NISTIR 3970

PERSONAL COMPUTER CODES FOR ANALYSIS OF PLANAR NEAR FIELDS

Lorant A. Muth
Richard L. Lewis
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
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. General Features</td>
<td>3</td>
</tr>
<tr>
<td>3. Data Management</td>
<td>5</td>
</tr>
<tr>
<td>4. Research Modules</td>
<td>7</td>
</tr>
<tr>
<td>5. Output Files</td>
<td>12</td>
</tr>
<tr>
<td>6. DOS Batch Files</td>
<td>12</td>
</tr>
<tr>
<td>7. Symbol Definitions</td>
<td>14</td>
</tr>
<tr>
<td>8. Subroutine Descriptors</td>
<td>14</td>
</tr>
<tr>
<td>9. Table of Dependencies</td>
<td>14</td>
</tr>
<tr>
<td>10. Conclusion</td>
<td>15</td>
</tr>
<tr>
<td>References</td>
<td>15</td>
</tr>
<tr>
<td>Table 1</td>
<td>16</td>
</tr>
<tr>
<td>Table 2</td>
<td>18</td>
</tr>
<tr>
<td>Table 3</td>
<td>20</td>
</tr>
<tr>
<td>Table 4</td>
<td>22</td>
</tr>
<tr>
<td>Table 5</td>
<td>25</td>
</tr>
<tr>
<td>Table 6</td>
<td>28</td>
</tr>
<tr>
<td>Table 7</td>
<td>32</td>
</tr>
<tr>
<td>Appendix A</td>
<td>43</td>
</tr>
<tr>
<td>Appendix B</td>
<td>44</td>
</tr>
<tr>
<td>Appendix C</td>
<td>46</td>
</tr>
<tr>
<td>Appendix D</td>
<td>49</td>
</tr>
<tr>
<td>Appendix E</td>
<td>51</td>
</tr>
</tbody>
</table>
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.

Revisions

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 UCBDDDAT (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 referring 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 dataset 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 ACTive file gets accessed first, and the ADDitional file gets accessed if the module requires two datasets. The current file gets accessed only if the ACTive file is set to zero, and any existing file can be accessed only if it is made ACTive, ADDitional 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 ACTive 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. Similarly, 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 ACTive, ADDitional 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:

```plaintext
x uunitun
x urdnff
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 uaddnfo
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 uaddnfo
```
x udivecbd

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 UADDF. Thus, to form all ratios we execute the following sequence:

x uswtoff
x uaddff0
x uincff
x uactff
x udecff
x udivecbd
x uincff
x uactff
x udecff
x udivecbd

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 (UADDF0), 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 fornf. 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:

\[
\begin{align*}
&x \text{ uact}0 \\
&x \text{ uswtonf} \\
&x \text{ unorm}1 \\
&x \text{ ucbdgrd} \\
&\text{plt \ nf} \\
&\text{and} \\
&x \text{ unorm}0 \\
&x \text{ uacctsz} \\
&x \text{ ucbdgrd} \\
&\text{plt \ nf} \\
&\text{and} \\
&x \text{ uact}0 \\
&x \text{ uswtoff} \\
&x \text{ unorm}1 \\
&x \text{ ucbdgrd} \\
&\text{plt \ ff}
\end{align*}
\]

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 $fform$ 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 creates 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 errorcor.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 occurrence 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 structure 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


Table 1
Definition of Symbols Used in Naming Modules

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>initial, set to 0</td>
</tr>
<tr>
<td>1</td>
<td>version 1</td>
</tr>
<tr>
<td>2</td>
<td>squared quantity</td>
</tr>
<tr>
<td>act</td>
<td>active, activate</td>
</tr>
<tr>
<td>act0</td>
<td>set ACTive switch to 0</td>
</tr>
<tr>
<td>add</td>
<td>additional</td>
</tr>
<tr>
<td>add0</td>
<td>set ADDitional switch to 0</td>
</tr>
<tr>
<td>amp</td>
<td>amplitude</td>
</tr>
<tr>
<td>ap</td>
<td>amplitude, phase</td>
</tr>
<tr>
<td>cbd</td>
<td>complex binary dataset</td>
</tr>
<tr>
<td>cor</td>
<td>correct, corrected, correction</td>
</tr>
<tr>
<td>db</td>
<td>in decibels</td>
</tr>
<tr>
<td>dif</td>
<td>difference</td>
</tr>
<tr>
<td>div</td>
<td>divide, divided (ratio)</td>
</tr>
<tr>
<td>drv</td>
<td>derivative</td>
</tr>
<tr>
<td>ds</td>
<td>direct sum</td>
</tr>
<tr>
<td>dacb</td>
<td>direct access complex binary (file)</td>
</tr>
<tr>
<td>dat</td>
<td>.DAT (file)</td>
</tr>
<tr>
<td>dbp</td>
<td>dB, phase complex storage</td>
</tr>
<tr>
<td>dc</td>
<td>decrement</td>
</tr>
<tr>
<td>deriv</td>
<td>derivative</td>
</tr>
<tr>
<td>dif2</td>
<td>difference between squared amplitudes</td>
</tr>
<tr>
<td>difa</td>
<td>difference in amplitude</td>
</tr>
<tr>
<td>dz</td>
<td>function dz</td>
</tr>
<tr>
<td>ec</td>
<td>error correction</td>
</tr>
<tr>
<td>err</td>
<td>error</td>
</tr>
<tr>
<td>ff</td>
<td>far field</td>
</tr>
<tr>
<td>ff0</td>
<td>original far field</td>
</tr>
<tr>
<td>grd</td>
<td>.GRD (DOS file extension)</td>
</tr>
<tr>
<td>hst</td>
<td>history</td>
</tr>
<tr>
<td>inc</td>
<td>increment</td>
</tr>
<tr>
<td>init</td>
<td>initialize</td>
</tr>
<tr>
<td>laplcn</td>
<td>Laplacian</td>
</tr>
<tr>
<td>make</td>
<td>make</td>
</tr>
<tr>
<td>nf</td>
<td>near field</td>
</tr>
<tr>
<td>nf0</td>
<td>original near field</td>
</tr>
<tr>
<td>nc</td>
<td>increment</td>
</tr>
<tr>
<td>norm0</td>
<td>normalization of original datasets</td>
</tr>
<tr>
<td>norm1</td>
<td>normalization with 1</td>
</tr>
<tr>
<td>Short</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>op</td>
<td>operator</td>
</tr>
<tr>
<td>prn</td>
<td>print</td>
</tr>
<tr>
<td>rbd</td>
<td>real binary dataset</td>
</tr>
<tr>
<td>rd</td>
<td>read</td>
</tr>
<tr>
<td>rms</td>
<td>root mean square</td>
</tr>
<tr>
<td>show</td>
<td>show</td>
</tr>
<tr>
<td>sw</td>
<td>switch</td>
</tr>
<tr>
<td>to</td>
<td>to</td>
</tr>
<tr>
<td>ts</td>
<td>Taylor series</td>
</tr>
<tr>
<td>u</td>
<td>utility</td>
</tr>
<tr>
<td>un</td>
<td>unit number</td>
</tr>
<tr>
<td>Module</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UAMP2CBD</td>
<td>read a near-field or a far-field dataset and write its squared amplitude to a complex binary data file</td>
</tr>
<tr>
<td>UAPDACB</td>
<td>read an amplitude, phase ASCII file and write a direct-access complex binary file</td>
</tr>
<tr>
<td>UCBDDAT</td>
<td>read a complex binary data file and create a .DAT file for x-y plots</td>
</tr>
<tr>
<td>UCBDGRD</td>
<td>read a complex binary data file and create a two-dimensional .GRD file for contour or surface plotting</td>
</tr>
<tr>
<td>UDBPDACB</td>
<td>read a dB, phase ASCII file and write a direct-access complex binary file</td>
</tr>
<tr>
<td>UDERIV</td>
<td>read a near-field dataset and write the derivative of some specified order</td>
</tr>
<tr>
<td>UDIF2CBD</td>
<td>read two far-field or near-field datasets and write the difference of the squared amplitudes to a CBD file</td>
</tr>
<tr>
<td>UDIFACBD</td>
<td>read two far-field or near-field datasets and write the difference of the amplitudes to a CBD file</td>
</tr>
<tr>
<td>UDIFCBD</td>
<td>read two far-field or near-field datasets and write the complex difference to a CBD file</td>
</tr>
<tr>
<td>UDIFDB</td>
<td>read two far-field or near-field datasets and write the difference of amplitudes in dBs and the phase difference to a CBD file</td>
</tr>
<tr>
<td>UDIVCBD</td>
<td>read two far-field or near-field datasets and write the complex ratio to a CBD file</td>
</tr>
<tr>
<td>UDIVRBD</td>
<td>read two real binary data files and write the ratio to a RBD file</td>
</tr>
<tr>
<td>UDSX</td>
<td>create a near-field dataset containing x-axis position errors using the direct-sum algorithm</td>
</tr>
<tr>
<td>UDSXY</td>
<td>create a near-field dataset containing both x-axis and y-axis position errors using the direct-sum algorithm</td>
</tr>
<tr>
<td>UDSXYZ</td>
<td>create a near-field dataset containing position errors along all three coordinate axes using the direct-sum algorithm</td>
</tr>
<tr>
<td>UDSY</td>
<td>create a near-field dataset containing y-axis position errors using the direct-sum algorithm</td>
</tr>
<tr>
<td>UDSZ</td>
<td>create a near-field dataset containing z-axis position errors using the direct-sum algorithm</td>
</tr>
<tr>
<td>UECX4</td>
<td>read a near-field dataset containing x-axis position errors and perform a fourth-order error correction</td>
</tr>
<tr>
<td>UECY4</td>
<td>read a near-field dataset containing y-axis position errors and perform a fourth-order error correction</td>
</tr>
<tr>
<td>UECZ2</td>
<td>read a near-field dataset containing z-axis position errors and perform a second-order error correction</td>
</tr>
<tr>
<td>UECZ3</td>
<td>read a near-field dataset containing z-axis position errors and perform a third-order error correction</td>
</tr>
</tbody>
</table>
read a near-field dataset containing z-axis position errors and perform a fourth-order error correction

