A GENERAL PURPOSE
IONOSPHERIC RAY TRACING PROCEDURE

by

C. J. Coleman

APPROVED FOR PUBLIC RELEASE

UNCLASSIFIED
SUMMARY

A general purpose ionospheric ray tracing procedure is developed. The procedure is based on a numerical solution to the Haselgrove ray tracing equations and includes calculations of ionospherically induced Doppler shift. The procedure can handle a wide variety of ionospheric descriptions and includes the effect of the Earth's magnetic field.

© COMMONWEALTH OF AUSTRALIA 1993

AUG 93

APPROVED FOR PUBLIC RELEASE
CONTENTS

1 INTRODUCTION ................................................... . 1
2 THEORY .......................................................... 1
3 IMPLEMENTATION .................................................... 3
4 THE PROCEDURE ..................................................... 5
5 CONCLUSIONS ..................................................... 7
REFERENCES .......................................................... 9

APPENDICES

A The Ray Tracing Software ........................................ 11
THIS IS A BLANK PAGE
1 INTRODUCTION

Radiowave systems that rely on the ionosphere have a need for reliable predictions of propagation performance. Unfortunately, the ionosphere has an extremely complex morphology that is continually changing and so these predictions are by no means easy. The ionospheric structure can be derived from soundings and models, but the effect on propagation can only be gauged from solutions to the appropriate electromagnetic field equations. For HF radiowaves, however, a geometric optics approximation is often adequate and so the propagation can be studied through a suitable ray tracing procedure. Such a procedure needs to take account of the Earth's magnetic field and to accept a variety of ionospheric descriptions. In addition, it will need to calculate quantities such as phase path, group path and ionospheric Doppler shift. It is the purpose of the current report to describe a computer procedure that can meet these demands.

Section 2 of this report presents the theoretical results upon which the procedure is based, Section 3 outlines the numerical techniques that are employed and Section 4 discusses the use of the procedure. The computer implementation is contained in Appendix A.

2 THEORY

The core of the procedure is based on a numerical solution to the Haselgrove ray tracing equations (Haselgrove, 1955, 1960 and 1963). These ordinary differential equations (ODE's) are given by

\[ \frac{dx_i}{d\tau} = J_{ij} - K Y v_i \]  
\[ \frac{du_i}{d\tau} = L \frac{\partial X}{\partial x_i} \sum_i (K u_i + M Y v_i) \frac{\partial (Y v)}{\partial x_i} \]  

where \( X = 8.06 \times 10^{-4} V / f^2 \), \( f \) is the wave frequency (MHz), \( N \) is the electron density (electrons \( cm^{-3} \)), \( Y = f / f_i \) and \( f_i \) is the gyrofrequency (MHz). Variable \( \tau \) parameterises the trajectory, \( \mathbf{x} \) is a position vector, \( \mathbf{u} \) is a vector that is normal to the wavefront and \( \mathbf{v} \) is a unit vector in the direction of the magnetic field. Quantities \( J, K, L \) and \( M \) are defined by

\[ J = 2(1 - X - Y^2) p - Y (1 + (\mathbf{v} \cdot \mathbf{v})) \]  
\[ K = -2 X (\mathbf{u} \cdot \mathbf{v}) (p Y - 1) \]  
\[ L = -(1 - Y^2) p^2 - 2 (1 - X - Y^2) p - Y \]  
\[ M = 2 X p (p Y - 1) \]  

where \( p \) satisfies the quadratic

\[ (1 - X - Y^2) p^2 + Y (1 + (\mathbf{v} \cdot \mathbf{v})) p - (\mathbf{v} \cdot \mathbf{v})^2 = 0 \]  

and the different roots correspond to the ordinary and extraordinary rays.
In the current work, the Earth's magnetic field is represented by a tilted dipole:

$$\mathbf{B} = -\nabla \left( \frac{M(x-x_0)}{|x-x_0|^3} \right)$$  \hspace{1cm} (6)

where $M$ is the magnetic moment (Davies, 1990) and $x_0$ is the position of the dipole.

