
UPRNCBD	dabd.iof, sub.prn	uprncbd.out
UPRRICBD	dabd.iof, sub.prn	uprricbd.out
URBDDAT	dabd.iof, plot.dir, sub.dat iregion.ff, iregion.nf	ff_y_mag.dat, ff_x_mag.dat nf_x_mag.dat, nf_y_mag.dat urbddat.out
URBDGRD	dabd.iof, plot.dir	ff_mag.grd, nf_mag.grd, urbdgrd.out
URDFFNF	filter.ff, dabd.iof ffnf.dz, ampnorm.ff ampnorm.nf, db.ff, db.nf	iregion.ff, iregion.nf, db.ff db.nf, urdffnf.out, ampnorm.ff ampnorm.nf, ffornf.iun, fort.xx[1]
URDNFFF	filter.ff, dabd.iof ampnorm.ff, ampnorm.nf db.ff, db.nf	iregion.ff, iregion.nf, db.ff, db.nf urdnfff.out, sub.ds, ampnorm.ff ampnorm.nf, ffornf.iun, fort.xx[1]
URMSCBD	dabd.iof, ampnorm.ff plot.dir	db.ff, db.nf, ampnorm.ff urmscbd.out, rms.dat, fort.xx[1]
USUBGRD	dbloss.grd, dabd.iof, plot.dir ampnorm.nf, ampordb.grd ampnorm.ff, dbmin.db, sub.grd	ffamp.grd, ffphase.grd nfamp.grd, nfphase.grd usubgrd.out
UTSNFAP	tsz.iun, dabd.iof	utsnfap.out, tsamp.iun, tsphs.iun fort.xx[1]
UTSTZ	dz.iun, dabd.iof fun.dz, scale.dz	utstz.out, tstz.iun fort.xx[1]
UTSX	dabd.iof, filter.ff, sub.prn dx.iun, fun.dx, scale.dx	iregion.ff, iregion.nf, tsx.iun utsx.out, fort.xx[1]

UTSXY	dabd.iof, filter.ff, sub.prn dx.iun, fun.dx, scale.dx dy.iun, fun.dy, scale.dy	iregion.ff, iregion.nf, tsxy.iun utsxy.out, fort.*xx*[1]
UTSY	dabd.iof, filter.ff, sub.prn dy.iun, fun.dy, scale.dy	iregion.ff, iregion.nf, tsy.iun utsy.out, fort.*xx*[1]
UTSZ	dabd.iof, filter.ff, sub.prn dz.iun, fun.dz, scale.dz	iregion.ff, iregion.nf, tsz.iun utsz.out, fort.*xx*[1]

[1] The DOS extension number *xx* added to the filename FORT is recorded in the appropriate .IUN file

[2] The brackets [*filename*] is to be understood as the contents of the *filename*. For example, the output file name is read in as a parameter from file adab.iof

[3] The DOS extension number *xx* added to filename DSNF and to filename UDS_OUT is recorded in file DS.IUN.

Table 5

List of Parameter and Data Management Files Used by the Data Management Modules

MODULE	PARAMETER FILES	OUTPUT FILE
UACT0		active.iun
UACTADD0		active.iun, add.iun
UACTDDB	difdb.iun	active.iun
UACTDIF	dif.iun	active.iun
UACTDIV	div.iun	active.iun
UACTDRV	drv.iun	active.iun
UACTDS	- ERROR[1] -	active.iun
UACTECX	ecx.iun	active.iun
UACTECY	ecy.iun	active.iun
UACTECZ	ecz.iun	active.iun
UACTFF	ff.iun	active.iun
UACTFF0	ff.iun	active.iun
UACTKEC	kec.iun	active.iun
UACTNF	nf.iun	active.iun
UACTNF0	nf.iun	active.iun
UACTTSX	tsx.iun	active.iun
UACTTSXY	tsxy.iun	active.iun
UACTTSY	tsy.iun	active.iun
UACTTSZ	tsz.iun	active.iun
UADD0		add.iun
UADDDRV	drv.iun	add.iun
UADDDS	- ERROR[1] -	add.iun
UADDECX	ecx.iun	add.iun
UADDECY	ecy.iun	add.iun
UADDECZ	ecz.iun	add.iun
UADDFF	ff.iun	add.iun
UADDFF0	ff.iun	add.iun
UADDNF	nf.iun	add.iun
UADDNF0	nf.iun	add.iun
UADDTSX	tsx.iun	add.iun
UADDTSXY	tsxy.iun	add.iun
UADDTSY	tsy.iun	add.iun
UADDTSZ	tsz.iun	add.iun
UDCDFDV	difdiv.iun, ffornf.iun ff.iun, nf.iun	difdiv.iun
UDECFF	ff.iun	ff.iun
UDECNF	nf.iun	nf.iun
UINCFF	ff.iun	ff.iun
UINCNF	nf.iun	nf.iun

UINITUN	init.iun, data.dir	active.iun, add.iun, amp2.iun ampordb.grd, asci.iun, dif.iun dif2.iun, difdb.iun, difdiv.iun div.iun, drv.iun, dx.iun, dy.iun dz.iun, ecx.iun, ecy.iun, ecz.iun ff.iun, ffnf.dz, filter.ff, fun.dx fun.dy, fun.dz, kec.iun, nf.iun order.drv, rdiv.iun, tsamp.iun tsphs.iun ,tstz.iun, tsx.iun tsxy.iun, tsy.iun, tsz.iun
UNCDFDV	difdiv.iun, ffornf.iun ff.iun, nf.iun	difdiv.iun
UNORM0	tempnorm.nf, tempnorm.ff ampnorm.nf, ampnorm.ff	tempnorm.nf, ampnorm.nf tempnorm.ff, ampnorm.ff
UNORM1	ampnorm.nf, ampnorm.ff tempnorm.nf, tempnorm.ff	tempnorm.nf, ampnorm.nf tempnorm.ff, ampnorm.ff
URESTFF	save.ffs	ff.iun
URESTNF	save.nfs	nf.iun
USAVEFF	ff.iun	save.ffs
USAVENF	nf.iun	save.nfs
USCLDX1	epsxyz.set, scale.dx	scale.dx
USCLDX2	epsxyz.set, scale.dx	scale.dx
USCLDX3	epsxyz.set, scale.dx	scale.dx
USCLDY1	epsxyz.set, scale.dy	scale.dy
USCLDY2	epsxyz.set, scale.dy	scale.dy
USCLDY3	epsxyz.set, scale.dy	scale.dy
USCLDZ1	epsxyz.set, scale.dz	scale.dz
USCLDZ2	epsxyz.set, scale.dz	scale.dz
USCLDZ3	epsxyz.set, scale.dz	scale.dz
USETDB1	dbmins.set	dbmin.db
USETDB2	dbmins.set	dbmin.db
USETDB3	dbmins.set	dbmin.db
USHOWUN	active.iun, add.iun, amp2.iun ampnorm.ff, ampnorm.nf, ampordb.grd asci.iun, data.dir, dif.iun, dif2.iun difdb.iun, difdiv.iun, div.iun, drv.iun ds.iun, dx.iun, dy.iun, dz.iun, ecx.iun ecy.iun, ecz.iun, ff.iun, ffnf.dz ffornf.iun, filter.ff, fun.dx, fun.dy fun.dz, kec.iun, nf.iu, order.drv rdiv.iun, scale.dx, scale.dy, scale.dz tsamp.iun, tsphs.iun ,tstz.iun, tsx.iun tsxy.iun, tsy.iun, tsz.iun	
USWBOTH		ffornf.iun

USWFFNF	ffornf.iun	ffornf.iun
USWTOAMP		ampordb.grd
USWTODB		ampordb.grd
USWTOFF		ffornf.iun
USWTONF		ffornf.iun
USWTONON		ampordb.grd

[1] Output from module UDS is not written to a FORT.*xx* file, but rather to a file named DSNF.*xx* which is stored in another file directory as specified by the local file DATA.DIR. Consequently, these unit numbers do not fit into a purely integer-unit numbering scheme.

