The Segmented Waveguide Program for Long Wavelength Propagation Calculations

J. A. Ferguson
F. P. Snyder

Naval Ocean Systems Center
San Diego, California 92152-5000

Approved for public release; distribution is unlimited.
NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152-5000

E. G. SCHWEIZER, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work was performed by Code 544 of Naval Ocean Systems Center and funded by the Defense Nuclear Agency.

Released by
J.A. Ferguson, Head
Modeling Branch

Under authority of
J.H. Richter, Head
Ocean and Atmospheric Sciences Division
**The Segmented Waveguide Program for Long Wavelength Propagation Calculations**

A computer program which obtains waveguide mode solutions for very low frequencies and low frequencies (VLF/LF) is described. The program allows for multiple homogeneous segments to be specified, allowing for consideration of variations in the earth-ionosphere waveguide. Path geometry and geophysical parameters can be computed by the program. Ionospheric disturbances due to man-made or naturally occurring events can also be modeled using the program.
CONTENTS

Introduction .................................................................................... 1
Program Control .................................................................................. 2
Calculation of Mode Parameters .......................................................... 4
  Path Calculations ........................................................................... 8
  Path Geometry ............................................................................... 8
  Presegmentation .......................................................................... 9
  Mode Tracing ............................................................................. 10
Output .............................................................................................. 11
Sample Input .................................................................................... 12
References ...................................................................................... 14

APPENDIX

Listing of the program ..................................................................... A-1

FIGURES

1. Sample Input using COORD option ........................................... 12
2. Sample input using PRESEG option ......................................... 13

TABLES

1. Summary of Control Strings ..................................................... 2
2. NAMELIST Inputs ..................................................................... 4
INTRODUCTION

The model of longwave propagation developed at the Naval Ocean Systems Center (NOSC) is based on a waveguide mode formulation. To determine signal levels in this approach the basic problem is to obtain the modal solutions to the specific waveguide under consideration (Pappert et al., 1970, Morfitt and Shellman, 1976), where complicated propagation paths are divided into horizontally homogeneous segments. The parameters of the segments are determined by the earth’s ground conductivity, the magnitude and orientation of the geomagnetic field with respect to the direction of propagation, and the state of the ionosphere. Well-known computer programs which make the necessary calculations for a single set of propagation parameters are “WAVEGUID” (Pappert et al., 1970) and “MODESRCH” (Morfitt and Shellman, 1976). The WAVEGUID computer program and related programs are described in a series of reports and familiarity with the important elements of this series is assumed (Pappert, Gossard and Rothmuller, 1967; Sheddy et al., 1968; Pappert, Moler and Shockey, 1970; Morfitt and Shellman, 1976). This report describes a modified version of that program designed to supply calculations of long wavelength propagation along segmented propagation paths. This program is called the Segmented Waveguide (SW).

The computer program obtains waveguide mode solutions for very low frequencies and low frequencies (VLF/LF). The program allows for multiple homogeneous segments to be specified, allowing for consideration of variations in the earth-ionosphere waveguide. Path geometry and geophysical parameters can be computed by the program. Ionospheric disturbances due to man-made or naturally occurring events can also be modeled using the program.

Essential features of this program include:

- Automatic segmentation of the propagation path
- Allowance for presegmentation of the propagation path
- Allowance for variation of the ionosphere along that path

The program operates on propagation paths defined by a transmitter location and either a direction or a receiver location. A propagation path is defined as the great circle on a spherical earth. Variation of the geophysical parameters are to be expected along realistic paths. The diurnal condition in large part determines the significance of the other parameters. For instance, under daytime conditions the effect of variation of the geomagnetic field along a path is usually not significant to the resulting mode parameters. The program incorporates routines for calculating the parameters of the geomagnetic field and for selecting the ground conductivity at any point on the earth’s surface.

Each of the path segments is treated as a horizontally homogeneous planar waveguide. Earth curvature is introduced by use of a modified refractive index. A set of possible solutions to the waveguide mode equation must be input. Each of these solutions is processed by an iteration routine. Each iteration requires computation of ionospheric and ground reflection coefficients. Calculation of the ionospheric reflection coefficients requires integration of the coefficients through the ionosphere. An approximate formulation may be used which requires a secondary set of complex angles be specified by the user. In that case, the ionospheric reflection coefficients are calculated for the secondary set of angles. These coefficients are used to interpolate the ionospheric reflection coefficients during the iteration of the primary set of possible solutions. This interpolation procedure requires much less computation time than does the more exact procedure.
The set of solutions for the first homogeneous segment must be input by the user. The program uses the results of up to three consecutive segments to extrapolate the solutions for successive segments. This reduces the number of iterations which are required for subsequent segments and allows for the tracing of mode solutions through a wide range of path variations.

The primary output of "SW" is data which may be used in mode summing programs. The strength of the electromagnetic field along the path can be obtained with either of two mode conversion models, one denoted "FULLMC" (Pappert and Shockey, 1972) and the other denoted "FASTMC" (Ferguson and Snyder, 1980). "FULLMC" does integrations through the ionosphere prior to calculating mode conversion coefficients and can be quite slow in execution time whereas "FASTMC" avoids the transionospheric integrations by use of approximations and runs quite quickly.

PROGRAM CONTROL

Program execution is controlled by strings containing mnemonic words. These strings and the sequence of operations initiated by them are described below. These control strings, as well as variable names and names of subroutines, appear in upper case. For clarity, the control strings are enclosed in single quotes and names of subroutines are enclosed in double quotes. Table 1 summarizes these control strings.

Table 1. Summary of control strings.

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>EIGEN</th>
<th>PROFILE</th>
<th>COLFREQ</th>
<th>COORD</th>
<th>PRESEG</th>
<th>QUIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>run identification</td>
<td>general NAMELIST data</td>
<td>alternate EIGEN list input</td>
<td>ionspheric specie profiles (i is 1 or 3)</td>
<td>ionspheric collision frequency profiles</td>
<td>automatic segmentation of propagation path</td>
<td>presegmented propagation path</td>
<td>end of job</td>
</tr>
</tbody>
</table>

'ID' indicates that the next line of data to be read is a general identification for the path under consideration. This identification appears in the printout. It is also written to the mode parameter output file.

'NAME' initiates the reading of general program data via the NAMELIST: DATUM. Table 2 lists all variables, their type (I indicates integers, R indicates real (floating point) variables, C indicates complex variables), their units where applicable, and their initial values. If a variable defines an array, then the dimension of the array follows the name in parentheses. The NAMELIST input format is quite flexible but requires that column 1 be blank. Variable names are followed by an equal sign and then by the value of the variable. Array variable names may be followed by a string of values separated by commas and/or spaces. Embedded blanks are not allowed in the variable names. Variable types must be considered; for example, values for integer variables may not contain decimal points, but values for real variables do not have to have decimal points. Logical variables may be specified with any of '.true.' '.t.' 't' '.false.' '.f.' 'f'. The values of character variables must be enclosed within quotes. The first record of the NAMELIST input must contain '&name' where 'name' is the
NAMELIST name (in this case, DATUM). The end of the NAMELIST is indicated by ' &end'. Some of the values to be found in the following text are of the form A(N) to indicate A to the power N.

'EIGEN' allows for input of the list of trial solutions to be made from a file. The format of the input is the same as with NAMELIST. The control string is followed by the name of the file (starting in column 9) containing the NAMELIST data. The source of these input solutions could be a previous run with the program or from one of the automatic mode searching programs such as the one described by Morfitt and Shellman (1976).

'PROFILE i' initiates reading of the ionospheric charged particle profile data used to model the upper boundary of the earth-ionosphere waveguide. The value of i indicates the number of ionospheric species to be used and it must have one of two possible values: 1 is for electrons only and 3 is for electrons and ions. If i is not specified, a value of 1 is assumed. The use of i is shown below. The 'PROFILE i' string is immediately followed by a single line of identification for the profile. The profile is input starting at the top of the ionosphere using a formatted input. Each line contains the height in km, the electron density at that height in electrons per cubic centimeter, and if ions are to be considered, the positive ion density at that height in ions per cubic centimeter. The height is in columns 1-7 and the electron and ion densities are required to be in columns 14-21 and 24-31, respectively. The end of the profile is indicated by a dummy height with value less than zero. A maximum of 50 heights may be used. If i is 1, then only the electron density is read. Consequently, only the electron density need be present in the data. If i is 3, then the electron and positive ion densities are read and the negative ion density is computed by subtracting the electron density from the positive ion density (to preserve charge neutrality).

In the integration of the reflection elements through the ionosphere the program interpolates exponentially between input heights. The profile should contain sufficient data to define the ionospheric structure with height. For example, an exponential profile should consist of only the top and bottom heights and densities. Many regularly spaced heights tend to slow the integration.

A purely exponential conductivity profile (electrons only) may be input via the NAMELIST variables BETA and HPRIME.

Additional specie parameters are needed for the waveguide mode computations and are described below.

'COLFREQ' initiates reading of an ionospheric collision frequency profile. This allows use of nonexponential collision frequencies. The 'COLFREQ' string is immediately followed by the collision frequency profile, starting with the highest height and ending with a dummy height of value less than zero as in the case of specie profile described above. These heights need not match those used under 'PROFILE i'. The format of the data is the same as used for 'PROFILE i' except that collision frequencies for all species must be input since the negative ion collision frequency cannot be simply computed from the other two. If only electrons are being used, then only that collision frequency need be present. As with the specie profiles, the program interpolates exponentially between input heights.

An exponential collision frequency specification may be input via the NAMELIST variables EXPNU and COEFNU.
'COORD' initiates automatic segmentation of the propagation path. This string must be placed after all pertinent data have been read. This option is best applied to simple cases such as all daytime. The basic input consists of the path specification, the environment, and the starting mode solutions.

'PRESEG' allows for previously determined segments to be used along the propagation path. This option requires most of the same inputs as 'COORD' except that the user supplies the distances at which segments begin. At each segment the user has the option of specifying the parameters of the geomagnetic field, the ground and the ionosphere.

**CALCULATION OF MODE PARAMETERS**

The inputs to the mode equation computations are supplied by geophysical routines and/or by the user. The subroutine which controls the calculations is "WVGUID". Most user supplied data values are input to the program via NAMELIST. These parameters are summarized in table 2 and are discussed in more detail below. In table 2 the data types are Integer (I), Real (R), and Complex (C).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ</td>
<td>R</td>
<td>0.0</td>
<td>Frequency in kHz</td>
</tr>
<tr>
<td>RHO</td>
<td>R</td>
<td>0.0</td>
<td>Distance from the transmitter of the current segment in Mm.</td>
</tr>
<tr>
<td>AZIM</td>
<td>R</td>
<td>0.0</td>
<td>Magnetic azimuth angle in degrees east of magnetic north.</td>
</tr>
<tr>
<td>CODIP</td>
<td>R</td>
<td>0.0</td>
<td>Magnetic co-dip angle in degrees.</td>
</tr>
<tr>
<td>MAGFLD</td>
<td>R</td>
<td>0.0</td>
<td>Intensity of the geomagnetic field in Webers/square meter.</td>
</tr>
<tr>
<td>SIGMA</td>
<td>R</td>
<td>4.64</td>
<td>Ground conductivity in Siemens/meter.</td>
</tr>
<tr>
<td>EPSR</td>
<td>R</td>
<td>81.0</td>
<td>Dielectric constant of the ground.</td>
</tr>
<tr>
<td>BETA</td>
<td>R</td>
<td>0.0</td>
<td>Slope of the exponential profile in km(-1).</td>
</tr>
<tr>
<td>HPRIME</td>
<td>R</td>
<td>0.0</td>
<td>Reference height of the exponential profile in km.</td>
</tr>
<tr>
<td>TLONG,TLAT</td>
<td>R</td>
<td>0.0,0.0</td>
<td>Transmitter coordinates in degrees.</td>
</tr>
<tr>
<td>RBEAR</td>
<td>R</td>
<td>720.0</td>
<td>Geographic bearing of the path in degrees.</td>
</tr>
<tr>
<td>RLONG,RLAT</td>
<td>R</td>
<td>0.0,0.0</td>
<td>Receiver coordinates in degrees.</td>
</tr>
<tr>
<td>DRMIN</td>
<td>R</td>
<td>0.125</td>
<td>Minimum distance step size in Mm.</td>
</tr>
<tr>
<td>DRMAX</td>
<td>R</td>
<td>0.5</td>
<td>Maximum distance step size in Mm.</td>
</tr>
<tr>
<td>DMAX</td>
<td>R</td>
<td>20.0</td>
<td>Maximum distance in Mm.</td>
</tr>
<tr>
<td>YEAR</td>
<td>I</td>
<td>0</td>
<td>Year</td>
</tr>
<tr>
<td>MONTH</td>
<td>I</td>
<td>0</td>
<td>Month number with January being 1</td>
</tr>
<tr>
<td>DAY</td>
<td>I</td>
<td>0</td>
<td>Day of the month</td>
</tr>
<tr>
<td>GMT</td>
<td>R</td>
<td>0.0</td>
<td>Greenwich meridian time in hours.</td>
</tr>
<tr>
<td>NPRINT</td>
<td>I</td>
<td>1</td>
<td>Flag used to control the amount of print out</td>
</tr>
<tr>
<td>NPROF</td>
<td>I</td>
<td>1</td>
<td>Flag to indicate which form of the profile is to be used</td>
</tr>
</tbody>
</table>

Table 2. NAMELIST Inputs.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDPNT</td>
<td>I</td>
<td>0</td>
<td>Flag to indicate that only the midpoint is to be considered</td>
</tr>
<tr>
<td>IGCD</td>
<td>I</td>
<td>0</td>
<td>Flag to indicate that the computed distance between the transmitter and receiver is to be used</td>
</tr>
<tr>
<td>IGND</td>
<td>I</td>
<td>0</td>
<td>Flag to indicate that the ground conductivity is to be determined from the ground map</td>
</tr>
<tr>
<td>MDIR</td>
<td>I</td>
<td>0</td>
<td>Flag used to reverse the direction of the magnetic field</td>
</tr>
<tr>
<td>EIGEN(60)</td>
<td>C</td>
<td>0.0,0.0</td>
<td>Initial solutions for the waveguide modes in degrees.</td>
</tr>
<tr>
<td>TLIST(30)</td>
<td>C</td>
<td>0.0,0.0</td>
<td>Angles where the reflection coefficients are computed for the inexact interpolation routine in degrees</td>
</tr>
<tr>
<td>DTHETA</td>
<td>C</td>
<td>0.01,0.001</td>
<td>Change in the mode solution used to define the iteration in degrees</td>
</tr>
<tr>
<td>LUB</td>
<td>C</td>
<td>0.05,0.005</td>
<td>Tolerance used to test the differential change in the mode solution. Used to stop the iteration</td>
</tr>
<tr>
<td>DEIGEN</td>
<td>C</td>
<td>0.05,0.005</td>
<td>Tolerance used to define duplicate modes in degrees</td>
</tr>
<tr>
<td>FTOL</td>
<td>R</td>
<td>1.0</td>
<td>Tolerance used to determine if the mode equation has been satisfied by the solutions</td>
</tr>
<tr>
<td>THTINC</td>
<td>R</td>
<td>0.5</td>
<td>Maximum change in degrees of either the real or imaginary part of the mode solution from one iteration to the next</td>
</tr>
<tr>
<td>MAXITR</td>
<td>I</td>
<td>7</td>
<td>Maximum number of iterations to attempt</td>
</tr>
<tr>
<td>ALPHA</td>
<td>R</td>
<td>3.14x10^(-4)</td>
<td>The earth curvature correction factor in km^(-1)</td>
</tr>
<tr>
<td>H</td>
<td>R</td>
<td>50.0</td>
<td>Height at which the modified refractive index is unity in km</td>
</tr>
<tr>
<td>D</td>
<td>R</td>
<td>0.0</td>
<td>Height at which the integration through the ionosphere is stopped in km</td>
</tr>
<tr>
<td>PREC</td>
<td>R</td>
<td>2.0</td>
<td>Factor which controls the precision of the reflection coefficient integration</td>
</tr>
<tr>
<td>WR0</td>
<td>R</td>
<td>2.5x10^5</td>
<td>Value of omega sub r used to define the reference height</td>
</tr>
<tr>
<td>ATNMAX</td>
<td>R</td>
<td>50.0</td>
<td>Maximum attenuation rate of modes to be retained in dB/Mm</td>
</tr>
<tr>
<td>DEBUG</td>
<td>I</td>
<td>0</td>
<td>Flag used to generate additional printout for debugging purposes</td>
</tr>
<tr>
<td>TYPITR</td>
<td>I</td>
<td>0</td>
<td>Flag used to define the form of the mode equation</td>
</tr>
<tr>
<td>RPOLY</td>
<td>I</td>
<td>1</td>
<td>Flag used to define the reflection coefficient calculation</td>
</tr>
<tr>
<td>NRTLST</td>
<td>I</td>
<td>5</td>
<td>Number of points to use in the interpolation of the reflection coefficients during inexact iterations</td>
</tr>
<tr>
<td>LUNIT7</td>
<td>I</td>
<td>7</td>
<td>Logical unit number to which the mode parameters data are output</td>
</tr>
<tr>
<td>CHARGE(3)</td>
<td>R</td>
<td>-1.0, 1.0</td>
<td>Charge of the ionospheric species</td>
</tr>
<tr>
<td>MRATIO(3)</td>
<td>R</td>
<td>1.0, 2.5x10^4</td>
<td>Ratio of the mass of the ionospheric species to that of electrons</td>
</tr>
<tr>
<td>COEFNU(3)</td>
<td>R</td>
<td>1.816x10^11, 2.5x10^10</td>
<td>Collision frequency of the ionospheric species at the ground in collisions/sec</td>
</tr>
<tr>
<td>EXPNU(3)</td>
<td>R</td>
<td>-0.15, 2.0^-0.15</td>
<td>Exponential slope of the collision frequency in km^(-1)</td>
</tr>
</tbody>
</table>
The radio frequency in kHz is specified by the variable FREQ. The input to subroutine "WVGUID" includes an ionospheric profile. The variable NPROF controls which ionospheric profile is to be used. A value of 0 indicates that the profile input via 'PROFILE i' is to be used. A value of 1 indicates that an exponential electrons only profile is to be used. The profile is specified by the exponential slope BETA in km(-1) and a reference height HPRIME in km (Wait and Spies, 1964). A value of 2 for NPROF is used to indicate that a series of profiles will be input. This option only applies to 'PRESEG'. The profiles are to be input using the same format as described under 'PROFILE i', including the control string. There must be a profile for each segment.

Additional specie parameters are needed. The number and order of these specie parameters must be consistent with the charged particle densities of 'PROFILE i'. The charges of the species are input as CHARGE (i.e., CHARGE = -1, +1, -1). The masses of the species relative to that of an electron are input as MRATIO. The collision frequencies may be defined with 'COLFREQ' (nonexponential) or with the variables COEFNU in collisions per second and EXPNU in km(-1). The collision frequency \( v \) at an altitude \( z \) is then defined by

\[
v = \text{COEFNU} \times \exp(\text{EXPNU} \times z)
\]

where \( z \) is in km.