The phase path $P$ satisfies

$$\frac{dP}{dt} = \mu \cos \alpha \frac{ds}{dt}$$  \hspace{1cm} (7)

the group path $P'$

$$\frac{dP'}{dt} = \mu' \cos \alpha \frac{ds}{dt}$$  \hspace{1cm} (8)

and the ionospheric doppler shift $\Delta f$ (Bennett, 1967)

$$\frac{d\Delta f}{dt} = \frac{f}{c} \frac{d\mu}{dt} \cos \alpha \frac{ds}{dt}$$  \hspace{1cm} (9)

where $s$ is the distance along the ray ($ds = |dx|$) and $\alpha$ is the angle between the wave normal and the ray direction. The refractive index $\mu$ is derived from the Appleton-Hartree equation (Davies, 1990)

$$\mu^2 = 1 - \frac{2X(1-X)}{2(1-X) - Y^2 \sin^2 \beta \pm \sqrt{Y^4 \sin^4 \beta + 4Y^2(1-X)^2 \cos^2 \beta}}$$  \hspace{1cm} (10)

where $\beta$ is the angle between the wave normal and the magnetic field of Earth. The expression for the group refractive index $\mu'$ is derived from $\mu' = \frac{d}{dt}(\mu)$. In the case that the magnetic field can be ignored, the trajectories are calculated according to

$$\frac{dx}{dt} = u,$$  \hspace{1cm} (11)

and

$$\frac{du}{dt} = -\frac{1}{2} \frac{\partial X}{\partial x},$$  \hspace{1cm} (12)

where the new $\tau$ can be interpreted as the group path.
3 IMPLEMENTATION

The two most important issues in the implementation of the above scheme are the choice of ionospheric representation and the technique for solving the ODE's. In the present work, the ODE's are solved by a Runge-Kutta-Fehlberg (RKF) scheme that can be described as follows. Consider the vector system

$$\frac{dz}{dt} = f(z, t)$$ (13)

with $z = z_k$ at time $t_k$. Form, in order, the vectors

$$k_1 = h f(t_k, z_k),$$ (14)

$$k_2 = h f(t_k + \frac{1}{4} h, z_k + \frac{1}{4} k_1),$$ (15)

$$k_3 = h f(t_k + \frac{3}{8} h, z_k + \frac{3}{32} k_1 + \frac{9}{32} k_2),$$ (16)

$$k_4 = h f(t_k + \frac{12}{13} h, z_k + \frac{1932}{2197} k_1 + \frac{7200}{2197} k_2 + \frac{7200}{2197} k_3),$$ (17)

$$k_5 = h f(t_k + h, z_k + \frac{439}{216} k_1 - \frac{25}{216} k_2 + \frac{469}{273} k_3 - \frac{221}{273} k_4),$$ (18)

$$k_6 = h f(t_k + \frac{1}{2} h, z_k + \frac{8}{27} k_1 + 2 k_2 - \frac{25}{28} k_3 + \frac{18}{21} k_4 - \frac{10}{21} k_5).$$ (19)

These will yield fourth order

$$z_{k+1} = z_k + \frac{25}{216} k_1 + \frac{1048}{2565} k_2 + \frac{5103}{2565} k_3 + \frac{2197}{4104} k_4 - \frac{1}{5} k_5,$$ (20)

and fifth order

$$z''_{k+1} = z_k + \frac{16}{135} k_1 + \frac{696}{12825} k_2 + \frac{28561}{56430} k_3 - \frac{9}{50} k_4 + \frac{2}{55} k_5.$$ (21)

Runge-Kutta approximations to the solution $z = z_{k+1}$ at time $t_{k+1} = t_k + h$. An estimate of the error incurred in the fourth order scheme is given by the difference between the fourth and fifth order approximations. In the RKF method, the steplength is adjusted such that this error remains below a preassigned value. The fifth order solution is used, however, for improved accuracy. The resulting scheme is extremely flexible and can adjust to the demands of a highly varying ionosphere. In the present application, the vector $z$ has nine components. These consist of the position and wave normal components together with the phase path, the group path and the Doppler shift. The corresponding ODE's are given by equations (1), (2), (7), (8) and (9).
Three options for the representation of ionospheric electron density have been incorporated into the current computer procedure. The simplest of these allows the electron density to be defined through the function subroutine ELDEN. This is an extremely general facility that can accept most $C^1$ functions. (Since numerical derivatives are used in this option, it might experience difficulties if the ionosphere has extremely rapid variations.)