Table 6

List of Symbols Used in Naming PNFC Subroutines

SYMBOL	MEANING
0	initialization designator
1	one dimensional, subsequent operation designator, alternate procedure designator
2	two dimensional
a	"a", access, array, ascii, amplitude
b	"b", backward, binary
c	change, convert to, column, complex, constant, copy
d	data, derivative, difference, dimensional, direct (access), disk (as a storage location), double precision
e	exponential, even
f	far, field, file, forward, formated, function
g	gamma, generate
i	imaginary, imaginary part, integer
k	k (integer constant), wave number, spectrum space k
l	l (integer constant)
m	m (integer constant), maximum, minimum, minus, multiple
n	near (fresnel region), negative quantity
o	odd, or
p	parameter(s), phase, plot, plus, power, print, product, pseudo
r	read, real (single precision), real part, row
s	shift, shifted, single precision, store, sum, plural designation
t	taylor (series), times, transform
w	weight, weighted, write
x	coordinate (distance along x axis), general variable designation name change letter (to avoid conflicts)
y	coordinate (distance along y axis)
z	coordinate (distance along z axis)
as	ascii
bd	binary data
ca	complex array "a"
cb	complex array "b"
cc	complex constant
ch	character variable
cm	centimeter
cr	create
da	direct access
db	decibel
df	difference
ds	direct sum
dx	derivative with respect to x, increment in x direction

dy	derivative with respect to y, increment in y direction
dz	derivative with respect to z, increment in z direction, z error terms
ec	error correction
eq	equality
ff	far field
fp	floating point
fs	files
hd	header
hi	high
im	imaginary
ik	third index of three dimensional array
iy	first (column) index of an array
jx	second (row) index of an array
ln	logarithm
mk	make
mm	maximum/minimum
mx	maximum
nf	near field
or	or
pf	plot file
ph	phase
rc	real constant
rd	read
re	real
rz	real dz array
sm	sum
sq	square, squared
to	to
ts	taylor series
ud	update
un	unit number, "unorm"
wl	wave length
wn	wave number
xp	exponential
amp	amplitude
add	add
ary	array
asc	ascii
box	box
cos	cosine
chk	check
cnt	center
dat	file extension designation for two-dimensional plot files
dif	difference

div	divide
dnf	derivative of near field
dot	dot product (of two vectors)
drv	derivative
end	end
err	error, error field
exp	exponent
fbt	forward/backward transform
fft	fast fourier transform
fil	file
flt	filter
fun	function
get	get
grd	file extension designation for perspective-plot files
hst	history
iof	input/output file
img	imaginary
inp	input
ins	insertion
int	integrate
iun	integer unit number
leq	less than or equal
log	logarithm
mak	make
mul	multiply
mod	modulate, modulated by
out	output
par	parameter
per	periodic
pff	psuedo far field
plt	plot
ply	polynomial
prg	program
prn	print
pws	plane-wave spectrum
scl	scale
set	set, setup
sft	shift
sin	sine
str	store
aray	array
bndr	boundry
char	character
gama	gamma

gamm	gamma
file	file
fils	files
find	find
fltr	filter
func	function
grid	grid (coordinate grid)
init	initalize
limt	limit
loss	loss
make	make
mult	multiply
mess	message
prnt	print
rndm	random
swap	swap
unit	unit
gamma	gamma
polyn	polynomial
ratio	ratio
const	constant
lngth	length
range	range
laplcan	Laplacian

Table 7

List of Subroutines of the PNFC

ACPCFFD	introduce Amplitude Change and Phase Change to Far-Field Data
ACPCNFD	introduce Amplitute Change and Phase Change to Near-Field Data
ACTIUN	return integer value from ACTIVE.IUN if nonzero
ADABIOF	get the Input Output File for ASCII to Direct Access Binary routines
ADABPAR	read the PARameters for the ASCII to Direct Access Binary conversion
ADDBOX	ADD a BOX function of amplitude EPS to complex CDATA
AMPDIF2	obtain the difference between the absolute values of different planes of a 3-dimensional complex array and obtain the new array's maxima and minima (2-dimensional)
APDSET1	convert two columns or rows of a 3-dimensional array to real-imaginary or amplitude-phase format and form a complex difference array (1-dimensional)
APNAME	append a given filename to the character string 'AP'
CABD	convert two ASCII datasets to a self-documented Complex Binary Dataset
CABDIOF	get the Input Output File for the Conversion of ASCII to Binary Data
CABDPAR	read PARameters for the Conversion of ASCII to Binary Data routines
CABS1	compute ABSolute values of a Complex array (1-dimensional)
CACBK3D	multiply a 3-dimensional complex array by a 2-dimensional complex array raised to an integer power
CACEIPH	Complex Array times Exponential to power I times real array (2-dimensional)
CADACB	read two ASCII files and create a self-documented Direct-Access Complex Binary file
CADD1	ADDition of two Complex arrays (1-dimensional)
CADD2	ADDition of two Complex arrays (2-dimensional)
CADD3	ADDition of two Complex arrays (3-dimensional)
CADDCC2	ADDition of a Complex Constant to a Complex Array (2-dimensional)
CAEIPH2	Complex Array times Exponential to power I times real array times real constant (2-dimensional)
CAEIPHC	Complex Array times Exponential to power I times Phase Constant (2-dimensional)
CANECB2	copy a complex array CA to complex array CB, where CA and CB may have unequal column lengths (2-dimensional)
CAPCCD1	Complex Array Plus Complex Constant in Double precision (1-dimensional)
CAPRI	convert Complex numbers with Amplitude-Phase format to Real-Imaginary format (2-dimensional)
CAPRI1	convert Complex numbers with Amplitude-Phase format to Real-Imaginary format (1-dimensional)
CAPRI2D	convert Complex numbers with Amplitude-Phase format numbers to Real-Imaginary format using double-precision intrinsic functions
CAPRNT	PRiNT the maximum and minimum values of a Complex Amplitude-Phase array
CARAYMX2	find a Complex ARrAY's MaXimum (2-dimensional)

CARBK3D	multiply a 3-dimensional double-precision complex array by a 2-dimensional double-precision array raised to an integer power (3-dimensional)
CARBKD3	multiply a 3-dimensional complex array by a 2-dimensional double-precision array raised to an integer power (3-dimensional)
CARCBD2	Complex Array plus Real weight times Complex double-precision array (2-dimensional)
CASUMSQ	Sum the squares of the absolute values of selected elements of a complex array (2-dimensional)
CATOCB1	Copy a complex array CA to CB (1-dimensional)
CATOCB2	Copy a complex array CA TO CB (2-dimensional)
CBDTODB	read a Complex Binary Dataset, and convert from (real, imaginary) TO (amplitude, phase) format, with amplitude in DB
CCA2B1	Copy a Complex Column of data from array CA (2-dimensional) to array CB (1-dimensional)
CDFSET1	Convert the DiFerence between two columns or rows of a 3-dimensional complex array from real-imaginary to amplitude-phase (1-dimensional)
CDIF1	Complex DIFerence of two arrays (1-dimensional)
CDIF2	Complex DIFerence of two arrays (2-dimensional)
CDIVDS2	Complex DIVision of single-precision array by Double-precision array
CDOT	Complex Dot product
CGATHER	sequentially copy regularly spaced elements of one array to another
CHKEQFP	CHecK for EQuality between two Floating Point numbers; stop if unequal
CHKEQI	CHecK whether two Integers are EQual; stop if unequal
CHKLEQI	CHecK whether one Integer is Less than or EQual to another; stop otherwise
CHKPAR0	CHecK if a parameter exceeds its specified maximum value; stop and deliver specialized error message
CHKPAR1	CHecK if a set of two parameters exceed their maximum values; stop and print specialized error message
CHKPAR2	CHecK if a set of two parameters exceed their maximum values; stop and print specialized error message
CHLAST	Locate the last non-blank character of a string 80 characters long
CHLNGTH	determine number of CHaracters up to the first blank in a character variable
CIMGSTR	SToRe the IMaGinary part of a specified row and column of a Complex array
CINIT1	INITialize a Complex array with a complex constant (1-dimensional)
CINIT2	INITialize a Complex array with a Complex constant (2-dimensional)
CIRCFLT	CIRCular FiLTer of a complex data array
CMULDS2	Complex MULtiplication of a Double-precision complex array by a Single-precision complex array (2-dimensional)
CMULRD2	Complex MULtiplication of Real Double-precision array by a complex array
CMULT1	Complex MULTiplication of two arrays (1-dimensional)
CMULT2	Complex MULTiplication of two arrays (2-dimensional)
CMULTR2	Complex MULTiplication of Real array by complex array (2-dimensional)
CNIMCC1	add Complex Constant to Negative Imaginary part of Complex array

CNTCACB	CeNTer Complex Array within a zero padded Complex array CB
COEFFTS	calculate and store the COEFFicients of the Taylor Series
CONST	function to return the value unity
CONSTAX	function to return the value unity
COS2	function to return COSine squared of x
COS3	function to return COSine cubed of x
COS4	function to return COSine of x raised to the fourth power
COSAX	function to return COS(A*X)
COSAX2	function to return COSine squared of A*X
COSX	function to return COS(X)
CRA2B1	copy a Row of a 2-dimensional Complex array into a 1-dimensional array
CRATIO2	Complex RATIO of two arrays (2-dimensional)
CRFUNC	CReate or initialize a direct access file to record FUNCtion names used
CRIAP	Convert Real-Imaginary complex array to Amplitude-Phase (2-dimensional)
CRIAP1	Convert Real-Imaginary complex array to Amplitude-Phase (1-dimensional)
CRIAP2D	Convert Real-Imaginary Complex array to Amplitude-Phase using double-precision trigonometic functions (2-dimensional)
CRITOC2	Form a Complex array from the Real part of one complex array and the Imaginary part of another complex array (2-dimensional)
CRNFERR	CReate a Near Field with ERRors using multiple Fourier transforms and a specified error function dz
CSMWCP1	Complex SuM (1-dimensional) of Product of real Weights (1-dimensional), real array (1-dimensional) and a Complex array (2-dimensional)
CSMWCP2	Complex SuM (2-dimensional) of Product of real Weights (1-dimensional), real array (2-dimensional) and a Complex array (2-dimensional)
CSQR1	obtain the square of the absolute values of selected elements of a complex array (1-dimensional)
CSUM1	SUM a 2-dimensional complex array by columns (1-dimensional)
CSUM2	Complex SUM over third dimension of a 3-dimensional array (2-dimensional)
CSUMCP1	Complex SUM by rows of the product of the elements of a 2-dimensional real array and a 2-dimensional complex array (1-dimensional)
CSUMCP2	Complex SUM over the third dimension of the element-by-element product of a 3-dimensional complex array and a 2-dimensional real array (2-dimensional)
CSUMIK2	Complex SUM of a 3-dimensional complex array over the third dimension (2-dimensional)
CSUMRW1	Complex SUM of Rows (over the second dimension) of a complex array Weighted by a real 1-dimensional array (1-dimensional)
CSUMRW2	Complex Sum over the third dimension of a complex array Weighted by real 1-dimensional array (2-dimensional)
CSWCPE2	Weighted Complex Sum of the Product of a 3-dimensional ComPlex array times a real array raised to an array of Exponents (2-dimensional)
CUTOFF1	Set an array element to zero when the difference between a constant squared and the array element squared is less than a specified value (1-dimensional)

