A User's Guide to Raytrace

DR. MICHAEL H. REILLY

Ionospheric Effects Branch
E. O. Hulburt Center for Space Research
Space Science Division

DR. ERIC L. STROBEL

Interferometrics, Inc.
8150 Leesburg Pike, Suite 1400
Vienna, VA 22180

February 8, 1989

Approved for public release; distribution unlimited.
The operation of the RAYTRACE (v.4.3) software for ionospheric raytracing is explained. First, the installation and setup of the program are discussed. Instructions for routine operation are given, followed by several tutorial examples. Some suggestions for regular usage are given. The final section and the appendices contain useful reference material, including the source code listing.
CONTENTS

INTRODUCTION .................................................................................................................. 1

OVERVIEW ....................................................................................................................... 2

Summary of Capabilities ............................................................................................... 2
Technical Background ................................................................................................. 3
Operating Requirements .............................................................................................. 4
Design Philosophy ........................................................................................................ 5
Conventions Used ......................................................................................................... 6

FILES .............................................................................................................................. 7

What’s Included ............................................................................................................. 7
Formats Available ......................................................................................................... 8

PROGRAM SETUP ......................................................................................................... 8

VMS Environment ......................................................................................................... 8
DOS Environment .......................................................................................................... 10

OPERATOR’S GUIDE ..................................................................................................... 13

GENERAL INSTRUCTIONS ............................................................................................ 14

Overall Program Flow .................................................................................................. 14
Ionospheric Input .......................................................................................................... 18
Operating Instructions .................................................................................................. 18
Main Menu .................................................................................................................... 22
Launch Parameters Menu ............................................................................................. 23
Elevation Parameters Menu .......................................................................................... 26
Output Files ................................................................................................................... 29

TUTORIAL EXAMPLES .................................................................................................. 31

New Session Using Keyboard Input ............................................................................. 31
New Session Using Data File Input ............................................................................. 50
Old (Saved) Session ..................................................................................................... 54
Remarks ......................................................................................................................... 57

EXAMINING RESULTS .................................................................................................. 57

TIPS, RESTRICTIONS, REMINDERS, ETC. ................................................................ 59

General Comments ...................................................................................................... 59
VMS Specific Comments ............................................................................................... 62
DOS Specific Comments ............................................................................................. 62
Uses and Usage Tips ...................................................................................................... 63

MENU QUICK REFERENCE .......................................................................................... 64

Main Menu .................................................................................................................... 65
Launch Parameters Menu ............................................................................................. 66
Elevation Parameters Menu .......................................................................................... 67
A USER'S GUIDE TO RAYTRACE

PART ONE:
INTRODUCTION
OVERVIEW

This manual is a user's guide to the operation of the program RAYTRACE (Version 4.3) and also covers related topics. As a user's guide, this document is meant to assist with the day to day usage of RAYTRACE. The greater theoretical detail that many researchers may wish to see is contained in a paper that is being published in Radio Science [1]. This user's guide is divided into two sections plus the appendices. The first section is introductory in nature and includes some background material, a 'packing list' of the files, and instructions for installation and setup. The second section is devoted to educating the user in the day to day operation of RAYTRACE. The appendices incorporate a variety of useful information.

The authors of the program RAYTRACE are Dr. Michael H. Reilly of the Naval Research Laboratory, and Dr. Eric L. Strobel of Interferometrics Inc. Dr. Reilly may be reached at the following address:

Dr. Michael H. Reilly
Code 4180.2
Naval Research Laboratory
Washington DC 20375

Phone: (202) 767-2891.

Dr. Strobel may be reached at:

Dr. Eric L. Strobel
Interferometrics Inc.
8150 Leesburg Pike
Vienna VA 22180

Phone: (703) 790-8500.

Summary of capabilities

The program RAYTRACE has a number of distinctive capabilities. It performs fully three-dimensional HF/VHF raytracing through a climatological ionosphere. (Magnetic field effects are, however, not currently implemented.) Single or multiple rays may be traced during a single run, allowing the user greater flexibility in modeling. The
size of the increments used in the raypath calculation is user adjustable so that appropriate compromises may be made between execution time and accuracy. The program can trace rays between points at different altitudes. Cutoff criteria may be set to stop the calculation at a particular range or altitude.

RAYTRACE calculates a number of important quantities about the ray. The change in signal intensity due to distance and ionospheric focusing or defocusing is calculated. The group and phase path length values are both calculated. The location, as well as information on the direction of the ray, are calculated for the end point of the ray and any intervening earth impact points (collectively these are called the data points for the ray).

Technical background

The RAYTRACE program is an implementation of a technique that is much more fully discussed in a paper that will be published in Radio Science [1]. A brief synopsis of that paper's discussion concerning program details is presented here. This section may be skipped with no loss of continuity.

The RAYTRACE algorithm utilizes a set of raypath differential equations that is separable under some approximation assumptions. These assumptions involve expanding the square of the index of refraction in a Taylor series in a local set of Cartesian coordinates and then truncating the series. If the coefficients of this expansion are known, then the raypath differential equations are integrable. The coefficients corresponding to the terms in the horizontal direction are obtained by spatial interpolation of sets of coefficients defined on a latitude and longitude grid. Those in the vertical direction are obtained by using the functional form assumed by whatever particular ionospheric model the raytracing program has been customized to work with.

Given the truncation of the Taylor series, the equations for the raypath are only valid for a limited distance from the origin of the Taylor expansion. For this reason, the modeled raypath is built up incrementally. The raypath equations, then, give the coordinates of the end point of a ray increment in the local Cartesian system.
The program utilizes an ionospheric specification on a latitude-longitude grid of points. The ray is constructed by successively calculating the coefficients of the series, evaluating the raypath equations, and updating the coordinates and other needed values in preparation for the next increment. The process continues until some cutoff criterion is met.

This raytracing algorithm is quite compact, as it utilizes analytical integration of the raypath differential equations, rather than numerical integration. The storage requirements are smaller than if numerical integration was used because a particular (perhaps piecewise-functional form for the vertical structure of the ionosphere is assumed. This exacts a price, however. If a change is made to the underlying ionospheric model, or if a different model is to be used, portions of the RAYTRACE code need to be changed. The current implementation of RAYTRACE corresponds to that detailed in the paper [1], so the model to which it is currently matched is the RADAR-C ionospheric model [2]. The vertical profile from this model and the number of parameters necessary to specify the profile determine the calculation of the expansion coefficients and the resulting parameters and derivatives.

Operating requirements

The current implementation of RAYTRACE is written in Fortran 77. Subject to very minor compiler dependencies, which are discussed in later sections, the program will run on most computers having Fortran 77 compilers. The total amount of disk storage recommended for use with RAYTRACE is about one megabyte. Given the extra space needed for the compiler and system files, use of a hard disk is virtually essential for microcomputer users of RAYTRACE. If necessary, it is possible to place the executable version of the program, the associated files, and a data file on a floppy disk. A minimum of 512 K of memory is recommended, with 640 K or greater suggested. The presence of a math coprocessor is essentially a requirement for microcomputer users.

An additional requirement that many users will face is a method of generating input to RAYTRACE. This will entail a program that will produce a file, with a format given in Appendix A, from either an iono-
spheric model or experimental data. These data may be entered by hand, but this method of data input to RAYTRACE is only practical for relatively small amounts of data.

Design philosophy

The RAYTRACE program has been developed to meet specific user needs, however the development process has been geared toward maintaining the greatest degree of generality in operation consistent with the original specific needs. For this reason, the program is user driven to the greatest extent possible. This provides some capability to do calculations in a "What if ... ?" manner if desired. The current version of the program represents an attempt to strike a balance between modularity, maintainability of the code, and compactness of both memory space and user interface.

Part of the basic outlook taken while developing RAYTRACE is that this program will most certainly be put to unanticipated uses. The program is therefore set up to be as straightforward as possible at least as simulation programs like this go to modify for such uses. Some ideas on customization are given later. Such customization is bound to occur as RAYTRACE becomes more widely disseminated. The program will grow, but some questions will inevitably come back to the original authors of RAYTRACE. It is therefore strongly recommended that any variations, changes, or additions to RAYTRACE be commented as profusely as possible in the source code. Additionally, the program's authors (and, in fact, the research community at large) ought to be kept abreast of such alterations when appropriate. If this occurs, it will not only aid RAYTRACE's authors in responding to questions, but will facilitate the evolutionary improvement of the program as a general research tool.
Conventions used

It is assumed that the user is familiar with the operating system for the machine that is used. Additionally, the user ought to be knowledgeable about the operation of the compiler and editor to be used. Filenames are given in all capital letters (FILE.EXT), with a three letter extension. The total length of filenames is not to exceed ten characters. Prompts are generally displayed in double quotes (" "), with responses given in single quotes ('). The quotes are not to be typed.

RAYTRACE is currently implemented on a DOS system and a VAX/VMS system, with subtle differences between the two implementations. This manual is written so that those with DOS systems see as little as possible about VMS systems, and vice versa. Although this approach results in some overall repetition, the user may skip over sections not dealing with his/her system without fear of missing key material.
FILES

What's included

The RAYTRACE package is best transported as source code which may then be copied onto a system meeting the requirements stated above for production of the executable version. What follows are brief descriptions of the RAYTRACE source code files and the various supporting files.

The source for the RAYTRACE program itself is broken into six files. The file RAYTRA4.FOR is essentially the user interface shell within which the actual raytracing routines are imbedded. The handling of access of data files is also within this set of routines. RRAYS8.FOR is the actual meat of the raytracing. This contains the routine which implements the raytracing algorithm. The ionospheric parameters and gradients necessary for the raytracing are calculated by routines in RIONO.FOR. Routines to support the calculation of phase path length information are contained in RP8ASE.FOR. The boundaries between segments of the vertical profile of the ionosphere will, in general, be tilted with respect to local vertical. The routines to take this tilting into account are contained in RTILT.FOR. Various other supporting routines are located in the remaining source file, RMISC.FOR.

When RAYTRACE is run, it looks to a few text files which contain information on the make up of the menus that the user will see. These are the files with the .MEN suffix. MASTER.MEN, as the name would indicate, is the master file that RAYTRACE looks for to determine what other menu files will be needed. MAIN.MEN contains the information on the construction of the main menu that the user interacts with during a raytracing run. LAUNCH.MEN and ELEV.MEN determine the menus by which the user will input parameters used in the raytracing.

One more file is included. READER2.FOR is the source code for a brief program that may be used to display the results of a raytracing. Since analysis needs will vary from user to user, this program is meant to provide the basis for more sophisticated programs to analyze raytracing results.
Table 1

Table of files

<table>
<thead>
<tr>
<th>Filename</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAYTRA4.FOR</td>
<td>50248</td>
</tr>
<tr>
<td>RIONO.FOR</td>
<td>63071</td>
</tr>
<tr>
<td>RMISC.FOR</td>
<td>42064</td>
</tr>
<tr>
<td>RPHASE.FOR</td>
<td>18811</td>
</tr>
<tr>
<td>RRAYS.B FOR</td>
<td>27181</td>
</tr>
<tr>
<td>RTILT.FOR</td>
<td>29584</td>
</tr>
<tr>
<td>ELEV.MEN</td>
<td>237</td>
</tr>
<tr>
<td>LAUNCH.MEN</td>
<td>237</td>
</tr>
<tr>
<td>MAIN.MEN</td>
<td>163</td>
</tr>
<tr>
<td>MASTER.MEN</td>
<td>47</td>
</tr>
<tr>
<td>READER2.FOR</td>
<td>3477</td>
</tr>
</tbody>
</table>

Formats available

RAYTRACE has been, to date, primarily used on a VAX under the VMS operating system, and on an AT-compatible microcomputer running MS-DOS. The program can be furnished in formats compatible with either of these two environments. It may also be possible to furnish it in other formats upon request.

PROGRAM SETUP

VMS environment

Setup and installation of RAYTRACE in the VMS environment will be made more convenient by appropriate planning of directories. What follows is one possible directory setup that has been of use while developing the program. Individual user needs and pre-existing directories will dictate the directory structuring for each site. The pri-
mary point is that the user of RAYTRACE will make usage easier by plan-
ning ahead.

The directory scheme used during development and testing of RAYTRACE involved the use of three parallel directories. A working directory was set up containing the source code. Editing, compiling, and linking are performed within this directory. When significant amounts of debugging are anticipated, the .MEN files and a trial input file are moved into the working directory, although by and large such duplication of files is avoided in order to save space.

A simulation directory is maintained for day-to-day usage of RAYTRACE. This directory holds .DAT (input) and the .MEN files, as well as the executable version of RAYTRACE. Numerous input files may be maintained by naming them in a mnemonic fashion.

The simulation directory is maintained in an uncluttered state by having a results directory to which the results of raytracings may be moved. This is particularly important because RAYTRACE simply uses generic filenames when it writes out results. If many runs are to be performed, it is to the user's advantage to rename and move the result files in order to keep them straight.

An additional directory that may be of help to some users is a versions directory. This may be a subdirectory of the working directory, to which older or different versions of the code are moved. Again, these files should be renamed in order to distinguish them at a later date. An advantage of such a subdirectory is keeping the working directory uncluttered.

The source code may now be copied into the desired directory. If the source is uploaded to the VMS system from a DOS system, some things may need to be edited before RAYTRACE will successfully compile. Some DOS-based Fortran compilers need markers at the beginning of subroutines so that subroutines will start on a new page when the compiler is asked to produce a listing file. These markers need to be removed.

Also, different compilers have different means of suppressing generation of a new line after a WRITE statement to the screen. This means that three lines of the subroutine KEYBRD need to be changed from one development environment to the next. This subroutine is located in the file RAYTRA4.FOR. Edit this file and go to the subroutine KEYBRD. There is a comment block below the variable declarations which describes the necessary changes.
RAYTRACE is now ready to be compiled and linked. If default naming is to be used, then RAYTRA4 should be listed first in the link command. The executable version may now be copied to the raytracing directory. The .OBJ files should be left intact. This way, when changes are made to one source file, only that one file need be recompiled and all the object files may be linked. If disk space is a consideration, the executable may certainly be deleted from the working directory.

READER2.FOR should also be compiled and linked. The executable may be copied to the results directory. The object and executable files left in the working directory may be deleted to conserve disk space.

The following files should now exist. The working directory should contain a complete set of both source and object files for RAYTRACE, as well as the source for READER2.FOR. The executable versions of RAYTRACE (default name: RAYTRA4.EXE) and READER2 should be in the locations that the user has chosen as appropriate. The .MEN files should be located in the same directory as the RAYTRACE executable.

The process of compiling, linking, and moving these files will be made considerably easier by making use of the capability of VMS to use .COM files. This is especially true if modifications to the programs are intended.

DOS environment

Setup and installation of RAYTRACE in the DOS environment will be made more convenient by appropriate planning of directories. What follows is one possible directory setup that has been of use while developing the program. Individual user needs and pre-existing directories will dictate the directory structuring for each site. The primary point is that the user of RAYTRACE will make usage easier by planning ahead.

The directory scheme used during development and testing of RAYTRACE involved the use of three parallel directories. A working directory was set up containing the source code. Editing, compiling, and linking are performed within this directory. When significant amounts of debugging are anticipated, the .MEN files and a trial input
file are moved into the working directory, although by and large such duplication of files is avoided in order to save space.

A simulation directory is maintained for day-to-day usage of RAYTRACE. This directory holds .DAT (input) and the .MEN files, as well as the executable version of RAYTRACE. Numerous input files may be maintained by naming them in some mnemonic fashion. This may become crucial, as with a large number of files it is entirely possible to accidentally destroy another file by attempting to have two files of the same name.

The simulation directory is maintained in an uncluttered state by having a results directory to which the results of raytracings may be moved. This is particularly important because RAYTRACE simply uses generic filenames when it writes out results. If many runs are to be performed, it is to the user's advantage to rename and move the result files in order to keep them straight. Again, this helps avoid the unpleasantness of unwanted destruction of result files.

An additional directory that may be of help to some users is a versions directory. This may be a subdirectory of the working directory, to which older or different versions of the code are moved. Again, these files should be renamed in order to distinguish them at a later date. An advantage of such a subdirectory is keeping the working directory uncluttered.

The source code may now be copied into the desired directory. If the source is downloaded to the DOS system from a VMS system, some things may need to be edited before RAYTRACE will successfully compile. Some DOS-based Fortran compilers need markers at the beginning of subroutines so that subroutines will start on a new page when the compiler is asked to produce a listing file. These markers need to be added.

Also, different compilers have different means of suppressing generation of a new line after a WRITE statement to the screen. This means that three lines of the subroutine KEYBRD need to be changed from one development environment to the next. This subroutine is located in the file RAYTRA4.FOR. Edit this file and go to the subroutine KEYBRD. There is a comment block below the variable declarations which describes the necessary changes.

RAYTRACE is now ready to be compiled and linked. If default naming is to be used, then RAYTRA4 should be listed first in the link com-
The executable version may now be copied to the raytracing directory. The .OBJ files should be left intact. This way, when changes are made to one source file, only that one file need be recompiled and all the object files may be linked. If disk space is a consideration, the executable may certainly be deleted from the working directory.

READER2.FOR should also be compiled and linked. The executable may be copied to the results directory. The object and executable files left in the working directory may be deleted to conserve disk space.

The following files should now exist. The working directory should contain a complete set of both source and object files for RAYTRACE, as well as the source for READER2.FOR. The executable versions of RAYTRACE (default name RAYTRA4.EXE) and READER2 should be in the locations that the user has chosen as appropriate. The .MEN files should be located in the same directory as the RAYTRACE executable.

There are several utilities available with some DOS-based compilers that ought to be taken advantage of if available. One of the drawbacks of most DOS Fortrans is the colossal size to which the executable file grows, relative to the size of the source or object files. A necessary utility is therefore some sort of program that compresses the executable to the size that it ought to be.

The other utility is a 'make' utility. This automates the process of building that executable by checking the files that go into constructing the executable to see which ones have changed. Compilation only occurs for those files that need it. If much alteration of RAYTRACE is to be done, a 'make' utility will save a great deal of time. If a 'make' utility is not available, then creative use of DOS batch files may serve as a substitute.
PART TWO:
OPERATOR'S GUIDE
GENERAL INSTRUCTIONS

Overall program flow

The general scheme for using RAYTRACE is much like that of other simulation programs. Input is usually from a data file, and the results are stored in output files. During the course of execution, the user is given the opportunity to change a number of parameters, allowing a variety of situations to be simulated in one session. These settings may be saved for later use. The user is also given the ability to redo entries which were made in error.

Figures 1a & 1b summarize the overall flow of RAYTRACE. When the program is started, the user may choose between doing a completely new problem, or one which has been done before and therefore has an already existing data file with options set. If an old problem is to be worked on, all of the problem's option settings and ionospheric data are loaded in and the user is deposited directly into the main menu. If a new problem is being performed, the ionospheric data must be input. This will usually be by means of a grid file. A grid file is simply a file containing the pertinent ionospheric information at a gridded set of latitudes and longitudes. Grids may be entered from the console, but this is only practical for relatively small grids. Again, after the ionospheric data is entered, the user is placed in the main menu.
FIGURE 1a.

Run RAYTRACE

Initial prompt: new or old problem?

Specify raytracing (input) data file

Specify input ionospheric grid file

Specify grid params. for console input

Specify ionospheric params. at each grid point

Specify S.S.# and time params.
FIGURE 1b.

Main Menu

Item 1: Launch params. menu

Items 1-8: enter ray launch params.

Item 2: Elev. & azi. params. menu

Items 1-7: enter elev. & azi. params.

Item 3: Perform raytracing

Prompt: Do another problem?

Item 4: Exit RAYTRACE

Change mind about ending?

END
The main menu of the program is a central point to which execution will return unless the session is explicitly ended. There are two menu items which lead to other menus. These are for the entry of parameter settings. The other selections are to execute the raytracing using the current parameter settings, and to exit from the program. All of this will be discussed in greater detail below.

During execution, RAYTRACE must deal with a number of other files. The general relationship of the program and these files is summarized in Figure 2. The file GRID.DAT is a grid file, as described briefly above. It contains the ionospheric specification for a given situation and is used as input on fresh problems. This file must be generated by some outside means, although if the specification is entered into RAYTRACE by hand, a GRID.DAT file will be produced. The RAYDAT.DAT file is the file which contains the settings from the previous sessions, as well as the ionospheric specification. The four files with the .MEN suffixes determine the appearance and text of the menus. Results are output into files of the type FOR##.DAT, where ### starts at 040 (This choice is an artifact of earlier versions.) and increments each time a new raytracing is done within a given session.

FIGURE 2.
Ionospheric input

One of the primary benefits of RAYTRACE is the ability to do realistic three-dimensional raytracing with a complicated ionospheric specification. This capability is implemented by means of the grid file(s) described briefly above. The ionospheric grid file may be produced by a numerical model, or the appropriate parameters may be obtained from experimental data and massaged into a grid file. A grid file may also be produced by direct keyboard entry of ionospheric parameters into RAYTRACE.

The grid file, however it is produced, is composed of a number of data arrays. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. Finally, other pertinent data are recorded. The six parameters are the maximum plasma frequencies for the E, F1, and F2 layers, the heights for the maximum frequencies for the F1 and F2 layers, and the semithickness of the F2 layer. The choice of these parameters is determined by the profile model that is built into the current version of RAYTRACE. Currently, RAYTRACE uses the vertical ionospheric profile contained in the RADAR-C model [2]. Because the raytracing is so intimately mated with this model, any input data coming from experiment or some other model must be converted into this form or RAYTRACE must be modified to accommodate the new model profile. The full details on the composition of a grid file are found in Appendix A.

Operating instructions

Before beginning operation of RAYTRACE, be sure that the instructions in the setup section above have been followed. Also, be sure that any input data that are desired exist in a grid file. RAYTRACE may now be executed in the manner appropriate to the system it is operating on.

When RAYTRACE is first run, the following prompt appears:

"Is this a new (1) or old (0) problem? (0): ".

18
This prompt is typical of those used in RAYTRACE and so it is appropriate to discuss it here in some detail. First, the prompt displays some text which describes the information sought. Any non-intuitive form of response, such as the 0 or 1 above, or other information which will in some way limit the acceptable response, is displayed as part of the prompt text. After the prompt text, the current (or default) value is displayed in parenthesis. The colon marks the end of the prompt, after which the user types his response. If the user merely types a return, then the default value is used as the response. REAL-valued responses may be typed without the decimal point if there is no fractional part to the response, and character responses are case-insensitive.

The above prompt is used to determine whether or not this RAYTRACE run is for an entirely new data set, or a previously used data set. A new problem is simply one for which no RAYDAT-type file exists, or for which no previously existing files of this type will be used. This type of file has been mentioned briefly already and will be discussed more fully below. By selecting the option for a new problem the user is therefore able to start either a genuinely new problem, or use an old ionospheric specification with all new options. An old problem, on the other hand, is one for which the ionospheric specification and the options have been stored in a file of the RAYDAT type. Please note that a response to this prompt MUST be TYPED, i.e. the default cannot be accepted. This is an attempt to ensure that the operator will make the correct choice.

The next prompt that comes up is the following:

"Enter filename for storage of ionospheric
information (10 char max): (RAYDAT.DAT): ."

This prompt asks for the name of the file which will contain both the ionospheric specification and the parameter options settings used in the particular problem. If the problem is a new one and the file given already exists, a Fortran error message will occur, the content of which will depend upon the compiler used. Aside from this, the response to this prompt is straightforward.

The program will now proceed to the main menu if the problem has been declared to be an old one. However, the ionospheric specification still needs to be obtained if this is a new problem. The next prompt asks for the name of the file containing this ionospheric information.
"The ionospheric grid file name is? (type NONE if none exists) \text{(GRID.DAT)}: ".

The data is retrieved from the grid file whose name is given and the program then proceeds to the main menu. If 'none' is typed (remember, the response is case-insensitive), the program then executes a section of code to read the appropriate information from the console.

Console entry of the ionospheric specification involves three sets of prompts. The first set of prompts asks the user to provide the necessary parameters for specifying the locations of the grid points. The prompts are:

\begin{verbatim}
"Input lat grid spacing (deg): \text{(0.000000000)}: \\
Input lon grid spacing (deg): \text{(0.000000000)}: \\
Input starting latitude and longitude in degrees. \text{LATITUDE}: \text{(0.000000000)}: \\
LONGITUDE \text{(east = positive)}: \text{(0.000000000)}: \\
Input # of grid points in lat.: \text{(0.000000000)}: \\
... in lon.: \text{(0.000000000)}: \\
Grid setup OK (Y/N) ? \text{(Y)}: ".
\end{verbatim}

The first two queries set the latitude and longitude spacing between grid locations. The next two fix the location of the south-west corner of the grid. Longitudes are measured from Greenwich with positive values eastward and negative values westward, so that the range is between -180 and +180 degrees. The last pair of queries gives the extent of the grid in increments of the grid spacing in the appropriate direction. One word of warning is appropriate here. Don't forget about counting BOTH end points when entering the number of grid points: Finally, the user is asked if the values entered are correct. If so, the execution of the program goes on. If not, the user is returned to the beginning of this set of questions and allowed to repeat the data entry. The current settings will be shown as the default values which eases the correction of the errant entry.

The next set of prompts is used to obtain the actual vertical profile data at each grid location. The location is displayed and the user is prompted to enter each of the six values needed to specify the vertical profile used with this version of RAYTRACE, as follows:
The values requested are the plasma frequencies (squared) for the three ionospheric layers, the E, F1, and F2, at the layer maximum. These are given in units of MHz-squared. The heights of the maximum plasma frequency for the F1 and F2 layers are given in kilometers as is the value of the semithickness of the F2 layer. As with the previous set of prompts, if the user acknowledges that the inputs are OK, the program will continue on. If the values entered are not correct, the user is given the chance to reenter them. These profile entry prompts continue to come up on the console until all grid locations have been exhausted.

The final of the three sets of prompts asks for the time of the simulation and the sunspot number:

```
" Input (integer) sunspot number: ( 0): 
Input year: ( 0): 
... month: ( 0): 
... day: ( 0): 
Input UT time (hr): ( 0): 
... UT time (min): ( 0): 
S.S. # and times OK (Y/N)? ( Y): ".
```

The sunspot number is the Zurich sunspot number. The year, month, and day are given as their usual integer values. The time is Universal Time and is integer also. After this prompt, the entry of the ionospheric specification is complete and the program proceeds to the main menu.

By whatever route, the program will now have reached the main menu. The main menu serves as the hub of the program as far as the user is concerned. All major actions that may be performed are done from the main menu. The menu contains four items:
MAIN MENU

1 - Edit launch parameters
2 - Edit elevation/azimuth parameters
3 - Proceed with raytracing
4 - Quit program

Your choice (1-4 only, please)? (0):
Press RETURN to continue.

This exhibits the typical structure of a menu in RAYTRACE. The menu is titled, and the items are given with corresponding numbers. Selection is made by entry of an item's number. The entry is checked and if it is out of bounds for that menu, the user is prompted for the choice again. The 'Press RETURN to continue.' prompt is self-explanatory, except for the opportunity it affords the user to abort the previous action. If the user types 'a' (for abort), the menu choice will be aborted and the menu will be presented again.

The first two items of the main menu are used to invoke menus for the entry of parameter values for the particular problem being done. These values are divided up into basically general values in the first menu and elevation/azimuth values in the second. The third and fourth items are self-explanatory.

Selection of item 1 of the main menu brings the Launch Parameters menu to the screen. The parameters which are entered by means of this menu are of a general sort. They include the starting location of the ray, the properties of the ray, and the conditions for cutting off the ray calculation. The Launch Parameters menu is:
LAUNCH PARAMETERS MENU

1 - Bounce limit
2 - Signal intensity
3 - Conductivity
4 - Launch point
5 - Launch height
6 - Range & height limits
7 - Ray path increment
8 - Wave frequency
9 - Done

Your choice (1-9 only, please)? ( 0):
Press RETURN to continue. ( 0): ".

The first item on the menu is the bounce limit. Selection of this item brings up the following prompt:

"Bounce limit ( 1): ".

When the ray propagation represents traditional HF communications type raypaths, i.e. rays which are reflected from the bottom side of the ionosphere and which return to earth, it is convenient to place a limit on the number of times that a ray can return to earth. This may be a maximum of 10 currently. In essence, this provides for a kind of fuzzy range cutoff to supplement the hard range cutoff which will be discussed later. Pertinent values for the ray are output to a file at bounce (earth impact) points and at the end of the ray. This item is somewhat meaningless for raypaths which are not restricted to the space between the bottom of the ionosphere and the earth's surface. One artifact of this is the blank record which may appear at the end of the results file for certain raypaths. This blank record is completely harmless and, to use current terminology, it's not a bug, it's a feature.

The next two items deal with the signal intensity calculations that will be done along the ray. The prompt for item 2:

"Do signal intensity (no = 1)? ( 0): ".

23
is used to enter a flag which turns the calculation of signal intensity on and off. Currently, the signal intensity computation that is performed is the geometric spreading of a ray bundle, including focusing and defocusing effects, and reflection loss upon earth impact. The third prompt:

"Reflect from ground(1) or water(0)? ( 0): ".

is used to determine the conductivity of the reflection surface used in the calculation of reflection loss. At present this is a global choice.

Menu selections 4 and 5 determine the geographic location and altitude of the starting point of the ray. The latitude and longitude are entered at the prompt:

"Enter launch pt. latitude ( .000000000 ):
Enter launch pt. longitude ( .000000000 ):

The values are entered in degrees with the longitude ranging from -180 to +180 degrees, positive values to the east of the prime meridian. The starting height is entered at the prompt:

"Enter starting ht. (km) ( .000000000 ):

This is the height of the starting point above the surface of the earth, defined as the spherical surface having a radius equal to the average earth radius.

Item six provides for the entry of cutoff criteria for the ray-tracing problem:

"Enter range limit (km) ( 3000.00000 ):
Enter altitude limit (km) ( 36000.00000 ):

The range in the range limit prompt is great circle range at the earth's surface. It should be thought of as an angular range along the great circle direction defined by the original azimuth of the ray, since this is how it is handled internally. By thinking of this limit as an angular one, it may be readily generalized to problems for which both end points of the ray are above the earth's surface. The altitude limit simply sets a cutoff altitude above the surface of the earth. The default value of 36000 km represents the altitude of geosynchronous satellites.
Selection seven of the Launch Parameters menu allows the user to set the size of the steps taken along the ray as its propagation is modeled. The prompt is as follows:

"Enter raypath increment (km) (4.00000000): ".

Settings for this represent a compromise between accuracy of the results and speed of computation. The default value of 4 km represents a rough optimum. Practically, values much smaller than 0.1 km cause the computation to take much too long and sufficiently small values can begin to cause a decrease in accuracy due to accumulation of roundoff errors from all over the program. Large values are limited by the theory underlying RAYTRACE, and should generally be less than 10-15 km.

The eighth menu item sets the frequency of the electromagnetic wave that the ray represents. The prompt is:

"Enter wave frequency (MHz) (5.00000000): ".

Because effects due to the earth's magnetic field are not implemented in the current version of RAYTRACE, use of frequencies much below about 2 or 3 MHz is questionable. At the high end, frequencies beyond a few GHz begin to cause inaccuracies because of repeated operations involving one minus a small number.

Selection of item nine in the Launch Parameters menu returns the user to the main menu. If the user has decided to alter any of the parameters set in the Launch Parameters menu, the menu may simply be reentered at this point and the values changed. Parameter settings for the raytracing may be changed over and over, the ray will be traced using the current settings.

The second item in the main menu invokes the Elevation Parameters menu. This menu has eight items:
ELEVATION PARAMETERS MENU

1 - Dimension of problem
2 - Starting azimuth
3 - Starting elevation
4 - Elevation resolution
5 - Azimuth resolution
6 - Elevation limit
7 - Azimuth limit
8 - Done

Your choice (1-8 only, please)? ( 0):
Press RETURN to continue. ( 0 ) : ".

This menu works in an identical fashion to the Launch Parameters menu, but with one twist. It will become clear from discussion of the individual menu items that some of them simply do not apply in all circumstances. In cases where this is true, those items which don't apply are disabled. Attempting to select such a disabled item will return the user to the menu.

Selection one of the Elevation Parameters menu allows entry of a quantity that is known as the dimension of the problem:

" Enter dimension of problem ( 0.0000000 ) : ".

The dimension may have values of 0, 1, or 2. The dimensionality is that of the pattern of rays to be sent out by RAYTRACE. A single ray may be sent (dimension 0). However, multiple rays may be sent in a single run. A vertical fan of rays may be specified, which is to say that the program may be instructed to step in elevation for a fixed azimuth. This is dimension 1. The program may be instructed to step both in elevation and azimuth. This is dimension 2. The aimpoints of the ray(s), if pictured on an elevation versus azimuth plot, then form either a one or two dimensional array, or a single point.

Continuing with this visualization, the array is taken to start in the lower left corner, that is, the lowest value of both elevation and azimuth. For a single ray, the array of values is simply restricted to just this single point and the dimension one problem uses the leftmost
column of the array. All cases require the input values for the starting corner of the array of elevation and azimuth values. Menu items two and three handle this:

"Enter starting azimuth (deg) \(0.000000000\): ",

and,

"Enter starting elevation (deg), use neg. for values >90 deg from zenith. \(0.000000000\): ".

The azimuth is given in the conventional manner. As the prompt for starting elevation states, elevation angles below horizontal are allowed. Acceptable values range from -90 degrees to +90 degrees, permitting rays which have their origin at altitude to be sent downward.

The next menu item, item 4, is used to enter the elevation spacing between rays.

"Enter elev. resolution (deg) \(0.000000000\): ".

This only applies when a multi-dimensional array of rays is being considered, and so may only be selected when the problem dimension has been declared to be one or two. The elevation values are stepped upward (i.e. toward increasing elevation) by this value. Related to this is item six, where the limiting value for the elevation is given. This prompt,

"Enter elev. limit (deg) \(0.000000000\): ",

is used to define the extent of the pattern of rays in elevation. As with the previous prompt, this may only be selected when the problem is declared to be of dimension one or two.

The fifth prompt is used for entry of the azimuthal spacing in the pattern of rays to be sent out. The prompt is:

"Enter azimuth resolution (deg) \(0.000000000\): ".

This prompt is coupled with the seventh prompt,

"Enter azimuthal limit (deg) \(0.000000000\): ",

which provides the program with the information on the azimuthal extent of the ray pattern. These selections only apply for the case when the dimension of the problem is two.

Finally, the eighth item of the Elevation Parameters menu will return the user to the main menu when selected. As with the Launch
Parameters menu, the user may continue to change the values entered in
the Elevation Parameters menu until they are satisfactory. These are
then the values which will be used in the raytracing.

The main menu's third selection will cause the actual computation
of the ray(s) to begin. The values entered in the Launch Parameters
and the Elevation Parameters data entry menus are the values that will
be used. At this point, if this is the first time through a new
problem, the parameters and ionospheric data are written out to the
RAYDAT-type file and the ray computation begins immediately. The first
time through an old problem, the user is prompted:

" Overwrite existing file (Y/N)?   (        ) : ".

about overwriting the old parameter values with the new ones. If not,
the user is asked:

" Backup the datafile (Y/N)?   (        ) : ".

whether a backup copy (another RAYDAT-type file) of the settings is
desired. After answering these questions, the ray calculation begins.
For either case, after the initial time in a problem, every succeeding
time the user is asked:

" Update the datafile (Y/N)?   (        ) : ".

whether or not the data file should be updated with the latest set-
tings. The interface is set up to give the user ample opportunity to
preserve the new parameter settings if desired, but the user has the
choice of not saving these settings if that is what is desired.

The user is informed of the start and finish of the ray calcula-
tions by the messages:

" BEGINNING RAY LOOPS.

RAY LOOPS DONE. ".

Upon completion of the computations, the user is informed of the file
that the results have been written to by the following message:

" Results written to file FOR040.DAT ".

The numeric part of the filename will increment each time the user does
another raytracing in a single session. Leaving the program resets the
numeric portion of the filename.
Now that a particular raytracing is done and the results are stored away, the user is prompted:

"Do another problem (Y/N)? ( ): ",

whether or not additional raytracing is desired. An affirmative answer will return the user to the main menu so that any changes to parameters may be made and more rays may be traced. A negative answer has the same effect as choosing item four from the main menu. First, the prompt:

"Press RETURN to continue. ( ): ",

is issued, allowing the user to abort the exiting of the problem by entering 'a'. If a return is entered, then the program exits with the following final message

"Session done. For safety's sake, copy the data and result files into a separate directory to prevent accidental overwriting. ".

This warning should be heeded. For use on VAX/VMS systems, a large number of virtually indistinguishable files with the same names and different version numbers may result in loss of results through confusion over which version is which. DOS based systems present an even greater problem, because the attempt to save results to a file already in existence will cause the old file to be destroyed.

Output Files

RAYTRACE produces three types of output files. The first is a GRID.DAT type of file, if the ionospheric specification has been entered by hand. This file has been discussed already. The second type of file is the RAYDAT.DAT type. This file is essentially just a GRID.DAT file with the additional user-entered parameters appended to it. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. Other pertinent data are recorded, including the date and time of the ionospheric specification, and the sunspot number. These previous data are just those recorded in the GRID.DAT file. Addition-
ally, the values from the Launch Parameters menu are recorded and finally the values from the Elevation Parameters menu are stored. Full detail of the RAYDAT.DAT type of file is given in Appendix A.

The remaining type of data file is the results file, FOR###.DAT. This file contains the results of a single run of RAYTRACE. First, the bounce limit (plus one for the end point of the ray), the number of azimuths, and the number of elevations are recorded. This enables any program reading the results to break the results out into those for individual rays. For each ray, the azimuth and elevation are recorded. Then the various results are saved for each data point along the ray. These data points are earth impact points and the end point. If the ray is terminated before the bounce limit is reached, blank records will occur and should be ignored. Each succeeding ray is recorded in a similar manner: first, azimuth and elevation, and then the results. The details of this particular file type are presented in Appendix A.

It should be restated that the user ought to be careful and develop defensive file maintenance procedures. After each session files should at the least be given more suitable names that are perhaps indicative of their contents. This will not only reduce the likelihood of accidental destruction of results, but will aid in later analysis. Also, the user should be aware of when the files get written to disk. The RAYTRACE produced GRID.DAT file has its contents written during the entry of the data. The RAYDAT.DAT file is written just before the rays are traced, and the FOR###.DAT (results) file is written during the raytracing.
Although the operation of the various commands of RAYTRACE has now been discussed, this knowledge will be reinforced by going step by step through some examples. These examples will be presented with actual samples of screens, for clarity. The first tutorial involves setting up RAYTRACE to use a spherically symmetric ionosphere with the specification entered by the console. This will create the data files used in the subsequent two tutorials. The second is to perform a new problem using a grid file as input. The last is to perform an old problem. It is hoped that the user will actually perform these tutorials on his/her system, even if operation of the program seems obvious by this time. At the least, the exercise will increase familiarity with the operation of the program, while potential problems or questions may be preempted by use of these tutorials.

New session using keyboard input

To begin this tutorial, run RAYTRACE. This will be a new problem, so respond with '1' to the first prompt. The next prompt will ask about the file to which all the ionospheric and parameter information will be written. The filename that has been chosen for this tutorial is TRIAL.DAT. At this time the screen should look like the following.
Is this a new (1) or old (0) problem? ( 0): 1

Enter filename for storage of ionospheric information (10 char max): (RAYDAT.DAT):trial.dat

The next prompt is the prompt which asks about the file from which the ionospheric specification will be read. For this example, there is none, so the appropriate response is 'none'. The screen that corresponds to this is Screen 2.
The ionospheric grid file name is?  
(type NONE if none exists)  
(GRID.DAT ):none

The next set of prompts that comes up is a set of prompts asking about the definition of the grid on which the ionosphere is to be specified. The first two prompts deal with the latitude and longitude spacing between the grid points. Since this example is a spherically symmetric case, only one grid point will be needed and so the spacing is immaterial. For this reason, a response of '1' is chosen in each case. The next two prompts ask for the starting latitude and longitude for the grid, in this case this is the location of the only grid point. The responses chosen are '45' for latitude and '-90' for longitude. This longitude corresponds to a west longitude. The final two informational prompts here ask about the size of the grid in the latitude and longitude directions. Since there is to be only one point, the number of points in each direction will be '1'. Finally, there is a prompt asking whether the input is correct. Assuming everything has been entered as shown, then typing return will accept the default choice of 'Y'. At this point, the screen should look like Screen 3.
Input lat grid spacing (deg): ( .000000000 ):.1
Input lon grid spacing (deg): ( .000000000 ):.1
Input starting latitude and longitude in degrees.
LATITUDE : ( .000000000 ):45
LONGITUDE (east = positive): ( .000000000 ):90
Input # of grid points in lat.: ( .000000000 ):1
... in lon.: ( .000000000 ):1
Grid setup OK (Y/N) ? ( Y ):

Now the program begins to prompt for the ionospheric information for the location specified above. The location of the point is given at the top of the prompt. The parameters prompted for occur ordered in terms of height. It should be noted that the values given here are merely fictitious. First is the plasma frequency squared for the maximum of the E-layer, with a value to be entered as '2'. Next the F1-layer is specified, by the height of the maximum of the F1, '150', and the plasma frequency squared of the F1 maximum, '7'. Finally, three F2-layer parameters are prompted for. They are the semithickness of the F2, '65', the height of the F2 maximum, '300', and the plasma frequency squared at the maximum of the F2, '20'. Again, the program prompts to see if all the input is OK. If it is, the response should be the same as for the similar prompt above. The console screen should now look like Screen 4.
The final set of prompts for the environmental specification involves recording the date and time of the problem, and the sunspot number. First the sunspot number is entered, in this example '80'. Next, in descending order of time scale, come the date and time prompts. As indicated by the defaults given, these are expected to be integers. This example uses the following responses in order: '1988', '2', '17', '12', and '00'. After the verification prompt, the screen should look like Screen 5.
At this point, the program brings the user to the main menu. It is from this menu that other menus are invoked for entry of the remaining parameters, and that the actions of tracing rays and quitting are taken. The parameters are currently a blank slate, so both of the data entry menus need to be selected and the parameters entered. Starting at the top, select menu '1'. The program will respond with another prompt. As it says, if the selection is the desired one, the user needs only type a return to proceed. As was mentioned earlier, though, the selection may be aborted by responding with an 'a'. If this is done, the user will be returned to the previous menu. Before typing return, the screen will look like Screen 6.
Selection of item one of the main menu brings the user to the Launch Parameters menu. This menu is used for the entry of various parameters relating to the ray calculation. These values include the location of the starting point for the ray(s), settings which are used to halt the calculation, properties of the ray(s), and values which determine whether some calculations are performed and if so, how. This example will simply make the selections from the menu in numerical order, therefore, the first entry will be '1'. This situation is reflected in Screen 7.
Screen 7

LAUNCH PARAMETERS MENU

1 - Bounce limit
2 - Signal intensity
3 - Conductivity
4 - Launch point
5 - Launch height
6 - Range & height limits
7 - Ray path increment
8 - Wave frequency
9 - Done

Your choice (1-9 only, please)? ( 0):1
Press RETURN to continue.

This selection from the Launch Parameters menu brings up a query about the limit to be placed on how many times a ray may return to earth. This is useful when dealing with low frequencies which will not penetrate the ionosphere. The problem then may be set up to terminate after a certain number of hops has occurred, regardless of the other termination conditions. For the purpose of this example, accepting the default value of '1' will be sufficient, so the response will be to just type a return. The screen should look like Screen 8. Note that in some of the screens that follow, only the lower portion of the screen is shown.
Once the entry of the data is made, the user is immediately returned to the Launch Parameters menu. This time the selection will be item '2'. Because the process by which items are selected from menus is uniform, the screen need not be repeated. Menu item two leads to a prompt which allows the program to know whether or not to perform signal intensity calculations. For the sake of speed in running the example, the value entered is '1' which corresponds to no. The screen example is given as Screen 9.

Returning to the Launch Parameters menu, the next selection made will be '3'. This item allows the user to designate a global value for the conductivity of the earth's surface. These values correspond to soil and water. They are average values. Conductivity values are used when signal intensities are calculated so that the loss due to reflection from the earth's surface may be accounted for. Since no signal intensity will be calculated for this example, this item was selected for instructional purposes only. Because any answer will suffice, the default may be accepted by typing return. This is shown in Screen 10.
Reflect from ground(1) or water(0)? (0):

The next selection from the Launch Parameters menu will be item number '4'. This prompts the user for the latitude and longitude of the launch site of the ray(s). The values are given in degrees. Remember that longitudes are to be given as positive to the east of Greenwich, with the prime meridian being 0 degrees and west longitudes negative. This gives a possible range of longitudes of -180 degrees to +180 degrees. The launch site that is used is south and east of the ionospheric point, with latitude '40' and longitude '-80'. This results in Screen 11.

Screen 11

Enter launch pt. latitude (0.000000000): 40
Enter launch pt. longitude (0.000000000): -80

Following this, item '5' should be selected from the Launch Parameters menu. The purpose of this item is to provide the third component of the location of the ray launch site, namely the starting height. The starting height is given in kilometers above the earth's surface. This exercise will start on the earth's surface, so the response that is needed is to simply type return so that the default value shown is accepted. Before hitting return, the screen will look like Screen 12.
Upon return to the menu, select item '6'. This set of prompts is used by the program to obtain from the user values which will determine when the problem should be terminated. The first parameter is the range limit. This represents great circle range on the earth's surface. Internally this is dealt with as an angular range, but it is easier to think in terms of ground range when entering the value. This must not be confused with the distance traveled along the raypath! The distance along the ray is, in general, longer than the associated ground range. The value selected for this example is '2000' km. The second value to be entered is the altitude cutoff. This gives the height above the surface of the earth of a spherical shell which, when pierced, causes the raytracing to terminate. For this example a reasonable value is '400' km. The resulting screen is Screen 13.

The next quantity to be entered is the raypath increment, selection '7'. This value determines how finely the raypath is broken up for the purposes of the calculation. There is an interplay here of the desire to make the path exceedingly fine for the sake of accuracy, with the desire that the raytracing complete in a finite (and preferably small) amount of time. The appropriate range of values has already been discussed. It is sufficient that the default value of '4' km
represents somewhat of a balance and so this is the value chosen. The prompt is displayed in Screen 14.

**Screen 14**

Enter raypath increment (km) (4.00000000):

The final important selection from the Launch Parameters menu is item number '8', the wave frequency. This is the value of the frequency of the radio wave that the ray represents. The appropriate range of values for this is discussed above in the operating instructions section. For the purposes of this tutorial, it is desired that the ray not penetrate the ionosphere, so a wave frequency of '4' MHz is selected. The default of five MHz is merely a placeholder so that the program doesn't accidently start with frequency equal to zero and promptly crash due to division by zero. The screen containing this prompt is represented by Screen 15.

**Screen 15**

Enter wave frequency (MHz) (5.00000000): 4

After completing this, all the necessary data has been entered in the Launch Parameters menu. The user may therefore enter '9', the selection indicating that the data entry is finished in this menu. The control of the program is returned to the main menu. From the main menu, the other data entry menu may now be entered. This is done by making selection '2' from the main menu. The Elevation Parameters menu now comes on screen.

As with the Launch Parameters menu, this tutorial will now proceed through the items in the Elevation Parameters menu. The first item in this menu deals with the array of azimuth and elevation values for which rays will be sent out. Select '1' as in Screen 16.
Screen 16

ELEVATION PARAMETERS MENU

1 - Dimension of problem
2 - Starting azimuth
3 - Starting elevation
4 - Elevation resolution
5 - Azimuth resolution
6 - Elevation limit
7 - Azimuth limit
8 - Done

Your choice (1-8 only, please)? (0): 1
Press RETURN to continue.

This selection will provide a prompt asking as to the dimension of the problem. This is defined earlier, however a brief rundown is in order. The term dimension refers to the dimension of the array that could be used to represent the elevation and azimuth values to be used. For this reason, dimension 0 represents just a single value of elevation and azimuth. Dimension 1 represents a single azimuth value and multiple elevation values. Finally, dimension 2 corresponds to the case where there are multiple elevations and azimuths to be used. So that all the menu items may be selected, a value of '2' should be entered here. This is represented by Screen 17.

Screen 17

Enter dimension of problem (0.00000000): 2
As usual, entering the response above returns the user to the menu. Proceeding down the menu in order, the next selection is '2'. This item allows the entry of the initial azimuth value for the problem. The azimuth is given in degrees and is taken relative to geographic north in the usual manner. Because this example involves only a spherically symmetric ionosphere, one direction is as good as another. The default value of '0' is therefore taken, by typing a return. This situation is depicted in Screen 18.

**Screen 18**

Enter starting azimuth (deg)  

Item three of the Elevation Parameters menu is the next one selected. This prompts for the starting elevation angle. Elevation is measured in degrees, with the zenith as +90 degrees, and the nadir as -90 degrees. Negative values of elevation angle come into play for cases when the launch point of the ray is at a non-zero altitude. For this example, a reasonable value to use is '10'. Screen 19 shows this.

**Screen 19**

Enter starting elevation (deg), use neg. for values >90 deg from zenith.  

The fourth selection from the Elevation Parameters menu deals with the elevation separation between rays. Very little coaching can be given about appropriate values for this parameter. The fineness or coarseness of the elevation resolution must be determined in terms of what is appropriate for a given problem. Since this problem is of an instructional nature, an arbitrary value of '0.5' degrees has been chosen. This is shown in Screen 20.
Next in line is the fifth menu item of the Elevation Parameters menu. The azimuth spacing between rays is entered at the resulting prompt. This is handled in a manner identical to that of the previous prompt. The value which is to be entered is the same as that for elevation, namely '0.5'. Screen 21 shows this.

The method of specification for the elevation and azimuth values depends upon having the starting value, the division size, and the ending value. The ending value for elevation is given by selection of item six from the Elevation Parameters menu. Internally, the difference between the start value and the limit is taken. The number of increments of size equal to the resolution value that will fit into this interval is then calculated. This is the number of elevation steps that will be taken. For the sake of completing the example problem quickly, the limit will be taken to be '10.5' so that rays will be sent out at only two elevations. The response is shown in Screen 22.
Lastly, menu item seven is selected. This allows the azimuth limit to be entered in a manner identical to the elevation limit. Again, the value is chosen so that only two values of azimuth will be used. The combination of these elevation and azimuth limits result in just four rays being calculated. The appropriate value to use for the azimuth limit is then '0.5'. It should be remembered in setting the both of the limits that if a specific number of rays is desired in elevation and in azimuth, that the end points must be counted properly. The prompt and response are shown in Screen 23.

All of the necessary parameter values have been entered now. Selection of item '8' of the Elevation Parameters menu causes a return to the main menu. Everything is now ready for the raytracing to be performed. Therefore, item '3' is now selected from the main menu. Because this is a fresh problem, no prompting about the disposition of the entered parameters is necessary. These values are automatically written out to the RAYDAT-type file that was specified at the beginning of this exercise, namely TRIAL.DAT. The user is notified that the raytracing has begun. When the calculations are complete, the user is notified and the file to which the results have been written is also noted. The user is then given the opportunity to do another raytracing problem. In order to give a feel for what the program will do, the
answer to this prompt is 'y' for yes. The screen should look like that shown in Screen 24.

Screen 24

BEGINNING RAY LOOPS.
RAY LOOPS DONE.

Results written to file FOR040.DAT

Do another problem (Y/N)? y

The affirmative response causes RAYTRACE to return to the main menu. For the second problem, just a single ray will be traced. To do this, select '2' from the main menu in order to proceed to the Elevation Parameters menu. Once at the Elevation Parameters menu, choose '1' to change the current setting for the dimension of the problem. Recall that a problem of dimension zero is a single ray problem. The prompt that comes on the screen for the dimension of the problem contains the current value, which is 2. Enter a value of '0'. This is shown in Screen 25.