The second option provides for a description in terms of layer parameters. These are defined on a regular geographic grid (fixed increments in longitude and latitude) for two time slices. There are three layers (E, F1 and F2) which are defined through their heights ($h_{mE}$, $h_{mF1}$ and $h_{mF2}$), the plasma frequencies at these heights ($f_{oE}$, $f_{oF1}$ and $f_{oF2}$) and the layer thicknesses ($y_{mE}$, $y_{mF1}$ and $y_{mF2}$). The electron density profile is composed of Chapman layers, that is

$$N = N_0 C(h - h_{mE}) + N_2 C(h - h_{mF1}) + N_3 C\left(\frac{\sqrt{2}}{y_{mF2}}(h - h_{mF2})\right)$$

where

$$C(h, x) = \exp(C(1 - x - \exp(-x)))$$

and the coefficients $N_i$ are chosen such the electron density has the correct values at the layer heights (a plasma frequency of $f_o$ MHz corresponds to electron density of $12407 f_o^2$ electrons per $cm^3$). In order to estimate the density values between the grid points, some sort of interpolation scheme is required. Although sophisticated approaches such as cubic splines are desirable, these are too inefficient for a general purpose procedure. Instead, the multi-dimensional interpolation is built out of a series of localised one dimensional cubic interpolations. Consider the meridian that passes through the point of interest and calculate its intersection with the four closest parallels that pass through grid points. Derive values at the points of intersection by use of one dimensional interpolations along each of the parallels. The value at the point of interest is obtained by means of a one dimensional interpolation between the points of intersection.

There are several possibilities for the one dimensional interpolation and two options are included in the current procedure. The first provides an approximation with $C^1$ continuity. Let the grid points be located a unit distance apart and the desired point lie between the middle points a distance $u$ from the second point. If the data has values $e$, $e_2$, $e_3$ and $e_4$ at the interpolation points, the value at the desired point is approximately

$$e_0 \left(\frac{u^2 - u^3}{2} + e_1 \left(\frac{5u^2}{2} - \frac{3u^3}{2}\right) + e_2 \left(\frac{u^3}{2} + u^2 - \frac{3u^3}{2}\right) + e_3 \left(\frac{3u^3}{2} - u^3\right)\right)$$

If the data points are sparse, it has been found advisable to sacrifice the $C^1$ continuity and to use Lagrange interpolation instead. In this case, the value at the desired point is approximately

$$-\frac{e_1}{6} u(u-1)(u-2) + \frac{e_2}{2} (u+1)(u-1)(u-2) - \frac{e_3}{2} (u+1)u(u-2) + \frac{e_4}{6} (u+1)u(u-1)$$

The third option for ionospheric representation replaces the Chapman layers with a set of samples above each geographical grid point. Sampling is uniform and so the above geographic interpolation scheme can be extended to include the extra dimension. This option is by far the most efficient and is recommended when a large number of rays are to be traced.
The ray tracing procedure has been implemented as the FORTRAN subroutine HASEL. Each call to this subroutine will trace rays that are launched at a given elevation, in a given direction, from a given starting point and with a given wave frequency. The rays can be traced for up to three hops when the effect of Earth's magnetic field is included and for up to ten hops when not. A typical call to the subroutine is of the form:

CALL HASEL(Long, Lat, Elav, Bear, Freq, Cha, Nos, Out, TypeTol, M, L)

where the input parameters are defined as follows:

1) Long is a real*8 variable that specifies the geographic longitude of the start point (in degrees).
2) Lat is a real*8 variable that specifies the geographic latitude of the start point (in degrees).
3) Elav is a real*8 variable that specifies the initial elevation of the ray (in degrees).
4) Bear is a real*8 variable that specifies the initial bearing of the ray (degrees East from North).
5) Freq is a real*8 variable that specifies the wave frequency (in MHz).
6) Tol is a real*8 variable that specifies the tolerance for the RKF method at each step (in kilometres).
7) Cha is a character variable that has the value ‘y’ if the effect of Earth’s magnetic field is to be included and ‘n’ if not.
8) Nos is an integer variable that specifies the number of hops that are required (this parameter is altered on exit from the subroutine).
9) M is an integer variable that specifies the type of ionospheric representation. The analytic representation using subroutine ELDEN is chosen when M=7, the layer model defined on a geographic grid when M=6 or -6 and the representation in terms of height samples on a geographic grid when M=10 or -10. (Negative values of M invoke the Lagrange interpolation option.)
10) L is an integer variable that specifies the channel number on which details concerning the ray trajectories are to be output (if L < 1 this information will be omitted). For each step of the RKF scheme, this option outputs the height above the ground, the longitude and the latitude. In addition, the angle between the ray and Earth’s magnetic field is output.

The function option (M=7) defines the ionosphere through the real*8 function subroutine ELDEN. This subroutine has four real*8 arguments consisting of time elapsed (measured in seconds from the start of ray tracing), longitude (in degrees), latitude (in degrees) and distance from Earth’s centre (in kilometres). A typical example of ELDEN is as follows:

```fortran
REAL*8 ELDEN(t,lon,lat)
implicit real*8 (a-l,n-z)
common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foF2,hmF2,ymF2,re,model
H=100.0d0
hmF2=350.0d0
foF2=12.0d0
vel=.04
x=(r-6378.1d0-hmF2-vel*t)/H
NF2=12407*foF2**2
ELDEN=NF2*exp(1.0d0-x-exp(-x))
return
end
```

This represents a Chapman layer ionosphere with peak height that rises at a constant speed.
The common block is optional, but can be used to pass back information concerning the layer heights and hence allow a meaningful labelling of rays.