CUTOFF2	Set an array element to zero when the element's value is smaller than a specified constant (2-dimensional)
CXPCLOG	add a Complex EXPonent array to the Complex LOGarithm of a complex array
DABDIOF	obtain the Direct-Access Binary Data Input Output File
DABDPAR	obtain Direct-Access Binary Data's PARameters
DATFILE	create an ASCII DATa FILE for multiple plots
DATORB1	copy a Double-precision Array TO a Real array (1-dimensional)
DB1	convert real part of complex array to dB (1-dimensional)
DB2	convert real part of complex array to dB (2-dimensional)
DCLN2	Double-precision Complex Logarithm of complex array (2-dimensional)
DNFDX	Derivative of Near Field with respect to the X coordinate
DNFDXDY	mixed Derivatives with respect to X and Y coordinates
DNFDY	Derivative of Near Field with respect to the Y coordinate
DNFDZ	Derivatives of Near Field with respect to the Z coordinate
DNFDZE	Derivatives of Near Field with respect to Z, Even orders
DNFDZO	Derivatives of Near Field with respect to Z, Odd orders
DSPWS	Direct Sum of Plane Wave Spectrum
DSPWSX	Direct Sum of Plane Wave Spectrum to compute the near field in the x-y plane, where the X coordinate can have arbitrary errors
DSPWSXY	Direct Sum of Plane Wave Spectrum to compute the near field in the x-y plane, where both the X and the Y coordinates can have arbitrary errors
DSPWSY	Direct Sum of Plane Wave Spectrum to compute the near field in the x-y plane, where the Y coordinate can have arbitrary errors
DSWCRP1	Double-precision Sum of the Weighted Product of a constant to a array of integer powers times a complex array (1-dimensional)
ECEXP	compute the Complex EXPonential of a double-precision complex array (2-dimensional)
ECX4	given a near field contaminated with X errors, apply the Error-Correction technique to $4th$ order
ECY4	given a near field contaminated with Y errors, apply the Error-Correction technique to $4th$ order
ECZ4	given a near field contaminated with Z errors, apply the Error-Correction technique to $4th$ order
EIGAMAZ	create an array of phase factors $e^{i\gamma z}$ (2-dimensional)
ERRMESS	print a set of specified ERRor MESSages
FAXSBYS	create the product of external Functions fx and fy, which are of the form A*(X-xS) and B*(Y-yS)
FFLIMTS	Far Field LIMiTS to specify the range of a far field coordinate when summing the plane wave spectrum
FFNF	given a Far-Field, compute the corresponding Near-Field, using the FFT
FFNFIUN	read the Far Field or Near Field Unit Numbers
FFNFX	given a Far Field, obtain the Near Field, when the X coordinate may have errors, using the direct-sum routines

FFNFXY	given a Far Field, obtain the Near Field, when the X and the Y coordinates may have errors, using the direct-sum routines
FFNFXYZ	given a Far Field, obtain the Near Field, when the X, Y and the Z coordinates may have errors, using the direct-sum routines
FFNFY	given a Far Field, obtain the Near Field, when the Y coordinate may have errors, using the direct-sum routines
FFNFZ	given a Far Field, obtain the Near Field, when the Z coordinate may have errors, using the direct-sum routines
FFORNF	read the FFORNF.IUN file
FFPFF	given a Far-Field, obtain the Pseudo Far-Field
FFRANGE	select the Far Field RANGE of variables for summing the plane wave spectrum
FFTFFT	Fast Fourier Transform followed by inverse Fast Fourier Transform
FILSIOF	get the Input Output File for reading output filenames
FILSPAR	read a list of output filenames
FINDEND	skip to the end of a file
FLTEIGZ	FiLTer the array containing values of $e^{i\gamma z}$ using a specified lower limit on γ^2
FLTLIMT	obtain data-point-spacing criteria for LIMiTing the plane-wave spectrum
FLTPWSG	FiLTer sum of logarithm of Plane-Wave Spectrum added to $e^{i\gamma z}$
FLTRHIK	transform a near field to a far field, FiLTeR HIgh frequencies in K-space, and transform back to a near field (possibly at different z coordinate)
FOURT	Multi-dimensional Cooley-Tukey fast Fourier transform (FFT)
FRBW	Formated Read and Binary Write of a real dataset
FRGRD	Formated Read of a .GRD dataset with conversion from decibels to amplitude
FRRAD	FoRmated Read of a Real ASCII Dataset
FRRADHD	FoRmated Read of a Real ASCII Dataset with HeaDer information
FUNAXBY	create an array equal to the product of FUNctions of the form A*X*B*Y
FUNCSCL	calculate a SCaLed grid-increment for a periodic function
FUNCXY	create an array equal to the product of FUNctions of the form f(X)*f(Y)
FWDCRA1	Formatted Write of a Real Array obtained form a Double-precision array (1-dimensional)
FWRAD1	Formated Write of Real ASCII Dataset (1-dimensional)
FWRAD2	Formated Write of Real ASCII Dataset (2-dimensional)
GAMMASQ	calculate real double-precision array γ^2
GETFILE	obtain the next filename from an array of filenames
GETWN	given the frequency, calculate the wavenumber
GETWND	given the frequency (in double-precision), calculate the wavenumber
GRDDACB	read amplitude and phase .GRD files, and write a Direct-Access Complex Binary Data file
GRID	create a single-precision GRID along an axis
GRIDD	create a Double-precision GRID along an axis
GTPRNPR	GeT the PRriNt PaRameters for rows and/or columns of an array
HSTAMP2	append file information to HiSTory file from program UAMP2CBD
HSTDFDB	append file information to HiSTory file from program UDIFDB

HSTDIF	append file information to HiSTory file from program UDIFCBD
HSTDIF2	append file information to HiSTory file from program UDIF2CBD
HSTDIFA	append file information to HiSTory file from program UDIFACBD
HSTDIV	append file information to HiSTory file from program UDIVCBD
HSTDRV	append file information to HiSTory file from program UDERIV
HSTDS	append file information to HiSTory file from program UDS
HSTDS3	append file information to HiSTory file from program UDSXYZ
HSTDSX	append file information to HiSTory file from program UDSX
HSTDSXY	append file information to HiSTory file from program UDSXY
HSTDSY	append file information to HiSTory file from program UDSY
HSTEC	append file information to HiSTory file from program UERRCOR
HSTFFNF	append file information to HiSTory file from program URDFFNF
HSTKEC	append file information to HiSTory file from program UKCORR
HSTMKDX	append file information to HiSTory file from program UMAKEDX
HSTMKDY	append file information to HiSTory file from program UMAKEDY
HSTMKDZ	append file information to HiSTory file from program UMAKEDZ
HSTNFFF	append file information to HiSTory file from program URDNFFF
HSTNRM	append file information to HiSTory file from program UOPNORM
HSTRDIV	append file information to HiSTory file from program UDIVRBD
HSTRMS	append file information to HiSTory file from program URMSCBD
HSTTS	append file information to HiSTory file from program UTS
HSTTSAP	append file information to HiSTory file from program UTSNFAP
HSTTST	append file information to HiSTory file from program UTST
HSTTSX	append file information to HiSTory file from program UTSX
HSTTSXY	append file information to HiSTory file from program UTSXY
HSTTSY	append file information to HiSTory file from program UTSY
HSTUN0	append file information to HiSTory file from program UNORM0
HSTUN1	append file information to HiSTory file from program UNORM1
INPDABP	INPut Direct-Access Binary file Parameters
INPDACB	INPut Direct-Access Binary file Parameters and the dataset
INPDRVP	INPut a subset of the direct-access binary file parameters needed for the module UDERIV
INPDZP	INPut a subset of the direct-access binary file parameters needed for the module UMAKEDZ
INPFFP	INPut a subset of the direct-access binary file Parameters needed to characterize the Far Field
INPFFP0	INPut Far Field Parameters and data from the direct-access binary file
INPFILS	INPut the filename pointing to a list of FILenameS and read the list
INPGRDP	INPut a subset of the direct-access binary file needed for module UCBDGRD
INPNFP	INPut a subset of the direct-access binary file Parameters needed to characterize the Near Field
INPNFP0	INPut Near-Field Parameters and data from direct-access binary file
INPRCBD	INPut two Real datasets into a Complex Binary Data array
INPTSP	INPut a subset of the direct-access binary file needed for the module UTS