Parameters of the geomagnetic field are specified by AZIM, the angle between magnetic north and the direction of propagation in the horizontal plane measured in degrees east of north, CODIP, the magnetic co-dip angle measured from the vertical (i.e., the north pole has a CODIP of 0), and MAGFLD, the magnetic intensity in Webers per square meter or in Gauss. The magnitude of MAGFLD is tested. If it is greater than 10(-2) then the input value is assumed to be in Gauss and is multiplied by 10(-4).

MDIR is a flag that when set to 1, causes the direction of propagation as input to be reversed. This allows for development of a data set appropriate to examining transmitter deployment.

Ground conditions are specified by SIGMA, the conductivity in Siemens/m, and EPSR, the relative dielectric constant.

The presegmentation option allows the magnetic field and ground parameters to be varied by the user.

The correction for earth curvature is controlled by ALPHA in km(-1) which is defined as 2 over the radius of the earth. For a curved earth ALPHA is 3.14x10(-4) km(-1) and for a flat earth ALPHA is 0.

Ionospheric altitude parameters are H, which is the height in km at which the modified refractive index is unity and is the height to which the mode solutions are referenced; D, which is the height in km below which ionospheric effects can be ignored. D must be equal to or greater than the bottom height of the ionospheric profiles used. It is usually sufficient to choose H equal to D. The choice of D and H is also discussed by Pappert et al. (1967).
The trial eigenangles follow the variable name EIGEN which is a complex variable. Up to 60 eigen angles may be input. If little is known about the expected solutions for a given set of conditions, a set of approximate solutions may be obtained using a TLIST. The TLIST is a list of as many as 30 complex angles which are used to set up an interpolation matrix of the ionospheric reflection elements (Sheddy et al., 1968). The program then uses this matrix to interpolate reflection elements during the iterative process used to obtain mode solutions. These solutions are referred to as "inexacts" in order to distinguish them from the more accurate solutions using integrated reflection coefficients. The variable NRTLST determines the maximum number of TLIST angles used in each interpolation. During the inexact iteration process, the program computes the magnitude of the complex difference between the current value of the solution and each of the TLIST angles. The program orders the TLIST angles from the smallest to the largest difference and selects the first NRTLST of them to be used in the interpolation. This improves the accuracy of the interpolated reflection coefficients and reduces the number of terms used.

If more than 30 EIGENs are input, then the program sorts the angles according to their attenuation rate and deletes those with attenuation rates greater than a user specified maximum. The initial value of this maximum is ATNMAX. If there are still more than 30 angles, then the maximum attenuation rate is reduced by 5 dB Mm and the input list is sorted again. This process is repeated until there are less than 30 angles in the list.

If the number of EIGEN or TLIST inputs varies from one NAMELIST to the next, then each EIGEN or TLIST list should be terminated with a zero. If RPOLY is not 0 and the first value of TLIST is 0, then TLIST is set equal to the first 30 EIGENs.

The Newton-Raphson iteration process, used to find the eigenangles which satisfy the modal equation is described by Sheddy et al. (1968). Iteration is performed for each input EIGEN. The iteration stops when the maximum number of iterations (MAXITR) is exceeded or when the change in the real and imaginary parts of the solution is calculated to be less than the real and imaginary parts of LUB, respectively.

The type of solution obtained is determined, in part, by RPOLY, which can have three values: 0 for exact solutions only, 2 for inexact (approximate) solutions only, and 1 for inexacts computed and used as inputs to obtain exact solutions. The use of RPOLY equal to 1 is described more fully below.

The flag TYPITR is used to obtain vertically polarized modes only (TYPITR equal to 1) or horizontally polarized modes only (TYPITR equal to 2). It is physically meaningful to apply this option only for nearly isotropic conditions, no magnetic field (MAGFLD set to 0), or east to west and west to east propagation at the geomagnetic equator (CODIP is 90 and AZIM is 90 or CODIP is 90 and AZIM is 270).

To ensure consistent mode sums and eliminate redundant solutions, each exact and inexact EIGEN solution is tested for several conditions. The first is that the imaginary part of the solution must be less than zero in order to have attenuating modes. The second is that the magnitude of the modal equation must be less than FTOL. This parameter is tested only for final solutions and only if the number of iterations required to obtain the solution is greater than or equal to MAXITR. It is generally true that if the iteration stops because the change in the mode solution is less than LUB, then the value of the mode equation is small. There are instances in which the test on FTOL will still fail. Consequently, the value of FTOL is set very high in order to allow the
program to continue execution. In some cases the user may want to modify the
default value in order to perform special tests. If RPOLY is 1, the inexacts are
treated as intermediate results. The third test is that the value of the EIGEN solution
must be different from all previous solutions by an amount DEIGEN which is input as
a complex number. The real and imaginary parts of DEIGEN are the tolerances for
the real and imaginary parts of the EIGEN solutions, respectively. If one of the above
tests results in a mode being dropped from the list of solutions, then the program
follows the procedures outlined below under the discussion of mode tracing.

Subroutine "WVGUID" computes and prints attenuation rate in dB/Mm, phase
velocity relative to the speed of light, the magnitude, and phase of Wait's excitation
factor (Wait, 1962) at the ground in dB and radians.

The headings for the number of iterations to go from the input angle to the final
solution, the final solution, the magnitude of the modal equation, and the magnitude
of the polarization mixing ratio are printed as ITER, EIGEN, MAG F, and MAG P,
respectively. The attenuation rate, phase velocity relative to the speed of light,
magnitude, and phase of Wait's excitation factor, and the final solution references to
the ground are printed under the headings of ATTEN, V/C, WAIT'S EXC, and
THETA', respectively.

The parameters YEAR, MONTH, DAY, and GMT are used only to pass the values
to the output files. These parameters are useful for helping to identify the output
data to programs which may use this information.

PATH CALCULATIONS

These calculations are controlled by "GCPATH". They can be divided into three
classes. The first automatically computes geometry and geophysical parameters which
are obtainable from just the location of a point on the propagation path or uses
presegmented distances and geophysical parameters. In addition, the program
extrapolates the EIGEN list and TLIST so as to trace modes along a path.

PATH GEOMETRY

All geometry calculations are for a spherical earth. Inputs to this portion of the
program consist of transmitter and receiver locations, path length, and path increments.
Transmitter and receiver longitude and latitude are input with TLONG and RLONG and
TLAT and RLAT. The convention used in the program is east longitude and south
latitude are negative. An alternate input for the receiver position is its bearing,
RBEAR, in degrees east of north. In the execution of the program RBEAR is tested.
If RBEAR is 720, then the program uses RLONG and RLAT to define the path. If
RBEAR is not 720, then it and the input path length, DMAX, are used to define the
path. DMAX is specified in megameters and must be less than or equal to 20. If
RLONG and RLAT are used, there are two path lengths possible. If the parameter
IGCD is equal to 1, the path length is set equal to the computed short great circle
distance between the transmitter and receiver. If IGCD is equal to 0, the path length
is unchanged from what was input (DMAX). If the path bearing is input, the path
length is always DMAX.

The starting value of the distance from the transmitter is input as RH0 and is in
Mm. The path increments are controlled by the mode tracing results and the variables
DRMIN and DRMAX which are all in units of Mm. The procedures of the values are
described as follows.
The geomagnetic field is computed at the first path point defined by $r_{ci} - r_{a}$ at the beginning of each path segment. The ground conductivity and relative dielectric constant are specified by the user through SIGMA and EPSR via NAMELIST or by searching the ground map. If IGND is 0, the ground map is not searched and ground conditions are assumed constant, as input, for the entire path for 'COORD' or as varied by the user for 'PRESEG'. If IGND is 1, the DECO-NRL 10 level ground conductivity map (Hauser, Garner, and Rhoads, 1969) is searched for the appropriate values of SIGMA and EPSR at the beginning of each path segment.

If the value is assumed that the entire propagation path can be described adequately by the conditions at the midpoint of the path, then the path conditions at that point can be obtained if MIDPNT is set to 1. The subsequent modal calculations will then be for the midpoint conditions.

At the transmitter and at the end of each path segment, the parameters to be used in the "WVGUID" calculations are printed next to the heading, PROPAGATION PATH PARAMETERS, as described below. In addition, the distance in megameters from the transmitter, the coordinates and the geographic bearing of the path at the current point are printed under the headings RHO, LAT, and BEAR, respectively. If the midpoint option is being used, then the above information for the midpoint is printed.

### PRESEGMENATION

In some instances, user segmentation of the propagation path is desired. The control string 'PRESEG' allows arbitrary segmentation of the path. This is accomplished by a succession of data lines in list directed format containing values for path distance in Mm, AZIM, CODIP, MAGFLD, SIGMA, EPSR, BETA, and HPRIME, respectively. List directed input is accomplished by entering values separated by commas or spaces. There must a data entry for each request in the input list. If a value is not to change from one data line to the next, then the value need not be entered but its omission must be indicated by a pair of commas. The first value of path distance need not be zero. The presegmentation is terminated by a distance value of 40.

If NPROF is 0, the ionospheric profile is constant for the path and is defined by 'PROFILE i'. If NPROF is 1, then the values of BETA and HPRIME on the presegmentation data lines are examined. If BETA is zero, then the previously defined values of BETA and HPRIME are used. The latter may be input via NAMELIST so that a constant ionosphere for the path can be obtained by using NAMELIST input. If either of the values of BETA and HPRIME are to change, both must be input. If NPROF is 2, then each presegmentation data line must have a corresponding 'PROFILE i' profile specification on logical unit 3.

The following conventions are used for using the values of the presegmentation data. If the value of MAGFLD is zero or blank, then the magnetic field parameters are calculated. If a nonzero value is specified, then all of the magnetic field parameters are taken from the presegmentation data. If constant magnetic parameters are desired along the path, the values must be specified on each presegmentation data line. If the value of SIGMA is not entered, then the previously specified value of SIGMA and EPSR are used. Otherwise, the values of these parameters are taken from the presegmentation data line. Constant values for the whole path may be specified via NAMELIST or in the first presegmentation card. If IGND is 1, then the ground
map is searched and the values of SIGMA and EPSR on the cards are ignored. If BETA is not entered, the currently defined values are used for the electron density profile. Even if only one value in the pairs SIGMA, EPSR, and BETA, HPRIME is to be changed, both values must be specified.

MODE TRACING

Efficient computation of mode parameters along the propagation path is best achieved by using RPOLY set to 1, which will be assumed for the rest of this discussion. At the first point on the path, solutions are best obtained by using a TLIST composed of angles which are believed to be approximately correct and an EIGEN list of many regularly spaced angles such as 88, -1, 87, -1, 86, -1, etc. Alternatively, the EIGEN list should be the list of approximately correct solutions with the TLIST set to zero or the 'EIGEN' control string could be used to specify solutions from some other source. The program computes inexact solutions for the conditions at the first point on the path. After exhausting the EIGEN list, obtaining inexact solutions, and deleting of those failing the tests discussed above, exact solutions are computed using the results of the inexact solutions.

Now the discussion must be separated for the two-path segmentation options. For the 'COORD' option, the second point on the path is DRMIN from the transmitter. For this point, the final solutions for the first point are placed in both TLIST and EIGEN and the same process of calculation of inexacts and exacts is repeated. If DRMIN is not too far from the transmitter and/or the geophysical parameters do not change too much, then this step in the extrapolation process is quite efficient. Now the program has two sets of final solutions and makes a linear extrapolation for TLIST and EIGEN angles for the third point on the path which is twice DRMIN from the transmitter. The sequence of calculations for the inexact and exact solutions is repeated. For the fourth and all subsequent points on the propagation path, the program uses the previous sets of final solutions to make second order extrapolations for TLIST and EIGEN angles. The distance increments are chosen as described below.

As the program steps out along the propagation path, modal solutions may be lost or removed. First, a mode may be lost in the screening process in subroutine "WVGUID" as described above. At the first point on the path, modes may be overlooked simply because of lack of adequate trial solutions and/or more than one EIGEN input resulting in the same final solution, perhaps due to closely spaced input EIGEN values. If computations are being made at the transmitter or at the midpoint, it is acceptable to lose a solution from the input EIGEN list. At all other points, when a mode is lost execution terminates in "WVGUID". After the tests on the solutions in "WVGUID" are completed at the first point on the path, the program assumes that it has a complete set of modes. After this set is established, solutions may be acceptably removed only in the extrapolation subroutine, "EXTRAP". The solutions produced by "WVGUID" for the current segment are used to compute attenuation rates. Those solutions whose attenuation rate exceeds ATNMAX are deleted from the list. The location of the solution in the set is marked and its removal is indicated by a blank line in the printout of solutions produced by subsequent "WVGUID" calculations.

If a mode is lost during "WVGUID" calculations, the path point is moved back to about halfway between the current point (where a solution was lost) and the previous point (where all solutions were obtained). The actual distance depends on the current value of the distance increment. If the increment is greater than DRMIN, then the
new increment is chosen that it is an integral multiple of DRMIN and is less than or equal to half the previous increment. Geophysical parameters at the new path point are computed, the EIGEN list is revised by "EXTRAP", and "WVGUID" calculations are repeated.

If no modes are lost and the number of iterations required to obtain the solutions is less than or equal to half of MAXITR, then the distance increment is increased by DRMIN. This increase in the distance increment can continue until the path increment is equal to DRMAX. If no modes are lost and the number of iterations required to obtain the solutions is greater than half of MAXITR, the path increment is decreased by DRMIN.

If modes are lost and the separation between the previous point (for which all modes were found) and the current point (for which modes were lost) is less than or equal to DRMIN, then the distance increment is halved. The geophysical parameters for the new path point are linearly interpolated using the parameters of the two points at which geophysical parameters were computed, the EIGEN list is revised by "EXTRAP", and "WVGUID" calculations are repeated. Solutions obtained for interpolated path points are not saved. They are used only to trace the mode solutions between the points for which the geophysical parameters are computed.

If no modes are lost and the number of iterations required to obtain the solutions is less than or equal to half of MAXITR, then the distance increment is doubled. This increase in the distance increment can continue until the path increment is equal to the distance to the end of the interpolation interval. If no modes are lost and the number of iterations required to obtain the solutions is greater than half of MAXITR, the path increment is halved. If the new distance increment is less than 15 km, then the program aborts.

For the 'PRESEG' option, the distance increment is controlled by the intervals between the presegmented distances. When modes are lost, the interpolation procedure for cases in which the backup interval is less than DRMIN described above is followed.

OUTPUT

Mode parameters from the program are written to the logical unit whose numerical value is LUNIT7. The first line of data written contains the transmitter location, path bearing, and the date and time, as input through NAMELIST. The identification which followed 'ID' is written next. If no identification was specified with 'ID', this line of data is blank. The identification is followed by a sequence of lines at each output distance.

The first line of data at each such distance contains the distance, frequency, AZIM, CODIP, MAGFLD, SIGMA, EPSR, and the reference height of the ionospheric profile. In descriptions of other programs, this first line of data at each distance is referred to as the RFACMSET header. This header line is followed by pairs of data lines, one pair for each mode. The quantities in these data lines are the mode solution as a complex angle in degrees, a flag, T1, T2, T3, and T4. The parameters T1, T2, T3, and T4 are described in detail by Ferguson and Snyder (1980). The last line of data at each output distance is blank. These data are suitable for use in "FASTMC" (Ferguson and Snyder, 1980).
If the program fails because of some problem at the first path point, it writes 'Failure at RHO 1' to logical unit 90. Otherwise, it writes the distance of the last point for which "WVGUID" successfully completed. If the end of the path is reached, then this distance is output as 40.

SAMPLE INPUT

Sample input files are shown in figures 1 and 2. In the first sample (figure 1), the path is to be run for all nighttime conditions assuming all seawater ground. The EIGEN list for the transmitter is input directly and the automatic path segmentation is to be used.

id
Sample run
name
&datum freq=23.4 h=50 d=75 lunit7=7 atnmax=50
tlong=150 tlat=20 rbear=10 dmax=10
lub=.005 .0005 dtheta=.01 .001
deigen=.05 .005 tthinc=.05
beta=.43 hprime=87
eigen= 85.678 -0.206 84.595 -0.688 81.806 -0.609 81.027 -0.255
  77.653 -0.791 77.023 -0.269 73.199 -0.825 72.980 -0.300
  68.926 -0.264 68.599 -0.955 64.751 -1.144
  60.457 -0.183
&end
coord

Figure 1. Sample input using COORD option.