Screen 25

Enter dimension of problem 2.00000000 :0
It is important to note that no other values in the Elevation Parameters menu need be changed. In fact, it will be very instructive to attempt to change any of the values corresponding to menu items 4-7. The program will not allow these items to be changed, instead the user is returned to the menu. These items are irrelevant to the calculation once the dimension has been set to zero. Because the purpose of this second portion of the problem is simply to do a second portion to the problem, this one change will suffice. Therefore, select '3' to return to the main menu, and then select '3' to proceed with the raytracing.

This time, before the raytracing is started, RAYTRACE notes the possibility that the user has made changes to parameter values. The user is asked whether or not to update the data file. If the answer is yes, the current values are saved out to the RAYDAT-type file. If the answer is no, then if the user chooses to end the session after the current raytracing, the changed parameter values disappear. For the purposes of this example, it doesn't matter whether the values are saved or not: the response of 'n' has been chosen. This may be seen in Screen 26.

**Screen 26**

Update the datafile (Y/N)? (n)

The program then signals that the tracing of the ray has begun. When the end of the calculation is reached, the display is similar to that seen before. This time, however, it is desired to end the problem. The response to the prompt is therefore 'n'. The screen will look like Screen 27.
Update the datafile (Y/N)? n

BEGINNING RAY LOOPS.

RAY LOOPS DONE.

Results written to file FOR041.DAT

Do another problem (Y/N)? n
Press RETURN to continue.

The final thing that the program does is to display a message. This message is to warn the user to take proper care of any data files that have been produced during the raytracing session. The point cannot be overemphasized that the filenames given are far from descriptive and results may get lost or destroyed if a myriad of such files exists. The closing message is reproduced in Screen 28.

Screen 28

Session done. For safety's sake, copy the data and result files into a separate directory to prevent accidental overwriting.

At this point, RAYTRACE has finished execution and control has returned to whatever system is being used. Several new files should now be in existence. TRIAL.DAT is the file of the RAYDAT type that contains the saved parameter settings and the ionospheric specifica-
tion. A grid file called GRID.DAT was created by entering the ionospheric data from the console. There are also two results files, FOR040.DAT and FOR041.DAT. As the final message warns, these files should either be moved or renamed. Examining the results files is discussed in a later section.

New session using data file input

This second example will build upon the foundation of the first. The grid file that will be used for input is going to be the GRID.DAT that was produced in the previous example. Reference will also be made to the first example for those portions of the two examples which are identical. It is expected that many RAYTRACE runs will be similar to the current example in that most fresh problems will be initiated by use of an input file specifying the ionosphere.

This example begins by starting RAYTRACE as before. The initial prompt comes up on the console. Since this example is to be a new problem, respond with '1'. The prompt asking for the name of the file to store the ionospheric specification and input parameters now appears. Recall that last time the file TRIAL.DAT was used. If this file hasn't been relocated or renamed, entering this name again may cause a problem. So that the files may be distinguished the name that is chosen for this example is 'TRIAL2.DAT'. After this response, the screen should look like Screen 29.

Screen 29

Is this a new (1) or old (0) problem? ( 0):1
Enter filename for storage of ionospheric information (10 char max): (RAYDAT.DAT):trial2.dat

The next prompt is the one asking for the file containing the ionospheric specification. Unlike the first example, one now exists. The file even has the name GRID.DAT. This is the file to be used, so
the default value of the prompt may be accepted by typing a return. This situation is depicted in Screen 30.

Screen 30

The ionospheric grid file name is?
(type NONE if none exists) (GRID.DAT):

The user should now notice a difference from the previous example. Because the ionospheric specification already exists, the program has no need to prompt the user for keyboard entry of the ionospheric parameters. RAYTRACE immediately proceeds to the main menu. From here the previous example may be picked up again. Menu item '1' from the main menu should be chosen. The user should now enter values for each of the items in the Launch Parameters menu. The values to be used are just those of the previous example. That example and its discussion may be followed and will not be repeated here.

Following completion of the Launch Parameters menu, the next step is to proceed to the Elevation Parameters menu by selecting '2' from the main menu. Again, the items of this menu will be completed in numerical order. Just as in the first example, enter '1' as the menu choice. For the sake of speed, just a single ray is desired. Therefore, the response to the resulting prompt will be to accept the default value by typing a return. This is shown in Screen 31.

Screen 31

Enter dimension of problem (0.00000000):

Next, choose item '2' from the Elevation Parameters menu. The value is chosen to be the same as in the previous example. Recall that this means the azimuth value will be 0 degrees. The default response is therefore to be accepted. Screen 32 show this.
The starting elevation value is the next to be set. Menu item '3' should be selected. At the resulting prompt, a value of '10' degrees should be entered. This ought to lead to a ray with identical results as one of those done in the first example. The results may be compared as a check. The elevation data entry is shown in Screen 33.

**Screen 33**

Enter starting elevation (deg), use neg. for values >90 deg from zenith.   

Because the dimension of the problem has been selected to be zero, there is no further data entry needed. Item '8' should therefore be selected from the Elevation Parameters menu. Upon return to the main menu, the user may choose item '3' as in the previous example. This will initiate the raytracing without any intervening prompts. Because this is a fresh problem, the data is automatically recorded in the RAYDAT-type file specified at the start. When the raytracing is complete, the program behaves in a manner exactly identical to that shown in the first example. For the purpose of practice, answer the prompt for another problem with 'y'. This is shown in Screen 34.
BEGINNING RAY LOOPS.
RAY LOOPS DONE.

Results written to file FOR040.DAT

Do another problem (Y/N)? y

The program returns to the main menu for another time through the problem. So that the problem will be somewhat different, a new elevation value will be used. To achieve this, select '2' from the main menu, and then '3' from the Elevation Parameters menu. An elevation angle of '8' degrees will be used. This will result in a screen that looks like Screen 35.

Screen 35

Enter starting elevation (deg), use neg. for values >90 deg from zenith. (10.0000000) 8

This is the only change that is necessary, so choose '8' from the Elevation Parameters menu. When control has returned to the main menu, selection '3' may be made so that the calculation may begin. Again, as with the first example, the query about updating the data file is seen. There is no need to preserve the change that has been made, so answer with 'n'. The raytracing then proceeds as usual, displaying the normal information. Because this example is essentially complete, respond with 'n' to the prompt asking about doing another problem. This is
shown in Screen 36 below. RAYTRACE then displays its final message and returns to the system.

Screen 36

Update the datafile (Y/N)? n
BEGINNING RAY LOOPS.
RAY LOOPS DONE.

Results written to file FOR041.DAT

Do another problem (Y/N)? n
Press RETURN to continue.

Old (saved) session

This third example illustrates the use of a RAYDAT-type file for input at the beginning of the problem. As with the second example, this example will be built upon the previous ones, and only those portions which are new for this example will get detailed attention. The type of session modeled by this example is representative of perhaps the majority of RAYTRACE sessions. By reading in a RAYDAT-type file, all of the parameters have some pre-existing setting. Only those parameters that are different for the particular problem at hand need be examined and changed.

First, start RAYTRACE in the usual manner. This time, however, at the initial prompt, the response will be '0' to indicate that this is an old problem that will initially use the parameter values stored in a RAYDAT-type file. The second prompt is the same as has been seen before. This time its purpose is to get the name of the file from which the parameters will be read. Any parameter changes may be stored
in this file also. For this example, the file created by the first example, 'TRIAL.DAT', will be used. This is shown in Screen 37.

Screen 37

Is this a new (1) or old (0) problem? ( 0): 0

Enter filename for storage of ionospheric information (10 char max): (RAYDAT.DAT): trial.dat

RAYTRACE now continues directly to the main menu. There is no need to read from a grid file because the pertinent data are stored in the RAYDAT-type file along with the parameters. At this point, if the parameters desired are the ones that were saved during the last session, the user could just select the third item from the main menu and proceed directly with the raytracing. For this example, however, recall from the first example that the parameters entered included a dimension of two for the problem. In the interests of speeding the example along, this should be changed to a dimension of zero. To do this, the user should select '2' from the main menu to reach the Elevation Parameters menu. Then '1' should be selected to reach the prompt from which the dimension may be set. Notice that the default value given is indeed two. Enter the new value of '0' as shown in Screen 38.

Screen 38

Enter dimension of problem ( 2.00000000 ) : 0

This is the only change that needs to be made, so enter a choice of '8' to return to the main menu. Now select '3' from the main menu to begin the raytracing. The program now presents prompts that have
not been seen until now. Because the parameters may have been changed, even though this is the first time through the problem, the user needs to be presented with the opportunity to preserve these changes. First, the user is asked if the values should be saved in the current file. If they are not to be saved there, the user is then asked if a new data file should be used to backup the new values. A negative answer to both of these prompts will mean that if the problem is ended the altered parameters will not be saved anywhere. It is not necessary that the change that has been made be saved, so answer 'n' to both prompts.

Some additional comments about these prompts are in order. If the old parameters are overwritten, then the data are safe and there is no need for the second prompt. RAYTRACE therefore skips it and goes on with the calculation. If a backup is desired, the user will be prompted for the filename to be used. Also, on succeeding times through the raytracing the prompt asking about updating the data file (as in Screen 36 above) will be referring to the last file that was written to.

After dealing with the above mentioned prompts, the raytracing is performed. The rest of the prompts are just as they have been before. There is really nothing further to be discovered, so 'n' may be entered in response to the query about another problem, as shown in Screen 39. After this the program ends in the usual fashion, returning to the system. This concludes the final example.
Screen 39

Overwrite existing file (Y/N)? ( ) : n
Backup the datafile (Y/N)? ( ) : n

BEGINNING RAY LOOPS.
RAY LOOPS DONE.

Results written to file FOR040.DAT

Do another problem (Y/N)? ( ) : n
Press RETURN to continue. ( ) : 

Remarks

The three examples given serve to illustrate essentially every feature of RAYTRACE. After following all three examples, the user should be familiar enough with the interface of the program to go ahead and "play around" by trying different things. Such "play" may be worth the time if the user feels the need for additional familiarity with the operation of the program. The menu reference sheet given later should also be of some help.

EXAMINING RESULTS

So far, the discussion has dealt with the use of the RAYTRACE program. The program is of very little use, though, unless there is some way of getting at the results from the raytracing runs. This section deals with the program READER2 that is provided with RAYTRACE.
READER2 represents a zeroth-order program for examining the results of RAYTRACE. While this program has proven adequate in the testing and development of RAYTRACE, it is certain that the end users of RAYTRACE will need to write their own, more sophisticated analysis software.

READER2 is very straightforward to use. When the user executes it, the program prompts for the name of the results file that it is to be read. Upon receiving the name, READER2 immediately displays the important values from the result file. This display is done for each pair of elevation and azimuth values. Results for each data point on the ray are displayed with the values for a particular point taking one screen. Recall that data points on the ray are earth impact points and the end point of the ray. After display of each data point, READER2 pauses and awaits the user’s response before continuing. When all the data are exhausted, the program simply ends and returns to the system.

The values displayed on the screen are by and large self-explanatory. The elevation and azimuth values are given in degrees. The direction cosines give the orientation and direction of the ray at that particular point in an SEZ coordinate system where X is South, Y is East, and Z is up. The deviation value represents the deviation away from where the ray would have landed had it stayed on a great circle path. This is a measure of the cross-path tilt of the ionosphere.

Several comments on the regular usage of READER2 are in order here. Although the display of the results on screen is useful, it is often more useful to have a hardcopy of these results. This is achieved not by READER2, but by the system that it is operating on and the console used. Many systems and consoles have the capability to either redirect output to a disk file, or perform screen dumps to a printer. Given the rather widespread presence of such a capability, no provision has been made to produce a printable file. Since the user is likely to develop individualized analysis software anyway, there is really no need for this capability.

Another comment concerns the blank record that will occasionally be displayed. This is entirely harmless. It occurs because provision must be made for the possibility that the limit on the number of bounces may be reached and there still needs to be the additional space for the end of ray point’s data. If the bounce limit isn’t reached, then this extra space is blank.
This section has the purpose of gathering together some of the comments that have either gone unsaid to this point, or bear repeating. First, there are some comments of a general nature. Comments specific to a particular implementation or environment follow. Finally, some usage hints are presented.

General comments

The speed with which RAYTRACE operates will be of some concern to many users. Some benchmarks have been run on various machines. The two primary implementations discussed in this User's Guide are the VAX/VMS and the DOS versions. The machines compared were an IBM AT-compatible (with math coprocessor) and a VAX 11/785. The benchmarked run uses the TRIAL.DAT file from the four ray example of the tutorial examples above. The run was repeatedly made and the results averaged. Timing was done by stopwatch from typing return at the last prompt before the execution of the ray loops until the message of "RAY LOOPS DONE" appeared on the screen. For the VAX, in addition to the apparent execution time as measured by stopwatch, the CPU time was taken by using the <CTRL>-T mechanism of VMS for obtaining the process time statistics. Because the VAX is a multiuser machine, the usage level is of some significance. The benchmark was run early in the morning before the many users were on the system. At the time of the runs, there were only four users, so that the machine could be termed lightly loaded.

The results of the benchmark are as follows:

<table>
<thead>
<tr>
<th>Machine</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT-clone</td>
<td>40.7 sec</td>
</tr>
<tr>
<td>VAX 11/785</td>
<td>9.2 sec (user's time)</td>
</tr>
<tr>
<td>VAX 11/785</td>
<td>5.1 sec (CPU time)</td>
</tr>
</tbody>
</table>
Due to the usage level on the VAX when the benchmark was run, it is fairly clear that the user time could have been somewhat better, but it also could have been much worse. Extrapolating from other experience, the execution time for this benchmark on any of the current (1987-88) crop of 32-bit microcomputers would lie somewhere midway between the VAX's user time and CPU time.

Several restrictions of RAYTRACE should be noted. Filenames are to be ten characters or less in length. The input files need to be located in the same directory as the executable for RAYTRACE, at least the way RAYTRACE is currently written. Another restriction is on the size of the input data. Currently, the grid of points cannot be larger than 30 in latitude by 60 in longitude. This trims the size of the memory allocated to a level tolerable to the DOS specifications stated previously. A third restriction is that rays may not be fired with an elevation angle of 90 degrees. This restriction only applies if the ray will make a reflection from the ionosphere. Rays which penetrate the ionosphere may have elevations of +/- 90 degrees.

It is important to make sure when doing a new problem that ALL of the menu items have been selected and checked to verify that their values are properly set. Failure to do so is perhaps one of the most frustrating of mistakes that can be made while using RAYTRACE, as the program is perfectly willing to use whatever the current values are even though they may not be what the user intends. As a further reminder, the following table lists some parameters and their range of values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latitude</td>
<td>+90 to -90 deg</td>
</tr>
<tr>
<td></td>
<td>with (+) north</td>
</tr>
<tr>
<td>Longitude</td>
<td>-180 to +180 deg</td>
</tr>
<tr>
<td></td>
<td>with (-) for west longitudes</td>
</tr>
<tr>
<td>Plasma frequency squared</td>
<td>Megahertz squared</td>
</tr>
<tr>
<td>Heights and distances</td>
<td>Kilometers</td>
</tr>
<tr>
<td>Elevation angles</td>
<td>+90 (zenith) to -90 (nadir) degrees</td>
</tr>
<tr>
<td>Azimuth angles</td>
<td>0 to 360 degrees with 0 north and values</td>
</tr>
<tr>
<td></td>
<td>increasing eastward</td>
</tr>
</tbody>
</table>
Some of the parameters that the user enters have either imposed or practical bounds. The wave frequency of the RF transmission that the ray represents is bounded on the lower side to a few MHz. This is because for lower frequencies magnetic field effects, which are not included in the current version of RAYTRACE, play an increasing role. As an upper bound, frequencies beyond a few GHz begin to lose reliability for two reasons. First, as a calculational matter, the ratio of the plasma frequency squared to the wave frequency squared becomes very small. Operations occur of the type \((1 - x)\) where \(x\) is a small number, with the resulting loss of accuracy. Second, refraction due to the neutral atmosphere becomes a much more important factor at these frequencies. These limits are then limits on the believability of the results. RAYTRACE will calculate over a much larger range of frequencies, but the results will be far from reliable.

The size of the raypath increment is another quantity that has practical limits. There is a lower limit due to accumulation of error and the duration of the calculation. For raypath increments smaller than a few tenths of kilometers, even modest calculations begin to take intolerable lengths of time. This conclusion is of course subjective and depends upon the computer being used at the time. The problem of duration of the calculation has, however, prevented rigorous testing for the degradation of the results due to accumulation of errors. An upper limit also exists on the size of the raypath increment. Increments which are too large begin to violate an approximation made in the theory underlying the raytracing algorithm. Typically, increment sizes of 10-15 km are about the maximum practical. As with the limits on the wave frequency, values outside these may be used, but the results must be viewed as ranging in quality from rough approximations to completely unreliable.

The cutoff height is a quantity that bears further discussion. The default value initially given is essentially the height of \(\gamma\)-synchronous orbit. This value represents a compromise. The ionosphere is considered to end at a height of 2000 km in the model used. Propagation to altitudes beyond this is modeled as straight line propagation through free space. While this is certainly not rigorously correct, it is by and large an accurate assumption. Clearly some cutoff must be
used so that the program doesn't calculate off to infinity. Since most radio operations currently take place within a sphere defined by the radius of geosynchronous orbit the default value seems convenient. The value may be reset to any value the user desires. Be warned, however, that the neglected refraction will eventually add up as the path length increases. Depending upon the application, this error could have a substantial effect.

VMS specific comments

The points that are mentioned here have been discussed in greater detail elsewhere and are referred to here as a reminder. When transporting the source code between a DOS-based and a VAX/VMS system, there are a few lines that may need to be changed. These changes are discussed in detail in the Setup section. The naming and renaming of files needs to be carefully considered. The number of results files having similar names but different version numbers can become bewildering. Also, the user should automate as much of the ray tracing process as possible through the use of command files.

When using VMS command files to perform batch RAYTRACE jobs, one drawback becomes obvious. There is no provision in such command files for a response that involves just typing a return. Every line of the command file must contain something. For this reason, any prompt of the 'Press RETURN to continue.' type will accept '.' and return as if it were just a plain return. Therefore, on any line of a command file where the desired response is just a return, a line with a single period on it will serve as the necessary substitute.

DOS specific comments

The points that are mentioned here have been discussed in greater detail elsewhere and are referred to here as a reminder. When transporting the source code between a VAX/VMS system and a DOS-based system, there are a few lines that may need to be changed. These changes are discussed in detail in the Setup section. The naming and renaming of files needs to be carefully considered. Because DOS
provides little protection against accidental destruction of files, the user must exercise extreme care to preserve files against disaster. Also, the user should automate as much of the raytracing process as possible through the use of batch files.

Uses and Usage tips

RAYTRACE is a tool that may be used to accomplish a number of tasks in the realm of propagation modeling. It may not be immediately clear, though, how to do some things with the program. The following is a discussion of a few sample uses.

The instructions for operating RAYTRACE discussed the concept of the dimensionality of the problem. From that discussion, it should be clear how to trace single rays, vertical fans of rays, and arrays of rays. A variation on this theme that might prove useful is the horizontal fan of rays. This may be achieved by using a two dimensional problem with only one allowed elevation angle. The mechanics of doing this involve setting the starting and ending elevation values equal (or nearly equal) and entering a step size that is arbitrary (or large compared to the difference between the starting and ending elevations).

Information on the coverage of an area by a broadcast transmitter may be obtained by tracing a two dimensional set of rays. It should be noted that the resulting array of landing points for these rays will not be evenly spaced in latitude and longitude. It is possible, though, to interpolate on such a grid and produce contours or other desired information. At present the only way to achieve (approximately) the desired spacing between points is by experimentation with a small number of rays.

Tracing rays between two specific points (homing) is also a desirable capability. While such a capability is not currently built into RAYTRACE, it is relatively straightforward to do manually. By tracing a pair of rays that are fairly close in elevation, the results for the end points of the pair of rays may be used in a linear interpolation scheme. Typically it requires only five or so iterations to reach the desired convergence on the specific target point. A similar process may be done for the azimuth, although the azimuth is usually
fairly well approximated by that of the great circle path between the two points. An azimuth search will usually also converge more quickly.

It is sometimes desirable to have detailed information along the raypath. There currently exists some code within RAYTRACE to report the height and range at each increment along the ray, for the purpose of later plotting of the raypath. This code is normally commented out, as the files of coordinates produced may become enormous. This code is located in the routine RAYSUB a few lines below line 30300 (consult the source code listing, Appendix D). It merely consists of a WRITE statement and its corresponding FORMAT statement. Uncommenting these lines and recompiling will yield a version of RAYTRACE that will produce an extra generic data file with pairs of values: ground range, height. These values may then be used in a plot of the trajectory of the ray. Certainly, more sophisticated schemes are possible, but such things are best tailored to a given need.

Ultimately, RAYTRACE is extensible. This is most easily achieved by extending the menu system. Adding to the menu system allows more parameters or flags to be entered by the user. Flag values entered in this manner could be used to activate or deactivate additional sections of code. This code could then operate upon existing and new parameters for the calculation of additional results. Details of adjusting the menu system are given in a comment block in the routine DOxDATA, just after line 12002. The menu files are discussed in Appendix A. The only other requirement is the addition of the new features or calculations at the appropriate locations in the existing program. This may only be determined by study of both RAYTRACE and the theoretical groundwork for the intended addition.

MENU QUICK REFERENCE

The three menus of RAYTRACE are presented here, with brief explanations of most menu items. They are given one per page, so that the user can easily photocopy them for handy reference. There is also ample room for notes. One suggestion, if a copy machine that reduces is available, is to copy the three menus reduced, then paste them on one sheet. Copying this sheet will give a single page "cheat sheet". 
MAIN MENU

1 - Edit launch parameters (Go to Launch Parameters menu)
2 - Edit elevation/azimuth parameters (Go to Elevation Parameters menu)
3 - Proceed with raytracing
4 - Quit program
LAUNCH PARAMETERS MENU

1 - Bounce limit
   (limit number of earth impacts)

2 - Signal intensity
   (flag to turn on/off intensity loss calculation)

3 - Conductivity
   (select type of earth surface for reflection loss)

4 - Launch point
   (latitude/longitude location of start point)

5 - Launch height
   (altitude of start point)

6 - Range & height limits
   (set great circle ground range and altitude cutoffs)

7 - Ray path increment
   (set step size along the ray)

8 - Wave frequency
   (frequency of transmission modeled by the ray)

9 - Done
   (return to the main menu)
### ELEVATION PARAMETERS MENU

1 - Dimension of problem (determine multi-ray launch pattern)
2 - Starting azimuth (set initial launch azimuth)
3 - Starting elevation (set initial launch elevation)
4 - Elevation resolution (set elevation step size for multi-ray calculations)
5 - Azimuth resolution (set azimuth step size for multi-ray calculations)
6 - Elevation limit (set limiting value on elevation for multi-ray calculations)
7 - Azimuth limit (set limiting value on azimuth for multi-ray calculations)
8 - Done (return to main menu)
APPENDICES
Appendix A:
File Formats

GRID.DAT

The grid file is a means by which ionospheric information may be input into RAYTRACE. The grid file is generally produced externally to RAYTRACE, either by use of an ionospheric model program or by processing of actual ionospheric data. The file contains the pertinent ionospheric information at a gridded set of latitudes and longitudes. Grids may be entered from the console, but this is only practical for relatively small grids.

Grid files are composed of a number of data arrays. First, the definition of the latitude and longitude grid is recorded. Following this, the six ionospheric parameters required for specification of the vertical profile are given for each point on the grid. The six parameters are the maximum plasma frequencies for the E, F1, and F2 layers, the heights for the maximum frequencies for the F1 and F2 layers, and the semithickness of the F2 layer. The choice of these parameters is determined by the profile model that is built into the current version of RAYTRACE. Finally, the date, time, and sunspot number are recorded.

The array of values that serves to define the latitude and longitude grid is a real array containing six parameters. They are the latitude and longitude spacing for the grid, followed by the initial latitude and longitude values for the grid. The starting point is taken as the southwest corner of the grid. Values are in degrees and longitude is given as east of Greenwich positive, west negative. This gives longitudes running from -180 to +180 degrees. The last two values in the array are the number of points that the grid is to contain in the latitude direction and in the longitude direction respectively. The grid must be rectangular and if necessary should be either padded with more data (if the grid is being generated from a model or interpolation
of actual data) or some points eliminated so that it becomes rectangular.

The arrays of ionospheric data are written as two dimensional arrays with the first index corresponding to latitude and the second to longitude. The indices represent the integer count of grid points traversed in the given direction to reach the coordinates of the grid point. The first array contains the peak plasma frequency squared values for the E-layer, for the entire grid. The second array holds the height of the peak of the F1-layer and the third holds the plasma frequency squared values for the F1 maximum. The last three arrays contain the values which are used to specify the F2-layer. They are, in order of appearance, the semithickness, the height of the layer maximum, and the plasma frequency squared at the maximum.

The final array is an integer array containing the time, date, and sunspot number for which the problem is being run. These values are contained in the array in this order: sunspot number, year, month, day, hour (UT), and minute. The current configuration of RAYTRACE no longer uses this information directly as the portions of code that use it have been commented out. They have been left in both for the possibility of future calculational use, and so that the information may be included in displays or reports if desired.

The format of the GRID.DAT type of data file is perhaps best illustrated by including fragments of code that RAYTRACE uses to read information from the grid file. The fragmented listing (Listing 1) includes the declaration statements. The file is written unformatted to save time and space. The filename is set either as a default value or from a prompt to the user. The correspondence between the arrays mentioned above and those in Listing 1 is fairly obvious. The variables are named according to the convention that is presented with the source code listing in Appendix D.
The RAYDAT.DAT type of file is used by RAYTRACE for the storage and retrieval of both the ionospheric information and parameters of the problem that the user has entered. For the sake of simplicity, the ionospheric information is stored in exactly the same manner as in the case of GRID.DAT type files. Arrays containing the other problem parameters are then added in after the ionospheric values. As a consequence of this file structure, only the arrays that are not repeated from GRID.DAT need be discussed.

The first of these arrays is an integer array containing three values. The bounce limit is stored first. This sets a bound on the number of times that a ray may impact the earth, which is useful in some cases dealing with subionospheric raypaths. Next is a flag that determines whether or not signal loss calculations will be performed.
Finally, there is a flag that determines what sort of surface will be used in reflection loss calculations for earth impacts of the ray.

The second array is a real array containing the latitude and longitude of the launch point. This is followed in the array by the range limit. This parameter sets a bound on the great circle ground range that the ray is allowed to propagate to. The fourth value in the array is the size of the raypath increment. Fifth is the wave frequency. The sixth value is the starting height of the ray, above the earth's surface. Finally, the seventh value is the altitude cutoff value.

The final array is a real array that contains the information necessary to specify the elevations and azimuths at which rays will be sent out. The first parameter is the dimensionality of the problem. This is defined by considering the aim points of the ray in an elevation versus azimuth plane. A single ray is a single point, and so has dimension zero. A fan of rays having a particular azimuth forms a line of points, and is considered dimension one. Lastly, a set of such fans produces an array of points on the plane and is therefore considered as dimension two. After the dimensionality of the problem comes the initial azimuth and the initial elevation. Following these values, the azimuth and elevation spacing between rays is stored. Last of all, the final azimuth and elevation values are stored.

As above, the easiest way to make the structure of the data file manifest is to present the fragments of code responsible for writing to the file. The code is presented in Listing 2. Again, the filename is set previously to its proper value. The additional arrays are RVxLAUN, RVxLAUN, and RVxOPTION.
INTEGER IVxPARAM(6), IVxLAUN(3)
C
REAL*8 RVxGRID(6), RVxLAUN(7)
REAL*8 RVxOPTION(7)
REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
C
CHARACTER*10 KVxDFIL

OPEN (30, FILE = KVxDFIL, FORM='UNFORMATTED')
REWIND (30)
WRITE (30) RVxGRID
WRITE (30) ((RVxION1(I,J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
WRITE (30) ((RVxION2(I,J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
WRITE (30) ((RVxION3(I,J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
WRITE (30) ((RVxION4(I,J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
WRITE (30) ((RVxION5(I,J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
WRITE (30) ((RVxION6(I,J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
WRITE (30) IVxPARAM
WRITE (30) IVxLAUN
WRITE (30) RVxLAUN
WRITE (30) RVxOPTION
CLOSE(30)

FOR###.DAT

This file is the results file for RAYTRACE. The ### represents a number that is assigned sequentially during the course of a given ray-tracing session so that the different runs during the session may be distinguished. Because results files may need to be of greater portability, the results are written to disk in formatted form. The results are stored by first saving the information necessary to determine the number of data points represented in the file. Data points consist of earth bounce points, and the terminal point on the ray. The azimuth and elevation of a particular ray are saved and then the array
containing the information from all the data points on that ray is stored. This then repeats until the data from all rays has been saved.

The number of data points per ray is determined from the maximum number of bounces allowed by the bounce limit, plus one for the terminal point of the ray. The proper number of rays are then accessed by using the number of azimuths and the number of elevations recorded. These set the limits on two loops, within which the file is accessed.

The array of data for a given ray is two dimensional. The first index is the number of the data point along the ray. The second represents the particular data element for that point. The meaning of these elements is given in a comment block in the source code for the subroutine RAYSUB, just after the variable declarations. The ones that are currently used are as follows. First are the latitude and the longitude. The third element is the group time delay along the ray to that point. The great circle ground range is next, followed by the signal loss in dB, due to spatial loss, focusing or defocusing, and possibly reflection loss. The sixth element contains the amount that the ray has deviated from a great circle path at that point. The ninth element of the array gives the height of the ray at the data point. Element ten contains the phase path length along the ray to that point. The next three elements give the x, y, and z direction cosines of the ray at that point. Here, the x, y, and z directions are in the coordinate system where x is south, y is east, and z is up.

The sections of code for writing the results out to the file have been excerpted from the main routine of RAYTRACE. The code is presented in Listing 3. Also presented is the code that produces the number associated with the file. The integer IVxIND is initialized to 40 and incremented every time a particular run is complete. It is then used to produce the filename of the results file.
**Listing 3**

```
INTEGER IVxIND, IVxLAUN(3), IVxAZLIM, IVxELLIM

REAL*8 RVxAZI, RVxELEV, RVxBOUN(11,15)

CHARACTER*2 KVxIND
CHARACTER*10 KVxOFIL

WRITE (IVxIND, (12')) 17x!ND
KVxOFIL = 'FORO'//KVxZND//'.OAT'

OPEN (IVxINDFILE = KVxOFIL)
WRITE (IVxIND, 22000) IVxLAUN(1)+1, IVxAZLIM, IVxELLIM

WRITE(IVxIND, 21003) RVxAZI, RVxELEV

WRITE(IVxIND, 21003) ((RVxBOUN(L,K),K=1,15), L=1, IVxLAUN(1)+1

CLOSE(IVxIND)

21003 FORMAT (G24.14)
22000 FORMAT (I4)
```
MENU FILES

The scheme for using and presenting the menus and menu files is detailed in the source code listing in subroutine IOxMENU. The method is reviewed here. The file MASTER.MEN is used to maintain a list of the rest of the menu files. It is merely a text file that contains the names of the other menu files, with each filename on a separate line. RAYTRACE is currently set up to handle up to ten menu files in addition to the master file. The filenames are ordered according to their numbering within RAYTRACE. Filenames are currently configured to be no more than ten characters long.

The other menu files contain the actual information that is displayed on the console as a menu. As with MASTER.MEN, the other .MEN files are just text files. The information needed for the display of a menu is as follows. The first line has two items that are separated by a comma. These are the number of items in the menu, followed by the title of the menu. The number of items is an integer. All of the text information in these files is read in using an A40 format, and so may be no longer than 40 characters long. After this first line, the menu items are given. The text that will be displayed for each item is put in the file with one menu item's text per line. Again, these are to be no longer than 40 characters. The last line of the file contains the text for the prompt that will be displayed at the end of the menu. This information is accessible so that the menu text may be customized at will by the user with just an ordinary text editor.

A more complete understanding of how this all works will only be obtained by examining the source code of subroutine IOxMENU. In particular, this routine should be examined very closely if any extensions to RAYTRACE are to be made. Any additional information that must be entered ought to be entered by means of a menu so that the user interface of the program will remain consistent. Examination of the existing menu files will also be of assistance.
Appendix B:
Inventory of COMMON blocks

This section lists the COMMON blocks used by RAYTRACE, along with the declarations of the variables involved. An explanation is given for some of the variables. A complete explanation of all the variables involved would require describing significant portions of the algorithms used, which is beyond the scope of this User's Guide.

MAINDAT:
This block contains the contents of RAYDAT.DAT. As may be seen from the EQUIVALENCE statements, the array RVxIONPT holds the six sets of ionospheric parameters input from the data file.

INTEGER IVxPARAM(6), IVxLAUN(3)

REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7), RVxOPTION
REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)

EQUIVALENCE (RVxION1,RVxIONPT), (RVxION2,RVxIONPT(1,1,2))
EQUIVALENCE (RVxION3,RVxIONPT(1,1,3))
EQUIVALENCE (RVxION4,RVxIONPT(1,1,4))
EQUIVALENCE (RVxION5,RVxIONPT(1,1,5))
EQUIVALENCE (RVxION6,RVxIONPT(1,1,6))

COMMON /MAINDAT/ RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN,
* RVxLAUN, RVxOPTION

RESULTS:
The RESULTS common block contains the array which holds the data for every data point on a particular ray. For each ray that is traced, this data array is recorded. Refer to the discussion above concerning the FOR###.DAT file format for further detail on this array.

REAL*8 RVxBOUN(11,15)

COMMON /RESULTS/ RVxBOUN
LPARM:

INTEGER IVxSSNUM, IVxTIME(5), IFxGRND
REAL*8 RVxYEAR, RVxA100, RVxTIM
COMMON /LPARM/ IVxSSNUM, IVxTIME, RVxYEAR, RVxA100, IFxGRND * RVxTIM

LOSSES:

This block holds the various components of signal loss that are computed by RAYTRACE. RVxLOSA is the absorption loss. The code to calculate this is based upon empirical formulae which were appropriate to use at earlier stages of the development of RAYTRACE, but cannot be used for general paths. This code is currently commented out. RVxLOSX has suffered a similar fate. This was used to represent the so-called "excess loss" that some earlier ionospheric models use. This value is inappropriate, though, for use in examining specific raypaths in that it becomes possible for two relatively adjacent rays to have anomalously different losses. The remaining two values are actually used if signal intensity calculations are to be performed at all. RVxLOSR gives the reflection loss, and RVxLOSG gives the geometric loss. The geometric loss is the combination of the effects of distance from the source, and also focusing/defocusing effects.

REAL*8 RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG
COMMON /LOSSES/ RVxLOSA, RVxLOSR, RVxLOSX, RVxLOSG

PRAM:

The values in this block are, in order: pi, a conversion factor to go from degrees to radians, the radius of the earth, and the height considered to be the top of the ionosphere.

REAL*8 RPxPI, RPxDTOR, RPxREARTH, RPxHTOP
COMMON /PRAM/ RPxPI, RPxDTOR, RPxREARTH, RPxHTOP
SCPS1:
This block contains information on the number and location of the grid points. ICxNSCP is the number of grid points, and the other two are arrays containing the latitudes and longitudes, respectively, of the grid points.

INTEGER ICxNSCP
REAL*8 RVxLATSC(1800), RVxLONSC(1800)
COMMON /SCPS1/ ICxNSCP, RVxLATSC, RVxLONSC

SCPS1A:
These arrays contain the six ionospheric parameters for all the grid points, with all the layer peak plasma frequency squared values in one array and all the height (and semithickness) values in the second.

REAL*8 RVxFNSQ(1800,3), RVxH(1800,3)
COMMON /SCPS1A/ RVxFNSQ, RVxH

OTHER:

REAL*8 RVxLATL1, RVxLON1, RVxHBOT
REAL*8 RVxFSQU, RVxRPI, RVxHCUT
COMMON /OTHER/ RVxLATL1, RVxLON1, RVxHBOT, RVxFSQU, RVxRPI, RVxHCUT

MISC:

REAL*8 RVxSEZGEC(3,3), RVxX7, RVxY7, RVxZ7
REAL*8 RVxR1, RVxR2, RVxHMIN
COMMON /MISC/ RVxSEZGEC, RVxX7, RVxY7, RVxZ7, RVxR1, RVxR2, RVxHMIN

81
START:
This block contains the initial position information for the raypath increment.

REAL*8 RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI
COMMON /START/ RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI

END:
This block contains information on the end of the raypath increment.

REAL*8 RVxH5, RVxF5, RVxF6, RVxF7
COMMON /END/ RVxF5, RVxF6, RVxF7, RVxH5

ION01:
This block stores some ionospheric parameters.

REAL*8 RVxALPHA, RVxBETA, RVxETA1, RVxETA2
REAL*8 RVxFNSB(3), RVxHB(3)
COMMON /ION01/ RVxALPHA, RVxBETA, RVxETA1, RVxETA2, RVxFNSB, RVxHB

ION02:
This block stores ionospheric interpolation results.

REAL*8 RVxB5(3), RVxB6(3), RVxA5(3), RVxA6(3)
REAL*8 RVxE5(3), RVxE6(3)
COMMON /ION02/ RVxA5, RVxA6, RVxB5, RVxB6, RVxE5, RVxE6
IONO3:

This holds still other quantities used in the ionospheric calculations.

    INTEGER IFxCASE
    REAL*8 RVxV1, RVxV2, RVxXX, RVxHBND
    COMMON /IONO3/ RVxV1, RVxV2, RVxXX, IFxCASE, RVxHBND

MORE:

    INTEGER IVxSX, IVxSY, IVxSZ
    REAL*8 RVxCX, RVxCY, RVxCZ
    COMMON /MORE/ IVxSX, IVxSY, IVxSZ, RVxCX, RVxCY, RVxCZ

TEMP1:

    REAL*8 RVxF40, RVxF65, RVxK1, RVxHL
    COMMON /TEMP1/ RVxF40, RVxF65, RVxK1, RVxHL

GORP:

    REAL*8 RVxDD6, RVxANGLIM
    COMMON /GORP/ RVxDD6, RVxANGLIM

TEMP2:

These are indices use in accessing grid arrays for interpolation.

    INTEGER IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
    COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4
TEMP3:

REAL*8 RTLxD5, RTLxI3, RTLxI4, RTLxI5
COMMON /TEMP3/ RTLxI3, RTLxI4, RTLxI5, RTLxD5

VAR1:

REAL*8 RVCxXL, RVCxXU, RVCxA0, RVCxHU, RVCxBO
COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxBO

VAR2:

REAL*8 RVCxHB1, RVCxH2, RVCxYS, RVCxSL1, RVCxSL2, RVCxH1L
COMMON /VAR2/ RVCxHB1, RVCxH2, RVCxYS, RVCxSL1, RVCxSL2, RVCxH1L

VAR3:

REAL*8 RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P
COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P

VAR4:

REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4, RVCxHT5
COMMON /VAR4/ RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4, RVCxHT5

RAID:

INTEGER IFCxSN
COMMON /RAID/ IFCxSN

84
TEMP4:

REAL*8 RTLXF3, RTLXF1, RTLXF2
COMMON /TEMP4/ RTLXF1, RTLXF2, RTLXF3

LOCAL:

REAL*8 RVLXELEV
COMMON /LOCAL/ RVLXELEV
Appendix C:
Error Messages

RAYTRACE will detect a few errors internally and generate error messages for these. These are in addition to those that may be generated as run-time errors as specified by the compiler being used, or may be operating system level errors. This section will discuss the error messages generated directly by RAYTRACE.

INVALID CHOICE.
This message indicates that a bad choice has been made from one of the menus. A bad choice is one that is out of the range of the possible selections.

HIT RETURN ONLY TO CONTINUE.
The routine that handles the 'Press RETURN to continue.' prompt will generate this message if anything other than an 'A', '.', or return is typed.

Conversion ERROR, Please RETRY Input
Any given prompt is expecting a particular type of response (REAL, INTEGER, etc.) and this message will be given if the value entered is not of the expected type. Note that if a REAL is expected, and an integer is entered, this will not occur, as a decimal point may be attached to the value.

RAY DOES NOT PROPAGATE!!!!
This message indicates that a ray has been given a starting point at an altitude inside the ionosphere and has parameters that are inconsistent with any propagation for the ray.
EXCEEDINGLY WEIRD CASE: POINT #1 (& a few values are also displayed)

If this message ever appears, something has gone quite wrong in the routine which calculates the ionospheric parameters and gradients. Theoretically, this should never be seen. Copy down ALL the values given, along with all other parameters from the menus and the ionospheric specification (if possible). Contact the authors of RAYTRACE with this information.

EXCEEDINGLY WEIRD CASE: POINT #2

This message is very similar in nature to the previous one. It, too, should never occur. If it does, however, follow the instructions given for the similar error above.
References


Index

.COM 9
.DAT 8, 10
FOR##.DAT 15, 28, 70, 74
FOR040.DAT 26, 45, 48, 51, 55
FOR041.DAT 47, 48, 52
GRID.DAT 15, 18, 27, 28, 31, 48, 49, 66 - 68
RAYDAT.DAT 15, 17, 27, 28, 30, 48, 53, 68, 74
TRIAL.DAT 29, 44, 47, 48, 53, 57
TRIAL2.DAT 48
.FOR
RAYTRA4.FOR 6 - 8, 10
READER2.FOR 6, 7, 9, 11
RIONO.FOR 6, 7
RMISC.FOR 6, 7
RPHASE.FOR 6, 7
RRAYSB.FOR 7
RTILT.FOR 6, 7
.MEN 6, 8 - 11, 15, 73
ELEV.MEN 6, 7, 15
LAUNCH.MEN 6, 7, 15
MAIN.MEN 6, 7, 15
MASTER.MEN 6, 7, 15, 73
Abort 20, 27, 34
Accuracy 2, 23, 39, 59
Algorithm 2, 6, 59, 74
Altitude 2, 22, 25, 42, 59, 64, 81
   altitude cutoff 22, 39, 64, 69
   altitude limit 22, 39
Approximation 2, 59
AT 7, 57
<table>
<thead>
<tr>
<th>Azimuth</th>
<th>Backup</th>
<th>Benchmark</th>
<th>Bounce limit</th>
<th>Comment</th>
<th>COMMON</th>
<th>Compile</th>
<th>Conductivity</th>
<th>Coordinates</th>
<th>Cutoff</th>
<th>altitude</th>
<th>height</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td>20, 22, 24, 25, 28, 35, 40 - 44, 49, 56, 58, 61 - 63, 65, 69 - 71</td>
<td>26, 54, 55</td>
<td>57, 58</td>
<td>21, 28, 36, 37, 56, 64, 68, 71</td>
<td>62, 67, 75</td>
<td>74 - 80</td>
<td>3 - 11</td>
<td>21, 22, 36, 37, 64</td>
<td>2, 62, 67</td>
<td>2, 3, 22, 59</td>
<td>22, 39, 64, 69</td>
<td>59</td>
<td>21, 64</td>
</tr>
<tr>
<td>limit 24, 25, 41, 44, 65</td>
<td>resolution 24, 25, 41, 43, 65</td>
<td>starting azimuth 24, 25, 41, 42, 50, 65</td>
<td>Backup 26, 54, 55</td>
<td>Benchmark 57, 58</td>
<td>Bounce limit 21, 28, 36, 37, 56, 64, 68, 71</td>
<td>Comment 62, 67, 75</td>
<td>Conductivity 21, 22, 36, 37, 64</td>
<td>Coordinates 2, 62, 67</td>
<td>Cutoff 2, 3, 22, 59</td>
<td>altitude 22, 39, 64, 69</td>
<td>height 59</td>
<td>range 21, 64</td>
</tr>
</tbody>
</table>
Elevation Parameters menu 23, 25, 28, 40 – 46, 49 – 51, 53, 65
EQUIVALENCE 74
Error 17, 59, 60, 81, 82
roundoff 23
Excess loss 75
Executable 3, 6, 8 – 11, 58
Execution time 2, 57, 53
F1 layer 16, 19, 32, 33, 66, 57
F2 layer 16, 19, 32, 33, 66, 67
Fan (of rays) 69
horizontal 61
vertical 24, 61
File 1, 3, 6 – 11, 15, 17, 21, 26, 27, 29, 30, 44, 46 – 48, 50,
      52 – 58, 60 – 62, 66 – 73
data 3, 12, 13, 26, 28, 29, 46 – 48, 51, 52, 54, 55, 62, 67,
      69, 74
grid 12, 13, 15, 16, 18, 27, 29, 31, 48, 49, 53, 66
menu 6, 62, 73
result 3, 10, 12, 21, 27, 28, 44, 45, 47, 48, 51, 52, 55,
      56, 60, 70, 71, 74
results 26
Filename 5, 7, 8, 10, 17, 26, 29, 30, 47, 48, 53, 54, 58, 67, 69,
      71, 73
Focusing 2, 22, 71, 75
FOR###.DAT 15, 28, 70, 74
FOR040.DAT 26, 45, 48, 51, 55
FOR041.DAT 47, 48, 52
Free space 59
Frequency
  plasma 16, 19, 32, 58, 59, 66, 67, 76
  wave 21, 23, 36, 40, 59, 64, 69
Geometric
  loss 75
  spreading 22
Geosynchronous 22, 59, 60
Grid 2, 3, 12, 13, 16, 18, 19, 27, 31, 32, 58, 61, 66, 67, 78
  file 12, 13, 15, 16, 18, 29, 31, 48, 49, 53, 66, 67
  point 13, 18, 31 – 33, 66, 67, 76

93
GRID.DAT 15, 18, 27, 31, 48, 49, 66 - 68
Group path 2
Group time delay 71
Height 16, 19, 32, 39, 58, 59, 62, 66, 67, 71, 75
cutoff 59
launch 21, 36, 64
limit 21, 36, 64
starting 22, 38, 69
Homing 61
Increment 2, 13, 43, 62
raypath 2, 21, 23, 36, 39, 40, 59, 64, 69, 77
Index
array 67, 71, 73
of refraction 2
Installation 1, 7, 9
Integration 3
Interpolation 2, 61, 66, 77, 78
Ionosphere 1 - 4, 6, 12, 13, 15 - 19, 21, 26, 27, 29 - 32, 35,
  38, 40, 42, 47 - 49, 53, 56, 58, 59, 66 - 68, 74 - 78
  81, 82
IOxMENU 73
Item
menu 14, 15, 19 - 23, 25, 27, 35, 37 - 44, 46, 49, 50, 53,
  58, 62, 73
Iteration 61
KEYBRD 8, 10
Latitude 18, 22, 31, 32, 38, 58, 61, 64, 69, 71
grid 2, 3, 12, 16, 18, 27, 31, 32, 58, 66, 67, 76
Launch Parameters menu 20, 23, 26, 28, 35 - 38, 40, 49, 63, 64
LAUNCH.MEN 6, 7
Limit 21, 59, 71
  altitude 22, 39
  azimuth 24, 25, 41, 44, 65
  bounce 21, 28, 36, 37, 56, 64, 68, 71
  elevation 24, 25, 41, 43, 44, 65
  height 21, 36, 64
  range 21, 22, 36, 39, 64, 69
Limiting value 25, 65

94
Longitude  18, 22, 31, 32, 38, 58, 61, 64, 66, 67, 69, 71
grid    2, 3, 12, 16, 18, 27, 31, 32, 58, 66, 76
Loss   75
  absorption  75
  excess  75
  geometric  75
  intensity  64
  of accuracy  59
  reflection  22, 37, 64, 69, 71, 75
  signal  68, 71, 75
  spatial  71
Magnetic field  1, 23, 59
Main menu  6, 12, 14, 15, 17 - 20, 23, 25, 27, 34, 35, 40,
            44 - 46, 49 - 51, 53, 63 - 65
MAIN.MEN  6, 7
MASTER.MEN  6, 7, 73
Memory  3, 4, 58
Menu  6, 15, 19, 20, 23, 34, 35, 39, 40, 42, 46, 49, 62, 73, 81.
        82
  choice  20, 49, 81
Elevation Parameters  14, 23, 25, 28, 40 - 46, 49 - 51, 53,
                     63, 65
file  6, 62, 73
item  15, 20, 23, 25, 37 - 44, 46, 49, 50, 53, 58, 62, 73
Launch Parameters  14, 20, 23, 26, 28, 35 - 38, 40, 49, 63.
        64
main  6, 12, 14, 15, 17 - 20, 23, 25, 27, 34, 35, 40,
        44 - 46, 49 - 51, 53, 63 - 65
quick reference  55, 62
selection  20, 22, 34, 35, 37 - 40, 42, 44, 49, 53, 81
system  62
New problem  12, 13, 17, 26, 29, 48, 58
Old problem  12, 13, 17, 26, 29, 52
Output  21, 56
  file  12, 15, 27
Overwrite  26, 55
Path length 60
  group 2
  phase 2, 6, 71

Plasma
  frequency 15, 19, 32, 53, 59, 66, 67, 76

Plot 24, 62

Problem 14, 15, 17, 20, 22, 26, 27, 33, 36, 39, 42 - 48, 50 - 52, 54, 55, 67, 68
  dimension 24, 25, 41, 45, 49, 50, 53, 61, 65, 69
  new 12, 13, 16, 17, 26, 29, 30, 48, 53, 58
  old 12, 13, 16, 17, 26, 29, 30, 48, 52, 53

Profile 3, 6, 16, 19, 27, 33, 66

Prompt 5, 13, 14, 16 - 22, 25 - 27, 29 - 34, 37 - 46, 48 - 54, 56, 57, 60, 67, 73, 81

Propagation 21, 23, 59, 61, 81

RADAR-C 3, 16

Range 62
  angular 22, 39
  cutoff 2, 21
  great circle 22, 39, 54, 69, 71
  ground 39, 62, 64, 69, 71
  limit 21, 22, 36, 39, 64, 69

Ray bundle 22

RAYDAT
  type 17, 26, 44, 46, 47, 50, 52, 53

RAYDAT.DAT 15, 17, 27, 28, 30, 48, 53, 68, 74

Raypath 21, 39, 62, 68, 75
  increment 23, 39, 40, 59, 69, 77
  theory 2, 3

RAYTRA4 9, 10

RAYTRA4.EXE 9, 11

RAYTRA4.FOR 6 - 8, 10

RAYTRACE 1 - 3, 5 - 18, 20, 23, 24, 27 - 29, 45 - 49, 52 - 62, 66
  - 68, 70, 71, 73 - 75, 81, 82

READER2 9, 11, 55, 56

READER2.FOR 6, 7, 9, 11

Rectangular
  grid 66
Reflection  22, 58
  loss  22, 37, 64, 69, 71, 75
Refraction  59, 60
  index of  2
Resolution
  azimuth  24, 25, 41, 43, 65
  elevation  24, 25, 41 - 43, 65
Restrictions  57, 58
Results  6, 8, 10, 12, 23, 26, 28, 47, 50, 55, 56, 59, 61, 62, 71, 74
  benchmark  57
  directory  8 - 11
  file  9, 10, 15, 21, 27, 28, 44, 45, 47, 48, 51, 52, 55, 56, 60, 70, 71
RIONO.FOR  6, 7
RMISC.FOR  6, 7
Roundoff error  23
RPHASE.FOR  6, 7
RRAYSB.FOR  6, 7
RTILT.FOR  6, 7
Selection
  menu  15, 20, 22 - 25, 34 - 44, 51, 81
Semithickness  16, 19, 32, 66, 67, 76
Session  12, 15, 26, 28, 29, 46 - 48, 52, 53, 70
Setup  1, 7, 9, 16, 60
SEZ coordinates  56
Signal
  intensity  2, 21, 36, 37, 64, 75
  loss  68, 71, 75
Simulation  4, 8, 10, 12, 19
Size  43, 61, 65
  file  7, 11, 58
  grid  31
  increment  2, 23, 59, 64, 69
Source
  code  1, 4, 6, 8 - 10, 60, 62, 67, 71, 73, 83
  files  6, 9, 11