INSLOSS	convert INSertion LOSS from decibels to amplitude and scale data array
INTNF3	INTegral of Near Field with respect to the 3rd coordinate Z
IUNIT	function to increment by 1 the current Integer UNIT number
IUNS	Obtain the unit numbers in files NF.IUN anf FF.IUN and check that additional unit numbers are available
IYJXCNT	determine the midpoint (CeNTer) of two integers
LAPLCAN	calculate the LAPLaCiAN of a near field
MAKEDZ	MAKE a function DZ, which is a function of X and Y
MAKPFF	MAKe a Pseudo Far Field equal to the Fourier transform of the box function
MDARB1	Make two single-precision copies of a Double-precision ARray (1-dimensional)
MDNFDX	Multiple Derivatives of the Near Field with respect to X
MDNFDY	Multiple Derivatives of the Near Field with respect to Y
MDNFDZ	Multiple Derivatives of the Near Field with respect to Z
MDNFDZE	Multiple derivatives of the Near Field with respect to Z, Even orders
MDNFDZO	Multiple derivatives of the Near Field with respect to Z, Odd orders
MIGAMMZ	calculate the array $-i\gamma Z$
MKGAMMA	MaKe the arrays γ^2 and KY and KX
MKPERDZ	MaKe a function DZ, which is a PERiodic function of X and Y
MKPERFN	MaKe a PERiodic FuNction of the form $A \cdot (x - x_0) \cdot B \cdot (y - y_0)$
MKPLYDZ	MaKe a PoLYnomial function DZ of X and Y
MKPOLYN	MaKe a POLYNomial function of x and y raise to a specified power
MMICA	obtain the Minimum and Maximum of a Complex Array's Imaginary part
MMRA	obtain the Minimum and Maximum values of a Real array
MMRCA	obtain the Minimum and Maximum of a Complex Array's Real part
NF	create a complex NF array using arbitrary functions of x and y
NFFF	given a Near Field, compute the corresponding Far Field
NFMODX	MODulate a complex NF array with a function of X
NFMODY	MODulate a complex NF array with a function of Y
NFPFF	given a Near Field, obtain the corresponding Pseudo Far Field
NFTSXK	calculate the Kth term of the Taylor Series in X from Near-Field data
NFTSXY	sum the X & Y Taylor Series terms and print out selected partial sum results
NFTSXYK	calculate the Kth term of the Taylor Series in X & Y from Near-Field data
NFTSYK	calculate the Kth term of the Taylor Series in Y from Near-Field data
NFTSZK	calculate the Kth term of the Taylor Series in Z from Near-Field data
NFZKTS	create a sequence of Near Fields at Z_K in a specified range and number of steps using the Taylor Series technique
NORM2	NORMalize a real array by a constant divided by the difference between the array's maximum and minimum values (2 dimensional)
OUTASC	convert a complex array to amplitude and phase format and OUTput the resulting real and imaginary parts as two ASCII datasets
OUTDACB	OUTput a Direct-Access Complex Binary dataset
OUTDPS	OUTput a complex array to Disk, a Print array and/or a Storage array
OUTGRD	OUTput the amplitude and phase of a complex array .GRD files
OUTPFS0	get 4 filenames and OUTput 4 .PLT files

OUTPFS1	OUTput the amplitude and phase of a specified column and a specified row of a complex array as .PLT FileS
OUTRGRD	OUTput a Real array to a .GRD file
PCCRGRD	Print a Column and/or Row of amplitude and phase data of a Complex array, and output amplitude and phase .GRD files
PERFUNC	specifies a PERiodic FUNCtion to be evaluated
PERFUNX	specify and record the name of the PERiodic FUNction of X to be evaluated
PERFUNY	specify and record the name of the PERiodic FUNction of Y to be evaluated
PERFUNZ	specify and record the name of the PERiodic FUNction of Z to be evaluated
PFCORR	PlotFile data obtained from a column or row of a complex array
PFCRAP	PlotFile data obtained from a column and/or row of a complex array which may have been converted to amplitude and phase
PFFFF	given a Pseudo Far Field, obtain the corresponding Far Field
PFFNF	given a Pseudo Far Field, obtain the corresponding Near Field
PFREAL	PlotFile data obtained from real array array
PFREIM	PlotFile data obtained from the REal or IMaginary part of complex array
PFSET	PlotFile SETup: specify column or row of a complex array and convert complex numbers to amplitude-phase format or vice versa (1-dimensional)
PLRDATA	write a PLotfile of Real Data consisting of 3 equally incremented column arrays and two real arrays (1-dimensional)
PLTFILE	output a real array to .PLT file
POLYN	POLYNomial function of a single variable at a single point
POLYNXY	sum POLYNomial functions of X and Y
PPFCRAP	Print and create Plot File data from a selected column and/or a row of a complex array, which may be converted to amplitude and phase
PPWSNF2	direct-sum of Plane-Wave Spectrum to compute the Near Field at specified points which can include arbitrary displacements in the x and y coordinates
PPWSNF3	direct-sum of the Plane-Wave Spectrum to compute the Near Field at specified points which can include arbitrary displacements in x, y and z coordinates
PPWSNFX	direct-sum of the Plane-Wave Spectrum to compute the Near Field at specified points which can include arbitrary displacements in the X coordinates
PPWSNFY	direct-sum of the Plane-Wave Spectrum to compute the Near Field at specified points which can include arbitrary displacements in the Y coordinates
PPWSNFZ	direct-sum of the Plane-Wave Spectrum to compute the Near Field at specified points which can include arbitrary displacements in the Z coordinates
PRDCTC2	PRoduct of a Double precision Column array to an integer power Times a Complex array (2-dimensional)
PRDRTC2	PRoduct of a Double precision Row array to an integer power Times a Complex array (2-dimensional)

PRDTC2	PRoduct of a Double precision 2-dimensional array to an integer power Times a Complex array (2-dimensional)
PRIMSUM	PeRIMeter SUM around a nested, successively larger squares within a 2-dimensional real array
PRNCORR	PRiNt the amplitude and phase of a Column OR a Row of a complex array
PRNFUNC	PRiNt the parameters of the FUNCtion used to create an displacement error file
PRNPLT	PRiNt and create 4 .PLT files the amplitude and phase of a column and a row of a complex array
PRNRCOR	PRiNt a Real-array's specified Column OR Row
PRNRICR	PRiNt the Real and Imaginary parts of a Column and/or Row of a 2-dimensional complex array
PRNTC1D	PRiNT a specified column and the maximum amplitude of a Complex array
PRNTR1D	PRiNT a specified column and the maximum and minimum values of a Real array
RADDRC2	Real array ADDed to a Real Constant (2-dimensional)
RANERB2	copy a Real array RA to RB when the two arrays may have unequal column lengths
RANGED	obtain the RANGE of values in a Double-precision array between two specified indeces (1-dimensional)
RANGES	obtain the RANGE of values in a Single-precision array between two specified indeces (1-dimensional)
RARYMM2	get Real ARraY's Maximum and Minimum values (2-dimensional)
RATORB1	copy a Real array RA TO RB (1-dimensional)
RATORB2D	copy a Real array RA TO a Double-precision array RB (2-dimensional)
RCA2B1	copy a Column of data from a specified row of a real 2-dimensional array to a 1-dimensional array
RCBD2	read a Real binary dataset and store in alternatimg locations in a real array (2-dimensional)
RCBDIOF	get the Input Output Filename to read 2 Real Binary datasets into a Complex array
RCBDPAR	Read PARameters and filenames to input 2 Real Binary datasets into a Complex array
RCBDSET	SET to read two Real Binary datasets into a Complex array
RDCBD1	ReaD a Complex Binary Dataset (1-dimensional)
RDCBD2	Read a Complex Binary Dataset (2-dimensional)
RDCBD3	Read a Complex Binary Dataset (3-dimensional)
RDDABP	ReaD the Direct-Access Binary Dataset's Parameters
RDDACBD	ReaD a Direct-Access Complex Binary Dataset
RDFUNC	ReaD the names of specific FUNCtions used to create current error-array file
RDIF2	calculate the DIFference between two Real arrays (2-dimensional)
RDOT	DOT product of Real arrays (2-dimensional)
RDRBD2	ReaD a Real Binary Dataset (2-dimensional)