The second example (figure 2) is a much more complicated case. It is for the same path of the first sample, but the ionosphere is to be varied according to the diurnal conditions along the path for July 15 at 1612Z. The transition from night to day has been modeled as five steps starting with BETA at 0.30 and HPRIME at 74 and ending with BETA at 0.43 and HPRIME at 87. In addition, the ground conductivity for the last profile takes on three values: 4, 10(-2), and 10(-3). In order to improve the efficiency of the mode tracing, additional segmentation has been performed so that each ground conductivity is processed separately. The segmentation does not produce final output that is monotonically increasing in distance from the transmitter. The necessary ordering of the segments must be performed by editing the final output file or by a user supplied program. The initial mode solutions for each segment have been already calculated and stored in a set of files named XMTR202.MFx where x ranges from 0 to 6.
id
Sample run
name
&datum freq=23.400 h=50. lunit7=7 atnmax=50.
lub=0.005 0.0005 dtheta=0.010 0.0010
deigen=0.050 0.0050 thtinc=0.05
year=84 month=7 day=15 gmt=16.2
tlong=158.150 tlat=21.417 dmax=4
rbear=202.0 &end
eigen xmtr202.mf0
preseg
 0.000,190.6, 50.8,0.350,4.E+00,0.81,0.30,74.0,
 40.0,0,0,0,0,0,0,
eigen xmtr202.mf1
preseg
 0.500,190.8, 56.8,0.337,4.E+00,0.81,0.32,76.2,
 40.0,0,0,0,0,0,0,
eigen xmtr202.mf2
preseg
 0.960,190.9, 63.1,0.329,4.E+00,0.81,0.34,78.3,
 40.0,0,0,0,0,0,0,
eigen xmtr202.mf3
preseg
 1.040,190.9, 64.2,0.327,4.E+00,0.81,0.37,80.5,
 1.240,190.9, 67.3,0.325,4.E+00,0.81,0.37,80.5,
 40.0,0,0,0,0,0,0,
eigen xmtr202.mf4
preseg
 1.340,190.9, 68.8,0.324,4.E+00,0.81,0.39,82.7,
 1.540,190.9, 72.1,0.323,4.E+00,0.81,0.39,82.7,
 40.0,0,0,0,0,0,0,
eigen xmtr202.mf5
preseg
 1.640,190.9, 73.8,0.322,4.E+00,0.81,0.41,84.8,
 1.820,190.9, 76.8,0.322,4.E+00,0.81,0.41,84.8,
 40.0,0,0,0,0,0,0,
eigen xmtr202.mf6
preseg
 1.940,190.8, 78.9,0.322,4.E+00,0.81,0.43,87.0,
 2.120,190.8, 82.2,0.323,4.E+00,0.81,0.43,87.0,
 2.300,190.7, 85.5,0.324,4.E+00,0.81,0.43,87.0,
 2.480,190.6, 88.8,0.326,4.E+00,0.81,0.43,87.0,
 2.660,190.6, 92.1,0.329,4.E+00,0.81,0.43,87.0,
 2.840,190.5, 95.5,0.333,4.E+00,0.81,0.43,87.0,
 3.020,190.4, 98.9,0.337,4.E+00,0.81,0.43,87.0,
 40.0,0,0,0,0,0,0,

Figure 2. Sample input using PRESEG option.
REFERENCES


APPENDIX: LISTING OF THE PROGRAM
SW: SEGMENTED WAVEGUID

#include 'common1.for/list'

common/input/freq,rho,azim,codip,magfld,sigma,epsr,beta,hprime,
hprout
common/path/pathid,tlong,tlat,rlong,rlat,rbear,dmax,drmin,drmax
year,month,day,gmt,nprint,nprof,npath,igcd,ignd,mdir,lost,
lunit7,lx
common/ionosp/htlist(50),lnlist(50,3),hclist(50),cflist(50,3),
charge(3),mratio(3),nrspec,lhtmx,lhtmn,lht,mhmx,mhtmn,mht

c

character*80 pathid
integer year,day
real *4 freq, rho, azim, codip, magfld, sigma, epsr, beta, hprime, hprout,
tlong, tlat, rlong, rlat, rbear, dmax, drmin, drmax, gmt,
htlist, lnlist, hclist, cflist, charge, mratio

#include 'common1.ini/list'

initialize common1

data freq/0./,rho/0./,azim/0./,codip/0./,magfld/0./,
sigma/4.64/,epsr/81./,beta/0./,hprime/0./,
tlong/0./,tlat/0./,rlong/0./,rlat/0./,rbear/720./,
dmax/20./,drmin/.125/,drmax/.5/,mdir/0/,
year/0/,month/0/,day/0/,gmt/0./,nprint/1/,nprof/1/,
igcd/0/,ignd/0/,mdir/0/,lunit7/7/,
charge/-1.,1.,-1../,mratio/1.,2*58000./,nrspec/1/

#include 'common2.for/list'

common/wg in/elist(2,30),tlist(2,30),dtheta(2),lub(2),deigen(2),
$thtinc,ftol,maxitr,alpha,h,d,prec,wr0,atnmax,debug,typitr,
rpoly,nrtlst

common/wg out/tp(30),tterm(4,30),nterm(30),mode(30),modes,nmds

c

c

c

c

c

#include 'common2.ini/list'

initialize common2

data elist/60*0./,tlist/60*0./,
dtheta/.01,.001/,lub/.05,.005/,deigen/.05,.005/,thtinc/.5/,
ftol/1000./,maxitr/7/,alpha/3.14e-4/,h/50./,d/0./,prec/2./,
wr0/2.5e5/,atnmax/50./,debug,typitr/0,0/,rpoly/1/,nrtlst/5/

c
c
c
c

c

name list/datum/freq,rho,azim,codip,magfld,sigma,epsr,beta,hprime,
tlong,tlat,rlong,rlat,rbear,dmax,drmin,drmax,
year,month,day,gmt,nprint,nprof,midpnt,igcd,ignd,mdir,
lunit7,charge,mratio,coefnu,expnu,
eigen,tlist,dtheta,lub,deigen,thtinc,ftol,maxitr,
$ alpha,h,d,prec,wr0,atnmax,typitr,rpol,y,nrtlist$

complex theta
character*8 branch
character*40 fname
character*80 bcd

Initialize MAIN

Unit: Usage:
2 input of alternate eigen list
3 input of profiles along the path
lunit7 output of mode parameters along the path

read(5,1000,end=999) bcd
print 1001,bcd
branch=bcd(1:8)
if(branch .eq. 'id' .or. branch .eq. 'ID') go to 20
if(branch .eq. 'name' .or. branch .eq. 'NAME') go to 100
if(branch .eq. 'eigen' .or. branch .eq. 'EIGEN') go to 130
if(branch .eq. 'profile' .or. branch .eq. 'PROFILE') go to 200
if(branch .eq. 'colfreq' .or. branch .eq. 'COLFREQ') go to 250
if(branch .eq. 'preseg' .or. branch .eq. 'PRESEG') go to 400
if(branch .eq. 'coord' .or. branch .eq. 'COORD') go to 500
if(branch .eq. 'quit' .or. branch .eq. 'QUIT') go to 999
print *, 'ABORT MAIN: Control card not recognized'
stop

Path identification
read(5,1000) pathid
print 1001,pathid
goto 10

NAMELIST input
read(5,datum)
if(nprint .gt. 1) print datum
if(freq .eq. 0.) then
print *, 'ABORT MAIN: FREQ not input'
stop
end if
if(magfld .gt. 1.e-02) magfld=magfld*1.e-04
goto 10

Separate EIGEN list input
read(bcd,1004) fname
open(unit=2,file=fname,status='old')
read(2,datum)
close(unit=2)
if(nprint .gt. 1) then
d0 131 m=1,60
if(eigen(1,m) .eq. 0.) go to 132

A-3
SW$MAIN

0115   131                     km=m
0116   132   print 1040,d,h,(eigen(1,k),eigen(2,k),k=1,km)
0117                           end if
0118   138   go to 10
0119   139                     c
0120   139                     c                     Profile input
0121   200   read(bcd,1002) number
0122                     nrspec=max0(1,number)
0123   138                     nprof=0
0124   138   call profin(5,1,50,nprint,nrspec,lhtmx,htlist,lnlist)
0125     if(lhtmx .le. 0) then
0126     print *, 'ABORT MAIN: Ionospheric profile missing'
0127     stop
0128                           end if
0129   138   go to 10
0130   139                     c
0131   139                     c                     Collision frequency profile input
0132   250                     nuflag=1
0133   call profin(5,2,50,nprint,nrspec,mhtmx,hclist,cflist)
0134     if(mhtmx .le. 0) then
0135     print *, 'ABORT MAIN: Collision frequency profile missing'
0136     stop
0137                           end if
0138   138   go to 10
0139   139                     c
0140   139                     c                     Presegmented path
0141   400                     npath=2
0142                       go to 600
0143   139                     c
0144   139                     c                     Automatic path segmentation
0145   500                     npath=midpnt
0146   139                     c
0147   139                     c                     Test all inputs before execution.
0148   139                     c
0149   139                     c                     Count the modes
0150   600                     do 602 m=1,60
0151     if(eigen(1,m) .eq. 0.) go to 603
0152   602                     nmds=m
0153   603                     if(nmds .le. 0) then
0154     print *, 'ABORT MAIN: No EIGEN list '
0155     stop
0156                           end if
0157   139                     c
0158   139                     c                     Delete duplicate modes using DEIGEN
0159   139                     c
0160     if(nmds .gt. 1) then
0161     m=1
0162     l=2
0163     do 607 k=1,nmds
0164       if(abs(eigen(1,m)-eigen(1,k)) .lt. deigen(1) .and.
0165       abs(eigen(2,m)-eigen(2,k)) .lt. deigen(2)) then
0166       c     Found a match so drop this mode.
0167       do 607 k=1,nmds
0168         eigen(1,k)=eigen(1,k+1)
0169         eigen(2,k)=eigen(2,k+1)
0170         eigen(1,nmds)=0.
0171         eigen(2,nmds)=0.
0172         nmds=nmds-1
0173       if(l .le. nmds) go to 605
0174     end if
0175     go to 607
0176   605                     end if

if(l .lt. nmds) then
  l=l+1
  go to 605
end if
if(m .lt. nmds) then
  m=m+1
  l=m+1
  go to 605
end if
end if
if(nmds .gt. 30) then
  Too many modes input, reduce the number by deleting input
  eigen list values which have attenuation rates in excess of atnmax and re-count the modes
  capk=1./(1.-.5*alpha*h)
  aconst=-182.0426*freq
  atnmx=atnmax
  do 614 m=1,nmds
    if(eigen(1,m) .eq. 0.) go to 615
    theta=cmplx(eigen(1,m),eigen(2,m))*(.01745329252,0.)
    if(aconst*aimag(capk*csin(theta)) .le. atnmx) then
      if(nm .eq. 30) then
        antmx=atnmx-5.
        go to 611
      else
        nm=nm+1
        elist(1,nm)=eigen(1,m)
        elist(2,nm)=eigen(2,m)
      end if
    end if
  end if
  continue
  nmds=nn
  if(nprint .gt. 1) then
    print 1042,atnmax,(elist(1,k),elist(2,k),k=1,nmds)
  end if
else
  Keep all input modes.
  do 616 m=1,nmds
    elist(1,m)=eigen(1,m)
    elist(2,m)=eigen(2,m)
  end if
  if(nmds .lt. 30) then
    elist(1,nmds+1)=0.
    elist(2,nmds+1)=0.
  end if
  if(rpoly .eq. 1 .and. tlist(1,1) .eq. 0.) then
    do 619 m=1,nmds
      tlist(1,m)=elist(1,m)
      tlist(2,m)=elist(2,m)
    end if
  else
    if(nuflag .eq. 0) then
      inhtmx=2
      hclist(1)=200.
      hclist(2)=0.
0229    do 641 n=1,nrspec
0230    en=alog(coefnu(n))
0231    cflist(1,n)=en+expnu(n)*hclist(1)
0232    end if
0233
0234    if(nprof .eq. 1) then
0235       if(betahprime .eq. 0. .and. npath .ne. 2) then
0236          This is not a presegmented path, the profile specification
0237          must be made in the NAMELIST.
0238          print *, 'ABORT MAIN: BETA or HPRIME not input'
0239       stop
0240    end if
0241    nrspec=1
0242    lhtmx=2
0243    hclist(1)=200.
0244    hclist(2)=0.
0245    hprout=hprime
0246    else
0247       if(nprof .eq. 2) then
0248          call gethpr(wr0,hprout)
0249       end if
0250    end if
0251    if(nprof .eq. 1) then
0252       call gcpath
0253    go to 10
0254    c
0255   -BEGIN:
0256    end
0257    999    stop
0260    1000    format(a)
0261    1001    format(1x,(a))
0262    1002    format(8x,1)
0263    1004    format(8x,a)
0264    1040    format('Input EIGEN list: D=',f5.2,' H=',f5.2/
0265       $     ' EIGEN =',6(f8.3,' '),/(8x,6(f8.3,' ')))
0266    1042    format('Reduced EIGEN list: ATNMAX=',f5.1/
0267       $     ' EIGEN =',6(f8.3,' '),/(8x,6(f8.3,' ')))
0268    end
function cdang(arg)
complex*16 arg
real*8 cdang, argr, argi
argr = dreal(arg)
argi = dimag(arg)
cdang = datan2(argi, argr)
if (argi .ge. 0.d0) return

return

end
subroutine comp f
implicit real*8 (a-h,o-z)
c include 'common2.for'
include 'common3.for'
c c=cdcos(theta*zdtr)
csq=c*c
s=cdsin(theta*zdtr)
ssq=s*s
call rbars
if(rpoly .eq. 0) then
call integ
else
call uspoly
end if
if(typitr-1) 5,10,15
 5 f=(rbar11*r11-zone)*(rbar22*r22-zone)
 10 $ -rbar11*rbar22*r12*r21
return
 15 f=rbar22*r22-zone
return
end
subroutine drvequ
implicit real *8 (a-h,o-z)
include 'commonifor'
include 'cogvuwn2.for'
include 'common3.for'
complex*16 k2i,il,im,in,capd,usqd,yud,ysqd,u,usq,t31,t42,t44,t32vrc,t14vrc,t34vrc,ct4l,
 Still,t31a,t41b,t32b,t44b,t32vrc,t14vrc,t34vrc,ct4l,
 Ss11a,d11a,s11b,d11b,c12,c21,
 Ssl2,dl2,s2l,d2l,s22,d22,bll,b22,b12,b21
real*8 lsq,msq,nsq,lm,ln,mn
real*4 ht0
dimension cx(3),capy(3),ysq(3)
data dtr/1.745326525d-2/,coeffx/1.758976111/,coeffy/3.182357d09/
entry intcmp
k2i=dcmplx(0.d0,-0.5d0*wn)
sindip=dsin(codip*dtr)
drcosl=sindip*dcos(azim*dtr)
drcosm=sindip*dsin(azim*dtr)
drcosn=dcos(codip*dtr)
il=dcmplx(0.d0,drcosl)
im=dcmplx(0.d0,drcosm)
in=dcmplx(0.d0,drcosn)
lsq=drcosl**2
msq=drcosm**2
nsq=drcosn**2
lm=drcosl*drcosm
ln=drcosl*drcosn
mn=drcosm*drcosn
co=coeffx/omega**2
cy=coeffy*magFld/omega
do 1 k=1,nrspec
cx(k)=co*charge(k)**2/mratio(k)
capy(k)=cy*charge(k)**2/mratio(k)
ysq(k)=capy(k)**2
call gethpr(100.*wr0,ht0)
toph=ht0
lhtmn=lht
mhtmn=mht
if(debug .le. 1) return
print 110
l=lhtmn
m=mhtmn
ht=topht
slopel=(ht-hlist(l+1))/(hlist(l)-hlist(l+1))
slopm=(ht-hlist(m+1))/(hlist(m)-hlist(m+1))
ed=dexp(lnlist(l+1,1)+(lnlist(l,1)-lnlist(l+1,1))*slopel)
en=dexp(cflist(m+1,1)+(cflist(m,1)-cflist(m+1,1))*slopm)
capx=ed*cx(1)
capz=en/omega
wr=omega*capx/capz
print 111,ht,ed,en,capx,capz,wr
if(ht .lt. toph) return
ht=d
do 11 j=1,lhtmx
if (d .ge. hlist(j)) go to 12
l=j
do 13 j=m,mhtmx
if (d .ge. hclist(j)) go to 10
m=j
c
entry smtrix
usqd=zero
yud=zero
ysqd=zero
slopel=(ht-hlist(lht+1))/(hlist(lht)-hlist(lht+1))
slophem=(ht-hclist(mht+1))/(hclist(mht)-hclist(mht+1))
do 20 k=1,nrspec
capx=dexp(Inlist(lht+1,k)-Inlist(lht+1,k))*slopel)*cx(k)
capz=dexp(cflist(mht+1,k)-cflist(mht+1,k))*slophem)/omega
u=dcmplx(1.d0,-capz)
usq=u*u
capd=-capx/(u*(usq-ysq(k)))
if (cdabs(capd) .gt. 1.d-30) then
usqd=usqd+usq*capd
yud=yud+capy(k)*u*capd
ysqd=ysqd+ysq(k)*capd
end if
continue
crvtrm=alpha*(h-ht)
m11=usqd-lsq*ysqd-crvtrm
m22=usqd-msq*ysqd-crvtrm
m33=usqd-nsq*ysqd-crvtrm
m12=-in*yud-im*ysqd
m21=in*yud-im*ysqd
m13=-in*yud-ln*ysqd
m31=-in*yud-ln*ysqd
m23=im*yud-mn*ysqd
m32=im*yud-mn*ysqd
m11=1m11-m13*m31*capd
m32=im*yud-mn*ysqd
capd=zone/(zone*m33)
t11=-s*m31*capd
t12vrc=s*m32*capd/c
t14vrc=(csq*m33)*capd/c
t31=m23*m31*capd-m21
t32vrc=c+(m22-m23*m32*capd)/c
t34vrc=s*m23*capd/c
t41=(zone+m11-m13*m31*capd)*c
t42=m32*m13*capd-m12
t14vrc=1t14vrc+c
t44=-s*m13*capd
s11a=t11+t44
d11a=t11-t44	s11b=t14vrc+ct41
d11b=t14vrc-ct41	s12=t12vrc+t42
d12=t12vrc-t42	s21=t34vrc+t31
d13=t34vrc-t31	s22=c+t32vrc
d22=c-t32vrc
if(ht .eq. topht) call intair

c entry rderiv

k=0
do 30 j=1,7,2
   k=k+1
   if(dabs(logr(j)) .gt. 15.d0)
      $ logrs(k)=dcmplx(dsign(15.d0,logr(j)),0.d0)
      rs(k)=cdexp(logrs(k))
      b11=r11*(d11a-d11b)
      b22=r22+d22
      b12=r12+d21
      b21=r21-s12
      c12=r12*s21
      c21=r21+d12
      dl11dh=k2i* 
      $ (b11+b12+b21-s11b-s11b*(r12*r21+d22+c12+c21-d11a-d11b)/r11)
      dl22dh=k2i* 
      $ (b12+b21+b22-s22-s22+(r12*r21*(d11a-d11b)+b12+b21+d22)/r22)
      dl12dh=k2i* 
      $ (b11+b12+b22+s11a-s11b-s22+(r11*s12+d12)*(r22+zone)/r12)
      dl21dh=k2i* 
      $ (b11+b21+b22+s11a-s11b-s22+(r11+d21+s21)*(r22+zone)/r21)
   if(debug .gt. 2) then
      print 100,ht~delh,logr,dlrdh
   end if
return

   100  format(f9.4,1pe12.4,4(1x,2e12.3)/21x,4(1x,2e12.3))
   110  format(/' Electron density parameters: ht den nu',
   118    $ 8x,'x z w')
   111  format(27x,f7.1,1p5e10.2)
end
subroutine extrap

This routine sets up and maintains the data sets for the quadratic extrapolation of eigen's down the propagation path.

include 'common1.for'
include 'common2.for'

logical brwstr
complex*8 t(30), y(30), ys(3,30), s, tb, stbcapk, coeff, ngsq, zero/(0.,0.), zone/(1.,0.), zmplxi/(0.,1.),
dimension xs(3)
equivalence (elist,y),(tlist,t)
data dtr/.01745329252/

if(lx.eq.0) then
  This is the first point on the propagation path.
  Set up constants and remove input modes with attenuation rates greater than atnmax.
  capk=cmpix(1.-.5*alpha*h,0.)
  coeff=cmplx(0.,182.0428*freq)/capk