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

read a near-field dataset and form the Laplacian and check that it satisfies the scalar wave equation

create an array DX using a specified error function and write a GRD file to plot the error function

create an array DY using a specified error function and write a GRD file to plot the error function

create an array DZ using a specified error function and write a GRD file to plot the error function

calculate the norm of the error operator

print the amplitude in decibels and the phase in degrees of specified rows and columns of a complex binary data file

print the magnitude and phase of specified rows and columns of a complex binary data file

print the real and imaginary values of specified rows and columns from a complex binary data file

read a real binary data file and create a .DAT file for plotting a specified row and column as x-y plots

read a real binary dataset and create a two-dimensional .GRD file for plotting contour or surface plots

read a far-field dataset and transform it to a near-field dataset

read a near-field dataset and transform it to a far-field dataset

sum the rms values at grid points of a CBD file, and create a .DAT file for plotting

convert a specified part (SUB) of a CBD file to a GRD file for plotting

create two additional near-field data sets by mixing amplitudes and phases from two existing near-field data sets

create an error-field dataset containing z-coordinate position errors using the Taylor series method

introduce x-coordinate position errors into a near-field dataset using the Taylor series method

introduce both x- and y-coordinate position errors into a near-field dataset using the Taylor series method

introduce y-coordinate position errors into a near-field dataset using the Taylor series method

introduce z-coordinate position errors into a near-field dataset using the Taylor series method

print the magnitude and phase of specified rows and columns of a complex binary data file

print the real and imaginary values of specified rows and columns from a complex binary data file

read a real binary data file and create a two-dimensional .GRD file for plotting contour or surface plots

read a far-field dataset and transform it to a near-field dataset

read a near-field dataset and transform it to a far-field dataset

sum the rms values at grid points of a CBD file, and create a .DAT file for plotting

convert a specified part (SUB) of a CBD file to a GRD file for plotting

create two additional near-field data sets by mixing amplitudes and phases from two existing near-field data sets

create an error-field dataset containing z-coordinate position errors using the Taylor series method

introduce x-coordinate position errors into a near-field dataset using the Taylor series method

introduce both x- and y-coordinate position errors into a near-field dataset using the Taylor series method

introduce y-coordinate position errors into a near-field dataset using the Taylor series method

introduce z-coordinate position errors into a near-field dataset using the Taylor series method

19
List of Modules That Perform Basic Data Management Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UACT0</td>
<td>set the active unit number to 0</td>
</tr>
<tr>
<td>UACTADD0</td>
<td>set the active and additional unit numbers to 0</td>
</tr>
<tr>
<td>UACTDDB</td>
<td>set the active unit number to the value in difdb.iun</td>
</tr>
<tr>
<td>UACTDIF</td>
<td>set the active unit number to the value in dif.iun</td>
</tr>
<tr>
<td>UACTDIV</td>
<td>set the active unit number to the value in div.iun</td>
</tr>
<tr>
<td>UACTDRV</td>
<td>set the active unit number to the value in drv.iun</td>
</tr>
<tr>
<td>UACTDS</td>
<td>set the active unit number to the value 0 and write error message</td>
</tr>
<tr>
<td>UACTECX</td>
<td>set the active unit number to the value in ecx.iun</td>
</tr>
<tr>
<td>UACTECY</td>
<td>set the active unit number to the value in ecy.iun</td>
</tr>
<tr>
<td>UACTECZ</td>
<td>set the active unit number to the value in ecz.iun</td>
</tr>
<tr>
<td>UACTFF</td>
<td>set the active unit number to the final value in ff.iun</td>
</tr>
<tr>
<td>UACTFF0</td>
<td>set the active unit number to the initial value in ff.iun</td>
</tr>
<tr>
<td>UACTKEC</td>
<td>set the active unit number to the value in kec.iun</td>
</tr>
<tr>
<td>UACTNF</td>
<td>set the active unit number to the final value in nf.iun</td>
</tr>
<tr>
<td>UACTNF0</td>
<td>set the active unit number to the initial value in nf.iun</td>
</tr>
<tr>
<td>UACTTSX</td>
<td>set the active unit number to the value in tsx.iun</td>
</tr>
<tr>
<td>UACTTSXY</td>
<td>set the active unit number to the value in tsxy.iun</td>
</tr>
<tr>
<td>UACTTSY</td>
<td>set the active unit number to the value in tsy.iun</td>
</tr>
<tr>
<td>UACTTSZ</td>
<td>set the active unit number to the value in tsz.iun</td>
</tr>
<tr>
<td>UADD0</td>
<td>set the additional unit number 0</td>
</tr>
<tr>
<td>UADDDR</td>
<td>set the additional unit number to the value in drv.iun</td>
</tr>
<tr>
<td>UADDDS</td>
<td>set the additional unit number to 0 and write error message</td>
</tr>
<tr>
<td>UADDECX</td>
<td>set the additional unit number to the value in ecx.iun</td>
</tr>
<tr>
<td>UADDECY</td>
<td>set the additional unit number to the value in ecy.iun</td>
</tr>
<tr>
<td>UADDECZ</td>
<td>set the additional unit number to the value in ecz.iun</td>
</tr>
<tr>
<td>UADDEFF</td>
<td>set the additional unit number to the final value in ff.iun</td>
</tr>
<tr>
<td>UADDFF0</td>
<td>set the additional unit number to the initial value in ff.iun</td>
</tr>
<tr>
<td>UADDNF</td>
<td>set the additional unit number to the final value in nf.iun</td>
</tr>
<tr>
<td>UADDNF0</td>
<td>set the additional unit number to the initial value in nf.iun</td>
</tr>
<tr>
<td>UADDTX</td>
<td>set the additional unit number to the value in tsx.iun</td>
</tr>
<tr>
<td>UADTTXY</td>
<td>set the additional unit number to the value in tsxy.iun</td>
</tr>
<tr>
<td>UADDTY</td>
<td>set the additional unit number to the value in tsy.iun</td>
</tr>
<tr>
<td>UADTTZ</td>
<td>set the additional unit number to the value in tsz.iun</td>
</tr>
<tr>
<td>UDCDFDV</td>
<td>decrement the unit number recorded in difdiv.iun</td>
</tr>
<tr>
<td>UDECCF</td>
<td>decrement the value of the current far-field unit number</td>
</tr>
<tr>
<td>UDECNF</td>
<td>decrement the value of the current near-field unit number</td>
</tr>
<tr>
<td>UINCFF</td>
<td>increment the unit number recorded in ff.iun</td>
</tr>
<tr>
<td>UINCNF</td>
<td>increment the unit number recorded in nf.iun</td>
</tr>
<tr>
<td>UINITUN</td>
<td>initialize the system parameters and unit numbers</td>
</tr>
<tr>
<td>UNCDFDV</td>
<td>increment the unit number recorded in difdiv.iun</td>
</tr>
<tr>
<td>UNORM0</td>
<td>set the far-field and near-field normalization constants to their initial values</td>
</tr>
</tbody>
</table>
UNORM1 set the far-field and near-field normalization constants to unity
URESTFF restore the unit numbers in \textit{ff.iun} to the values saved in \textit{save.ffs}
URESTNF restore the unit numbers in \textit{nf.iun} to the values saved in \textit{save.nfs}
USAVEFF save the unit numbers recorded in \textit{ff.iun} in \textit{save.ffs}
USAVENF save the unit numbers recorded in \textit{nf.iun} in \textit{save.nfs}
USCLDX1 set the value of \textit{scale.dx} to the first number in \textit{epsxyz.set}
USCLDX2 set the value of \textit{scale.dx} to the second number in \textit{epsxyz.set}
USCLDX3 set the value of \textit{scale.dx} to the third number in \textit{epsxyz.set}
USCLDY1 set the value of \textit{scale.dy} to the first number in \textit{epsxyz.set}
USCLDY2 set the value of \textit{scale.dy} to the second number in \textit{epsxyz.set}
USCLDY3 set the value of \textit{scale.dy} to the third number in \textit{epsxyz.set}
USCLDZ1 set the value of \textit{scale.dx} to the first number in \textit{epsxyz.set}
USCLDZ2 set the value of \textit{scale.dx} to the second number in \textit{epsxyz.set}
USCLDZ3 set the value of \textit{scale.dx} to the third number in \textit{epsxyz.set}
USETDB1 copy the first set of values of \textit{dbctoff},\textit{dbfloor} in \textit{dbmins.set} to \textit{dbmin.db}
USETDB2 copy the second set of values of \textit{dbctoff},\textit{dbfloor} in \textit{dbmins.set} to \textit{dbmin.db}
USETDB3 copy the third set of values of \textit{dbctoff},\textit{dbfloor} in \textit{dbmins.set} to \textit{dbmin.db}
USHOWUN display on the screen the current system parameter settings
 USW BOTH record the value \textit{ffnf} into \textit{ffornf.iun}
USWFFNF toggle the value recorded in \textit{ffornf.iun} between \textit{ff} and \textit{nf}
USWTOAMP record the value \textit{amp} in \textit{ampordb.grd}
USWTODB record the value \textit{dB} in \textit{ampordb.grd}
USWTOFF record the value \textit{ff} in \textit{ffornf.iun}
USWTONF record the value \textit{nf} in \textit{ffornf.iun}
USWTONON record the value \textit{none} in \textit{ampordb.grd}
### Table 4