RDRBD2D	ReaD a Real Binary Dataset in Double precision (2-dimensional)
REARANG	amplitude, phase, distance correction and swap to obtain far-field data
RINIT1	INITialization with a Real constant (1-dimensional)
RINIT2	INITialization with a Real constant (2-dimensional)
RKCARBK2	weighted complex sum of a 2-dimensional Complex Array times a Real Array raised to an integer power (2-dimensional)
RMSQR	Convert the square root of an array's elements into decibels and return the number of non-zero elements (1-dimensional)
RMULT2	Real MULTiplication of two arrays (2-dimensional)
RNDM	function call to return a RANDoM number
RNDMDZ	create a RaNDoM array DZ
RNDMFCT	create a RaNDoM FunCTion and store in an real array (2 dimensinal)
RRA2B1	copy a Row of data from a apecified column of a real 2-dimensional array to a 1-dimensional array
RRATIO2	calculate the RATIO of the elements of two Real arrays (2-dimensional)
RSUMCOL	Sum a COLumn of elements of a Real array (2-dimensional)
RSUMROW	Sum a ROW of elements of a Real array (2-dimensional)
RZTORC1	Real array TO a Real Constant power (1-dimensional)
RZTORC2	Real array TO a Real Constant power (2-dimensional)
SCALE	an array of consecutive integers multipled by a real constant (1-dimensional)
SCLCC1	SCaLe a Complex array by a Complex Constant (1-dimensional)
SCLCC2	SCaLe a Complex array by a Complex Constant (2-dimensional)
SCLRC1	SCaLe a Complex array by a Real Constant (1-dimensional)
SCLRC1D	SCaLe a Complex array by a Real Double precision Constant (1-dimensional)
SCLRR2	SCaLe a Real array by a Real Constant (1-dimensional)
SCLRR2D	SCaLe a Real Double precision array by a Double precision constant (2-dimensional)
SETBNDR	SET the BouNDaRy of a complex array equal to a complex constant
SETFILS	SET up to read a list of FILenameS
SETFIOF	get the Input Output File for reading Filenames
SETFPAR	read the output Filenames
SETTSX	SET up the necessary arrays for a Taylor Series in X calculation
SETTSXY	SET up the necessary arrays for calculating a Taylor Series along both the X and Y coordinates
SETTSY	SET up the necessary arrays for a Taylor Series in Y calculation
SETTSZ	SET up the necessary arrays for Taylor Series in Z calculations
SFTCACB	ShiFT the location of Complex data in zero padded array to array center
SFTRARB	ShiFT the location of Real data in zero padded array to array center
SIN4X	calculate the function SINe of X raised to the fourth power
STRNGLN	find the locations of the first and last non-blank characters in a string
SUMPRM	SUM the values on the PeRiMeter of a square embedded within a real array (2-dimensional)
SUMPRM1	SUM the values on the PeRiMeter of a square embedded within a real array, omitting the corners from the sum (2-dimensional)

SUMSUMS	an array of SUMs of a real array's elements from successively increasing array locations to the end (1-dimensional)
SWAP	SWitch begining to end Array-element Positions of both rows and columns
SWCRIP2	Weighted Complex Sum of 3-dimensional Complex array times a Real array raised to an array of Integer Powers (2-dimensional)
TIMER	store system TIME on first call, return time difference on second call
TIMERS	multiple TIME initilizations, time differences returned on second call
TODAY	write current date to screen
TSCOEF	calculate and store the Taylor Series COEFficients and an array of integer powers
TSXSLM	Taylor Series in X Summed from Low to Maximum order
TSXYSN	Taylor Series in X and Y Summed to order N
TSYSLM	Taylor Series in Y Summed from Low to Maximum order
TSZK	Taylor Series in Z term of order K
TSZK1	Taylor Series in Z term of order K and output partial sum
TSZSLM	Taylor Series in Z Summed from Low to Maximum order
TSZSLM0	Taylor Series in Z Summed from Low to Maximum order after initialization
TSZSLM1	Taylor Series in Z Summed from Low to Maximum order, and output each partial sum
UDASCUN	UpDate ASCII output file Unit Number
UDDSIUN	UpDate Direct-Sum output file Unit Number
UDDXIUN	UpDate DX error array's output file Unit Number
UDDYIUN	UpDate DY error array's output file Unit Number
UDDZIUN	UpDate DZ error array's output file Unit Number
UDFFIUN	UpDate Far Field's output file Unit Number
UDNFIUN	UpDate Near Field's output file Unit Number
WCBD1	Write an unformatted Complex Binary Dataset (1-dimensional)
WCBD2	Write unformatted Complex Binary Dataset to *fort.xx* file (2-dimensional)
WCBD3	Write unformatted Complex Binary Dataset to *fort.xx* file (3-dimensional)
WDACBD	Write a self-documented Direct-Access Complex Binary Dataset
WDSCBD	Write unfromatted Complex Binary Dataset to *dsnf.xx* file (2-dimensional)
WLTOCM	convert a WaveLength TO CentiMeters
WLTOCMD	convert WaveLengths TO Centimeters in double precision
WRBD2	Write an unformatted Real Binary Dataset to a *fort.xx* file
WRBD2D	Write an unformatted Double-precision Real Binary Dataset to a *fort.xx* file (2-dimensional)
WRCHKF	WRite a CHecK list of parameters to a print File
WRFUNC	WRite the specified FUNCtion name to specified file
XCHAR	eXpress an integer modulus 100 as a CHARacter variable
XSCHAR	eXpress an integer modulus 10 as a Single CHARacter variable
XYGRIDS	create X and Y GRIDs in Single precision

Appendix A

Creating the Original Direct Access Binary Dataset

Two modules, UAPDACB and UDBPDACB, are provided for inputing ASCII data files to create direct-access complex binary datasets.

Module UAPDACB reads two real ASCII data files, one containing amplitude data, and one containing phase data (in degrees). The data in each ASCII file are interpreted as successive columns of data, with each column having a constant X coordinate. The names of the two ASCII data files, and their data formats, are specified in a user-supplied parameter file, whose filename is recorded in file ADAB.IOF. The contents of this parameter file are defined in the following table:

List of User-Supplied Parameters in the File Named by ADAB.IOF

FFORNF	data type specifier specifying either *Far-Field* OR *Near-Field* data
LABEL	character variable used only for identification
FILE1, FILE2	Filenames of the two ASCII input files
FILE3	Filename of the direct-access complex binary dataset that is created
FORM	Fortran data-format specification for the ASCII input files
NY, NX	the number of respective rows and columns in the ASCII data files
DY, DX	data point spacing in near-field datasets along the Y and X axes
FREQ	operating frequency [GHz]
Z0	z coordinate [cm] location of the near-field measurement plane

The three files specified in this parameter file are located in a directory whose path is given in a local file named DATA.DIR. The two input ASCII data files are read by subroutine FRRAD, which is called by module UAPDACB. Alternatively, subroutine FRRADHD, which assumes that a 120 character HeaDer preceedes the data, can be used.

The module UDBPDACB also inputs two ASCII data files to create a direct-access complex binary dataset, but it assumes that one ASCII file contains amplitude data expressed in decibels and one contains phase data expressed in degrees. The data in each file are interpreted as successive rows of data, with each row having a constant Y coordinate. Both files are assumed to have been setup as .GRD files suitable for input to the system plot package. As before, these files and their associated parameters are specified in a user-supplied parameter file whose filename is recorded in ADAB.IOF.

The input data are written to a direct-access complex binary dataset as successive records each consisting of one entire column of data. The first seven records in each original dataset contain essential parameters of the dataset. The first record gives the file record LENGTH, which is numerically set equal to 8*NY. The next six records are the entries in the above table (except those printed in italics) and are written in the order listed.

Appendix B

System Initialization

At the beginning of any research project the system has to be initialized to properly set the the system parameters and unit numbers. This is accomplished by executing module UINITUN, which will write the following table to the screen:

THE INITIAL SETTINGS are:

filter.ff: cksqrd=:	0.0000000E+00	
ffnf.dz: dzinc=:	0.0000000E+00	
order.drv: idrvinc,iorder=:	1	1
ampordb.grd: ampordb=:	dB	
fun.dx: funtype=:	per	
fun.dy: funtype=:	per	
fun.dz: funtype=:	per	
active.iun: iactive=:	0	
add.iun: iadd=:	0	
amp2.iun: iunamp2=:	0	
asci.iun: iunasci=:	7	
difdiv.iun: idifdiv=:	1	
dif2.iun: iundif2=:	0	
dif.iun: iundif=:	0	
difdb.iun: iundfdb=:	0	
div.iun: iundiv=:	0	
drv.iun: iundrv=:	0	
dx.iun: iundx=:	61	
dy.iun: iundy=:	62	
dz.iun: iundz=:	63	
ecx.iun: iunecx=:	0	
ecy.iun: iunecy=:	0	
ecz.iun: iunecz=:	0	
ff.iun: iunff=:	60	
kec.iun: iunkec=:	0	
nf.iun: iunnf=:	40	
rdiv.iun: iunrdiv=:	0	
tsx.iun: iuntsx=:	0	
tsxy.iun: iuntsxy=:	0	
tsy.iun: iuntsy=:	0	
tsz.iun: iuntsz=:	0	
tsamp.iun: iuntsa=:	0	
tsphs.iun: iuntsp=:	0	
tstz.iun: iuntst=:	0	

STOP: UINITUN: normal termination

On each line in the above table the first entries give the name of the file where the information is recorded, while the second entries give the name(s) of the fortran variable(s) that contain the value(s), which are shown last. The key abbreviations in the filenames and variable names can be deciphered by consulting Table 1. For example, *iunasci* specifies the *current* setting of the ascii output unit number, and *iundz* specifies the unit number of the *dz* dataset. Many of the unit numbers are set to 0, simply signifying that no data has yet been created for these fields. There are a few remaining variables included in the table that have special meanings. These are defined below:

ampordb	set to *dB* to create .GRD files in decibels; alternatively can be set to *amp*
cksqrd	filter limit for truncating plane-wave spectrum
dzinc	incremental distance to be added to the measurement-plane distance when calculating the near field in module URDFFNF
idrivinc	increment by which *iorder* is increased whenever *order.drv* is accessed by module UDERIV
iorder	order of the derivative to be calculated by module UDERIV, after which its value is incremented by *idrivinc*
funtype	TYPE of FUNction used by any of the modules UMAKEDX, UMAKEDY, or UMAKEDZ to create error fields. Permitted values are *per* (periodic), *poly* (polynomial), or *ran* (random) function