  Get Brewster mode
  if(sigma .lt. 1.e-3) then
    ngsq=cmplx(epsr,-1.7975e7*sigma/freq)
    stb=csqrt(ngsq/(ngsq+zone))*capk
    atten=coeff*stb
    if(atten .le. atnmax) then
      The Brewster mode is contained within the normal set.
      tb=(90.,0.)
      brwstr=.false.
    else
      The Brewster mode is outside the normal set.
      if(atten .le. 2.*atnmax) then
        The attenuation rate is not excessive.
        tb=cmplx(0.,1./dtr)*clog(csqrt(zone-stb**2)+zmlpxi*stb)
        brwstr=.true.
      else
        The attenuation rate is excessive.
        tb=(90.,0.)
        brwstr=.false.
      end if
    end if
  else
    tb=(90.,0.)
    brwstr=.false.
  end if
end if

if(real(y(k)).gt.0.) then
  if(brwstr) then
    If this mode is near the Brewster, then keep it.
    if(abs(real(y(k)-tb)) .le. 1. .and.
      abs(aimag(y(k)-tb)) .le. .5) go to 139
  end if
  atten=coeff*csin(y(k)*dtr)
end if

if(atten .gt. atnmax) then
do 138 l=k,30
        t(l)=t(l+1)
        y(l)=y(l+1)
        t(30)=zero
        y(30)=zero
        go to 137
        end if
    end if
end if
continue
return
end if

x=rho
if(nprint .gt. 1) print 1000,x
nmds=ls
do 143 k=1,nmds
s=zero
do 142 l1=1,lx
p=1.
do 141 l2=1,lx
if(l1 .eq. l2) go to 141
p=p*(x-xs(l2))/(xs(l1)-xs(l2))
do 142
s=s+p*ys(l1,k)
do 143
end if

if(nprint .gt. 1) print 1001,mode(k),s

end if

continue
if(nmds .lt. 30) then
    t(nmds+1)=zero
    y(nmds+1)=zero
    end if
    return
entry xsave

This entry point is called after execution of WVGUID.
It updates the data sets used to do the quadratic extrapolation.

x=rho
if(lx .lt. 3) then
    lx=lx+1
else
    do 21 l=1,2
        xs(l)=xs(l+1)
    do 21 k=1,nmds
        ys(l,k)=ys(l+1,k)
end if
EXTRAP

0143  ls=nm2s
0144  xs(lx)x=
0145  do 25 k=1,nm2s
0146  25  ys(lx,k)=y(k)
0147  c
0148  c  Keep all eigen's at first input distance.
0149  if(lx.eq.1) then
0150     modes=nm2s
0151     return
0152  end if
0153  c  j is counter for modes to be output by SAVEMC
0154  c  k is counter for modes to be used by WVGUID
0155     j=0
0156     k=1
0157  251  if(k.gt.nm2s) return
0158     j=j+1
0159  c  if(bwrst) then
0160       c  Check if this mode is near the Brewster; if so, then keep it.
0161       if(abs(real(y(k)-tb)).le.1.and.
0162          abs(aimag(y(k)-tb)).le.5) go to 256
0163  end if
0164      atten=coeff*cs in (y(k)*dtr)
0165  if(atten.gt.atnmax) then
0166       c  Delete k-th mode
0167       do 253 l=1,4
0168       253       T term(l,j)=zero
0169       nm2s=nm2s-1
0170     ls=nm2s
0171  if(nm2s.eq.0) then
0172     print *, 'ERROR EXTRAP: All modes have been deleted'
0173       lost=1
0174     t(1)=zero
0175     y(1)=zero
0176      return
0177  end if
0178  if(k.gt.nm2s) then
0179  257       modes=modes-1
0180  if(real(T term(1,modes)).ne.0.) return
0181     go to 257
0182  else
0183    do 254 l=k,nm2s
0184    mode(l)=mode(l+1)
0185    t(l)=t(l+1)
0186    y(l)=y(l+1)
0187    do 254 m=1,lx
0188    ys(m,l)=ys(m,l+1)
0189  254    continue
0190    t(nm2s+1)=zero
0191    y(nm2s+1)=zero
0192    go to 251
0193  end if
0194  end if
0195  256       k=k+1
0196  go to 251
0197  c
0198  1000  format('/ Extrapolated EIGEN list for x = ',f8.3)
0199  1001  format(i5,5x,2f8.3,f12.3)
0200  end
subroutine gcdbr(dl,clt1,clt2,rho,br,inb)

Returns great circle distance and geographic bearing angle

Input: DL is longitude of point 2 minus longitude of point 1
CLT1 is co-latitude of point 1
CLT2 is co-latitude of point 2

INB=0: RHO is computed and
BR from point 1 thru point 2 is computed at 1

INB=1: RHO is input and
BR from point 1 thru point 2 is computed at 2

Output: RHO is great circle distance between the input points
BR is geographic bearing angle measured clockwise from
due North

All coordinates, RHO and BR are in radians
Sign convention is + for West and North

data pi/3.14159265e0/,twopi/6.28318531e0/

reduce(arg)=sign(amin1(abs(arg),1.),arg)

cc1t1=cos(clt1)
sclt1=sin(clt1)
cclt2=cos(clt2)
sclt2=sin(clt2)

adl=abs(dl)
if(adl .ge. pi) then
  dl=amod(di,twopi)
else
  adl=abs(dl)
end if
if(inb .eq. 1) then
  if(rho .gt. pi) then
    gcd=twopi-rho
  else
    gcd=rho
  end if
else
  gcd=rho
end if
end if
if(abs(clt1) .le. 1.e-6 .or. abs(clt1-pi) .le. 1.e-6) go to 10
if(adl .le. 1.e-6) go to 20
if(adl-pi) .le. 1.e-6) go to 30
if(adl .ge. pi) then
  if(dl .ge. 0.) then
    dl=dl-twopi
  else
    dl=dl+twopi
  end if
else
  dlt=dl+twopi
end if
if(inb .eq. 0) then
  cgcd=cclt1*cclt2+sclt1*sclt2*cos(dl)
gcd=acos(reduce(cgcd))
if(abs(cgcd-1.) .le. 1.e-6) then
  br=0.
else

A-15
GCDBR

0058 \[ \text{br} = \text{acos}\left(\frac{\text{reduce}(\text{cclt2} - \text{cclt1} \times \text{gcd})}{(\text{sclt1} \times \sin(\text{gcd}))}\right) \]

0059 end if

0060 else

0061 \text{if}(\text{abs}(\text{gcd}) \leq 1. \times 10^{-6}) \text{then}

0062 \text{br} = 0.

0063 \text{else}

0064 \text{br} = \pi - \text{acos}\left(\frac{\text{reduce}(\text{cclt1} - \text{cclt2} \times \text{gcd})}{(\text{sclt2} \times \sin(\text{gcd}))}\right)

0065 \text{end if}

0066 \text{end if}

0067 \text{if}(\text{dl} < 0.) \text{br} = \text{two} \pi - \text{br}

0068 \text{go to 40}

0069 \text{c point 1 is at one of the poles}

0070 \text{c 10 if}(\text{inb} = 0) \text{gcd} = \text{abs}(\text{cltl} - \text{clt2})

0071 \text{if}(\text{abs}(\text{cltl}) \leq 1. \times 10^{-6}) \text{then}

0072 \text{br} = \pi - \text{dl}

0073 \text{else}

0074 \text{br} = \text{dl}

0075 \text{end if}

0076 \text{go to 40}

0077 \text{c coordinates are on same longitude}

0078 \text{c 20 dc} = \text{cltl} - \text{clt2}

0079 \text{if}(\text{dc} \geq 0.) \text{then}

0080 \text{br} = 0.

0081 \text{else}

0082 \text{dc} = -\text{dc}

0083 \text{br} = \pi

0084 \text{end if}

0085 \text{if}(\text{inb} = 0) \text{gcd} = \text{dc}

0086 \text{go to 40}

0087 \text{c coordinates are on opposite longitudes}

0088 \text{c 30 dc} = \text{cltl} + \text{clt2}

0089 \text{if}(\text{dc} \leq \pi) \text{then}

0090 \text{if}(\text{inb} = 0) \text{then}

0091 \text{br} = 0.

0092 \text{else}

0093 \text{br} = \pi

0094 \text{end if}

0095 \text{else}

0096 \text{br} = \pi

0097 \text{end if}

0098 \text{else}

0099 \text{dc} = \text{two} \pi - \text{dc}

0100 \text{if}(\text{inb} = 0) \text{then}

0101 \text{br} = \pi

0102 \text{else}

0103 \text{br} = 0.

0104 \text{end if}

0105 \text{end if}

0106 \text{if}(\text{inb} = 0) \text{gcd} = \text{dc}

0107 \text{c long path calculations}

0108 \text{c 40 if}(\text{inb} = 1) \text{then}

0109 \text{if}(\text{rho} > \pi) \text{then}

0110 \text{if}(\text{br} < \pi) \text{then}

0111 \text{br} = \text{br} + \pi

0112 \text{else}

0113 \text{br} = \text{br} - \pi

0114 \text{end if}

A-16
GCDBR

0115     end if
0116     end if
0117     end if
0118     c
0119     90
0120     if (inb .eq. 0) rho = gcd
0121     return
0122     end
subroutine gcpath

sign convention: + for west and north, - for east and south

include 'common1.for'
include 'common2.for'

dimension prof1(50,3), prof2(50,3)
real lng, long, lat, m1, m2, mgf
character*72 bcd, preseg
logical first

data dtr/1.745329e-2/, re/6.366/, alt/80./

min=0 --- normal
=1 --- interpolating between preseg values
=2 --- last interpolation interval

lost=0 -- no trouble with modes
=1 -- dropped a mode in WVGUID or EXTRAP
=2 -- all modes found but one or more changed significantly

from the extrapolated values

nprof=0 - use profile from MAIN
1 - use exponential profile
2 - read non-exponential profiles along path

WARNING: the heights must match in each profile

first=.true.
write(90,*) 'Failure at RHO 1'

Ix=rO
min=C)
bta=0.
sig=0.
mgf =0.
sigma1=0.

tlng=tlong*dtr
tclt=(90.-tlat)*dtr
rhoO=rho
rhop=rho
drho=drmin

if(rbear .eq. 720.) then
call gcdbr((tlong-rlong)*dtr,tclt,(90.-rlat)*dtr,gcd,xtr,0)
brng=xtr/dtr
if(igcd .eq. 1) then
rhomax=gcd*re
else
rhomax=dmax
end if
else
xtr=rbear*dtr
brng=rbear
rhomax=dmax
end if

if(npath .eq. 2) then
Presegmented path
GCPATH

0086  read(5,2000,end=900) preseg
0087  read(preseg,*) rho,azm,cdp,mgf,sig,eps,bta,hprm
0088  if(rho.eq.40.) then
0089       print *,'End of preseg data'
0090       rewind 90
0091       write(90,2003)
0092       go to 999
0093  else if(rho.gt.rhomax) then
0094       print *,'DMAX reached before end of preseg data'
0095       rewind 90
0096       write(90,2003)
0097       go to 900
0098  end if
0099  if(first) then
0100     rho0=rho
0101     rhop=rho
0102  end if
0103  drho=rho-rhop
0104  if(drho.lt.0.) then
0105       print *,'ABORT GCPATH: Preseg rhos out of order'
0106       go to 900
0107  end if
0108  if(nprof.eq.2) then
0109     read(3,2000) bcd
0110     if(bcd(1:8).ne.'PROFILE') then
0111        print *,'ABORT GCPATH: PROFILE control string missing'
0112        go to 900
0113  end if
0114  read(bcd,2001) nn
0115  if(nrspec.ne.max0(1,nn)) then
0116        print *,'ABORT GCPATH: Number of species is incorrect'
0117        go to 900
0118  end if
0119  call profin(3,1,50,nprint,nrspec,lhtmx,htlist,lnlist)
0120  if(lhtmx.lt.0) then
0121     print *,'ABORT GCPATH: Profile missing'
0122     go to 900
0123  end if
0124  if(lhtmx.gt.0) then
0125     if(lhtmx.ne.lhtmx1) then
0126        print *,'ABORT GCPATH: Number of heights is incorrect'
0127        go to 900
0128  end if
0129     call gethpr(wro,hprout)
0130  end if
0131  else
0132     if(nprof.eq.1) then
0133        if(bta.gt.0.) then
0134           beta=bta
0135           hprime=hprm
0136        end if
0137        if(beta|hprime.eq.0.) then
0138           print *,'ABORT GCPATH: BETA or HPRIME not input'
0139           go to 900
0140        end if
0141     end if
0142  end if

A-19
GCPATH

Calculate exponential profile:
\[ \ln l(1,1) = c f(1,1) + \beta (h(t) - h') - 9.4517306 \]
\[ \ln l(2,1) = c f(2,1) + \beta (h(t) - h') - 9.4517306 \]
hprout = hprime

\[ \text{if}(npath .eq. 1) \text{ then} \]
\[ \quad \text{Calculate midpoint distance:} \]
\[ \quad \rho = 0.5 \times \rho_{\text{max}} \]
\[ \text{else} \]
\[ \quad \text{if}(\rho .eq. 0.) \text{ then} \]
\[ \quad \quad \text{Begin at xtr} \]
\[ \quad \quad \text{lng} = tlng \]
\[ \quad \quad \text{clt} = tclt \]
\[ \quad \quad \text{br} = xtr \]
\[ \quad \quad \text{end if} \]
\[ \quad \text{end if} \]
\[ \quad \text{if}(\rho .gt. 0.) \text{ then} \]
\[ \quad \quad \text{gcd} = \rho / \rho_e \]
\[ \quad \quad \text{call recvr(lng,clt,xtr,gcd,lng,clt)} \]
\[ \quad \text{end if} \]
\[ \text{end if} \]
\[ \text{if}(\text{sig} .gt. 0.) \text{ then} \]
\[ \quad \text{sigma} = \text{sig} \]
\[ \quad \text{epsr} = \text{eps} \]
\[ \text{else} \]
\[ \quad \text{if}(\text{ignd} .eq. 1) \text{ call ground(lng,clt,ncon,sigma,epsr)} \]
\[ \text{end if} \]
\[ \text{if}(\text{mgf} .gt. 0.) \text{ then} \]
\[ \quad \text{azim} = \text{azm} \]
\[ \quad \text{codip} = \text{cdp} \]
\[ \quad \text{magfld} = \text{mgf} \]
\[ \quad \text{if}(\text{magfld} .gt. 1.e-02) \text{ magfld} = \text{magfld} * 1.e-04 \]
\[ \quad \text{else} \]
\[ \quad \quad \text{call newmag(0,alt,lng,clt,bmf,dip,b,br,bp,prt)} \]
\[ \quad \quad \text{azim} = \text{bpath-bmf} / \text{dtr} \]

A-20
if(azim .lt. 0.) then
   azim=azim+360.
else if(azim .gt. 360.) then
   azim=azim-360.
end if

codip=90.-dip/dtr
magfld=b*1.0e-04

if(mdir .eq. 1) then
   azim=azim-180.
   if(azim .lt. 0.) then
      azim=azim+360.
   else if(azim .gt. 360.) then
      azim=azim-360.
   end if
   end if

end if

print 1000,rho, long, lat, bpath, azim, codip, magfld, sigma, epsr

40
lost=0
x=rho
call extrap
if(lost .eq. 1) go to 100
call wvguid
if(nmds .eq. 0 .and. (rho .eq. 0. .or. npath .eq. 1)) then
   print *, 'ABORT GCPATH: Failure at starting rho'
   go to 900
end if
if(lost .eq. 1) go to 100
call xsave
if(lost .eq. 1) go to 100
rhop=rho
if(min .eq. 1) go to 50
if(first) then
   Primary output file:
   open(unit=lunit7,status='new')
   if(year .eq. 0 .and. month .eq. 0 .and. day .eq. 0) then
      write(lunit7,1030) tiong, tlat, brng, beta, hprime, pathid
   else
      write(lunit7,1031) tlong, tlat, brng, beta, hprime,
      mod(year,100), month, day, gmt, pathid
   end if
first=.false.
end if
call savemc
if(npath .eq. 1) then
   rewind 90
   write(90,2003)
   go to 999
else
   rewind 90
   write(90,2002) rho
end if
if(rho+.002 .ge. rhomax) go to 900
GCPATH
0257     rho1=rho
0258     a1=azim
0259     c1=azid
0260     m1=magfd
0261     el=epsr
0262     sl=aalog(sigma)
0263     sigmal=sigma
0264     if(nprof.gt.0) then
0265         lhtmx=lhtmx
0266         do 48 l=1,lhtmx
0267         do 48 m=1,nrspec
0268     48 prof1(l,m)=lnlist(l,m)
0269     end if
0270     if(min.eq.2) then
0271         min=0
0272         go to 70
0273     end if
0274     c
0275     if(lost.eq.2) then
0276         if(min.eq.0) then
0277             drho=amax1(drho-drmin,drmin)
0278         else
0279             drho=.5*(rho2-rho)
0280         end if
0281     else
0282         if(min.eq.0) then
0283             drho=amin1(drho+drmin,drmax)
0284         else
0285             drho=rho2-rho
0286         end if
0287     end if
0288     c
0289     70 if(min.eq.0.and.npath.eq.2) go to 20
0290     rho=rho+drho
0291     if(rho+.002.gt.rhomax) then
0292         drho=drho-rhomax
0293     end if
0294     if(min.eq.1) go to 120
0295     go to 30
0297     c
0298     Back up on propagation path
0299     100 if(rho.eq.rho0) then
0300         print *, 'ABORT GCPATH: Failure at starting rho'
0301         go to 900
0302     end if
0303     if(min-1) 101,102,103
0304     101 if(npath.eq.2) go to 105
0305     if(drho.le.drmin) go to 110
0306     nrd=.5*drho/drmin
0307     if(nrd.eq.0) go to 110
0308     drho=nrd*drmin
0309     rho=rhol+drho
0310     go to 30
0311     103 min=1
0312     drho=drhop
0313     102 drho=.5*drho

A-22
GCPATH

0314 if(drho .lt. .015125) then
0315 print *, 'ABORT GCPATH: Backup interval is less than 0.015125'
0316 go to 900
0317 end if
0318 104 rho=rhop+drho
0319 go to 120
0320 c
0321 105 if((rho-rhol)/drmin .gt. 10.) then
0322 print *, 'ABORT GCPATH: Preseg interval too large for efficient processing'
0323 go to 900
0324 end if
0325 c
0326 Begin interpolation
0327 110 min=1
0328 drho=.5*(rho-rhol)
0329 if(drho .lt. .015125) then
0330 print *, 'ABORT GCPATH: Backup interval is less than 0.015125'
0331 go to 900
0332 end if
0333 rho2=rho
0334 a2=azim
0335 if(a2-a1 .gt. 180.) then
0336 a2=a2-360.
0337 else if(a2-a1 .lt. -180.) then
0338 a2=a2+360.
0339 end if
0340 c2=codip
0341 m2=magfld
0342 e2=epsr
0343 s2=a log(sigma)
0344 sigma2=sigma
0345 if(nprof .gt. 0) then
0346 do 111 l=1, lhtmx
0347 do 111 m=1,nrspec
0348 prof2(l,m)=lnlist(l,m)
0349 end if
0350 rho=rhol+drho
0351 c
0352 120 if(rho+.002 ge. rho2) then
0353 End of interpolation
0354 121 min=2
0355 drhop=drho
0356 drho=drmin
0357 rho=rho2
0358 azim=a2
0359 if(azim .lt. 0.) then
0360 azim=azim+360.
0361 else if(azim .gt. 360.) then
0362 azim=azim-360.
0363 end if
0364 codip=c2
0365 magfld=m2
0366 epsr=e2
0367 sigma=sigma2
0368 if(nprof .gt. 0) then
0369 go to 104
GCPATH