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

<table>
<thead>
<tr>
<th>MODULE</th>
<th>PARAMETER FILES</th>
<th>DATA FILES</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAMP2CBD</td>
<td>dabd.iof, difdiv.iun</td>
<td>uamp2cbd.out, amp2.iun, fort.xx^1</td>
</tr>
<tr>
<td>UAPDACB</td>
<td>dabd.iof</td>
<td>[adab.iof]^2</td>
</tr>
<tr>
<td>UCBDDAT</td>
<td>dabd.iof, ampordb.grd, ampnorm.nf, ampnorm.ff, difdiv.iun, dbmin.db, dbloss.grd, iregion.nf, iregion.ff, plot.dir</td>
<td>nfyamp.dat, nfyphase.dat, nfxamp.dat, nfxphase.dat, ffyamp.dat, ffyphase.dat, ucbddat.out</td>
</tr>
<tr>
<td>UCBDGRD</td>
<td>ampordb.grd, ampnorm.nf, ampnorm.ff, dbmin.db, dabd.iof, plot.dir</td>
<td>nfamp.grd, nfphase.grd, ffamp.grd, ffphase.grd, ucbdgd.out</td>
</tr>
<tr>
<td>UDBPDACB</td>
<td>dabd.iof</td>
<td>[adab.iof]^2</td>
</tr>
<tr>
<td>UDERIV</td>
<td>order.drv, dabd.iof, sub.grd, dbmin.db, plot.dir</td>
<td>order.drv, uderiv.out, fort.xx^1, drv.iun, drvamp.grd, drvphase.grd</td>
</tr>
<tr>
<td>UDIF2CBD</td>
<td>difdiv.iun, dabd.iof</td>
<td>udif2cbd.out, dif2.iun, fort.xx^1</td>
</tr>
<tr>
<td>UDIFACBD</td>
<td>difdiv.iun, dabd.iof</td>
<td>udifacbd.out, dif2.iun, fort.xx^1</td>
</tr>
<tr>
<td>UDIFCBD</td>
<td>difdiv.iun, dabd.iof</td>
<td>udifcbd.out, dif.iun, fort.xx^1</td>
</tr>
<tr>
<td>UDIFDB</td>
<td>ampnorm.nf, ampnorm.ff, difdiv.iun, dabd.iof, dbmin.db, dbloss.grd, iregion.nf, iregion.ff, plot.dir</td>
<td>ddbamp.grd, ddbphase.grd, udfdb.out, difdb.iun, fort.xx^1</td>
</tr>
<tr>
<td>UDIVCBD</td>
<td>difdiv.iun, dabd.iof, dbmin.db, iregion.ff, iregion.nf</td>
<td>udivcbd.out, div.iun, fort.xx^1</td>
</tr>
<tr>
<td>UDIVRBD</td>
<td>difdiv.iun, dabd.iof, iregion.ff, iregion.nf</td>
<td>udivrbd.out, rdiv.iun, fort.xx^1</td>
</tr>
<tr>
<td>UDSX</td>
<td>sub.ds, filter.ff, scale.dx, sub.prn, dabd.iof, fun.dx, dx.iun</td>
<td>ds.iun, uds.out.xx, dsnf.xx^3</td>
</tr>
<tr>
<td>UDSXY</td>
<td>sub.ds, filter.ff, scale.dx, scale.dy, sub.prn, dabd.iof, fun.dx, fun.dy, dx.iun, dy.iun</td>
<td>ds.iun, uds.out.xx, dsnf.xx^3</td>
</tr>
<tr>
<td>UDSXYZ</td>
<td>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</td>
<td>ds.iun, uds.out.xx, dsnf.xx^3</td>
</tr>
<tr>
<td>UDSY</td>
<td>filter.ff, scale.dy, dabd.iof, sub.ds, sub.prn, fun.dy, dx.iun, dy.iun</td>
<td>ds.iun, uds.out.xx, dsnf.xx^3</td>
</tr>
<tr>
<td>UDSZ</td>
<td>filter.ff, scale.dx, dabd.iof, sub.ds, sub.prn, fun.dz, dx.iun</td>
<td>ds.iun, uds.out.xx, dsnf.xx^3</td>
</tr>
<tr>
<td>UECX4</td>
<td>tsx.iun, dx.iun, fun.dx, scale.dx, dabd.iof</td>
<td>uecx4.out, ecx.iun, fort.xx^1</td>
</tr>
<tr>
<td>UECY4</td>
<td>tsy.iun, dy.iun, fun.dy, scale.dy, dabd.iof</td>
<td>uecy4.out, ecy.iun, fort.xx^1</td>
</tr>
</tbody>
</table>
UECZ2  tsz.iun, dz.iun, fun.dz  scale.dz, dabd.iof  uecz2.out, ecz.iun, fort.xx
UECZ3  tsz.iun, dz.iun, fun.dz  scale.dz, dabd.iof  uecz3.out, ecz.iun, fort.xx
UECZ4  tsz.iun, dz.iun, fun.dz  scale.dz, dabd.iof  uecz4.out, ecz.iun, fort.xx
UKCORR dabd.iof, tsz.iun, dz.iun  fun.dz, scale.dz  ukcorr.out, kec.iun
ULAPLCN dabd.iof, plot.dir  lnamp.grd, amp0.grd, ulaplcn.out
UMADEXX dabd.iof, fun.dx, polydx.par  perdx.par, randx.par  dx.iun, umakedx.out, fort.xx
UMAEDYY dabd.iof, fun.dy, polydy.par  perdy.par, randy.par  dy.iun, umakedy.out, fort.xx
UMAEDZZ dabd.iof, fun.dz, polydz.par  perdz.par, randz.par  dz.iun, umakedz.out, fort.xx
UOPNORM dabd.iof, scale.dz, iregion.ff  difdiv.iun, iregion.nf, fun.dz  uopnorm.out
UPRDBCBD dabd.iof, dbloss.grd  dbmin.db, sub.prn  uprdbcbd.out
UPRNCBD dabd.iof, sub.prn  uprncbd.out
UPPRICBD dabd.iof, sub.prn  uppricbd.out
URBDDAT dabd.iof, plot.dir, sub.dat  ff_y_mag.dat, ff_x_mag.dat  urbdatout.dat
URBDGRD dabd.iof, plot.dir  ff_mag.grd, nf_y_mag.dat  urbdgrd.out
URDFFFN filter.ff, dabd.iof  ffnf.dz, ampnorm.ff  fffmag.grd, nf_mag.grd, urdfnff.out
URDNFFF filter.ff, dabd.iof  ampnorm.ff, ampnorm.nf  db.mag.grd, nf_mag.grd, urdnff.out
URMSCBD dabd.iof, ampnorm.ff  plot.dir  urmscbd.out, rms.dat, fort.xx
USUBGRD dbloss.grd, dabd.iof, plot.dir  ampnorm.nf, ampordb.grd  ffamp.grd, ffphased.grd
UTSNFAP tsz.iun, dabd.iof  utsnfap.out, tsamp.iun, tsphs.iun
UTSTZ dz.iun, dabd.iof  fun.dz, scale.dz  utstz.out, tstz.iun
UTSX dabd.iof, filter.ff, sub.prn  dx.iun, fun.dx, scale.dx  utsx.out, fort.xx
<table>
<thead>
<tr>
<th>UTSSXY</th>
<th>dabd.iof, filter.ff, sub.prn</th>
<th>iregion.ff, iregion.nf, tsxy.iun</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dx.iun, fun.dx, scale.dx</td>
<td>utsx.y.out, fort.xx¹</td>
</tr>
<tr>
<td></td>
<td>dy.iun, fun.dy, scale.dy</td>
<td></td>
</tr>
<tr>
<td>UTSSY</td>
<td>dabd.iof, filter.ff, sub.prn</td>
<td>iregion.ff, iregion.nf, tsy.iun</td>
</tr>
<tr>
<td></td>
<td>dy.iun, fun.dy, scale.dy</td>
<td>utsy.out, fort.xx¹</td>
</tr>
<tr>
<td>UTSSZ</td>
<td>dabd.iof, filter.ff, sub.prn</td>
<td>iregion.ff, iregion.nf, tsz.iun</td>
</tr>
<tr>
<td></td>
<td>dz.iun, fun.dz, scale.dz</td>
<td>utsz.out, fort.xx¹</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>MODULE</th>
<th>PARAMETER FILES</th>
<th>OUTPUT FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UACT0</td>
<td></td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTADD0</td>
<td></td>
<td>active.iun, add.iun</td>
</tr>
<tr>
<td>UACTDDB</td>
<td>difdb.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTDIF</td>
<td>dif.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTDIV</td>
<td>div.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTDRV</td>
<td>drv.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTDS</td>
<td>- ERROR¹ -</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTECX</td>
<td>ecx.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTECY</td>
<td>ecy.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTECZ</td>
<td>ecz.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTFF</td>
<td>ff.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTFF0</td>
<td>ff.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTKEC</td>
<td>kec.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTNF</td>
<td>nf.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTNF0</td>
<td>nf.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTTSX</td>
<td>tsx.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTTSXY</td>
<td>tsxy.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTTSY</td>
<td>tsy.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UACTTSZ</td>
<td>tsz.iun</td>
<td>active.iun</td>
</tr>
<tr>
<td>UADD0</td>
<td></td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDDRV</td>
<td>drv.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDDS</td>
<td>- ERROR¹ -</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDECX</td>
<td>ecx.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDECY</td>
<td>ecy.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDECZ</td>
<td>ecz.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDFF</td>
<td>ff.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDF0</td>
<td>ff.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDNF</td>
<td>nf.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDNF0</td>
<td>nf.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDTSX</td>
<td>tsx.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDTXY</td>
<td>tsxy.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDTSY</td>
<td>tsi.yun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UADDTSZ</td>
<td>tsz.iun</td>
<td>add.iun</td>
</tr>
<tr>
<td>UDCDFDV</td>
<td>difdiv.iun, ffornf.iun</td>
<td>difdiv.iun</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ff.iun, nf.iun</td>
</tr>
<tr>
<td>UDECFF</td>
<td>ff.iun</td>
<td>ff.iun</td>
</tr>
<tr>
<td>UDECNF</td>
<td>nf.iun</td>
<td>nf.iun</td>
</tr>
<tr>
<td>UINCF</td>
<td>ff.iun</td>
<td>ff.iun</td>
</tr>
<tr>
<td>UINCNF</td>
<td>nf.iun</td>
<td>nf.iun</td>
</tr>
</tbody>
</table>
UNITUN init.iun, data.dir