Appendix C

System Status Reports

After the execution of any module one can request a status report for the system to examine the system parameter settings and the unit number settings. This is accomplished by executing USHOWUN. One might do this to check the sequence of executions for correctness and to decide what data management steps one needs to take to access the next dataset needed to continue the research correctly. When USHOWUN is executed after UMAKEDZ and URDNFFF have been executed only once, the following table is displayed:

THE CURRENT SETTINGS are:

ampordb.grd:	dB	
filter.ff:	0.0000000E+00	
order.drv:	1	1
scale.dx:	0.0000000E+00	0.1000000
scale.dy:	0.0000000E+00	0.1000000
scale.dz:	0.0000000E+00	0.2000000
ampff,invff:	1987.822	5.0306314E-04
ampnf,invnf:	1.059250	9.4406420E-01
ffnf.dz:	0.0000000E+00	
ffornf:	ff	
fun.dx:	per	
fun.dy:	per	
fun.dz:	per	
active.iun:	0	
add.iun:	0	
amp2.iun:	0	
asci.iun:	7	8
inc difdiv:	1	
dif2.iun:	0	
dif.iun:	0	
difdb.iun:	0	
div.iun:	0	
drv.iun:	0	
ds.iun:	-1	0
dx.iun:	0	61
dy.iun:	0	62
dz.iun:	63	63
ecx.iun:	0	
ecy.iun:	0	
ecz.iun:	0	
ff.iun:	60	60

kec.iun:	0	
nf.iun:	40	40
rdiv.iun:	0	
tsx.iun:	0	
tsxy.iun:	0	
tsy.iun:	0	
tsz.iun:	0	
tsamp.iun:	0	
tsphs.iun:	0	
tstz.iun:	0	

USHOWUN: unit status report complete

Most of features and entries in the above table have been explained in Appendix A. Here, however, some of the entries show two unit numbers. The combinations of two equal unit numbers signifies that the modules writing these unit numbers have only been executed once, thereby making the initial unit numbers, as defined in Appendix A, the *current* unit numbers. In the case of DS.IUN the initial values are shown, indicating that none of the *direct sum* utility modules have been executed. Initialization of file DS.IUN is the responsibility of the user.

After creating all the datasets required by the error correction research problem (see Section 4) and after executing UDSZ, USHOWUN can be executed to get an overview of the system status. The output table appears as below:

THE CURRENT SETTINGS are:

ampordb.grd:	dB	
filter.ff:	0.0000000E+00	
order.drv:	1	1
scale.dx:	0.0000000E+00	0.1000000
scale.dy:	0.0000000E+00	0.1000000
scale.dz:	0.0000000E+00	0.2000000
ampff,invff:	1987.822	5.0306314E-04
ampnf,invnf:	1.059250	9.4406420E-01
ffnf.dz:	0.0000000E+00	
ffornf:	ff	
fun.dx:	per	
fun.dy:	per	
fun.dz:	per	
active.iun:	58	
add.iun:	60	
amp2.iun:	0	
asci.iun:	7	17
inc difdiv:	1	
dif2.iun:	0	
dif.iun:	0	

difdb.iun:	0	
div.iun:	0	
drv.iun:	0	
ds.iun:	0	0
dx.iun:	0	61
dy.iun:	0	62
dz.iun:	63	63
ecx.iun:	0	
ecy.iun:	0	
ecz.iun:	42	
ff.iun:	60	56
kec.iun:	0	
nf.iun:	40	44
rdiv.iun:	0	
tsx.iun:	0	
tsxy.iun:	0	
tsy.iun:	0	
tsz.iun:	41	
tsamp.iun:	0	
tsphs.iun:	0	
tstz.iun:	0	

USHOWUN: unit status report complete

Now we see that two unequal unit numbers appear in some of the entries. These indicate the range of unit numbers for the particular type of field, (*ff* or *nf*), that *exist* after repeated executions of the various modules. The first unit number indicates the initial unit number created and the last number indicates the *current* value of the unit number. The dataset referred to by the *current* value of the unit number will be automatically accessed if the value in ACTIVE.IUN is 0. In addition, all special types of near-field datasets that have been created during the course of the research are recorded in their respective unit number files. For example, the entry under TSZ.IUN is 41, meaning that the dataset with filename *fort.41* contains the error-contaminated near-field dataset that was created using the Taylor series method. The datasets indicated by DS.IUN are stored separately from this scheme. Thus, the entry indicates that file *dsnf.00* has been stored in a separate directory, whose path is specified in file DATA.DIR.

Appendix D

The Research Modules and their Subroutine Dependencies

UAMP2CBD:	inpffp inpnfp rdcbd2 csqr1 wrbd2 udffiun udnfiun hstamp2 udascun
UAPDACB:	cadacb
UCBDDAT:	ffornf inpgrdp getwn chlngth gtprnpr udascun xygrids rdcbd2 insloss criap db1 cnimcc1 ppfcrap outpfs0 xchar xschar
UCBDGRD:	inpgrdp getwn chlngth xygrids ranges rdcbd2 insloss udascun outgrd
UDBPDACB:	grddacb
UDERIV:	chlngth inpdrvp rdcbd2 getwn settsz dnfdz wcbd2 udnfiun hstdrv udascun prncorr xygrids ranges outgrd
UDIF2CBD:	inpffp inpnfp rdcbd2 csqr1 rdif2 wrbd2 udffiun udnfiun hstdif2 udascun
UDIFACBD:	inpffp inpnfp rdcbd2 cabs1 rdif2 wrbd2 udffiun udnfiun hstdifa udascun
UDIFCBD:	inpffp inpnfp rdcbd2 cdif2 wcbd2 udffiun udnfiun hstdif udascun
UDIFDB:	inpffp getwn inpnfp cbdtodb cdif2 wcbd2 udffiun udnfiun hstdfdb udascun xygrids ranges outgrd
UDIVCBD:	inpffp inpnfp rdcbd2 cinit1 cratio2 wcbd2 udffiun udnfiun hstdiv udascun
UDIVRBD:	inpffp inpnfp rdrbd2 rinit1 rratio2 wrbd2 udffiun udnfiun hstrdiv udascun
UDSX:	uddsiun inpffp0 inpffp rdcbd2 getwn mkgamma cinit1 inpnfp0 inpnfp nfff chkleqi catocb2 ffnf gtprnpr chlngth udascun xchar prnfunc wltocm prncorr rdrbd2 sclrr2 ffnfx wdscbd hstdsx
UDSXY:	uddsiun inpffp0 inpffp rdcbd2 getwn mkgamma cinit1 inpnfp0 inpnfp nfff chkleqi catocb2 ffnf gtprnpr chlngth udascun xchar prnfunc wltocm prncorr rdrbd2 sclrr2 ffnfxy wdscbd hstdsxy
UDSXYZ:	uddsiun inpffp0 inpffp rdcbd2 getwn mkgamma cinit1 inpnfp0 inpnfp nfff chkleqi catocb2 ffnf gtprnpr chlngth udascun xchar prnfunc wltocm prncorr rdrbd2 sclrr2 raddrc2 ffnfxyz wdscbd hstds3
UDSY:	uddsiun inpffp0 inpffp rdcbd2 getwn mkgamma cinit1 inpnfp0 inpnfp nfff chkleqi catocb2 ffnf gtprnpr chlngth udascun xchar prnfunc wltocm prncorr rdrbd2 sclrr2 ffnfy wdscbd hstdsy
UDSZ:	uddsiun inpffp0 inpffp rdcbd2 getwn mkgamma cinit1 inpnfp0 inpnfp nfff chkleqi catocb2 ffnf gtprnpr chlngth udascun xchar prnfunc wltocm prncorr rdrbd2 sclrr2 raddrc2 ffnfz wdscbd hstds
UECX4:	inptsp getwn udascun prnfunc wltocm rdrbd2 sclrr2 rdcbd2 ecx4 gtprnpr prncorr wcbd2 udnfiun hstec
UECY4:	inptsp getwn udascun prnfunc wltocm rdrbd2 sclrr2 rdcbd2 ecy4 gtprnpr prncorr wcbd2 udnfiun hstec
UECZ2:	inptsp wltocm getwn rdrbd2 sclrr2 rdcbd2 settsz mdnfdz cmultr2 cadd2 dnfdzo sclrc1 catocb2 csumik2 udascun prncorr wcbd2 udnfiun hstec
UECZ3:	inptsp wltocm getwn rdrbd2 sclrr2 rdcbd2 settsz mdnfdz cmultr2 cadd2 dnfdzo sclrc1 dnfdze csumik2 udascun prncorr wcbd2 udnfiun hstec
UECZ4:	inptsp getwn udascun prnfunc wltocm rdrbd2 sclrr2 rdcbd2 ecz4 gtprnpr prncorr wcbd2 udnfiun hstec
UKCORR:	inptsp wltocm getwn rdrbd2 sclrr2 rdcbd2 caceiph wcbd2 udnfiun udascun hstkec