The grid options (M = 6, -6, 10 or -10) require further information to be passed from the calling program through a common block. The following statement will need to be placed in the calling program:

```common /grid/MinLa,MinLo,MinR,DLa,DLo,DR,DT,Grid,NR,NLa,NLo```

with the parameters defined as follows:

- **MinLa** is a real*8 variable that specifies the minimum latitude on the grid (in degrees).
- **MinLo** is a real*8 variable that specifies the minimum longitude on the grid (in degrees).
- **MinR** is a real*8 variable that specifies the minimum distance (in kilometres) of the grid from Earth’s centre (options 10 or -10).
- **DLa** is a real*8 variable that specifies the increment in latitude on the grid (in degrees).
- **DLo** is a real*8 variable that specifies the increment in longitude on the grid (in degrees).
- **DR** is a real*8 variable that specifies the increment in height on the grid (in kilometres).
- **DT** is a real*8 variable that specifies the increment in time between the time slices (in seconds).
- **GRID** is a three argument real*4 array that contains grid values of either the layer parameters or the electron density. The third argument of the array fixes the time slice and can only have the values 1 or 2. Geographical location is specified by the second argument. As this argument increases, it runs eastwards along rows of constant latitude in turn. The rows start at the minimum latitude and move upwards. The meaning of the first argument depends on the value of M. For M=6 or -6, the argument values 1 to 9 label the layer parameters foE, hmE, ymE, foF1, hmF1, ymF1, foF2, hmF2, ymF2 in order (frequencies in MHz and lengths in kilometres). In the case that M=10 or -10, the first argument labels the height samples of electron density (electrons cm\(^{-3}\)) starting at the lowest height and moving upwards. The dimensions of the first and second arguments should be set to the largest values that will be required. (For options M=6, -6, 10 or -10, the array GRID must be defined in the calling program.)
- **NR** is an integer variable that specifies the number sample heights.
- **NLa** is an integer variable that specifies the number of sample latitudes.
- **NLo** is an integer variable that specifies the number of sample longitudes.

The output from HASEL is contained in the variables Out, Type and Nos. These are defined as follows:

1. **Nos** is an integer variable that specifies the total number of rays traced.
2. **Out** is a two argument real*8 array (dimensions 11 and 14 respectively) that contains information about the rays. Each ray is labelled by the second argument (values from 1 to Nos). The first argument labels the ray attributes as follows:

   - Out(1,* ) = Doppler shift (Hz)
   - Out(2,* ) = ground range (kilometres)
   - Out(3,* ) = group path (kilometres)
   - Out(4,* ) = phase path (kilometres)
   - Out(5,* ) = maximum height (kilometres)
Out(6,*) = initial elevation (degrees)
Out(7,*) = final elevation (degrees)
Out(8,*) = final longitude (degrees)
Out(9,*) = final latitude (degrees)
Out(10,*) = bearing (degrees East from North)
Out(11,*) = wave frequency (MHz).

3) Type is a character*21 array (dimension 14) that contains information concerning the mode of the ray that corresponds to the argument.

There is an additional facility for situations where the ionosphere is periodic in longitude. If the value of variable nos is negated, the geographic mesh (options m= 10, -10, 6 or -6 only) will be treated as one period of a periodic structure. This facility is extremely useful for an ionosphere that does not vary with longitude (set nlo=1 in this case).

5 CONCLUSIONS

The present report has described a ray tracing procedure that can analyse the propagation of radiowaves through a variety of ionospheres and includes the effect of the Earth's magnetic field. There are numerous uses for the procedure, mainly in the areas of propagation prediction and ionospheric research. In particular, two recent applications are the simulation of backscatter ionograms and some investigations of the Doppler characteristics of radiowaves that propagate through travelling ionospheric disturbances. A version that estimates irregularity induced Doppler spreading has also been developed and this has been used to investigate the phenomenon of Doppler spread clutter.