97
Spacing  25, 43, 61, 69
       grid  18, 31, 32, 66
Spherically symmetric  29, 31, 42
Starting
   azimuth  24, 25, 41, 42, 50, 65
   elevation  24, 25, 41, 42, 50, 51, 61, 65
   height  22, 38, 39, 69
   latitude  22, 31, 32, 56
   location  20, 35, 31
   longitude  18, 22, 31, 32, 66
Subdirectory  8, 10
Sunspot number  19, 27, 33, 34, 66, 67
System  6, 16, 29, 47, 52, 54, 56, 57
   coordinate  2, 56, 71
   menu  62
   operating  3, 5, 7, 8, 10, 27, 60, 81
Terminal point  70, 71
Three-dimensional  1, 16
Tilt  6, 56
Trajectory  62
TRIAL.DAT  29, 44, 47, 48, 53, 57
TRIAL2.DAT  48
Tutorial  29, 40, 57
Update  26, 46, 47, 52
User interface  4, 6, 73
VAX  7, 58
VAX 11/785  57
VAX/VMS  5, 27, 57, 60
Vertical profile  3, 6, 16, 18, 27, 66
VMS  5, 7 – 10, 57, 60
Warning  18, 27
Wave frequency  21, 23, 36, 40, 59, 64, 69
WRITE  8, 10, 62, 70, 72
Zenith  25, 42, 50, 51, 58
RAYTRACE USER'S GUIDE

APPENDIX D:

SOURCE CODE LISTING
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naming Convention</td>
<td>i</td>
</tr>
<tr>
<td>Line numbering Convention</td>
<td>i</td>
</tr>
<tr>
<td>Flag variables</td>
<td>ii</td>
</tr>
<tr>
<td>Notes</td>
<td>iii</td>
</tr>
<tr>
<td>Hierarchy of Routines</td>
<td>iv</td>
</tr>
</tbody>
</table>

## RAYTRA4.FOR

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>1</td>
</tr>
<tr>
<td>DOXDATA</td>
<td>12</td>
</tr>
<tr>
<td>IOxMENU</td>
<td>25</td>
</tr>
<tr>
<td>IOxPRET</td>
<td>28</td>
</tr>
<tr>
<td>KEYBRD</td>
<td>30</td>
</tr>
<tr>
<td>Globals</td>
<td>33</td>
</tr>
</tbody>
</table>

## RAYSUB.FOR

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAYSUB</td>
<td>34</td>
</tr>
<tr>
<td>ROTSEZ</td>
<td>50</td>
</tr>
<tr>
<td>LATLON3</td>
<td>52</td>
</tr>
<tr>
<td>Globals</td>
<td>53</td>
</tr>
</tbody>
</table>

## RIONO.FOR

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IONOPAR</td>
<td>55</td>
</tr>
<tr>
<td>INTERP</td>
<td>69</td>
</tr>
<tr>
<td>CASE1</td>
<td>72</td>
</tr>
<tr>
<td>CASE2</td>
<td>76</td>
</tr>
<tr>
<td>CASE3</td>
<td>80</td>
</tr>
<tr>
<td>CASE4</td>
<td>84</td>
</tr>
<tr>
<td>CASE5</td>
<td>88</td>
</tr>
<tr>
<td>PGVAL</td>
<td>92</td>
</tr>
<tr>
<td>PGF1L</td>
<td>96</td>
</tr>
<tr>
<td>PGF1P</td>
<td>100</td>
</tr>
<tr>
<td>PGF2</td>
<td>103</td>
</tr>
<tr>
<td>PGFB</td>
<td>106</td>
</tr>
<tr>
<td>Globals</td>
<td>108</td>
</tr>
</tbody>
</table>
### RPHASE.FOR

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENPFT</td>
<td>110</td>
</tr>
<tr>
<td>PHSPL</td>
<td>118</td>
</tr>
<tr>
<td>PHSPL3</td>
<td>121</td>
</tr>
<tr>
<td>PHSPL2</td>
<td>123</td>
</tr>
<tr>
<td>Globals</td>
<td>124</td>
</tr>
</tbody>
</table>

### RTILT.FOR

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TILTS</td>
<td>125</td>
</tr>
<tr>
<td>HUTLT</td>
<td>128</td>
</tr>
<tr>
<td>H5TLT</td>
<td>130</td>
</tr>
<tr>
<td>H1P0LT</td>
<td>132</td>
</tr>
<tr>
<td>DHPDP</td>
<td>136</td>
</tr>
<tr>
<td>H4TLT</td>
<td>138</td>
</tr>
<tr>
<td>DH4DP</td>
<td>141</td>
</tr>
<tr>
<td>H1LTLT</td>
<td>143</td>
</tr>
<tr>
<td>DH1LDP</td>
<td>147</td>
</tr>
<tr>
<td>H2TLT</td>
<td>149</td>
</tr>
<tr>
<td>Globals</td>
<td>151</td>
</tr>
</tbody>
</table>

### RMISC.FOR

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEWCS</td>
<td>152</td>
</tr>
<tr>
<td>TRIANG</td>
<td>157</td>
</tr>
<tr>
<td>ACCFSP</td>
<td>160</td>
</tr>
<tr>
<td>FREESP</td>
<td>164</td>
</tr>
<tr>
<td>TIMES</td>
<td>166</td>
</tr>
<tr>
<td>LOSS</td>
<td>169</td>
</tr>
<tr>
<td>GCDEV</td>
<td>177</td>
</tr>
<tr>
<td>ANRANG</td>
<td>130</td>
</tr>
<tr>
<td>INBOX</td>
<td>134</td>
</tr>
<tr>
<td>INTSIGN</td>
<td>136</td>
</tr>
<tr>
<td>Globals</td>
<td>136</td>
</tr>
</tbody>
</table>
Variables in the source code are generally named according to the following convention. In some cases, variables do not fall neatly into an appropriate category and the naming necessarily becomes a bit arbitrary. Cases that do not follow this convention at all are ones contained in code that has been obtained from other sources. Names are defined by:

(2 or 3-char prefix) (separator char) (mnemonic suffix)

(Type) (Purpose) (Context) x (Mnemonic)

**Type:**
- I INTEGER*4
- J INTEGER*8
- F REAL*4
- R REAL*8
- C COMPLEX*16
- K CHARACTER
- L LOGICAL

**Purpose:**
- C Constant
- V Variable
- T Temporary
- P Parameter
- F Flag

**Context:**
- none in MAIN program
- L local to a subprogram
- S passed as an argument
- C passed in a COMMON block

**EXAMPLES:**
- RTLxH == Real*8, temporary, local variable H
- IVxELLIM == Integer*4, variable ELLIM in MAIN
Line Numbering Convention

Line numbers are generally designated as follows:

<table>
<thead>
<tr>
<th>#</th>
<th>#</th>
<th>#</th>
<th>#</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>designates non-zero for general use as in FOR-loops.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>major indicates branching, and with FORMATs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>subdivisions FORMAT stmt.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Flag Variables

IFxEXI: 1 indicates user's choice to quit the program, 0 otherwise

IFxNEW: 1 indicates a new problem is being done, 0 otherwise

IFxSTAY: 1 indicates that the user is continuing a problem, 0 otherwise

IFxBUN: # of extra ray in ray bundle for calculation of intensity

IFxGEN: 1 = no special condition exists 2 = ray increment has been adjusted so that a layer boundary is not overshot 3 = a z-reflection has occurred 4 = an x-reflection has occurred 5 = a y-reflection has occurred 6 = ray is traveling downward in free space propagation 7 = ray is traveling upward in free space propagation
IFxN4:
1 if the present location is not interior to a triangle of grid points for interpolation
2 otherwise

IFxGRND:
1 if reflection loss calculations are to use soil's conductivity
2 if water's conductivity is to be used

IFxSFL:
0 to activate signal intensity calculations
1 to deactivate them

IFLxOUT:
1 if the present location is outside the grid of points
0 otherwise

Notes

There will be discrepancies in the consecutive numbering of source code lines within a source code file, as the blank lines between subroutines have been skipped. Also, the consecutive numbering runs to the end of a particular source file, starting over for the next file.

The symbol table after each subroutine contains a column labeled "Class". This corresponds to the portion of the variable prefix denoted as "Context" in the naming convention above. The notable difference being that a Class of 'param' is the same as a prefix of 'S', meaning that the quantity was passed as an argument.

Also in the symbol table, the character 'x', which is used under the variable naming convention to connect the prefix with the mnemonic suffix, has been converted to an upper case letter, 'X'.
Hierarchy of Routines

Main

- DOxDATA
- IOxPRET
- KEYBRD
- RAYSUB

- IOxMENU
- KEYBRD

- IOxPRET
- KEYBRD

- KEYBRD

- ANRANG
- GCDEV
- LATLON3
- NEWCS
- PHSPL
- ROTSEZ
- TIMES

- ENDPT
- LOSS
- FREESP
- TILTS

- ACCFSP

- PHSPL2
- PHSPL3

- H1LTLT
- H1PTLT
- H2TLT
- H4TLT
- H5TLT
- H6LT

- DH1LDP
- DH1FDP
- DH4DP

- CASE1
- INBOX
- CASE2
- CASE3
- INTERP
- CASE4
- TRIANG
- CASE5

- PGVAL
- PGF2
- PGFB
- PGVAL
- PGFL
- PGF2
- PGFB
- PGVAL
- PGF2
- PGFL
- PGFB
- PGF2

- PGVAL
- PGFL
- PGFB
- PGFL
- PGFB
- PGFB

105
PROGRAM RAYTRACE

C
---

RAYTRACE -- 3D RAYTRACING WITH ACCELERATED FREE SPACE PROPAGATION CALCULATION

C
---

AUTHORS: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 03/19/83

VERSION: 4.1

---------------------------------------------------------------

REVISION: 07/25/86 -- Initial revision. Translated from Tektronix Basic to VAX Fortran by Eric L. Strobel.

07/30/86 -- V1.1. Change over to use of REAL*8 precision in the calculations.

08/12/86 -- V2.0. Add the capability to perform propagation loss calculations. Also greatly massaged over the output. A number of minor and MAJOR bugs corrected.

09/05/86 -- V2.3. Add horizon focusing and handle low angle rays. Record data for proper handling of skip focusing. Record maximum height. Calculate deviation from great circle path. Minor bug fixes.

09/17/86 -- V2.4. Fix calculation of great circle path deviation. Introduce cutoff in number of bounces. Automagically obtain approximate bottom of the ionosphere. Eliminating the need for the prompt. Also fixed rotation matrix used in calculating the new c-values.

09/17/86 -- V2.5. Fix calculation of great circle path deviation. Introduce cutoff in number of bounces. Automagically obtain approximate bottom of the ionosphere. Eliminating the need for the prompt. Also fixed rotation matrix used in calculating the new c-values.

10/16/86 -- V3.0. Lots of changes. Read & write to files. User-friendly means of getting information into the program. The old Raytrace is now a subroutine. Capability to do many rays at a time. Able to handle large ionospheric specifications. Able to change problem parameters without restarting.

106
the program.


09/01/87 -- V4.0. Major revision. Now uses the RADAR-C ionosphere. Able to launch rays inside and above the ionosphere. Calcs. & rpts. the direction cosines now. Uses new routine to obtain user info. Absorption & excess loss calcs. commented out. Geometric loss now done correctly. Phase path calculated. Data files now BINARY (vs. ASCII). Upper bound at radius of GEO dst. Derivatives of range w.r.t. elev. are incorrect but unfixed at this point. WARNING: Cannot send rays arbitrarily close to vertical!!! C-values are now properly normalized.

09/30/87 -- V4.1. Phase path length calculations added. Tilt routines changed to conform to the requirements of using the RADAR-C model.

01/15/88 -- V4.2. Bug fixes and minor corrections made over a period of time. Added feature: will now stop the ray at the designated range. not at the end of the next increment, as it previously did. Also, some disused variables have been weeded out and some arrays which were larger than currently necessary have been redimensioned. The code is now smaller and it runs faster. The source code is now somewhat universal, in that only the slightest of changes is needed to convert from VAX to DOS or back. or between various DOS Fortrans.

03/18/88 -- V4.3. A number of corrections and improvements have been made. Some elevation and azimuth options were being reset when they shouldn't have been (under some circumstances). Angular range is now correctly calculated. Before, it was based on the launch azimuth, which isn't correct if the ray bends. Now, the actual azimuth between the two points is calculated and used. Also, the actual angular range, and not the cosine of the angular range, is used for the range cutoff.

Performs 3-D raytracing of radio propagation through the ionosphere. Incorporates a specific model for true height profiles of electron
density. Can utilizes data from ionospheric sounders.

INTEGER ITXQ, IFxNEW, IFxSTAY, IFxEXI, IVxIND
INTEGER IVxAZLIM, IVxELLIM, IVxPARAM(6), IVxLAUN(3)
INTEGER IVxVAR, IVxTYP
REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7)
REAL*8 RVxOPTION(7), RVxAZI, RVxELEV
REAL*8 RVxBOUN(11,15), RVxVAR
REAL*8 RVxION1(30,60), RVxION2(30,60), RVxION3(30,60)
REAL*8 RVxION4(30,60), RVxION5(30,60), RVxION6(30,60)
CHARACTER*10 KVxDFIL, KVxTMP, KVxOFIL
CHARACTER*2 KVxIND
CHARACTER*41 KTxANS, CH
CHARACTER*40 KVxPMT
COMMON /MAINDAT/ RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN.
* RVxLAUN. RVxOPTION
COMMON /RESULTS/ RVxBOUN

Stuff the big array with pieces of more manageable size.

This is in order to compensate for the unfortunate DCS
restrictions on record sizes.

EQUIVALENCE (RVxION1,RVxIONPT), (RVxION2,RVxIONPT(1,1,2))
EQUIVALENCE (RVxION3,RVxIONPT(1,1,3))
EQUIVALENCE (RVxION4,RVxIONPT(1,1,4))
EQUIVALENCE (RVxION5,RVxIONPT(1,1,5))
EQUIVALENCE (RVxION6,RVxIONPT(1,1,6))

Data setup section

KVxPMT = ' '
IVxTYP = 0
ITXQ = 0
IVxVAR = 0
RVxVAR = 0.0
KTxTMP = ' '
IVxIND = 40
IVxELLIM = 0
IVxAZLIM = 0
RVxAZI = 0.0D00
RVxELEV = 0.0D00
IF $\text{STAY} = 1$ indicates that the user has responded with 'yes' to the query about performing another problem.

IF (IF$\text{STAY}.EQ.1) \text{ GO TO 20000}

WRITE (6,11001)

10005 KV$\text{PMT} = \text{ 'Is this a new (1) or old (0) problem? '}

A new problem is one for which no RAYDAT.DAT or similar file exists. Such a file does exist for an old problem and contains the user inputs from a previous run.

CALL KEYBRD(KV$\text{PMT}.1,.ITXQ,.IF$\text{NEW},.RV$\text{VAR},.KV$\text{TMP})

IF (.ITXQ.LT.1.OR....NOT.(IF$\text{NEW}.EQ.1.OR..IF$\text{NEW}.EQ.0).

* \text{ GO TO 10005}

-----------------------------------------------------------------------

Section to handle the setup of the problem's data
by reading from file and editing some of the data.

-----------------------------------------------------------------------

IF (IF$\text{NEW}.EQ.1) THEN

-----------------------------------------------------------------------

IF$\text{NEW} = 1 \rightarrow$ new problem, so DO$\text{xDATA}$ to get the user options and ionospheric data, then write out to a datafile.

-----------------------------------------------------------------------

WRITE (6,11004)
KV$\text{DFIL} = \text{ 'RAYDAT.DAT'}
KV$\text{PMT} = \text{ 'information (10 char max.: '}

Ask the user where to store the problem's information.

CALL KEYBRD(KV$\text{PMT}.J,.ITXQ,.IV$\text{VAR},.RV$\text{VAR},.KV$\text{DFIL})

OPEN (30,.FILE=KV$\text{DFIL},.FORM='UNFORMATTED'.STATUS='NEW'

CALL DO$\text{xDATA}(IF$\text{STAY},.IF$\text{NEW},.IF$\text{EXI},

REWIND (30)

WRITE (30) RV$\text{GRID}

WRITE (30) ((RV$\text{ION1}(I,J),J=1,RV$\text{GRID6},I=1.RV$\text{GRID5}

WRITE (30) ((RV$\text{ION2}(I,J),J=1,RV$\text{GRID6},I=1.RV$\text{GRID5}

WRITE (30) ((RV$\text{ION3}(I,J),J=1,RV$\text{GRID6},I=1.RV$\text{GRID5}

WRITE (30) ((RV$\text{ION4}(I,J),J=1,RV$\text{GRID6},I=1.RV$\text{GRID5}

WRITE (30) ((RV$\text{ION5}(I,J),J=1,RV$\text{GRID6},I=1.RV$\text{GRID5}

WRITE (30) ((RV$\text{ION6}(I,J),J=1,RV$\text{GRID6},I=1.RV$\text{GRID5}

WRITE (30) IV$\text{PARAM}
WRITE (30) IVxLAUN
WRITE (30) RVxLAUN
WRITE (30) RVxOPTION
CLOSE(30)

IF IFxEXI = 1 indicates that the user has chosen to quit the program.

IF IFxEXI.EQ.1 GO TO 20700
ELSE IF IFxNEW.EQ.0 THEN

------------------------------------------------------------------
IFxNEW = 0 --> old problem, so read from the data file.
then DOxDATA to check for any changes the user wants to make to the options.
------------------------------------------------------------------

WRITE (6,11004)
KVxDFIL = 'RAYDAT.DAT'
KVxPMT = 'information (10 char max):'

Ask the user where to store the problem's information.

CALL KEYBRD(KVxPMT, ITxQ, IVxVAR, RVxVAR, RVxDFIL)
OPEN (30, FILE=KVxDFIL, FORM='UNFORMATTED', STATUS='OLD')
REWIND (30)
READ (30) RVxGRID
READ (30) ((RVxION1(I,J),J=1, RVxGRID(6),I=1,RVxGRID(5))
READ (30) ((RVxION2(I,J),J=1, RVxGRID(6),I=1,RVxGRID(5))
READ (30) ((RVxION3(I,J),J=1, RVxGRID(6),I=1,RVxGRID(5))
READ (30) ((RVxION4(I,J),J=1, RVxGRID(6),I=1,RVxGRID(5))
READ (30) ((RVxION5(I,J),J=1, RVxGRID(6),I=1,RVxGRID(5))
READ (30) ((RVxION6(I,J),J=1, RVxGRID(6),I=1,RVxGRID(5))
READ (30) IVxPARAM
READ (30) IVxLAUN
READ (30) RVxLAUN
READ (30) RVxOPTION
CALL DOxDATA(IFxSTAY, IFxNEW, IFxEXI)
IF (IFxEXI.EQ.1) THEN

The user has chosen to quit, so close the file and quit.

CLOSE (30)
GO TO 20700
ENDIF

10010 KVxDFIL = 'Overwrite existing file (Y/N)?'

Give the user the choice of replacing the old problem's data with the (possibly new) current data, or
CALL KEYBRD(KVxPMT, ITxQ, IVxVAR, RVxVAR, KTxANS)
IF (KTxANS.EQ.'Y') THEN
  IF (KTxANS.EQ.'N') THEN
    CLOSE(30)
  KVxPMT = 'Backup the datafile (Y/N)?'
  ENDIF
  CALL KEYBRD(KVxPMT, ITxQ, IVxVAR, RVxVAR, KTxANS)
ENDIF
GO TO 10015
ELSE IF (KTxANS.EQ.'N') THEN
  KVxPMT = 'New filename (10 char max)?'
ENDIF
GO TO 20000
ELSE
  KVxDFIL = 'New filename (10 char max)?'
ENDIF
GO TO 10010
ENDIF
GO TO 10005
ELSE
  ENDIF
ENDIF
FORMAT ('Enter filename for storage of ionospheric')
End data set up section.

If we've been running and decide to do another problem, (STAY in the program) re-cycle starts here. DOxDATA to get the (possibly new) user values for the next problem & give the user the opportunity to update the datafile.

CALL DOxDATA(IFxSTAY, IFxNEW, IFxEXI)

IFxEXI = 1 indicates that the user has decided to quit.

IF (IFxEXI.EQ.1) GO TO 20700

KTxANS = ' '

KVxPMT = 'Update the datafile (Y/N)?'

CALL KEYBRD(KVxPMT, J, ITxQ, IVxVAR, RVxVAR, KTxANS)

IF (KTxANS.EQ. 'Y') THEN

OPEN (30, FILE = KVxDFIL, FORM='UNFORMATTED')

REWIND (30)

WRITE (30) RVxGRID
WRITE (30) ((RVxION1(I,J),J=1,RVxGRID(6)).I=I
WRITE (30) ((RVxION2(I,J),J=1,RVxGRID(6)).I=I
WRITE (30) ((RVxION3(I,J),J=1,RVxGRID(6)).I=I
WRITE (30) ((RVxION4(I,J),J=1,RVxGRID(6)).I=I
WRITE (30) ((RVxION5(I,J),J=1,RVxGRID(6)).I=I
WRITE (30) ((RVxION6(I,J),J=1,RVxGRID(6)).I=I
WRITE (30) IVxPARAM
WRITE (30) IVxLAUN
WRITE (30) RVxLAUN
WRITE (30) RVxOPTION
CLOSE (30)

ENDIF

KTxANS = ' '
Begin actually firing out some rays. Results are stored in consecutively numbered files, FORnnn.DAT, starting with nnn = 040. Beware that if the program is run on DOS or similar systems, starting another run of the program may cause data loss, since the new result files will simply write over the old ones.

--------------------------------------------------------------

First, select the appropriate limits on the DO loops.

IF (IDINT(RVXOPTION(1)).LE.1) THEN
  IVXELLIM = 1
  IVXAZLIM = 1
ELSE IF (IDINT(RVXOPTION(1)).EQ.1) THEN
  IVXAZLIM = 1
  IVXELLIM = 1+IDINT((RVXOPTION(7)-RVXOPTION(3))/RVXOPTION(5))
ENDIF

BEGINNING RAY LOOPS.

These first numbers are written to the result file so that an analysis program can determine how many points of information there are.

DO 20100 J = 1, IVXAZLIM
   DO 20200 I = 1, IVXELLIM
      RVXAZI = RVXOPTION(2) + (I-1)*RVXOPTION(4)
      IF (RVXAZI.GE.360.0D0) RVXAZI = RVXAZI - 360.0
      RVXELEV = RVXOPTION(3) + (J-1)*RVXOPTION(5)
      WRITE(IIVXIND, 21003) RVXAZI, RVXELEV
   END DO 20200
END DO 20100

Print *,' BEGINNING RAY LOOPS.'
RAYSUB is the routine that actually traces the rays.

CALL RAYSUB(RVxAZI, RVxELEV)

Now, write the results of the ray. The elements of the result array are discussed in the comment block near the beginning of RAYSUB.

WRITE(IVxIND, 21003) ((RVxBOUN(L,K),K=1,15), L=1.

DO 20099 K = 1, 15
    DO 20098 L = 1. IVxLAUN(L)=1

Clear the result array for the next ray.

RVxBOUN(L,K) = 0.0

CONTINUE

CONTINUE

CONTINUE

CLOSE(IVxIND)

PRINT *, ' RAY LOOPS DONE.'

PRINT *, ' RAY LOOPS DONE.'

WRITE (6,21004) KVxOFIL

Increment the unit number that the results will be written to.

IVxIND = IVxIND + 1

End section that shoots rays.

KVxPMT = 'Do another problem (Y/N)?'

Give the user a chance to quit.

CALL KEYBRD(KVxPMT, 3, ITxQ, IVxVAR, RVxVAR, KTxANS)

IF (KTxANS.EQ.'Y') THEN
    IFxSTAY = 1
    GO TO 10000

ELSE IF (ITxQ.LT.1.OR.KTANS.NE.'N') THEN

GO TO 20500
NDERIF
KTXANS = ' '
20700 CALL IOXPRET(CH)
C
C Now, give the user the chance to abort the quit.
C
IF (CH.EQ. 'A') GO TO 20500
C
PRINT the closing message.

WRITE (6,21007)
WRITE (6,21008)
WRITE (6,21009)
21002 FORMAT (A1)
21003 FORMAT (G24.14)
21004 FORMAT (/*,IX,'Results written to file ',A10,/*)
21007 FORMAT (IX,'Session done. For safety's sake, copy the data'
21008 FORMAT (IX,'and result files into a separate directory to'
21009 FORMAT (IX,'prevent accidental overwriting. ')
22000 FORMAT (I4)
END

main Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITXQ</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>I</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>J</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>K</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>CH</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
</tr>
<tr>
<td>L</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IFXEXI</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IFXNEW</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVXIND</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>KVXIND</td>
<td>local</td>
<td>CHAR*2</td>
<td>2</td>
</tr>
<tr>
<td>KVXDFIL</td>
<td>local</td>
<td>CHAR*10</td>
<td>10</td>
</tr>
<tr>
<td>KTXANS</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
</tr>
<tr>
<td>IVXVAR</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXOFIL</td>
<td>local</td>
<td>CHAR*10</td>
<td>10</td>
</tr>
<tr>
<td>RVXAZI</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>KTXMP</td>
<td>local</td>
<td>CHAR*10</td>
<td>10</td>
</tr>
<tr>
<td>IFXSTAY</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXVAR</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>KVXPM</td>
<td>local</td>
<td>CHAR*40</td>
<td>40</td>
</tr>
<tr>
<td>IVXELLIM</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXELLEV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IVXTYP</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Type</td>
<td>Size</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>IVXAZLIM</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVXPARAM</td>
<td>MAINDAT</td>
<td>INTEGER*4</td>
<td>24</td>
</tr>
<tr>
<td>IVXLAUN</td>
<td>MAINDAT</td>
<td>INTEGER*4</td>
<td>12</td>
</tr>
<tr>
<td>RVXGRID</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>48</td>
</tr>
<tr>
<td>RVXIONPT</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>36400</td>
</tr>
<tr>
<td>RVXLAUN</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>56</td>
</tr>
<tr>
<td>RVXOPTION</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>56</td>
</tr>
<tr>
<td>RVXBOUN</td>
<td>RESULTS</td>
<td>REAL*8</td>
<td>1320</td>
</tr>
<tr>
<td>RVXION1</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVXION2</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVXION3</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVXION4</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVXION5</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVXION6</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
</tbody>
</table>
SUBROUTINE DOXDATA(IFSxSTA, IFSxNEW, IFSxEXI)

DOXDATA -- SUBROUTINE TO MAINTAIN AND ACCESS THE DATA FOR CONVENIENT USE OF THE WHOLE RAYTRACING PROGRAM

CALLED BY: MAIN
CALLS: DOXPRET, KEYBRD

AUTHOR: ERIC L. STROBEL
DATE: 01/15/88
VERSION: 2.1
REVISED: 10/09/86 -- V1.0. Initial revision.
09/01/87 -- V2.0. Uses the new KEYBRD routine and unformatted reads and writes.
01/15/88 -- V2.1. Contains necessary concessions to the limited environment that DOS provides.

USES: IFSxSTA A FLAG THAT INDICATES THAT THE USER HAS REMAINED IN THE PROBLEM
IFSxNEW A FLAG THAT INDICATES THAT THIS IS A NEW PROBLEM
TO DECIDE HOW TO PRESENT AN INTERFACE TO THE USER, IN ORDER TO MAINTAIN THE DATABASE FILES

RETURNS: IFSxEXI A FLAG THAT INDICATES THAT THE USER HAS CHOSEN TO EXIT

INTEGER IFSxSTA, IFSxNEW, IFSxEXI, ITLxQ
INTEGER IVxPARAM(6), ITLxCL, ITLxC2, ITLxC3
INTEGER ITLxTMP, IVxLAUN(3), IVLxTYP, IVLxVAR
REAL*8 RVLxLAT, RVLxLON, RVxGRID(6), RVxIONPT(30,60,117)
REAL*8 RTLx1, RTLx2, RVxLAUN(7), RVxOPTION(7), RTxA
REAL*8 RVxVAR. RPxDTOR
REAL*8 RVxION1(30.60), RVxION2(30.60), RVxION3(30.60)
REAL*8 RVxION4(30.60), RVxION5(30.60), RVxION6(30.60)

CHARACTER*10 KVLxFIL, KTLxTMP
CHARACTER*40 KVLxPMT
CHARACTER*1 KVLxANS
CHARACTER*7 KCLxLBL

COMMON / MAINDAT/ RVxGR:D,RVxZONPT, VxPARAM,:*VxLAUN.
4
RVxLAUN, RVxOPTION

C
C Stuff the big array with pieces of more manageable size.
C This is in order to compensate for the unfortunate DOS
C restrictions on record sizes.

EQUIVALENCE (RVxION1, RVxIONPT), (RVxION2, RVxIONPT(1,1,2))
EQUIVALENCE (RVxION3, RVxIONPT(1,1,3))
EQUIVALENCE (RVxION4, RVxIONPT(1,1,4))
EQUIVALENCE (RVxION5, RVxIONPT(1,1,5))
EQUIVALENCE (RVxION6, RVxIONPT(1,1,5))

DATA KCLxLBL /'foE'**2', 'hmF1', 'foFl'**2', 'ymF2'/

-----------------------------------------------------------
KVLxPMT = ' '
IVLxTYP = 0
IVLxVAR = 0
RVxVAR = 0.0
KTLxTMP = ' '
RPxDTOR = 0.0174532925D00

-----------------------------------------------------------
If this is the first time thru on a new problem, read
in either an ionospheric grid produced by another
program (RADAR-C right now) or a grid that is to be
entered by hand.

-----------------------------------------------------------
10000 IF (IFSxNEW.EQ.1.AND.IFSxSTA.EQ.0) THEN
WRITE(6,11001)
WRITE(6,11002)
KVLxFIL = 'GRID.DAT'
KVLxPMT = ' (type NONE if none exists) '
CALL KEYBRD(KVLxPMT,3, TLxQ,IVLxVAR, RVxVAR, KVLxFIL)
IF (KVLxFIL.EQ.'NONE') THEN
No external grid file exists, so build it by hand.

OPEN(20, FILE='GRID.DAT', FORM='UNFORMATTED',
      STATUS='NEW')
REWIND (20)

WRITE (6, '(X'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'/ '/'

RVxGRID gives info needed to obtain the lat's & lon's of the grid.

KVLxPMT = 'Input lat grid spacing (deg):'
RTLx1 = RVxGRID(1)
RTLx2 = RVxGRID(2)

Prompt for the latitude spacing of the lat-lon grid.

CALL KEYBRD(KVLxPMT, 2, ITLxQ, IVLxVAR, RTLx1, KTLxTMP)
IF (ITLxQ.GE.1) RVxGRID(1) = RTLx1
KVLxPMT = 'Input lon grid spacing (deg):'

Prompt for the longitude spacing of the lat-lon grid.

CALL KEYBRD(KVLxPMT, 2, ITLxQ, IVLxVAR, RTLx2, KTLxTMP)
IF (ITLxQ.GE.1) RVxGRID(2) = RTLx2

WRITE(6, 11007)
KVLxPMT = 'LATITUDE :'
RTLx1 = RVxGRID(3)

Prompt for the grid's starting latitude.

CALL KEYBRD(KVLxPMT, 2, ITLxQ, IVLxVAR, RTLx1, KTLxTMP)
IF (ITLxQ.GE.1) RVxGRID(3) = RTLx1
KVLxPMT = 'LONGITUDE (east = positive):'
RTLx2 = RVxGRID(4)

Prompt for the grid's starting longitude.

CALL KEYBRD(KVLxPMT, 2, ITLxQ, IVLxVAR, RTLx2, KTLxTMP)
IF (ITLxQ.GE.1) RVxGRID(4) = RTLx2
KVLxPMT = 'Input # of grid points in lat.:
RTLx1 = RVxGRID(5)

Prompt for the number of grid point in the latitude direction.

CALL KEYBRD(KVLxPMT, 2, ITLxQ, IVLxVAR, RTLx1, KTLxTMP)
IF (ITLxQ.GE.1) RVxGRID(5) = RTLx1
Prompt for the number of grid point in the longitude direction.

CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)

IF (ITLxQ.GE.1) RVxGRID(6) = RTLx1

Verify the grid inputs and give the user a chance to re-enter them to fix errors.

KVLxANS = 'Y'

KVLxPMT = 'Grid setup OK (Y/N)?'

CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR,KVLxANS)

IF (KVLxANS.EQ.'N'.OR.KVLxANS.EQ.'n') GO TO 10020

Write the results out into a GRID file.

WRITE(20) RVxGRID

WRITE (6,'(1X,/////////////////////////////////)

DO-loops for the entry of ionospheric parameters at each grid point.

DO 10200 I=1, RVxGRID(5)

RVLxLAT = RVxGRID(3) + (I-1)*RVxGRID(1)

DO 10100 J=1, RVxGRID(6)

RVLxLON = RVxGRID(4) - (J-1)*RVxGRID(2)

WRITE (6,12002) RVLxLAT, RVLxLON

DO 10050 K=1,6

The elements of KCLxLBL may be found in the DATA statement at the beginning of this routine.

KVLxPMT = 'Input '/KCLxLBL(K)/':'

RTLx1 = RVxIONPT(I,J,K)

CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)

IF (ITLxQ.GE.1) RVxIONPT(I,J,K) = RTLx1

CONTINUE

KVLxANS = 'Y'

KVLxPMT = 'Profile inputs OK (Y/N)?'

For each grid point, verify correct entry of values and give the user a chance to fix errors.

CALL KEYBRD(KVLxPMT,3,ITLxQ,IVLxVAR,RVLxVAR,KVLxANS)

IF (KVLxANS.EQ.'N'.OR.KVLxANS.EQ.'n') GO TO 10049

WRITE (6,'(1X,/////////////////////////////////)'
Write the grid file in the same manageable pieces that everything else expects it to be in.

```
WRITE (20) ((RVXION1(I,J),J=1,RVXGRID(6)),I=1
* RVXGRID(5))
WRITE (20) ((RVXION2(I,J),J=1,RVXGRID(6)),I=1
* RVXGRID(5))
WRITE (20) ((RVXION3(I,J),J=1,RVXGRID(6)),I=1
* RVXGRID(5))
WRITE (20) ((RVXION4(I,J),J=1,RVXGRID(6)),I=1
* RVXGRID(5))
WRITE (20) ((RVXION5(I,J),J=1,RVXGRID(6)),I=1
* RVXGRID(5))
WRITE (20) ((RVXION6(I,J),J=1,RVXGRID(6)),I=1
* RVXGRID(5))
```

Do a series of prompts for the sunspot number and time values.

```
KVLXPMT = 'Input (integer) sunspot number: '
CALL KEYBRD(KVLXPMT,1,ITLXQ,ITLXTMP,RVLXVAR,RTLXTMP)
IF (ITLXQ.GE.1) IVXPARAM(1) = ITLXTMP

KVLXPMT = 'Input year: '
ITLXTMP = IVXPARAM(2)
CALL KEYBRD(KVLXPMT,1,ITLXQ,ITLXTMP,RVLXVAR,RTLXTMP)
IF (ITLXQ.GE.1) IVXPARAM(2) = ITLXTMP

KVLXPMT = '... month: '
ITLXTMP = IVXPARAM(3)
CALL KEYBRD(KVLXPMT,1,ITLXQ,ITLXTMP,RVLXVAR,RTLXTMP)
IF (ITLXQ.GE.1) IVXPARAM(3) = ITLXTMP

KVLXPMT = '... day: '
ITLXTMP = IVXPARAM(4)
CALL KEYBRD(KVLXPMT,1,ITLXQ,ITLXTMP,RVLXVAR,RTLXTMP)
IF (ITLXQ.GE.1) IVXPARAM(4) = ITLXTMP

KVLXPMT = 'Input UT time (hr) : ' 
ITLXTMP = IVXPARAM(5)
CALL KEYBRD(KVLXPMT,1,ITLXQ,ITLXTMP,RVLXVAR,RTLXTMP)
IF (ITLXQ.GE.1) IVXPARAM(5) = ITLXTMP

KVLXPMT = '... UT time (min): ' 
ITLXTMP = IVXPARAM(6)
CALL KEYBRD(KVLXPMT,1,ITLXQ,ITLXTMP,RVLXVAR,RTLXTMP)
IF (ITLXQ.GE.1) IVXPARAM(6) = ITLXTMP
```
WRITE (20) IVxPARAM
KVLPMT = 'S.S. # and times OK (Y/N)? '
KVLANS = 'Y'

Again, give the user a chance to verify the entries and to correct errors.

CALL KEYBRD(KVLPMT, 3. ITLxQX IVxVARxVARxVARxVARxKVLANS);
IF (KVLANS.EQ. 'N'. OR. KVLANS.EQ. 'n') GO TO 10201

Close the GRID file and go on to get the rest of the user's input.

CLOSE(20)
GO TO 20000
ENDIF

Read the externally constructed grid file.

OPEN(20, FILE=KVLFILxFORM='UNFORMATTED')
REWIND (20)
READ (20) RVxGRID
READ (20) ((RVxION1(I, J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
READ (20) ((RVxION2(I, J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
READ (20) ((RVxION3(I, J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
READ (20) ((RVxION4(I, J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
READ (20) ((RVxION5(I, J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
READ (20) ((RVxION6(I, J), J=1, RVxGRID(6)), I=1, RVxGRID(5))
READ (20) IVxPARAM
CLOSE(20)
ENDIF

FORMAT(11001)
 FORMAT(11002)

Done reading in data, now build/alter database of options, by processing menu choices.

It should be noted that by simply adding to the list of lines contained in the computed GO TOs, additional items may be attached to the existing menus, or additional menus may be added. If the menu system is extended in this way, the routine IOxMENU is of sufficient generality to handle this. The appropriate menu
files must be changed/added and care should be
taken to make sure that any one menu list doesn’t
become so large that part of it scrolls up off the
screen before the prompt comes up. The menu file
scheme is discussed in a comment block in the
routine IOXMENU.

---

MENU 1 -- Main menu: 4 choices

CALL IOXMENU(1, ITLxCl)
GO TO (20010, 20200, 20400, 20600), ITLxCl

---

MENU 2 -- Launch parameters menu: 9 choices

CALL IOXMENU(2, ITLxC2)
GO TO (20020, 20030, 20040, 20050, 20060, 20070, 20080, 20090, 20000), ITLxC2

---

IF (IVxLAUN(1).EQ.0) IVxLAUN(1)=1
KVLxPMT = 'Bounce limit'

CALL KEYBRD(KVLxPMT, ITLxQ, ITLxQTMP, KTLxTMP)
IF (ITLxQ.GE.1) IVxLAUN(1) = IVxLAUN(1)
ITLxTMP = IVxLAUN(1)

Prompt for the limit in the number of times the ray
will be allowed to return to earth, if this is in
fact an applicable concept. The default value is
1.

CALL KEYBRD(KVLxPMT, ITLxQ, ITLxQTMP, RVLxVAR, KTLxTMP)
IF (ITLxQ.GE.1) IVxLAUN(2) = ITLxTMP
GO TO 20010

KVLxPMT = 'Do signal intensity (no = l)?'
ITLxTMP = IVxLAUN(2)

Prompt to find out whether or not the user wants signal
intensity to be calculated.

CALL KEYBRD(KVLxPMT, ITLxQ, ITLxQTMP, RVLxVAR, KTLxTMP)
IF (ITLxQ.GE.1) IVxLAUN(2) = ITLxTMP
GO TO 20010

KVLxPMT = 'Reflect from ground(l) or water(0)?'
ITLxTMP = IVxLAUN(3)

Prompt the user to choose a generic conductivity for the
reflection loss calculation at an earth bounce point.

Somedy this should be replaced with a global conductivity
CALL KEYBRD(KVLxPMT,1,ITLxQ,ITLxTMP,RVLxVAR,KTLxTMP)
IF (ITLxQ.GE.1) RVxLAUN(3) = ITLxTMP
GO TO 20010
KVLxPMT = 'Enter launch pt. latitude '
Do a pair of prompts for the latitude, longitude location of the starting point of the ray.
RTLx1 = RVxLAUN(1)
CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
IF (ITLxQ.GE.1) RVxLAUN(1) = RTLx1
RTLx2 = RVxLAUN(2)
CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
IF (ITLxQ.GE.1) RVxLAUN(2) = RTLx2
GO TO 20010
KVLxPMT = 'Enter starting ht. (km) '
RTLx1 = RVxLAUN(6)
Prompt for the altitude of the starting point for the ray.
CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
IF (ITLxQ.GE.1) RVxLAUN(6) = RTLx1
GO TO 20010
IF (RVxLAUN(3).EQ.0.0) RVxLAUN(3) = 3000.0D0
IF (RVxLAUN(7).EQ.0.0) RVxLAUN(7) = 36000.0D0
A pair of prompts to establish the range and altitude cutoff values necessary to prevent the problem from running on forever. The range is the ground range along the ground track of the ray, irregardless of the altitude and actual path of the ray. The default value for the ground range is 3000 km. and the default for the altitude limit is 36000 km. or approximately geostationary altitude.
KVLxPMT = 'Enter range limit (km) '
RTLx1 = RVxLAUN(3)
CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx1,KTLxTMP)
IF (ITLxQ.GE.1) RVxLAUN(3) = RTLx1
KVLxPMT = 'Enter altitude limit (km) '
RTLx2 = RVxLAUN(7)
CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLx2,KTLxTMP)
IF (ITLxQ.GE.1) RVxLAUN(7) = RTLx2
GO TO 20010
IF (RVxLAUN(4).EQ.0.0) RVxLAUN(4) = 4.0D0
KVLxPMT = 'Enter raypath increment (km) '
RTLx1 = RVxLAUN(4)
Prompt for the size of the nominal raypath increment. The default value of 4 km represents a compromise between execution time (longer increments = faster execution) and accuracy (shorter increments = more accurate).

CALL KEYBRD(KVLXPM1,2,ITLXQ,IVLXVAR,RTLX1,RTLXTMP)
IF (ITLXQ.GE.1) RVXLAUN(4) = RTLX1
GO TO 20010
KVLXPM1 = 'Enter wave frequency (MHz)' RVXLAUN(5) = 5.0D00
RTLX1 = RVXLAUN(5)

Prompt for the frequency of the HF transmission that the ray is supposed to represent. The default value of 5 MHz is somewhat arbitrary. Since the program performs frequent divisions by wave frequency squared, some sort of non-zero default value is needed just to prevent 'unexplained' crashes of the program for new problems.

CALL KEYBRD(KVLXPM1,2,ITLXQ,IVLXVAR,RTLX1,RTLXTMP)
IF (ITLXQ.GE.1) RVXLAUN(5) = RTLX1
GO TO 20010

------------------------------
MENU 3 -- Elevation parameters menu: 8 choices.
------------------------------
CALL IOXMENU(3,ITLXC3)
GO TO (20210,20220,20230,20240,20250,20260,20270,20000,ITLXC3)
KVLXPM1 = 'Enter dimension of problem'
RTLX1 = RVXOPTION(1)
The dimension of the problem goes as follows (the reason for the term dimension should become obvious):
0 = a single ray (a single point on the sky)
1 = a vertical fan of rays at a single azimuth (a line of points on the sky)
2 = a set of fans of rays (an array of points on the sky)

CALL KEYBRD(KVLXPM1,2,ITLXQ,IVLXVAR,RTLX1,RTLXTMP)
IF (ITLXQ.LT.1) THEN
GO TO 20200
ELSE IF (IDNINT(RTLX1).GT.2.OR.IDNINT(RTLX1).LT.0) THEN
GO TO 20210
ELSE
RVxOPTION(1) = RTLxl
GO TO 20200
ENDIF
20220 FVLxPMT = 'Enter starting azimuth (deg) '
RTLxl = RVxOPTION(2)
C
Prompt for the initial azimuth for the problem.
C
CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLxl,KTLxTMP)
IF (ITLxQ.GE.1) RVxOPTION(2) = RTLxl
GO TO 20200
20230 WRITE (6,22009)
RTLxl = RVxOPTION(3)
C
Prompt for the initial elevation for the problem.
C
For a starting point at greater than zero altitude, elevation angles may be greater than 90 degrees from the zenith. Such angles are designated by negative values.
C
CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLxl,KTLxTMP)
IF (ITLxQ.GE.1) RVxOPTION(3) = RTLxl
GO TO 20200
20240 IF (IDNINT(RVxOPTION(1)).EQ.0) THEN
IF (RVxOPTION(5).EQ.0.0) RVxOPTION(5) = 1.0D00
GO TO 20200
ENDIF
KVLxPMT = 'Enter elev. resolution (deg) '
RTLxl = RVxOPTION(5)
C
Prompt for the spacing of rays in the elevation direction. This prompt is only displayed if it is appropriate, namely if the dimension of the problem is 1 or 2. Otherwise, a default value of 1 is set and the user is returned directly to the elevation parameters menu.
C
CALL KEYBRD(KVLxPMT,2,ITLxQ,IVLxVAR,RTLxl,KTLxTMP)
IF (ITLxQ.GE.1) RVxOPTION(5) = RTLxl
GO TO 20200
20250 IF (IDNINT(RVxOPTION(1)).LT.2) THEN
IF (RVxOPTION(4).EQ.0.0) RVxOPTION(4) = 1.0D00
GO TO 20200
ENDIF
KVLxPMT = 'Enter azimuth resolution (deg) '
RTLxl = RVxOPTION(4)
C
Prompt for the azimuth spacing of the rays. This prompt is only displayed if it is appropriate.
Line#  Source Line
974  C  namely if the dimension of the problem is 2.
975  C  Otherwise, a default value of 1 is set and the user
976  C  is returned directly to the elevation parameters
977  C  menu.
978  C
979  CALL KEYBRD(KVLxPMT, 2, ITLxQ, IVLxVAR, RTLx1, KTLxTMP)
980  IF (ITLxQ.GE.1) RVxOPTION(4) = RTLx1
981  GO TO 20200
982  20260 IF (IDNINT(RVxOPTION(1)).EQ.0) THEN
983   IF (RVxOPTION(7).EQ.0.0) RVxOPTION(7) = RVxOPTION(3)
984   GO TO 20200
985  ENDIF
986  KVLxPMT = 'Enter elev. limit (deg)'
987  RTLx1 = RVxOPTION(7)
988  C
989  C  Prompt to establish the other end of the range of
990  C  elevation values to be stepped through, the range
991  C  beginning with the starting elevation from above.
992  C  This prompt is only displayed if appropriate, namely
993  C  if there are to be multiple rays. Otherwise, the
994  C  limit is set to be equal to the starting elevation and
995  C  the user is returned directly to the elevation
996  C  parameters menu.
997  C
998  CALL KEYBRD(KVLxPMT, 2, ITLxQ, IVLxVAR, RTLx1, KTLxTMP)
999  IF (ITLxQ.GE.1) RVxOPTION(7) = RTLx1
1000  GO TO 20200
1001  20270 IF (IDNINT(RVxOPTION(1)).LT.2) THEN
1002   IF (RVxOPTION(6).EQ.0.0) RVxOPTION(6) = RVxOPTION(2)
1003   GO TO 20200
1004  ENDIF
1005  KVLxPMT = 'Enter azimuthal limit (deg)'
1006  RTLx1 = RVxOPTION(6)
1007  C
1008  C  Prompt for the other end of the range of azimuth values
1009  C  which will be stepped through, beginning with the
1010  C  starting azimuth value. This prompt is only displayed
1011  C  when appropriate, i.e. when there the dimension of the
1012  C  problem is 2. Otherwise, the limit is set to be equal
1013  C  to the starting azimuth and the user is returned
1014  C  directly to the elevation parameters menu.
1015  C
1016  CALL KEYBRD(KVLxPMT, 2, ITLxQ, IVLxVAR, RTLx1, KTLxTMP)
1017  IF (ITLxQ.GE.1) RVxOPTION(6) = RTLx1
1018  GO TO 20200
1019  20400 RETURN
1020  C
1021  C  Line 20400 is selected when the decides that all necessary
1022  C  values have been entered and that it is time to start
1023  C  sending out the rays.
This sets the EXIT flag to 1, indicating that the user has decided to exit the program.

RETURN

FORMAT(1X,'Enter starting elevation (deg). use neg.')
**DOXDATA Local Symbols**

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVXION6</td>
<td>.</td>
<td>. .</td>
<td>MAINDAT REAL*8</td>
</tr>
</tbody>
</table>


SUBROUTINE IOxMENU(N, CHOICE)

IOxMENU -- SUBROUTINE TO SET UP A MENU BY READING 'TEMPLATES'
FOR THE MENUS FROM FILES. THEN RETURNING THE MENU CHOICE

CALLED BY: DOxDATA
CALLS: IOxPRET

AUTHOR: ERIC L. STROBEL
DATE: 10/09/86
VERSION: 1.0
REVISED: 10/09/86 -- V1.0. Initial revision.

USES: N NUMBER DENOTING WHICH MENU RESOURCE TO USE
TO PRESENT THE USER WITH A MENU AND THEN SEND THE USER'S
CHOICE BACK. (It should be noted that this routine
was written before adoption of the variable naming
convention. Due to the short length of the routine
it has never seemed worth the change.)

RETURNS: CHOICE THE MENU CHOICE THAT WAS MADE

INTEGER N, CHOICE, ITEMS, Q
REAL*8 DUMMY
CHARACTER*10 FILES(10), FILNAM, KDUMMY
CHARACTER*40 TITLE, ITEMLIST(10), PROMPT
CHARACTER*1 CH

Read the menu resource from a file set. The scheme for
maintaining menus goes as follows... The file MASTER.MEN
contains a list of the names of the menu files, in the
order by which they are numbered in DOxDATA. The indivi-
dual menu files have the following format:

- # of items, MENU TITLE
- Text
- of
- each
- item
- (line-by-line, one line per item)
- Prompt text

The existing menu files should be consulted as examples.
IF (CHOICE.LT.1 OR CHOICE.GT.ITEMS) THEN
   PRINT *, 'INVALID CHOICE.'
   GO TO 550
ENDIF

CALL IOXPRET(CH)

IF (CH.EQ.'A') GO TO 550
RETURN

END

IOXMENU  Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHOICE</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TITLE</td>
<td>local</td>
<td>CHAR*40</td>
<td>40</td>
</tr>
<tr>
<td>ITEMS</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>I</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>J</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>CH</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
</tr>
<tr>
<td>DUMMY</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>Q</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>KDUMMY</td>
<td>local</td>
<td>CHAR*10</td>
<td>10</td>
</tr>
<tr>
<td>PROMPT</td>
<td>local</td>
<td>CHAR*40</td>
<td>40</td>
</tr>
<tr>
<td>ITEMLIST</td>
<td>local</td>
<td>CHAR*40</td>
<td>400</td>
</tr>
<tr>
<td>FILES</td>
<td>local</td>
<td>CHAR*10</td>
<td>100</td>
</tr>
<tr>
<td>FILNAM</td>
<td>local</td>
<td>CHAR*10</td>
<td>10</td>
</tr>
</tbody>
</table>
SUBROUTINE IOXPRET(XXX)

IOXPRET -- SUBROUTINE TO PROMPT THE USER TO PRESS RETURN TO CONTINUE. AND THEN CLEAR THE SCREEN

CALLED BY: MAIN. IOXMENU

CALLS: KEYBRD

------------------------------------------------------------------

AUTHOR: ERIC L. STROBEL

DATE: 09/01/87

VERSION: 2.0

REVISED: 10/09/86 -- V1.0. Initial revision.

09/01/87 -- V2.0. Uses KEYBRD and accepts a period '.' as a response (for VAX batching mode).

This routine displays a 'Press RETURN to continue.' prompt. Upon receiving a RETURN, the screen is cleared. After two botched attempts, if the user types something out of the ordinary, the routine gently reminds the user to stick to typing RETURN. IMPORTANT FEATURES: 1) The user may abort by typing 'A' (for abort) in response to the 'Press ...' prompt. This provides a mechanism to allow the user to have a chance to avoid doing something irrevocable if the choice was made by mistake. 2) In an effort to allow the program to be run under VMS in batch mode, a response of '.' & RETURN has the same effect as just typing RETURN. For whatever reason, a VMS batch file cannot contain a blank line.