ULAPLCN:	chlngth cinit1 inpnfp0 inpnfp rdcbd2 getwn mkgamma udascun caraymx2 prncorr laplcan catocb2 sclrc1 xygrids ranges pccrgrd cadd2
UMAKEDX:	crfunc inpdzp rinit1 wltocm mkpolyn funcscl mkperfn perfunx rndmfct norm2 iyjxcnt sftrarb grid ranges wrbd2 uddxiun hstmkdx udascun rdfunc rarymm2 outrgrd
UMAKEDY:	crfunc inpdzp rinit1 wltocm mkpolyn funcscl mkperfn perfuny rndmfct norm2 iyjxcnt sftrarb grid ranges wrbd2 uddyiun hstmkdy udascun rdfunc rarymm2 outrgrd
UMAKEDZ:	crfunc inpdzp rinit1 wltocm mkpolyn funcscl mkperfn perfunz rndmfct norm2 iyjxcnt sftrarb grid ranges wrbd2 uddziun hstmkdz udascun rdfunc rarymm2 outrgrd
UOPNORM:	inpffp inpnfp rdcbd2 casumsq hstnrm chlngth udascun
UPRDBCBD:	inpffp inpnfp rdcbd2 insloss criap db2 udascun xchar prnricr
UPRNCBD:	inpffp inpnfp rdcbd2 udascun xchar prncorr
UPRRICBD:	inpffp inpnfp rdcbd2 udascun xchar prnricr
URBDDAT:	inpgrdp getwn chlngth xchar rdrbd2 chkleqi xygrids ranges udascun rarymm2 prnrcor pltfile
URBDGRD:	udascun inpgrdp getwn chlngth xygrids ranges rdrbd2 mmra outrgrd
URDFFNF:	inpffp0 inpffp rdcbd2 wltocm getwn mkgamma catocb2 ffnf wcbd2 udffiun udnfiun udascun caraymx2
URDNFFF:	cinit1 inpnfp0 inpnfp rdcbd2 catocb2 getwn mkgamma nfff iyjxcnt sftcacb wcbd2 udnfiun udffiun udascun caraymx2
URMSCBD:	chlngth inpffp0 wcbd2 udffiun inpffp rdcbd2 chkleqi getwn udascun caraymx2 csqr1 primsum sumsums rmsqr plrdata hstrms
USUBGRD:	inpgrdp getwn chlngth xygrids ranges rdcbd2 insloss udascun outgrd
UTSNFAP:	inpnfp rdcbd2 criap critoc2 capri wcbd2 udnfiun udascun hsttsap
UTSTZ:	udascun inptsp wltocm rdcbd2 rdrbd2 sclrr2 getwn catocb2 settsz cinit1 tszslm xchar prncorr wcbd2 udnfiun hsttst
UTSX:	inptsp chkleqi getwn inpffp0 inpffp rdcbd2 mkgamma ffnf cinit1 inpnfp0 iyjxcnt sftcacb udascun prnfunc wltocm gtprnpr rdrbd2 sclrr2 prncorr coeffts settsx nftsxk cadd2 xchar wcbd2 udnfiun hsttsx
UTSXY:	inptsp chkleqi getwn inpffp0 inpffp rdcbd2 mkgamma ffnf cinit1 inpnfp0 iyjxcnt sftcacb udascun prnfunc wltocm rdrbd2 sclrr2 coeffts settsxy nftsxy wcbd2 udnfiun hsttsxy
UTSY:	inptsp chkleqi getwn inpffp0 inpffp rdcbd2 mkgamma ffnf cinit1 inpnfp0 iyjxcnt sftcacb udascun prnfunc wltocm gtprnpr rdrbd2 sclrr2 prncorr coeffts settsy nftsyk cadd2 xchar wcbd2 udnfiun hsttsy
UTSZ:	inptsp chkleqi getwn inpffp0 inpffp rdcbd2 mkgamma ffnf cinit1 inpnfp0 iyjxcnt sftcacb udascun prnfunc wltocm gtprnpr rdrbd2 sclrr2 prncorr coeffts settsz nftszk cadd2 xchar wcbd2 udnfiun hstts

Appendix E

The PNFC Subroutines and Their Subroutine Dependencies

ACPCFFD:	
ACPCNFD:	
ACTIUN:	
ADABIOF:	chlngth
ADABPAR:	errmess
ADDBOX:	
AMPDIF2:	
APDSET1:	cca2b1 cra2b1 criap1 capri1 cdif1
APNAME:	strngln
CABD:	cabdiof cabdpar frbw
CABDIOF:	chlngth
CABDPAR:	errmess
CABS1:	
CACBK3D:	
CACEIPH:	
CADACB:	adabiof adabpar frradhd capri wdacbd
CADD1:	
CADD2:	
CADD3:	
CADDCC2:	
CAEIPH2:	
CAEIPHC:	
CANECB2:	
CAPCCD1:	
CAPRI:	
CAPRI1:	
CAPRI2D:	
CAPRNT:	mmrca prntr1d mmica
CARAYMX2:	
CARBK3D:	
CARBKD3:	
CARCBD2:	
CASUMSQ:	
CATOCB1:	
CATOCB2:	
CBDTODB:	rdcbd2 criap1 cnimcc1 db1
CCA2B1:	
CDFSET1:	cca2b1 cra2b1 cdif1 criap1 capri1
CDIF1:	
CDIF2:	
CDIVDS2:	
CDOT:	

CGATHER:	
CHKEQFP:	
CHKEQI:	
CHKLEQI:	
CHKPAR0:	
CHKPAR1:	
CHKPAR2:	
CHLAST:	
CHLNGTH:	
CIMGSTR:	
CINIT1:	
CINIT2:	
CIRCFLT:	
CMULDS2:	
CMULRD2:	
CMULT1:	
CMULT2:	
CMULTR2:	
CNIMCC1:	
CNTCACB:	iyjxcnt sftcacb
COEFFTS:	
const: [1]	
constax: [1]	
cos2: [1]	
cos3: [1]	
cos4: [1]	
cosax: [1]	
cosax2: [1]	
cosx: [1]	
CRA2B1:	
CRATIO2:	
CRFUNC:	chlngth
CRIAP:	
CRIAP1:	
CRIAP2D:	
CRITOC2:	
CRNFERR:	nfpff catocb2 ffnf
CSMWCP1:	
CSMWCP2:	
CSQR1:	
CSUM1:	
CSUM2:	
CSUMCP1:	
CSUMCP2:	

CSUMIK2:	
CSUMRW1:	
CSUMRW2:	
CSWCPE2:	
CUTOFF1:	
CUTOFF2:	
CXPCLOG:	
DABDIOF:	chlngth
DABDPAR:	chkpar0 chkpar1
DATFILE:	
DATORB1:	
DB1:	
DB2:	
DCLN2:	
DNFDX:	catocb2 prdrtc2 sclcc1 fourt acpcnfd swap
DNFDXDY:	prdctc2 prdrtc2 sclcc1 fourt acpcnfd swap
DNFDY:	catocb2 prdctc2 sclcc1 fourt acpcnfd swap
DNFDZ:	dnfdze cmulds2
DNFDZE:	catocb2 prdtc2 fourt acpcnfd swap sclrc1
DNFDZO:	migammz cmulds2 dnfdze
DSPWS:	
DSPWSX:	
DSPWSXY:	
DSPWSY:	
DSWCRP1:	
ECEXP:	
ECX4:	settsx mdnfdx cmultr2 sclrc1 catocb2 cadd2 csumik2
ECY4:	settsy mdnfdy cmultr2 sclrc1 catocb2 cadd2 csumik2
ECZ4:	settsz mdnfdz cmultr2 dnfdzo dnfdze sclrc1 catocb2 cadd2 csumik2
EIGAMAZ:	
ERRMESS:	
FAXSBYS:	fx fy (unspecified functions)
FFLIMTS:	
FFNF:	ffpff pffnf
FFNFIUN:	strngln
FFNFX:	migammz dcln2 ppwsnfx sclcc2
FFNFXY:	migammz dcln2 ppwsnf2 sclcc2
FFNFXYZ:	migammz dcln2 ppwsnf3 sclcc2
FFNFY:	migammz dcln2 ppwsnfy sclcc2
FFNFZ:	migammz dcln2 ppwsnfz sclcc2
FFORNF:	
FFPFF:	eigamaz fltlimt flteigz cmulds2
FFRANGE:	fflimts
FFTFFT:	sclrr2 fourt

FILSIOF:	
FILSPAR:	
FINDEND:	
FLTEIGZ:	
FLTLIMT:	
FLTPWSG:	fltlimt
FLTRHIK:	nfff ffnf
FOURT:	
FRBW:	
FRGRD:	
FRRAD:	
FRRADHD:	
FUNAXBY:	fx fy (unspecified functions)
FUNCSCL:	
FUNCXY:	fx fy (unspecified functions)
FWDCRA1:	datorb1 fwrad1
FWRAD1:	
FWRAD2:	
GAMMASQ:	
GETFILE:	
GETWN:	
GETWND:	
GRDDACB:	adabiof adabpar frgrd capri wdacbd
GRID:	
GRIDD:	
GTPRNPR:	ffornf chkleqi
HSTAMP2:	findend
HSTDFDB:	findend
HSTDIF:	findend
HSTDIF2:	findend
HSTDIFA:	findend
HSTDIV:	findend
HSTDRV:	findend
HSTDS:	findend
HSTDS3:	findend
HSTDSX:	findend
HSTDSXY:	findend
HSTDSY:	findend
HSTEC:	findend
HSTFFNF:	findend
HSTKEC:	findend
HSTMKDX:	findend
HSTMKDY:	findend
HSTMKDZ:	findend