REFERENCES


THIS IS A BLANK PAGE
APPENDIX A The Ray Tracing Software

The following routine performs ionospheric ray tracing. It can accept a variety of ionospheres and uses a Runge-Kutta-Fehlberg scheme to solve the Haselgrove equations.

Unless otherwise stated, all distances are in kilometres, angles in degrees, frequencies in MegaHertz and time in seconds.

***** input *****

REAL*8
along = longitude of the start point (degrees E of W)
alat = latitude of start point (degrees, -ve for S of Equator)
elev = initial elevation (degrees from horizontal)
bear = initial bearing (degrees east from north)
f = wave frequency
tol = tolerance (Kms) at each step of raytracing (a value of 1.d-6 is sufficient in most cases)

CHARACTER
cha = 'y' if magnetic fields are to be included, 'n' otherwise

INTEGER
nos = number of hops (changed on output to total number of rays)
l = channel number for ray trajectory information (0 if no information required)
m = 7 for an analytic model that is described by REAL*8 function ELDEN
   6 for a description in terms of layer parameters on a 2D grid at two time slices
   10 for a description in terms of electron density values on a 3D grid at two time slices

***** output *****

REAL*8
out(1,*) = doppler shift (Hz)
out(2,*) = ground range
out(3,*) = group path
out(4,*) = phase path
out(5,*) = maximum height
out(6,*) = initial elevation * refers to the ray number
out(7,*) = final elevation
out(8,*) = final longitude
out(9,*) = final latitude
out(10,*) = bearing
out(11,*) = frequency

CHARACTER*21
 typo(*) = mode

INTEGER
noa = number of rays traced

Note:
The program outputs the ray trajectory and values of the angle between the ray and Earth's magnetic field. Each line of output will consist of a height, a longitude, a latitude and an angle.

***** subroutines required *****

LAYERHT, RKF, PUNC, LAYERX, TERP, DTIRP, ELDEN, ELECTRONR, ELDHNS, ELECTRON, MAGNETIC

***** common block (to be set by user under certain options) *****

The parameters of the common block GRID are, in order,

REAL*8
blhla = lowest value of latitude
blhlo = lowest value of longitude
rmg = height of ionosphere above the Earth centre
dla = latitude increment
dlo = longitude increment
* **drg** = height increment (for m = 10 only)
* **dtz** = time between temporal slices
* **INTEGER**
* **nr** = number of height samples for m = 10 or 9 for m = 6
* **nla** = number of latitude samples
* **nlo** = number of longitude samples
* **REAL*4**
  * **yp(11,12,13)** = electron density samples if m = 10
  or layer parameter samples if m = 6

for m=6 yp contains the il\textsuperscript{th} layer parameter in the list
foE,hmB,ymB,foF1,hmF1,ymF1,foF2,hmF2,ymF2 (frequencies
in MHz and heights in Km)

for m=10 yp contains electron density (electrons per cubic cm)
samples at height rmg+(11-1)*drg

12 = (i-1)*nlo+j for the samples at latitude blhla+(i-1)*dla
and longitude blhlo+(j-1)*dlo

13 labels the time slice (1 or 2)

Notes:
1) Maximum values of il and 12 have been set at 61 and 961. If
these limits are to be changed, remember to alter for all
occurrences of yp.
2) A maximum value of 10Km is advised for drg , 5 degrees for dla
and 5 degrees for dlo.

switches
1) If tol = -tolerence the program will execute faster, but the
ray path will be based on a stationary ionosphere.
2) If 1<1 no ray path data will be output.
3) If m<0 Lagrange interpolation is used.
4) If nos<0 the ionosphere will be given a periodic extension in
longitude.

common blocks that are internal

**DAT** holds (in order) current values (real*8) of layer parameters
foE, hmB, ymB, foF1, hmF1, ymF1, foF2, hmF2 and ymF2. The final two
parameters are the Earths radius (real*8 in Kms) and the current
model (an integer).

**APPROX** consists of a single character variable that has value ‘L’
for Lagrange interpolation and ‘S’ for smooth jointed interpolation.

**MAG** has eight real*8 variables followed by an integer variable. The
real variables are the current gyrofrequency (MHz), the wave
frequency (MHz), the Cartesian coordinates (Earth centred) of the
magnetic dipole and the dipole moments. The integer variable has
the value 0 when the magnetic is to be ignored, a value 1 when
ordinary rays are to be traced and value -1 when extraordinary rays
are to be traced.