Please note that this routine was largely written before the adoption of the variable naming convention used throughout most of the rest of the program. This routine is brief enough that it has proven so far unnecessary to change it.

------------------------------------------------------------------

CHARACTER*40 PRMT
1193 CHARACTER*10 CHTMP
1194 CHARACTER*1 XXX
1195 INTEGER YYY, ZZZ, ITMP, ITYPE
1196 REAL*8 DTMP
1197 XXX = ' '
1198 PRMT = ' Press RETURN to continue. '
1199 CALL KEYBRD(PRMT,3,YYY,ITMP,DTMP,XXX)
1200 IF (YYY.LT.1.OR.XXX.EQ.'A'.OR.XXX.EQ.' ') THEN
1201 WRITE(6,300)
1202 300 FORMAT(///////////////////////////////////////////////////////)
1203 RETURN
1204 ELSE
1205 ZZZ = ZZZ + 1
1206 IF (ZZZ.GT.2) PRINT *, ' HIT RETURN ONLY TO CONTINUE.'
1207 GO TO 50
1208 ENDIF
1209 RETURN
1210 END

IOXPRT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXX</td>
<td>param</td>
<td>CHAR*40</td>
<td>40</td>
</tr>
<tr>
<td>PRMT.</td>
<td>local</td>
<td>CHAR*40</td>
<td>4</td>
</tr>
<tr>
<td>YYY</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>ZZZ</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>DTMP.</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>ITMP.</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
</tbody>
</table>
SUBROUTINE KEYBRD(PSTR, ITYP, LN, IVAR, FVAR, CVAR)

KEYBRD -- SUBROUTINE TO TAKE KEYBRD INPUT INTELLIGENTLY AND PARSE IT.
CALLED BY: MAIN, DODATA, IOXPRET

AUTHOR: ERIC L. STROBEL
DATE: 09/01/87
VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: PSTR The prompt string.
      ITYP The type of answer expected to be returned. 1 = INT, 2 = FLOAT, 3 = CHAR.

To present the user with a prompt and to discern what sort of response is expected. This routine attempts to be somewhat intelligent about parsing the typed response. Real values entered without a decimal point are actually read correctly, and strings are converted to all CAPS so that the response typed in appears to be case insensitive. This is a modification of a routine authored by Frank Rhoads. Please note the comment block below referring to where changes need to be made in converting from system to system.

RETURNS: LN The number of characters typed in the keyboard.
         IVAR The integer value to be passed.
         FVAR The floating value to be passed.
         CVAR The character string to be passed.
Display the appropriate prompt, including a default value.

This is also where the changes must be made in translating from system to system. In the three WRITEs below, the formatting ends with ... " ,2H);" and then whatever is the appropriate thing to suppress the return to the beginning of the next line. On the VAX (VMS) this is ";S". With Microsoft Fortran (DOS) it is ";\", and with RyanMcFarland Fortran (DOS) it is nothing at all.

Get the user's response as a character string.

Find the length of the response.

No response, so Return key must have been typed.

IF (LN.EQ.0) RETURN
CVAR=CVAR
IF(ITYP.LT.3) THEN

If the expected answer is a numeric type, then read it from the internal string.

IF(ITYP.EQ.1) THEN
  WRITE(IFMT,'(2H(I,12,1H))') LN
  READ(CVR,IFMT,ERR=12) IVAR
ELSE
  WRITE(IFMT,'(2H(F,12.3H.0))') LN
  READ(CVR,IFMT,ERR=12) FVAR
ENDIF

ELSE

For a character string answer, convert to all caps.

DO 21 ICI=1.LN
  K=ICHAR(BYTS(ICI))
  IF(K.GE.96) THEN
    K=K-32
    CVAR(ICI:ICI)=CHAR(K)
  ELSE
    CVAR(ICI:ICI)=BYTS(ICI)
  ENDIF
21 CONTINUE
ENDIF

END

WRITE(6,'((7H Conversion ERROR, Please RETRY Input)')
GO TO 9
RETURN
Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOXDATA</td>
<td></td>
<td>FSUBRT</td>
<td></td>
</tr>
<tr>
<td>IOXMENU</td>
<td></td>
<td>FSUBRT</td>
<td></td>
</tr>
<tr>
<td>IOXPRET</td>
<td></td>
<td>FSUBRT</td>
<td></td>
</tr>
<tr>
<td>KEYBRD</td>
<td></td>
<td>FSUBRT</td>
<td></td>
</tr>
<tr>
<td>MAINDAT</td>
<td></td>
<td>common</td>
<td>86596</td>
</tr>
<tr>
<td>RAYSUB</td>
<td></td>
<td>extern</td>
<td></td>
</tr>
<tr>
<td>RESULTS</td>
<td></td>
<td>common</td>
<td>1320</td>
</tr>
<tr>
<td>main</td>
<td></td>
<td>FSUBRT</td>
<td></td>
</tr>
</tbody>
</table>

Code size = 56db \((22235)\)
Data size = 3a16 \((2582)\)
SUBROUTINE RAYSUB(RVXAZI, RVXELEV)

RAYSUB -- 3D RAYTRACING WITH ACCELERATED FREE SPACE
PROPAGATION CALCULATION

CALLED BY: MAIN

CALLS: Almost everything else. as this is really
the core of the program.

-----------------------------

AUTHORS: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 03/18/38

VERSION: 4.3

-----------------------------

REVISED: 07/25/86 -- Initial revision. Translated
from Tektronix Basic to VAX Fortran by
Eric L. Strobel.

07/30/86 -- V1.1. Change over to use of
REAL*8 precision in the calculations.

08/12/86 -- V2.0. Add the capability to
perform propagation loss calculations.
Also greatly massaged over the output.
A number of minor and MAJOR bugs corrected.

09/05/86 -- V2.3. Add horizon focusing and
handle low angle rays. Record data for
proper handling of skip focusing. Record
maximum height. Calculate deviation from
great circle path. Minor bug fixes.

09/17/86 -- V2.4. Fix calculation of great
circle path deviation. Introduce cutoff in
number of bounces. Automagically obtain
approximate bottom of the ionosphere. elim-
inating the need for the prompt. Also fixed
rotation matrix used in calculating the new
c-values.

10/10/86 -- V3.0. Former program, now denoted
to a subroutine. Appropriate changes made.
Duties performed by subroutine GEC now done elsewhere. Portion of IONOPAR that searches for triangles of points now streamlined so that the search won't outlast the Earth!

09/01/87 -- V4.0. The essentials have been listed in the comment block at the beginning of MAIN (RAYTRA4.FOR), so you can read about the changes there.

09/30/87 -- V4.1. Phase path calcs. added.

01/15/88 -- V4.2. The (incorrect) derivative values are no longer reported. One of the rays in the bundle has been deleted, for speed's sake. The range cutoff has been tweaked so that it will occur nearly exactly, not just at the end of the next increment. Some of the other changes have been discussed in the comment block at the beginning of MAIN (RAYTRA4.FOR).

03/18/88 -- V4.3. Corrections made for angular range. (see comment block at beginning of MAIN) In the main body of this routine, the azimuth is the running value from the launch point to the previous raypoint. The C's are now normalized at every step, which should enhance the accuracy. Some other minor tweaking.

USES: RVxAZI Ray launch azimuth
RVxELEV Ray launch elevation
/MAINDAT/ A common block the ionospheric and user data that was read in the previous routines.

To perform 3-D raytracing of radio propagation through the ionosphere. Incorporates a specific model for true height profiles of electron density. Does propagation loss due to focusing and to reflection from the ground. Doesn't (yet) incorporate magnetic field effects, or deviative absorption effects.

RETURNS: RVxBOUN The array of results at the earth impact and ray end points.
INTEGER IVxPARAM(6), IVxLAUN(3)
INTEGER IFxBUN, IVxSNUM, IVxTIME(5), IVxNBOUT, IFxGRND
INTEGER IVxSCRTING, IVxSCRTI2, IVxSCRTI3
INTEGER IVxMSCP, IVxJ, IVxSX, IVxSY, IVxSZ, IFxN4, IFxGEN
INTEGER IVxBL, IFxSFL, IFxK, IFxEND
INTEGER IFxPPF, IFxCASE, IFxCCAL

LOGICAL LVxRCUT, LVxHCUT, LVxEND

REAL*8 RVxGRID(6), RVxIONPT(30,60,6), RVxLAUN(7)
REAL*8 RVxOPTION(7), RVxLA, RVxLO, RVxDD6
REAL*8 RVxBNDL(4,3,11), RVxBOUN(11,15), RVxELEV0
REAL*8 RVxAZIO, RVxYEAR, RVxJAI00, RVxTIM
REAL*8 RVxPI, RVxDTOR, RVxREARTH, RVxLATSC, RVxLONSC, RVxLONSC, RVxLONSC
REAL*8 RVxFNSQ(1800,3), RVxH(1800,3), RVxSEZGEC(3,3)
REAL*8 RVxJ, RVxSX, RVxSY, RVxSZ, IFxN4, IFxEND
REAL*8 RVxNBL, RVxSFL, IFxK, IFxEND
REAL*8 RVxIPF, RVxCASE, IFxCCAL

C
101
C
102
C
103
C
104
C
105
C
106
C
107
C
108
C
109
C
110
C
111
C
112
C
113
C
114
C
115
C
116
C
117
C
118
C
119
C
120
C
121
C
122
C
123
C
124
C
125
C
126
C
127
C
128
C
129
C
130
C
131
C
132
C
133
C
134
C
135
C
136
C
137
C
138
C
139
C
140
C
141
C
142
C
143
C
144
C
145
C
146
C
147
C
148
C
149
C
150
C

NOTE: IVxTIME(1) = YEAR: IVxTIME(2) = MONTH
IVxTIME(3) = DAY; IVxTIME(4) = HOUR
IVxTIME(5) = MINUTE

RVxBNDL = RVxBNDL( RAY 1,2,3,OR 4; 1-LAT, 2-LON, 3-ALT:
# OF BOUNCES)

RVxBOUN = RVxBOUN(N-TH BOUNCE; PARAMS).
PARAMS--
  1 -> LAT
  2 -> LON
  3 -> GROUP PATH LENGTH
  4 -> ANGULAR RANGE

---------------------------------------------------------------------
LINE 151 C 5 -> DB DOWN
LINE 152 C 6 -> DEVIATION FROM GR. CIRCLE PATH
LINE 153 C 7 -> UNUSED
LINE 154 C 8 -> UNUSED
LINE 155 C 9 -> HEIGHT OF THE DATA POINT
LINE 156 C 10 -> PHASE PATH LENGTH
LINE 157 C 11 -> X DIRECTION COSINE OF RAY
LINE 158 C 12 -> Y DIRECTION COSINE OF RAY
LINE 159 C 13 -> Z DIRECTION COSINE OF RAY
LINE 160 C 14,15 -> RESERVED FOR FUTURE USE

LINE 163 C
LINE 164 COMMON /MAIN/DAT, RVxGRID, RVxIONPT, IVxPARAM, IVxLAUN
LINE 165 #.RVxLAUN, RVxOPTION
LINE 166 COMMON /RESULTS/RVxBOUN
LINE 167 COMMON /LPARM/ IVxSSNUM, IVxTIME, RVxYEAR, RVxA100, IFxGRND
LINE 168 *.RVxTIM
LINE 169 COMMON /LOSSES/ RVxLOSES, RVxLOS, RVxLSX, RVxLSG
LINE 170 COMMON /PRAM/ RPxPI, RPxDTOR, RPxREARTH, RPxHTOP
LINE 171 COMMON /SCPS1/ ICxNSCP, RVxLATSC, RVxLONSC
LINE 172 COMMON /SCPS1A/ RVxFSNSQ, RVxH
LINE 173 COMMON /OTHER/ RVxLAT1, RVxLON1, RVxHBO, RVxFSQ, RVxRPI, RVxHCUT
LINE 174 COMMON /MISC/ RVxSEZGEC, RVxX7, RVxY7, RVxZ7, RVxR1, RVxR2, RVxHMIN
LINE 175 COMMON /START/ RVxXI, RVxYI, RVxZI, RVxLATI, RVxLONI
LINE 176 COMMON /END/ RVxF, RVxYF, RVxFZ, RVxH5
LINE 177 COMMON /IONO1/ RVxALPHA, RVxBETA, RVxETAL, RVxETA2, RVxFSNB, RVxHB
LINE 178 COMMON /IONO2/ RVxA5, RVxA6, RVxB5, RVxB6, RVxE5, RVxE6
LINE 179 COMMON /IONO3/ RVxV1, RVxV2, RVxxXX, IFxCASE, RVxHBND
LINE 180 COMMON /MORE/ IVxSX, IVxSY, IVxSZ, RVxCX, RVxCY, RVxCZ
LINE 181 COMMON /TEMP1/ RVxF40, RVxF65, RVxKL, RVxHL
LINE 182 COMMON /GORP/ RVxDD6, RVxANGLIM
LINE 183 C
LINE 184 C Initialize things
LINE 185 C
LINE 186 DO 9000 I=1,5
LINE 187 9000 IVxTIME(I) = IVxPARAM(I+1)
LINE 188 C
LINE 189 RPxPI = 3.141592654D00
LINE 190 RPxDTOR = 0.0174532925D00
LINE 191 RPxREARTH = 6.371.2D00
LINE 192 RPxHTOP = 2000.0D00
LINE 193 RVxHMIN = RVxLAUN(7)
LINE 194 ITxK = 0
LINE 195 C Initialize some things that are specific to the RADAR-C model.
LINE 196 C
LINE 197 RVxHBOT = 40.0D00
LINE 198 RVxF40 = 4.03D-04
Line# Source Line
201     RVxK1 = 0.12D00
202     RVxF65 = RVxF40*DEXP(25.0D00 * RVxK1)
203     RVxHL = 120.48384357
204 C
205 C Begin input section, i.e. break out the arrays that will
206 C be needed from the large array.
207 C
208 10000  ICxNSCP = IDNINT(RVxGRID(5)*RVxGRID(6))
209 DO 10200 I=1, RVxGRID(5)
210     RVxLA = RVxGRID(3) + (I-1)*RVxGRID(1)
211     DO 10100 J = 1, RVxGRID(6)
212     RVxLO = RVxGRID(4) + (J-1)*RVxGRID(2)
213     ITxK = ITxK + 1
214     RVxLATSC(ITxK) = RVxLA
215     RVxLONSC(ITxK) = RVxLO
216     RVxFNSQ(ITxK,1) = RVxIONPT(I,J,1)
217     RVxFNSQ(ITxK,2) = RVxIONPT(I,J,3)
218     RVxFNSQ(ITxK,3) = RVxIONPT(I,J,6)
219     RVxH(ITxK,1) = RVxIONPT(I,J,2)
220     RVxH(ITxK,2) = RVxIONPT(I,J,5)
221     RVxH(ITxK,3) = RVxIONPT(I,J,4)
222     RVxLATSC(ITxK) = RVxLATSC(ITxK) * RPxDTOR
223     RVxLONSC(ITxK) = RVxLONSC(ITxK) * RPxDTOR
224 10100  CONTINUE
225 10200  CONTINUE
226 C
227 C More initializing.
228 C
229     RVxLAUN = RVxLAUN(1)
230     RVxLONU = RVxLAUN(2)
231     RVxLAT0 = RVxLAT0 * RPxDTOR
232     RVxLONO = RVxLONO * RPxDTOR
233     RVxRALIM = RVxLAUN(3)
234     RVxANGLM = RVxRALIM/RPxREARTH
235     RPxS4 = 0.0D00
236     RVxSSNUM = RVxPARAM(1)
237     IFxGRND = RVxLAUN(3)
238     IFxPPF = 1
239     RVxNBL = RVxLAUN(1)
240     IF (RVxNBL.GT.10) RVxNBL = 10
241     IF (RVxNBL.LT.0) RVxNBL = 0
242     IFxSFL = RVxLAUN(2)
243     RVxFREQ = RVxLAUN(5)
244     RVxFSQU = RVxFREQ + RVxFREQ
245     RVxRPI = RVxLAUN(4)
246     RVxHCUT = RVxLAUN(7)
247     IF (RVxELEV.EQ.0.0D00) RVxELEV = 0.01D00
248 C
249 C BEGIN CALCULATIONAL SECTION
250 C
Determine the elevation & azimuth for each ray in the bundle.

Still more initializing.

Do the initial coordinate setup and transformations.
Get the initial values for the ionospheric parameters and (if any) gradients.

CALL IONOPAR(IVXJ, IFXN4, IFXGEN, IVXSCEING, IVXSCTR1I
*, IVXSCTR12, IVXSCTR13)
RVXMU2 = 1.0D00 - RVXX/RVXFQU
IF (RVXMU2.LT.0.0D00) THEN
  PRINT *, ' RAY DOES NOT PROPAGATE!!!!'
STOP
ENDIF

RVXCOSEL = DCOS(RVXELEV)
RVXSINEL = DSIN(RVXELEV)
RVXMU = DSQRT(RVXMU2)
RVXCX = -RVXMU * RVXCOSEL * DCOS(RVXAZI)
IVXSX = INTSIGN(RVXCX)
RVXCY = RVXMU * RVXCOSEL * DSIN(RVXAZI)
IVXSY = INTSIGN(RVXCY)
RVXCZ = RVXMU * RVXSINEL
IVXSZ = INTSIGN(RVXCZ)
RVxPFL = 0.0D00
RVXGPLTOT = 0.0D00
IVXIII = 0
LVXEND = .FALSE.
LVXHCUT = .FALSE.
LVXRCUT = .FALSE.

Begin the ray increment by estimating the z increment for the upcoming raypath increment.

20106 RVXDDSZ1 = IVXSZ * DSQRT(RVXCZ / (RVXCX + RVXCY + RVXZZ))
IF (RVXALPHA.NE.0.0D00.AND.RVXBETA.NE.0.0D00) THEN
  RTXDZCALC = (RVXDDSZ1 - RVXSINEL) / RVXDD6
ELSE
  RTXDZCALC = 0.0D00
ENDIF
RVXDELZ = IVXSZ * RVXRPI * (RVXDDSZ1 + RTXDZCALC*RVXRPI.C.0D00)
RVXDELZ = DABS(RVXDELZ)
This section is here to ensure that the calculation cuts off in range almost exactly when it is supposed to.
RVxDEL = DSQRT(RVxRPI*RVxRPI - RVxDELZ*RVxDELZ)
RVxRTMP = (RTxS4 + RVxDELZ/(RVxHO+RPxREARTH)) * RPxREARTH
RVxRMAX = RVxANGLIM * RPxREARTH
IF (RVxRTMP.GE.RVxRMAX) THEN
    RVxDEL2 = RVxDEL - ((RVxRTMP-RVxRMAX) * (RVxHO+RPxREARTH)) #/RPxREARTH
ENDIF
RVxDELZ = RVxDELZ * RVxDEL2 / RVxDEL

The angle of ray incidence at H = 100 km. This is used in the absorption loss calculation.

IF (RVxHZ - 100.0D00).LT.(RVxRPI/2.0D00) RVxA100 =
#DCOS((RVxDELZ/RVxRPI)

RVxDELZ = DMAX1(RVxDELZ, 1.0D-04)
IFxGEN = 1
RVxHZ = RVxHO + IVxSZ*RVxDELZ
Check to see if a boundary has been crossed.

IF (IVxSZ.LT.0) THEN
    IF (RVxHZ.LT.RVxHBND) THEN
        RVxDELZ = RVxHO - RVxHBND
    IFxGEN = 2
ENDIF
GO TO 20133
ENDIF
IF (RVxHZ.GT.RVxHBND) THEN
    RVxDELZ = RVxHBND - RVxHO
    IFxGEN = 2
ENDIF

20133 RVxHZ = RVxHO + IVxSZ*RVxDELZ
Use the z increment and height to calculate where the ray ends up at.

CALL ENDP(T(IFxGEN, RVxHO, RVxHZ, RTxS4, RVxGPLINC)
RVxHZ = RVxHO + RVxZF
RTxRT1 = RPxREARTH + RVxHZ
RVxH5 = DSQRT(RVxFXF*RVxFXF + RVxYF*RVxYF + RTxRT1*RTxRT1)
#- RPxREARTH
IF (RVxH5.LT.0.0D00) RVxH5 = 0.0D00
IF (RVxH5.LT.RVxHMIN) RVxHMIN = RVxH5
If desired, calculate the phase path increment. The value of IFxPPF is currently 'hard-wired' to be 1.

IF (IFxPPF.EQ.1) CALL PHSP(L(IFxGEN, RVxPPI)
IVXIII = IVXIII + 1
IFXCCAL = 0
IF (IFXGEN .NE. 6 .OR. RVXH5 .GT. RVXHBOI) GO TO 20150
C
Will be FALSE only if the ray is below the bottom of the ionosphere & headed downward.
C
RVXZI = RVXHZ
RVXXI = RVXXF
RVXI = RVXF
CALL LATLON3
CALL NEWCS(IFXGEN)
IFXCCAL = 1
C
SMALL SECTION TO HANDLE EARTH BOUNCE
C
CALL ANRANG(RVXAZI,RVXLATI,RVXLONI,RVXLAT0,RVXLONO,RTXS4)
IVXNBOU = IVXNBOU + 1
C
Check to see if the cutoff in the number of bounces is met.
C
IF (IVXNBOU.GT.IVXNBL) THEN
   IVXNBOU = IVXNBOU - 1
   LVXEND = .TRUE.
   GO TO 20150
ENDIF
C
-----------------------------------------------------------------
C If this is the primary ray in the bundle, then all of the information about the bounce point needs to be recorded. otherwise only the parts that will be needed later are stored.
-----------------------------------------------------------------
IF (IFXBN. EQ. 0) THEN
RVXBOUN(IVXNBOU,1) = RVXLATI
RVXBOUN(IVXNBOU,2) = RVXLONI
RVXBOUN(IVXNBOU,3) = RVXGPLTOT + RVXGPLINC
RVXBOUN(IVXNBOU,4) = RTXS4
IF (IFXSF. NE. 1) THEN
   TIMES is a routine which calculates some quantities for the (currently commented out) absorption loss calculation.

   CALL TIMES(RVXBOUN, IVXNBOU, RVXLONO)
   CALL LOSS(RVXBOUN, RVXBNDL, IVXNBOU, RVXLAT0, RVXLONO,
   #RVXELEV0, IFXEND)
ENDIF
C
RVXBOUN(IVXNBOU,5) = RVXLOSR + RVXLOS G
CALL GCDEV(IVXNBOU, RVXAZI0, RVXLAT0, RVXLONO, RVXBOUN, RVXDEV)
C
RVxBOUN(IVxNB0U.6) = RVxDEV
RVxBOUN(IVxNB0U.9) = RVxH5
IF (IFxPPF.EQ.1) RVxBOUN(IVxNB0U.10) = RVxPPL + RVxPPI
RVxCSUM = RVxCX + RVxCY + RVxCZ
RVxBOUN(IVxNB0U.11) = IVxSX*DSQRT(RVxCX)/DSQRT(RVxCSUM)
RVxBOUN(IVxNB0U.12) = IVxSY*DSQRT(RVxCY)/DSQRT(RVxCSUM)
RVxBOUN(IVxNB0U.13) = IVxSZ*DSQRT(RVxCZ)/DSQRT(RVxCSUM)
ELSE
RVxBNDL(IFxBUN.1,IVxNBCU) = RVxLATI
RVxBNDL(IFxBUN.2,IVxNBCU) = RVxLONI
RVxBNDL(IFxBUN.3,IVxNBCU) = RVxH5
ENDIF

END IF

END EARTH BOUNCE HANDLING

Accumulate the group and phase path lengths

20150 RVxGPLTOT = RVxGPLTOT + RVxGPLINC
IF (IFxPPF.EQ.1) RVxPPL = RVxPPL + RVxPPI

TAKE CARE OF BOUNDARY CROSSING

First check for the existence of a boundary crossing.

IF (IFxGEN.NE.2) GO TO 30000

If this is a spherically symmetric case, there are no

tilts, so go on.

IF (IFxN4.EQ.1) GO TO 30000

Call the routine that will actually determine the spatial
tilt of the boundary.

CALL TILTS(RVxDDXHB, RVxDDYHB, IVxJ)

Now, we attempt to correct this particular raypath increment

to insure that it ends exactly ON the boundary, and doesn't
overshoot it.

RVxUDXDS = IVxSX * DSQRT(RVxCX - RVxETA1*RVxFF)
RVxUDYDS = IVxSY * DSQRT(RVxCY - RVxETA2*RVxFF)
RVxUDZDS = IVxSZ * DSQRT(RVxCZ - RVxALPHA*RVxFF + RVxBETA*RVxFF*

*RVxFF)
RVxDDZ = RVxDDDS / RVxDDDS
RVxDDYDZ = RVxDDYD / RVxDDDS
RVxDDGRAD = RVxDDXHB
RVxDDGRAD = RVxDDYHB
RVxDDC5OMAT(1, 1) = 1.0D00 - RVxDDZ * RVxDDGRAD
RVxDDC5OMAT(1, 2) = -RVxDDZ * RVxDDGRAD
RVxDDC5OMAT(2, 1) = -RVxDDYDZ * RVxDDGRAD
RVxDDC5OMAT(2, 2) = 1.0D00 - RVxDDYDZ * RVxDDGRAD
RVxDDC5OMAT(1, 1) = RVxDDC5OMAT(1, 1) * RVxDDC5OMAT(2, 2) - RVxDDC5OMAT(1, 2) * RVxDDC5OMAT(2, 1)
RTxP1 = (RVxDDF * RVxDDC5OMAT(2, 2) - RVxDDYD * RVxDDC5OMAT(1, 2)) / RVxDDC5OMAT(1, 1)
RTxP2 = (RVxDDC5OMAT(1, 1) * RVxDDYD - RVxDDC5OMAT(2, 1) * RVxDDF) / RVxDDC5OMAT(1, 1)
RTxP3 = RVxDDF + RVxDDGRAD * RTxP1 + RVxDDGRAD * RTxP2
RVxDDF = RTxP1
RVxDDYD = RTxP2

NOW, correct the group and phase path length totals. 
RVxDDGPLTOT = RVxDDGPLTOT + (RTxP3 - RVxDDF) / RVxDDDS 
IF (IFxDDPF.EQ.1) THEN
RVxDDPPL = RVxDDPPL + (RVxDDDS * RVxDDDS + RVxDDYD * RVxDDYD)
ENDIF
RVxDDF = RTxP3
RVxDDH = RVxDDH + RVxDDF
RTxRT1 = RPxREARTH + RVxDDH
RVxDDH = DSQRT(RVxDDF * RVxDDF + RVxDDYD * RVxDDYD + RTxRT1 * RTxRT1)
IF (RVxDDH.LT.0.0D00)RVxDDH = 0.0D00
BEGIN UPDATE FOR NEXT ITERATION.
RVxDDI = RVxDDF
RVxDDY = RVxDDF
RVxDDZI = RVxDDH
CALL LALTN3
LVxDDCUT = (RVxDDH.GE.RVxDDCUT.AND.IVxSZ.GT.0)
IF (IFxDDCAL.EQ.0.OR.LVxDDCUT) CALL NEWCS(IFxDDGEN)
If it is the end of the problem, there's a bunch of stuff that doesn't need to be done, so skip it.
IF (LVxDDCUT.OR.LVxDDEND) GO TO 30300
CALL IONOPAR(IVxDD, IFxDDN4, IFxDDGEN, IVxDDCUT, IVxDDSING, IVxDDSTRI1
*, IVxDDSSTRI2, IVxDDSTRI3)
Renormalize the c-values in order to reduce the errors that might otherwise propagate through the program.
RVxCSUM = RVxCX + RVxCY + RVxCZ
RVxMU2 = 1.0000 - RVxx/RVxFSQ
RVxCX = RVxCX * RVxMU2 / RVxCSUM
RVxCY = RVxCY * RVxMU2 / RVxCSUM
RVxCZ = RVxCZ * RVxMU2 / RVxCSUM

RVxH0 = RVxH5
RVxLAT1 = RVxLATI
RVxLON1 = RVxLONI
RVxSINEL = RVxDDS1
CALL ROTSEZ

CALL ANRANG(RVxAZI, RVxLATI, RVxLAT0, RVxLONO, RTxS4)

Evaluate the problem cutoff criteria.

LVxRCUT = (RTxS4.GE.RVxANGLIM)
LVxHCUT = (RVxH5.GE.RVxHCUT.AND.IVxSZ.GT.0)
WRITE(70,30301) RTxS4,RPxREARTH, RVxH5
C30301 FORMAT(1X,G18.8,G18.8)

Check for whether the ray has reached a cutoff condition.

IF (LVxRCUT.OR.LVxHCUT.OR.LVxEND) THEN

A cutoff criterion for this problem has been met. If it is not the primary ray, record the necessary numbers and re-cycle for the next ray in the bundle, otherwise, record all the information on the endpoint and return to MAIN.

IF (IFxBUN.NE.0) THEN
RVxBNDL(IFxBUN,1,IVxNBOU+1) = RVxLATI
RVxBNDL(IFxBUN,2,IVxNBOU+1) = RVxLONI
RVxBNDL(IFxBUN,3,IVxNBOU+1) = RVxH5
IFxBUN = IFxBUN - 1
RVxHMIN = RVxLAU(7)
GO TO 20050
ENDIF

IFxEND = 1
RVxLAT1 = RVxLATI
RVxLON1 = RVxLONI
CALL ROTSEZ
IVxNBOU = IVxNBOU + 1
RVxBOUN(IVxNBOU,1) = RVxLATI
RVxBOUN(IVxNBOU,2) = RVxLONI
RVxBOUN(IVxNBOU,3) = RVxGPLTOT
RVxBOUN(IVxNBOU,4) = RTxS4
IF (IFxSFL.NE.1) THEN
TIMES is called to calculate some values needed by the (currently commented out) absorption loss calculation.

CALL TIMES(RVxBOUN, IVxNB0U, RVxL0NO)

CALL LOSS(RVxBOUN, RVxnBDL, IVxNB0U, RVxLAT0, RVxL0NO).

"RVxELEV0. IFxEND"

ENDIF

RVxBOUN(IVxNB0U, 5) = RVxLOS0 + RVxLOSR

CALL GCDEV(IVxNB0U, RVxAZI0, RVxLAT0, RVxL0NO, RVxBOUN, RVxDEV);

RVxBOUN(IVxNB0U, 6) = RVxDEV

RVxBOUN(IVxNB0U, 9) = RVxH5

RVxBOUN(IVxNB0U, 10) = RVxPPL

RVxCSUM = RVxCX + RVxCY + RVxCZ

RVxBOUN(IVxNB0U, 11) = IVxSX*DSQRT(RVxCX)/DSQRT(RVxCxCSUM)

RVxBOUN(IVxNB0U, 12) = IVxSY*DSQRT(RVxCY)/DSQRT(RVxCxCSUM)

RVxBOUN(IVxNB0U, 13) = IVxSZ*DSQRT(RVxCZ)/DSQRT(RVxCxCSUM)

DO 30500 I=1, IVxNB0U

RVxBOUN(I, 1) = RVxBOUN(I, 1)/RPxDTOR

RVxBOUN(I, 2) = RVxBOUN(I, 2)/RPxDTOR

RVxBOUN(I, 3) = RVxBOUN(I, 3)/300.0DO0

RVxBOUN(I, 4) = RVxBOUN(I, 4)*RPxREARTH

30500 CONTINUE

RETURN

ENDIF

GO TO 20106

END
**RAYSUB Local Symbols**

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFXPPF</td>
<td>local</td>
<td>INTEGER *4</td>
<td>4</td>
</tr>
<tr>
<td>RVXLAT0</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>LVXEND</td>
<td>local</td>
<td>LOGICAL *4</td>
<td>4</td>
</tr>
<tr>
<td>IVXI</td>
<td>local</td>
<td>INTEGER *4</td>
<td>4</td>
</tr>
<tr>
<td>IVXNB</td>
<td>local</td>
<td>INTEGER *4</td>
<td>4</td>
</tr>
<tr>
<td>RVXMU2</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXAZI0</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>ATVRTI</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RTXAINC</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXDEL1</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXLONG</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXGLTOT</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXLO</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXELEV0</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXDET</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXDEV</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXBNDL</td>
<td>local</td>
<td>REAL *8</td>
<td>1056</td>
</tr>
<tr>
<td>RVXMU</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXHZ</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXDZS21</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXDELR</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXPP</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>IFXN4</td>
<td>local</td>
<td>INTEGER *4</td>
<td>4</td>
</tr>
<tr>
<td>RVXDDXB</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXDDYHB</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>IVXNB</td>
<td>local</td>
<td>INTEGER *4</td>
<td>4</td>
</tr>
<tr>
<td>RVXPP</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>IVXSC1</td>
<td>local</td>
<td>INTEGER *4</td>
<td>4</td>
</tr>
<tr>
<td>RVXFR</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>LVXHCUT</td>
<td>local</td>
<td>LOGICAL *4</td>
<td>4</td>
</tr>
<tr>
<td>IVXSC2</td>
<td>local</td>
<td>INTEGER *4</td>
<td>4</td>
</tr>
<tr>
<td>RVXDELS</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RTXVIC</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>IVXSC13</td>
<td>local</td>
<td>INTEGER *4</td>
<td>4</td>
</tr>
<tr>
<td>RTXTEM</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXRAD</td>
<td>local</td>
<td>REAL *8</td>
<td>3</td>
</tr>
<tr>
<td>RVXCE</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXGRAD</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVYGRAD</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXMAX</td>
<td>local</td>
<td>REAL *8</td>
<td>3</td>
</tr>
<tr>
<td>RVXCS</td>
<td>local</td>
<td>REAL *8</td>
<td>8</td>
</tr>
<tr>
<td>RVXH</td>
<td>local</td>
<td>REAL *8</td>
<td>3</td>
</tr>
<tr>
<td>LVXRCUT</td>
<td>local</td>
<td>LOGICAL *4</td>
<td>4</td>
</tr>
<tr>
<td>RVXDXZ</td>
<td>local</td>
<td>REAL *8</td>
<td>3</td>
</tr>
<tr>
<td>RVXCOF</td>
<td>local</td>
<td>REAL *8</td>
<td>32</td>
</tr>
</tbody>
</table>

152
<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFXCCAL</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXSYDZ</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXSINEL</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXGPLINC</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IFXEND</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVXSCSING</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXPI</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXH</td>
<td>SCPS1A</td>
<td>REAL*8</td>
<td>43200</td>
</tr>
<tr>
<td>RVXV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXSEZGEC</td>
<td>MISC</td>
<td>REAL*8</td>
<td>72</td>
</tr>
<tr>
<td>RVXLOS1</td>
<td>LOSSES</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXFSB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXLOS1R</td>
<td>LOSSES</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOSX</td>
<td>LOSSES</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOSG</td>
<td>LOSSES</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHBLT</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHBLM</td>
<td>GORP</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHFL</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXFSU</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHST</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHMIN</td>
<td>MISC</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHPI</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXCX</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXX</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXCY</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXZ</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHND</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IVXPARAM</td>
<td>MAINDAT</td>
<td>INTEGER*4</td>
<td>24</td>
</tr>
<tr>
<td>IVXLATN</td>
<td>MAINDAT</td>
<td>INTEGER*4</td>
<td>12</td>
</tr>
<tr>
<td>RVXH1</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXJ</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IVXSNUM</td>
<td>LPARM</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXF40</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IVXTIME</td>
<td>LPARM</td>
<td>INTEGER*4</td>
<td>20</td>
</tr>
<tr>
<td>RVXH5</td>
<td>END</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXF65</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXK1</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IFXGRND</td>
<td>LPARM</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXHL</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXXF</td>
<td>END</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>ICXNSCP</td>
<td>SCPS1</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXYF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Type</td>
<td>Size</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>JVXSX</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXZF</td>
<td>END</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IVXSY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVXSZ</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IFXCASE</td>
<td></td>
<td>IONO3 INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXGRID</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>48</td>
</tr>
<tr>
<td>RVXALPHA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXIONPT</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>36400</td>
</tr>
<tr>
<td>RVXLAUN</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>56</td>
</tr>
<tr>
<td>RVXBETA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXOPTION</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>56</td>
</tr>
<tr>
<td>RVXETA1</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXETA2</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXLATI</td>
<td>START</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXDD6</td>
<td>GORP</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXLDN1</td>
<td>START</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXLAT1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXBOUN</td>
<td>RESULTS</td>
<td>REAL*8</td>
<td>1320</td>
</tr>
<tr>
<td>RVXLDN1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXX7</td>
<td>MISCT</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVX7</td>
<td>MISCT</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXSEAR</td>
<td>LPARM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXZ7</td>
<td>MISCT</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVX5100</td>
<td>LPARM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXR1</td>
<td>MISCT</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXTIM</td>
<td>LPARM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXR2</td>
<td>MISCT</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVX58</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RFXP1</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVX58</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RFXDTOR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXA5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RFXREARTH</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXLATSC</td>
<td>SCP51</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVX6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXLDN5SC</td>
<td>SCP51</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVX5E</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXE6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXFNSQ</td>
<td>SCP51A</td>
<td>REAL*8</td>
<td>43200</td>
</tr>
</tbody>
</table>
SUBROUTINE ROTSEZ

ROTSEZ -- SUBROUTINE TO ESTABLISH THE TRANSFORMATION
MATRIX FROM S.E.Z. TO G.E.C. COORDINATES

CALLED BY: RAYSUB

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 07/30/86

VERSION: 1.1

REVISED: 07/25/86 -- INITIAL REVISION. TRANSLATED
FROM TEKTRONIX BASIC TO VAX FORTRAN BY
ERIC L. STROBEL.

07/30/86 -- V1.1. Change over to use of
REAL*8 precision in the calculations.

USES:

RVCXLA LAT OF START POINT
RVCXLO LON OF START POINT

TO CALCULATE THE ELEMENTS OF THE ROTATION MATRIX.

RETURNS:

RVCXSEZTX(3,3) THE ROTATION MATRIX

REAL*8 RVCXLA, RVCXLO, RVCXSEZTX(3,3), RTLX1
REAL*8 RTLX2, RTLX3, RTLX4, RTLX5, RTLX6, RTLX7, RTLX8
REAL*8 RTLX9, RTLX10, RTLX11, RTLX12, RTLX13, RTLX14

COMMON /MISC/ RVCXSEZTX, RTLX5, RTLX6, RTLX7, RTLX8, RTLX9, RTLX11
COMMON /OTHER/ RVCXLA, RVCXLO, RTLX11, RTLX12, RTLX13, RTLX14

RTLX1 = DSIN(RVCXLA)
RTLX2 = DCOS(RVCXLA)
RTLX3 = DSIN(RVCXLO)
RTLX4 = DCOS(RVCXLO)
RVCXSEZTX(1,1) = RTLX1 * RTLX4
RVCXSEZTX(1,2) = -RTLX3
RVCXSEZTX(1, 3) = RTLx2 * RTLx4
RVCXSEZTX(2, 1) = RTLx1 * RTLx3
RVCXSEZTX(2, 2) = RTLx4
RVCXSEZTX(2, 3) = RTLx2 * RTLx3
RVCXSEZTX(3, 1) = -RTLx2
RVCXSEZTX(3, 2) = 0.0D00
RVCXSEZTX(3, 3) = RTLx1
RETURN
END

ROTSEZ Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTLX1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX3</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX4</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCTXLA</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCTXLO</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCTXSEZTX</td>
<td>MISC</td>
<td>REAL*8</td>
<td>72</td>
</tr>
<tr>
<td>RTLX5</td>
<td>MISC</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX6</td>
<td>MISC</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX7</td>
<td>MISC</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX8</td>
<td>MISC</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX9</td>
<td>MISC</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX10</td>
<td>MISC</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX11</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX12</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX13</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX14</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>

156
SUBROUTINE LATLON3

LATLON3 -- SUBROUTINE TO OBTAIN LAT, LON OF THE INITIAL
POINT IN THE RAYPATH INCREMENT.

CALLED BY: RAYSUB

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 07/30/86
VERSION: 1.1
REVISED: 07/25/86 -- INITIAL REVISION. TRANSLATED
FROM TEKTRONIX BASIC TO VAX FORTRAN BY
ERIC L. STROBEL.
07/30/86 -- V1.1. Change over to use of
REAL*8 precision in the calculations.

USES: RVC\textsubscript{X}, Y, Z
INITIAL X, Y, Z FOR THE RAYPATH
INCREMENT
RVC\textsubscript{X}MAT(3,3)
SEZ-GE\textsubscript{C} TRANSFORMATION MATRIX
RPC\textsubscript{X}R
EARTH RADIUS

TO CALCULATE THE CORRESPONDING LAT, LON FOR THE GIVEN
POINT.

RETURNS: RVC\textsubscript{X}LA, LO
LAT, LON FOR THE INITIAL
RAYPATH POINT.

REAL*8 RVC\textsubscript{X}1, RVC\textsubscript{X}2, RVC\textsubscript{X}3, RVC\textsubscript{X}4, RVC\textsubscript{X}5, RVC\textsubscript{X}6
REAL*8 RPC\textsubscript{X}PI, RPC\textsubscript{X}DTOR, RPC\textsubscript{X}R
REAL*8 RVC\textsubscript{X}MAT(3,3), RVC\textsubscript{X}X, RVC\textsubscript{XY}, RVC\textsubscript{X}Z
REAL*8 RVC\textsubscript{X}LA, RVC\textsubscript{X}LO, RTL\textsubscript{X}1, RTL\textsubscript{X}2
COMMON /START/ RVC\textsubscript{X}X, RVC\textsubscript{XY}, RVC\textsubscript{X}Z, RVC\textsubscript{X}LA, RVC\textsubscript{X}LO
COMMON /PRAM/ RPC\textsubscript{X}PI, RPC\textsubscript{X}DTOR, RPC\textsubscript{X}R, RTL\textsubscript{X}2
COMMON /MISC/ RVC\textsubscript{X}MAT, RVC\textsubscript{X}1, RVC\textsubscript{X}2, RVC\textsubscript{X}3, RVC\textsubscript{X}4, RVC\textsubscript{X}5, RVC\textsubscript{X}6

157
\begin{verbatim}
739  RTLx1 = RPCxR + RVCxZ
740  RVCx4 = DSQRT(RVCxX*RVCxX + RVCxY*RVCxY + RTLx1*RTLx1)
741  C
742  C
743  C
744  RVCx1 = RVCxMAT(1,1)*RVCxX + RVCxMAT(1,2)*RVCxY + RVCxMAT(1,3) *RTLx1
745  *RTLx1
746  RVCx2 = RVCxMAT(2,1)*RVCxX + RVCxMAT(2,2)*RVCxY + RVCxMAT(2,3) *RTLx1
747  *RTLx1
748  RVCx3 = RVCxMAT(3,1)*RVCxX + RVCxMAT(3,2)*RVCxY + RVCxMAT(3,3) *RTLx1
749  *RTLx1
750  C
751  RVCx5 = DSQRT( RVCx1*RVCx1 + RVCx2*RVCx2 )
752  RVCxLA = DASIN( RVCx3 / RVCx4 )
753  RVCxLO = DASIN( RVCx2 / RVCx5 )
754  IF (RVCx1.GE.0.000) RETURN
755  RVCxLO = -RVCxLO + IINTSIGN(RVCx24)*RPCxP:
756  IF (.INTSIGN(RVCx2) .EQ.O) RVCxLO = RPCxPI
757  RETURN
758  END

LATLON3 Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTLX1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCX1</td>
<td>Misc</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCX2</td>
<td>Misc</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCX3</td>
<td>Misc</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCX4</td>
<td>Misc</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCX5</td>
<td>Misc</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCX6</td>
<td>Misc</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXDTOR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXMAT</td>
<td>Misc</td>
<td>REAL*8</td>
<td>72</td>
</tr>
<tr>
<td>RVCXX</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXY</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXZ</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXLA</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXLO</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLX2</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>

Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANRANG</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
</tbody>
</table>
\end{verbatim}
Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>common</td>
<td>***</td>
<td>32</td>
</tr>
<tr>
<td>ENDP</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>GCDEV</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>GORP</td>
<td>common</td>
<td>***</td>
<td>16</td>
</tr>
<tr>
<td>INTSIGN</td>
<td>extern</td>
<td>**INTEGER*4</td>
<td>***</td>
</tr>
<tr>
<td>IONG1</td>
<td>common</td>
<td>***</td>
<td>30</td>
</tr>
<tr>
<td>IONG2</td>
<td>common</td>
<td>***</td>
<td>144</td>
</tr>
<tr>
<td>IONG3</td>
<td>common</td>
<td>***</td>
<td>36</td>
</tr>
<tr>
<td>IONOPAR</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>LATLON3</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>LOSS</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>LOSSES</td>
<td>common</td>
<td>***</td>
<td>32</td>
</tr>
<tr>
<td>LPARM</td>
<td>common</td>
<td>***</td>
<td>52</td>
</tr>
<tr>
<td>MAINDAT</td>
<td>common</td>
<td>***</td>
<td>86596</td>
</tr>
<tr>
<td>MISC</td>
<td>common</td>
<td>***</td>
<td>120</td>
</tr>
<tr>
<td>MORE</td>
<td>common</td>
<td>***</td>
<td>36</td>
</tr>
<tr>
<td>NEWCS</td>
<td>exter</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>OTHER</td>
<td>common</td>
<td>***</td>
<td>48</td>
</tr>
<tr>
<td>PHSPL</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>PRAM</td>
<td>common</td>
<td>***</td>
<td>32</td>
</tr>
<tr>
<td>RAYSUB</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>RESULTS</td>
<td>common</td>
<td>***</td>
<td>1320</td>
</tr>
<tr>
<td>SCPS1</td>
<td>common</td>
<td>***</td>
<td>28804</td>
</tr>
<tr>
<td>SCPS1A</td>
<td>common</td>
<td>***</td>
<td>86400</td>
</tr>
<tr>
<td>START</td>
<td>common</td>
<td>***</td>
<td>40</td>
</tr>
<tr>
<td>TEMP1</td>
<td>common</td>
<td>***</td>
<td>32</td>
</tr>
<tr>
<td>TILTS</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
</tbody>
</table>

Code size = lde (7630)
Data size = 00bf (191)
SUBROUTINE IONOPAR (IVSXJ, IFSXN4, IFSXG, IVSXSCS, IVSXSCST1, IVSXSCST3)

IONOPAR -- SUBROUTINE TO CALCULATE THE PROPERTIES OF THE IONOSPHERE FOR A GIVEN POSITION.
CALLED BY: RAYSUB
CALLS: TRIANG, INBOX, INTERP, CASE1-5

AUTHORS: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 03/18/88
VERSION: 3.1

REVISED: 07/25/86 -- INITIAL REVISION. TRANSLATED FROM TEKTRONIX BASIC TO VAX FORTRAN BY ERIC L. STROBEL.
07/30/86 -- V1.1. Change over to use of REAL*8 precision in the calculations.
10/10/86 -- V2.0. Altered to fit the new status of the old RAYTRACE program as a subroutine. Important change in sorting out which triangle a pt. is in: With large arrays of points the old algorithm would have taken nearly forever (REALLY!). Since the points are now being defined on a grid, the problem is vastly simpler.
09/01/87 -- V3.0. Now incorporates the RADAR-C ionosphere. The interpolation routine has been modularized. Estimates of the nearest boundary are now reported.
03/18/88 -- V3.1. Some corrections have been made so that the manipulation of grid indices works when the date line is crossed by the grid.
USE: IVSxJ  J-TH IONOSPHERE LAYER
IFSxN4,G  FLAGS
RVcxX,Y,Z7  INTERMEDIATE COORD. VALUES FROM LAT, LON COMPUTATION
RVcxRL  ANOTHER INTERMEDIATE VALUE
RVcxLAI,LOI  INITIAL PT'S. LAT, LON
RVcxH5  NEW HEIGHT
ICcxN  NUMBER OF S.C.P.'S
RPCxRE  RADIUS OF THE EARTH
RVcxFN(1800,3)  X
RVcxLA(1800,3)  X
RVcxLA(1800)  X-DATA FOR THE S.C.P.'S
IVcxSZ  SIGN FOR Z
RVcxFREQ  WAVE FREQ. SQUARED

TO CALCULATE THE VALUES OF THE IONOSPHERIC PARAMETERS
USED TO UPDATE THE RAYPATH.

RETURNS:
RVcxFB(3)  FN^2 VALUES OF THE LAYERS
RVcxHB(3)  H & Y VALUES OF THE LAYERS
RVcxALPH  X
RVcxBETA  X
RVcxET1  X-COEF. VALUES FOR MU^2
RVcxET2  X
RVcxB5&6  ]
RVcxE5&6  ]
RVcxV1  ] SOME INTERMEDIATE VALUES
RVcxV2  ]
IVSxSCS  VBL. SIGNIFYING USE OF SINGLE S.C.P.
IVSxSCT1-3  VBL'S. FOR THE TRIANGLE OF S.C.P.'S USED

-------------------------------------------------------------------

INTEGER IVSxSCS, IVSxSCT1, IVSxSCT2, IVSxSCT3
INTEGER IVcxSCS, IVcxSCT1, IVcxSCT2, IVcxSCT3, IFcxN4
INTEGER IVcxSI, IVcxSY, IVSxJ, IFcxSN
INTEGER IFSxN4, IFSxG, IVcxSZ, ICCxN
INTEGER IFLxOUT, IVLxIN, IVLxFIN, IVcxCASE
REAL*8 RVxGRID(6), RPCxRE, RPCxPI, RPCxDR
REAL*8 R TLxD1, RTLxD2, RTLxD5, RTLx1
REAL*8 RVLxTR1, RVLxTR2, RTLxTH
REAL*8 RTLxH, RTLxI3, RTLxI4, RTLxI5
REAL*8 RVcxA5(3), RVcxA6(3), RVcxLA1, RVcxLOI
REAL*8 RVcxFN(1800,3), RVcxH(1800,3), RVcxLA(1800), RVcxLO(1800)
REAL*8 RVcxALPH, RVcxBETA, RVcxHBND, RVcxF40, RVcxF65
101 REAL*8 RVCxET1, RVCxET2, RVCxFB(3), RVCxHB(3), RVCxB5(3)
102 REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3)
103 REAL*8 RVCxV1, RVCxV2, RVCxH5, RVCxFREQ
104 REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVLxHTV, RVLxSLV
105 REAL*8 RVCxXI, RVCxYI, RVCxZI, RVCxFX, RVCyFX, RVCxZF
106 REAL*8 RVCxE1, RVCxHL, RVCXXX, RVCxXL, RVCxUX, RVCxA0
107 REAL*8 RVCxB1, RVCxHU, RVCxBO, RVCxH2, RVCxY5, RVCxSL1
108 REAL*8 RVCxSL2, RVCxH1L, RVCxA1, RVCxB1, RVCxC1, RVCxH1P
109 REAL*8 RVCxA2, RVCxB2, RVCxG2, RVCxHT3, RVCxHT4, RVCxHT5
110 REAL*8 RVLxLATS, RVLxLONS, RVLxLATE, RVLxLONE, RVLxLOI
111 REAL*8 RVLxF98, RVLxK2, RVCxH2P, RTCxA, RTCxB, RTCxC

C
COMMON /MAINDAT/ RVCxGRID
COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RTCxA
COMMON /SCLS/ ICCxN, RVCxLA, RVCxLO
COMMON /SCLS1/ ICCxNL, RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RTCxB, RTCxC
COMMON /START/ RVCxXI, RVCxYI, RVCxZI, RVCxLA2, RVCxLOI
COMMON /END/ RVCxXI, RVCxYI, RVCxZI, RVCxLA2, RVCxLOI
COMMON /OTHER/ RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RTCxB, RTCxC
COMMON /TEMP1/ RVCxH40, RVCxH10, RVCxH1, RVCxH5
COMMON /TEMP2/ RVCxS0, RVCxS1, RVCxS2, RVCxS3, RVCxS4, RVCxS5
COMMON /TEMP3/ RVLXH, RVLX1, RVLX2, RVLX3, RVLX4, RVLX5
COMMON /VAR1/ RVCxAI, RVCxA1, RVCxAL, RVCxB1, RVCxL1, RVCxH1P, RVCxA0
COMMON /VAR2/ RVCxH1, RVCxH2, RVCxHS, RVCxSL1, RVCxSL2, RVCxH1L
COMMON /VAR3/ RVCxAI, RVCxB1, RVCxAL, RVCxH1P, RVCxA0
COMMON /VAR4/ RVCxA2, RVCxB2, RVCxG2, RVCxHT3, RVCxHT4, RVCxHT5
COMMON /RAID/ ICCxN

10000 RTLXH = RVCxH5
106 RVCxHBND = 0.0000
116 RVCxLOI = RVCxLAI / RPCxDR
118 RVCxLOI = RVCxLAI / RPCxDR
129 RVLxLOI = RVCxLAI
140 IF (ICCxN.GT.1) THEN
141
143 C If the number of points is not 1, then there is a grid. If
144 C calculate the grid boundaries, then call INBOX to determine
145 C whether the present location is w/in the grid. If not in
146 C the grid, then do a spherically symmetric ionosphere based
147 C upon the nearest grid point's parameters. Within the grid
148 C call TRIANG to determine the three points that are to be
149 C used in the interpolation.
Line#    Source Line
151 C
152 RVLxLATS = RVxGRID(3)
153 RVLxLONS = RVxGRID(4)
154 RVLxLATE = RVLxLATS + (RVxGRID(5) - 1.0D00) * RVxGRID(1)
155 RVLxLONE = RVLxLONS + (RVxGRID(6) - 1.0D00) * RVxGRID(2)
156 C
157 C If the grid spans the date line, it's longitude values
158 C will not be negative. While this is good for purposes
159 C of the interpolation, ray points with west longitudes
160 C will spuriously be considered to be outside the grid.
161 C Hence, the need for the temporary copy of the longitude.
162 C If necessary, the next statement will bring the longitude
163 C of the ray point into the same longitude system as the
164 C grid.
165 C
166 IF (RVLxLOI.LT.RVLxLONS) RVLxLOI = RVLxLOI + 360.0
167 C
168 IFLxOUT = -1
169 CALL INBOX(RVCxLAI,RVLxLOI,RVLxLATS,RVLxLONS,RVLxLATE
170 &,RVLxLONE,IFLxOUT)
171 C
172 C IFLxOUT = 1 means that the current location is outside the
173 C ionospheric specification grid, so only the nearest grid
174 C point is used.
175 C
176 IF (IFLxOUT.EQ.1) THEN
177 IFCxN4 = 1
178 IF (RVCxLAI.LT.RVLxLATS) THEN
179 ITLxA = 1
180 ELSE IF (RVCxLAI.GT.RVLxLATE) THEN
181 ITLxA = IDNINT(RVxGRID(5))
182 ELSE
183 ITLxA = IDNINT(((RVCxLAI-RVLxLATS)/RVxGRID(1)) + 1.0)
184 ENDIF
185 IF (RVLxLOI.LT.RVLxLONS) THEN
186 ITLxB = 1
187 ELSE IF (RVLxLOI.GT.RVLxLONE) THEN
188 ITLxB = IDNINT(RVxGRID(6))
189 ELSE
190 ITLxB = IDNINT(((RVLxLOI-RVLxLONS)/RVxGRID(2)) + 1.0)
191 ENDIF
192 IVCxSCS = IDNINT((ITLxA - 1.0D00) * RVxGRID(6) + ITLxB)
193 ELSE
194 CALL TRIANG
195 IFCxN4 = 2
196 ENDIF
197 ELSE
198 C
199 C These are the values to use if only one specification point
200 C is given.
Some initializations.