UNCFDV dif.div.iun, fFornf.iun
   ff.iun, nf.iun

UNORM0 tempnorm.nf, tempnorm.ff
   ampnorm.nf, ampnorm.ff

UNORM1 ampnorm.nf, ampnorm.ff
   tempnorm.nf, tempnorm.ff

URESTFF save.fFs

URESTNF save.nFs

USAVERF fF.iun

USAVENF nf.iun

USCLDX1 epsxyz.set, scale.dx

USCLDX2 epsxyz.set, scale.dx

USCLDX3 epsxyz.set, scale.dx

USCLDY1 epsxyz.set, scale.dy

USCLDY2 epsxyz.set, scale.dy

USCLDY3 epsxyz.set, scale.dy

USCLDZ1 epsxyz.set, scale.dz

USCLDZ2 epsxyz.set, scale.dz

USCLDZ3 epsxyz.set, scale.dz

USETDB1 dbmins.set

USETDB2 dbmins.set

USETDB3 dbmins.set

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.dr
   rdiv.iun, scale.dx, scale.dy, scale.dz
   tsamp.iun, tsphs.iun, tstz.iun, tsx.iun
   tsxy.iun, txy.iun, tsz.iun

USWboth fFornf.iun
USWFFNF    ffornf.iun
USWTOAMP
USWTODB
USWTOFF
USWTONF
USWTONON

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

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>initialization designator</td>
</tr>
<tr>
<td>1</td>
<td>one dimensional, subsequent operation designator, alternate procedure designator</td>
</tr>
<tr>
<td>2</td>
<td>two dimensional</td>
</tr>
<tr>
<td>a</td>
<td>&quot;a&quot;, access, array, ascii, amplitude</td>
</tr>
<tr>
<td>b</td>
<td>&quot;b&quot;, backward, binary</td>
</tr>
<tr>
<td>c</td>
<td>change, convert to, column, complex, constant, copy</td>
</tr>
<tr>
<td>d</td>
<td>data, derivative, difference, dimensional, direct (access), disk (as a storage location), double precision</td>
</tr>
<tr>
<td>e</td>
<td>exponential, even</td>
</tr>
<tr>
<td>f</td>
<td>far, field, file, forward, formatted, function</td>
</tr>
<tr>
<td>g</td>
<td>gamma, generate</td>
</tr>
<tr>
<td>i</td>
<td>imaginary, imaginary part, integer</td>
</tr>
<tr>
<td>k</td>
<td>k (integer constant), wave number, spectrum space k</td>
</tr>
<tr>
<td>l</td>
<td>l (integer constant)</td>
</tr>
<tr>
<td>m</td>
<td>m (integer constant), maximum, minimum, minus, multiple</td>
</tr>
<tr>
<td>n</td>
<td>near (fresnel region), negative quantity</td>
</tr>
<tr>
<td>o</td>
<td>odd, or</td>
</tr>
<tr>
<td>p</td>
<td>parameter(s), phase, plot, plus, power, print, product, pseudo</td>
</tr>
<tr>
<td>r</td>
<td>read, real (single precision), real part, row</td>
</tr>
<tr>
<td>s</td>
<td>shift, shifted, single precision, store, sum, plural designation</td>
</tr>
<tr>
<td>t</td>
<td>taylor (series), times, transform</td>
</tr>
<tr>
<td>w</td>
<td>weight, weighted, write</td>
</tr>
<tr>
<td>x</td>
<td>coordinate (distance along x axis), general variable designation</td>
</tr>
<tr>
<td>y</td>
<td>name change letter (to avoid conflicts)</td>
</tr>
<tr>
<td>z</td>
<td>coordinate (distance along y axis)</td>
</tr>
<tr>
<td>as</td>
<td>ascii</td>
</tr>
<tr>
<td>bd</td>
<td>binary data</td>
</tr>
<tr>
<td>ca</td>
<td>complex array &quot;a&quot;</td>
</tr>
<tr>
<td>cb</td>
<td>complex array &quot;b&quot;</td>
</tr>
<tr>
<td>cc</td>
<td>complex constant</td>
</tr>
<tr>
<td>ch</td>
<td>character variable</td>
</tr>
<tr>
<td>cm</td>
<td>centimeter</td>
</tr>
<tr>
<td>cr</td>
<td>create</td>
</tr>
<tr>
<td>da</td>
<td>direct access</td>
</tr>
<tr>
<td>db</td>
<td>decibel</td>
</tr>
<tr>
<td>df</td>
<td>difference</td>
</tr>
<tr>
<td>ds</td>
<td>direct sum</td>
</tr>
<tr>
<td>dx</td>
<td>derivative with respect to x, increment in x direction</td>
</tr>
</tbody>
</table>
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  pseudo 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
bdnr  boundary
char  character
gama  gamma
<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACPCFFD</td>
<td>introduce Amplitude Change and Phase Change to Far-Field Data</td>
</tr>
<tr>
<td>ACPCNFD</td>
<td>introduce Amplitude Change and Phase Change to Near-Field Data</td>
</tr>
<tr>
<td>ACTIUN</td>
<td>return integer value from ACTIVE.IUN if nonzero</td>
</tr>
<tr>
<td>ADABIOF</td>
<td>get the Input Output File for ASCII to Direct Access Binary routines</td>
</tr>
<tr>
<td>ADABPAR</td>
<td>read the PARameters for the ASCII to Direct Access Binary conversion</td>
</tr>
<tr>
<td>ADDBOX</td>
<td>ADD a BOX function of amplitude EPS to complex CDATA</td>
</tr>
<tr>
<td>AMPDIF2</td>
<td>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)</td>
</tr>
<tr>
<td>APDSET1</td>
<td>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)</td>
</tr>
<tr>
<td>APNAME</td>
<td>append a given filename to the character string ‘AP’</td>
</tr>
<tr>
<td>CABD</td>
<td>convert two ASCII datasets to a self-documented Complex Binary Dataset</td>
</tr>
<tr>
<td>CACB3D</td>
<td>get the Input Output File for the Conversion of ASCII to Binary Data</td>
</tr>
<tr>
<td>CADD1</td>
<td>ADDition of two Complex arrays (1-dimensional)</td>
</tr>
<tr>
<td>CADD2</td>
<td>ADDition of two Complex arrays (2-dimensional)</td>
</tr>
<tr>
<td>CADD3</td>
<td>ADDition of two Complex arrays (3-dimensional)</td>
</tr>
<tr>
<td>CADDCC2</td>
<td>ADDition of a Complex Constant to a Complex Array (2-dimensional)</td>
</tr>
<tr>
<td>CACEIPH2</td>
<td>Complex Array times Exponential to power I times real array (2-dimensional)</td>
</tr>
<tr>
<td>CADDCC</td>
<td>read two ASCII files and create a self-documented Direct-Access Complex Binary file</td>
</tr>
<tr>
<td>CADDCC3</td>
<td>Complex Array Plus Complex Constant in Double precision (1-dimensional)</td>
</tr>
<tr>
<td>CADDCC4</td>
<td>convert Complex numbers with Amplitude-Phase format to Real-Imaginary format (2-dimensional)</td>
</tr>
<tr>
<td>CADDCC5</td>
<td>convert Complex numbers with Amplitude-Phase format to Real-Imaginary format (1-dimensional)</td>
</tr>
<tr>
<td>CADDCC6</td>
<td>convert Complex numbers with Amplitude-Phase format numbers to Real-Imaginary format using double-precision intrinsic functions</td>
</tr>
<tr>
<td>CAPCCD1</td>
<td>Complex Array Plus Complex Constant in Double precision (1-dimensional)</td>
</tr>
<tr>
<td>CAPRI</td>
<td>convert Complex numbers with Amplitude-Phase format to Real-Imaginary format (2-dimensional)</td>
</tr>
<tr>
<td>CAPRI1</td>
<td>convert Complex numbers with Amplitude-Phase format to Real-Imaginary format (1-dimensional)</td>
</tr>
<tr>
<td>CAPRI2D</td>
<td>convert Complex numbers with Amplitude-Phase format numbers to Real-Imaginary format using double-precision intrinsic functions</td>
</tr>
<tr>
<td>CARAYMX2</td>
<td>find a Complex ARray’s MaXimum (2-dimensional)</td>
</tr>
</tbody>
</table>
CMULT2 multiply a 3-dimensional double-precision complex array by a 2-dimensional double-precision array raised to an integer power (3-dimensional)
CMULTL3 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
CHKEEQFP 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
CRIAO2 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 trigonometric 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

DNFDXY  mixed Derivatives with respect to X and Y coordinates

DNFDY  Derivative of Near Field with respect to the Y coordinate

DNFDZ  Derivative 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 $e^{iz}$ (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
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