0371    do 121 l=1,lhtmx
0372    do 121 m=1,nrspec
0373 121   lnlist(l,m)=prof2(l,m)
0374    end if
0375    else
0376    c Interpolate
0377    slope=(rho-rhol)/(rho2-rhol)
0378    azim=azim+slope*(a2-a1)
0379    if(azim .lt. 0.) then
0380      azim=azim+360.
0381    else if(azim .gt. 360.) then
0382      azim=azim-360.
0383    end if
0384    codip=c1+slope*(c2-c1)
0385    magfld=m1+slope*(m2-m1)
0386    epsr=e1+slope*(e2-e1)
0387    sigma=exp(sl+slope*(s2-s1))
0388    if(nprof .gt. 0) then
0389      do 122 l=1,lhtmx
0390      do 122 m=1,nrspec
0391      lnlist(l,m)=prof1(l,m)+slope*(prof2(l,m)-prof1(l,m))
0392      end if
0393    end if
0394    print 1002, rho,azim,codip,magfld,sigma,epsr
0395    go to 40
0396    c
0397 900  if(npath .eq. 2) then
0398 903    read(5,2000,end=999) bcd
0399    if(bcd(1:5) eq. '40,0,'.) go to 999
0400 901  go to 903
0401 999   write(lunit7,1032)
0402  close(unit=lunit7)
0403  print *, 'Execution terminating for this path'
0404  return
0405 c
0407 1000  format(/' Propagation path parameters: rho long lat',
0408 $ 4x,'bear azim codip magfld sigma epsr'/
0409  $ 26x,f10.3,f10.2,4f9.2,el1.2,1pe11.2,2pef8.2)
0410 1002  format(/' Interpolated path parameters: rho azim',
0411  $ 5x,,'codip magfld sigma epsr'/
0412  $ 38x,f10.3,f10.2,f9.2,el1.2,1pe11.2,2pef8.2)
0413 1030  format('sw xmtr',f7.1,2f6.1,' prof',f5.2,5f5.2/a80)
0414 1031  format('sw xmtr',f7.1,2f6.1,' prof',f5.2,5f5.1,
0415  $ '(',3(2.2,'/'),4.1,')')/a80)
0416 1032  format('r 40.')
0417 2000  format(a72)
0418 2001  format(8x,i1)
0419 2002  format(f6.3)
0420 2003  format('40')
0421   end
subroutine gethpr(wr,hpr)
  c
  Routine to determine the height where omega sub r is a
  c
  specific value. The value returned is to nearest km.
  c
  include 'common1.for'
  c
data coeffx/3.182357e9/
  c
  Start at the bottom of the profile and work up.
  c
  lht=lhtmx-1
  mht=mhtmx-1
  ht=amin1(htlist(lhtmx),hclist(mhtmx))

  10 if(lht .gt. 1 .and. ht .ge. htlist(lht-1)) then
      lht=lht-1
      go to 10
  end if
  12 if(mht .gt. 1 .and. ht .ge. hclist(mht-1)) then
      mht=mht-1
      go to 12
  end if
  slope l=(ht-htlist(lht+1))/(htlist(lht)-htlist(lht+1))
  slope m=(ht-hclist(mht+1))/(hclist(mht)-hclist(mht+1))
  sum=0.
  do 14 n=1,nrspec
      dn=exp(lnlist(lht+1,n)+(lnlist(lht,n)-lnlist(lht+1,n))*slope l)
      cf=exp(cflist(mht+1,n)+(cflist(mht,n)-cflist(mht+1,n))*slope m)
      sum=sum+coeffx*dn/(mratio(n)*cf)
  14 if(sum .gt. wr) then
      hpr=ht
      if(nprint .gt. 1) then
          print *, 'GETHPR: wr, hpr=', wr, hpr
      end if
      return
  end if
  ht=ht+1.
  go to 10
end
subroutine ground(xlong,xlat,ncodesigma,epsr)

Returns conductivity code, conductivity, dielectric constant

Input: XLONG is West longitude in degrees
(i.e., -117.3 for 117 degrees, 18 minutes East)
XLAT is North latitude in degrees
(i.e., 32.8 for 32 degrees, 48 minutes North)

Output: NCODE is conductivity code from GRNDMAP.DAT
(ncode=0 is sea water, =1 is ice; see DATA below)
SIGMA is mho/m
EPSR is the dielectric constant
A list of sigma and epsr is also placed into a common.

Requires: GRNDMAP.DAT

include ' [305021.jaflib]data files.for/list'

character*40 grnd$d '/user$disk$3:[305021.jaflib]grndmap.dat'/
character*40 itsn$d '[305021.jaflib]itsnoise.dat'/
character*40 wrld$d '[305021.jaflib]world.dat'/

character*40 grnd$d/ sss(10), rrr(10)
logical first/.true./
dimension lcode(361), map(4530), sss(10), rrr(10)
data ss/1.e-5,3.e-5,1.e-4,3.e-4,1.e-3,3.e-3,1.e-2,3.e-2,.1,4./
data rr/5.,5.,10.,10.,15.,15.,15.,15.,80.,81./

if(first) then
open(unit=8, file=grnd$d,status='old', readonly)
read(8,1) lcode, map
1
format(gi8) close(unit=8)
do 2 l=1,10
sss(l)=ss(l)
2 rrr(l)=rr(l)
first=.false.
end if

phi=xlong
if(phi .gt. 180.) then
phi=phi-360.
else
if(phi .lt. -180.) phi=phi+360.
end if
if(abs(phi) .gt. 180. .or. abs(xlat) .gt. 90.01) then
print 11,xlong,xlat
format(''/****** Error in GROUND: Xlong    Xlat' /26x,2f9.2)
stop
end if
lat=181.-2.*xlat
if(lat .gt. 360) lat=360
long=361.-2.*phi
if(long .gt. 720) long=1
l1=lcode(lat)
l2=lcode(lat+1)-1
do 21 l=l1,l2
maplm1=map(l)/10000

A-26
GROUND

0058    map = map(l) - 10000 * map1m1
0059    mlong = map / 10
0060    if (mlong .ge. long) go to 31
0061 21  continue
0062 31  ncd = map - mlong * 10
0063    mlong = map1m1 / 10
0064    if (mlong .lt. long) go to 41
0065    ncd = map1m1 - mlong * 10
0066 41  if (ncd .lt. 0 .or. ncd .gt. 9) then
0067      print 51, xlong, xlat, long, lat, i1, i2, mlong, ncode
0068 51  format(/'*********** Error in GROUND: Xlong   Xlat',
0069         /'   long    lat   i1   i2   mlong   ncode',
0070       /26x,2f9.2,4i6,2i8)
0071    stop
0072    end if
0073    ncode = ncd
0074    if (ncode .eq. 0) ncode = 10
0075    sigma = ss(ncode)
0076    epsr = rr(ncode)
0077    return
0078    end
subroutine intaIr
implicit real *8 (a-h,o-z)
c include 'common2.for'
c include 'common3.for'
c complex*16 q,p,t,d11,d13,d31,d33,delta,fnsq,froot,
 dimension phase1(8), phase2(8), p(2), t(2), q(4)
equivalence (logrll,phase1(1))
data pi/3.141592653d0/, twopi/6.283185307d0/
c if(isotrpl-1) 10,100,102
c com1=zone*m11
com3=zone*m33
csqm22=csq*m22
csqm33=csq*m33
b3=0.25d0*s*(m13+m31)/com3
b2=(-csqm33*com1+m13*m31-com3*csqm22+m23*m32)/(6.d0*com3)
b1=s*(m12*m23+m32*m12-csqm22*(m13+m31))/(4.d0*com3)
b0=(com1*csqm22+csqm33*m12*m31+m32*m21-m13-m31*csqm22)
c call qartic(q,b3,b2,bl,b0,debugnewq)
do 30 n=1,2
d11=zone*m11-q(n)**2
d13=m13+s*q(n)
d31=m31+s*q(n)
d33=zone*m33-s**2
delta=d11+d33-d13*d31
p(n)=(-m12*d33+d13*m32)/delta
t(n)=q(n)*p(n)-s*(-d11*m32+m12*d31)/delta
pyntng=t(n)*dconjg(p(n))+q(n)
if(pyntng .lt. 0.) print 201,theta,q(n),pyntng
continue
delta=(t(1)*c+p(1))*(c+q(2))-(t(2)*c+p(2))*(c+q(1))
  r11 =((t(1)*c+p(1))*(c+q(2))-(t(2)*c+p(2))*(c+q(1)))/delta
  r22 =((t(1)*c+p(1))*(c+q(2))-(t(2)*c+p(2))*(c+q(1)))/delta
  r12 =-2.d0*c*(t(1)*p(2)-t(2)*p(1))/delta
  r21 =-2.d0*c*(q(1)-q(2))/delta
logrll=cdlog(r11)
logr12=cdlog(r12)
logr21=cdlog(r21)
logr22=cdlog(r22)
if(adjflg .eq. 1) then
do 70 n=2,8,2
if(phase1(n)-phase2(n) .le. pi) go to 60
phase1(n)=phase1(n)-twopi
go to 50
if(phase2(n)-phase1(n) .le. pi) go to 70
phase1(n)=phase1(n)+twopi
go to 60
continue
end if
do 90 n=2,8,2
phase2(n)=phase1(n)
if(debug .gt. 2) print 202
INTALR

0095     return
0096   100     ir=1
0097     fnsq=zone+m11
0098     froot=cdsqrt(csq+m11)
0100     go to 106
0101    101     r11=(fnsq*c-froot)/(fnsq*c+froot)
0102     r22=(c-froot)/(c+froot)
0103     go to 105
0104   c
0105  102     ir=2
0106     fnsq=zone+m11
0107     froot=cdsqrt(csq+m11+m13**2/fnsq)
0108     go to 106
0109  103     coml=(s*froot+m13)/(s*fnsq+m13)
0110     r11=(c-coml)/(c+coml)
0111   c
0112  103     ir=3
0113     froot=cdsqrt(csq+m22)
0114     go to 106
0115  104     r22=(c-froot)/(c+froot)
0116     r12=(1.d-20,0.d0)
0117     r21=(1.d-20,0.d0)
0118     go to 40
0119   c
0119  106     if(dimag(froot) .gt. 0.d0) froot=-froot
0120     if(ir-2) 101,103,104
0121   c
0122  201     format(' for theta=',f7.4,f9.4,' q=',1p2e11.3,
0123     $     ' poynting(z)='e11.3)
0124  202     format(/4x,'ht',7x,'delh')
0125     end

A-29
subroutine integ
implicit real *8 (a-h,o-z)

c include 'common1.for'
c include 'common2.for'
c include 'common3.for'

c real*8 logr0
c integer sf lag
dimension logr0(8), dlrdh0(8), dlogr0(8), dlogr1(8), dlogr2(8)
data dlhmin/1.953125d-3/,dlgrmx/1.d20/
c
factor=10.d0**(-prec)
emax=factor*3.d0
emin=factor*.3d0
ht=topht
lht=lhtmn
mht=mhtmn
delh=3.125d-2
svdelh=delh
if(debug .gt. 2) print 200,theta
call smtrix
c runge kutta
if(debug .gt. 2) print 201
10 sflag=0
11 if(lht .lt. lthmx .and. ht .le. htlist(lht+1)) then
lht=lht+1
go to 11
end if
13 if(mht .lt. mthmx .and. ht .le. hclist(mht+1)) then
mht=mht+1
go to 13
end if
if(ht-delh .lt. htlist(lht+1)) then
sflag=1
saveht=htlist(lht+1)
delh=ht-saveht
end if
if(ht-delh .lt. d) then
sflag=1
saveht=d
delh=ht-saveht
end if
do 30 i=1,8
logr0(i)=logr(i)
30 dlrdh0(i)=dlrdh(i)
c
Try again
do 50 i=1,8
dlrdh0(i)=-dlrdh0(i)*delh
dlogr0(i)=logr0(i)+0.5d0*dlogr0(i)
ht=ht-0.5d0*delh
call smtrix
do 60 i=1,8
dlrdh(i)=dsign(dmin1(dlgrmx,dabs(dlrdh(i))),dlrdh(i))
dlogr1(i)=-dlrdh(i)*delh
INTEC

0110  60  logr(i)=logr0(i)+0.5d0*logr1(i)
0111    call rderiv
0112    do 70  i=1,8
0113      dlrdh(i)=dsign(dmin1(dlgrmx,dabs(dlrdh(i))),dlrdh(i))
0114      dlogr2(i)=-dlrdh(i)*delh
0115    70  logr(i)=logr0(i)+dlogr2(i)
0116    ht=ht-0.5d0*delh
0117    call smtrix
0118    error=0.d0
0119    do 80  i=1,8
0120      dlrdh(i)=dsign(dmin1(dlgrmx,dabs(dlrdh(i))),dlrdh(i))
0121      dlogr4=((-dlrdh(i)*delh+dlogr0(i))/2.d0+dlogr1(i)+dlogr2(i))/3.d0
0122      logr(i)=logr0(i)+dlogr4
0123    80  error=error+(dlogr2(i)-dlogr4)**2
0124    error=dsqrt(error/8.d0)
0125    if(error .lt. emax .or. delh .le. dlhmin) go to 100
0126    sflag=0
0127    ht=ht+delh
0128    delh=0.5d0*delh
0129    if(delh .lt. dlhmin) delh=dlhmin
0130    go to 40
0131  100  call rderiv
0132    if(error .lt. emin) delh=2.*delh
0133    if(sflag .eq. 1) then
0134        delh=svdelh
0135        ht=saveht
0136    end if
0137    svdelh=delh
0138    if(ht .gt. d) go to 10
0139    return
0140    c
0141  200  format(/' DEBUG: theta =',2f9.4)
0142  201  format(' ')
0143    end
subroutine iterat
implicit real *8 (a-h,o-z)
cThis routine drives the iteration to obtain solutions to the modal equation.
cinclude 'common2.for'
include 'common3.for'
ccomplex*16 theta0,f0,dlthta
real*4 absr,absi
cnriter=0
if(debug .gt. 1) then
  if(rpoly .eq. 0) then
    print 300
  else
    print 301
  end if
else
  print 302
end if
nc Store the starting angle
theta0=theta
theta=theta-dlthta
call comp f
f0=f
call comp f
theta=theta+dlthta
c Store the magnitude of the f-function for the starting angle
if(nriter .eq. 0) fmag0=cdabs(f)
nriter=nriter+1
dfdtht=(f-f0)/dlthta
dlhta=-f/dfdtht
if(debug .gt. 1) then
  fmag=cdabs(f)
  print 303,theta,fmag,dlhta,dfdtht
end if
c absr=dabs(dreal(dlhta))
absi=dabs(dimag(dlhta))
if(absr .gt. thtinc) dlhta=dlhta*(thtinc/absr)
if(absi .gt. thtinc) dlhta=dlhta*(thtinc/absi)
theta=theta+dlhta
if(nriter .lt. maxitr .and. absr .gt. lub(1) .or. absi .gt. lub(2))) go to 10
end if
nc nriter=nriter+1
call comp f
dfdtht=(f-f0)/dlhta
if(debug .gt. 1) then
  fmag=cdabs(f)
dlhta=zero
  print 303,theta,fmag,dlhta,dfdtht
end if
if(rpoly .eq. 1) then
  $ (absr .gt. lub(1) .or. absi .gt. lub(2))) go to 10
end if
c Test the magnitude of the f-function of the final angle
ITERAT

0095     fmag = cdabs(f)
0096     if(fmag .gt. fmagO) then
0097         print 304, fmagO, fmag
0098         theta = theta0
0099     end if
0100   end if
0101   if(typitr .gt. 0) then
0102       if(typitr .eq. 1) then
0103           dfdtht = (rbar22*r22 - zone)*dfdtht
0104       else
0105           dfdtht = (rbar11*r11 - zone)*dfdtht
0106       end if
0107   end if
0108 return
0109 300 format('0Iterations: exact')
0110 301 format('0Iterations: inexact')
0111 302 format(8x,'real imag fmag d real d imag',5x,
0112 $                       'dfdtht real ddfdtht imag')
0113 303 format(5x,2f8.4,1pe12.3,2(1x,2e11.3))
0114 304 format(' Warning ITERAT: During RPOLY=1, starting fmag (',
0115 $                      '1pe10.4,) is smaller than final fmag (',1pe10.4,')')
0116 end
subroutine mdhnkl (z,h1,h2,h1prme,h2prme,theta,idbg)

implicit complex*16 (a-h,o-z)
complex*16 i,mpower,mterm
real*8 a,b,c,d,cap,part1,part2,zmag
character*4 idbg

dimension a(30), b(30), c(30), d(30), cap(30), part1(2), part2(2)
equivalence (part1,term4), (part2,sum4)

data c(30), part2, zmag / 
  data a / 
  data b / 
  data c / 
  data d / 

A-34
0058  $  2.696328218460259742d+02,  7.98912064728959...0'*0
0059  $  1.902715826881013924d+01,  3.7188105233492256682...00
0060  $  6.0764877832340288572d-01,  8.420204895828535444d-02
0061  $  1.002621486551016149d-02,  3.1435482251472901638...03
0062  $  9.367869420580235442d-05,  1.0363012784032058021d-03
0063  $  5.3507368429183773360d-07,  1.0209780508963274722d-10
0064  $  9.3867869420580235442d-05,  7.5124345274574017960d-06
0065  $  5.3507368429183773360d-07,  1.0209780508963274722d-10
0066  $  4.8341763773500352579d-12,  2.0913590211334783723d-13
0067  $  8.2990437346566602039d-15,  3.0316141496462685641d-16
0068  $  9.2821903191776400453d-21,  2.5103962999804300309d-22
0069  $  1.0416666666666666663d-01,  8.3550347222222222116d-02
0070  $  1.2822657455632716019d-01,  2.9184902646414046315d-01
0071  $  8.816276744375674874d-01,  3.3214082818627675264d+00
0072  $  1.4995762986862554546d+01,  7.8923013011586517530d+01
0073  $  4.777126865134715582d+02,  3.2074900908906619004d+03
0074  $  2.40865496804000605d+04,  1.9892311910462685641d+05
0075  $  1.79190200776538063d+06,  1.748377180034121023d+07
0076  $  1.83707397633072978d+08,  2.067940329451555108d+09
0077  $  2.482751937593858847d+10,  3.166945491734887315d+11
0078  $  4.2771126865134715582d+12,  6.0971132411392560749d+13
0079  $  9.148662234356396792d+14,  4.143352571000935010d+16
0080  $  2.378844395175757942d+17,  4.104608160094692188d+18
0081  $  7.39004915704853993d+19,  1.385220004603943141d+21
0082  $  2.7030825903275761623d+22,  5.474774861645573335d+23
0083  $  1.149893701438633524d+25,  2.5014180692753603969d+26
0084  data i/0.d0,0.d0/
0085  data one/(1.d0,0.d0)/, two/(2.d0,0.d0)/, zero/(0.d0,0.d0)/
0086  data root3/(1.73205080756888d0,0.d0)/
0087  data alpha/(8.53667205720572d0,0.d0)/
0088  data const1/(2.58819045102522d-01,-9.65925826289067d-01)/
0089  data const2/(2.58819045102522d-01,9.65925826289067d-01)/
0090  data const3/(-9.65925826289067d-01,2.58819045102522d-01)/
0091  data const4/(-9.65925826289067d-01,-2.58819045102522d-01)/
0092  c
0093  zpower=one
0094  sum3=zero
0095  sum4=zero
0096  zmag=cdabs(z)
0097  if(zmag .gt. 6.d0) go to 70
0098  sum1=zero
0099  sum2=zero
0100  zterm=-z**3/(200.d0,0.d0)
0101  do 50 m=1,30
0102  sum1=sum1+dcmplx(a(m),0.d0)*zpower
0103  sum2=sum2+dcmplx(b(m),0.d0)*zpower
0104  sum3=sum3+dcmplx(c(m),0.d0)*zpower
0105  sum4=sum4+term4
0106  if(dabs(part1(1)) .le. 1.d-17*dabs(part2(1))) and.
0107  if(dabs(part1(2)) .le. 1.d-17*dabs(part2(2))) go to 60
0108  zpower=zpower*zterm
0109  50
0110  60
0111  gm2f=i*(z*sum2-two*sum1)/root3
0112  gpmfp=i*(sum4+two*sum1*sum3)/root3
0113  h1=z*sum2+gm2f
0114  h2=h1-two*gm2f
0115  h1prme=sum4*gm2f