IVSxSCS = IVCxSCS
IVSxSCT1 = IVCxSCT1
IVSxSCT2 = IVCxSCT2
IVSxSCT3 = IVCxSCT3
IFSxN4 = IFCxN4
RVCxET1 = 0.0D00
RVCxET2 = 0.0D00
RVCxALPH = 0.0D00
RVCxBETA = 0.0D00
RVCXXX = 0.0D00
RVCxLOI = RVCxLOI * RPCxDR
RVLxLOI = RVLxLOI * RPCxDR
RVCxLAI = RVCxLAI * RPCxDR

For calculation purposes here, move slightly off of the boundary, if we're at one.

IF (IFSxG.EQ.2.OR.IFSxG.GE.6) RTLxH = #RTLxH = IVCxSZ*1.0D-02

Now the enumeration of possible height regions begins.
First, handle what is needed for the free-space underlying the ionosphere (below 40 km). Also, the lower part of the D layer is exponential and spherically symmetric, so do that now too, since it involves no interpolation. For full details see the RADAR-C report cited in the documentation.

IF (RTLxH.LE.40.0D00) THEN
  IF (IVCxSZ.GT.0) THEN
    RVCxHBND = 40.0D00
    IVSxJ = 1
  ELSE

164
RVCxHBND = 0.0D00
IVSxJ = 0
ENDIF
RETURN

The lower D region of RADAR-C

IF (RTLxH.LE.65.0D00) THEN
RVCxxXX = RVCxF40 * DEXP(RVCxK1 * (RTLxH - 40.0D00))
RVCxALPH = RVCxK1 * RVCxxXX / RVCxFREQ
RVCxBETA = -RVCxK1 * RVCxALPH / 2.0D00
IF (IVCxSZ.GT.0) THEN
RVCxHBND = 65.0D00
IVSxJ = 2
ELSE
RVCxHBND = 40.0D00
IVSxJ = 1
ENDIF
RETURN
ENDIF

Here is where, if the case is a non-spherically symmetric one, that interpolation begins to be needed.

Note: FB(1) --> f0E ** 2
FB(2) --> f0F1 ** 2
FB(3) --> f0F2 ** 2

HB(1) --> hmF1
HB(2) --> hmF2
HB(3) --> YmF2

The interpolation is done as follows.
X = A + B*Lat + E*Lon, where A, B, and E are determined by setting up the three equations at the three grid points surrounding the point in question. This is done in INTERP.

Recall that IFSxN4 = 1 for the spherically symmetric case.

IF (IFSxN4.EQ.1) THEN
RVCxV1 = RVCxFB(1) = RVCxFN(IVSxSCS,1)
ELSE
RVCxV1 = RPCxRE + RTLxH
RVCxV2 = RVCxV1 * DCOS(RVCxLAI)

165
RTLxI3 = RVCxLA(IVSxSCT1) * RVCxLO(IVSxSCT2) - RVCxLA(IVSxSCT1)
"RVCxLO(IVSxSCT1)
RTLxI4 = RVCxLA(IVSxSCT2) * RVCxLO(IVSxSCT3) - RVCxLA(IVSxSCT2)
"RVCxLO(IVSxSCT2)
RTLxI5 = RVCxLA(IVSxSCT3) * RVCxLO(IVSxSCT1) - RVCxLA(IVSxSCT1)
"RVCxLO(IVSxSCT3)
RTLxD5 = RTLxI3 + RTLxI4 + RTLxI5

IVLxIN and FIN are used to get INTERP to operate only on wna:
is needed to do the calculations for the D & E regions.

IVLxIN = 1
IVLxFIN = 1
CALL INTERP(IVLxIN, IVLxFIN)
RVCxFB(1) = RVCxA5(1) + RVCxB5(1) * RVCxLAI + RVCxES(1) * RVLxLO
ENDIF
IF (RTLxH.LE.98.0D00) THEN

The upper D layer is exponential.

RVLxF98 = 0.64000 * RVCxFB(1)
RVLxK2 = (DLOG(RVLxF98) - DLOG(RVCxF65)) / 33.0D00
RVCxXX = RVCxF65 * DEXP(RVLxK2 * (RTLxH - 65.0D00))
RVCxALPH = RVLxK2 - RVCxXX / RVCxFREQ
RVCxBETA = -RVLxK2 * RVCxALPH / 2.0D00
IF (IVCXSZ.GT.0) THEN
    RVCxHBND = 98.0D00
    IVSxJ = 3
ELSE
    RVCxHBND = 65.0D00
    IVSxJ = 2
ENDIF
IF (IFSxN4.NE.1) THEN
    RTLx1 = ((RTLxH - 65.0D00) * RVCxXX) / (#(33.0D00 * RVCxFB(1) * RVCxFREQ)
    RVCxET1 = -RTLx1 * RVCxB5(1) / RVCxV1
    RVCxET2 = RTLx1 * RVCxE5(1) / RVCxV2
ENDIF
RETURN
ENDIF
IF (RTLxH.LE.RVCxHL) THEN

The E layer is parabolic, with only foE as a non-constant
parameter.

RVLxHTV = (RTLxH - 110.0D00) / 20.0D00
RVLxSLV = RVCxFB(1) / 20.0D00
RVCxXX = RVCxFB(1) * (1.0D00 - RVLxHTV * RVLxHTV)
RVCxALPH = -2.0D00 * RVLxHTV * RVLxSLV / RVCxFREQ
RVCxBETA = RVLxSLV / (20.0D00 * RVCxFREQ)
351 IF (IVCxSZ.GT.0) THEN
352  RVCxHBND = RVCxHL
353  IVSxJ = 4
354 ELSE
355  RVCxHBND = 98.0D00
356  IVSxJ = 3
357 ENDIF
358 IF (IFSxN4.NE.1) THEN
359  RTLx1 = (1.0D00 - RVLxHTV*RVLxHTV) / RVCxFREQ
360  RVCxET1 = -RTLx1 * RVCxB5(1) / RVCxV1
361  RVCxET2 = RTLx1 * RVCxE5(1) / RVCxV2
362 ENDIF
363 RETURN
364 ENDF
365 C
366 C Now. get the rest of the values needed to do the calculations in the F1 & F2 layers.
367 C
368 IF (:FSxN4.EQ.1) THEN
369  RVCxFB(2) = RVCxFN(IVSxSCS,2)
370  RVCxFB(3) = RVCxFN(IVSxSCS,3)
371  RVCxHB(1) = RVCxH(IVSxSCS,1)
372  RVCxHB(2) = RVCxH(IVSxSCS,2)
373  RVCxHB(3) = RVCxH(IVSxSCS,3)
374 ELSE
375  IVLxIN = 2
376  IVLxFIN = 3
377  CALL INTERP(IVLxIN, IVLxFIN)
378 C-------------------------------------------------------------------------
379 C The F1 values may be 0 in this model if the Fl isn't seen in the profile. This botches up the interpolation by introducing an artificially large gradient. This gets fixed by the next snippet of code.
380 C-------------------------------------------------------------------------
381 C
382 IF (RVCxH(IVSxSC1.1).EQ.0.0.OR.RVCxH(IVSxSC2.1).EQ.0.0) THEN
383  RVCxA5(2) = 0.0
384  RVCxB5(2) = 0.0
385  RVCxE5(2) = 0.0
386  RVCxA6(1) = 0.0
387  RVCxB6(1) = 0.0
388  RVCxE6(1) = 0.0
389 ENDIF
390 C
391 RVCxFB(2) = RVCxA5(2) + RVCxB5(2)*RVCxLAI
392 #+ RVCxE5(2)*RVLxLOI
393 RVCxFB(3) = RVCxA5(3) + RVCxB5(3)*RVCxLAI
394 #+ RVCxE5(3)*RVLxLOI
395 167
RVCxHB(1) = RVCxA6(1) + RVCxB6(1)*RVCxLAI
RVCxHB(2) = RVCxA6(2) + RVCxB6(2)*RVCxLAI
RVCxHB(3) = RVCxA6(3) + RVCxB6(3)*RVCxLAI
RVCxHB(2) = RVCxA6(2)
RVCxB6(2)*RVCxLAI
RVCxHB(3) = RVCxA6(3)
RVCxB6(3)*RVCxLAI
RVCxHB = RVCxA6 + RVCxB6*RVCxLAI
ENDIF
C-----------------------------------------------------------------------
C From here on, the logic is complicated, but it follows the
procedures in the RADAR-C report. Basically, it must be
determined whether the F1 is linear or parabolic, and
where the E, valley, F1, F2, and topside profiles fall
and where their intersection points are.
-----------------------------------------------------------------------
RVCXL = 0.8516D00 + 0.3516D00 * RVCxFB(1)
RVCXU = 0.98000 + 0.98000 * RVCxFB(1)
RVCXHU = RVCXHB(2) - RVCXHB(3) * DSQRT(1.0000 -
(#(RVCXHU/RVCxFB(3)))
RVCXHB1 = 0.75D00 + RVCXHB(1)
RVCxAO = (RVCXHU*RVCXL - RVCXHL*RVCXU)/(RVCXHU - RVCXHL)
RVCxB0 = (RVCXHU - RVCXHL) / (RVCXHU - RVCXHL)
RVCXHS = RVCXHB(2) + 0.25D00*RVCXHB(3)
IF (RVCxFB(2).LE.RVCXHU) THEN
IVCXCASE = 1
GO TO 20000
ENDIF
IF (RVCxFB(2).GT.RVCXHU) GO TO 10750
RVCXH2 = RVCXHB(2) - RVCXHB(3) * DSQRT(1.0000 -
(#(RVCXHU/RVCxFB(3)))
IF (RVCXHB(2).LE.RVCXH2) GO TO 10750
RVCXYS = MAX(1.0000, RVCXH2 - RVCXHB1)
RVCXSL1 = RVCxFB(2) / RVCXYS
RVCXSL2 = 2.0000 + RVCxFB(3) * (RVCXHB(2) - RVCXH2)/
#(RVCXHB(3)*RVCXHB(3))
IF (RVCXSL1.GE.RVCXSL2) THEN
RVCXHIL = (RVCXHBI*RVCXSL1 + RVCxAO)/
#(RVCXSL1 - RVCxB0)
IF (RVCXHIL.LT.RVCXHU) THEN
IVCXCASE = 3
ELSE
IVCXCASE = 1
ENDIF
GO TO 20000
ENDIF
10750 RVCxA1 = 16.0000 + RVCxFB(2) / (RVCXHB(1)*RVCXHB(1));
RVCxB1 = (32.0000 + RVCxFB(2) / RVCXHU - RVCxB0;
RVCxC1 = 15.0000 + RVCxFB(2) + RVCxA0
RTLxD1 = RVCxB1*RVCxB1 - 4.0000 * RVCxA1 * RVCxC1
168
451 IF (RTLxD1.LT.0.0000) THEN
452 IVCxCASE = 1
453 GO TO 20000
454 ENDIF
455 RVCxH1P = (RVCxB1 - INTSIGN(RVCxB1) * DSQRT(RTLxD1));
456 #/(2.0D00 * RVCxA1)
457 RVCxA2 = (RVCxFB(3)/(RVCxHB(3)*RVCxHB(3)) -
458 #/(16.0D00 * RVCxFB(2)/(RVCxHB(1)*RVCxHB(1)))
459 RVCxB2 = (32.0D00 * RVCxFB(2)/RVCxHB(1)) -
460 #/(2.0D00 * RVCxFB(3) * RVCxHB(2)/(RVCxHB(3)*RVCxHB(3)))
461 RVCxC2 = RVCxFB(3)*((RVCxHB(2)*RVCxHB(2))/(RVCxHB(3)*
462 #RVCxHB(3)) - 1.0D00) - 15.0D00 = RVCxFB(2)
463 RTLxD2 = RVCxB2 * RVCxB2 - 4.0D00 * RVCxA2 * RVCxC2
464 IF (RTLxD2.LT.0.0000) THEN
465 IVCxCASE = 1
466 GO TO 20000
467 ENDIF
468 RVCxHT3 = (-RVCxB2 + INTSIGN(RVCxB2) * DSQRT(RTLxD2));
469 #/(2.0D00 * RVCxA2)
470 RVCxHT4 = (-RVCxB2 + INTSIGN(RVCxB2) *DSQRT(RTLxD2))/
471 #/(2.0D00 * RVCxA2)
472 IF (RVCxH1P.LE.RVCxHU) THEN
473 RVCxH2P = (RVCxB1 + INTSIGN(RVCxB1) *DSQRT(RTLxD1));
474 #/(2.0D00 * RVCxA1)
475 IF (RVCxH2P.LT.RVCxHU) THEN
476 IVCxCASE = 5
477 GO TO 20000
478 ENDIF
479 IVCxCASE = 2
480 IF (RVCxB2.GT.0.0000) THEN
481 RVLxTR1 = RVCxHT4
482 RVLxTR2 = RVCxHT3
483 IFCxSN = 1
484 ELSE
485 RVLxTR1 = RVCxHT3
486 RVLxTR2 = RVCxHT4
487 IFCxSN = -1
488 ENDIF
489 RTLxTH = RVLxTR1
490 10850 IF (RTLxTH.GT.RVCxHU.AND.RTLxTH.LT.RVCxHTS) THEN
491 RVCxHT4 = RTLxTH
492 GO TO 20000
493 ENDIF
494 IF (RTLxTH.EQ.RVLxTR2) THEN
495 C
496 C
497 C
498 C
499 C
500 PRINT *, 'EXCEEDINGLY WIERD CASE: POINT #1'
500 PRINT *, 'HT3: ',RVCxHT3,' HT4: ',RVCxHT4
PRINT * ', H5 : ' RVCxH5,' B2 : ' RVCxB2
PRINT * ', LAT : ' RVCxLAI/RPCxDR,' LON : ',
RVCxLOI/RPCxDR
PRINT * ', HT5 : ' RVCxHT5,' HU : ' RVCxHU
PRINT * ', HIP : ' RVCxHIP
STOP

ELSE
RTLxTH = RVLxTR2
IFCxSN = -IFCxSN
GO TO 10850
ENDIF

IF (RVCxHT3.LE.RVCxHIP) THEN
IVCxCASE = 1
GO TO 20000
ENDIF

IF (RVCxHT3.LT.RVCxHT4.AND.RVCxHT4.LT.RVCxHT5) THEN
IVCxCASE = 4
ELSE
Another diagnostic as outlined above.
PRINT * ', EXCEEDINGLY WIERD CASE: POINT #3 '
STOP
ENDIF

These routines continue the calculations, now that it has been decided which path to take.

IF (IVCxCASE.EQ.1) THEN
CALL CASE1(RTLxH, IVSxJ)
RETURN
ELSE IF (IVCxCASE.EQ.2) THEN
CALL CASE2(RTLxH, IVSxJ)
RETURN
ELSE IF (IVCxCASE.EQ.3) THEN
CALL CASE3(RTLxH, IVSxJ)
RETURN
ELSE IF (IVCxCASE.EQ.4) THEN
CALL CASE4(RTLxH, IVSxJ)
ELSE
CALL CASE5(RTLxH, IVSxJ)
ENDIF
### IONOPAR Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVSXSCST3</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVSXSCST2</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVSXSCST1</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVSXSCS</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFSXG</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFSXN4</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVSXJ</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVXLXATS</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXD2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXSLV</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>ITLXA</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>ITLXB</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVLX9F98</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXLONS</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXK2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXH</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IVLXIN</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVLXFIN</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXKTR1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXKTR2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXTH</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IFLXOUT</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXLOI</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLATE</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOONE</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXI</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXHTV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXD1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXET2</td>
<td>local</td>
<td>IONO1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXFB</td>
<td>local</td>
<td>IONO1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXHB</td>
<td>local</td>
<td>IONO1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXBS</td>
<td>local</td>
<td>IONO2</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXB6</td>
<td>local</td>
<td>IONO2</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXEB</td>
<td>local</td>
<td>IONO2</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXE6</td>
<td>local</td>
<td>IONO3</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXV1</td>
<td>local</td>
<td>IONO3</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXV2</td>
<td>local</td>
<td>IONO3</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXH5</td>
<td>local</td>
<td>END</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXFREQ</td>
<td>local</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXLD1</td>
<td>local</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXLO1</td>
<td>local</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXHBOT</td>
<td>local</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXXI</td>
<td>local</td>
<td>START</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXYI</td>
<td>local</td>
<td>START</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXZI</td>
<td>local</td>
<td>START</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVXXF</td>
<td>local</td>
<td>END</td>
<td>REAL*8</td>
</tr>
</tbody>
</table>

171
# IONOPAR Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RV沅YF</td>
<td>END</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IV沅SCS</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅ZP</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅K1</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IV沅SC1</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅HL</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IV沅SC2</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅XX</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IV沅SC3</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IF沅N4</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅XL</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅XU</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IV沅SX</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅A0</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IV沅SY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅HB1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IF沅SN</td>
<td>RAID</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IV沅S2</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅HU</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>ICCXN</td>
<td>SCPS1</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅B0</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅H2</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅S5</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅SL1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IV沅CASE</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RV沅SL2</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅GRID</td>
<td>MAINDAT</td>
<td>REAL*8</td>
<td>43</td>
</tr>
<tr>
<td>RV沅H1L</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅A1</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RP沅X2</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅B1</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RP沅X1</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅C1</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RP沅X4</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅H1P</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅A2</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅B2</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>6</td>
</tr>
<tr>
<td>RTLX5</td>
<td>TEMP3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅C2</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅H3</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV沅H4</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RV沅H5</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX6</td>
<td>TEMP3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLX7</td>
<td>TEMP3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX8</td>
<td>TEMP3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>

172
<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVCA5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCLAZ</td>
<td>START</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCL0I</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCH2P</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCFN</td>
<td>SCPS1A</td>
<td>REAL*8</td>
<td>43200</td>
</tr>
<tr>
<td>RTCXA</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCH</td>
<td>SCPS1A</td>
<td>REAL*8</td>
<td>43200</td>
</tr>
<tr>
<td>RTCXB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCLinear</td>
<td>SCPS1</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RTCX</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCL0</td>
<td>SCPS1</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVCLalph</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCLbeta</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCLbound</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCF40</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCF65</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCMET1</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE INTERP(IVSxIN, IVSxFIN)

INTERP -- SUBROUTINE TO PERFORM THE LATITUDE AND LONGITUDE
INTERPOLATION REQUIRED BY IONOPAR

CALLED BY: IONOPAR

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/01/87

VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: IVSxIN & IVSxFIN TO DETERMINE WHICH SETS
OF INTERPOLATION COEFFICIENTS TO DO.

The interpolation is done as follows,

\[ X = A + B \cdot \text{Lat} + E \cdot \text{Lon}, \]

where \( A, B, \) and \( E \) are determined
by setting up the three equations at the three grid
points surrounding the point in question. These
equations are then solved by use of Kramer's rule.

RETURNS: /IONO2/ THE COMMON BLOCK CONTAINING THE
INTERPOLATION COEFFICIENTS \( [A, B, E] \),
WHERE \( A, B, \) AND \( E \) ARE AS ABOVE, AND THE 5
REPRESENTS A FREQUENCY COEFFICIENT AND THE 6
REPRESENTS A HEIGHT COEFFICIENT.

INTEGER IVSxIN, IVSxFIN, IVCxSCT1, IVCxSCT2
INTEGER IVCxSCT3, ICCxN, IVCxSCS, ITLx1

REAL*8 RVCxI3, RVCxI4, RVCxI5, RVCxDS
REAL*8 RVCxFN(1800,3), RVCxH(1800,3), RVCxLA(1800)
REAL*8 RVCxLO(1800), RVCxAS(3), RVCxA6(3), RVCxB5(3)
REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3)
DO 10000 I = IVSXIN, IVSXFIN
   RVCx5(I) = (RVCxFN(IVCXSCST1, I) * RVCxI4 + RVCxFN(IVCXSCST2, I) * RVCxI5 + RVCxFN(IVCXSCST3, I) * RVCxI6) / RVCxD5
   RVCxA(I) = (RVCxH(IVCXSCST1, I) * RVCxI4 + RVCxH(IVCXSCST2, I) * RVCxI5 + RVCxH(IVCXSCST3, I) * RVCxI6) / RVCxD5
   RVCxB(I) = RVCxFN(IVCXSCST1, I) * (RVCxLO(IVCXSCST2) - RVCxLO(IVCXSCST3)) + RVCxFN(IVCXSCST2, I) * (RVCxLO(IVCXSCST3) - RVCxLO(IVCXSCST1))
   RVCxC(I) = RVCxFN(IVCXSCST3, I) * (RVCxLO(IVCXSCST1) - RVCxLO(IVCXSCST2)) / RVCxD5
   RVCx6(I) = RVCxFN(IVCXSCST1, I) * (RVCxLA(IVCXSCST3) - RVCxLA(IVCXSCST2)) + RVCxFN(IVCXSCST2, I) * (RVCxLA(IVCXSCST1) - RVCxLA(IVCXSCST3))
   RVCx8(I) = RVCxFN(IVCXSCST3, I) * (RVCxLA(IVCXSCST1) - RVCxLA(IVCXSCST2)) / RVCxD5
   RVCx9(I) = RVCxFN(IVCXSCST1, I) * (RVCxLA(IVCXSCST3) - RVCxLA(IVCXSCST2)) + RVCxFN(IVCXSCST2, I) * (RVCxLA(IVCXSCST1) - RVCxLA(IVCXSCST3))
   RVCx10(I) = RVCxFN(IVCXSCST3, I) * (RVCxLA(IVCXSCST1) - RVCxLA(IVCXSCST2)) / RVCxD5
CONTINUE
10000 CONTINUE
RETURN
END

INTERP  Local Symbols

Name   Class   Type      Size
IVSXFIN . . . . . . param
IVSXIN   . . . . . . param
I       . . . . . . local  INTEGER*4  4
IVCXSCST1. . . . . TEMP2  INTEGER*4  4
IVCXSCST2. . . . . TEMP2  INTEGER*4  4
IVCXSCST3. . . . . TEMP2  INTEGER*4  4
ICCXN . . . . . . SCPS1  INTEGER*4  4
IVCXSCS . . . . . . TEMP2  INTEGER*4  4
ITLX1 . . . . . . TEMP2  INTEGER*4  4
RVCXI3. . . . . . TEMP3  REAL*8   8
### INTERP Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVXI4</td>
<td>TEMP3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXI5</td>
<td>TEMP3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXD5</td>
<td>TEMP3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXFN</td>
<td>SCPS1A</td>
<td>REAL*8</td>
<td>43200</td>
</tr>
<tr>
<td>RVXH</td>
<td>SCPS1A</td>
<td>REAL*8</td>
<td>43200</td>
</tr>
<tr>
<td>RVXLA</td>
<td>SCPS1</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVXLO</td>
<td>SCPS1</td>
<td>REAL*8</td>
<td>14400</td>
</tr>
<tr>
<td>RVXAOA</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXAOB</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXBOA</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXBOB</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXX5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVXX6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
</tbody>
</table>

176
SUBROUTINE CASE1(RVSXH, IVSXJ)

CASE1 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
       IS IN THE CASE 1 PROFILE

CALLED BY: IONOPAR

CALLS: PGVAL, PGF2, PGFB

---------------------------------------------------------------

AUTHOR: ERIC L. STROBEL & MICHAEL H. REILLY

DATE: 09/01/87

VERSION: 1.0

---------------------------------------------------------------

REVISED: 09/01/87 -- V1.0. Initial revision.

---------------------------------------------------------------

USES: RVSXH

The current height.

To determine which part of the CASE 1 profile is being
operated on. The CASE 1 profile consists of (above the E)
the linear valley, the parabolic F2, and the topside.

RETURNS: IVSXJ

This helps distinguish which
boundary is being approached.

---------------------------------------------------------------

INTEGER IVSXJ, IVCXSZ, IVCXCASE, IVCXSX, IVCXSY

REAL*8 RVSXH, RVCXHU, RVCXHBND, RVCXHL, RVCXHT5
REAL*8 RPCXHTP, RPCXPI, RPCXDR, RPCXRE
REAL*8 RVCXV1, RVCXV2, RVCXX
REAL*8 RVCXF40, RVCXF65, RVCXK1, RVCXKL, RVCXXU
REAL*8 RVCXA2, RVCXB2, RVCXC2, RVCXHT3, RVCXHT4
REAL*8 RVCXL1, RVCXL01, RVCXHBOT, RVCXFREQ, RVCXRPI
REAL*8 RVCXHCT, RTLX1, RTLX2

COMMON /PRAM/ RPCXPI, RPCXDR, RPCXRE, RPCXHTP
COMMON /IONO3/ RVCXV1, RVCXV2, RVCXXX, IVCXCASE, RVCXHBND
COMMON /MORE/ IVCXSX, IVCXSY, IVCXSZ
COMMON /TEMP1/ RVCXF40, RVCXF65, RVCXK1, RVCXHL
682 COMMON /VAR1/ RVCXZL, RVCXZU, RVCXHU, RTLZ1, RTLZ2
683 COMMON /VAR4/ RVCXA2, RVCXB2, RVCXC2, RVCXHT3, RVCXHT4, RVCXHT5
684 COMMON /OTHER/ RVCXLI, RVCXL01, RVCXHBOT, RVCXFREQ, RVCXRPI, RVCXHCT

685 C  
686 IF (RVSXH.LE.RVCXHU) THEN
687 C  
688 C  We're in the valley region, so go calculate the parameters
689 C  from the valley profile.
690 C  
691 CALL PGVAL(RVSXH)  
692 IF (IVCXSZ.GT.0) THEN
693 C  
694 RVCXHBND = RVCXHU
695 C  
696 ELSE
697 C  
698 RVCXHBND = RVCXHL
699 ELSE
700 RETURN
701 ENDIF
702 IF (RVSXH.LE.RVCXHT5) THEN
703 C  
704 C  We're in the F2 region, so go calculate the parameters
705 C  from the F2 profile.
706 C  
707 CALL PGF2(RVSXH)  
708 IF (IVCXSZ.GT.0) THEN
709 C  
710 RVCXHBND = RVCXHTS
711 C  
712 ELSE
713 C  
714 RVCXHBND = RVCXHU
715 ELSE
716 RETURN
717 ENDIF
718 IF (RVSXH.LE.RVCXHTP) THEN
719 C  
720 C  We're in the topside region, so go calculate the parameters
721 C  from the topside profile.
722 C  
723 CALL PGFB(RVSXH)  
724 IF (IVCXSZ.GT.0) THEN
725 C  
726 RVCXHBND = RVCXHTP
727 C  
728 ELSE
729 C  
730 ELSE
731 C  We're beyond the cutoff of the ionosphere.
### CASE1 Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVSXJ</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXH</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVCXXX</td>
<td></td>
<td>IONO3</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXF40</td>
<td></td>
<td>TEMP1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXF65</td>
<td></td>
<td>TEMP1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXK1</td>
<td></td>
<td>TEMP1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXXL</td>
<td></td>
<td>VAR1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXXU</td>
<td></td>
<td>VAR1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXA2</td>
<td></td>
<td>VAR4</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXB2</td>
<td></td>
<td>VAR4</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXC2</td>
<td></td>
<td>VAR4</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXHT3</td>
<td></td>
<td>VAR4</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXHT4</td>
<td></td>
<td>VAR4</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXLI</td>
<td></td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXLO1</td>
<td></td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXHBO</td>
<td></td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXFREQ</td>
<td></td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXRPI</td>
<td></td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXHCT</td>
<td></td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RTLX1</td>
<td></td>
<td>VAR1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RTLX2</td>
<td></td>
<td>VAR1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>IVCXSZ</td>
<td></td>
<td>MORE</td>
<td>INTEGER*4</td>
</tr>
<tr>
<td>IVCXCASE</td>
<td></td>
<td>IONO3</td>
<td>INTEGER*4</td>
</tr>
<tr>
<td>IVCXXS</td>
<td></td>
<td>MORE</td>
<td>INTEGER*4</td>
</tr>
<tr>
<td>IVCXSY</td>
<td></td>
<td>MORE</td>
<td>INTEGER*4</td>
</tr>
<tr>
<td>RVCXHU</td>
<td></td>
<td>VAR1</td>
<td>REAL*8</td>
</tr>
</tbody>
</table>
## CASE1 Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RV CXHBND</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RV CXHL</td>
<td>TEMPI</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXHTS</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXHTP</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXD R</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXDR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RV CXV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE CASE2(RVSxH, IVSxJ)

CASE2 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM
IS IN THE CASE 2 PROFILE

CALLED BY: IONOPAR

CALLS: PGVAL, PGF1P, PGF2, PGFB

AUTHOR: ERIC L. STROBEL & MICHAEL H. REILLY

DATE: 09/01/87

VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: RVSxH

To determine which part of the CASE 2 profile is being
operated on. The CASE 2 profile consists of (above the E)
the linear valley, a parabolic F1, the parabolic F2,
and the topside.

RETURNS: IVSxJ

This helps distinguish which boundary is being approached.

INTEGER IVSxJ, IVCxSZ, IVCxSX, IVCxSY, IVCxCASE

REAL*8 RVSxH, RVCxH1P, RVCxHBND, RVCxHL, RVCxHT4

REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE

REAL*8 RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, RVCxRPI

REAL*8 RVCxV1, RVCxV2, RVCxxx

REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxA1, RVCxB1, RVCxC1

REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RTCxA

COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP

COMMON /OTHER/ RVCxL1, RVCxLO1, RVCxHBOT, RVCxFREQ, R'VCxRPI, RVCxHCT

COMMON /IONO3/ RVCxV1, RVCxV2, RVCxxx, IVCxCASE, RVCxHBND

COMMON /MORE/ IVCxSZ, IVCxSY, IVCxSZ
COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RTCxA
COMMON /VAR4/ RVCxA2, RVCxB2, RVCxC2, RVCxH1T3, RVCxH1T4, RVCxH1T5

IF (RVSxH.LE.RVCxH1P) THEN
  CALL PGVAL(RVSxH)
  IF (IVCxSZ.GT.0) THEN
    RVCxHBND = RVCxH1P
    IVSxJ = 5
  ELSE
    RVCxHBND = RVCxHL
    IVSxJ = 4
  ENDIF
  RETURN
ENDIF

IF (RVSxH.LE.RVCxHT4) THEN
  CALL PGF1P(RVSxH)
  IF (IVCxSZ.GT.0) THEN
    RVCxHBND = RVCxHT4
    IVSxJ = 6
  ELSE
    RVCxHBND = RVCxH1P
    IVSxJ = 5
  ENDIF
  RETURN
ENDIF

IF (RVSxH.LE.RVCxHT5) THEN
  CALL PGF2(RVSxH)
  IF (IVCxSZ.GT.0) THEN
    RVCxHBND = RVCxHT5
    IVSxJ = 7
  ELSE
    RVCxHBND = RVCxHT4
    IVSxJ = 6
  ENDIF
  RETURN
ENDIF

IF (RVSxH.LE.RPCxHTP) THEN
We're in the topside region, so go calculate the parameters from the topside profile.

CALL PGFB(RVSXH)

IF (IVCXSZ.GT.0) THEN
  RVCXHBND = RPCXHTP
  IVSxJ = 3
ELSE
  RVCXHBND = RVCXHT5
  IVSxJ = 7
ENDIF

ELSE
  We're beyond the cutoff of the ionosphere.

IF (IVCXSZ.GT.0) THEN
  RVCXHBND = RVCXHCT
ELSE
  RVCXHBND = RPCXHTP
ENDIF

ENDIF

Take care of things if the ray is headed toward a boundary that's above the cutoff height.

IF (IVCXSZ.GT.0.AND.RVCXHBND.GE.RVCXHCT) THEN
  RVCXHBND = RVCXHCT
  IVSxJ = 1
ENDIF

RETURN

END
CASE2  Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVCXK1</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXAI</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXBI</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXCI</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXAO</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXBO</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXC2</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXHT3</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IVCXSZ</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSK</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXCASE</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCXHIP</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHBD</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHL</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHT4</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHT5</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXHTP</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHCT</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXIR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXRE</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXLI</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE CASE3(RVSxH, IVSxJ)

CASE3 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM IS IN THE CASE 3 PROFILE

CALLED BY: IONOPAR

CALLS: PGVAL, PGFIL, PGF2, PGFB

AUTHOR: ERIC L. STROBEL & MICHAEL H. REILLY

DATE: 09/01/87

VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: RVSxH

To determine which part of the CASE 3 profile is being operated on. The CASE 3 profile consists of (above the E, the linear valley, a linear F1, the parabolic F2, and the topside.

RETURNS: IVSxJ

This helps distinguish which boundary is being approached.

INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY

REAL*8 RVSxH, RVCxH1L, RVCxHBND, RVCxHL, RVCxH2
REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR
REAL*8 RPCxRE, RVCxL1, RVCxLO1, RVCxHBO1, RVCxFREQ
REAL*8 RPCxRPI, RVCxV1, RVCxV2, RVCxxX, RVCxF40, RVCxF65
REAL*8 RVCxHBL, RVCxYS, RVCxSL1, RVCxSL2, RVCxA2, RVCkx1
REAL*8 RVCxB2, RVCxC2, RVCxHT3, RVCxHT4

COMMON /PRAM/ RPCxPI, RPCxRE, RPCxHTP
COMMON /OTHER/ RVCxL1, RVCxLO1, RVCxHBO1, RVCxFREQ, RPCxRPI, RVCxHT3
COMMON /IONO3/ RVCxV1, RVCxV2, RVCxxX, IVCxCASE, RVCxHBND
COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
COMMON /TEMPI/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
COMMON /VAR2/ RVCxHB1,RVCxHS2,RVCxUL5.RVCxUL2.RVCxH1L
COMMON /VAR4/ RVCxA2,RVCxB2,RVCxC2,RVCxHT3.RVCxHT4.RVCxHT5

C

IF (RVSxH.LE.RVCxH1L) THEN

C We're in the valley region. so go calculate the parameters

CALL PGVAL(RVSxH)
IF (IVCxSZ.GT.0) THEN
  RVCxHBND = RVCxH1L
  IVSxJ = 5
ELSE
  RVCxHBND = RVCxH1L
  IVSxJ = 4
ENDIF
RETURN
ENDIF

IF (RVSxH.LE.RVCxH2) THEN

C We're in the F1 region. so go calculate the parameters

CALL PGF1L(RVSxH)
IF (IVCxSZ.GT.0) THEN
  RVCxHBND = RVCxH2
  IVSxJ = 6
ELSE
  RVCxHBND = RVCxH1L
  IVSxJ = 5
ENDIF
RETURN
ENDIF

IF (RVSxH.LE.RVCxHT5) THEN

C We're in the F2 region. so go calculate the parameters

CALL PGF2(RVSxH)
IF (IVCxSZ.GT.0) THEN
  RVCxHBND = RVCxHT5
  IVSxJ = 7
ELSE
  RVCxHBND = RVCxH2
  IVSxJ = 6
ENDIF
RETURN
ENDIF

IF (RVSxH.LE.RVCxHTP) THEN
We're in the topside region, so go calculate the parameters from the topside profile.

```fortran
CALL PGFB(RVSXH)
IF (IVCSZ.GT.0) THEN
    RVCHBND = RPCXHTP
    IVS UN = 3
ELSE
    RVCHBND = RVCHT5
    IVS UN = 7
ENDIF
ELSE
    We're beyond the cutoff of the ionosphere.
ENDIF
Take care of things if the ray is headed toward a boundary that's above the cutoff height.
IF (IVCSZ.GT.0.AND.RVCHBND.GE.RVCHCT) THEN
    RVCHBND = RVCHCT
    IVS UN = 1
ENDIF
RETURN
END
```

**CASE3 Local Symbols**

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVSUN</td>
<td>.......</td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSXH</td>
<td>.......</td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVCHCT</td>
<td>.......</td>
<td>OTHER</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>.......</td>
<td>PRAM</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RPCXDR</td>
<td>.......</td>
<td>PRAM</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RPCXRE</td>
<td>.......</td>
<td>PRAM</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVCKLL1</td>
<td>.......</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCKLO1</td>
<td>.......</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCKHBO1</td>
<td>.......</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCKFPEQ</td>
<td>.......</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCKRPI</td>
<td>.......</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
</tbody>
</table>
**CASE3 Local Symbols**

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVVXV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVVXV2</td>
<td>IONO3</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVVXX</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVVXFR0</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXFR65</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXH1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXS</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXSL1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXSL2</td>
<td>VAR4</td>
<td>REAL*4</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXK1</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXB2</td>
<td>VAR4</td>
<td>REAL*4</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXC2</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVVXXHT3</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXH4</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>IVVXXS</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVVXXCASE</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVVXXSX</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVVXXSY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVVXXH1L</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXHBND</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXH</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXR2</td>
<td>VAR2</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVVXXHT5</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXHTP</td>
<td>PRAM</td>
<td>REAL*3</td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE CASE4(RVSxH, IVSxJ)

CASE4 -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM IS IN THE CASE 4 PROFILE

CALLED BY: IONOPAR

CALLS: PGVAL, PGF12, PGF2, PGF3

AUTHOR: ERIC L. STROBEL & MICHAEL H. REILLY

DATE: 09/01/87

VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: RVSxH

The current height.

To determine which part of the CASE 4 profile is being operated on. The CASE 4 profile consists of above the linear valley, a portion of the $F_2$, a parabolic $F_1$, the rest of the parabolic $F_2$, and the topside.

RETURNS: IVSxJ

This helps distinguish which boundary is being approached.

INTEGER IVSxJ, IVCXSZ, IVCXCASE, IVCXSX, IVCXSY

REAL*8 RVSxH, RVCxHU, RVCxHBND, RVCxHL, RVCxHT3, RVCxHT4
REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE
REAL*8 RPCxL1, RVCxL01, RVCxHBOT, RVCxFREQ, RVCxSPI
REAL*8 RPCxL1, RVCxV2, RVCxXX, RVCxAA2, RVCxLB, RVCxCC2
REAL*8 RVCxF40, RVCxF65, RVCxKL, RVCxXL, RVCxUX
REAL*8 RTLxl, RTLx2

COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP
COMMON /OTHER/ RVCxL1, RVCxL01, RVCxHBOT, RVCxFREQ, RVCxSPI, RVCxHTP
COMMON /IONO3/ RVCxV2, RVCxXX, IVCXCASE, RVCxHBND
COMMON /MORE/ IVCXSX, IVCXSY, IVCXSZ
COMMON /TEMPI/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RTLx1, RTLx2
COMMON /VAR4/ RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4, RVCxHT5

IF (RVSxH.LE.RVCxHU) THEN
  CALL PGVAL(RVSxH)
  IF (IVCxSZ.T.GT.3) THEN
    RVCxHBN = RVCxHU
    IVSXJ = 5
  ELSE
    RVCxHBN = RVCxHL
    IVSXJ = 4
  ENDIF
ENDIF
RETURN

IF (RVSxH.LE.RVCxHT3) THEN
  CALL PGF2(RVSxH)
  IF (IVCxSZ.T.GT.0) THEN
    RVCxHBN = RVCxHT3
    IVSXJ = 7
  ELSE
    RVCxHBN = RVCxHT3
    IVSXJ = 5
  ENDIF
ENDIF
RETURN

IF (RVSxH.LE.RVCxHT4) THEN
  CALL PGF1P(RVSxH)
  IF (IVCxSZ.T.GT.0) THEN
    RVCxHBN = RVCxHT4
    IVSXJ = 7
  ELSE
    RVCxHBN = RVCxHT4
    IVSXJ = 6
  ENDIF
ENDIF
RETURN

We're in the valley region. so go calculate the parameters
from the valley profile.

We're in the F1 region. so go calculate the parameters
from the F1 profile.

A portion of the F2 parabola is visible under the F1 layer.
(It's unknown how likely this is, but it at least appears
to be possible in the RADAR-C model, so it is safest to
include code for the possibility.)
We're in the more ordinary portion of the F2 region, so go calculate the parameters from the F2 profile.

CALL PGF2(RVSxH)
IF (IVCXSZ.GT.0) THEN
RVCXBND = RVCxHT5
IVSxJ = 8
ELSE
RVCXBND = RVCxHT4
IVSxJ = 7
ENDIF
RETURN

We're in the topside region, so go calculate the parameters from the topside profile.

CALL PGFB(RVSxH)
IF (IVCXSZ.GT.0) THEN
RVCXBND = RPCxHTP
IVSxJ = 9
ELSE
RVCXBND = RVCxHT5
IVSxJ = 8
ENDIF

We're beyond the cutoff of the ionosphere.

IF (IVCXSZ.GT.0) THEN
RVCXBND = RVCxHCT
ELSE
RVCXBND = RPCxHTP
ENDIF

Take care of things if the ray is headed toward a boundary that's above the cutoff height.

IF (IVCXSZ.GT.0.AND.RVCXBND.GE.RVCxHCT) THEN
RVCXBND = RVCxHCT
IVSxJ = 1
ENDIF
RETURN
END
<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVSXJ</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSSXH</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVCXHU</td>
<td></td>
<td>VAR1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHBND</td>
<td></td>
<td>ION03 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHL</td>
<td></td>
<td>TEMP1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHT3</td>
<td></td>
<td>VAR4 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHT4</td>
<td></td>
<td>VAR4 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHT5</td>
<td></td>
<td>VAR4 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXHTP</td>
<td></td>
<td>PRAM REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHCT</td>
<td></td>
<td>OTHER REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXPI</td>
<td></td>
<td>PRAM REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXDR</td>
<td></td>
<td>PRAM REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXRE</td>
<td></td>
<td>OTHER REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXL1</td>
<td></td>
<td>OTHER REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXLO1</td>
<td></td>
<td>OTHER REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHBOT</td>
<td></td>
<td>OTHER REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXFREQ</td>
<td></td>
<td>OTHER REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXRPI</td>
<td></td>
<td>OTHER REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXVI</td>
<td></td>
<td>ION03 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXV2</td>
<td></td>
<td>ION03 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXXK</td>
<td></td>
<td>ION03 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXA2</td>
<td></td>
<td>VAR4 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXB2</td>
<td></td>
<td>VAR4 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXC2</td>
<td></td>
<td>VAR4 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXF40</td>
<td></td>
<td>TEMP1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXF65</td>
<td></td>
<td>TEMP1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXK1</td>
<td></td>
<td>TEMP1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXKL</td>
<td></td>
<td>VAR1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXXU</td>
<td></td>
<td>VAR1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX1</td>
<td></td>
<td>VAR1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX2</td>
<td></td>
<td>VAR1 REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IVCXSZ</td>
<td></td>
<td>MORE INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXCASE</td>
<td></td>
<td>ION03 INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSX</td>
<td></td>
<td>MORE INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSY</td>
<td></td>
<td>MORE INTEGER*4</td>
<td>4</td>
</tr>
</tbody>
</table>
SUBROUTINE CASE5(RVSxH, IVSxJ)

CASES -- SUBROUTINE TO SORT OUT WHERE IN HEIGHT THE PROGRAM IS IN THE CASE 5 PROFILE

CALLED BY: NOPAR

CALLS: PGVAL, PGFIP, PGFC, PGFB

---------------------------------------------------------------

AUTHOR: ERIC L. STROBEL & MICHAEL H. REILLY

DATE: 09/01/87

VERSION: 1.0

---------------------------------------------------------------

REVISED: 09/01/87 -- V1.0. Initial revision.

---------------------------------------------------------------

USES: RVSxH The current height.

To determine which part of the CASE 5 profile is being operated on. The CASE 5 profile consists of (above the E) the linear valley, a parabolic Fl, an additional valley segment, the parabolic F2, and the topside.

RETURNS: IVSxJ This helps distinguish which boundary is being approached.

----------------------------------------------------------------

INTEGER IVSxJ, IVCxSZ, IVCxCASE, IVCxSX, IVCxSY

REAL*8 RVSxH, RVCxHU, RVCxHBND, RVCxHL, RVCxHT3, RVCxHT4

REAL*8 RVCxHT5, RPCxHTP, RVCxHCT, RPCxPI, RPCxDR, RPCxRE

REAL*8 RVCxL1, RVCxL01, RVCxHBOT, RVCxFREQ, RVCxRPI

REAL*8 RVCxV1, RVCxV2, RVCxxX, RVCxA2, RVCxB2, RVCxC2

REAL*8 RVCxF40, RVCxF65, RVCxK1, RVCxXL, RVCxXU

REAL*8 RVCxA1, RVCxB1, RVCxCL, RVCxH1P, RVCxH2P

REAL*8 RTLx1, RTLx2

COMMON /PRAM/ RPCxPI, RPCxDR, RPCxRE, RPCxHTP

COMMON /OTHER/ RVCxL1, RVCxL01, RVCxHBOT, RVCxFREQ, RVCxRPI, RVCxHTT

COMMON /IONO3/ RVCxV1, RVCxV2, RVCXX, IVCxCASE, RVCxHBND
COMMON /MORE/ IVCxSX, IVCxSY, IVCxSZ
COMMON /TEMPI/ RVCxF40, RVCxF65, RVCxK1, RVCxHL
COMMON /VAR1/ RVCxXL, RVCxxU, RVCxHU, RTLxL, RTLx2
COMMON /VAR3/ RVCxA1, RVCxB1, RVCxCl, RVCxH1P, RVCxH2P
COMMON /VAR4/ RVCxA2, RVCxB2, RVCxC2, RVCxH4T3, RVCxH4T4, RVCxHT5

C IF (RVSxH.LE.RVCxH1P) THEN
C We're in the valley region, so go calculate the parameters
C from the valley profile.
C CALL PGVAL(RVSxH)
C IF (IVCxSZ.GT.0) THEN
C RVCxHBN = RVCxH1P
C IVSxJ = 5
C ELSE
C RVCxHBN = RVCxH1L
C IVSxJ = 4
C ENDIF
C RETURN
C ENDF
C IF (RVSxH.LE.RVCxH2P) THEN
C We're in the F1 region, so go calculate the parameters
C from the F1 profile.
C CALL PGF1P(RVSxH)
C IF (IVCxSZ.GT.0) THEN
C RVCxHBN = RVCxH2P
C IVSxJ = 6
C ELSE
C RVCxHBN = RVCxH1P
C IVSxJ = 5
C ENDIF
C RETURN
C ENDF
C IF (RVSxH.LE.RVCxHU) THEN
C Surprise! A portion of the valley profile is seen above
C the F1. And yes, this really can in fact happen in the
C RADAR-C model.
C CALL PGVAL(RVSxH)
C IF (IVCxSZ.GT.0) THEN
C RVCxHBN = RVCxHU
C IVSxJ = 7
C ELSE
C RVCxHBN = RVCxH2P
C IVSxJ = 6
C ENDIF
RETURN
1271 ENDIF
1272 IF (RVSxH.LE.RVCxHT5) THEN
1273 C We're in the F2 region, so go calculate the parameters
1274 C from the F2 profile.
1275 C
1276 CALL PGF2(RVSxH)
1277 IF (IVCxSZ.GT.0) THEN
1278 RVCxHBND = RVCxHT5
1279 IVSxJ = 3
1280 ELSE
1281 RVCxHBND = RVCxHU
1282 IVSxJ = 7
1283 ENDIF
1284 RETURN
1285 ENDIF
1286 IF (RVSxH.LE.RPCxHT?) THEN
1287 C We're in the topside region, so go calculate the parameters'
1288 C from the topside profile.
1289 C
1290 CALL PGFB(RVSxH)
1291 IF (:VCxSZ.GT.0) THEN
1292 RVCxHBND = RPCxHTP
1293 IVSxJ = 9
1294 ELSE
1295 RVCxHBND = RVCxHT5
1296 IVSxJ = 3
1297 ENDIF
1298 ELSE
1299 C We're beyond the cutoff of the ionosphere.
1300 C
1301 IF (IVCxSZ.GT.0) THEN
1302 RVCxHBND = RVCxHCT
1303 ELSE
1304 RVCxHBND = RPCxHTP
1305 ENDIF
1306 ENDIF
1307 C Take care of things if the ray is headed toward a
1308 C boundary that's above the cutoff height.
1309 C
1310 IF (IVCxSZ.GT.0.AND.RVCxHBND.GE.RVCxHCT) THEN
1311 RVCxHBND = RVCxHCT
1312 IVSxJ = 1
1313 ENDIF
1314 C Return
CASE5  Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVSXJ</td>
<td>param</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXH</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVCXH</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXCASE</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXZ</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXZS</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXHU</td>
<td>VARI</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXHBBND</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHBL</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXHT3</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVXHT4</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVXHT5</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RPCXHTP</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXHCT</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXDR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXRED</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXHL</td>
<td>OTHER</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOL</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXHBT</td>
<td>OTHER</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVXFREQ</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXRP1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXV1</td>
<td>IONO3</td>
<td>REAL*4</td>
<td>3</td>
</tr>
<tr>
<td>RVXV2</td>
<td>IONO3</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVXX</td>
<td>IONO3</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVXH2</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXH1</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXF40</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXF65</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXKL</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXKL</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXXU</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXV1</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXV2</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXV3</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXH1</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXH2</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLX1</td>
<td>VAR1</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RTLX2</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>

196
SUBROUTINE PGVAL(RVSxH)

PGVAL -- SUBROUTINE to calculate the ionospheric parameters for the linear valley profile.