FPFFF given a Far-Field, obtain the Pseudo Far-Field

FFRINGE 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

FLTTEIGZ FiLTeR the array containing values of $e^{i\gamma z}$ using a specified lower limit on $\gamma^2$

FLTTLIMIT obtain data-point-spacing criteria for LIMiTing the plane-wave spectrum

FLTLPWSG FiLTeR sum of logarithm of Plane-Wave Spectrum added to $e^{i\gamma z}$

FLTRHIIK 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 FoRmated 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 $\gamma^2$

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 PRiNt 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
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSTDIF</td>
<td>append file information to HiSTory file from program UDFICBD</td>
</tr>
<tr>
<td>HSTDIF2</td>
<td>append file information to HiSTory file from program UDF2ICBD</td>
</tr>
<tr>
<td>HSTDFA</td>
<td>append file information to HiSTory file from program UDFIFACBD</td>
</tr>
<tr>
<td>HSTDIV</td>
<td>append file information to HiSTory file from program UDIVCBD</td>
</tr>
<tr>
<td>HSTDRV</td>
<td>append file information to HiSTory file from program UDERIV</td>
</tr>
<tr>
<td>HSTDS</td>
<td>append file information to HiSTory file from program UDS</td>
</tr>
<tr>
<td>HSTDS3</td>
<td>append file information to HiSTory file from program UDSXYZ</td>
</tr>
<tr>
<td>HSTDSX</td>
<td>append file information to HiSTory file from program UDSX</td>
</tr>
<tr>
<td>HSTDSXY</td>
<td>append file information to HiSTory file from program UDXY</td>
</tr>
<tr>
<td>HSTDSY</td>
<td>append file information to HiSTory file from program UDSY</td>
</tr>
<tr>
<td>HSTEC</td>
<td>append file information to HiSTory file from program UERCCOR</td>
</tr>
<tr>
<td>HSTFFNF</td>
<td>append file information to HiSTory file from program URDFFNF</td>
</tr>
<tr>
<td>HSTFFC</td>
<td>append file information to HiSTory file from program UKCORR</td>
</tr>
<tr>
<td>HSTMKDX</td>
<td>append file information to HiSTory file from program UMAKEDX</td>
</tr>
<tr>
<td>HSTMKDY</td>
<td>append file information to HiSTory file from program UMAKEDY</td>
</tr>
<tr>
<td>HSTNFFF</td>
<td>append file information to HiSTory file from program URDNFFF</td>
</tr>
<tr>
<td>HSTNRM</td>
<td>append file information to HiSTory file from program UOPNORM</td>
</tr>
<tr>
<td>HSTRDIV</td>
<td>append file information to HiSTory file from program UDIVRB</td>
</tr>
<tr>
<td>HSTRMS</td>
<td>append file information to HiSTory file from program URMSCBD</td>
</tr>
<tr>
<td>HSTDTS</td>
<td>append file information to HiSTory file from program UTS</td>
</tr>
<tr>
<td>HSTTSAP</td>
<td>append file information to HiSTory file from program UTSNFAP</td>
</tr>
<tr>
<td>HSTTSST</td>
<td>append file information to HiSTory file from program UTSST</td>
</tr>
<tr>
<td>HSTTSX</td>
<td>append file information to HiSTory file from program UTSX</td>
</tr>
<tr>
<td>HSTTSXY</td>
<td>append file information to HiSTory file from program UTSXY</td>
</tr>
<tr>
<td>HSTTSY</td>
<td>append file information to HiSTory file from program UTSY</td>
</tr>
<tr>
<td>HSTUN0</td>
<td>append file information to HiSTory file from program UNORM0</td>
</tr>
<tr>
<td>HSTUN1</td>
<td>append file information to HiSTory file from program UNORM1</td>
</tr>
<tr>
<td>INPDABP</td>
<td>INPut Direct-Access Binary file Parameters</td>
</tr>
<tr>
<td>INPDACB</td>
<td>INPut Direct-Access Binary file Parameters and the dataset</td>
</tr>
<tr>
<td>INPDVRP</td>
<td>INPut a subset of the direct-access binary file parameters needed for the module UDERIV</td>
</tr>
<tr>
<td>INPDZP</td>
<td>INPut a subset of the direct-access binary file parameters needed for the module UMAKEDZ</td>
</tr>
<tr>
<td>INPFFP</td>
<td>INPut a subset of the direct-access binary file Parameters needed to characterize the Far Field</td>
</tr>
<tr>
<td>INPFFP0</td>
<td>INPut Far Field Parameters and data from the direct-access binary file</td>
</tr>
<tr>
<td>INPFLS</td>
<td>INPut the filename pointing to a list of FILEnameS and read the list</td>
</tr>
<tr>
<td>INPGRDP</td>
<td>INPut a subset of the direct-access binary file needed for module UCBDRD</td>
</tr>
<tr>
<td>INPNFP</td>
<td>INPut a subset of the direct-access binary file Parameters needed to characterize the Near Field</td>
</tr>
<tr>
<td>INPNFPO</td>
<td>INPut Near-FieldParameters and data from direct-access binary file</td>
</tr>
<tr>
<td>INPRCBD</td>
<td>INPut two Real datasets into a Complex Binary Data array</td>
</tr>
<tr>
<td>INPTSP</td>
<td>INPut a subset of the direct-access binary file needed for the module UTS</td>
</tr>
</tbody>
</table>