```
subroutine HASEL(along, alat, elav, bear, f, cha, nos, out, typo, tol, m, l)
implicit real*8 (a-h, o-z)
real*4 yp(61,961,2)
character type*21, typo*21, cha, app
dimension x(11), ytmp(11), xx(11), xtmp(11), out(11,14), typo(14)
& v(3), ryz(3,3)
common /dat/foEhmB, ymE, foF1, hmF1, ymF1, foF2, hmF2, ymF2, re, model
common /grid/blhla, blhlo, rmg, dla, dlo, drg, dtz, yp, nr, nla, nlo
common /approx/app
common /mag/fhs, from, ym, zm, px, py, pz, ir
re=6378.135
scale=45./atan(1.)
n=9
nor=90
fr=1
if(m.lt.0) then
  app='S'
else
  app='L'
endif
model=abs(m)
if(model.eq.7) dtz=1.
```
if(tol.lt.0.0) then
dtw=dtz
dtz=0.0
endif
tolx=abs(tol)

****** dipole moment = (pxt,pyt,pzt) and dipole location = (xm,ym,zm) ******
dlat=77.75
dlong=295.75
pxt=cos(dlat/scale)*cos(dlong/scale)
pyt=cos(dlat/scale)*sin(dlong/scale)
pzt=sin(dlat/scale)
xm=-434.66
ym=199.47
zm=80.25
fhs=2.8*3.1*re**3
px=fhs*pxt/fr
py=fhs*pyt/fr
pz=fhs*pzt/fr
if(cha.ne.'y') fhs=0.0

trhla=blhla+real(nla-l)*dla
trhlo=blhlo+real(nlo-l)*dlo

if(abs(nos).eq.2) nhop=3
if(abs(nos).ge.3) nhop=7
endif

********** ray type loop ******
if(ira.eq.0) then
chi='N'
else if(ira.eq.1) then
chi='0'
else if(ira.eq.2) then
chi='X'
endif
dr=sin(elav/scale)
dt=cos(elav/scale)*cos(bear/scale)
dp=cos(elav/scale)*sin(bear/scale)
phi=atan(1.)*along/45.
theta=2.*atan(1.)-atan(1.)*alat/45.
x(1)=re*sin(theta)*cos(phi)
x(2)=re*sin(theta)*sin(phi)
x(3)=re*cos(theta)
x(4)=dr*sin(theta)*cos(phi)+dt*cos(theta)*cos(phi)-dp*sin(phi)
x(5)=dr*sin(theta)*sin(phi)+dt*cos(theta)*sin(phi)+dp*cos(phi)
x(6)=dr*cos(theta)-dt*sin(theta)
t=0.0
istep=1
type='s'

x(7)=0.0
x(8)=0.0
x(9)=0.0

********** hop loop ******
zmax=0.0
sc=sqrt(x(4)**2+x(5)**2+x(6)**2)
x(4)=x(4)/sc
x(5)=x(5)/sc
x(6)=x(6)/sc
if(chl.eq.'y.' and ihop.gt.1) then
  if(ihop.eq.2) then
do k=1,n
  xtmp(k)=x(k)
  enddo
  ir=1
  chl='O'
else
  if(ihop.eq.1 or ihop.eq.4 or ihop.eq.6) ir=-1
  if(ihop.eq.5 or ihop.eq.7) ir=1
  if(ihop.eq.3) then
do k=1,n
  ytmp(k)=x(k)
  x(k)=xtmp(k)
  enddo
c11=type
  ita1=ita+1
endif
if(ihop.eq.4) then
do k=1,n
  xtmp(k)=x(k)
  cl2=type
  ita2=ita+1
endif
if(ir.eq.1) chl='O'
if(ir.eq.-1) chl='X'
if(ihop.eq.4 or ihop.eq.5) then
do k=1,n
  x(k)=ytmp(k)
  enddo
type=c11
  ita=ita1
endif
if(ihop.eq.6 or ihop.eq.7) then
do k=1,n
  x(k)=xtmp(k)
  type=c12
  ita=ita2
endif
endif
endif
zad=0.0
zed=0.0
ita=ita1
type(ita:ita+1)='S'/chl//'
itx2
if(chl.eq.'W') then
type(ita:ita+1)=' '
itb=ita+1
itx=itx+1
endif
**** start of calculations for an individual hop **************
999 istep=istep+1
do i=1,6
  xx(i)=x(i)
endo
dopi=x(7)
dopx=x(8)
pp=x(9)
sbd=zad
zbd=zad
xlong=xlong
xlat=xlat
**** ODES solver ***********************
hmax=50.
htol=0.0
h=hmax
call rkf(t,x,n,h,htol,hmin,hmax)
* x(1) to x(3) = Cartesian coordinates of current ray position
* x(7) to x(9) = group path, doppler shift and phase path at this
* location
UNCLASSIFIED

---

red=sqrt(x(1)**2+x(2)**2+x(3)**2)
zed=zed
zed=red-red
rlat=90.-acos(x(3)/red)*scale
if(abs(x(1)).lt.1.0d-10) the:
rlong=atan2(x(2),1.0d-10)*scale
else
rlong=atan2(x(2),x(1))*scale
endif
if(rlong.lt.0.0) rlong=rlong+360.0
if(1.gt.0) then
    call magnetic(x,v,ryz,ha,dig,dcl,m)
v=x(4)*v(1)+x(5)*v(2)+x(6)*v(3)
x=sqrt(abs(x(4)**2+x(5)**2+x(6)**2))
angle=acos(.999999*vz/(x*x));scale
if(zed.gt.0.0) write(1,fmt='*4f15.7') zed,rlong,rlat,angle