CALLED BY: CASE1, CASE2, CASE3, CASE4, CASE5

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/01/87

VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: RVSxH The current height.

To calculate the A, N1, and N2 parameters, as well as the plasma frequency squared XX. For more details, see the Radio Science paper and the RADAR-C report referred to in the documentation.

RETURNS: RVCxALPH A coef. in the index of refraction.

RVCxET1 A coef. in the index of refraction.

RVCxET2 A coef. in the index of refraction.

RVCXXX The plasma frequency squared.

INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3

INTEGER ITCxA
### Line# Source Line

1373  REAL*8 RVSXH, RVCXXX, RVCXAO, RVCXB0, RVCXALPH, RVCXFREQ
1374  REAL*8 RVLXHUXE, RVCXB(3), RVCXBFB(3), RVCXHU, RVLXHTR
1375  REAL*8 RVCXHT3, RVCXHL, RVLXXXE, RVLXHUXE, RVLXXXH2
1376  REAL*8 RVLXXX2, RVLXXY2, RVCXET1, RVCXET2, RVCXB5(3)
1377  REAL*8 RVCXB6(3), RVCXV1, RVCXET2, RVCXES(3), RVCXES(3)
1378  REAL*8 RVCXV2, RVCXL1, RVCXLO1, RVCXHBOT
1379  REAL*8 RVCXBETA, RVCXAS(3), RVCXAS(3), RVCXF40, RVCXC2
1380  REAL*8 RVCXF55, RVCXKL, RVCXHL, RVCXHU, RVCXAO, RVCXB2
1381  REAL*8 RTCXH, RTCXB, RTCXC, RTCXD, RTCXE
1382  C
1383  COMMON /OTHER/ RVCXL1, RVCXLO1, RVCXHBOT, RVCXFREQ, RTCXA, RTCXB
1384  COMMON /IONO1/ RVCXALPH, RVCXBETA, RVCXETL, RVCXET2, RVCXBFB, RVCXHB
1385  COMMON /IONO2/ RVCXAO, RVCXAS, RVCXB5, RVCXB6, RVCXES, RVCXES
1386  COMMON /IONO3/ RVCXV1, RVCXV2, RVCXXX, RTCXH, RTCXC
1387  COMMON /TEMPI/ RVCXF40, RVCXFB5, RVCXK1, RVCXHL
1388  COMMON /TEMP2/ RVCXSCS, RVCXST2, RVCXST3, IFCXN4
1389  COMMON /VAR1/ RVCXKL, RVCXLU, RVCXHU, RVCXAO, RVCXB0
1390  COMMON /VAR2/ RVCXAO, RVCXLB, RVCXCL, RTCXH, RTCXD, RTCXE
1391  C
1392  RVCXAO = RVCXAO + RVCXBO - RVSXH
1393  RVCXALPH = RVCXBO / RVCXFREQ
1394  IF (IFCXN4 .NE. 1) THEN
1395  RVLXHUXE = (0.98000*RVCXB(3)) * (0.98000*RVCXB(3))
1396  RVLXHUXE = RVLXHUXE / (2.0D00*RVCXB(3) * (RVCXB(3) - RVCXHU)
1397  RVLXHTR = (RVCXHT3 - RVCXHL) / (RVCXHU - RVCXHL)
1398  RVLXXXE = RVCXAO / RVCXFB(1) - RVCXBO * RVLXHTR * RVLXHUXE
1399  RVLXHUX2 = -RVLXHUXE * RVCXFB(1) / RVCXFB(1)
1400  RVLXXH2 = -RVCXAO * RVLXHTR
1401  RVLXXX2 = RVLXXX2 * RVLXHUX2
1402  RVLXXY2 = -(RVCXB(2) - RVCXHU) / RVCXB(3)
1403  RVLXXH2 = RVLXXH2 * RVLXHU2
1404  RVCXET1 = RVLXXXE * RVCXB5(1) + RVLXXX2 * RVCXB5(3)
1405  RVCXET1 = RVCXET1 + RVLXXH2 * RVCXB6(2) + RVLXXY2 * RVCXB6
1406  RVCXET1 = -RVCXET1 / (RVCXV1 * RVCXET1)
1407  RVCXET2 = RVLXXXE * RVCXES(5) + RVLXXX2 * RVCXES(3)
1408  RVCXET2 = RVCXET2 + RVLXXH2 * RVCXES(6) + RVLXXY2 * RVCXES
1409  RVCXET2 = RVCXET2 / (RVCXV2 * RVCXET1)
1410  ENDIF
1411  RETURN
1412  END

### PGVAL Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXH</td>
<td>param</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXX2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXHUXE</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXX2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>

198
<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVLXXXXE.</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXHUK2.</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXHUY2.</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXHUK2.</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXHUY2.</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXHUY2.</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA2.</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXB2.</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCA</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTXC</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTXD</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IFCXN4.</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>TCXSCS.</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>TCXST1.</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>TCXST2.</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>TCXST3.</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>TCXST4.</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>TCXST5.</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>ITCX</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCA3.</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA0.</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA0.</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA1.</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA2.</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA3.</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA4.</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA5.</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA6.</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA7.</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA8.</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA9.</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA10.</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA11.</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA12.</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA13.</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA14.</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA15.</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA16.</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA17.</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA18.</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA19.</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA20.</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA21.</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA22.</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA23.</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA24.</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA25.</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
PGVAL  Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVCXK1</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXXL</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXXU</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE PGFL RVSxH:

PGFL -- SUBROUTINE to calculate the ionospheric parameters for the linear F1 region.

CALLED BY: CASE3

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/01/87

VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: RVSxH The current height.

To calculate the A, N1, and N2 parameters, as well as the plasma frequency squared XX. For more details, see the Radio Science paper and the RADAR-C report referred to in the documentation.

RETURNS: RVCxALPH A coef. in the index of refraction.

RVCxET1 A coef. in the index of refraction.

RVCxET2 A coef. in the index of refraction.

RVCxxx The plasma frequency squared.

INTEGER IFCXN4, ICXSCT, ICXSCT1, ICXSCT2, ICXSCT3

INTEGER ITCxA
1465 REAL*8 RVCCXXX, RVCCSL1, RVLxHTV, RVCCALPH, RVCCxFREQ
1466 REAL*8 RVCCXHB1, RVCCYS, RVLxS1X1, RVLxS1X1, RVLxYSX1
1467 REAL*8 RVCCxHB(3), RVCCxFB(3), RVCCxHB2, RVLxXXX1, RVLxXXX1
1468 REAL*8 RVCCxET1, RVCCxET2, RVCCxB5(3), RVCCxB6(3), RVCCxE5(3)
1469 REAL*8 RVCCxE6(3), RVCCxV1, RVCCxV2, RVLxS1X2, RVLxXXX2
1470 REAL*8 RVLLXXH2, RVLxYSY2, RVLxS1Y2, RVLxXXXY2, RVCCxA5(3)
1471 REAL*8 RVCCxA6(3), RVCCxL1, RVCCxLO1, RVCCxHBOT, RVCCxETRA, RVCCxH2
1472 REAL*8 RTCxA, RTCxB, RTCxC, RTCxD, RTCxE
1473 COMMON / OTHER / RVCCxL1, RVCCxLO1, RVCCxHBOT, RVCCxFREQ, RTCxA, RTCxB
1474 COMMON / TON01 / RVCCxALPH, RVCCxET1, RVCCxET2, RVCCxFB, RVCCxHB
1475 COMMON / TON02 / RVCCxA5, RVCCxET1, RVCCxET2, RVCCxB6, RVCCxE5
1476 COMMON / TON03 / RVCCxE6, RVCCxV1, RVCCxV2, RVCCxV3, RTCxA, RTCxC
1477 COMMON / TEMP2 / RVCCxET2, RVCCxET1, RVCCxET2, RVCCxET3, IFCN4
1478 COMMON / VAR2 / RVCCxHB1, RVCCxHB2, RVCCxYS, RVCCxSL1, RTCxD, RTCxE
1479 COMMON /0TON02 / RVLLXXH1, RVCCxHY2, RVCCxSY2, RVCCxSH1, RVCCxSH1
1480 COMMON /0TON03 / RVLLXXH2, RVLLXXXH2, RVCCxET2, RVCCxET1, RVCCxFB
1481 IF (IFCN4.EQ.1) RETURN
1482 IF (RVCCxYS.EQ.1) THEN
1483 RVLLXXH1 = 1.0D00
1484 RVLLxS1X1 = 1.0D00
1485 RVLLxS1X1 = 1.0D00
1486 RVLLxS1X1 = 1.0D00
1487 RVLLxS1X1 = 1.0D00
1488 ELSE
1489 RVLLxS1X1 = RVCCxHB(3) * RVCCxHB(3) * 2.0D00 * RVCCxFB(3) * RVCCxFB(3)
1490 # (RVCCxHB(3) - RVCCxHB(3))
1491 RVLLxS1X1 = (1.0D00 - (RVCCxFB(3) * RVCCxYSX1, RVCCxYS, RVCCxH2)
1492 RVCCxS1H1 = RVCCxB2 / (RVCCxYS * RVCCxYS)
1493 ENDIF
1494 RVLLXXX1 = RVLLxHTV + RVLLxS1X1
1495 RVLLXXX1 = 0.75D00 * (RVCCxSL1 + RVLLxHTV + RVLLxS1X1)
1496 RVCCxET1 = -RVLLXXX1 * RVCCxB5(2) - RVLLXXX1 * RVCCxB6(1)
1497 RVCCxET2 = RVLLXXX1 * RVCCxE5(2) - RVLLXXX1 * RVCCxE6(1)
1498 IF (RVCCxYS.EQ.1) THEN
1499 RVCCxET1 = RVCCxET1 / (RVCCxV1 * RVCCxFREQ)
1500 RVCCxET2 = RVCCxET2 / (RVCCxV2 * RVCCxFREQ)
1501 ENDIF
1502 RVLLxS1X2 = RVLLxS1H1 * RVLLxYSX1 * RVCCxFB(2) / RVCCxFB(3)
1503 RVLLxS1X2 = RVLLxHTV + RVLLxS1X2
1504 RVLLxXXH2 = -RVLLxHTV + RVLLxS1X1
1505 RVLLxSY2 = (RVCCxH2 - RVCCxHB(2)) / RVCCxHB(3)
1506 RVLLxS1Y2 = -RVLLxS1H1 / RVCCxH2
1507 RVLLxS1Y2 = RVLLxHTV + RVLLxS1Y2
1508 RVLLxET1 = RVCCxET1 - RVLLXXX2 * RVCCxB5(3) - RVLLXXX2 * RVCCxB6(3) - RVLLXXX2 * RVCCxE5(3)
1509 RVCCxET2 = RVCCxET2 + RVLLXXX2 * RVCCxE6(3) + RVLLXXX2 * RVCCxE6(3)
1510 RETURN
1511 END
<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXH</td>
<td>param</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXYSX1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXYSY2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXX1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXXX1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXXXY2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXSIH1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXSI1X2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXSI1Y2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXXXH1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXXX2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXSIY2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXXHI</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXSIX1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXSIX2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXXXH2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXSHTV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IFCXN4</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSCS</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSTC1</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSTC2</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSTC3</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>ITCXAF</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCXSSX</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXSL1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>9</td>
</tr>
<tr>
<td>RVCXALPH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXFRQ</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHB1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXYS</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXHS</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXFB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXH2</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXET</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXET2</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXEB</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXEB2</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXEB3</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXEB6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXEB6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXEV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXEV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXAV5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXAV6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXLI</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXLO1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXHBIT</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXEBITA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTCXA</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTCXB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
### Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCXC</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXD</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>9</td>
</tr>
<tr>
<td>RTCXE</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE PGF1P(RVSXH)

PGF1P -- SUBROUTINE to calculate the ionospheric parameters for the parabolic F1 region.

CALLED BY: CASE2, CASE4, CASE5

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 09/01/87
VERSION: 1.0
REVISED: 09/01/87 -- V1.0. Initial revision.

USES: RVSXH The current height.
To calculate the A, B, N1, and N2 parameters, as well as the plasma frequency squared XX. For more details, see the Radio Science paper and the RADAR-C report referred to in the documentation.

RETURNS: RVCXALPH A coef. in the index of refraction.
RVCXBETA A coef. in the index of refraction.
RVCXET1 A coef. in the index of refraction.
RVCXET2 A coef. in the index of refraction.
RVCXX The plasma frequency squared.
INTEGER IFCxN4, IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3
INTEGER ITCxA

REAL*8 RVSxH, RVLxYF1, RVLxHTV, RVCxB(3), RVCxX
REAL*8 RVCxFB(3), RVCxFREQ, RVCxALPH, RVCxBETA, RVLxXXX1
REAL*8 RVLxxXX1H, RVCxET1, RVCxET2, RVCxB5(3), RVCxB6(3)
REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxL1, RVCxL01
REAL*8 RVCxHBOT, RVCxA5(3), RVCxA6(3), RTCxA, RTCxB
REAL*8 RTCxC

COMMON /OTHER/ RVCxL, RVCxL01, RVCxBOT, RVCxFREQ, RTCxA, RTCxB
COMMON /ION01/ RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVCxFB, RVCxHB
COMMON /ION02/ RVCxA5, RVCxA6, RVCxB5, RVCxB6, RVCxE5, RVCxE6
COMMON /ION03/ RVCxV1, RVCxV2, RVCxXX, ITCxA, RTCxC
COMMON /TEMP2/ IVCxSCS, IVCxSCT1, IVCxSCT2, IVCxSCT3, IFCxN4

REAL*8 RVLxYF1 = 3.50000 * RVCxB(1)
REAL*8 RVLxHTV = (RVCxB(1) - RVSxH) / RVLxYF1
REAL*8 RVCxX = RVCxFB(2) * (1.0000 - RVLxHTV * RVLxHTV)
REAL*8 RVCxALPH = 2.0000 * RVCxFB(2) * RVLxHTV / (RVLxYF1 * RVCxBETA)
REAL*8 RVCxBETA = RVCxALPH / (2.0000 * RVLxHTV * RVLxYF1)

IF (IFCxN4 .EQ. 1) RETURN

RVLxxXX1 = RVCxX / (RVCxFB(2) * RVCxFREQ)
RVLxxXXH1 = -2.0000 * RVCxBETA * RVSxH *

#(1.0000 - RVSxH/RVCxB(1))
RVCxET1 = RVLxxXX1 * RVCxB5(2) + RVLxxXXH1 * RVCxB6(1)
RVCxET1 = -RVCxET1 / RVCxV1
RVCxET2 = RVLxxXX1 * RVCxE5(2) + RVLxxXXH1 * RVCxE6(1)
RVCxET2 = RVCxET2 / RVCxV2

RETURN

PGF1P Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSxH</td>
<td>param</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLxxXX1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLxYF1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLxxXXH1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLxHTV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IFCxN4</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCxSCS</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCxSCT1</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCxSCT2</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCxSCT3</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>ITCxA</td>
<td>ION03</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxHB</td>
<td>ION01</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxxX</td>
<td>ION03</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Type</td>
<td>Size</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>RVCFXFB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCFREQ</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCFALPH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCFBETA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCFET1</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCFET2</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCFX5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCFX6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCFX5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCFX6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCFV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCFV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCL1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>9</td>
</tr>
<tr>
<td>RVCL01</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>9</td>
</tr>
<tr>
<td>RVCHBOT</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCA5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCA6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RTCX0</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTCX0</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTCX0</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE PGF2(RVSxH)

PGF2 -- SUBROUTINE to calculate the ionospheric parameters for the parabolic F2 region.

CALLED BY: CASE1. CASE2. CASE3. CASE4. CASE5

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/01/87

VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: RVSxH The current height.

To calculate the A, B, N1, and N2 parameters, as well as the plasma frequency squared XX. For more details, see the Radio Science paper and the RADAR-C report referred to in the documentation.

RETURNS: RVCxALPH A coef. in the index of refraction.

RVCxBETA A coef. in the index of refraction.

RVCxET1 A coef. in the index of refraction.

RVCxET2 A coef. in the index of refraction.

RVCxxx The plasma frequency squared.
Line#  Source Line

1650  INTEGER  IFxN4,  IVCxSCS,  IVCxSCT1,  IVCxSCT2,  IVCxSCT3
1651  INTEGER  ITCxA
1652  REAL*8  RVSxH,  RVLxHTV,  RVCxHB(3),  RTLx1,  RVCxXX
1653  REAL*8  RVCxBF(3),  RVCxFREQ,  RVCxALPH,  RVCxBETA
1654  REAL*3  RVLxXXX2,  RVLxXXH2,  RVLxXXY2,  RVCxE1,  RVCxE2
1655  REAL*8  RVCxB5(3),  RVCxB6(3),  RVCxE5(3),  RVCxE6(3)
1656  REAL*8  RVCxV1,  RVCxV2,  RVCxXX,  RVCxLOL,  RVCxBOT
1657  REAL*8  RVCxA5(3),  RVCxA6(3),  RTCxA,  RTCxB,  RTCxC
1658  COMMON  /OTHER/  RVCxL1,  RVCxLOL,  RVCxHBOT,  RVCxFREQ,  RTCxA,  RTCxB
1659  COMMON  /IONO1/  RVCxALPH,  RVCxBETA,  RVCxE1,  RVCxE2,  RVCxFB,  RVCxH2
1660  COMMON  /IONO2/  RVCxA5,  RVCxA6,  RVCxB5,  RVCxB6,  RVCxE5,  RVCxE6
1661  COMMON  /IONO3/  RVCxV1,  RVCxV2,  RVCxXX,  ITxCxA,  RTCxC
1662  COMMON  /TEMP2/  IVCxSCS,  IVCxSCT1,  IVCxSCT2,  IVCxSCT3,  IFxN4
1663  REAL*8  RVLxHTV  =  (RVCxHB(2)  -  RVSxH)  /  RVCxHB(3)
1664  REAL*8  RTLx1  =  1.0000  -  RVLxHTV  *  RVLxHTV
1665  REAL*8  RVCxBETA  =  RVCxFB(3)  /  (RVCxHB(3)  *  RVCxBETA)
1666  REAL*8  RVCxALPH  =  2.0000  *  RVCxBETA  *  (RVCxHB(2)  -  RVSxH)
1667  IF  (IFxN4.EQ.1)  RETURN
1668  REAL*8  RVLxXXX2  =  RTLx1
1669  REAL*8  RVLxXXH2  =  -2.0000  *  RVCxBF(3)  *  RVLxHTV  /  RVCxHB(3)
1670  REAL*8  RVLxXXY2  =  -RVLxXXH2  *  RVLxHTV
1671  REAL*8  RVCxE1  =  RVLxXXX2  *  RVCxB5(3)  +  RVLxXXH2  *  RVCxB6(2)
1672  REAL*8  RVCxE2  =  -(RVCxE1  +  RVLxXXY2  *  RVCxB6(3))  /  (RVCxV1  *  RVCxFREQ)
1673  REAL*8  RVCxE2  =  (RVCxE2  +  RVLxXXY2  *  RVCxE6(3))  /  (RVCxV2  *  RVCxFREQ)
1674  RETURN
1675  END

PGF2  Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXH</td>
<td>param</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXXX2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXXX2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXXX2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLX1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXHTV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXB5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXB6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXE5</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXE6</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXLL</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Type</td>
<td>Size</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>RVCXLO1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXHBOT</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXA5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXA6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RTCXA</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXC</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IFCXN4</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCSXCS</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCSXCT1</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCSXCT2</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCSXCT3</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>ITCXA</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCXHB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXXX</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXFB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXFREQ</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXALPH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXBETA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXET1</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXET2</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE PGFB(RVSxH)

PGFB -- SUBROUTINE to calculate the ionospheric parameters
for the Bent topside region.

CALLED BY: CASE1, CASE2, CASE3, CASE4, CASE5

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/01/87

VERSION: 1.0

REVISED: 09/01/87 -- V1.0. Initial revision.

USES: RVSxH The current height.

To calculate the A, B, N1, and N2 parameters, as well as the
plasma frequency squared XX. For more details, see the
Radio Science paper and the RADAR-C report referred to
in the documentation.

RETURNS: RVCxALPH A coef. in the index of
refraction.
RVCxBETA A coef. in the index of
refraction.
RVCxET1 A coef. in the index of
refraction.
RVCxET2 A coef. in the index of
refraction.
RVCxxXX The plasma frequency squared.
Line#  Source Line
1733 INTEGER IFCN4, IVCXSCTS, IVCXSCT1, IVCXSCT2, IVCXSCT3
1734 INTEGER ITCA
1735 C
1736 REAL*8 RVLXALP, RVCXHB(3), RVLXA7, RVCXFB(3), RVSXH
1737 REAL*8 RVCXHT5, RVCXXX, RVCXLALPH, RVCXBETA, RVCXFREQ
1738 REAL*8 RVLXXXX2, RVLXXXH2, RVLXXXY2, RVCXET1, RVCXET2
1739 REAL*8 RVCXAS(3), RVCXA6(3), RVCXBS(3), RVCXB6(3)
1740 REAL*8 RVCXES(3), RVCXV1, RVCXV2, RVCXL1, RVCXL01
1741 REAL*8 RVCXBHBOT, RVCXAO2, RVCXB2, RVCXC2, RVCXHT3, RVCXHT4
1742 REAL*8 RTCXAM, RTCXBM, RTCXM
1743 COMMON /OTHER/ RVCXL1, RVCXL01, RVCXBHBOT, RVCXFREQ, RTCXAM, RTCXM
1744 COMMON /IONO1/ RVCXALPH, RVCXBETA, RVCXET1, RVCXET2, RVCXFB, RVCXHB
1745 COMMON /IONO2/ RVCXAS, RVCXA6, RVCXBS, RVCXB6, RVCXES, RVCXEX
1746 COMMON /IONO3/ RVCXV1, RVCXV2, RVCXX, RTCXAM, RTCXM
1747 COMMON /TEMP2/ IVCXSCTS, IVCXSCT1, IVCXSCT2, IVCXSCT3, IFCN4
1748 COMMON /VAR4/ RVCXAO2, RVCXB2, RVCXC2, RVCXHT3, RVCXHT4, RVCXHT5
1749 REAL*8 RVLXALP = 1.0000 / (1.375D00 * RVCXHB(3))
1750 REAL*8 RVLXA7 = 15.0000 * RVCXFB(3) * DEXP(RVLXALP*RVCXHT5)/16.0000
1751 REAL*8 RVCXALPH = -RVLXALP * RVCXXX / RVCXFREQ
1752 REAL*8 RVCXBETA = RVLXALP / 2.0000
1753 IF (IFCN4.EQ.1) RETURN
1754 RVLXXXX2 = RVCXXX / (RVCXFB(3) * RVCXFREQ)
1755 RVLXXXH2 = -RVCXALPH
1756 RVLXXXY2 = -RVCXALPH * (0.25000 - 1.375D00*RVLXALP*
1757 # (RVCXHT5-RVSXH))
1758 RVCXET1 = RVLXXXX2*RVCXBS(3) + RVLXXXH2*RVCXBS(3)
1759 RVCXET1 = -(RVCXET1 - RVLXXXY2*RVCXBS(3)) / RVCXV1
1760 RVCXET2 = (RVCXET2 + RVLXXXY2*RVCXBS(3)) / RVCXV2
1761 RETURN
1762 END

PGFB  Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXH</td>
<td>param</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXAO7</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RVLXXX2</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RVLXXX2</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RVLXALP</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXXH2</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>IFCN4</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSCT1</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSCT2</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
</tbody>
</table>

212
## PGFB Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVCXSCT3</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>ITCXA</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCXHB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXFB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXHT5</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXXX</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXALPH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXBETA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXFREQ</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXET1</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXET2</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXAS</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXBS</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXEB</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXEB2</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXLB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXHCT1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXRES2</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXRES3</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXRES4</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXRES5</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTXCA</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXC</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>

## Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE1</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>CASE2</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>CASE3</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>CASE4</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>CASE5</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>END</td>
<td>common</td>
<td>***</td>
<td>32</td>
</tr>
<tr>
<td>INBOX</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>INTERP</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>INTSIGN</td>
<td>extern</td>
<td>INTEGER*4</td>
<td>***</td>
</tr>
</tbody>
</table>
Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IONO1</td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>IONO2</td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>IONO3</td>
<td></td>
<td></td>
<td>144</td>
</tr>
<tr>
<td>IONOPAR</td>
<td></td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>MAINDAT</td>
<td></td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>MORE</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>OTHER</td>
<td></td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>PGF1L</td>
<td>SUBRT</td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>PGF1P</td>
<td>SUBRT</td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>PGF2</td>
<td>SUBRT</td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>PGFB</td>
<td>SUBRT</td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>PGVAL</td>
<td>SUBRT</td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>PRAM</td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>RAID</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>SCPS1</td>
<td></td>
<td></td>
<td>28804</td>
</tr>
<tr>
<td>SCPS1A</td>
<td></td>
<td></td>
<td>86400</td>
</tr>
<tr>
<td>START</td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>TEMP1</td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>TEMP2</td>
<td></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>TEMP3</td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>TRIANG</td>
<td></td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>VAR1</td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>VAR2</td>
<td></td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>VAR3</td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>VAR4</td>
<td></td>
<td></td>
<td>48</td>
</tr>
</tbody>
</table>

Code size = 356f (13679)
Data size = 019f (415)
SUBROUTINE ENDPT(IFSG, RVSXHO, RVSXHZ, RVSXS4, RVSGPI)

ENDPT -- SUBROUTINE TO CALCULATE THE ENDPOINTS OF THE INTERVAL
CALLED BY: RAYSUB
CALLS: ACCFSP

AUTHOR: MICHAEL H. REILLY
DATE: 07/25/36
VERSION: 1.1

REVISED: 07/25/86 -- INITIAL REVISION. TRANSLATED
FROM TEKTRONIX BASIC TO VAX FORTRAN BY
ERIC L. STROBEL.
07/30/86 -- V1.1. Change over to use of
REAL*8 precision in the calculations.
09/01/87 -- V2.0. Changed to prevent
problems with calculations on small numbers.

USES: IFSG A FLAG
       RVSXHO INITIAL HEIGHT
       RVSXHZ HEIGHT
       RVCXH5 NEW HEIGHT
       RVCPXALPH X
       RVCPXBETA X
       RVCPXET1 X-IONOSPHERIC PARAMETERS
       RVCPXET2 X
       RVCXHB HEIGHT OF IONOSPHERE BOTTOM
       RVCPXCY.CY.CZ
       RVCPXSX.SY.SZ

TO COMPUTE THE S.E.Z. COORDINATES FOR THE END OF THE
RAYPATH INCREMENT. (See the Radio Science paper
referred to in the documentation.)

RETURNS: IFSG
 Line#  Source Line
51 C          RVSxGPI    GROUP PATH INCREMENT
52 C          RVcxxXF,YF,ZF COORDINATES OF THE ENDPOINT
55 C          LOGICAL LTLxXT1, LTLxXT2, LTLxXT3, LTLxXT4
58 C          INTEGER IFSCXG, IVCXSX, IVCXSY, IVCXSZ
60 REAL*3 RVCCXCX, RVCCXCY, RVCCXCZ, RVCCCXLAT, RVCCXLON
61 REAL*3 RVCCXHB, RVCCXXF, RVCCXYP, RVCCXZF, RTLxTEMP
62 REAL*3 RVCCXH5, RVSXXH0, RVSXXHZ, RVSxGPI, RVCCXALPH
63 REAL*3 RVCCXBETA, RVCCXET1, RVCCXET2, RVLxEPS
64 REAL*3 RTLxXF1, RTLxXR2, RTLxXF6, RTLxXF7, RTLxBR1
65 REAL*3 RTLxAX1, RTLxXF5, RTLxG3, RTLxG4, RTLxGF1
66 REAL*3 RTLxGF1, RTLxGC5, RTLxGF7, RTLxGF2, RTLxGF2
67 REAL*8 RPCxPI, RPCxDTR, RPCxRE, RPCxHTP, RVSX54
69 COMMON /OTHER/ RVCCXLT0, RVCCXLON, RVCCXHB
70 COMMON /END/ RVCCXXF, RVCCXYP, RVCCXZF, RVCCXH5
71 COMMON /ZON1 / RVCCXALPH, RVCCXBETA, RVCCXET1, RVCCXET2
72 COMMON /MORE/ IVCXSX, IVCXSY, IVCXSZ, RVCCXCK, RVCCXCY, RVCCXZ
73 COMMON /PRAM/ IVCXSX, IVCXSY, IVCXSZ, RVCCXCK, RVCCXCY, RVCCXZ
74 COMMON /TEMP4 / RTLxXF1, RTLxGF2, RTLxGF2
78 C          Initializations.
79 RTLxG2 = 0.0
80 RTLxG3 = 0.0
81 RTLxG4 = 0.0
82 RTLxG5 = 0.0
83 RTLxG6 = 0.0
84 RTLxG7 = 0.0
85 RTLxGF5 = 0.0
86 RTLxGF6 = 0.0
87 RTLxGF7 = 0.0
88 RTLxGF8 = 0.0
89 RTLxGF9 = 0.0
90 RTLxGF10 = 0.0
91 RVCCXXF = 0.0D00
92 RVCCXYP = 0.0D00
93 RVSxGPI = 0.0D00
94 RVLxEPS = 1.0D-09
96 C          Check for being in free space.
98 IF (RVCCXH5.GE.(RVCCXHB = 0.02D00).AND.RVCCXH5.LE.
99 #(RPCxHTP = 0.02D00)) GO TO 10262
100 IF (IVCXSZ.GE.) THEN

216
IFSxG = 7
ELSE
IFSxG = 6
ENDIF

ACCFSF is the routine that does the free space propagation.

CALL ACCFSF(RVSxHO.RVSxS4.IFSxG)
GO TO 10267

RVCxZF = RVCxHZ - RVSxHO

Try to prevent loss of precision in the quantity
(sqrt(x) - sqrt(x-y)) with y small. First, define
'small'. then round the offending terms to zero.

IF (RVCxBETA.NE.0.0000) THEN
LTLxT1 = DABS(RVCxBETA*RVCxZF*RVCxZF)*LT.(RVLxEPS*RVCxX)
LTLxT2 = DABS(RVCxALPH*RVCxZF)*LT.(RVLxEPS*RVCxX)
IF (LTLxT1.AND.LTLxT2) THEN
RVCxALPH = 0.0D00
RVCxBETA = 0.0D00
GO TO 10267
ENDIF
IF (IFSxG.EQ.3) GO TO 10282
GO TO 10280
ENDIF

IF (RVCxALPH.NE.0.0000) THEN
LTLxT2 = DABS(RVCxALPH*RVCxZF)*LT.(RVLxEPS*RVCxX)
IF (LTLxT2) THEN
RVCxALPH = 0.0
ELSE
IF (IFSxG.EQ.3) GO TO 10272
GO TO 10269
ENDIF
ENDIF

From here on, the code follows the logic outlined in the
Radio Science paper referred to in the documentation.

RTLxF3 = RVCxXZ*RVCxZF / DSQRT(RVCxXZ)
GO TO 10293

RTLxR2 = RVCxXZ - RVCxALPH*RVCxZF
IF (RTLxR2.GT.0.0D00) THEN
RTLxF3 = DSQRT(RTLxR2)
ELSE
IFSxG = 3
RVCxZF = RVCxXZ / RVCxALPH
RTLxF3 = 0.0D00
GO TO 10265
ENDIF
152 10272 RTLxF6 = DSQRT(RVCxCZ)
153 RTLxF3 = RTLxF6 - RTLxF3
154 RTLxF3 = 2.0D00*IVCxSZ*RTLxF3 / RVCxALPH
155 GO TO 10293
156 10280 RTLxR2 = RVCxBETA*RVCxZF*RVCxZF - RVCxALPH*RVCxZF - RVCxCZ
157 IF (RTLxR2.GT.0.0D00) THEN
158 RTLxF4 = DSQRT(RTLxR2)
159 ELSE
160 IFsxG = 3
161 RVCxZF = (RVCxALPH - IVCxSZ*DSQRT(RVCxALPH
162 #*RVCxALPH - 4.0D00*RVCxBETA*RVCxCZ)) / (2.0D00*RVCxBETA)
163 RTLxF4 = 0.0D00
164 GO TO 10263
165 ENDIF
166 10282 RTLxB1 = DSQRT(ABS(RVCxBETA))
167 IF (RVCxBETA.LT.0.0) THEN
168 RTLxF6 = SQRT(RVCxALPH*RVCxALPH - 4.0*RVCxBETA*RVCxCZ)
169 IF (IFSxG.EQ.3) THEN
170 RTLxF4 = INTSIGN(-RVCxALPH)
171 ELSE
172 RTLxF4 = (2.0*RVCxBETA* RVCxZF - RVCxALPH)/RTLxF6
173 ENDIF
174 IF (DABS(-RVCxALPH/RTLxF6).GT.1.0) THEN
175 RTLxTEMP = DASIN(DFLOAT(INTSIGN(-RVCxALPH/RTLxF6))):.
176 ELSE
177 RTLxTEMP = DASIN(-RVCxALPH/RTLxF6)
178 ENDIF
179 RTLxF3 = RTLxTEMP - DASIN(RTLxF4)
180 ELSE
181 RTLxA2 = RVCxALPH / (2.0D00*RTLxB1)
182 RTLxF5 = DSQRT(RVCxCZ)
183 RTLxF6 = -RTLxA2 + RTLxF5
184 RTLxG3 = (RTLxB1*RVCxZF - RTLxA2 + RTLxF4) / RTLxF6
185 RTLxF3 = DLOG(RTLxG3)
186 ENDIF
187 RTLxF3 = IVCxSZ * RTLxF3 / RTLxB1
188 10293 RTLxC4 = DSQRT(RVCxCX)
189 RVSxGPI = RTLxF3
190 IF (IFSxG.GT.5) GO TO 10352
191 IF (RVCxET1.EQ.0.0D00) THEN
192 GO TO 10320
193 ELSE
194 RTLxT3 = (DSQRT(RVCxCX)*DABS(RVCxET1*RVCxZF))
195 # .LT. (RVLxEPS*RVCxCX*DSQRT(RVCxCZ))
196 IF (RTLxT3) THEN
197 RVCxET1 = 0.0D00
198 GO TO 10320
199 ENDIF
200 ENDIF
IF (IVCXSX.EQ.0.0) IVCXSX = -INTSIGN(RVCXET1)
RTLXC5 = RTLXC4 - RVCXET1*RTLXF3/(2.0D00*IVCXSX)
IF (RTLXC5.GT.0.0D00) GO TO 10320
RVCXF = RVCXCX / RVCXET1
IF (DABS(RVCXET1*RVCXF).LT.(RVLXEPS*RVCXCX)) THEN
  RVCXET1 = 0.0D00
  GO TO 10320
ENDIF
IFSXG = 4
RTLXF1 = 2.0D00*RTLXC4*IVCXSX / RVCXET1
RVSXGPI = RTLXF1
IF (RVCXET2.EQ.0.0D00) THEN
  RVCXF = RTLXF1*DSQRT(RVCXCZ)/IVCXSX
ELSE
  RVCXF = (RVCXCZ - (RTLXF6 - RVCXALPH*RTLXF1/ *2.0D00*IVCXSX))*2.0D00 / RVCXALPH
ENDIF
ELSE
  IF (RVCXET2.LT.0.0) THEN
    IF (DABS(-RVCXALPH/RTLXF6).GT.1.0) THEN
      RTLXTEMP = DASIN(DFLOAT(INTSIGN(-RVCXALPH/RTLXF6)
    ELSE
      RTLXTEMP = DASIN(-RVCXALPH/RTLXF6)
    ENDIF
    RTLXF4 = RTLXTEMP - RTLXBI*RTLXF1*IVCXSZ
    RTLXF4 = DSIN(RTLXF4)
    RVCXZF = (RVCXALPH + RTLXF6*RTLXF4)/(2.0*RVCXBETA
    ELSE
      RTLXG1 = RTLXF6*DEXP(RTLXBI*RTLXF1/IVCXSZ) + RTLXAL
      RVCXZF = (RTLXG1*RTLXG1 - RVCXCZ)/(2.0D00*RTLXBI*RTLX3
    ENDIF
  ENDIF
  IF (RVCXET2.EQ.0.0D00) THEN
    GO TO 10318
  ELSE
    LTLXT4 = (DSQRT(RVCXCY)*DABS(RVCXET2*RVCXZF) ^ .LT.(RVLXEPS*RVCXCY)DSQRT(RVCXCZ))
    IF (LTLXT4) THEN
      RVCXET2 = 0.0D00
      GO TO 10318
    ENDIF
  ENDIF
RTLXC6 = DSQRT(RVCXCY)
RTLXC7 = RTLXC6 - RVCXET2*RTLXF1/(2.0D00*IVCXSY)
IF (RTLXC7.LE.0.0D00) GO TO 10320
RVCXYF = (RVCXCY - RTLXC7*RTLXC7)/RVCXET2
GO TO 10341
10318 RVCXYF = RTLXF1 * DSQRT(RVCXCY) / IVCXSY
GO TO 10341
RTLxC6 = DSQRT(RVCxCY)
IF (RVCxET2.EQ.0.0D00) THEN
  GO TO 10341
ELSE
  LTLxT4 = (DSQRT(RVCxCY) * DABS(RVCxET2 * RVCxZF))
  IF (LTLxT4) THEN
    RVCxET2 = 0.0D00
  GO TO 10341
ENDIF

ENDIF

IF (IVCxSY.EQ.0) IVCxSY = INTSIGN(-RVCxET2)
RTLxC7 = RTLxC6 - RVCxET2 * RTLxF3 / (2.0D00 * IVCxSY)
IF (RTLxC7.GT.0.0D00) GO TO 10341
RVCxFY = RVCxCY / RVCxET2
IF (DABS(RVCxET2 * RVCxFY).LT.(RVLxEPS * RVCxCY)) THEN
  RVCxET2 = 0.0D00
  GO TO 10341
ENDIF

IFSxG = 5
RTLxF2 = 2.0D00 * RTLxC6 * IVCxSY / RVCxET2
RVSxGPI = RTLxF2
IF (RVCxBETA.EQ.0.0D00) THEN
  IF (RVCxALPH.EQ.0.0D00) THEN
    RVCxZF = RTLxF2 * DSQRT(RVCxCZ) / IVCxSZ
      * 2.0D00 * IVCxSZ) ** 2.0D00 / RVCxALPH
  ELSE
  ENDIF
ELSE
  IF (RVCxBETA.LT.0.0) THEN
    IF (DABS(-RVCxALPH/RTLxF6).GT.1.0) THEN
      RTLxTEMP = DASIN(DFLOAT(INTSIGN(-RVCxALPH/RTLxF6))
    ELSE
      RTLxTEMP = DASIN(-RVCxALPH/RTLxF6)
    ENDIF
    RTLxF4 = RTLxTEMP - RTLxBI * RTLxF2 * IVCxSZ
    RTLxF4 = DSIN(RTLxF4)
    RVCxFZ = (RVCxALPH + RTLxF6 * RTLxF4) / (2.0 * RVCxBETA)
    ELSE
    RTLxG2 = RTLxF6 * DEXP(RTLxB1 * RTLxF2 / IVCxSZ) - RTLxAL
    RTLxFZ = (RTLxG2 * RTLxG2 - RVCxCZ) / (2.0D00 * RTLxB1 * RTLxG2
      - RVCxALPH
      ENDIF
  ENDIF
ENDIF
IF (RVCxET1.EQ.0.0D00) THEN
  RVCxF = RTLxF2 * DSQRT(RVCxCX) / IVCxSX
ELSE
  RTLxC5 = RTLxC4 - RVCxET1 * RTLxF2 / (2.0D00 * IVCxSX)
RVCXF = (RVCX - RTLX5*RTLX5)/RVCXET1

ENDIF

IF (IFSXG.GT.2) RETURN

IF (RVCXET1.NE.0.0D00) THEN

RVCXF = (RVCX - RTLX5*RTLX5)/RVCXET1

ELSE IF (RVCXNE.0.0D00) THEN

RVCXF = RTLX4*IVCXSX*RTLX3

ENDIF

IF (RVCXET2.NE.0.0D00) THEN

RVCYF = (RVCXCY - RTLX7*RTLX7)/RVCXET2

RETURN

ENDIF

IF (RVCXCY.EQ.0.0000) RETURN

RVCYF = RTLX6*IVCXSY*RTLX3

RETURN

END

ENDPT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXGPI</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXS4</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXHZ</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXHO</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFSXG</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTLXTEMP</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXC4</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXG1</td>
<td>local</td>
<td>REAL*9</td>
<td>9</td>
</tr>
<tr>
<td>RTLXG5</td>
<td>local</td>
<td>REAL*9</td>
<td>9</td>
</tr>
<tr>
<td>RTLXG2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXG6</td>
<td>local</td>
<td>REAL*9</td>
<td>9</td>
</tr>
<tr>
<td>RTLXG7</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXG3</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXF4</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXF5</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXF6</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>LTLXT1</td>
<td>local</td>
<td>LOGICAL*4</td>
<td>4</td>
</tr>
<tr>
<td>LTLXT2</td>
<td>local</td>
<td>LOGICAL*4</td>
<td>4</td>
</tr>
<tr>
<td>LTLXT3</td>
<td>local</td>
<td>LOGICAL*4</td>
<td>4</td>
</tr>
<tr>
<td>LTLXT4</td>
<td>local</td>
<td>LOGICAL*4</td>
<td>4</td>
</tr>
<tr>
<td>RTLXR2</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXEPS</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXAX2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXB1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IVCXSX</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSZ</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCX CX</td>
<td>MORE</td>
<td>REAL*3</td>
<td>3</td>
</tr>
</tbody>
</table>
### ENDPT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVCCY</td>
<td>MORE</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCCZ</td>
<td>MORE</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCLAT0</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCLONO</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCHB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXXF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXYF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXZF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXH5</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXALPH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXBETA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXET1</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXET2</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXF3</td>
<td>TEMP4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXF1</td>
<td>TEMP4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXF2</td>
<td>TEMP4</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXDTR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXRE</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXHTP</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE PHSPL(IKSXG, RVSXPI)

PHSPL -- SUBROUTINE TO CALCULATE THE PHASE PATH LENGTH INCREMENT FOR THE CURRENT RAY PATH INCREMENT.

CALLED BY: RAYSUB
CALLS: PHSPL2, PHSPL3

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 09/30/87
VERSION: 1.0

REVISED: 09/30/87 -- V1.0. Initial revision.

USES:
RVCSXALPH \ RVCSXET1 / To decide on which PPI calc.
RVCSXBETA \ RVCSXET2 / to use.
RVCSXET1 /

IVCSX.Y.Z \ RVCSXXYZF / To calc. PPI in regions of constant plasma frequency.

IFSXG A flag variable.

To either pass off the phase path increment length calculation to PHSPL2.3 or to do it locally for the simplest case. The phase path calculations are done by considering the integral, where (mu) is the index of refraction. (See the Radio Science paper referred to in the documentation. eq. 8).

RETURNS: RVSXPI The increment of phase path length.
INTEGER IFSXG, IVCXSX, IVCXSY, IVCXSZ

REAL*8 RVSXPPI, RVCFX1, RVCFX2, RVCFX3, RVCFXBETA, RVCFXF
REAL*8 RVCFYF, RVCFZF, RVCFCX, RVCFCY, RVCFCZ, RVCFALPH
REAL*8 RVSPPIX, RVSPPIY, RVSPPIZ, RVCFET1, RVCFET2
REAL*8 RTCX A

COMMON /END/ RVCFXF, FVCX YF, RVCFZF, RTCX A
COMMON /MORE/ IVCXSX, IVCXSY, IVCXSZ, RVCFCX, RVCFCY, RVCFCZ
COMMON /ION01/ RVCFALPH, RVCFXBETA, RVCFET1, RVCFET2
COMMON /TEMP4/ RVCFXF1, RVCFXF2, RVCFXF3

IFSXG = 4 to flag an x-reflection & = 5 to flag a y-reflection.

IF (IFSXG.EQ.4) RVCFXF3 = RVCFXF1
IF (IFSXG.EQ.5) RVCFXF3 = RVCFXF2
IF (RVCFXBETA.NE.0.0D00) THEN
CALL PHSPL3 to calculate the Z phase path increment when beta is not equal to zero.
CALL PHSPL3 (RVCFXFZ, RVCFXF3, IVCXSZ, RVCFCXZ, RVCFALPH, RVCFXBETA)
ELSE IF (RVCFALPH.EQ.0.0D00) THEN
RVSXPPIZ = IVCXSZ * RVCFZ * DSQRT (RVCFXZ)
ELSE
Call PHSPL2 to calculate phase path increment for a direction in which the plasma frequency has a linear dependence.
CALL PHSPL2 (RVCFXF, IVCXSX, RVCFCX, RVCFALPH, RVSPPIZ)
ENDIF

IF (RVCFET1.EQ.0.0D00) THEN
RVSXPPIX = IVCXSX * RVCFXF * DSQRT (RVCFCX)
ELSE
CALL PHSPL2 (RVCFXF, IVCXSX, RVCFCX, RVCFET1, RVSPPIX)
ENDIF

IF (RVCFET2.EQ.0.0D00) THEN
RVSXPPIY = IVCXSY * RVCFYF * DSQRT (RVCFCY)
ELSE
CALL PHSPL2 (RVCFYF, IVCXSY, RVCFCY, RVCFET2, RVSPPIY)
ENDIF

RVSXPPI = RVSPPIX + RVSPPIY + RVSPPIZ
RETURN
END
<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXPPPI</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFSXG</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXPPPIX</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVSXPPPTY</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVSXPPPIZ</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCCCY</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXZ</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXALPH</td>
<td>IONO1</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXET1</td>
<td>IONO1</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXET2</td>
<td>IONO1</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>END</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>IVCCSX</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCCSY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCCSZ</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCCXF1</td>
<td>TEMP4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXF2</td>
<td>TEMP4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXF3</td>
<td>TEMP4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXBETA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCCXF</td>
<td>END</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXV</td>
<td>END</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXZ</td>
<td>END</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCCXC</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE PHSPL3(RVSxZ5,RVSxF3,IVSxSZ,RVSxCZ,RVSxALPH
#,RVSxBETA,IFSxG,RVSxPPIZ)

C--
C-- PHSPL3 -- SUBROUTINE TO CALCULATE THE Z INCREMENT OF
PHASE PATH LENGTH FOR BETA EQUALS ZERO.

C CALLED BY: PHSPL

C------------------------------------------------------------------
C
AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 09/30/37
VERSION: 1.0
REVISED: 09/30/87 -- V1.0. Initial revision.

USES:
RVSxZ5 SEZ z value at end of RPI.
RVSxF3 Complete group path length increment.
IVSxSZ Propagation parameters.
RVSxCZ
RVSxALPH
RVSxBETA
IFSxG A flag variable.

To calculate the increment of phase path length in the
z-direction when the z dependence of the plasma freq.
is quadratic (i.e. beta.NE.0).

RETURNS: RVSxPPIZ The z increment of phase path length.

INTEGER IVSxSZ,IFSxG
467 REAL*8 RVSXZ5, RVSXF3, RVSXCZ, RVSXALPH, RVSXBETA
468 REAL*8 RVSXPIZ, RVLXR2, RVLXDEL
469 C
470 IF (IFSXG.EQ.3) THEN
471 RVLXR2 = 0.0D00
472 ELSE
473 RVLXR2 = DSQRT(RVSXCZ - RVSXALPH*RVSXZ5 + RVSXBETA*#
474 *RVSXZ5*RVSXZ5)
475 ENDIF
476 RVLXDEL = 4.0D00 * RVSXBETA * RVSXCZ - RVSXALPH * RVSXALPH
477 RVSXPIZ = (2.0D00 * RVSXBETA * RVSXZ5 - RVSXALPH) * RVLXR2
478 RVSXPIZ = RVSXPIZ + RVSXALPH * DSQRT(RVSXCZ)
479 RVSXPIZ = (IVSXSZ * RVSXPIZ + RVLXDEL * RVSXF3/2.0D00)
480 RVSXPIZ = RVSXPIZ / (4.0D00 * RVSXBETA)
481 RETURN
482 END

PHSPL3 Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXPIZ</td>
<td>param</td>
<td>REAL*8</td>
<td></td>
</tr>
<tr>
<td>IFSXG</td>
<td>param</td>
<td>REAL*8</td>
<td></td>
</tr>
<tr>
<td>RVSXBETA</td>
<td>param</td>
<td>REAL*8</td>
<td></td>
</tr>
<tr>
<td>RVSXALPH</td>
<td>param</td>
<td>REAL*8</td>
<td></td>
</tr>
<tr>
<td>RVSXCZ</td>
<td>param</td>
<td>REAL*8</td>
<td></td>
</tr>
<tr>
<td>IVSXSZ</td>
<td>param</td>
<td>REAL*8</td>
<td></td>
</tr>
<tr>
<td>RVSXF3</td>
<td>param</td>
<td>REAL*8</td>
<td></td>
</tr>
<tr>
<td>RVSXZ5</td>
<td>param</td>
<td>REAL*8</td>
<td></td>
</tr>
<tr>
<td>RVLXR2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXDEL</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>

227
SUBROUTINE PHSPL2(RVSXW, IVSX, RVSXC, RVSXN, RVSXPPIW)

PHSPL2 -- SUBROUTINE TO CALCULATE THE PHASE PATH LENGTH INCREMENT FOR LINEAR PLASMA FREQUENCY PATHS.

CALLED BY: PHSPL

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/30/37

VERSION: 1.0

REVISED: 09/30/87 -- V1.0. Initial revision.

USES:
RVSXW The generalized SEZ coord. value at the end of the RPI.
IVSX The generalized propagation parameters.
RVSXC
RVSXN

To calculate the phase path length increment for the particular coordinate direction of interest. The PPI is calculated for a plasma frequency that is linearly dependent on the coordinate involved. This translates to having \( N = N_x \) or \( N_y \), or if \( B = 2 \), \( N = A \). The same routine can then be used in any of the coordinate directions.

RETURNS:
RVSXPPIW The phase path increment for the particular coordinate direction.
Line# Source Line
535 \[ RVLxR = RVSxR - RVSxN * RVSxW \]
536 IF (RVLxR.GT.0.0D00) THEN
537 \[ RVLxR2 = -RVLxR * DSQRT(RVLxR) \]
538 ELSE
539 \[ RVLxR2 = 0.0D00 \]
540 ENDIF
541 RVSxPPIW = 2.0D00 * RVSxS * (RVLxR2 + RVSxS * DSQRT(RVSxS))
542 RVSxPPIW = RVSxPPIW / 3.0D00 * RVSxN
543 RETURN
544 END

PHSPL2 Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXPPPIW</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXN</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXSC</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXSC</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXSC</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVLX2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXR</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>

Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCFSP</td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>END</td>
<td>common</td>
<td>***</td>
<td>32</td>
</tr>
<tr>
<td>ENDPT</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>INTSIGN</td>
<td>extern</td>
<td>INTEGER*4</td>
<td>***</td>
</tr>
<tr>
<td>ION1</td>
<td>common</td>
<td>***</td>
<td>12</td>
</tr>
<tr>
<td>MORE</td>
<td>common</td>
<td>***</td>
<td>16</td>
</tr>
<tr>
<td>OTHER</td>
<td>common</td>
<td>***</td>
<td>24</td>
</tr>
<tr>
<td>PHSPL</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>PHSPL2</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>PHSPL3</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>PRAM</td>
<td>common</td>
<td>***</td>
<td>32</td>
</tr>
<tr>
<td>TEMP4</td>
<td>common</td>
<td>***</td>
<td>24</td>
</tr>
</tbody>
</table>

Code size = 155e (5470)
Data size = 0044 (68)
SUBROUTINE TILTS(RVSxDXHB, RVSxDYHB, IVSxJ)

TILTS -- SUBROUTINE TO CALCULATE THE IONOSPHERIC BOUNDARY TILTS
CALLED BY: RAYSUB

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 09/30/87
VERSION: 2.0

REVISED: 07/25/86 -- INITIAL REVISION. TRANSLATED FROM TEKTRONIX BASIC TO VAX FORTRAN BY ERIC L. STROBEL.
07/30/86 -- V1.1. Change over to use of REAL*3 precision in the calculations.
09/30/87 -- V2.0. Extensively modified to accommodate the use of RADAR-C. This routine now serves as a dispatcher for the real processing.