A-35
MDHNKL

0115 h2prme=h1prme-two*gpmfp
0116 go to 999
0117 70 mpower=one
0118 sum1=one
0119 sum2=one
0120 rtz=cdsqrt(z)
0121 sqrtzb=rtz*z
0122 zterm=i/sqrtzb
0123 mterm=-zterm
0124 dm=zero
0125 term3=one
0126 do 80 m=1,30
0127 zpower=zpower*zterm
0128 mpower=mpower*mterm
0129 dm=dm+one
0130 term1=dcmplx(cap(m),0.d0)*zpower
0131 term2=dcmplx(cap(m),0.d0)*mpower
0132 if(cfabs(term2/term3) .ge. 1.d0) go to 81
0133 sum1=sum1+term1
0134 sum2=sum2+term2
0135 sum3=sum3+dm*term1
0136 term4=dm*term2
0137 if(dabs(part1(1)/part2(1)) .le. 1.d-17 .and.
$ dabs(part1(2)/part2(2)) .le. 1.d-17) go to 81
0139 80 term3=term2
0140 81 zterm=(-1.5d0,0.d0)/z
0141 sum3=sum3+zterm
0142 sum4=sum4+zterm
0143 term1=((-0.25d0,0.d0)-i*sqrtzb)/z
0144 term2=((-0.25d0,0.d0)+i*sqrtzb)/z
0145 exp1=cdexp((0.d0,0.666666666666666d0)*sqrtzb)
0146 exp2=const1*exp1
0147 exp3=const2/exp1
0148 exp4=const3/exp1
0149 exp5=const4/exp1
0150 zterm=alpha/cdsqrt(rtz)
0151 term4=z
0152 if(part1(1) .ge. 0.d0 .or. part1(2) .ge. 0.d0) go to 90
0153 h1=zterm*(exp2+sum2*exp5*sum1)
0154 h1prme=zterm*(exp2*(sum2*term2*sum4)+exp5*(sum1*term1*sum3))
0155 go to 110
0156 90 h1=term2*exp2+sum2
0157 h1prme=term2*exp2*(sum2*term2*sum4)
0158 110 if(part1(1) .ge. 0.d0 .or. part2(2) .lt. 0.d0) go to 120
0159 h2=term2*(exp3+sum1*exp4*sum2)
0160 h2prme=term2*(exp3*(sum1*term1*sum3)+exp4*(sum2*term2*sum4))
0161 go to 999
0162 120 h2=term2*exp3*sum1
0163 h2prme=term2*exp3*(sum1*term1*sum3)
0164 c calculate wronskian as partial check on validity
0165 999 sum4=h1*h2prme-h1prme*h2
0166 if(cfabs(term2(1)) .le. 1.d-8 .and.
$ dabs(term2(2)+1.457495441040461d0) .le. 1.d-8) go to 1000
0168 1000 print 1001,sum4,theta,idbg
0169 1001 format(' ****** possible error in mdhnkl: w = ',1p2e15.6,
MDHNLK

0172 $ ' for theta = ',0p2f10.4,' at ',a4)
0173 end
subroutine newmag(j,rphij,thet,bmf,dip,b,br,bp,bt)

Returns parameters of the geomagnetic field

Input:  J=0: Use spherical earth
        J=1: Use spheroidal earth
        R is altitude in km
        PHIJ is West longitude in radians
        THET is co-latitude in radians

Output: BMF is declination of the geomagnetic field
        DIP is dip angle
        B is total field
        BR is radial component
        BP is longitudinal component
        BT is latitudinal component

dimension g(10,10), bm(10)

data g/.0,3.032193e04,2.522093e03,-3.285459e03,-4.170639e03,1.692810e03,-
       6.684202e02,-1.900312e03,-2.405232e03,-9.358458e02,-5.755070e03/
       data bm/9.933492e04,2.522093e03,-3.285459e03,-4.170639e03,1.692810e03,-
       6.684202e02,-1.900312e03,-2.405232e03,-9.358458e02,-5.755070e03/
       data nmax/10/,berr/0.0001/

p22=abs(sin(thet))

if(p22 .eq. 0.) p22=1.e-6
p21=sqrt(1.-p22*p22)
re=6356.912+p22*p22*(21.3677+.108*p22*p22)
ar=(re+r)/6371.2

if(thet .le. 1.57079627e0) go to 70
p21=-p21

if(j .eq. 0) go to 90
ssq=p22*p22
ar=ar+(14.288-ssq*(21.3677+.108*ssq))/6371.2

n= 2
dp22=p21

phi=phij

A-38
if(\(\phi\)) 92, 96, 94
phi=\(-\phi\)
go to 96
\(\phi=6.2831853e0-\phi\)
dp21=-p22
aor=ar*ar*ar
\(c2=g(2,2)*cp2+g(1,2)*sp2\)
\(br=-(aor+aor)*(g(2,1)*p21+c2*p22)\)
\(bt=aor*\left(g(2,1)*dp21+c2*dp22\right)\)
\(bp=aor*\left(g(1,2)*cp2-g(2,2)*sp2\right)*p22\)
if(nmax .lt. 3) go to 260
aor=aor*ar
err=err*sqrt((bp/p22)**2+br**2+bt**2)
if(bm(3)*aor .le. err) go to 260
sp3=(sp2+sp2)*cp2
cp3=(cp2+sp2)*\((cp2-sp2)\)
p31=p21*p21-0.333333333e0
p32=p21*p22
p33=p22*p22
dp31=-p32-p32
dp32=2*p21*p21-p33
dp33=-dp31
c2=g(3,2)*cp2+g(1,3)*sp2
c3=g(3,3)*cp3+g(2,3)*sp3
br=br-3.0*aor*\((g(3,1)*p31+c2*p32+c3*p33)\)
bt=bt+aor*\((g(3,1)*dp31+c2*dp32+c3*dp33)\)
bp=bp-aor*\((g(3,2)*sp2-g(2,3)*sp3)\)
s*p33)
cn= 4
if(nmax .lt. 4) go to 260
aor=aor*ar
if(bm(4)*aor .le. err) go to 260
sp4=sp2*cp3+cp2*sp3
cp4=cp2*cp3-sp2*sp3
p41=p21*p31-0.266666666e0*p21
dp41=p21*dp31+dp21*p31-0.266666666e0*dp21
p42=p21*p32-0.200000000e0*p22
dp42=p21*dp32+dp21*p32-0.200000000e0*dp22
p43=p21*p33
dp43=p21*dp33+dp21*p33
dp44=p22*p33
dp44=3.0*p43
c2=g(4,2)*cp2+g(1,4)*sp2
c3=g(4,3)*cp3+g(2,4)*sp3
c4=g(4,4)*cp4+g(3,4)*sp4
br=br-4.0*aor*\((g(4,1)*p41+c2*p42+c3*p43+c4*p44)\)
bt=bt+aor*\((g(4,1)*dp41+c2*dp42+c3*dp43+c4*dp44)\)
bp=bp-aor*\((g(4,2)*sp2-g(1,4)*cp2)\)
$p43*3.0*\((g(4,4)*sp4-g(1,3)*sp3)\)
if(nmax .lt. 5) go to 260
aor=aor*ar
if(bm(5)*aor .le. err) go to 260
sp5=(sp3+sp3)*sp3
sp5=(cp3+sp3)*sp3

A-39
p51 = p21*p41 - 0.25714285e0*p31
p52 = p21*p42 - 0.22857142e0*p32
p53 = p21*p43 - 0.14285714e0*p33
dp51 = p21*dp41 + dp21*p41 - 0.25714285e0*dp31
dp52 = p21*dp42 + dp21*p42 - 0.22857142e0*dp32
dp53 = p21*dp43 + dp21*p43 - 0.14285714e0*dp33
p54 = p21*p44
dp54 = p21*dp44 + dp21*p44
p55 = p22*p44
dp55 = 4.0*p54
c2 = g(5,2)*cp2 + g(1,5)*sp2
c3 = g(5,3)*cp3 + g(2,5)*sp3
c4 = g(5,4)*cp4 + g(3,5)*sp4
c5 = g(5,5)*cp5 + g(4,5)*sp5
br = br - 5.0*aor*(g(5,1)*p51 + c2*p52 + c3*p53 + c4*p54 + c5*p55)
b = bt + aor*(g(5,1)*dp51 + c2*dp52 + c3*dp53 + c4*dp54 + c5*dp55)
b = bp - aor*((g(5,2)*sp2 - g(1,5)*cp2)*p52 + 2.0*(g(5,3)*sp3 - g(2,5)*cp3)*p53 + 0.14285714e0*p33)
$p = p53 + 3.0*(g(5,4)*sp4 - g(3,5)*cp4)*p54 + 4.0*(g(5,5)*sp5 - g(4,5)*cp5)*p55
n = 6
if(nmax .lt. 6) go to 260
aor = aor*ar
if(bm(6) .gt. aor .le. err) go to 260
sp6 = sp2*cp5 + cp2*sp5
cp6 = cp2*cp5 - sp2*sp5
p61 = p21*p51 - 0.25396825e0*p41
dp61 = p21*dp51 + dp21*p51 - 0.25396825e0*dp41
p62 = p21*p52 - 0.23809523e0*p42
dp62 = p21*dp52 + dp21*p52 - 0.23809523e0*dp42
p63 = p21*p53 - 0.19047619e0*p43
dp63 = p21*dp53 + dp21*p53 - 0.19047619e0*dp43
p64 = p21*p54 - 0.1111111e0*p44
dp64 = p21*dp54 + dp21*p54 - 0.1111111e0*dp44
p65 = p21*p55
dp65 = p21*dp55 + dp21*p55
p66 = p22*p55
dp66 = 5.0*p65
c2 = g(6,2)*cp2 + g(1,6)*sp2
c3 = g(6,3)*cp3 + g(2,6)*sp3
c4 = g(6,4)*cp4 + g(3,6)*sp4
c5 = g(6,5)*cp5 + g(4,6)*sp5
c6 = g(6,6)*cp6 + g(5,6)*sp6
br = br - 6.0*aor*(g(6,1)*p61 + c2*p62 + c3*p63 + c4*p64 + c5*p65 + c6*p66)
b = bt + aor*(g(6,1)*dp61 + c2*dp62 + c3*dp63 + c4*dp64 + c5*dp65 + c6*dp66)
b = bp - aor*((g(6,2)*sp2 - g(1,6)*cp2)*p62 + 2.0*(g(6,3)*sp3 - g(2,6)*cp3)*p63 + 0.14285714e0*p63)
$p = p63 + 3.0*(g(6,4)*sp4 - g(3,6)*cp4)*p64 + 4.0*(g(6,5)*sp5 - g(4,6)*cp5)*p55
if(nmax .lt. 7) go to 260
aor = aor*ar
if(bm(7) .gt. aor .le. err) go to 260
sp7 = (sp4 + sp4)*cp4
cp7 = (cp4 + cp4)*sp4
p71 = p21*p61 - 0.25252525e0*p51
dp71 = p21*dp61 + dp21*p61 - 0.25252525e0*dp51
p72 = p21*p62 - 0.24242424e0*p52
dp72 = p21*dp62 + dp21*p62 - 0.24242424e0*dp52
p73 = p21*p63 - 0.21212121e0*p53
0172 \( dp73 = p21 \cdot dp63 + dp21 \cdot p63 - 0.21212121e0 \cdot dp53 \)
0173 \( p74 = p21 \cdot p64 - 0.16161616e0 \cdot p54 \)
0174 \( dp74 = p21 \cdot dp64 + dp21 \cdot p64 - 0.16161616e0 \cdot dp54 \)
0175 \( p75 = p21 \cdot p65 - 0.09090909e0 \cdot p55 \)
0176 \( dp75 = p21 \cdot dp65 + dp21 \cdot p65 - 0.09090909e0 \cdot dp55 \)
0177 \( p76 = p21 \cdot p66 \)
0178 \( dp76 = p21 \cdot dp66 + dp21 \cdot p66 \)
0179 \( p77 = p22 \cdot p66 \)
0180 \( dp77 = 6.0 \cdot p76 \)
0181 \( c2 = g(7, 2) \cdot cp2 + g(1, 7) \cdot sp2 \)
0182 \( c3 = g(7, 3) \cdot cp3 + g(2, 7) \cdot sp3 \)
0183 \( c4 = g(7, 4) \cdot cp4 + g(3, 7) \cdot sp4 \)
0184 \( c5 = g(7, 5) \cdot cp5 + g(4, 7) \cdot sp5 \)
0185 \( c6 = g(7, 6) \cdot cp6 + g(5, 7) \cdot sp6 \)
0186 \( c7 = g(7, 7) \cdot cp7 + g(6, 7) \cdot sp7 \)
0187 \( br = br - 7.0 \cdot aor \cdot (g(7, 1) \cdot p71 + c2 \cdot p72 + c3 \cdot p73 + c4 \cdot p74 + c5 \cdot p75 + c6 \cdot p76 + c7 \cdot p77) \)
0188 \( bt = bt + aor \cdot (g(7, 1) \cdot dp71 + c2 \cdot dp72 + c3 \cdot dp73 + c4 \cdot dp74 + c5 \cdot dp75 + c6 \cdot dp76 + c7 \cdot dp77) \)
0189 \( bp = bp - aor \cdot ((g(7, 2) \cdot sp2 - g(1, 7) \cdot cp2) \cdot p72 + 2.0 \cdot (g(7, 3) \cdot sp3 - g(2, 7) \cdot cp3) \cdot p73 + 3.0 \cdot (g(7, 4) \cdot sp4 - g(3, 7) \cdot cp4) \cdot p74 + 4.0 \cdot (g(7, 5) \cdot sp5 - g(4, 7) \cdot cp5) \cdot p75 + 5.0 \cdot (g(7, 6) \cdot sp6 - g(5, 7) \cdot cp6) \cdot p76 + 6.0 \cdot (g(7, 7) \cdot sp7 - g(6, 7) \cdot cp7) \cdot p77 \)
0190 \( c = n = 8 \)
0191 \( if(nmax < it. 8) go to 260 \)
0192 \( aor = aor \cdot ar \)
0193 \( if(bm(8) \cdot aor \cdot le. err) go to 260 \)
0194 \( sp8 = sp2 \cdot cp7 + cp2 \cdot sp7 \)
0195 \( cp8 = cp2 \cdot cp7 - sp2 \cdot sp7 \)
0196 \( p81 = p21 \cdot p71 - 0.25174825e0 \cdot p61 \)
0197 \( p82 = p21 \cdot p72 - 0.24475524e0 \cdot p62 \)
0198 \( p83 = p21 \cdot p73 - 0.22377622e0 \cdot p63 \)
0199 \( p84 = p21 \cdot p74 - 0.18881118e0 \cdot p64 \)
0200 \( p85 = p21 \cdot p75 - 0.13986013e0 \cdot p65 \)
0201 \( p86 = p21 \cdot p76 - 0.07692307e0 \cdot p66 \)
0202 \( p87 = p21 \cdot p77 \)
0203 \( p88 = p22 \cdot p77 \)
0204 \( dp88 = dp77 + dp21 \cdot p77 \)
0205 \( c2 = g(8, 2) \cdot cp2 + g(1, 8) \cdot sp2 \)
0206 \( c3 = g(8, 3) \cdot cp3 + g(2, 8) \cdot sp3 \)
0207 \( c4 = g(8, 4) \cdot cp4 + g(3, 8) \cdot sp4 \)
0208 \( c5 = g(8, 5) \cdot cp5 + g(4, 8) \cdot sp5 \)
0209 \( c6 = g(8, 6) \cdot cp6 + g(5, 8) \cdot sp6 \)
0210 \( c7 = g(8, 7) \cdot cp7 + g(6, 8) \cdot sp7 \)
0211 \( c8 = g(8, 8) \cdot cp8 + g(7, 8) \cdot sp8 \)
0212 \( br = br - 8.0 \cdot aor \cdot (g(8, 1) \cdot p81 + c2 \cdot p82 + c3 \cdot p83 + c4 \cdot p84 + c5 \cdot p85 + c6 \cdot p86 + c7 \cdot p87) \)
0213 \( bp = bp - aor \cdot ((g(8, 2) \cdot sp2 - g(1, 8) \cdot cp2) \cdot p82 + 2.0 \cdot (g(8, 3) \cdot sp3 - g(2, 8) \cdot cp3) \cdot p83 + 3.0 \cdot (g(8, 4) \cdot sp4 - g(3, 8) \cdot cp4) \cdot p84 + 4.0 \cdot (g(8, 5) \cdot sp5 - g(4, 8) \cdot cp5) \cdot p85 + 5.0 \cdot (g(8, 6) \cdot sp6 - g(5, 8) \cdot cp6) \cdot p86 + 6.0 \cdot (g(8, 7) \cdot sp7 - g(6, 8) \cdot cp7) \cdot p87) \)
0214 \( bp = bp - aor \cdot ((g(8, 2) \cdot sp2 - g(1, 8) \cdot cp2) \cdot p82 + 2.0 \cdot (g(8, 3) \cdot sp3 - g(2, 8) \cdot cp3) \cdot p83 + 3.0 \cdot (g(8, 4) \cdot sp4 - g(3, 8) \cdot cp4) \cdot p84 + 4.0 \cdot (g(8, 5) \cdot sp5 - g(4, 8) \cdot cp5) \cdot p85 + 5.0 \cdot (g(8, 6) \cdot sp6 - g(5, 8) \cdot cp6) \cdot p86 + 6.0 \cdot (g(8, 7) \cdot sp7 - g(6, 8) \cdot cp7) \cdot p87) \)
0215 \( bp = bp - aor \cdot ((g(8, 2) \cdot sp2 - g(1, 8) \cdot cp2) \cdot p82 + 2.0 \cdot (g(8, 3) \cdot sp3 - g(2, 8) \cdot cp3) \cdot p83 + 3.0 \cdot (g(8, 4) \cdot sp4 - g(3, 8) \cdot cp4) \cdot p84 + 4.0 \cdot (g(8, 5) \cdot sp5 - g(4, 8) \cdot cp5) \cdot p85 + 5.0 \cdot (g(8, 6) \cdot sp6 - g(5, 8) \cdot cp6) \cdot p86 + 6.0 \cdot (g(8, 7) \cdot sp7 - g(6, 8) \cdot cp7) \cdot p87) \)