HSTNFFF:	findend
HSTNRM:	findend
HSTRDIV:	findend
HSTRMS:	findend
HSTTS:	findend
HSTTSAP:	findend
HSTTST:	findend
HSTTSX:	findend
HSTTSXY:	findend
HSTTSY:	findend
HSTUN0:	findend
HSTUN1:	findend
INPDABP:	dabdiof dabdpar rddabp getwn wrchkf
INPDACB:	dabdiof dabdpar rddacbd getwn wrchkf
INPDRVP:	dabdiof dabdpar rddabp
INPDZP:	dabdiof dabdpar rddabp
INPFFP:	dabdiof dabdpar rddabp
INPFFP0:	dabdiof dabdpar rddacbd
INPFILS:	filsiof filspar
INPGRDP:	dabdiof dabdpar rddabp
INPNFP:	dabdiof dabdpar rddabp
INPNFP0:	dabdiof dabdpar rddacbd
INPRCBD:	rcbdiof rcbdpar rcbdset
INPTSP:	dabdiof dabdpar rddabp
INSLOSS:	sclcc1
INTNF3:	migammz fourt swap cdivds2 acpcffd acpcnfd
iunit: [1]	
IUNS:	ffnfiun
IYJXCNT:	
LAPLCAN:	nfpff dnfdze dnfdx cadd2 dnfdy pffnf
MAKEDZ:	grid funaxby sclrr2 (fx fy = unspecified externals)
MAKPFF:	cinit1 addbox nfpff
MDARB1:	datorb1
MDNFDX:	dnfdx
MDNFDY:	dnfdy
MDNFDZ:	mdnfdze mdnfdzo
MDNFDZE:	dnfdze
MDNFDZO:	cmulds2 mdnfdze
MIGAMMZ:	
MKGAMMA:	gridd gammasq
MKPERDZ:	grid faxsbys rarymm2 sclrr2 (fx fy = unspecified externals)
MKPERFN:	grid faxsbys (fx fy = unspecified externals)
MKPLYDZ:	grid polynxy rarymm2 sclrr2
MKPOLYN:	grid polynxy rztorc1

MMICA:	
MMRA:	
MMRCA:	
NF:	fx fy (unspecified functions)
NFFF:	nfpff pffff
NFMODX:	f (unspecified function)
NFMODY:	f (unspecified function)
NFPFF:	setbndr fourt swap acpcffd
NFTSXY:	gtprnpr prncorr nftsyk cadd2 nftsxk nftsxyk
NFTSXK:	dnfdx rkcarbk2
NFTSXYK:	dnfdxdy rkcarbk2
NFTSYK:	dnfdy rkcarbk2
NFTSZK:	dnfdzo dnfdze rkcarbk2
NFZKTS:	catocb2 ffnf tszSLM0
NORM2:	rarymm2 sclrr2
OUTASC:	criap fwrad2
OUTDACB:	setfils wdacbd
OUTDPS:	prncorr catocb2 wcbd2
OUTGRD:	criap db1 cnimcc1 mmrca getfile outrgrd mmica
OUTPFS0:	getfile pltfile
OUTPFS1:	pfcrap outpfs0
OUTRGRD:	
PCCRGRD:	caraymx2 prncorr outgrd
perfunc: [1]	cosax (or cosax2, etc.)
perfunx: [1]	wrfunc cosax (or cosax2, etc.)
perfuny: [1]	wrfunc cosax (or cosax2, etc.)
perfunz: [1]	wrfunc cosax (or cosax2, etc.)
PFCORR:	pfset pfreim
PFCRAP:	pfcorr pfreim
PFFFF:	migammz cxpclog ecexp
PFFNF:	fourt acpcnfd swap
PFREAL:	
PFREIM:	
PFSET:	cca2b1 cra2b1 criap1 capri1
PLRDATA:	scale pfreal pltfile
PLTFILE:	
polyn: [1]	
POLYNXY:	polyn
PPFCRAP:	prncorr prnricr pfcrap
PPWSNF2:	ffrange fltpwsg carcbd2 dspwsxy
PPWSNF3:	ffrange fltpwsg carcbd2 dspwsxy
PPWSNFX:	ffrange fltpwsg carcbd2 dspwsx
PPWSNFY:	ffrange fltpwsg carcbd2 dspwsy
PPWSNFZ:	ffrange fltpwsg carcbd2 dspws

PRDCTC2:	
PRDRTC2:	
PRDTC2:	
PRIMSUM:	sumprm sumprm1
PRNCORR:	cca2b1 criap1 cra2b1
PRNFUNC:	chlngth rdfunc
PRNPLT:	prncorr outpfs1
PRNRCOR:	rca2b1 rra2b1
PRNRICR:	cca2b1 cra2b1
PRNTC1D:	caraymx2
PRNTR1D:	
RADDRC2:	
RANERB2:	
RANGED:	
RANGES:	
RARYMM2:	
RATORB1:	
RATORB2D:	
RCA2B1:	
RCBD2:	errmess
RCBDIOF:	chlngth
RCBDPAR:	
RCBDSET:	rcbd2
RDCBD1:	
RDCBD2:	xchar
RDCBD3:	xchar
RDDABP:	chlngth chkpar2 errmess
RDDACBD:	chlngth chkpar2 errmess
RDFUNC:	chlast
RDIF2:	
RDOT:	
RDRBD2:	xchar
RDRBD2D:	xchar
REARANG:	swap
RINIT1:	
RINIT2:	
RKCARBK2:	
RMSQR:	
RMULT2:	
rndm: [1]	
RNDMDZ:	rndm
RNDMFCT:	rndm
RRA2B1:	
RRATIO2:	

RSUMCOL:	
RSUMROW:	
RZTORC1:	
RZTORC2:	
SCALE:	
SCLCC1:	
SCLCC2:	
SCLRC1:	
SCLRC1D:	
SCLRR2:	
SCLRR2D:	
SETBNDR:	cinit1
SETFILS:	setfiof setfpar
SETFIOF:	
SETFPAR:	
SETTSX:	gridd cutoff1 catocb2 nfpff
SETTSXY:	gridd cutoff1 catocb2 nfpff
SETTSY:	gridd cutoff1 catocb2 nfpff
SETTSZ:	mkgamma cutoff1 cutoff2 migammz catocb2 nfpff
SFTCACB:	
SFTRARB:	
sin4x: [1]	
STRNGLN:	
SUMPRM:	rsumcol rsumrow
SUMPRM1:	rsumcol rsumrow
SUMSUMS:	
SWAP:	
SWCRIP2:	
TIMER:	
TIMERS:	
TODAY:	
TSCOEF:	
TSXSLM:	dnfdx swcrip2
TSXYSN:	dnfdxdy swcrip2
TSYSLM:	dnfdy swcrip2
TSZK:	dnfdzo dnfdze swcrip2
TSZK1:	tszK outdps
TSZSLM:	dnfdzo dnfdze swcrip2
TSZSLM0:	settsz tszSLM
TSZSLM1:	dnfdz swcrip2 criap caprnt catocb2 wcbd2
UDASCUN:	
UDDSIUN:	
UDDXIUN:	
UDDYIUN:	

UDDZIUN:	
UDFFIUN:	
UDFIUN:	
UDNFIUN:	
WCBD1:	
WCBD2:	xchar
WCBD3:	xchar
WDACBD:	
WDSCBD:	chlngth xchar
WLTOCM:	
WLTOCMD:	
WRBD2:	xchar
WRBD2D:	xchar
WRCHKF:	wltocm
WRFUNC:	chlngth
xchar: [1]	
xschar: [1]	
XYGRIDS:	grid

[1] Function subprogram name designation

BL-114A (5-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 3970

2. PERFORMING ORGANIZATION REPORT NUMBER
June 1991

3. PUBLICATION DATE

4. TITLE AND SUBTITLE
Personal Computer Codes for Analysis of Planar Near Fields

5. AUTHOR(S)
Lorant A. Muth and Richard L. Lewis

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)
U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
BOULDER, COLORADO 80303-3328

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)
Air Force Guidance and Metrology Center
Newark Air Force Base, Ohio 43057

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

We have developed Fortran codes for analysis of planar near-field data. We describe some of the inner workings of the codes, the data management schemes, and the structure of the input/output sections to enable scientists and programmers to use these codes effectively as a research tool in antenna metrology. The open structure of the codes allows a user to incorporate into the package new applications for future use with relative ease. The subroutines currently in existence are briefly described, and a table showing the interdependence among these subroutines is constructed. Some basic research problems, such as transformation of a near field to the far field and correction of probe position errors, are carried out from start to finish to illustrate use and effectiveness of these codes. Sample outputs are shown. The advantage of a high degree of modularization is demonstrated by the use of DOS batch files to execute Fortran modules in a desired sequence.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)
antenna metrology; computer codes; data management; planar near fields; far fields; research tool; subroutines

13. AVAILABILITY

X	UNLIMITED
	FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
	ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
X	ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
64

15. PRICE

ELECTRONIC FORM

☆U.S. GOVERNMENT PRINTING OFFICE: 1991 - 576-212/45136