37
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSLOSS</td>
<td>convert INSection LOSS from decibels to amplitude and scale data array</td>
</tr>
<tr>
<td>INTNF3</td>
<td>INTegral of Near Field with respect to the 3rd coordinate Z</td>
</tr>
<tr>
<td>IUNIT</td>
<td>function to increment by 1 the current Integer UNIT number</td>
</tr>
<tr>
<td>IUNS</td>
<td>Obtain the unit numbers in files NF.IUN and FF.IUN and check that additional unit numbers are available</td>
</tr>
<tr>
<td>IYJXCN1</td>
<td>determine the midpoint (CeNTer) of two integers</td>
</tr>
<tr>
<td>LAPLCAN</td>
<td>calculate the LAPLaCiAN of a near field</td>
</tr>
<tr>
<td>MAKEDZ</td>
<td>MAKE a function DZ, which is a function of X and Y</td>
</tr>
<tr>
<td>MAKPFF</td>
<td>MAKe a Pseudo Far Field equal to the Fourier transform of the box function</td>
</tr>
<tr>
<td>MDARB1</td>
<td>Make two single-precision copies of a Double-precision ARray (1-dimensional)</td>
</tr>
<tr>
<td>MDNFDX</td>
<td>Multiple Derivatives of the Near Field with respect to X</td>
</tr>
<tr>
<td>MDNFDY</td>
<td>Multiple Derivatives of the Near Field with respect to Y</td>
</tr>
<tr>
<td>MDNFDZ</td>
<td>Multiple Derivatives of the Near Field with respect to Z</td>
</tr>
<tr>
<td>MDNFDZE</td>
<td>Multiple derivatives of the Near Field with respect to Z, Even orders</td>
</tr>
<tr>
<td>MDNFDZO</td>
<td>Multiple derivatives of the Near Field with respect to Z, Odd orders</td>
</tr>
<tr>
<td>MIGAMMZ</td>
<td>calculate the array (-i\gamma Z)</td>
</tr>
<tr>
<td>MKGAMMA</td>
<td>MaKe the arrays (\gamma^2) and (KY) and (KX)</td>
</tr>
<tr>
<td>MKPERDZ</td>
<td>MaKe a function DZ, which is a PERiodic function of X and Y</td>
</tr>
<tr>
<td>MKPERFN</td>
<td>MaKe a PERiodic FuNction of the form (A \cdot (x - x_0) \cdot B \cdot (y - y_0))</td>
</tr>
<tr>
<td>MKPLYDZ</td>
<td>MaKe a PoLYnomial function DZ of X and Y</td>
</tr>
<tr>
<td>MKPOLYN</td>
<td>MaKe a POLYnomial function of x and y raise to a specified power</td>
</tr>
<tr>
<td>MMICA</td>
<td>obtain the Minimum and Maximum of a Complex Array’s Imaginary part</td>
</tr>
<tr>
<td>MMRA</td>
<td>obtain the Minimum and Maximum of a Real array</td>
</tr>
<tr>
<td>MMRCA</td>
<td>obtain the Minimum and Maximum of a Complex Array’s Real part</td>
</tr>
<tr>
<td>NF</td>
<td>create a complex NF array using arbitrary functions of x and y</td>
</tr>
<tr>
<td>NFFF</td>
<td>given a Near Field, compute the corresponding Far Field</td>
</tr>
<tr>
<td>NFMODX</td>
<td>MODulate a complex NF array with a function of X</td>
</tr>
<tr>
<td>NFMODY</td>
<td>MODulate a complex NF array with a function of Y</td>
</tr>
<tr>
<td>NFPFF</td>
<td>given a Near Field, obtain the corresponding Pseudo Far Field</td>
</tr>
<tr>
<td>NFTSXK</td>
<td>calculate the Kth term of the Taylor Series in X from Near-Field data</td>
</tr>
<tr>
<td>NFTSXY</td>
<td>sum the X &amp; Y Taylor Series terms and print out selected partial sum results</td>
</tr>
<tr>
<td>NFTSXYK</td>
<td>calculate the Kth term of the Taylor Series in X &amp; Y from Near-Field data</td>
</tr>
<tr>
<td>NFTSYK</td>
<td>calculate the Kth term of the Taylor Series in Y from Near-Field data</td>
</tr>
<tr>
<td>NFZKTS</td>
<td>create the Kth term of the Taylor Series in Z from Near-Field data</td>
</tr>
<tr>
<td>NORM2</td>
<td>NORMalize a real array by a constant divided by the difference between the array’s maximum and minimum values (2 dimensional)</td>
</tr>
<tr>
<td>OUTASC</td>
<td>convert a complex array to amplitude and phase format and OUTput the resulting real and imaginary parts as two ASCII datasets</td>
</tr>
<tr>
<td>OUTDAGB</td>
<td>OUTput a Direct-Access Complex Binary dataset</td>
</tr>
<tr>
<td>OUTDPS</td>
<td>OUTput a complex array to Disk, a Print array and/or a Storage array</td>
</tr>
<tr>
<td>OUTGRD</td>
<td>OUTput the amplitude and phase of a complex array .GRD files</td>
</tr>
<tr>
<td>OUTFPF50</td>
<td>get 4 filenames and OUTput 4 .PLT files</td>
</tr>
</tbody>
</table>
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
FFFF given a Pseudo Far Field, obtain the corresponding Far Field
FFNF given a Pseudo Far Field, obtain the corresponding Near Field
FREAL PlotFile data obtained from real array
FREIM PlotFile data obtained from the REal or IMaginary part of complex array
FSET 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
PFCRAP 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
PRNCFUNC  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
PRNRCOR  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)
RCB21  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 alternating 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)
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDRBD2D</td>
<td>ReaD a Real Binary Dataset in Double precision (2-dimensional)</td>
</tr>
<tr>
<td>REARANG</td>
<td>amplitude, phase, distance correction and swap to obtain far-field data</td>
</tr>
<tr>
<td>RINIT1</td>
<td>INITialization with a Real constant (1-dimensional)</td>
</tr>
<tr>
<td>RINIT2</td>
<td>INITialization with a Real constant (2-dimensional)</td>
</tr>
<tr>
<td>RKCARBK2</td>
<td>weighted complex sum of a 2-dimensional Complex Array times a Real Array raised to an integer power (2-dimensional)</td>
</tr>
<tr>
<td>RMSQR</td>
<td>Convert the square root of an array’s elements into decibels and return the number of non-zero elements (1-dimensional)</td>
</tr>
<tr>
<td>RMULT2</td>
<td>Real MULTiplication of two arrays (2-dimensional)</td>
</tr>
<tr>
<td>RNDM</td>
<td>function call to return a RANDoM number</td>
</tr>
<tr>
<td>RNDMDZ</td>
<td>create a RandOM array DZ</td>
</tr>
<tr>
<td>RNDMFCT</td>
<td>create a RandOM FunCTion and store in an real array (2 dimensional)</td>
</tr>
<tr>
<td>RRA2B1</td>
<td>copy a Row of data from a apecified column of a real 2-dimensional array to a 1-dimensional array</td>
</tr>
<tr>
<td>RRATIO2</td>
<td>calculate the RATIO of the elements of two Real arrays (2-dimensional)</td>
</tr>
<tr>
<td>RSUMCOL</td>
<td>Sum a COlumn of elements of a Real array (2-dimensional)</td>
</tr>
<tr>
<td>RSUMROW</td>
<td>Sum a ROW of elements of a Real array (2-dimensional)</td>
</tr>
<tr>
<td>RZTORC1</td>
<td>Real array TO a Real Constant power (1-dimensional)</td>
</tr>
<tr>
<td>RZTORC2</td>
<td>Real array TO a Real Constant power (2-dimensional)</td>
</tr>
<tr>
<td>SCALE</td>
<td>an array of consecutive integers multiplied by a real constant (1-dimensional)</td>
</tr>
<tr>
<td>SCLCC1</td>
<td>SCALe a Complex array by a Complex Constant (1-dimensional)</td>
</tr>
<tr>
<td>SCLCC2</td>
<td>SCALe a Complex array by a Complex Constant (2-dimensional)</td>
</tr>
<tr>
<td>SCLRC1</td>
<td>SCALe a Complex array by a Real Constant (1-dimensional)</td>
</tr>
<tr>
<td>SCLRC1D</td>
<td>SCALe a Complex array by a Real Double precision Constant (1-dimensional)</td>
</tr>
<tr>
<td>SCLRR2</td>
<td>SCALe a Real array by a Real Constant (1-dimensional)</td>
</tr>
<tr>
<td>SCLRR2D</td>
<td>SCALe a Real Double precision array by a Double precision constant (2-dimensional)</td>
</tr>
<tr>
<td>SETBNDR</td>
<td>SET the BouNDaRy of a complex array equal to a complex constant</td>
</tr>
<tr>
<td>SETFILS</td>
<td>SET up to read a list of FILenames</td>
</tr>
<tr>
<td>SETFIOF</td>
<td>get the Input Output File for reading Filenames</td>
</tr>
<tr>
<td>SETFPAR</td>
<td>read the output Filenames</td>
</tr>
<tr>
<td>SETTSX</td>
<td>SET up the necessary arrays for a Taylor Series in X calculation</td>
</tr>
<tr>
<td>SETTSXY</td>
<td>SET up the necessary arrays for calculating a Taylor Series along both the X and Y coordinates</td>
</tr>
<tr>
<td>SETTSSY</td>
<td>SET up the necessary arrays for a Taylor Series in Y calculation</td>
</tr>
<tr>
<td>SETTTSZ</td>
<td>SET up the necessary arrays for Taylor Series in Z calculations</td>
</tr>
<tr>
<td>SFTCACB</td>
<td>ShiFT the location of Complex data in zero padded array to array center</td>
</tr>
<tr>
<td>SPTRARB</td>
<td>ShiFT the location of Real data in zero padded array to array center</td>
</tr>
<tr>
<td>SIN4X</td>
<td>calculate the function SINE of X raised to the fourth power</td>
</tr>
<tr>
<td>STRNGLN</td>
<td>find the locations of the first and last non-blank characters in a string</td>
</tr>
<tr>
<td>SUMPRM</td>
<td>SUM the values on the PeRiMeter of a square embedded within a real array (2-dimensional)</td>
</tr>
<tr>
<td>SUMPRM1</td>
<td>SUM the values on the PeRiMeter of a square embedded within a real array, omitting the corners from the sum (2-dimensional)</td>
</tr>
</tbody>
</table>
SUMS

SWAP

SWCRIP2

TIMER

TIMERS

TODAY

TSCOEF

TSXSLM

TSXYSN

TSYSLM

TSZK

TSZK1

TSZSLM

TSZSLM0

TSZSLM1

UDASCUN

UDDSUN

UDDXIUN

UDDYIUN

UDZIUN

UDFFIUN

UDNFIUN

WCBD1

WCBD2

WCBD3

WDACBD

WDSCBD

WLTOCM

WLTOCMD

WRBD2

WRBD2D

WRCKF

WRFUNC

XCHAR

XSCHAR

XYGRIDS

an array of SUMs of a real array’s elements from successively increasing array locations to the end (1-dimensional)

SWitch beginning to end Array-element Positions of both rows and columns

Weighted Complex Sum of 3-dimensional Complex array times a Real array raised to an array of Integer Powers (2-dimensional)

store system TIME on first call, return time difference on second call

multiple TIME initializations, time differences returned on second call

write current date to screen

calculate and store the Taylor Series COEFficients and an array of integer powers

Taylor Series in X Summed from Low to Maximum order

Taylor Series in X and Y Summed to order N

Taylor Series in Y Summed from Low to Maximum order

Taylor Series in Z term of order K

Taylor Series in Z term of order K and output partial sum

Taylor Series in Z Summed from Low to Maximum order

Taylor Series in Z Summed from Low to Maximum order after initialization

Taylor Series in Z Summed from Low to Maximum order, and output each partial sum

UpDate ASCII output file Unit Number

UpDate Direct-Sum output file Unit Number

UpDate DX error array’s output file Unit Number

UpDate DY error array’s output file Unit Number

UpDate DZ error array’s output file Unit Number

UpDate Far Field’s output file Unit Number

UpDate Near Field’s output file Unit Number

Write an unformatted Complex Binary Dataset (1-dimensional)

Write unformatted Complex Binary Dataset to fort.xx file (2-dimensional)

Write unformatted Complex Binary Dataset to fort.xx file (3-dimensional)

Write a self-documented Direct-Access Complex Binary Dataset

Write unformatted Complex Binary Dataset to dsnf.xx file (2-dimensional)

convert a WaveLength TO CentiMeters

convert WaveLengths TO Centimeters in double precision

Write an unformatted Real Binary Dataset to a fort.xx file

Write an unformatted Double-precision Real Binary Dataset to a fort.xx file (2-dimensional)

WRite a CHecK list of parameters to a print File

WRite the specified FUNCtion name to specified file

eXpress an integer modulus 100 as a CHARacter variable

eXpress an integer modulus 10 as a Single CHARacter variable

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 inputting 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:

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

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**: iundfdbl= 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**: iuntsx= 0
- **tsy.iun**: iuntsy= 0
- **tsz.iun**: iuntsz= 0
- **tsamp.iun**: iuntsa= 0
- **tsphs.iun**: iunsp= 0
- **tstz.iun**: iunst= 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ampordb.grd:</td>
<td>dB</td>
</tr>
<tr>
<td>filter.ff:</td>
<td>0.0000000E+00</td>
</tr>
<tr>
<td>order.drv:</td>
<td>1</td>
</tr>
<tr>
<td>scale.dx:</td>
<td>0.0000000E+00</td>
</tr>
<tr>
<td>scale.dy:</td>
<td>0.0000000E+00</td>
</tr>
<tr>
<td>scale.dz:</td>
<td>0.0000000E+00</td>
</tr>
<tr>
<td>ampff_invff:</td>
<td>1987.822</td>
</tr>
<tr>
<td>ampnf_invnf:</td>
<td>1.059250</td>
</tr>
<tr>
<td>fnf.dz:</td>
<td>0.0000000E+00</td>
</tr>
<tr>
<td>ffnf.dz:</td>
<td>0.0000000E+00</td>
</tr>
<tr>
<td>fun.dx:</td>
<td>per</td>
</tr>
<tr>
<td>fun.dy:</td>
<td>per</td>
</tr>
<tr>
<td>fun.dz:</td>
<td>per</td>
</tr>
<tr>
<td>active.iun:</td>
<td>0</td>
</tr>
<tr>
<td>add.iun:</td>
<td>0</td>
</tr>
<tr>
<td>amp2.iun:</td>
<td>0</td>
</tr>
<tr>
<td>asci.iun:</td>
<td>7</td>
</tr>
<tr>
<td>inc difdiv:</td>
<td>1</td>
</tr>
<tr>
<td>dif2.iun:</td>
<td>0</td>
</tr>
<tr>
<td>dif.iun:</td>
<td>0</td>
</tr>
<tr>
<td>difdb.iun:</td>
<td>0</td>
</tr>
<tr>
<td>div.iun:</td>
<td>0</td>
</tr>
<tr>
<td>drv.iun:</td>
<td>0</td>
</tr>
<tr>
<td>ds.iun:</td>
<td>-1</td>
</tr>
<tr>
<td>dx.iun:</td>
<td>0</td>
</tr>
<tr>
<td>dy.iun:</td>
<td>0</td>
</tr>
<tr>
<td>dz.iun:</td>
<td>63</td>
</tr>
<tr>
<td>ecx.iun:</td>
<td>0</td>
</tr>
<tr>
<td>ecy.iun:</td>
<td>0</td>
</tr>
<tr>
<td>ecz.iun:</td>
<td>0</td>
</tr>
<tr>
<td>ff.iun:</td>
<td>60</td>
</tr>
</tbody>
</table>

46
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: \(0.0000000\times10^0\)
  order.drv: 1 1
  scale.dx: \(0.0000000\times10^0\) 0.1000000
  scale.dy: \(0.0000000\times10^0\) 0.1000000
  scale.dz: \(0.0000000\times10^0\) 0.2000000
  ampf,invff: 1987.822 5.0306314E-04
  ampnf,invnf: 1.059250 9.4406420E-01
  ff,dz: \(0.0000000\times10^0\)
  ffornf: ff
  fun.dx: 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
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: inppfp inpnfp rdcbd2 csqr1 wrbd2 udffiu udnfiun hstamp2 udascun
UAPDACB: cadacb
UCBDDAT: ffornf inpgrdp getwn chlnth gtprrnr udascun xygrids rdcbd2
         insloss criap db1 cnimcc1 pfpcrep outpfs0 xchar xschar
UCBDGRD: inpgrdp getwn chlnth xygrids ranges rdcbd2 insloss udascun outgrd
UDBPDACB: grddacb
UDERIV: chlnth inpdrvp rdcbd2 getwn settsz dnfzd wcbd2 udnfiun hstdrv
         udascun prncorr xygrids ranges outgrd
UDIF2CBD: inppfp inpnfp rdcbd2 csqr1 rdif2 wrbd2 udffiu udnfiun hstdif2 udascun
UDIFACBD: inppfp inpnfp rdcbd2 cabs1 rdif2 wrbd2 udffiu udnfiun hstdifa udascun
UDIFCBD: inppfp getwn inpnfp cbsdtodb cdif2 wcbd2 udnfiun hstdfududascun
UDIFDB: inppfp getwn inpnfp cbsdtodb cdif2 wcbd2 udnfiun hstdfbd
         udascun xygrids ranges outgrd
UDIVCBD: inppfp inpnfp rdcbd2 cinit1 cratio2 wcbd2 udnfiun hstdiv udascun
UDIVRBD: inppfp inpnfp rdbd2 cinit1 rratio2 wrbd2 udnfiun hstdriv udascun
UDSX: uddsiun inppfp0 inppfp rdcbd2 getwn mkgamma cinit1 inppfp0 inppfp
       nff chkglei catocb2 ffnf gtprrnr chlnth udascun xchar prnfunc wltocm
       prncorr rdbd2 scrr2 ffnfx wdscbd hstdsx
UDSXY: uddsiun inppfp0 inppfp rdcbd2 getwn mkgamma cinit1 inppfp0 inppfp
       nff chkglei catocb2 ffnf gtprrnr chlnth udascun xchar prnfunc wltocm
       prncorr rdbd2 scrr2 ffnfx wdscbd hstdsx
UDSZ: uddsiun inppfp0 inppfp rdcbd2 getwn mkgamma cinit1 inppfp0 inppfp
       nff chkglei catocb2 ffnf gtprrnr chlnth udascun xchar prnfunc wltocm
       prncorr rdbd2 scrr2 raddrc2 ffnfxz wdscbd hstds3
UDSY: uddsiun inppfp0 inppfp rdcbd2 getwn mkgamma cinit1 inppfp0 inppfp
       nff chkglei catocb2 ffnf gtprrnr chlnth udascun xchar prnfunc wltocm
       prncorr rdbd2 scrr2 scsrl2 scrl2 ffnfy wdscbd hstdsy
UECX4: inptsp getwn udascun prnfunc wltocm rdbd2 scrl2 rdcbd2 ecx4 gtprrnr
       prncorr wcbd2 udnfiun hstec
UECY4: inptsp getwn udascun prnfunc wltocm rdbd2 scrl2 rdcbd2 ecy4 gtprrnr
       prncorr wcbd2 udnfiun hstec
UECZ2: inptsp wltocm getwn rdbd2 scrl2 rdcbd2 settsz mdnfzd cmultr2 cadd2
dnfdzo scrlc1 catocb2 csuimk2 udascun prncorr wcbd2 udnfiun hstec
UECZ3: inptsp wltocm getwn rdbd2 scrl2 rdcbd2 settsz mdnfzd cmultr2 cadd2
dnfdzo scrlc1 dnfdze csuimk2 udascun prncorr wcbd2 udnfiun hstec
UECZ4: inptsp getwn udascun prnfunc wltocm rdbd2 scrl2 rdcbd2 ecz4 gtprrnr
       prncorr wcbd2 udnfiun hstec
UKCORR: inptsp wltocm getwn rdbd2 scrl2 rdcbd2 caceiph wcbd2 udnfiun
         udascun hstkec