A-41
0229 $\times p83 + 3.0*(g(8,4)*sp4-g(3,8)*cp4)*p84 + 4.0*(g(8,5)*sp5-g(4,8)*cp5)*p$
0230 $85 + 5.0*(g(8,6)*sp6-g(5,8)*cp6)*p86 + 6.0*(g(8,7)*sp7-g(6,8)*cp7)*p87$
0231 $+ 7.0*(g(8,8)*sp8-g(7,8)*cp8)*p88$
0232 if (nmax .lt. 9) go to 260
0233 aor = aor * ar
0234 if (bni(9) .lt. 1.0e. err) go to 260
0235 sp9 = (sp5 + sp5) * cp5
0236 cp9 = (cp5 + sp5) * (cp5 - sp5)
0237 p91 = p21 * p81 - 0.25128205e0 * dp91
0238 dp91 = p21 * dp81 + dp21 * p81 - 0.25128205e0 * dp71
0239 p92 = p21 * p82 - 0.24615384e0 * dp92
0240 dp92 = p21 * dp82 + dp21 * p82 - 0.24615384e0 * dp72
0241 p93 = p21 * p83 - 0.23076923e0 * dp93
0242 dp93 = p21 * dp83 + dp21 * p83 - 0.23076923e0 * dp73
0243 p94 = p21 * p84 - 0.20512820e0 * dp94
0244 dp94 = p21 * dp84 + dp21 * p84 - 0.20512820e0 * dp74
0245 p95 = p21 * p85 - 0.16923076e0 * dp95
0246 dp95 = p21 * dp85 + dp21 * p85 - 0.16923076e0 * dp75
0247 p96 = p21 * p86 - 0.13207962e0 * dp96
0248 dp96 = p21 * dp86 + dp21 * p86 - 0.13207962e0 * dp76
0249 p97 = p21 * p87 - 0.06666666e0 * dp97
0250 dp97 = p21 * dp87 + dp21 * p87 - 0.06666666e0 * dp77
0251 p98 = p21 * p88
0252 dp98 = p21 * dp88 + dp21 * p88
0253 p99 = p22 * p88
0254 dp99 = 8.0 * p98
0255 c2 = g(9, 2) * sp2 + g(1, 9) * cp2
0256 c3 = g(9, 3) * sp3 + g(2, 9) * cp3
0257 c4 = g(9, 4) * sp4 + g(3, 9) * cp4
0258 c5 = g(9, 5) * sp5 + g(4, 9) * cp5
0259 c6 = g(9, 6) * sp6 + g(5, 9) * cp6
0260 c7 = g(9, 7) * sp7 + g(6, 9) * cp7
0261 c8 = g(9, 8) * sp8 + g(7, 9) * cp8
0262 c9 = g(9, 9) * sp9 + g(8, 9) * cp9
0263 br = br - 9.0 * aor * (g(9, 1) * p91 + c2 * p92 + c3 * p93 + c4 * p94 + c5 * p95 + c6 * p96 + c7 * p97)
0264 $7 + c8 * p98 + c9 * p99$
0265 bt = bt * aor * (g(9, 1) * dp91 + c2 * dp92 + c3 * dp93 + c4 * dp94 + c5 * dp95 + c6 * dp96 + c7 * dp97)
0266 $+ c8 * dp98 + c9 * dp99$
0267 bp = bp * aor * ((g(9, 2) * sp2 - g(1, 9) * cp2) * p92 + 2.0 * (g(9, 3) * sp3 - g(2, 9) * cp3))
0268 $+ p93 + 3.0 * (g(9, 4) * sp4 - g(3, 9) * cp4) * p94 + 4.0 * (g(9, 5) * sp5 - g(4, 9) * cp5) * p$
0269 $+ 5.0 * (g(9, 6) * sp6 - g(5, 9) * cp6) * p96 + 6.0 * (g(9, 7) * sp7 - g(6, 9) * cp7) * p97$
0270 $+ 7.0 * (g(9, 8) * sp8 - g(7, 9) * cp8) * p98 + 8.0 * (g(9, 9) * sp9 - g(8, 9) * cp9) * p99$
0271 n = 10
0272 if (nmax .lt. 10) go to 260
0273 aor = aor * ar
0274 if (bni(10) .lt. 1.0e. err) go to 260
0275 sp10 = sp2 + cp9 + sp2
0276 cp10 = cp2 + cp9 - sp2
0277 p101 = p21 * p91 - 0.25098039e0 * p81
0278 dp101 = p21 * dp91 + dp21 * p91 - 0.25098039e0 * dp81
0279 p102 = p21 * p92 - 0.24705882e0 * p82
0280 dp102 = p21 * dp92 + dp21 * p92 - 0.24705882e0 * dp82
0281 p103 = p21 * p93 - 0.23529411e0 * p83
0282 dp103 = p21 * dp93 + dp21 * p93 - 0.23529411e0 * dp83
0283 p104 = p21 * p94 - 0.21568627e0 * p84
0284 dp104 = p21 * dp94 + dp21 * p94 - 0.21568627e0 * dp84
0285 p105 = p21 * p95 - 0.18823529e0 * p85

A-42
NEWMAG

0286  dp105=dp21*dp95+dp21*dp95-0.18823529e0*dp85
0287  dp106=p21*dp96+dp21*dp96-0.15294117e0*dp86
0288  dp107=p21*dp97-0.10980392e0*dp87
0289  dp107=p21*dp97-0.10980392e0*dp87
0290  dp108=p21*dp98+dp21*dp98-0.05882352e0*dp88
0291  dp109=p21*p99
0292  dp109=p21*dp99+dp21*dp99
0293  dp1010=9.0*p109
0294  c2=g(10,2)*cp2*g(1,10)*sp2
0295  c3=g(10,3)*cp3*g(2,10)*sp3
0296  c4=g(10,4)*cp4*g(3,10)*sp4
0298  c5=g(10,5)*cp5*g(4,10)*sp5
0299  c6=g(10,6)*cp6*g(5,10)*sp6
0300  c7=g(10,7)*cp7*g(6,10)*sp7
0301  c8=g(10,8)*cp8*g(7,10)*sp8
0302  c9=g(10,9)*cp9*g(8,10)*sp9
0303  c10=g(10,10)*cp10+g(9,10)*sp10
0304  br=br-10.0*aor*(g(10,1)*p101+c2*p102+c3*p103+c4*p104+c5*p105+c6*p1
0305  +c7*p107+c8*p108+c9*p109+c10*p1010)
0306  bt=bt+aor*(g(10,1)*dp101+c2*dp102+c3*dp103+c4*dp104+c5*dp105+c6*dp
0307  +c7*dp107+c8*dp108+c9*dp109+c10*dp1010)
0308  bp=bp-aor*((g(10,2)*sp2-g(1,10)*cp2)*p102+2.0*(g(10,3)*sp3-g(2,10)
0309  +g(10,4)*sp4-g(3,10)*cp4)*p104+4.0*(g(10,5)*sp5-g(4
0310  +g(10,6)*sp6-g(5,10)*cp6)*p106+6.0*(g(10,7)*sp7
0311  +g(10,8)*sp8-g(7,10)*cp8)*p108+8.0*(g(10,9
0312  +g(10,10)*sp9-g(9,10)*cp9)*p109+9.0*(g(10,10)*sp10-g(9,10)*cp10)*p1010)
0313  $*cp3)*p103+3.0*(g(10,4)*sp4-g(3,10)*cp4)*p104+4.0*(g(10,5)*sp5-g(4
0314  +g(10,6)*sp6-g(5,10)*cp6)*p106+6.0*(g(10,7)*sp7
0315  +g(10,8)*sp8-g(7,10)*cp8)*p108+8.0*(g(10,9
0316  +g(10,10)*sp9-g(9,10)*cp9)*p109+9.0*(g(10,10)*sp10-g(9,10)*cp10)*p1010)
0317  bp=bp/p22*1.e-5
0318  bt=bt+1.e-5
0319  br=br+1.e-5
0320  b=sqrt(br*br+bt*bt+bp*bp)
0321  bh=sqrt(bt*bt+bp*bp)
0322  bmf=3.141592654e0-acos(bt/bh)
0323  if(bp .lt. 0.) bmf=-bmf
0324  dip=acos(bh/b)
0325  if(br .gt. 0.) dip=-dip
0326  return
0327 end
Subroutine profin(lu,type,maxhts,nprint,nrspec,lmax,hlist,alogen)

Reads ionospheric profiles

Type=1: electron and ion densities

Type=2: collision frequencies

Integer type

Character*80 bcd

Dimension hlist(maxhts),alogen(maxhts,3),en(3)

If(type .ne. 2) then

Read(lu,1010) bcd

If(nprint .gt. 1) print 1011,bcd

End if

Do 202 i=1,maxhts+l

Read(lu,1020,end=900) ht,en

If(ht .lt. 0.) then

lmax=l-1

Return

End if

If(i .ne. 1 .and. ht .ge. hlist(l-1)) then

Print *, 'ERROR PROFIN: Profile heights out of order'

Go to 999

End if

Hlist(l)=ht

If(type .eq. 1 .and. nrspec .eq. 3) en(3)=en(2)-en(1)

If(nprint .gt. 1) print 1021,ht,(en(k),k=1,nrspec)

Do 201 k=1,nrspec

Alogen(l,k)=alog(amax1(en(k),1.e-20))

Continue

Print *, 'ERROR PROFIN: Too many heights in profile'

Go to 999

Print *, 'ERROR PROFIN: Profile input not properly terminated'

Imax=-1

Return

Format(a80)

Format(/lx,a80)

Format(f7.2,4x,3e10.2)

Format(f8.2,4x,1p3e10.2)

End
subroutine qartic(q,b3,b2,b1,b0,debug,newq)

implicit real *8 (a-h,o-z)
complex*16 b3,b2,b1,b0,q,b3sq,h,i,g,hprime,grprime,sqroot,
$ p1,p2,cbert0,cbert1,cbert2,omegal,omega2,
$ rootp,rootq,rootr,fncton,ctemp,dfdq,dq

integer debug
dimension diff(4),q(4)
data omegal/(-5.d-1, 8.660254038d-1)/
data omega2/(-5.d-1,-8.660254038d-1)/
data tol/1.d-06/,imax/5/

again=0
if(newq .eq. 1) go to 30
newq=1
10 b3sq=b3**2
 h=b2-b3sq
 i=b0-(4.d0,0.d0)*b3*b1+(3.d0,0.d0)*b2**2
 g=b1+b3*((-3.d0,0.d0)*b2+(2.d0,0.d0)*b3sq)
 hprime=-i/(12.d0,0.d0)
 gprime=-g**2/(4.d0,0.d0)-h*(h**2.(3.d0,0.d0)*hprime)
 c sqroot=cdsqrt(gprime**2+(4.d0,0.d0)*hprime**3)
p1=(-.5d0,0.d0)*gprime-sqroot)
p2=(-.5d0,0.d0)*gprime+sqroot)
if(cfabs(p1) .lt. cbfabs(p2)) p1=p2
 cbert0=cdexp(cdblog(p1)/(3.d0,0.d0))
 cbert1=omegal*cbert0
 cbert2=omega2*cbert0
 c rootp=cdsqrt(cbert0-hprime/cbert0-h)
 rootq=cdsqrt(cbert1-hprime/cbert1-h)
 rootr=cdsqrt(cbert2-hprime/c bert2-h)
if(cfabs(g) .gt. 1.d-30) then
 sign=--rootp*rootq*rootr*(2.d0,0.d0)/g
 if(sign .lt. 0.d0) rootr=-rootr
end if
 if(debug .gt. 2) print 100,b3,b2,bl,b0
do 60 n=1,4
do 40 iter=1,imax
 fncton=(((q(n)+(4.d0,0.d0)*b3)*q(n)+(6.d0,0.d0)*b2)*q(n)
 $ +((4.d0,0.d0)*b1)*q(n)+b0
 dfdq=(((4.d0,0.d0)*q(n)+(12.d0,0.d0)*b3)*q(n)
 $ +((12.d0,0.d0)*b2)*q(n)+(4.d0,0.d0)*b1
 dq=fncton/dfdq
 q(n)=q(n)+dq
 testdq=cbfabs(dq/q(n))
 if(testdq .le. tol) go to 60
 continue
 if(again .eq. 1) then
 fncton=(((q(n)+(4.d0,0.d0)*b3)*q(n)+(6.d0,0.d0)*b2)*q(n)
 $ +((4.d0,0.d0)*b1)*q(n)+b0
 print 101,n,q(n),fncton,dq,again
QARTIC

0058         stop
0059       else
0060     iagain=1
0061   go to 10
0062 end if
0063       60 continue
0064       c
0065       l=0
0066   do 80 m=2,4
0067   do 80 n=m,4
0068 if(dimag(q(n)) .gt. 0.d0) go to 80
0069       l=l+1
0070 ctemp=q(n)
0071 q(n)=q(m-1)
0072 q(m-1)=ctemp
0073 80 continue
0074 if(l .eq. 2) go to 99
0075   do 81 n=1,4
0076 angq=cdang(q(n))*57.295779513d0
0077 if(angq .lt. 135.d0) angq=angq+360.d0
0078 81 diff(n)=dabs(angq-315.d0)
0079   do 82 nm=2,4
0080   do 82 n=nm,4
0081 if(diff(n) .gt. diff(nm-1)) go to 82
0082 temp=diff(n)
0083 diff(n)=diff(nm-1)
0084 diff(nm-1)=temp
0085 ctemp=q(n)
0086 q(n)=q(nm-1)
0087 q(nm-1)=ctemp
0088 82 continue
0089       c
0090       99 return
0091       100 format(/' In QARTIC: b''s =',4(1pe13.4,e12.4))
0092       101 format(8h q root ,i1,2h =,1p2e13.5,3x,10hfunction =,2e13.5,3x,
0093 $ 4hdq =,2e13.5,3x,8hiagain =,i1)
0094       end
subroutine rbars
implicit real*8 (a-h,o-z)
include 'common1.for'
include 'common2.for'
include 'common3.for'
complex*16 ngsq,sqroot,ratio,ikc,exd,exdsq,z1,z2,$
p,h10,h20,h1prm0,h2prm0,caph10,caph20,$
pd,h1d,h2d,h1prmd,h2prmd,caph1d,caph2d,$
alst,a2nd,a3rd,a4th,a1,a2,a3,a4,f1,f2$
real*8 kvraot,kvrratt,ndsq,n0sq

ngsq=dcmplx(dble(epsr),-dble(sigma)/(omega*8.85434d-12))$ sqroot=cdsqrt(ngsq-ssq)$
c if(dimag(theta) .lt. -10.d0 .or. alpha .eq. 0.) go to 20
if(d .eq. 0.) go to 10$ kvraot=dexp(dlog(wn/alpha)/3.d0)$ kvrratt=dvraot**2
avrkot=1.d0/kvraot$ avrktt=avrkot**2*0.5d0
n0sq=1.-alpha*h$ ratio=n0sq/ngsq*sqroot
p0=kvrratt*(n0sq-ssq)$ call mdhnk1(p0,h10,h20,h1prm0,h2prm0,theta,'rb 1')
caph10=h1prm0+avrktt*h10$ caph20=h2prm0+avrktt*h20
alst=caph20-zmplxi*ratio*kvraot+h20$ a2nd=caph10-zmplxi*ratio*kvraot+h10
a3rd=h2prm0-zmplxi*kvraoct*sqroot+h20$ a4th=h1prm0-zmplxi*kvraot*sqroot+h10
ndsq=1.-alpha*(h-d)$ pd=kvrratt*(ndsq-ssq)$ call mdhnk1(pd,h1d,h2d,h1prmd,h2prmd,theta,'rb 2')
caph1d=h1prmd+avrktt*h1d$ caph2d=h2prmd+avrktt*h2d
f1=h2d*a2nd-h1d*a1st
f2=h2d*a4th-h1d*a3rd
a1=ndsq*f1$ a2=zmplxi*avrko(n(caph1d*a1st-caph2d*a2nd))
a3=zmplxi*avrko(n(h2prmd*a4th-h1prmd*a3rd))
a4=c*f2$ rbar11=(a1-a2)/(a1+a2)$ rbar22=(a3+a4)/(a4-a3)$
hg=exp(-.5+alpha+.*d)*(h20*a2nd-h10*a1st)/f1
norm11=f1*f1$ norm22=f2*f2$ norm12=f1*f2$ return$ c
rbar11=(ngsq*c-sqroot)/(ngsq*c+sqroot)$ rbar22=(c-sqroot)/(c+sqroot)$
hg=zone$ norm11=(-.2.124292958d0,0.d0)$ norm22=norm11
0110  norml2=norml1
0111  return
0112  c
0113  c  flat earth
0114  20  ikc=dcmplx(0.d0,-wn)*c
0115  exd=cdexp(ikc*d)
0116  exdsq=exd*exd
0117  z1=(ngsq*c-sqroot)/(ngsq*c+sqroot)
0118  z2=(c-sqroot)/(c+sqroot)
0119  rbar11=z1*exdsq
0120  rbar22=z2*exdsq
0121  hg=exd*(zone+z1)/(zone+rbar11)
0122  norml1=(zone+rbar11)*(zone+rbar11)/exdsq
0123  norm22=(zone+rbar22)*(zone+rbar22)/exdsq
0124  norml2=(zone+rbar11)*(zone+rbar22)/exdsq
0125  return
0126  end
subroutine recvr(tlng,tclt,xtr,rho,rlng,rclt)

Returns coordinates of a point which is at a specified great circle distance and bearing angle from the input point

Input: TLNG is longitude of transmitter
TCLT is co-latitude of transmitter
XTR is geographic bearing angle of receiver
RHO is great circle distance to the receiver