USES: IVSxJ A boundary index.
IVCxCASE 
RVCxBND Height of the upcoming boundary.
IFCxSN A flag parameter needed in one of the cases.

To decide which of the tilt calculation routines is to be used, if at all. The J-index values have different meanings depending on which CASE applies.

RETURNS: RVSxDXHB \( \frac{d}{dx} \text{Boundary Height} \)
RVSxDYHB \( \frac{d}{dy} \text{Boundary Height} \)
51 C-----------------------------------------------
52 C
53 C INTEGER IVSXJ, JVCXCASE, IFCXSN
54 C
55 C REAL*8 RVSXDHB, RVSXDYHB, RPCXPI, RPCXDR, RPCXRE
56 C REAL*8 RVCXV1, RVCXV2, RPCXHTP, RVCXXX, RVCXHBND
57 C
58 C COMMON /PRAM/ RPCXPI, RPCXDR, RPCXRE, RPCXHTP
59 C COMMON /IONO3/ RVCXV1, RVCXV2, RVCXXX, JVCXCASE, RVCXHBND
60 C
61 C
62 C
63 C
64 C
65 C
66 C
67 C
68 C
69 C
70 C
71 C
72 C
73 ELSE IF (IVCXCASE.EQ.2) THEN
74 IF (IVSXJ.EQ.5) CALL H1PTLT (RVSXDHB, RVSXDRH)
75 IF (IVSXJ.EQ.6) CALL H2PTLT (RVSXDHB, RVSXDRH)
76 IF (IVSXJ.EQ.7) CALL H3PTLT (RVSXDHB, RVSXDRH)
77 ELSE IF (IVCXCASE.EQ.3) THEN
78 IF (IVSXJ.EQ.5) CALL H1LTTLT (RVSXDHB, RVSXDRH)
79 IF (IVSXJ.EQ.6) CALL H2LTTLT (RVSXDHB, RVSXDRH)
80 IF (IVSXJ.EQ.7) CALL H3LTTLT (RVSXDHB, RVSXDRH)
81 ELSE IF (IVCXCASE.EQ.4) THEN
82 IF (IVSXJ.EQ.5) CALL H1TLTT (RVSXDHB, RVSXDRH)
83 IF (IVSXJ.EQ.6) CALL H2TLTT (RVSXDHB, RVSXDRH)
84 IF (IVSXJ.EQ.7) CALL H3TLTT (RVSXDHB, RVSXDRH)
85 IF (IVSXJ.EQ.8) CALL H4TLTT (RVSXDHB, RVSXDRH)
86 ELSE IF (IVCXCASE.EQ.5) THEN
87 IF (IVSXJ.EQ.5) CALL H1TLT (RVSXDHB, RVSXDRH)
88 IF (IVSXJ.EQ.6) CALL H2TLT (RVSXDHB, RVSXDRH)
89 IF (IVSXJ.EQ.7) CALL H3TLT (RVSXDHB, RVSXDRH)
90 IF (IVSXJ.EQ.8) CALL H4TLT (RVSXDHB, RVSXDRH)
91 ENDIF
92 RETURN
93 END

TILTS Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVSXJ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Type</td>
<td>Size</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>--------------</td>
<td>------</td>
</tr>
<tr>
<td>RVSXDYHB</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXDXXHB</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVCSCASE</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IFCSN</td>
<td>RAID</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXDR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXRE</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXHTP</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXXX</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXHBND</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE HUTLT(RVSxDXHB, RVSxDYHB)

HUTLT -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR VALLEY-F2 BOUNDARY

CALLED BY: TILTS

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/30/87

VERSION: 1.0

REVISED: 09/30/87 -- V1.0. Initial revision.

USES: The parameters from the IONO1.2.3 blocks.

To calculate the rate of change of the boundary in the locally horizontal directions. The boundary here is the boundary between the linear valley profile and the F2 parabolic profile (for those cases where the intersection is visible on the final profile). For further details, see the RADAR-C report referred to in the documentation.

RETURNS: RVSxDXHB (d/dx) Height of Boundary.

RVSxDYHB (d/dy) Height of Boundary.

INTEGER IVCxCASE

REAL*8 RVSxDXHB, RVSxDYHB, RTLxD, RVCxHB(3), RVCxBND

REAL*8 RVLxHUY2, RVLxHUXE, RVCxFB(3), RVLxHU2, RVLxHUH1

REAL*8 RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxE6(3)

REAL*8 RVCxA5(3), RVCxA6(3), RVCxV1, RVCxV2, RVCxXX

REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2

COMMON /IONO1/ RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVCxFB, RVCxHB
COMMON /I0N02/ RVCxA5, RVCxA6, RVCxB5, RVCxB6, RVCxE5, RVCxE6
COMMON /ION03/ RVCxV1, RVCxV2, RVCxxx, IVCxCASE, RVCxHBND

C

RTLxD = (RVCxHB(2) - RVCxHBND) / RVCxHB(3)

RVlxHUXE = 0.9604D00 * RVCxHB(3) / (2.0D00 * RTLxD * RVCxFB(3))

RVlxHU2 = -RVlxHUXE * RVCxFB(1) / RVCxFB(3)

RVlxHUH2 = 1.0D00

RVsxDXHB = RVlxHUxE * RVCxB5(1) + RVlxHU2 * RVCxB5(3)

RVsxDXHB = RVsxDXHB + RVlxHuH2 * RVCxB6(2) + RVlxHU2 * RVCxB6(3)

RVsxDYHB = -RVsxDXHB / RVCxV1

RVsxDYHB = RVsxDXHB * RVlxHU2 * RVCxE5(1) + RVlxHU2 * RVCxE5(3)

RVsxDYHB = RVsxDYHB + RVlxHUH2 * RVCxE6(2) + RVlxHU2 * RVCxE6(3)

RETURN

END

HUTLT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVsxDYHB</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVsxDXHB</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVlxHU2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLxD</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVlxHU2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVlxHU2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVlxHU2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IVCxCASE</td>
<td>ION03</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCxHB</td>
<td>ION01</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxHBND</td>
<td>ION03</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxFB</td>
<td>ION01</td>
<td>REAL*3</td>
<td>24</td>
</tr>
<tr>
<td>RVCxB5</td>
<td>ION02</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxB6</td>
<td>ION02</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxE5</td>
<td>ION02</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxE6</td>
<td>ION02</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxA5</td>
<td>ION02</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxA6</td>
<td>ION02</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxV1</td>
<td>ION03</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCxV2</td>
<td>ION03</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCxxx</td>
<td>ION03</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVxALPH</td>
<td>ION01</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVxBETA</td>
<td>ION01</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCxE1</td>
<td>ION01</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCxE2</td>
<td>ION01</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE H5TLT(RVSxDXHB, RVSxDYHB)

H5TLT -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
F2-TOPSIDE BOUNDARY

CALLED BY: TILTS

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/30/87

VERSION: 1.0

REVISED: 09/30/87 -- V1.0. Initial revision.

USES: The parameters from the ION02,3 blocks.

To calculate the rate of change of the boundary in the
locally horizontal directions. The boundary here is
the boundary between the parabolic F2 profile and the
Bent topside profile. For further details, see the
RADAR-C report referred to in the documentation.

RETURNS: RVSxDXHB (d/dx) Height of Boundary.

RVSxDYHB (d/dy) Height of Boundary.

INTEGER ITCxA

REAL*8 RVSxDXHB, RVSxDYHB, RVLxH5Y2, RVLxH5H2
REAL*8 RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxE6(3)
REAL*8 RVCxA5(3), RVCxA6(3), RVCxV1, RVCxV2
REAL*8 RTCxA, RTCxB
COMMON /ION02/ RVCxA5, RVCxA6, RVCxB5, RVCxB6, RVCxE5, RVCxE6
COMMON /ION03/ RVCxV1, RVCxV2, RTCxA, ITCxA, RTCxB

RVLxH5H2 = 1.0D00

235
Line# Source Line
213        RVLXH5Y2 = 0.25D00
214        RVSXDXHB = -(RVLXH5H2*RVCXB6(2) + RVLXH5Y2*RVCXB6(3)) / RVCXV2
215        RVSXDYHB = (RVLXH5H2*RVCXE6(2) + RVLXH5Y2*RVCXE6(3)) / RVCXV2
216        RETURN
217        END

HSTLT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXDYHB</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSXDXHB</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVLXH5H2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXH5Y2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>ITCXA</td>
<td></td>
<td>IONO3</td>
<td>4</td>
</tr>
<tr>
<td>RVCXB5</td>
<td>IONO2</td>
<td>REAL*9</td>
<td>24</td>
</tr>
<tr>
<td>RVCXB6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXE5</td>
<td>IONO2</td>
<td>REAL*9</td>
<td>24</td>
</tr>
<tr>
<td>RVCXE6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXA5</td>
<td>IONO2</td>
<td>REAL*9</td>
<td>24</td>
</tr>
<tr>
<td>RVCXA6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXV1</td>
<td>IONO3</td>
<td>REAL*9</td>
<td>3</td>
</tr>
<tr>
<td>RVCXV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTCXA</td>
<td>IONO3</td>
<td>REAL*9</td>
<td>8</td>
</tr>
<tr>
<td>RTCXB</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE HIPTLT(IVSxSGN°RVSxDXHB, RVSxDYHB)

C-----------------------------------------------
C        HIPTLT -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
C                      VALLEY-F1 (PARABOLIC) BOUNDARY
C-----------------------------------------------
C        CALLED BY: TILTS
C        CALLS: DH1PDP
C-----------------------------------------------
C        AUTHOR:  MICHAEL H. REILLY & ERIC L. STROBEL
C        DATE: 09/30/87
C        VERSION: 1.0
C-----------------------------------------------
C        REVISED: 09/30/87 -- V1.0. Initial revision.
C-----------------------------------------------
C        USES: The parameters from the COMMON blocks.
C-----------------------------------------------
C        To calculate the rate of change of the boundary in the
C               locally horizontal directions. The boundary here is
C               the boundary between the linear valley profile and the
C               parabolic F1 profile. The routine DH1PDP is just
C               d (HIP) / d (parameter). For further details, see the
C               RADAR-C report referred to in the documentation.
C-----------------------------------------------
C        RETURNS: RVSxDXHB  (d/dx) Height of Boundary.
C                    RVSxDYHB  (d/dy) Height of Boundary.
C-----------------------------------------------
C        INTEGER IVSxSGN, ITCxA
C-----------------------------------------------
C        REAL*8 RVSxDXHB, RVSxDYHB, RVLexD1, RVLexA1, RVLexB1, RVLexC1
C        REAL*8 RVLexBB, RVLexB0, RVLexHU, RVLexHL, RVLexHB(3), RVLexFB(3)
C        REAL*8 RVLexHUXE, RVLexB1XE, RVLexC1XE, RVLexA0, RVLexHPXE
C        REAL*8 RVLexA1X1, RVLexB1X1, RVLexHPX1, RVLexA1H1, RVLexB1H1
C        REAL*8 RVLexHPH1, RVLexHUX2, RVLexB1X2, RVLexC1X2, RVLexHPX2
C        REAL*8 RVLexB1H2, RVLexC1H2, RVLexHPH2, RVLexHYU2, RVLexB1Y2
REAL*8 RVLxClY2, RVLxHPy2, RVCxB5(3), RVCxB6(3), RVCxE5(3)
REAL*8 RVCxB6(3) RVCxV1, RVCxV2, RVCxALPH, RVCxBETA
REAL*8 RVCxET1, RVCxET2, RVCxAI(3), RVCxB6(3)
REAL*8 RVCxET0, RVCxET1, RVCxET2, RVCxAI(3), RVCxB6(3)
REAL*8 RTCXA, RTCXB, RTCXC, RTCXD

COMMON /IONO1/ RVCxAI(3), RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxV1, RVCxV2, RTCxA, RTCXB
COMMON /IONO2/ RVCxAI(3), RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxV1, RVCxV2, RTCxA, RTCXB
COMMON /IONO3/ RVCxAI(3), RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxV1, RVCxV2, RTCxA, RTCXB
COMMON /TEMPC/ RVCxF40, RVCxF65, RVCxKI, RVCxHL
COMMON /VAR1/ RVCxAI(3), RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxV1, RVCxV2
COMMON /VAR2/ RVCxAI(3), RVCxB5(3), RVCxB6(3), RVCxE5(3), RVCxV1, RVCxV2

RVLxClY1 = DSQR(RVCxB5(1)*RVCxB5(1) - 4.0D00*RVCxAI(1)*RVCxC1)
RVLxBBB = RVCxB0 / (RVCxHU - RVCxHL)
RVLxHUX1 = (0.98000 * RVCxHB(3) - (0.98000 * RVCxHB(3)))
RVLxHUX1 = RVLxHUX1 / (2.0D00 * RVCxHB(3) * (RVCxHB(2) - RVCxHU))
RVLxB1X2 = RVCxB0 = (-1.0D00 + 2.0D00 * RVCxHB(1) * RVLxHUX1) /
#(RVCxHU - RVCxHL) / RVCxFB(1)
RVLxC1X2 = RVCxA0/RVCxFB(1) + RVCxHL*RVLxBB*RVLxHUX1
CALL DH1PDP(0.0D00, RVLxB1X2, RVLxC1X2, RVLxHPU, RVLxDPX)

#IVSxSGN)
RVLxB1X1 = 16.0D00 / (RVCxHB(1) * RVCxHB(1))
RVLxB1X1 = 32.0D00 / (RVCxHB(1))
CALL DH1PDP(0.0D00, RVLxB1X1, RVLxB1X1, 15.0D00, RVLxHPX1, RVLxD1)
#IVSxSGN)
RVLxB1H1 = -32.0D00*RVCxFB(2) / (RVCxHB(1) * RVCxHB(1))
RVLxB1H1 = RVLxB1H1 * RVCxHB(1)
CALL DH1PDP(RVLxB1H1, RVLxB1H1, 0.0D00, RVLxHPH1, RVLxCL1)
#IVSxSGN)
RVLxHUX2 = -RVLxHUX1 * RVCxFB(1) / RVCxFB(1)
RVLxB1X2 = RVLxBB * RVLxHUX2
RVLxC1X2 = RVCxHL * RVLxB1X2
CALL DH1PDP(0.0D00, RVLxB1X2, RVLxC1X2, RVLxHPX2, RVLxD1)
#IVSxSGN)
RVLxB1H2 = RVLxBB
RVLxC1H2 = RVCxHL * RVLxBB
CALL DH1PDP(0.0D00, RVLxB1H2, RVLxC1H2, RVLxHPH2, RVLxD2)
#IVSxSGN)
RVLxHUY2 = -(RVCxHB(2) - RVCxHU) / RVCxHB(3)
RVLxB1Y2 = RVLxBB * RVLxHUY2
RVLxC1Y2 = RVCxHL * RVLxB1Y2
CALL DH1PDP(0.0D00, RVLxB1Y2, RVLxC1Y2, RVLxHPY2, RVLxC2)
#IVSxSGN)
RVSxDPX = RVLxHPX2*RVCxSB(1) + RVLxHPX1*RVCxSB(2) +
#RVLxHPX2*RVCxSB(3)
RVSxDPX = RVLxDPX2*RVCxSB(1) + RVLxHPX1*RVCxSB(2) + RVLxHPX2*RVCxSB(3)
++ RVLxHPY2*RVCxSB(3)
RVSxDPX = -RVSxDPX2 / RVCxV1
RVSxDPX = RVLxHPX2*RVCxSB(1) + RVLxHPX1*RVCxSB(2) +
RVSxDPX = RVLxHPX2*RVCxSB(3)
RVSxDPX = -RVSxDPX2 / RVCxV1
RVSxDPX = RVLxHPX2*RVCxSB(1) + RVLxHPX1*RVCxSB(2) +

238
Line#   Source line
320   #RVLxHPX2*RVCxE5(i)
321   RVSxDYHB = RVSxDYHB + RVLxHPH1*RVCxE6(1) + RVLxHPH2*RVCxE6(2)
322   #: RVLxHPY2*RVCxE6(3)
323   RVSxDYHB = RVSxDYHB / RVCxV2
324   RETURN
325   END

HiPTLT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXDYHB</td>
<td>.......</td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSXDXHB</td>
<td>.......</td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>VVSXSGN</td>
<td>.......</td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVLXHPX2E</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXD1</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXHUXE</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXAI1X1</td>
<td>.......</td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVLXBI1X1</td>
<td>.......</td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVLXBI1X2</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXBI1Y2</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXCI1X2</td>
<td>.......</td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVLXCI1Y2</td>
<td>.......</td>
<td>local</td>
<td>REAL*9</td>
</tr>
<tr>
<td>RVLXBB</td>
<td>.......</td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVLXBI1XE</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXHPH1</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXCI1XE</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXHPH2</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXHPX1</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXHPX2</td>
<td>.......</td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVLXHPY2</td>
<td>.......</td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVLXHUX2</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXHUY2</td>
<td>.......</td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVLXA1H1</td>
<td>.......</td>
<td>local</td>
<td>REAL*9</td>
</tr>
<tr>
<td>RVLXB1H1</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVLXB1H2</td>
<td>.......</td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVLXC1H2</td>
<td>.......</td>
<td>local</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVCXA0</td>
<td>.......</td>
<td>VAR1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXB5</td>
<td>.......</td>
<td>IONO2</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXB6</td>
<td>.......</td>
<td>IONO2</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXE5</td>
<td>.......</td>
<td>IONO2</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVCXE6</td>
<td>.......</td>
<td>IONO2</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXV1</td>
<td>.......</td>
<td>IONO3</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXV2</td>
<td>.......</td>
<td>IONO3</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXALPH</td>
<td>.......</td>
<td>IONO1</td>
<td>REAL*3</td>
</tr>
<tr>
<td>RVCXBETA</td>
<td>.......</td>
<td>IONO1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXET1</td>
<td>.......</td>
<td>IONO1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXET2</td>
<td>.......</td>
<td>IONO1</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCXA5</td>
<td>.......</td>
<td>IONO2</td>
<td>REAL*9</td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Type</td>
<td>Size</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>RVCPX6</td>
<td></td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>RVCPX40</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX65</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX1</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX2</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX3</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX4</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX5</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX6</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX7</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX8</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX9</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX10</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX11</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX12</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX13</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX14</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX15</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX16</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX17</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX18</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX19</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX20</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX21</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX22</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX23</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX24</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX25</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX26</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX27</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX28</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX29</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX30</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX31</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX32</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX33</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX34</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX35</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX36</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX37</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX38</td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>RVCPX39</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RVCPX40</td>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE DH1PDP(RVSxAIP, RVSxBIP, RVSxCIP, RVSxDD, RVSxDI, IVSxSGN)

C---------------------------------------------------------------
C DH1PDP -- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE
C BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.
C CALLED BY: H1PTLT
C---------------------------------------------------------------
C AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
C DATE: 09/30/87
C VERSION: 1.0
C---------------------------------------------------------------
C REVISED: 09/30/87 -- V1.0. Initial revision.
C---------------------------------------------------------------
C USES: RVSxA,B,CIP The partials of the parabolic profile parameters w.r.t.
C the parameter of interest.
C RVSxDD1 The discriminant of the parabolic parameters.
C IVSxSGN Determines which of the sol'n's. of the quadratic eqn. applies.
C---------------------------------------------------------------
C To calculate the partial derivative of H1P with respect to
C the chosen parameter. For further details, see the
C RADAR-C report referred to in the documentation.
C---------------------------------------------------------------
C RETURNS: RVSxDD The aforementioned partial derivative.
C---------------------------------------------------------------
C INTEGER IVSxSGN
C REAL*8 RVSxAIP, RVSxBIP, RVSxCIP, RVSxDD, RVSxDI, RVLxH1
C REAL*8 RVCxAL, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P

241
COMMON /VAR3/ RVCxA1, RVCxB1, RVCxC1, RVCxH1P, RVCxH2P

C

RVLxH1 = RVCxH1P
RVSxDD = (-RVLxH1 + IVSxSGN*RVCxC1/RVSxDD) * RVSxA1P
RVSxDD = RVSxDD + (1.0000 - IVSxSGN*RVCxB1/RVSxDD)
RVSxDD = (*RVSxB1P/2.0000)
RVSxDD = (RVSxDD + IVSxSGN*(RVCxA1/RVSxD1) * RVSxC1P)

RETURN

END

+H1PD+P Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVSxSGN</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSxD1</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSxDD</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSxC1P</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSxB1P</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSxA1P</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVLxH1</td>
<td></td>
<td>local</td>
<td>REAL*8</td>
</tr>
<tr>
<td>RVCxA1</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCxB1</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCxC1</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCxH1P</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCxH2P</td>
<td>VAR3</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
SUBROUTINE H4TLT(RVSxSN, RVSxDXHB, RVSxDYHB)

-- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR
PARABOLIC F1-F2 BOUNDARY

CALLED BY: TILTS
CALLS: DH4DP

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 09.30.37
VERSION: 1.0

REVISED: 09/30/37 -- V1.0. Initial revision.

USES: The parameters from the COMMON blocks.

To calculate the rate of change of the boundary in the
locally horizontal directions. The boundary here is
the boundary between the parabolic F1 profile and the
parabolic F2 profile. The routine DH4DP is just
d (H4) / d (parameter). For further details, see the
RADAR-C report referred to in the documentation.

RETURNS: RVSxDXHB (d/dx) Height of Boundary.
RVSxDYHB (d/dy) Height of Boundary.

INTEGER IVSxSN, ITCxA

REAL*8 RVSxDXHB, RVSxDYHB, RVLxD2, RVCxA2, RVCxB2, RVCx12
REAL*8 RVCxHT4, RVCxHB (3), RVCxFB (3), RVLxSB1, RVCxHT1
REAL*8 RVLxB2B1, RVLxA2X1, RVLxB2X1, RVLxH4X1, RVLxA2X2
REAL*8 RVLxB2X2, RVLxC2X2, RVLxH4X2, RVLxA2H1, RVLxB2H1
REAL*8 RVLxH4H1, RVLxB2H2, RVLxC2H2, RVLxH4H2, RVLxA2Y2
REAL*8 RVLxB2Y2, RVLxC2Y2, RVLxH4Y2, RVCxBS(3), RVCxB6(3)
C REAL*S RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxALPH
C REAL*S RVCxBETA, RVCxET1, RVCxET2, RVCxA5(3), RVCxA6(3).
C REAL*S RTCxA, RTCxB

COMMON /IONO/ RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVCxFB, RVCxHB
COMMON /IONO2/ RVCxA5, RVCxA6, RVCxB5, RVCxB6, RVCxE5, RVCxE6
COMMON /IONO3/ RVCxV1, RVCxV2, RTCxA, RTCxB
COMMON /VAR4/ RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4

C RVLxD2 = C0.5(RVCxB2*RVCxB2 - 4.0000*RVCxA2*RVCxC2) + IVSXSN

*INTSIGN(RVCxB2)
RVCxHT4 = -(RVCxB2 + RVxLD2) / (2.0000 + RVCxA2)
RVLxBB1 = 15.0000 / (RVCxHB(1) + RVCxHB(1));
RVLxBB2 = 1.0000 / (RVCxHB(3) + RVCxHB(3));
RVLxA2X1 = -RVLA2X1;
RVLxB2X1 = 32.0000 / (RVCxHB(1) + RVCxHB(1));
CALL DH4DP(RVLxA2X1, RVLxB2X1, -15.0000, RVLxH4X1, RVLxLD2);
RVLxA2H1 = 2.0000 / (RVCxB1 + RVCxB1) / (RVCxHB(1) + RVCxHB(1));
RVLxB2H1 = -RVLA2H1 * RVCxHB(1);
CALL DH4DP(RVLxA2H1, RVLxB2H1, 0.0000, RVLxH4H1, RVLxLD2);
RVLxA2X2 = RVLxB2B2;
RVLxB2X2 = -2.0000 + RVCxHB(2) + RVCxB2;
RVLxC2X2 = (-RVLxB2X2 + RVCxHB(3) * RVCxHB(3)) + 1.0000;
CALL DH4DP(RVLxA2X2, RVLxB2X2, RVLxC2X2, RVLxH4X2, RVLxLD2);
RVLxA2H2 = -2.0000 + RVCxHB(3) + RVCxB2;
RVLxB2H2 = -RVLA2H2 * RVCxHB(2);
CALL DH4DP(0.0000, RVLxB2H2, RVLxC2H2, RVLxH4H2, RVLxLD2);
RVLxA2Y2 = RVLxB2H2 / (RVCxHB(3) + RVCxHB(3));
RVLxB2Y2 = -2.0000 + RVLxA2Y2 + RVCxHB(2);
RVLxC2Y2 = -RVLxA2Y2 * RVCxHB(2) + RVCxHB(2);
CALL DH4DP(RVLxA2Y2, RVLxB2Y2, RVLxC2Y2, RVLxH4Y2, RVLxLD2);
RVSxDXHB = RVLxH4X1 + RVCxB5(2) + RVLxH4X2 + RVCxB5(3) +
#RVLxH4H1*RVCxB6(1)
RVSxDYHB = RVLxH4H1 + RVCxB6(2) + RVLxH4Y2 + RVCxB6(3) +
RVSxDXHB = RVLxH4H1 + RVCxB6(1) + RVLxH4Y2 + RVCxB6(3) +
RVSxDYHB = RVSxDYHB + RVLxH4H1 + RVCxB6(2) + RVLxH4Y2 + RVCxB6(3) +
RETURN
END

H4TLT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXDYHB</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>RVSXDHB</td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td>IVSXSN</td>
<td></td>
<td>param</td>
<td></td>
</tr>
</tbody>
</table>
### H4TLT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVLXH4H1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXBB1</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXH4H2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXBB2</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXD2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXA2X1</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXA2X2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXB2X1</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXA2Y2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXB2Y2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXC2X2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXC2Y2</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXH4X1</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXH4X2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH4Y2</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLX2A2H1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXB2H1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXB2H2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLX4C2H2</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>ITXCA</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCXA2</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCXB2</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCXC2</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCXH5</td>
<td>VAR4</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCXH6</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>14</td>
</tr>
<tr>
<td>RVCXH7</td>
<td>IONO1</td>
<td>REAL*3</td>
<td>24</td>
</tr>
<tr>
<td>RVCXH3</td>
<td>VAR4</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCX5</td>
<td>IONO2</td>
<td>REAL*3</td>
<td>24</td>
</tr>
<tr>
<td>RVCX6</td>
<td>IONO2</td>
<td>REAL*3</td>
<td>24</td>
</tr>
<tr>
<td>RVCX5E3</td>
<td>IONO2</td>
<td>REAL*3</td>
<td>24</td>
</tr>
<tr>
<td>RVCX6E6</td>
<td>IONO2</td>
<td>REAL*3</td>
<td>24</td>
</tr>
<tr>
<td>RVCXV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCV2</td>
<td>IONO3</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RVCXALPH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXET1</td>
<td>IONO1</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCXET2</td>
<td>IONO1</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCXAS</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCXAS6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RTCXA</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>IONO3</td>
<td>REAL*3</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE DH4DP(RVSxA2D, RVSxB2D, RVSxC2D, RVSxH4D, RVSxD2)

-- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.

CALLED BY: H4TLT

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 09/30/87
VERSION: 1.0
REVISED: 09/30/87 -- V1.0, Initial revision.

USES: RVSxA,B,C2D The partials of the parabolic profile parameters w.r.t. the parameter of interest.
RVSxD2 The discriminant of the parabolic parameters.

To calculate the partial derivative of H4 with respect to the chosen parameter. For further details, see the RADAR-C report referred to in the documentation.

RETURNS: RVSxH4D The aforementioned partial derivative.

REAL*8 RVSxA2D, RVSxB2D, RVSxC2D, RVSxH4D, RVSxD2, RVCxHT4
REAL*8 RVCxA2, RVCxB2, RVCxC2, RVCxHT3
COMMON /VAR4/ RVCxA2, RVCxB2, RVCxC2, RVCxHT3, RVCxHT4

RVSxH4D = (-RVCxHT4 + RVCxC2/RVSxD2) * RVSxA2D
RVSxH4D = RVSxH4D - (1.0D00 + RVCxB2/RVSxD2) * RVSxB2D/2.0D00
RVSxH4D = (RVSxH4D - (RVCxA2/RVSxD2) * RVSxC2D) / RVCxA2
Line#    Source Line
533           RETURN
534           END

DH4DP  Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXD2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXH4D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXC2D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXB2D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXA2D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVCXC2</td>
<td></td>
<td>VAR4</td>
<td></td>
</tr>
<tr>
<td>RVCXHT3</td>
<td></td>
<td>VAR4</td>
<td></td>
</tr>
<tr>
<td>RVCXHT4</td>
<td></td>
<td>VAR4</td>
<td></td>
</tr>
<tr>
<td>RVCXA2</td>
<td></td>
<td>VAR4</td>
<td></td>
</tr>
<tr>
<td>RVCXB2</td>
<td></td>
<td>VAR4</td>
<td></td>
</tr>
</tbody>
</table>

247
SUBROUTINE HILTLT(RVSxDXHB, RVSxDYHB)

HILTLT -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR LINEAR VALLEY - LINEAR F1 BOUNDARY

CALLED BY: TILTS
CALLS: DHILDP

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL
DATE: 09/30/37
VERSION: 1.0
REVISED: 09/30/37 -- V1.0. Initial revision.

USES: The parameters from the COMMON blocks.

To calculate the rate of change of the boundary in the locally horizontal directions. The boundary here is the boundary between the linear valley profile and the linear F1 profile. The routine DHILDP is just \( \frac{d}{d} \) (HIL) / \( \frac{d}{d} \) (parameter). For further details, see the RADAR-C report referred to in the documentation.

RETURNS: RVSxDXHB \( \frac{d}{d} \) x Height of Boundary.
RVSxDYHB \( \frac{d}{d} \) y Height of Boundary.

INTEGER ITCxA

REAL*8 RVSxDXHB, RVSxDYHB, RVLxB2I, RVCxB0, RVCxHU, RVCxH2
REAL*8 RVLxHUXE, RVCxHB(3), RVCxFB(3), RVLxYSX1, RVCxH2
REAL*8 RVLxB2, RVCxYS, RVLxAQXE, RVCxAO, RVLxBOXZ, RVLxH1X2
REAL*8 RVLxS1X1, RVLxH1X1, RVLxS1H1, RVLxH1H1, RVLxHUX2
REAL*8 RVLxAQX2, RVLxBOX2, RVLxS1X2, RVLxH1X2, RVLxAQH2
REAL*8 RVLxBOH2, RVLxS1H2, RVLxH1H2, RVLxHUY2, RVLxAQY2

248
REAL*8 RVLxBOY2, RVLxS1Y2, RVLxH1Y2, RVCxB5(3), RVCxB6(3)
REAL*8 RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2, RVCxALPH
REAL*8 RVCxETA, RVCxET2, RVCxA5(3), RVCxA6(3)
REAL*S RVCxBETA, RVCxET1, RVCxF65, RVCxK1, RVCxXL, RVCxXU, RVCxHB1
REAL*8 RTCxA, RTCxB, RTCxC, RTCxD, RTCxE
COMMON /ION01/ RVCxALPH, RVCxETA, RVCxET2, RVCxFB, RVCxHB
COMMON /ION02/ RVCxA5, RVCxA6, RVCxB5, RVCxB6, RVCxE5, RVCxE6
COMMON /ION03/ RVCxV1, RVCxV2, RTCxA, RTCxB
COMMON /TEMP1/ RVCxF40, RVCxF65, RVCxK1, RVCxXL
COMMON /VAR1/ RVCxXL, RVCxXU, RVCxAO, RVCxB0
COMMON /VAR2/ RVCxHB1, RVCxH2, RVCxB0, RVCxBO
RVLxBBI = RVCxB0 / (RVCxHU - RVCxHL)
RVLxHUXE = 0.9604000 * RVCxHB(3) - RVCxHB(3) / (2.0000 * RVCxB(3))
RVLxYSX1 = RVCxHB(3) * RVCxHB(3) / (2.0000 * RVCxFB(3))
RVLxBB2 = RVCxFB(2) / (RVCxYS * RVCxYS)
RVLxA0XE = RVCxAO/RVCxFB(1) + RVCxHL*RVLxBBI*RVLxHUXE
RVLxBOX = RVCxB0/RVCxFB(1) - RVLxBBI*RVLxHUXE
CALL DH1LDP(RVLxA0XE, 0.0000, RVLxBOX, 0.0000, RVLxHUXE)
RVLxS1X1 = 0.0000 / RVCxYS
IF (RVCxYS.NE.1.0000) RVLxS1X1 = RVLxS1X1 - RVLxBB2*RVLxY;X1
CALL DH1LDP(0.0000, 0.0000, 0.0000, RVLxS1X1, RVLxH1X1)
RVLxS1H1 = 0.0000
IF (RVCxYS.NE.1.0000) RVLxS1H1 = 0.75000 * RVLxBB2
CALL DH1LDP(0.0000, 0.0000, 0.0000, RVLxS1H1, RVLxH1H1)
RVLxHUX2 = -RVLxHUXE * RVCxFB(1) / RVCxFB(3)
RVLxA0X2 = RVCxHL * RVLxBB1 * RVLxHUX2
RVLxBOX2 = -RVLxBB1 * RVLxHUX2
RVLxBSX2 = 0.0000
IF (RVCxYS.NE.1.0000) RVLxS2X2 = RVLxBB2*RVLxYSX1*RVCxFB(2)
RVLxAOY2 = RVCxB0/RVCxB5(1) + RVLxH1X1*RVCxB5(3)
RVLxBB2 = RVCxB2/RVCxB5(3)
RVLxS1Y2 = 0.0000
IF (RVCxYS.NE.1.0000) RVLxS1Y2 = RVLxBB2*(RVCxHB(2) - RVCxHL)
CALL DH1LDP(RVLxA0Y2, 0.0000, RVLxBB2, RVLxS1Y2, RVLxH1Y2)
RVLxHXHB = RVLxH1XE*RVCxB5(1) + RVLxH1X1*RVCxB5(2)
RVLxH1X2 = RVLxH1X2*RVCxB5(3)
RVLxHXHB = RVLxH1X2*RVCxB5(3)
Line# Source Line
637          #= RVLxH1Y2•RVCxB6(3)
638          RVSxDXHB = -RVSxDXHB / RVCxV1
639          RVSxDYHB = RVLxH1X•RVCxE5(1) + RVLxH1X•RVCxE5(2)
640          #= RVLxH1X2•RVCxE5(3)
641          RVSxDYHB = RVSxDYHB + RVLxH1H1•RVCxE6(1) + RVLxH1H2•RVCxE6(2)
642          #= RVLxH1Y2•RVCxE6(3)
643          RVSxDYHB = RVSxDYHB / RVCxV2
644          RETURN
645          END

HILTLT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXDYHB.</td>
<td>param</td>
<td>REAL•8</td>
<td>3</td>
</tr>
<tr>
<td>RVSXDXB.</td>
<td>param</td>
<td>REAL•8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXYSX1.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXBB1.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXBB2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXHUXE.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXAOX2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXAOY2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXS1H1.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXS1H2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXHX1X.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH1X2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH1Y2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH1Y1.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXSI1X1.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXSI1X2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXAOXE.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXS1Y2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXBOXE.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH1XE.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXHUX2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXHUY2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXAOH2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXB0H2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH1H1.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH1H2.</td>
<td>local</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>TCEX</td>
<td>VAR2</td>
<td>REAL•8</td>
<td>8</td>
</tr>
<tr>
<td>TCEA</td>
<td>IONO3</td>
<td>INTEGER•4</td>
<td>4</td>
</tr>
<tr>
<td>VCXB0.</td>
<td>VAR1</td>
<td>REAL•8</td>
<td>3</td>
</tr>
<tr>
<td>VCXHU.</td>
<td>VAR1</td>
<td>REAL•8</td>
<td>3</td>
</tr>
<tr>
<td>VCXHL.</td>
<td>TEMP1</td>
<td>REAL•8</td>
<td>3</td>
</tr>
<tr>
<td>VCXHB.</td>
<td>IONO1</td>
<td>REAL•8</td>
<td>24</td>
</tr>
<tr>
<td>VCXFB.</td>
<td>IONO1</td>
<td>REAL•8</td>
<td>24</td>
</tr>
<tr>
<td>VCXH2.</td>
<td>VAR2</td>
<td>REAL•8</td>
<td>3</td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Type</td>
<td>Size</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>RV CXYS</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXA0</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXB5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RV CXB6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RV CXE5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RV CXE6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RV CXV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXALPH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXBETA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXET1</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXET2</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXA5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RV CXA6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RV CXF40</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RV CXF65</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RV CXK1</td>
<td>TEMP1</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RV CXXL</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXUX</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RV CXH1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXC</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXD</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE DHILDPRVSxAOP RVSxHID. RVSxBOP, RVSxSIP, RVSxDD)

---

DHILDPRVSxAOP -- SUBROUTINE TO CALCULATE THE DERIVATIVE OF THE BOUNDARY HEIGHT W.R.T. A PARTICULAR PARAMETER.

---

CALLED BY: HILTLT

---

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

---

DATE: 09/30/87

---

VERSION: 1.0

---

REVISED: 09/30/87 -- V1.0. Initial revision.

---

REAL*8 RVSxAOP, RVSxHID, RVSxBOP, RVSxSIP, RVSxDD

---

REAL*8 RVCxSL1, RVCxB0, RVCxHB1, RVCxA0, RVCxXL, RVCxX1

---

REAL*8 RVCxHU, RVCxH2, RVCxYS, RTCxA, RTCxB

---

COMMON /VAR1/ RVCxXL, RVCxXU, RVCxHU, RVCxA0, RVCxB0

---

COMMON /VAR2/ RVCxHB1, RVCxH2, RVCxYS, RVCxSL1, RTCxA, RTCxB

---

REAL*8 RVSxHID.SIP The partials of the linear profile parameters w.r.t. the parameter of interest.

---

RVSxDD The aforementioned partial derivative.
698  \( \text{RTLx1} = \text{RVCxSL}1 - \text{RVCxBO} \)
699  \( \text{RVSxDD} = \text{RTLx1} \times (\text{RVSxAOP} + 0.75 \times \text{RVCxSL}1 \times \text{RVSxHID}) \)
700  \( \text{RVSxDD} = \text{RVSxDD} + (\text{RVCxHB1} \times \text{RVCxSL}1 + \text{RVCxA0}) \times \text{RVSxBOP} \)
701  \( \text{RVSxDD} = (\text{RVSxDD} - (\text{RVCxHB1} \times \text{RVCxBO} + \text{RVCxA0}) \times \text{RVSxS1P}) \)
702  \#(\text{RTLx1} = \text{RTLx1})
703  \text{RETURN}
704  \text{END}

DHILDP Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXDD</td>
<td>param</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVSXS1P</td>
<td>param</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVSXBOP</td>
<td>param</td>
<td>REAL 8</td>
<td>3</td>
</tr>
<tr>
<td>RVSXH1D</td>
<td>param</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVSXAO</td>
<td>param</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLX1</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RVCXSL1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXBO</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXHB1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXA0</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXXL</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXXU</td>
<td>VAR1</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCXHU</td>
<td>VAR1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXH2</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXYS</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE H2TLT(RVSxDXHB, RVSxDYHB)

H2TLT -- SUBROUTINE TO CALCULATE BOUNDARY TILT FOR LINEAR F1 - F2 BOUNDARY

CALLED BY: TILTS

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/10/87

VERSION: 1.0

REVISED: 09/30/87 -- V1.0. Initial revision.

USES: The parameters from the COMMON blocks.

USES:

To calculate the rate of change of the boundary in the locally horizontal directions. The boundary here is the boundary between the linear F1 profile and the parabolic F2 profile. For further details, see the RADAR-C report referred to in the documentation.

RETURNS:

RVSxDXHB : (d/dx) Height of Boundary.
RVSxDYHB : (d/dy) Height of Boundary.

INTEGER ITCxA

REAL*8 RVSxDXHB, RVSxDYHB, RVLxH2X2, RVCxB(3), RVCxFB(3)
REAL*8 RVCxH2, RVLxH2X2, RVLxH2H2, RVLxH2Y2, RVCxB5(3)
REAL*8 RVCxB6(3), RVCxE5(3), RVCxE6(3), RVCxV1, RVCxV2
REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVCxA5(3)
REAL*8 RVCxA6(3), RVCxHB1, RTCxA, RTCxB, RTCxC, RTCxF
REAL*8 RTCxE, RTCxF

COMMON /ION01/ RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVCxFB, RVCxEB
COMMON /ION02/ RVCxA5, RVCxA6, RVCxB5, RVCxB6, RVCxE5, RVCxE6

254
COMMON /IONO3/ RVCxV1, RVCxV2, RTCXA, RTCXA, RTCXB
COMMON /VAR2/ RVCxHB1, RVCxH2, RTCXC, RTCXD, RTCXE, RTCXF

C

RVLxH2X1 = (RVCxHB(3) * RVCxHB(3)) / (2.0D00 * RVCxFB(3)) *

#(RVCxHB(2) - RVCxH2); RVLxH2X2 = -RVLxH2X1 * RVCxFB(2) / RVCxFB(3)
RVLxH2H2 = 1.0D00
RVLxH2Y2 = -RVLxH2X1 * RVCxB5(2) - RVLxH2X2 * RVCxB5(3)
RVSxDXHB = RVSxDXHB + RVLxH2H2 * RVCxB6(2) + RVLxH2Y2 * RVCxB6(3)
RVSxDXHB = -RVSxDXHB / RVCxV1
RVSxDYHB = RVLxH2X1 * RVCxE5(2) - RVLxH2X2 * RVCxE5(3)
RVSxDYHB = RVSxDYHB + RVLxH2H2 * RVCxE6(2) + RVLxH2Y2 * RVCxE6(3)
RETURN

H2TLT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXDYHB</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXDXHB</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVLXH2H2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH2X1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXH2X2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVLXH2Y2</td>
<td>local</td>
<td>REAL*8</td>
<td>9</td>
</tr>
<tr>
<td>RTCXA</td>
<td>IONO3</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCxHB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxFB</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxH2</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCxB5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxB6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxEB</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxE6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxV1</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCxV2</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCxALPH</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCxBTA</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCxET1</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCxET2</td>
<td>IONO1</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCxA5</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxA6</td>
<td>IONO2</td>
<td>REAL*8</td>
<td>24</td>
</tr>
<tr>
<td>RVCxHB1</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>IONO3</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXC</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCD</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTCXE</td>
<td>VAR2</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
### H2TLT Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCXF</td>
<td></td>
<td>VAR2</td>
<td>REAL*8</td>
</tr>
</tbody>
</table>

### Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DH1LDP</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>DH1PDP</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>DH4DP</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>H1TLT</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>H1PTLT</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>H2TLT</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>H4TLT</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>H5TLT</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>HUTLT</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>INTSIGN</td>
<td></td>
<td>extern</td>
<td>INTEGER*4</td>
</tr>
<tr>
<td>ION01</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>ION02</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>ION03</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>PRAM</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>RAID</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>TEMP1</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>TILTS</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
</tr>
<tr>
<td>VAR1</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>VAR2</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>VAR3</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
<tr>
<td>VAR4</td>
<td></td>
<td>common</td>
<td>***</td>
</tr>
</tbody>
</table>

**Code size** = 13f3 (5107)
**Data size** = 007a (122)
SUBROUTINE NEWCS(IFSxG)

NEWCS -- SUBROUTINE TO CALCULATE NEW CX, CY, CZ VALUES

CALLED BY: RAYSUB

CALLS: FREESP

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 09/17/86

VERSION: 2.0

REVISED: 07/25/36 -- INITIAL REVISION. TRANSLATED FROM TEKTRONIX BASIC TO VAX FORTRAN BY ERIC L. STROBEL.

07/30/86 -- V1.1. Change over to use of REAL*8 precision in the calculations.

09/17/86 -- V2.0. Got rid of the use of approximate rotation matrix in calc. of the new c-values.

USES: IFSxG A FLAG

IVCXSX, SY, SZ OLD SIGN VARIABLES

RVCX XF, YF, ZF FINAL INTERVAL COORDINATES

RVCXLAI PREVIOUS POINT'S LAT

RVCXLO1 PREVIOUS POINT'S LON

RVCXLAI LAT OF BEGINNING OF INTERVAL

RVCXLOI LON OF BEGINNING OF INTERVAL

RVCXALPH X

RVCXBETA X-IONOSPHERIC PARAMETERS

RVCXET1 X

RVCXET2 X

RVCX CX, CY, CZ OLD C-VALUES

TO CALCULATE THE NEW C AND S VALUES.

RETURNS:

IVCXSX, SY, SZ THE NEW SIGN VALUES
THE NEW C VALUES

C

- INTEGER IFSxG, IVCxSX, IVCxSY, IVCxSZ

C

- REAL*8 RVCxCX, RVCxCY, RVCxCZ, RVCxXF, RVCxYF, RVCxZF

C

- REAL*8 RVCxH5, RVCxLAl, RVCxLO1, RVCxHBOT

C

- REAL*8 RVCxALPH, RVCxBETA, RVCxET1, RVCxET2, RVLxI(5), RVLxJ-5.

C

- REAL*8 RTLxA4, RTLxQ5, RTLxQ6, RTLxQ7, RTLxD6, RVCxH6

C

- REAL*8 RTCxA, RTCxB, RTCxC, RTCxD

C

- COMMON/MORE/ IVCxSX, IVCxSY, IVCxSZ, RVCxCX, RVCxCY, RVCxCZ

C

- COMMON/OTHER/ RVCxLAl, RVCxLO1, RVCxHBOT, RTCxA, RTCxB, RTCxC

C

- COMMON/END/ RVCxXF, RVCxYF, RVCxZF, RVCxH5

C

- COMMON/IONO1/ RVCxALPH, RVCxBETA, RVCxET1, RVCxET2

C

- COMMON/GORP/ RVCxH6, RTCxD

C

- 10000 RTLxC5 = RVCxCX

C

- RTLxC6 = RVCxCY

C

- RTLxC7 = RVCxCZ

C

- RVLxI(4) = DSIN(RVCxLAl)

C

- RVLxI(5) = DCOS(RVCxLAl)

C

- A routine to concoct the rotation matrix for the calculation of the new C values.

C

- CALL FREESP(RVLxI, RVLxJ, RVLxK)

C

- Recall that the flag IFSxG can take on values of 3, 4, or 5, corresponding to the occurrence of reflections in the z, x, and y directions, respectively.

C

- IF (IFSxG.EQ.4) THEN

C

- RTLxA4 = 0.0D00

C

- ELSE

C

- RTLxA4 = IVCxSX*DSQRT(RTLxC5 - RVCxET1*RVCxXF)

C

- ENDIF

C

- IF (IFSxG.EQ.5) THEN

C

- RTLxA2 = 0.0D00

C

- ELSE

C

- RTLxA2 = IVCxSY*DSQRT(RTLxC6 - RVCxET2*RVCxYF)

C

- ENDIF

C

- IF (IFSxG.EQ.3) THEN

C

- RTLxA3 = 0.0D00

C

- ELSE

C

- RTLxA3 = IVCxSZ*DSQRT(RVCxBETA*RVCxZF - RVCxALPH*RVCxZF + RTLxC7)

C

- ENDIF

258
**Line#** | **Source Line**
---|---
101 | C
102 | RTLxQ5 = RVLxI(1) * RTLxA4 + RVLxI(2) * RTLxA2 + RVLxI(3) * RTLxA3
103 | RTLxQ6 = RVLxJ(1) * RTLxA4 + RVLxJ(2) * RTLxA2 + RVLxJ(3) * RTLxA3
104 | RTLxQ7 = RVLxK(1) * RTLxA4 + RVLxK(2) * RTLxA2 + RVLxK(3) * RTLxA3
105 | RTLxD6 = DSQRT(RVCxF*RVCxF + RVCyF*RVCyF + RVCzF*RVCzF)
106 | RVCxD6 = RTLxD6
107 | IF (IFSxG.NE.4) GO TO 104_4
108 | C
109 | Begin handling for an x-reflection.
110 | C
111 | IF (INTSIGN(RTLxQ5).EQ.IVCxSX) GO TO 10410
112 | IVCxSX = -IVCxSX
113 | GO TO 10416
114 | C
115 | A limit is set to ensure that the calculation doesn’t take
116 | an infinite time to creep up to a reflection point, so
117 | when the limit is reached, a reflection is forced.
118 | C
119 | IF (RTLxD6.GT.1.0D-04) GO TO 10416
120 | IVCxSX = -IVCxSX
121 | RTLxQ5 = 0.0D00
122 | GO TO 10416
123 | IVCxSX = INTSIGN(RTLxQ5)
124 | GO TO 10419
125 | C
126 | The x-reflection is complete, so handle updating the values
127 | in the y and z directions.
128 | C
129 | IVCxSY = INTSIGN(RTLxQ6)
130 | IVCxSZ = INTSIGN(RTLxQ7)
131 | GO TO 10446
132 | IF (IFSxG.NE.5) GO TO 10428
133 | C
134 | Begin handling for a y-reflection.
135 | C
136 | IF (INTSIGN(RTLxQ6).EQ.IVCxSY) GO TO 10424
137 | IVCxSY = -IVCxSY
138 | GO TO 10430
139 | C
140 | Again a limit is set to ensure that the calculation doesn’t
take an infinite time to creep up to a reflection point.
141 | so when the limit is reached, a reflection is forced.
142 | C
143 | IF (RTLxD6.GT.1.0D-04) GO TO 10430
144 | IVCxSY = -IVCxSY
145 | RTLxQ6 = 0.0D00
146 | GO TO 10430
147 | IVCxSY = INTSIGN(RTLxQ6)
148 | GO TO 10433
149 | C
150 | C

259
The y-reflection is complete, so handle updating the values in the x and z directions.

```
10430 IVCSX = INTSIGN(RTLxQ5)
10431 IVCSZ = INTSIGN(RTLxQ7)
10432 GO TO 10446
10433 IF (IFSxG .NE. 3) GO TO 10442
10434 C Begin handling for a z-reflection.
10435 IF (INTSIGN(RTLxQ7).EQ.IVCSZ) GO TO 10438
10436 IVCSZ = -IVCSZ
10437 GO TO 10446
10438 IF (RTLxD6.GT.1.0D-04) GO TO 10444
10439 IVCSZ = -IVCSZ
10440 RTLxQ7 = 0.0D00
10441 GO TO 10444
10442 IVCSZ = INTSIGN(RTLxQ7)
10443 GO TO 10446
10444 C The z-reflection is complete, so handle updating the values in the x and y directions.
10445 IVCSX = INTSIGN(RTLxQ5)
10446 IVCSY = INTSIGN(RTLxQ6)
10447 RVCCX = RTLxQ5 * RTLxQ5
10448 RVCCY = RTLxQ6 * RTLxQ6
10449 RVCCZ = RTLxQ7 * RTLxQ7
10450 C IFSxG = 6 which means that the ray is outside the bounds of the ionosphere, headed down, so presumably the program is handling an earth bounce event. Therefore, a specular reflection is forced.
10451 IF (IFSxG.EQ.6.AND.RVCxH5.LT.RVCxH6) IVCSZ = -IVCSZ
10452 RETURN
10453 END
```

Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSXG</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLXC5</td>
<td>local</td>
<td>REAL*8</td>
<td>4</td>
</tr>
</tbody>
</table>
### NEWCS Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTLXC6</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXD6</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXC7</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXQ5</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXQ6</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXQ7</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXI</td>
<td>local</td>
<td>REAL*8</td>
<td>40</td>
</tr>
<tr>
<td>RVLXJ</td>
<td>local</td>
<td>REAL*8</td>
<td>40</td>
</tr>
<tr>
<td>RVLXX</td>
<td>local</td>
<td>REAL*8</td>
<td>40</td>
</tr>
<tr>
<td>RTLXA2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXA3</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXA4</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>IVCXSX</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSZ</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVCXCX</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXCY</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXCZ</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXXF</td>
<td>END</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXYF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXZF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXH5</td>
<td>END</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXLA1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXL01</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXHBOT</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXALPH</td>
<td>ION01</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXBETA</td>
<td>ION01</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXET1</td>
<td>ION01</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXET2</td>
<td>ION01</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXD6</td>
<td>GORP</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXC</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXD</td>
<td>GORP</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE TRIANG

TRIANG -- SUBROUTINE TO CHECK IF A LOCATION IS WITHIN A TRIANGLE OF S.C.P.'S

CALLED BY: IONOPAR

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 02/03/88

VERSION: 2.2

REVISED:
07/25/86 -- INITIAL REVISION. TRANSLATED FROM TEKTRONIX BASIC TO VAX FORTRAN BY ERIC L. STROBEL.
07/30/86 -- V1.1. Change over to use of REAL*8 precision in the calculations.
10/10/86 -- V2.0. Changed to conform to the change over to RAYTRACE as a subroutine.
09/01/87 -- V2.1. Slight changes to speed things up a bit.
02/03/88 -- V2.2. More streamlining of INTs and NINTs. Logical variable added to help correctly handle the possibly worrisome case of the grid at the geographic north pole.