49
ULAPLCN: chlngth cinit1 inpnfp0 inpnfp rdcbd2 getwn mkgamma udascun caraymx2 prncorr lapcan catocb2 sclrc1 xygrids ranges pccrgrd cadd2
UMAKEDX: crfunc inpdzp rinit1 wltocm mkpolyn funcsc1 mkperfn perfunx rndmfct norm2 iyjxcnt sftrarb grid ranges wrbd2 uddxiun hstmkdx udascun rdfunc rarymm2 outrgrd
UMAKEDY: crfunc inpdzp rinit1 wltocm mkpolyn funcsc1 mkperfn perfuny rndmfct norm2 iyjxcnt sftrarb grid ranges wrbd2 uddyiun hstmkdy udascun rdfunc rarymm2 outrgrd
UMAKEDZ: crfunc inpdzp rinit1 wltocm mkpolyn funcsc1 mkperfn perfunz rndmfct norm2 iyjxcnt sftrarb grid ranges wrbd2 uddyiun hstmkdz udascun rdfunc rarymm2 outrgrd
UOPNORM: inpffp inpnfp rdcbd2 casumsq hstnrn chlngth udascun
UPRDBCBD: inpffp inpnfp rdcbd2 insloss criap db2 udascun xchar prnrcorr
UPRNCBD: inpffp inpnfp rdcbd2 udascun xchar prnrcorr
UPRRICBD: inpffp inpnfp rdcbd2 udascun xchar prnrcorr
URBDDAT: inpgrdp getwn chlngth xchar rdrbd2 chkleqi xygrids ranges udascun rarymm2 prnrcor pltfile
URBDGRD: udascun inpgrdp getwn chlngth xygrids ranges rdrbd2 mmra outrgrd
URDFNF: inpffp0 inpnfp rdcbd2 wltocm getwn mkgamma catocb2 ffnf wcbd2 udfiuni udfiuni udascun caraymx2
URDNFFF: cinit1 inpnfp0 inpnfp rdcbd2 catocb2 getwn mkgamma nfff iyjxcnt sftcaeb wcbd2 udfiuni udfiuni udascun caraymx2
URMSCBD: chlngth inpffp0 wcbd2 udfiuni inpffp rdcbd2 chkleqi getwn udascun caraymx2 csqr1 primsun sumsums rmsqr plrdata hstrms
USUBGRD: inpgrdp getwn chlngth xygrids ranges rdcbd2 insloss udascun outrgrd
UTSNFAP: inpffp rdcbd2 criap ciritoc2 capri wcbd2 udfiuni udascun hsttsap
UTSTZ: udascun inptsp wltocm rdcbd2 rdrbd2 sclrr2 getwn catocb2 settsz cinit1 tszslm xchar prncorr wcld2 udfiuni hstttst
UTSX: inptsp chkleqi getwn inpffp0 inpnfp rdcbd2 mkgamma ffnf cinit1 inpnfp0 iyjxcnt sftcaeb udascun prnfunc wltocm gtpnrpr rdrbd2 sclrr2 prncorr coeffts settsx nftsxk cadd2 xchar wcbd2 udfiuni hsttxs
UTSXY: inptsp chkleqi getwn inpffp0 inpnfp rdcbd2 mkgamma ffnf cinit1 inpnfp0 iyjxcnt sftcaeb udascun prnfunc wltocm gtpnrpr rdrbd2 sclrr2 prncorr coeffts settsy nftsxy wcbd2 udfiuni hsttxsy
UTSY: inptsp chkleqi getwn inpffp0 inpnfp rdcbd2 mkgamma ffnf cinit1 inpnfp0 iyjxcnt sftcaeb udascun prnfunc wltocm gtpnrpr rdrbd2 sclrr2 prncorr coeffts settsy nftsyk cadd2 xchar wcbd2 udfiuni hsttsy
UTSZ: inptsp chkleqi getwn inpffp0 inpnfp rdcbd2 mkgamma ffnf cinit1 inpnfp0 iyjxcnt sftcaeb udascun prnfunc wltocm gtpnrpr rdrbd2 sclrr2 prncorr coeffts settsz nftszk cadd2 xchar wcbd2 udfiuni 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:
CAEIPH:
CADACB: adabiof adabpar frradhd capri wdacbd
CADD1:
CADD2:
CADD3:
CADDCC2:
CAEIPH2:
CAEIPHC:
CANECEB2:
CAPCCD1:
CAPRI:
CAPRI1:
CAPRI2D:
CAPRNT: nmrea prntr1d mmica
CARAYMX2:
CARBK3D:
CARBK3D:
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:
CHLENGTH:
CIMGSTR:
CINIT1:
CINIT2:
CIRCFLT:
CMULDS2:
CMULRD2:
CMULT1:
CMULT2:
CMULTR2:
CNIMCC1:
CNTCACB: iyjxcnt sftca cb
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 cato cb2 ffnf
CSMWCP1:
CSMWCP2:
CSQR1:
CSUM1:
CSUM2:
CSUMCP1:
CSUMCP2:
CSUMIK2:
CSUMRW1:
CSUMRW2:
CSWCPE2:
CUTOFF1:
CUTOFF2:
CXPLOG:
DABDIOF: chlength
DABDPAR: chkpar0 chkpar1
DATFILE:
DATORB1:
DB1:
DB2:
DCLN2:
DNFDX: catocb2 prdrtc2 sclc1 fourt acpenfd swap
DNFDXDY: prdrtc2 prdrtc2 sclc1 fourt acpenfd swap
DNFDY: catocb2 prdrtc2 sclc1 fourt acpenfd swap
DNFDZ: dnfdze cmulds2
DNFDZE: cato cb2 prdrtc2 fourt acpenfd swap sclrc1
DNFDZO: migamnz cmulds2 dnfdze
DSPWS:
DSPWSX:
DSPWSXY:
DSPWSY:
DSWCRRP1:
ECEXP:
ECX4: settsx mdnf dx cmultr2 sclrc1 cato cb2 cadd2 csunik2
ECY4: settsy mdnf dy cmultr2 sclrc1 cato cb2 cadd2 csunik2
ECZ4: settsz mdnf dz cmultr2 dnfdzo dnfdze sclrc1 cato cb2 cadd2 csunik2
EIGAMAZ:
ERRMESS:
FAXSBYS: fx fy (unspecified functions)
FFLMTS:
FFNF:
FFNF1UN: strngln
FFNFX: migamnz dcln2 ppwsnf x sclc2
FFNFXY: migamnz dcln2 ppw snf2 sclc2
FFNFXYZ: migamnz dcln2 ppw snf3 sclc2
FFNFY: migamnz dcln2 ppw snfy-sclc2
FFNFZ: migamnz dcln2 ppw snfz sclc2
FFORNF:
FFPPF: eigamaz fitlimt flteigz cmulds2
FFRANGE: fflimts
FFTFFT: sclrr2 fourt
FILSIOF: 
FILSPAR: 
FINDEND: 
FLTEIGZ: 
FLTLIMT: 
FVTLPWSG: ftlimt 
FLTRHIK: nff fnf 
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: ffonf chkleqi 
HSTAMP2: findend 
HSTDDB: findend 
HSTDIF: findend 
HSTDIF2: findend 
HSTDIFA: findend 
HSTDIV: findend 
HSTDIV: findend 
HSTDS: findend 
HSTD3: findend 
HSTDX: findend 
HSTDSXY: findend 
HSTDSY: findend 
HSTEC: findend 
HSTFFNF: findend 
HSTKEC: findend 
HSTMKDX: findend 
HSTMKDY: findend 
HSTMKDJ: findend 

54
HSTNFFF: findend
HSTNRM: findend
HSTRDIV: findend
HSTRMS: findend
HSTTS: findend
HSTTSAP: findend
HSTTTST: 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
INPFPP0: dabdiof dabdpar rddacbd
INPFILS: filsiop filspar
INPGRDP: dabdiof dabdpar rddabp
INPNFP: dabdiof dabdpar rddabp
INPNFP0: dabdiof dabdpar rddacbd
INPRCBD: rcbdiof rcbdpar rcbdset
INPTSP: dabdiof dabdpar rddabp
INSLOSS: selcc1
INTNF3: migammz fourt swap cdivds2 acpcffd acpcnfd
iunit: 1
IUNS: ffnfiun
IYJXCNT:
LAPLCAN: npff dnfds dnfds cadd2 dnfds pffnp
MAKEDZ: grid funaxby sclrr2 (fx fy = unspecified externals)
MAKPFF: cinit1 addbox npff
MDARB1: datorb1
MDNFDX: dnfds
MDNFDY: dnfds
MDNFDZ: mdnfdze mdnfdzo
MDNFDZE: dnfds
MDNFDZO: cmulds2 mdnfdze
MIGAMMZ: gridd gammasq
MKGAMMA: grid funaxby sclrr2 (fx fy = unspecified externals)
MKPERDZ: grid funaxby sclrr2 (fx fy = unspecified externals)
MKPERFLN: grid funaxby (fx fy = unspecified externals)
MKPLYDZ: grid polynxy sclrr2
MKPOLYN: grid polynxy rztorc1
MMICA:
MMRA:
MMRCA:
NF: fx fy (unspecified functions)
NFFF: nfpff pfff
NFMODX: f (unspecified function)
NFMODY: f (unspecified function)
NFPFF: setbndr fourt swap acpcff
NFTSXY: gtpnpr prncorr nftsysk cadd2 nftsxk nftsxyk
NFTSXK: dnfdx rkcarbk2
NFTSXKY: dnfdxdy rkcarbk2
NFTSYK: dnfdy rkcarbk2
NFTSZK: dnfdo dnfdez rkcarbk2
NFZKTS: catocb2 ffnf tszSLM0
NORM2: rarymm2 slrr2
OUTASC: criap fwrad2
OUTDACB: setfils wdacbd
OUTDPS: prncorr catocb2 wcbd2
OUTGRD: criap db1 cnimcc1 mmrca getfile outrgrd mmica
OUTPFS0: getfile pltfile
OUTPFS1: pfcrap outrfs0
OUTRGRD:
PCCRGRD: caraymx2 prncorr outrgrd
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: pfsset pfreim
PFCRAP: pfcrap pfreim
PFFFF: migammz expclog ecexp
PFNF: fourt acpcnfd swap
PFREAL:
PFREIM:
PFSET: cca2b1 cra2b1 criap1 capri1
PLRDATA: scale pfreal pltfile
PLTFILE:
polyn: 1
POLYNXY: polyn
PPFCRAP: prncorr prnrcri pfcrap
PPWSNF2: ffrange ftpwsg carcbd2 dspwsxy
PPWSNF3: ffrange ftpwsg carcbd2 dspwsxy
PPWSNFX: ffrange ftpwsg carcbd2 dspwsx
PPWSNFY: ffrange ftpwsg carcbd2 dspwsy
PPWSNFZ: ffrange ftpwsg carcbd2 dspws
PRDCTC2: sumprm sumprm1
PRDRTC2: cca2b1 criap1 cra2b1
PRDTC2: chlngth rdfunc
PRIMSUM: prncorr outpfs1
PRRTC2: rca2b1 rra2b1
PRNCORR: cca2b1 cra2b1
PRNFUNC: caraymx2
PRNPLT: rca2b1
PRNRCOR: rca2b1
PRNRICR: rca2b1
PRNTCID: caraymx2
PRNTR1D: cca2b1
RADDRC2: cca2b1
RANERB2: cca2b1
RANGED: cca2b1
RARYMM2: cca2b1
RATORB1: cca2b1
RATORB2D: cca2b1
RCA2B1: cca2b1
RCBD2: errmess
RCBDIOF: chlngth
RCBDPAR: chlngth
RCBDSET: rcbd2
RDCBD1: xchar
RDCBD2: xchar
RDCBD3: xchar
RDDABP: chlngth chkpar2 errmess
RDDACBD: chlngth chkpar2 errmess
RDFUNC: chlast
RDIF2: chlast
RDOT: xchar
RDRBD2: xchar
RDRBD2D: xchar
REARANG: swap
RINIT1: swap
RINIT2: swap
RKCARBK2: swap
RMSQR: swap
RMULT2: swap
RNDMDZ: swap
RNDMFCT: swap
RRA2B1: swap
RRATIO2: swap
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 migammsz catocb2 nfpff
SFTRARB:
SFTCACB:
SFTCIP2:
sin4x: 1
STRNGLN:
SUMPRM:
rsumcol rsumrow
SUMPRM1:
rsumcol rsumrow
SUMSUMS:
SWAP:
SWCRIP2:
TIMER:
TIMERS:
TODAY:
TSCOEF:
dnfdx swcrip2
TSXSLM:
dnfdx dy swcrip2
TSXYSN:
dnfdx dy swcrip2
TSYSLM:
dnfdy swcrip2
T SZK:
dnfdzo dnfdze swcrip2
T SZK1:
tszK outdps
TSZSLM:
dnfdzo dnfdze swcrip2
TSZSLMO:
settsz tszSLM
TSZSLM1:
dnfdz swcrip2 criap caprnt catocb2 wcd2
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
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.