Output: RLNG is longitude of receiver
RCLT is co-latitude of receiver

All coordinates, RHO and XTR are in radians
Sign convention is + for West and North

data pi/3.14159265e0,,twopi/6.28318531e0/
reduce(arg)=sign(amin1(abs(arg),1.),arg)

ctclt=cos(tclt)
stclt=sin(tclt)
br=xtr
gcd=rho

if(abs(br) .lt. twopi) go to 2
br=amod(br,twopi)
2 if(br .ge. 0.) go to 3
br=br+twopi
3 if(gcd .lt. pi) go to 5
gcd=twopi-gcd
br=br+pi
5 if(br .ge. twopi) br=br-twopi
if(br .le. 1.e-6) go to 10
if(abs(br-pi) .le. 1.e-6) go to 14
if(abs(gcd-pi) .le. 1.e-6) go to 14
cgcd=cos(gcd)
sgcd=sin(gcd)
crclt=ctclt*cgcd+stclt*sgcd*cos(br)
srclt=sqrt(1.-crlt**2)
rlng=acos(reduce(crlt))
delta=acos(reduce((cgcd-ctclt*crlt)/(stclt*srclt)))
if(br .lt. pi) delta=-delta
rlng=tlng+delta
go to 20

receiver is due north, south or on opposite longitude
10 rclt=tclt-gcd
11 rln=tlng

rcrlt=cos(rclt)
srclt=sin(rclt)
go to 99
12 rclt=-rclt
13 rln=tlng+pi
rclt=cos(rclt)
srclt=sin(rclt)
RECVR

0058  go to 20
0059   14 rclt=tclt+gcd
0060 if(rclt .lt. pi) go to 11
0061   rclt=twopi-rclt
0062  go to 13
0063  c
0064  20 if(rlng .gt. pi) go to 21
0065  if(rlng .lt. -pi) go to 22
0066  go to 99
0067  21 rlng=rlng-twopi
0068  go to 99
0069  22 rlng=rlng+twopi
0070  c
0071  99 return
0072  end
subroutine rplynm
implicit real*8(a-h,o-z)
include 'common2.for'
include 'common3.for'
complex*16 lgmtrx(30,4),prod,tlist1,tlist2
complex*8 theta
real*4 dst(30)
integer use(30)
c
if(m le 30 .and. tlist(1,m) gt 0) then
    thetar=tlist(1,m)
    theta=tlist(2,m)
c=cdcos(theta*zdtr)
    csq=c*c
    s=cdsn(theta*zdtr)
    ssq=s*s
    call nteg
    do 12 n=1,4
        lgmtrx(m,n)=logrs(n)
        adjflg=1
        m=m+1
    end if
if(max le 1) then
    print *, 'ERROR RPLYNM Insufficient tlist'
stop
else
    max=max(n,ntlist)
    adjflg=0
    return
end if
entry expoly
c Distance from theta to tlist angles
theta=theta
do 24 is=1,max
    use'1'=1
    dst1=sqrt(real(theta),t ist(is))**2
    dst2=mag(theta),tist(is)**2
24         ! use'1'=1, dst1=dst2, tlist(is)
order tlist angles according to distance
callsort(dst, ,max,use,max,1,max)
c Use only tlist angles
do 50 n=1,4
ogrsr=0
do 45 j=1,max
    use'1'=1
45     tlist(is),decomp/dbl(tlist(is) is dbi/ tlist(is) is t1 is t2 is..)
prodzone
do 44 ,j=1,max
        2=use'1',2
        if(i ne 2) then
            tlist(is),decomp/dbl(tlist(is) is dbl/tlist(is) is t) is t2..)
            prod=prod*theta,(tlist(is) is t2)/(t-list(is) is t2)
end if
continue
logrs(n) = logrs(n) * prod * lgmtrx(i1, n)
rs(n) = cdexp(logrs(n))
return
end
subroutine savemc

This routine writes the mode parameters out to the logical unit defined by LUNIT7.

include 'common1.for'
include 'common2.for'

write(lunit7,100) rho,freq,azim,codip,magfld,sigma,epsr,hprout
do 10 m=1,modes
write(lunit7,101) tp(m),nterm(m),t term(1,m),t term(2,m),
tp(m),nterm(m),t term(3,m),t term(4,m)
write(lunit7,102)
end

A-53
subroutine sortr(array,nra,index,nri,ii,jj)

algorithm 347, r.c. singleton, communications of the acm, v12, n3, mar69

sorts array into order of increasing value, from index ii to jj
also orders index simultaneously if nri > 1
the only arithmetic operation on array is subtraction
the user should consider the possibility of integer overflow
arrays iu(k) and il(k) permit sorting up to \(2^{(k+1)-1}\) elements
dimension array(1), index(1), iu(36), il(36)

if(jj .gt. nra) print *, 'warning from sortr: jj > nra'
m=1
i=ii
j=jj

5 if(i .ge. j) go to 70
10 k=i
ij=(i+j)/2
t=array(ij)
if(nri .le. 1) go to 15
n=index(ij)
15 if(array(i) .le. t) go to 20
array(ij)=array(i)
t=array(ij)
if(nri .le. 1) go to 25
index(ij)=index(i)
n=index(ij)
20 l=j
if(array(j) .ge. t) go to 40
array(ij)=array(j)
array(j)=t
t=array(ij)
if(nri .le. 1) go to 25
index(ij)=index(j)
n=index(ij)
25 if(array(i) .le. t) go to 40
array(ij)=array(i)
array(i)=t
t=array(ij)
if(nri .le. 1) go to 25
index(ij)=index(i)
n=index(ij)
30 array(l)=array(k)
array(k)=tt
if(nri .le. 1) go to 40
index(l)=index(k)
index(k)=nn
40 l=l-1
if(array(l) .gt. t) go to 40
tt=array(l)
if(nri .le. 1) go to 50
nn=index(l)
50 k=k+1
SORTR

0058  if(array(k) .lt. t) go to 50
0059  if(k .le. 1) go to 30
0060  if(l-i .le. j-k) go to 60
0061  il(m)=i
0062  iu(m)=j
0063  i=k
0064  m=m+1
0065  go to 80
0066  il(m)=k
0067  iu(m)=j
0068  j=l
0069  m=m+1
0070  go to 80
0071  m=m-1
0072  if(m .eq. 0) return
0073  i=il(m)
0074  j=iu(m)
0075  if(j-i .ge. 11) go to 10
0076  if(i .eq. ii) go to 5
0077  i=i+1
0078  if(i .eq. j) go to 70
0079  t=array(i+1)
0080  if(nri .le. 1) go to 95
0081  n=index(i+1)
0082  if(array(i) .le. t) go to 90
0083  k=i
0084  array(k+1)=array(k)
0085  if(nri .le. 1) go to 105
0086  index(k+1)=index(k)
0087  k=k-1
0088  if(t .lt. array(k)) go to 100
0089  array(k+1)=t
0090  if(nri .le. 1) go to 90
0091  index(k+1)=n
0092  go to 90
0093  end
This routine drives the generation of mode parameters using the input elist. If RPOLY is 0, then all calculations are made exactly. If RPOLY is 2, then all calculations are made approximately using the routine RPLYNM. If RPOLY is 1, then the initial calculations are approximate to refine the initial solutions and the final solutions are obtained using the exact formulation.

INCLUDE 'COMMON1.FOR/LIST'

COMMON/INPUT/FREQ,RHO,AZIM,COD,MAGFD,SIGMA,EPSPR,BETA,HPRIME,HPROUT
$ PATHID,TLONG,TLAT,RLONG,RLAT,RBEAR,DMAX,DRMIN,DRMAX
$ YEAR,MONTH,DAY,GMT,NPRINT,NGCD,INGD,MINDIR,LOST,
$ UNIT7,UX

COMMON/IONOSP/HLIST(50),INLIST(50,3),HCLIST(50),CFLIST(50,3),
$ CHARGE(3),MRATIO(3),NRSPEC,HTMX,HTRN,LHT,MHTMX,MHTMN,MHT

CHARACTER*80 PATHID
INTEGER YEAR,DAY
REAL*4 FREQ,RHO,AZIM,CODIP,MAGFD,SIGMA,EPSPR,BETA,HPRIME,HPROUT,
$ TLONG,TLAT,RLONG,RLAT,RBEAR,DMAX,DRMIN,DRMAX,GMT,
$ HTLIST,HCLIST,CFLIST,CHARGE,MRATIO

INCLUDE 'COMMON2.FOR/LIST'

COMMON/WG_IN/ELIST(2,30),TLIST(2,30),DTHETA(2),LUB(2),DEIGEN(2),
$ THTINC,FTOI,MAXITR,ALPHA,H,D,PREC,W0,ATMAX,DEBUG,TYPITR,
$ RPOLY,NRLIST

COMMON/WG_OUT/TP(30),TERM(4,30),NTERM(30),MODE(30),MODES,NMDS

REAL*4 TP,TERM,DTHETA
INTEGER DEBUG,TYPITR,RPOLY
REAL*4 ELIST,TLIST,DTHETA,LUB,DEIGEN,THTINC,FTOL,ALPHA,H,D,PREC,
$ W0,ATMAX

REAL*8 EQUIVALENCE (DTHETA,DTHETA)

INCLUDE 'COMMON3.FOR/LIST'

COMMON/FUNCN/OMEGA,EN,THETA1,THETA2,C,S,CAS,Q,F,DFF,DTHETA,
$ HG,NORM1,NORM2,NORM3,RBAR1,RBAR2,
$ NU,THER,NEWQ,ADJFIG,ISOPTP

COMMON/R MATRIX/R11,R22,R12,R21,
$ LOG11,LOG22,LOG12,LOG21,
$ D11DH,D12DH,D12DH,D12DH,HT,DEIH,TOPHT

COMMON/MATRIX/M11,M12,M13,M21,M22,M23,M31,M32,M33

INTEGER ADJFIG
REAL*8 OMEGA,EN,THETA1,THETA2,HT,DEIH,TOPHT,R(8),LOG(8),DLR(8)
COMPLEX*16 Theta,C,S,CAS,Q,F,DFF,DTHETA,
$ HG,NORM1,NORM2,NORM3,RBAR1,RBAR2,
$ R11,R22,R12,R21,R(4)
$ LOG11,LOG22,LOG12,LOG21,LOGR(4),LOGR(4)
$dll1dh, dll2dh, dll12dh, dll21dh, dllrsdh(4),
m11, m12, m13, m21, m22, m23, m31, m32, m33,
zero/(0.d0, 0.d0)/, zone/(1.d0, 0.d0)/,
zmplxi/(0.d0, 1.d0)/, zdtr/(1.745329252d-2, 0.d0)/
equivalence (thetar, theta),
(r11, rs), (logr11, logrs), (dll1dh, dllrsdh),
(r11, r), (logr11, logr), (dll1dh, dllrdh)

complex*16 theta0, stp, ratio, store1, store2, store3,
$ wterm, ecomp, mik
complex* 8 eigen(30)
real*8 cdang, reflht, capk, stpr, stpi
integer psave
character*20 reason, blank
equivalence (elist, eigen)
data blank/, reflht/70.d0/

psave=rpoly
capk=1/(1.-.5*alpha*h)
omega=6.2831853063*freq
wn=2.0958426d-2*freq
wterm=dcmplx(0.d0, -.5d0*wn*reflht)
mik=dcmplx(0.d0, -1.d3*wn)
debug=nprint
adjfig=0
newq=0
if(magfld le. 1.e-10) then
  isotrp=1
else
  if(codip eq 90. and. (azim eq 90. or. azim eq. 270.) then
    isotrp=2
  else
    isotrp=0
  end if
end if
call intcmp
if(rpoly eq 1) call rplynm
if(nprint gt 0) print 1010
kn=0
ms=0
index=1
if(elist(1, index) eq 0) go to 62
theta0=eigen(index)
kn=kn+1
ms=ms+1
mn=mode(kn)
reason=blank
theta=theta0
call iterat
fmag=dabs(f)
if(nriter ge maxitr and fmag gt ftol) then
  write(reason, 2000) fmag
  go to 50
end if
p1mag=dabs(rbar11er12/(zone-rbar11er11))
thtr=thetar
thti=thetai
if(thti .ge. 0.) then
  write(reason,2001)
go to 50
end if
if(kn .gt. 1) then
do 30 kd=1,kn-1
  if(abs(thtr-elist(1,kd)) .gt. deigen(1)) go to 30
  if(abs(thti-elist(2,kd)) .gt. deigen(2)) go to 30
  write(reason,2002) kd
  go to 50
end if
30 continue
if(ms .eq. inn) go to 35
if(rpoly .eq. 0 .and. nprint .gt. 0) print 1003
ms=ms+1
go to 33
35 if(rpoly .eq. 1) go to 60
c
if(nriter .gt. maxitr/2) then
  print *, 'Warning WVGD: Excessive iterations for this mode:'
  lost=2
end if
s=cdsin(theta*zdtr)
stp=s*capk
at=-8.6858896d3*wn*dimag(stp)
v=1.0/dreal(stp)
tp(mn)=-zmplx*cdlog(cdsqrt(zone-stp*stp)+zmplx*stp)/zdtr
c
ratio=cdsqrt(s)/(dfdht/zdtr)
store1=(zone+rbar11)**2*(zone-rbar22*r22)*ratio/rbar11
store2=(zone+rbar11)***(zone+rbar22)*ratio
store3=(zone+rbar11)**2*(zone-rbar11*r11)*ratio/rbar22
ecomp=wterm*store1*(s+hm)**2
wm=20.0*dlog10(cdabs(ecomp))
wa=cdang(ecomp)
if(nprint .gt. 0) print 1011,theta0,mn,nriter,eigen(kn),fmag,

$  pmag,at,vc,wm,wa,tp(mn)
  c
t term(1,mn)=store1/norm11
t term(2,mn)=store2/norm22
t term(3,mn)=store2/norm12*r21
t term(4,mn)=r12/r21
if(cdabs(zone-r11*rbar11) .ge. cdabs(zone-r22*rbar22)) then
  nterm(mn)=2
else
  nterm(mn)=1
end if
50 go to 60
if(rpoly eq 1) go to 63
if(nprint .gt. 0) print 1012,theta0,nriter,theta,fmag,pmag
if(rho eq 0 or npath eq 1) then
  if(rpoly eq 1) go to 63
  c OK to drop a mode at the transmitter
if (kn .eq. 30 .or. index .eq. 30) then
    kn = kn - 1
    go to 62
  end if
  do 53 m = kn, 30
  if (k .eq. 30 .or. index .eq. 30)
    go to 62
  end if
  print *, 'ERROR WVGD: Lost mode', mn, ' because ', reason
  lost = 1
  go to 999
else
  print *, 'ERROR WVGD: Lost mode', mn, ' because ', reason
  lost = 1
  go to 999
end if

index = index + 1
go to 13
nmds = kn
eigen(nmds + 1) = (0., 0.)
if (nmds .eq. 0)
  go to 65
if (rpoly .ne. 1)
  go to 999
rpoly = 0
go to 10
print *, 'ERROR WVGD: Lost all modes'
lost = 1
print *, 'ERROR WVGD: Lost all modes'
lost = 1
if (nprint .gt. 0) print 1003
rpoly = psave
return

format('initial', 6x, 'eigen', 8x, 'mag f', 5x,
$ 'mag p', 5x, 'atten', 4x, 'v/c', 8x, 'wait''s exc', 8x, 'theta''s')
format(1x, 2f7.3, i5, 2f7.3, 2(1x, 1pe9.3), 1x, 0pf8.3, 1x, f9.5,
$ 1x, f9.3, 1x, f6.3, 1x, 2f8.3)
format(1x, 2f7.3, 4x, i5, 2f7.3, 2(1x, 1pe9.3))
format('fmag=', 1pe8.2)
format('thetai .gt. 0.')
format('it matches mode', 3)
end
## DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>Agency</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Director Defense Intelligence Agency</td>
<td>Attn: VP-TPO Washington, DC 20340-6537</td>
</tr>
<tr>
<td>Director Defense Intelligence Agency</td>
<td>Attn: DIR Washington, DC 20340-6537</td>
</tr>
<tr>
<td>Defense National Technical Information Center Cameron Station Alexandria, VA 22314-6145</td>
<td></td>
</tr>
<tr>
<td>Commander Livermore Det (FC-1)</td>
<td>Field Command/DNA Lawrence Livermore National Lab P.O. Box 808 (L-317) Attn: FC-1 Livermore, CA 94550-0662</td>
</tr>
<tr>
<td>Director Joint Strat TGT Planning Staff JCS</td>
<td>Attn: JLA. (Attn: DNA REP) Offutt AFB Omaha, NE 68113</td>
</tr>
<tr>
<td>Director National Security Agency</td>
<td>Attn: B432 (C. Goodale) Ft. Meade, MD 20755-6000</td>
</tr>
<tr>
<td>Commander US Army Atmospheric Sciences Laboratory</td>
<td>Attn: SLCAS-AE-E White Sands Missile Range, NM 88002</td>
</tr>
<tr>
<td>Commander US Army Communications R&amp;D Command</td>
<td>Attn: DRCO-COM-RY (W. Kesselman) Fort Monmouth, NJ 07703</td>
</tr>
<tr>
<td>Commander US Army Nuclear &amp; Chemical Agency</td>
<td>Attn: Library 7500 Backlick Road Building 2073 Springfield, VA 22150-3198</td>
</tr>
<tr>
<td>Defense Communications Engineering Center</td>
<td>Attn: 123 (Library) 1800 Wiehl Reston, VA 22090-5285</td>
</tr>
<tr>
<td>Director Defense Intelligence Agency</td>
<td>Attn: DT-1B Washington, DC 20340-6537</td>
</tr>
<tr>
<td>Director Defense Intelligence Agency</td>
<td>Attn: DC-7B Washington, DC 20340-6537</td>
</tr>
<tr>
<td>Director Defense Nuclear Agency</td>
<td>Attn: RAAE Washington, DC 20305-1000</td>
</tr>
<tr>
<td>Commander Field Command Defense Nuclear Agency</td>
<td>Attn: FCTX0 Kirtland, NM 87115-5000</td>
</tr>
<tr>
<td>Joint Chiefs of Staff</td>
<td>Attn: C3S Washington, DC 20301</td>
</tr>
<tr>
<td>Chief Livermore Division Fld Command DNA Lawrence Livermore Laboratory</td>
<td>Attn: FC-1 P.O. Box 808 Livermore, CA 94550-0662</td>
</tr>
<tr>
<td>Director National Security Agency</td>
<td>Attn: Technical Library Ft. Meade, MD 20755-8000</td>
</tr>
<tr>
<td>Commander Harry Diamond Laboratories</td>
<td>Attn: SHCED-LW-P 2800 Powder Mill Road Adelphi, MD 20783-1197</td>
</tr>
<tr>
<td>Commander US Foreign Science &amp; Tech. Center</td>
<td>Attn: DRXST-SD 220 7th Street, NE Charlottesville, VA 22901-5396</td>
</tr>
<tr>
<td>US Army Strategic Defense Command</td>
<td>Attn: ATC-O (W. Davies) P.O. Box 1500 Huntsville, AL 35807</td>
</tr>
</tbody>
</table>
END
8-87
DTIC