USES: IVSXSCT1,2,3 THE THREE S.C.P.'S THAT MAKE THE TRIANGLE
RVcxx.Y,Z INTERMEDIATE COORDINATES
RVcxx.V.W(900) THE S.C.P. COORDINATES

TO DETERMINE WHETHER OR NOT A LOCATION IS WITHIN A TRIANGLE OF S.C.P.'S. First, the locations & array indices of the four surrounding grid points are obtained, and then the potential triangles involving three of the corners of this surrounding rectangle
are checked in an attempt to make sure that the point
of interest is as far within a triangle as possible.
The three grid points that make up this triangle are
reported. These are the points from which the
interpolation is based. Recall the following
definitions: RVxGRID(1) = lat spacing
(2) = lon spacing
(3) = starting lat
(4) = starting lon
(5) = # in lat
(6) = # in lon.

RETURNS:
IFSxN4 FLAG (=1 IF NOT INTERIOR TO A
TRIANGLE OF S.C.P.'S)

INTEGER IVLxT1, IVLxT2, IVLxT3, IVLxT4, IVCxSCT1
INTEGER IVCxSCT2, IVCxSCT3, IVCxSCTS, IVLxNLN, IVLxILN
REAL*8 RVLxLAIN, RVLxLOIN, RVCxLAI, RVxGRID(6), RVCxLOI
REAL*8 RVLxP1, RVLxP2, RVCxXI, RVCxYI, RVCxZI
LOGICAL LVLxPOLE
COMMON /MAINDAT/ RVxGRID
COMMON /START/ RVCxXI, RVCxYI, RVCxZI, RVCxLAI, RVCxLOI
COMMON /TEMP2/ IVCxSCTS, IVCxSCT1, IVCxSCT2, IVCxSCT3

RVLxLAIN = (RVCxLAI - RVxGRID(3)) / RVxGRID(1)
RVLxLOIN = (RVCxLOI - RVxGRID(4)) / RVxGRID(2)
IVLxILN = INT(RVLxLAIN)
RVLxP1 = RVLxLAIN - IVLxILN
RVLxP2 = RVLxLOIN - INT(RVLxLOIN)
IVLxNLN = NINT(RVxGRID(6))
IVLxT1 = IVLxILN * IVLxNLN + INT(RVLxLOIN) - 1
IVLxT2 = IVLxT1 + 1
IVLxT3 = IVLxT1 + IVLxNLN
IVLxT4 = IVLxT3 + 1
LVLxPOLE = NINT(RVxGRID(3) + (IVLxILN + 1) * RVxGRID(1)) .EQ. 0
IF (RVLxP1.LT.0.5 OR LVLxPOLE) THEN
IVECSCT1 = IVLxT1
IVECSCT2 = IVLxT2
IF (RVLxP2.LT.0.5D0) THEN
IVECSCT3 = IVLxT3
ELSE
IVECSCT3 = IVLxT4
ENDIF
294    ELSE
295        IVCXSCT2 = IVLXT1
296        IVCXSCT3 = IVLXT4
297        IF (RLXLP2.LT.0.5D00) THEN
298            IVCXSCT1 = IVLXT1
299        ELSE
300            IVCXSCT1 = IVLXT2
301        ENDIF
302        RETURN
303    END

TRIANG Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVLXT1</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVLXT2</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVLXT3</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVLXT4</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RLXLP1</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RLXLP2</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>IVLXLIN</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVLXNLN</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RLXLAIN</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>LVLXPOLE</td>
<td>local</td>
<td>LOGICAL*4</td>
<td>4</td>
</tr>
<tr>
<td>RLXLOIN</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IVCXSCT1</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSCT2</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSCT3</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVCXSCT4</td>
<td>TEMP2</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXGRID</td>
<td>START</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVXGRID</td>
<td>MAINDAT</td>
<td>REAL*3</td>
<td>48</td>
</tr>
<tr>
<td>RVXLOI</td>
<td>START</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>

264
SUBROUTINE ACCFSP(RVSXH0, RVSX54, IFSxG)

ACCFSP -- SUBROUTINE TO SPEED UP CALCULATION OF FREE SPACE PROPAGATION

CALLED BY: ENDPT

AUTHOR: MICHAEL H. REILLY & ERIC L. STROBEL

DATE: 03/13/88

VERSION: 2.2

REVISED: 07/25/86 -- INITIAL REVISION. TRANSLATED FROM TEKTRONIX BASIC TO VAX FORTRAN BY ERIC L. STROBEL.

07/30/86 -- V1.1. Change over to use of REAL*8 precision in the calculations.

09/01/87 -- V2.0. Changed to allow ray to be above the top of the ionosphere.

12/08/87 -- V2.1. A change has been made to make sure that the ray stops at the desired point.

03/18/88 -- V2.2. Minor changes made to accommodate the new usage of angular range.

USES: IVCXSX, SY, SZ

RVCXRE

RVSXH0

RVCXCX, CY, CZ

TO OBTAIN THE ENDPOINT OF A RAYPATH INCREMENT THAT IT THROUGH FREESPACE.

RETURNS:

RVCXF, YF, ZF

THE ENDPOINT COORDINATES

---------------------------------------------------------------
Calculate the ray direction cosines.

The calculation is the solution of the quadratic equation resulting from a straight line piercing a sphere. First, the radii of the pertinent spheres are obtained, based on whether the freespace propagation is above or below the bounds of the model ionosphere. Based on the discriminant, the proper solution is chosen. The linear distance covered is calculated (RVLxS) and using the direction cosines, the final coordinate values are obtained.

```fortran
IVLxI = 0

IF (RVCxH5.GE.RPCxHTP) THEN
  IVLxJ = 3
  RVLxRR(3) = RPCxHTP
  RVLxRR(4) = RVCxHCT
ELSE
  IVLxJ = 1
  RVLxRR(1) = 0.3D00
  RVLxRR(2) = RVCxHB
ENDIF
```

IF (IVCxSZ.LE.0) GO TO 10000
```
IVLxJ = IVLxJ + 1
IVLxSSS = -1
RTLxRD = RTLxRC2 + (RTLxRR(IVLxJ) - RVSxHO) * (RTLxT1 + #RTLxRR(IVLxJ) + RPCxRE)

IF (RTLxRD.LT.0.0D00) THEN
  PRINT *,' Ray passes through a minimum of altitude.'
  IFSXG = 7
  GO TO 9000
ENDIF

IVLxSSS = -RTLxC - IVLxSSS * DSQRT(RTLxD)
RTLxXF = RTLxQ1*IVLxSS
RTLxYF = RTLxQ2*IVLxSS
RTLxZF = RTLxQ3*IVLxSS

DONE 12/3. After some preliminary calculations, a check is made to see if the straight line endpoint is beyond the range cutoff. If so, the distance traveled along the line is backed off by an appropriate amount and the endpoint coordinates are recalculated.

RTLxT2 = RTLxZF + RTLxT1
RTLxT3 = RTLxXF + RTLxT2 + RTLxT2
RTLxRP = DSQRT(RTLxT3 + RTLxT2*RTLxT2)
RTLxTH = RTLxXF / DSQRT(RTLxT3)
RTLxTH = DATAN(RTLxTH)
RTLxTHE = RPCxPI / 2.0 + RTLxTH
RTLxDEL = RTLxS * DSIN(RTLxTHE) / RTLxRP
RTLxDEL = DASIN(RTLxDEL)

IF ((RVSxS4 + RTLxDEL).GE.RVCxALIM) THEN
  RTLxDP = RVCxALIM - RVSxS4
  RVCxSBR = RTLx1 * DSIN(RTLSR) / (RTLxRP * DSIN(RTLxDEL - RTLxDP))

  RVCxXF = RTLxQ1*RTLxSBR
  RVCxYF = RTLxQ2*RTLxSBR
  RVCxZF = RTLxQ3*RTLxSBR
  RTLxT2 = RTLxZF + RTLxT1
  RTLxT3 = RTLxXF + RTLxT2 + RTLxT2
  RTLxRP = DSQRT(RTLxT3 + RTLxT2*RTLxT2)

  IF (DABS(RTLxRP-RPCxRE).GT.1.0D-02) THEN
    IFSxG = 7
    ENDIF
ENDIF

RETURN
END
### ACCFSP Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>.FSXG</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVXKS4</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVXH0</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTLXQ1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>TLXKX1</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RTLXK2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>TLXKJ</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RTLXQ3</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXD</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>TLXAT1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXAT2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXAT3</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXDP</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXDEL</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLS</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXATHE</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXRP</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>TLXXR</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>TLXATTH</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VLXSSS</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>TLXSPR</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>TLXC2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VCXSX</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>VCXSY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>VCSXZ</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>FCXRE</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VCXCY</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VCKZC</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VCKXF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VXCEF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VCKZF</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>PCXPI</td>
<td>END</td>
<td>PRAM</td>
<td>REAL*8</td>
</tr>
<tr>
<td>VCHXCT</td>
<td>END</td>
<td>OTHER</td>
<td>REAL*8</td>
</tr>
<tr>
<td>PCXDTI</td>
<td>END</td>
<td>PRAM</td>
<td>REAL*8</td>
</tr>
<tr>
<td>VCHX5</td>
<td>END</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>PCXHTP</td>
<td>END</td>
<td>PRAM</td>
<td>REAL*8</td>
</tr>
<tr>
<td>TLXL1</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>TLXL2</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>TLXL4</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>TLXL5</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VCHHS</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>TLXZZZ</td>
<td>GORP</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>VCSALIM</td>
<td>GORP</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE FREESP(RVSXI, RVSXJ, RVSXK)

FREESP -- SUBROUTINE TO OBTAIN SOME VALUES USED TO COMPUTE NEW C-VALUES IN THE CASE OF FREESPACE PROPAGATION.

CALLED BY: NEWCS

AUTHOR: MICHAEL H. REILLY

DATE: 07/30/86

VERSION: 1.1

REVISED: 07/25/86 -- INITIAL REVISION. TRANSLATED FROM TEKTRONIX BASIC TO VAX FORTRAN BY ERIC L. STROBEL.

07/30/86 -- V1.1. Change over to use of REAL*8 precision in the calculations.

USES: RVCXLAI LAT OF BEGINNING OF INTERVAL
RVCXLOI LON OF BEGINNING OF INTERVAL
RVSXJ(5) X-ARRAYS OF INTERMEDIATE VALUES
RVSXK(5) X

TO CALCULATE SOME MORE INTERMEDIATE VALUES INVOLVED IN COMPUTING THE C-VALUES FOR FREESPACE PROPAGATION

RETURNS:
RVSXJ(5) X-ARRAYS OF INTERMEDIATE VALUES
RVSXK(5) X

REAL*8 RVSXJ(5), RVSXK(5), RVCXLOI, RVCXLAI, RVCXLOL, RTCXZ, RTCXN

REAL*8 RTCXZ, RTCXN
COMMON /START/ RVCXI, RVCXI, RVCXZI, RVCXLOI, RVCXLOI

COMMON /OTHER/ RVCXLI, RVCXLOI, RTCXA, RTCXB, RTCXC, RTCXD

RVSXJ(4) = DSIN(RVCXLA)
RVSXJ(5) = DCOS(RVCXLA)
RVSXJ(4) = DSIN(RVCXLOI - RVCXLOI)
RVSXK(5) = DCOS(RVCXLOI - RVCXLOI)
RVSXI(1) = RVSXJ(4) * RVSXJ(4) * RVSXK(5) + RVSXI(5) * RVSXJ(5)
RVSXI(2) = RVSXJ(4) * RVSXK(4)
RVSXI(3) = RVSXJ(5) * RVSXJ(4) * RVSXK(5) - RVSXI(4) * RVSXJ(5)
RVSXJ(1) = -RVSXI(4) * RVSXK(4)
RVSXJ(2) = RVSXK(5)
RVSXJ(3) = -RVSXI(5) * RVSXK(4)
RVSXK(1) = RVSXI(4) * RVSXJ(5) * RVSXK(5) - RVSXI(5) * RVSXJ(4)
RVSXK(2) = RVSXJ(5) * RVSXK(4)
RVSXK(3) = RVSXI(5) * RVSXJ(5) * RVSXK(5) + RVSXI(4) * RVSXJ(4)
RETURN
END

FREESE: Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXK</td>
<td>param</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVSXJ</td>
<td>param</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVSXI</td>
<td>param</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXYI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXZI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXLAI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXLOI</td>
<td>START</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXLI</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXLOI</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXC</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXD</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE TIMES(RVSxBUN. IVSXBUN, RVSLON0)

TIMES -- SUBROUTINE TO CONVERT GIVEN TIME INTO THE TIME
VARIABLES NEEDED LATER

CALLED BY: RAYSUB

AUTHOR: ERIC L. STROBEL

DATE: 03/13/38

VERSION: 2.0

REVISED: 08/07/86 -- INITIAL REVISION.

10/10/36 -- V1.1. Changed to conform to
the status of RAYTRACE as a subroutine.

03/18/38 -- V2.0. Vastly rewritten to give
a better idea of the mid-hop longitude, and
to better handle being off by factors of
(2 * pi) in the mid-hop longitude.

USES: RVSxBUN(ll,15) ARRAY OF BOUNCE PT. DATA
IVSXBUN BOUNCE COUNT
RVSLON0 LONG. OF LAUNCH PT.
ICLXM(ll) NUMBER OF DAYS IN EACH MC.
IVCXTIM(5) DATE AND TIME ARRAY

REVISED:

TO CALCULATE THE DECIMAL YEAR VALUE AND THE TIME SINCE
MIDNIGHT IN RADIANS

RETURNS:
RVCXyr DECIMAL YEAR
RVCXTIM TIME SINCE MIDNIGHT IN RADIANS

REAL*8 RVCXyr, RVCXTIM, RVSxBUN(ll,15), RPCXPI, RPCXSTOR
REAL*8 RVCA, RVLON0, RVSLON0, RTCXA, RTCXB, RTLXL0
REAL*8 RTLXL1, RTLXDEL

271
Line# Source Line
576        INTEGER ICLxM(11), IVCxTIM(5), ITLxA, ITLxB, IVSxNB
577        INTEGER IVCxSSN, IFCxGND, IVLxNR
578        COMMON /PRAM/ RPCxPI, RPCxDTOR, RTCxA, RTCxB
579        COMMON /LPARM/ IVCxSSN, IVCxTIM, RVCxYR, RVCxA, IFCxGND.
580        *RVCTxTIM
581        COMMON /?RAM/ R.PCxPI, RPCxDTOR, RTC:-A, RTCxB
582        COMMON ,LPARM/ IVCXSSN, IVCxTIM, RVCxYR, RVCxA, IFCxGND.
583        DATA ICLxM, 3: 28.31, 30.31, 30.31, 30.31, 30.31, 30./
584        C
585        10000  ITLxA = IVCxTIM(2) - 1
586        IVLxNR = 1
587        ITLxB = IVCxTIM(3)
588        DO 10100 I = 1, ITLxA
589            ITLxB = ITLxB - ICLxM(I)
590        10100  CONTINUE
591        RVCxYR = IVCxTIM(1) - ITLxB/365.0D00 - 1980.0D00
592        RVCxTIM = IVCxTIM(4) + IVCxTIM(5)/60.0D00
593        C
594        C Compute the longitude of the midpoint of a hop, for the
595        absorption calculation.
596        C
597        IF (IVSxNB.GT.1) THEN
598            RTLxLO = RVSxBUN(IVSxNB-1,2)
599        ELSE
600            RTLxLO = RVSxLON0
601        ENDIF
602        RTLxl1 = RVSxBUN(IVSxNB,2)
603        RTLxDEL = RTLxl1 - RTLxLO
604        IF (DABS(RTLxDEL).GT.RPCxPI) RTLxDEL = RTLxDEL-INTSIGN(RTLxDEL)
605        **(2.0D00 * RPCxPI)
606        RTLxLON = RTLxLO + RTLxDEL/2.0D00
607        C
608        RVCxTIM = RTLxLON + 15.0D00 * RPCxDTOR * RVCxTIM
609        10200  IF (RVCxTIM.GE.(2.0D00*RPCxPI)) THEN
610            RVCxTIM = RVCxTIM - 2.0D00*RPCxPI
611            GO TO 10200
612        ELSE IF (RVCxTIM.LT.0.0D00) THEN
613            RVCxTIM = RVCxTIM + 2.0D00*RPCxPI
614            GO TO 10200
615        ENDIF
616        RETURN
617        END

TIMES Local Symbols

Name                  Class Type Size
RVSxLON0. . . . . . . . param
IVSxNB. . . . . . . . param

272
<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXBUN</td>
<td>param</td>
<td>Integer*4</td>
<td>44</td>
</tr>
<tr>
<td>ICLXM</td>
<td>local</td>
<td>Integer*4</td>
<td>4</td>
</tr>
<tr>
<td>ITLXA</td>
<td>local</td>
<td>Integer*4</td>
<td>4</td>
</tr>
<tr>
<td>ITLXB</td>
<td>local</td>
<td>Integer*4</td>
<td>4</td>
</tr>
<tr>
<td>RTLXLO</td>
<td>local</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLXL1</td>
<td>local</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>I</td>
<td>local</td>
<td>Integer*4</td>
<td>4</td>
</tr>
<tr>
<td>RTLXDEL</td>
<td>local</td>
<td>Real*3</td>
<td>3</td>
</tr>
<tr>
<td>IVLXNR</td>
<td>local</td>
<td>Integer*4</td>
<td>4</td>
</tr>
<tr>
<td>RTLXLEN</td>
<td>local</td>
<td>Real*3</td>
<td>3</td>
</tr>
<tr>
<td>RVCTXMR</td>
<td>Lparm</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCTXMI</td>
<td>Lparm</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>Pram</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXDTOR</td>
<td>Pram</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCTXA</td>
<td>Lparm</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>Pram</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>Pram</td>
<td>Real*8</td>
<td>3</td>
</tr>
<tr>
<td>IVCTXTIM</td>
<td>Lparm</td>
<td>Integer*4</td>
<td>20</td>
</tr>
<tr>
<td>IVCTXSSN</td>
<td>Lparm</td>
<td>Integer*4</td>
<td>4</td>
</tr>
<tr>
<td>IFCXGND</td>
<td>Lparm</td>
<td>Integer*4</td>
<td>4</td>
</tr>
</tbody>
</table>
SUBROUTINE LOSS(RVSXBOU,RVSXBUN,IVSXNB,RVSXLAT0,RVSXLONO
* RVSXEL0,IFSXEND)

LOSS -- SUBROUTINE TO EVALUATE THE LOSSES ALONG THE RAY'S PATH
CALLED BY: RAYSUB

AUTHOR: ERIC L. STROBEL
DATE: 09/01/87
VERSION: 3.0

REVISED: 08/07/86 -- INITIAL REVISION.
09/05/86 -- V2.0. ADD HORIZON EFFECTS.
BUG FIX. COSMETIC CHANGES.
10/10/86 -- V2.1. Changed to conform to the new status of RAYTRACE as a subroutine.
Also trimmed some of the commented out code.
04/09/87 -- V2.2. Corrected error in calculation of geometric loss.
09/01/87 -- V3.0. Changed to comment out absorption loss calculation in anticipation of a better routine. Also, the kludgy "excess loss" has been commented out. The focusing calculation has been generalized and some corrections have been made.

USES: RVSXBOU (41,5)
RVSXBUN(4,3,11)
IVSXNB
RVSXLAT0
RVSXLONO
IFSXEND
RVCXFSQ

BOUNCE PT. ARRAY
RAY BUNDLE ARRAY
BOUNCE COUNT
LAT OF LAUNCH POINT
LON OF LAUNCH POINT
FLAGS THE LOSS COMP. FIP
END OF THE RAY
WAVE FREQUENCY SQUARED
<table>
<thead>
<tr>
<th>Line#</th>
<th>Source Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>670</td>
<td>C IVCxSSN SUNSPOT NUMBER</td>
</tr>
<tr>
<td>671</td>
<td>C IVCxTIM(5) DATE AND TIME ARRAY</td>
</tr>
<tr>
<td>672</td>
<td>C RVxYEAR DECIMAL YEAR</td>
</tr>
<tr>
<td>673</td>
<td>C RVxALT00 ANGLE OF INCIDENCE AT 100 KM ALTITUDE</td>
</tr>
<tr>
<td>674</td>
<td>C IFCxGND TYPE OF REFLECTION SURF.</td>
</tr>
<tr>
<td>675</td>
<td>C RVxTIM TIME SINCE MIDNIGHT IN RADIANS</td>
</tr>
</tbody>
</table>

678 TO CALCULATE THE SIGNAL LOSS CONTRIBUTIONS FROM:

679 ABOSPTION

681 REFLECTION

682 OTHER PROCESSES

683 GEOMETRIC FOCUSING/DEFOCUSING

685 RETURNS:

686 RVxLOS A ABSORPTION LOSS

687 RVxLOSR REFLECTION LOSS

688 RVxLOSSX EXCESS LOSS

689 RVxLOSG GEOMETRIC LOSS

693 REAL*8 RTLxK1, RVxMOT, RVxYEAR, RVxDEC, RVxLAT

694 REAL*8 RVxSxBUN(4, 3, 11), RVxLAT0, RVxLONG0, RVxZEN

695 REAL*8 RVxZ12, RVxAL00, RVxM, RVxK, RVxX, RVxA

696 REAL*8 RTLxA, RTLxK, RTLxDEN, RVxLOS A, RTLxRE, RTLxIM

697 REAL*8 RTLxARG, RVxLOSR, RCTxB, RCTx2, RCTx3, RVxLELV

698 REAL*8 RCTxA, RCTxB, RCTx3, RCTx5, RCTxGP, RVxLOD

699 REAL*8 RTLxG, RVxG, RTLxLO, RVxM, RVxC

700 REAL*8 RVxGML, RVxLOS, RVxXU, RTLxR, RVxX3(3, 3)

701 REAL*8 RTLxV1(3), RTLxV2(3), RTLxDE, RTLxCT:3), RTLxCV:3)

702 REAL*8 RVxAREA, RTLxT(3), RTLxN:3), RTLxMAG1, RTLxMAGC

703 REAL*8 RVxLOSG, RCPxPI, RCPxTOR, RCPxRE, RVxSCC:3, 11, 15

704 REAL*8 RVxII, RVxKJ, RVxKK, RVxSFQ, RVxLFRE

705 REAL*8 RVxTIM, RVxHOR, RVxB, RVxZL, RVxSE10

706 REAL*8 RVxSE2TX:3, 3), RCTx1, RCTx2, RCTx3, RCTx4, RCTx5

707 REAL*8 RCTx6, RCTx7, RCTx8, RVxHMIN, RVxH5, RVxCK

708 REAL*8 RCTx9, RCTx10, RCTxC

709 C

710 COMPLEX*16 C TLxNSQ, C TLxCSQ, C TLxSHI, C TLxSQ

711 COMPLEX*16 CVLxRH, CVLxRSV

712 C

713 INTEGER IVxSxK, IVxSY, IVxSxZ

714 INTEGER IVxSxIB, IVxTIM(5), IFCxGND, IVxSSN, IFxEND

715 C

716 COMMON /MISC/ RVxSE2TX:3, RCTx1, RCTx2, RCTx3, RCTx4, RCTx5

717 #RVxHMIN

718 COMMON /MORE/ IVxSxK, IVxSY, IVxSxZ, RVxCK, RVxCY, RVxCZ

719 COMMON /END/ RCTx6, RCTx7, RCTx8, RVxH5
Line# | Source Line
---|---
720 | COMMON /LOCAL/ RVLxELEV
721 | COMMON /PRAM/ RPCxPI, RPCxDTOR, RPCxRE, RTCxC
722 | COMMON /OTHER/ RVCxII, RVxKJ, RVxKK, RVxFSQ, RTCxA, RTCxB
723 | COMMON /LRAM/ IVCxSSN, IVCxTIM, RVxYEAR, RVxA100, IFxCxGN
724 | * RVxTIM
725 | COMMON /LOSS/ RVxLOSS, RVxLOS. RVxLOS. RVxLOS
726 | COMMON /OTHER/ RVxHI, RVxJ, RVxK, RVxFSQ, RTCxA, RTCxB
727 | COMMON /PARM/ RVxSSN, IVCxTI, RVxYEAR, RVxA, 00, IFxCxGN
728 | COMMON /ZOSSES/ RVxLOSA, RVxLOS. RVxLOS
729 | COMMON /ZOSSES/ RVxLOSA, RVxLOS. RVxLOS
730 | COMMON /ZOSSES/ RVxLOSA, RVxLOS. RVxLOS
731 | COMMON /ZOSSES/ RVxLOSA, RVxLOS. RVxLOS
732 | COMMON /ZOSSES/ RVxLOSA, RVxLOS. RVxLOS
733 | COMMON /ZOSSES/ RVxLOSA, RVxLOS. RVxLOS
734 | COMMON /ZOSSES/ RVxLOSA, RVxLOS. RVxLOS
735 | ABSORPTION LOSS CALC (3.3x is a temporary vbl. to denote which ray is being done)
736 | G10000 IF (FSxEND.EQ.1.AND.IVSxNB.EQ.1.AND.RVCxMIN.GT.30.)
737 | #GO TO 20000
738 | IF (FSxEND.EQ.1.AND.IVSxNB.GT.1.AND.RVCxH5.LT.60.)
739 | #GO TO 20000
740 | RVLxMOT = (RVxYEAR - JIDINT(RVCxYEAR)) * 365.0000 + 16.3000
741 | * 30.41667000
742 | RVLxDEC = 0.398D00 * DSIN(RPCxPI * (RVLxMOT - 0.1700) / 6.3000)
743 | RVLxDEC = DSIN(RVLxDEC)
744 | RVLxLAT = (0.2000 * IVSxNB - 1.0000) / (2.0000 * IVSxNB)
745 | * (RVxBOU(IVSxNB, 1) - RVxLAT0) + RVxLAT0
746 | RVLxZEN = DSIN(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
747 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
748 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
749 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
750 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
751 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
752 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
753 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
754 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
755 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
756 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
757 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
758 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
759 | RVLxZEN = DCOS(RVLxLAT) * DSIN(RVLxDEC) = DCOS(RVLxLAT) * DCOS(RVLxDEC)
760 | ENDIF
761 | IF (RVLxLAT.GE.30.0000) THEN
762 | IF (IVCxTIM(2).EQ.11) THEN
763 | RNLxK1 = 0.003D00
764 | ELSE IF (IVCxTIM(2).EQ.12) THEN
765 | RNLxK1 = 0.0083D00
766 | ELSE IF (IVCxTIM(2).EQ.1) THEN
767 | RNLxK1 = 0.0269D00
768 | ELSE IF (IVCxTIM(2).EQ.2) THEN
769 | RNLxK1 = 0.0089D00
770 | ENDIF
771 | IF (RVLxZEN.GT.90.0000) THEN
772 | RVLxZ = 0.01D00
773 | ELSE IF (IVCxTIM(2).EQ.11) THEN
774 | RVLxZ = 0.0283D00
775 | ELSE IF (IVCxTIM(2).EQ.12) THEN
776 | RVLxZ = 0.0269D00
777 | ELSE IF (IVCxTIM(2).EQ.1) THEN
778 | RVLxZ = 0.0089D00
779 | ENDIF
780 | IF (RVLxLAT.GE.60.0000) THEN
781 | RVLxW = 2.25000 - 0.0033D00 * RVLxLAT
782 | ELSE
783 | RVLxW = 2.25000 - 0.0033D00 * RVLxLAT
784 | ENDIF
785 | IF (RVLxZEN.GT.90.0000) THEN
786 | RVLxX = 0.01D00
787 | ELSE
788 | RVLxA = DCOSD(0.393D00 * RVLxZEN)
789 | RVLxB = DCOSD(0.833D00 * RVLxZ12)
790 | RVLxC = DCOSD(RVLxZ12)
791 | 276
C RTLxC = RTLxC ** RVLxM
771 C RVLxK = (1.0D00 + 0.005D00*IVCxSSN) * RTLxC * RTLxG / RTLxZ
772 C ENDIF
773 C RVLxFRE = DSQRT(RVCxFSQ)
774 C RTLxDEN = 10.2D00 + (RVLxFRE + 1.4D00) * (RVLxFRE + 1.4D00)
775 C RTLxDEN = RTLxDEN * DCOSD(RVCxA100)
776 C RVCxLOSA = RVCxLOSA + (286.0D00 * (1.0D00 + 0.0087D00 * RVLxLAT) * * RVLxW * RVLxK / RTLxDEN)
777 C ENDIF
778 C REFLECTION LOSS CALCULATION. A Fresnel coefficient calculation.
779 C For details, see the Radio Science paper referred to in the documentation.
780 C
781 C 20000 IF (IVSxNB-1.0D.0) GC TO 40000
782 C IF (IFCxGND.EQ.1) THEN
783 C RTLxRE = 7.0D00
784 C RTLxIM = -0.005D00 * 18000.0D00 / RVLxFRE
785 C CTLxNSQ = DCMPLX(RTLxRE, RTLxIM)
786 C ELSE
787 C RTLxRE = 30.0D00
788 C RTLxIM = -5.000D00 * 18000.0D00 / RVLxFRE
789 C CTLxNSQ = DCMPLX(RTLxRE, RTLxIM)
790 C ENDIF
791 C CTLxCSQ = DCMPLX(DCOS(RVLxEL), DCOS(RVLxEL))
792 C CTLxSIN = DCMPLX(DSIN(RVLxEL))
793 C CTLxSQT = CDSQT(CTLxNSQ - CTLxCSQ)
794 C CVLxRH = (CTLxNSQ - CTLxSIN - CTLxSQT)/((CTLxNSQ - CTLxSIN + CTLxSQT)
795 C CVLxRV = (CTLxNSQ * CTLxSIN - CTLxSQT) / (CTLxNSQ * CTLxSIN + CTLxSQT)
796 C RTLxARG = (CDABS(CVLxRV) * CDABS(CVLxRV) + CDABS(CVLxRH) * CDABS
797 C * CVLxRH)) / 2.0D00
798 C RVCxLOSR = RVCxLOSR + DABS(10.0D00 * DLOG10(RTLxARG))
799 C EXCESS LOSS CALCULATION
800 C
801 C 30000 RCTxA2 = 11.18959D00
802 C RCTxB2 = -0.02916595D00
803 C RCTxC2 = -8.5401D-04
804 C RCTxA3 = -70.74865D00
805 C RCTxB3 = -0.04418982D00
806 C RCTxC3 = 0.002169971D00
807 C RTLxGC = RCTxA2 + RVCxYEAR * (RCTxB2 + RVCxYEAR * RCTxC2)
808 C RTLxGP = RCTxA3 + RVCxYEAR * (RCTxB3 + RVCxYEAR * RCTxC3)
809 C RVLxLON = ((2.0D00 * IVSxNB) - 1.0D00) / (2.0D00 * IVSxNB)
810 C * (IVSxBOU(IVSxNB.2) - RVSxLONO) + RVSxLONO
811 C RTLxGLA = (90.0D00 - RTLxGC) * RPCxDTOR
812 C RTLxGLO = RTLxGP * RPCxDTOR
813 C RTLxDLA = RTLxGLA - RVLxLAT * RPCxDTOR
814 C RTLxDLO = RTLxGLO - RVLxLON
815 C RVLxMC = DCOS(RTLxDLA) * DCOS(RTLxDLO)
816 C RVLxMC = DACOS(RVLxMC)
RVGLxGML = (RPCxPI/2.0D00 - RVGLxMC) / RPCxDTOR  
IF (RVGLxGML.GT.60.0D00) THEN  
  RVCLxLOSX = 11.3D00  
ELSE  
  RVCLxLOSX = 9.3D00  
ENDIF  

GEOMETRIC LOSS CALCULATION  

RCLxAU = 1.903853875D-09*DCOS(RVSLxEL0)  

RCLxAU is the cross-sectional area of a bundle of rays bounded by a square defined by azimuth and elevation deviations of 0.0025 degrees from one corner. This cross-section is taken at a distance of 1 km from the starting point of the problem. Only three rays are necessary to define the area, the area may be taken as represented by the vector cross product of the vector from the 'corner' point to one of the other points, with the other similarly taken vector. This is in fact how the calculation is done.  

DO 40100 I = 1,3  

RVSxBUN contains the endpoint locations for the rays in the bundle.  

RTLxR = RPCxRE - RVSxBUN(I,1:3)  
The RTLxS are the GEC coordinate values for the endpoints of the three rays that define the area.  

RTLxS(I,1) = RTLxR*DCOS(RVSxBUN(I,1:3)) + DCOS(RVSxBUN)  
RTLxS(I,2) = RTLxR*DCOS(RVSxBUN(I,1:3)) + DSIN(RVSxBUN)  
RTLxS(I,3) = RTLxR*DSIN(RVSxBUN(I,1:3))  

40100 CONTINUE  

These next two quantities are the vectors (in GEC components representing the deviation of two of the points from the third (the 'corner').  

DO 40200 I = 1,3  

RTLxV1(I) = RTLxS(I,1) - RTLxS(I,2)  
RTLxV2(I) = RTLxS(I,2) - RTLxS(I,3)  

40200 CONTINUE  

RTLxDE = SQRT(RVCxCX+RVCxCY+RVCxCZ)  

This next vector is the ray direction in SEZ components.
Next, the ray direction vector is transformed into SEC coordinates.

Now, the triple vector product is calculated which will give the cross-sectional area of the ray bundle in the plane defined by the ray direction vector.

A calculation now begins to obtain the angle between the ray direction and the surface determined by the cutoff/earth bounce condition.

The ordinary geometric loss calculation.
then the empirical horizon focusing calculation is done. The result of this is then compared to the normal focusing result and the greater loss is used.

```fortran
IF (ABS(RVLxELEV/RPCxDTOR).LT.1.0D00) THEN
  RVLxHOR = 60.012D00 + 20.0D00*DLOG10(DBLE(RVSxNB))/3.0D00
  *-10.0D00*DLOG10(DCOS(RVLxELEV)) + 10.0D00*DLOG10(DSIN(RVSxBCU)
  *IVSxBU.4)) - 10.0D00*DLOG10(RVLxFRE)/3.0D00
ENDIF

RETURN
END
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFSXEND</td>
<td>param</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVSXEL0</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLONG0</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLAT0</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IVSXBUN</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXBOU</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>J</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RCLXAU</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RTLXMA1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXMA2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXV1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXV2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXDE</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>CVLXRH</td>
<td>local</td>
<td>COMPLEX*15</td>
<td>15</td>
</tr>
<tr>
<td>RTLXR</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RTLXS</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RTLXAT</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXCM</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXCT</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RTLXRE</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>CTLXCSQ</td>
<td>local</td>
<td>COMPLEX*16</td>
<td>16</td>
</tr>
<tr>
<td>RTLXCV</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RTLXARG</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>CVLXRV</td>
<td>local</td>
<td>COMPLEX*15</td>
<td>15</td>
</tr>
<tr>
<td>CTLXLSIN</td>
<td>local</td>
<td>COMPLEX*16</td>
<td>16</td>
</tr>
<tr>
<td>RVLXAREA</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RVLXFRE</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>CTLXNSQ</td>
<td>local</td>
<td>COMPLEX*16</td>
<td>16</td>
</tr>
<tr>
<td>RTLXCRS</td>
<td>local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>CTLXSQT</td>
<td>local</td>
<td>COMPLEX*16</td>
<td>16</td>
</tr>
</tbody>
</table>

280
<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVLXHOR</td>
<td>Local</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVCXLOSG</td>
<td>LOSSES</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXDTOR</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXRE</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXII</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXJJ</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXKK</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXFSQ</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXTIM</td>
<td>LPARM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXCY</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXCZ</td>
<td>MORE</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RVXSEZTX</td>
<td>MISC</td>
<td>REAL*8</td>
<td>72</td>
</tr>
<tr>
<td>RTCX1</td>
<td>MISC</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX2</td>
<td>MISC</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXYEAR</td>
<td>LPARM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX3</td>
<td>MISC</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX4</td>
<td>MISC</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX5</td>
<td>MISC</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX6</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX7</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCX8</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXA100</td>
<td>LPARM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXMIN</td>
<td>MISC</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXH5</td>
<td>END</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVCXCK</td>
<td>MORE</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>OTHER</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXC</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOSA</td>
<td>LOSSES</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVXLOSIR</td>
<td>LOSSES</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IVXSX</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVXSY</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVXSZ</td>
<td>MORE</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXLELEV</td>
<td>LOCAL</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IVXTIM</td>
<td>LPARM</td>
<td>INTEGER*4</td>
<td>10</td>
</tr>
<tr>
<td>IFCXGND</td>
<td>LPARM</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>IVXSSN</td>
<td>LPARM</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVXLOSX</td>
<td>LOSSES</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE GCDEV(IVSxNB, RVSxAZ, RVSxLAO, RVSxLO0, RVSxBou, RVSxDEV)

GCDEV -- SUBROUTINE TO EVALUATE THE DEVIATION OF A RAY FROM
A GREAT CIRCLE PATH

CALLED BY: RAYSUB

AUTHOR: ERIC L. STROBEL

DATE: 10/10/36

VERSION: 2.1

REVISED: 09/05/86 -- INITIAL REVISION.

09/17/86 -- V2.0. Practically a complete rewrite. Now does the spherical trig OK
even in the special cases... I think.

10/10/36 -- V2.1. Changed to conform to
the new status of RAYTRACE as a subrout.

USES: RVSxBOU(11,15) BOUNCE PT. ARRAY
IVSxNB BOUNCE COUNT
RVSxAZ LAUNCH AZIMUTH
RVSxLAO LAUNCH LATITUDE
RVSxLO0 LAUNCH LONGITUDE

TO CALCULATE THE DEVIATION OF THE RAYPATH AWAY FROM ITS
EXPECTED LANDING POINT (IF IT HAD FOLLOWED A
GREAT CIRCLE PATH).

RETURNS:
RVSxDEV DEVIATION IN KM

INTEGER IVSxNB, ITLxS, ITLxC
REAL*8 RVSxDEV, RVSxBou(11,15), RVSxAZ
REAL*8 RVSxLAO, RVSxLO0, RPCxPI, RPCxRE, RPCxDTR
REAL*8 RVLxR, RTLxL, RVLxLAP, RVLxL0P, RVLxSDL
REAL*8 RVLxCDL, RVLxDL, RTLxA, RTLxB, RTLxD, RTCxA
COMMON /PRAM/ RPCxPI, RPCxDTR, RPCxRE, RTCxA
C
First, do the spherical trig to get the great circle landing point (i.e., where the ray would have landed, given the range that it went, if it had stayed on the great circle path at the original azimuth).

RVLxR = RVSxBOU(IVSxNB, 4)
RTLx1 = DSIN(RVSxAZ) * DCOS(RVLxR) + DCOS(RVSxLAO) * DSIN(RVLxR) * DCOS(RVSxAZ)

RVLxLAP = DASIN(RTLx1)
RVLxSDL = DSIN(RVSxAZ) * DSIN(RVLxR) / DCOS(RVLxLAP)
RVLxCDL = (DCOS(RVLxR) - DSIN(RVSxLAO) * DSIN(RVLxLAP)) / (DCOS(RVSxLAO) * DCOS(RVLxLAP))

RTLxA = RVSxBOU(IVSxNB, 1)
RTLxB = RVSxBOU(IVSxNB, 2)
RTLxD = RTLxB - RVLxLOP
RVSxDEV = DSIN(RTLxA) * DSIN(RVLxLAP) + DCOS(RTLxA) * #DCOS(RVLxLAP) * DCOS(RTLxD)

IF ((RVSxDEV - 1.0D00) .GT. 0.0D00) RVSxDEV = 1.0D00
IF ((RVSxDEV + 1.0D00) .LT. 0.0D00) RVSxDEV = -1.0D00
RVSxDEV = RPCxRE * DACOS(RVSxDEV)
RETURN
END

GCDEV Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSxDEV</td>
<td>.</td>
<td>. . . .</td>
<td>.</td>
</tr>
<tr>
<td>RVSxBOU</td>
<td>.</td>
<td>. . . .</td>
<td>.</td>
</tr>
<tr>
<td>RVSxLOQ</td>
<td>.</td>
<td>. . . .</td>
<td>.</td>
</tr>
<tr>
<td>RVSxLAO</td>
<td>.</td>
<td>. . . .</td>
<td>.</td>
</tr>
<tr>
<td>RVSxAZ</td>
<td>.</td>
<td>. . . .</td>
<td>.</td>
</tr>
<tr>
<td>IVSXNB</td>
<td>.</td>
<td>. . . .</td>
<td>.</td>
</tr>
<tr>
<td>ITLxC</td>
<td>.</td>
<td>. . . .</td>
<td>.</td>
</tr>
<tr>
<td>RTLxA</td>
<td>.</td>
<td>. . . .</td>
<td>.</td>
</tr>
</tbody>
</table>

283
## GCDEV Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTLXB</td>
<td>Local</td>
<td>REAL*9</td>
<td>3</td>
</tr>
<tr>
<td>RTLXD</td>
<td>Local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>ITLXS</td>
<td>Local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVLXDL</td>
<td>Local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLRX</td>
<td>Local</td>
<td>REAL*3</td>
<td>8</td>
</tr>
<tr>
<td>RVLXCDL</td>
<td>Local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXLAP</td>
<td>Local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXSDL</td>
<td>Local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXLOP</td>
<td>Local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RTLXI</td>
<td>Local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
<tr>
<td>RPCXRE</td>
<td>PRAM</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RPCXDTR</td>
<td>PRAM</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RTCXA</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE ANRANG(RVSxAZ, RVSxLAl, RVSxLO1, RVSxLA0, RVSxLO0, RVSxR)

ANRANG -- SUBROUTINE TO COMPUTE THE ANGULAR RANGE BETWEEN TWO POINTS

CALLED BY: RAYSUB

AUTHOR: ERIC L. STROBEL
DATE: 03/13/88
VERSION: 2.0

REVISED: 05/01/87 -- V1.0. Initial revision.

03/18/88 -- V2.0. Rewritten to handle using the correct azimuth between the launch point and the current point, as opposed to using the launch azimuth. Also, now correctly handles azimuths of 0 & 180 degrees, as well as angular ranges > 180 degrees.

USES:
RVSxAZ Azimuth.
RVSxLAl \ RVSxLO1 \ RVSxLA0 ; The endpoints of the path.
RVSxLO0 /

To calculate the angular range along the great circle path between the two points.

RETURNS: RVSxR The angular range.

REAL*8 RTLxl, RVSxAZ, RVSxLAl, RVSxLO1, RVSxLA0, RVLxTHC
REAL*8 RVSxLO0, RVSxR, RVLxCR, RPCxPI, RVLxPIO2, RVLxTH1
REAL*8 RTCxA, RTCxB, RTCxC, RVLxS0, RVLxC0, RVLxS1, RVLxC1
REAL*8 RVLxS2, RVLxC2, RVLxTAD, RVLxAC1, RVLxAC2, RVLxTAN
REAL*8 RVLxAC, RVLxD1, RVLxD2
INTEGER RVLSxS, RVLSxC

C

COMMON /PRAM/RPCxPI, RTCxA, RTCxB, RTCxC

RVLxPIO2 = RPCxPI/2.0D00
RTLx1 = RVSxLO1 - RVSxLO0
RVLxTH0 = RVLxPIO2 - RVSxLA0
RVLxTH1 = RVLxPIO2 - RVSxLA1
RVLxS0 = DSIN(RVLxTH0)
RVLxC0 = DCOS(RVLxTH0)
RVLxS1 = DSIN(RVLxTH1)
RVLxC1 = DCOS(RVLxTH1)
RVLxS2 = DSIN(RRTLx1)
RVLxC2 = DCOS(RRTLx1)
RVLxC2 = RVLxC2*RVLxS0*RVLxS1 + RVLxC0*RVLxC1
RVLxTAD = RVLxC1 - RVLxC2*RVLxS0
RVLxCR = RVLxC2*RVLxS0*RVLxS1 + RVLxC0*RVLxC1

C

We're trying to push spherical trig to yield correct values
even when the sides of the triangle are greater than 180 deg.
Here, the correct azimuth between the endpoints of the range
is being calculated. In this way, if the range is greater
than 180 degrees, we can get enough information to determine
the correct value.

IF (RVLxTAD.EQ.0.D0) THEN
RVLxAC1 = RVLxPIO2
RVLxAC2 = 3.0*RVLxPIO2
ELSE IF (RVLxS2.EQ.0.D0) THEN
RVLxAC1 = 0.D0
RVLxAC2 = RPCxPI
ELSE
RVLxTAN = RVLxS0*RVLxS1*RVLxS2
RVLxAC = DATAN(RVLxTAN/RVLxTAD)
IF (RVLxAC.LT.0.D0) RVLxAC = RVLxAC + RPCxPI
RVLxAC1 = RVLxAC
RVLxAC2 = RVLxAC + RPCxPI
ENDIF
RVLxD1 = DABS(RVSxAZ - RVLxAC1)
RVLxD2 = DABS(RVSxAZ - RVLxAC2)
IF (RVLxD1.LT.RVLxD2) THEN
RVSxAZ = RVLxAC1
ELSE
RVSxAZ = RVLxAC2
ENDIF
Now calculate the angular range.

IF (RVSXAZ.EQ.0.0 .OR. RVSXAZ.EQ.RPCXPI) THEN

RVSXR = (RVLXTH0 - RVLXC2*RVLXTH1)*DCOS(RVSXAZ)

ELSE

IVLXSS = INTSIGN(RVLXS2*RVLXS1/DSIN(RVSXAZ))

RVSXR = DACOS(RVLXCR)

ENDIF

Range will be negative if greater than 130 degrees.

Therefore the following line will give the desired value.

IF (RVSXR.LT.0.0) RVSXR = 2.3DO0*RPCXPI + RVSXR

RETURN

END

ANRANG Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RVSXR</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLOO</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLA0</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLO1</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLA1</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXAZ</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVLXC1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXD1</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXAC1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXC2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXD2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXAC2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXS0</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXAC</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXS1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXS2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXTH0</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXTH1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXCR</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXTAD</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RVLXPIO2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>IVLXSS</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
</tr>
<tr>
<td>RVLXTAN</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLX1</td>
<td>local</td>
<td>REAL*3</td>
<td>3</td>
</tr>
<tr>
<td>RVLXCO</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RPCXPI</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>8</td>
</tr>
</tbody>
</table>
### ANRANG Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCXA</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXB</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTCXC</td>
<td>PRAM</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
SUBROUTINE INBOX(RVSXLAI, RVSXLOI, RVSXLAS, RVSXLOS, RVSXLAE, RVSXLOE, IVSOUT)

INBOX -- SUBROUTINE TO DETERMINE WHETHER A GIVEN PT. IS WITHIN THE IONOSPHERIC SPECIFICATION GRID.

CALLED BY: IONOPAR

AUTHOR: ERIC L. STROBEL

DATE: 03/01/87

VERSION: 1.0

REVISED: 03/01/87 -- V1.0. Initial revision.

USES: RVSXLAI, LOI The coord. of the pt. of interest.

RVSXLAS \ RVSXLOE / The starting and ending (SW & NE) corners of the specification grid.

To determine whether or not the point of interest lies within the grid over which the ionosphere is specified. If it is outside, then the program will use a spherically symmetric ionosphere having the parameters of the nearest grid point, otherwise the program will do the interpolation.

RETURNS: IVSOUT A flag specifying whether the pt. of interest is in or out of the grid. (0 = in, 1 = out)

REAL*8 RVSXLAI, RVSXLOI, RVSXLAS, RVSXLOS
REAL*8 RVSXLAE, RVSXLOE
REAL*8 RTLx1, RTLx2, RTLx3, RTLx4

INTEGER IVSXOUT

RTlx1 = MIN(RVSXLOS, RVSXLOE)
RTlx2 = MAX(RVSXLOS, RVSXLOE)
RTlx3 = MIN(RVSXLAS, RVSXLAE)
RTlx4 = MAX(RVSXLAS, RVSXLAE)

IF (((RVSXLAI-RTLx4).LE.0.0D00).AND. ((RVSXLAI-RTLx3).GE.0.0D00)).AND. ((RVSXLOI-RTLx2).LE.0.0D00)).AND. ((RVSXLOI-RTLx1).GE.0.0D00)) THEN
IVSXOUT = 0
ELSE
IVSXOUT = 1
ENDIF
RETURN

INBOX Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVSXOUT</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLOE</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLAE</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLOS</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLAS</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLOI</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RVSXLAI</td>
<td>param</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTLx1</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLx2</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLx3</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
<tr>
<td>RTLx4</td>
<td>local</td>
<td>REAL*8</td>
<td>3</td>
</tr>
</tbody>
</table>
FUNCTION INTSIGN(R)

GIVES 1, -1, or 0 as the sign of a REAL*8 number. This mimics the SIGN function available in most BASICs.

REAL*8 R

INTEGER INTSIGN

IF (R.EQ.0.0D00) THEN
    INTSIGN = 0
    RETURN
ENDIF

INTSIGN = IDNINT(R/DABS(R))

RETURN
END

INTSIGN Local Symbols

Name       Class Type Size
------------- ------- ------
R           param
INTSIGN     param

Global Symbols

Name       Class Type Size
------------------------ ------
ACCFSP      FSUBRT ***
ANRANG      FSUBRT ***
END         common ***
FREESP      FSUBRT ***
GCDEV       FSUBRT ***
GORP        common ***
INBOX       FSUBRT ***
INTSIGN     PPUNCT INTEGER*4 ***
ION01       common ***
LOCAL       common ***
LOSS        FSUBRT ***
LOSSES      common ***
LPARM       common ***
MAINDAT     common ***
MISC        common ***
MORE        common ***
NEWCS       FSUBRT ***
OTHER       common ***
PRA M       common ***
START       common ***
Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMP2</td>
<td>common</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>TIMES</td>
<td></td>
<td>FSUBRT</td>
<td></td>
</tr>
<tr>
<td>TRIANG</td>
<td></td>
<td>FSUBRT</td>
<td></td>
</tr>
</tbody>
</table>

Code size = 1f47 (3007)
Data size = 0116 (278)