endif
if((rlat.gt.trhla.or.rlat.lt.blala).and.model.ne.7) go to 909
if((rlong.gt.trlho.or.rlong.lt.nlhio) .and.model.ne.7.and.noz.gt.0) gc to 909

***** At the current stage of solution
***** zed= altitude of the ray
***** rlong = longitude of ray
***** rlat = latitude of ray

***** label for highest reflection layer of current hop

if(zed.gt.max(zbd,zed,zmax)) then
    if(model.eq.10) then
        call layerparm(xlong,xlat,hl,b2,h3,f1,f2,f3)
        hM0=hl
        hM2=h2
        hM3=h3
    endif
    if(zed.le.hM0) then
        type(ita, itb+1)='E'/chl/'
        itx=2
    else if(zad.le.hM2) then
        type(ita, itb)='F'/chl
        itx=3
    else if(zad.le.hM3) then
        type(ita, itb)='F2'/chl
        itx=3
    else
        type(ita, itb)='S'/chl
        itx=2
    endif
    if(chl.eq.'N') then
        type(itb, itb+1)='L'/chl/
        itx=2
    endif
endif

zmax=max(zad,zmax)
if(zad.le.600.) go to 909
if(zed.gt.0.0) go to 999
if((hop.eq.1.or.cha.ne.'y') .and.noz.gt.0) gc to 909

***** interpolation for values at ground level

pop=zed1/(zed1-zed)
do i=1,6
x(i)=xx(i)*pop*(x(i)-xx(i))
enddo
x1=0.0
xx1=0.0
cosa=0.0
do i=1,3
xl=x1*x1)**2
xx1=xx1*x1**(i+3)**2

---
cota = cosa * x(i) * x(i+3)
enddo
sd = sqrt((x(1)-xl)**2 + (x(2)-xt)**2 + (x(3)-xt)**2)
elavf = 0.5*acos(aoul/sqrt(xltt))/atan(1.)
pl = pl + pop*(x(7)-pl)
pp = pp + pop*(x(9)-pp)
dopx = dopx + pop*(x(8)-dopx)
x(7) = pl
x(8) = dopx
x(9) = pp
xlong = xlong + pop*(rlong-xlong)
xlat = xlat + pop*(rlat-xlat)
nos = nos + 1
out(1,nos) = dopx
out(2,nos) = sd
out(3,nos) = pl
out(4,nos) = pp
out(5,nos) = xmax
out(6,nos) = elavf
out(7,nos) = elavf
out(8,nos) = xlong
out(9,nos) = xlat
out(10, nos) = bear
out(11,nos) = fr

typo(nos) = type
ss = x(1)**2 + x(2)**2 + x(3)**2
xr = x(4)*x(1) + x(5)*x(2) + x(6)*x(3)
xr = x(4) - 2*u*x(1)/ss
x5 = x(5) - 2*ux*x(2)/ss
x6 = x(6) - 2*ux*x(3)/ss
if(1.gt.0) then
z=0.0
write(1,fmt='(4f15.7)') z, xlong, xlat, angle
endif
continue
continue
if(tol<0.0) dtz = dtw
return
end

**** LAYERPARM **************************************************

**** layer parameters at a given location ********************

**** input ********************************************
* real*8
* elong = longitude (deg) of sample point
* elat = latitude (deg) of sample point

**** output ********************************************
* real*8
* h1 = height of E layer
* h2 = height of F1 layer
* h3 = height of F2 layer
* f1 = critical frequency of E layer
* f2 = critical frequency of F1 layer
* f3 = critical frequency of F2 layer

**** subroutines required ****************************
* ELDENX

**** common block ****************************
* GRID (defined in HASEL)

subroutine layerparm(elong, elat, h1, h2, h3, f1, f2, f3)
implicit real*8 (a-h, o-z)
real*4 yp(61, 961, 2)
common /grid/bhla, bhlo, rmg, dla, dlo, drg, dtz, yp, xr, nla, nlo
ra = 6.378139
r0 = rmg
e0 = eldenx(elong, elat, r0, d10, dthet, dphi, 1)
r1 = r0 + drg
e1 = eldenx(elong, elat, r1, d11, dthet, dphi 1)
h0=rmg-re
h1=0.0
h2=0.0
h3=0.0
ea=0.0
eb=0.0
ec=0.0
do i=3, nr
r2=rmg+drg*(i-1)
e2=eldenx(elong, elat, r2, d12, dthet, dphi, 1)
d21=-(d12-d10)/(r2-r0)
hh=0.0
ee=0.0
hz=0.0
if(d11*d10.le.1.d-33.and.abs(d11-d10).gt.1.d-33) then
    hhs=(d11*r0-d10*r1)/(d11-d10)-re
eees=(d11*e0-d10*e1)/(d11-d10)
    if(hh.gt.100.0) then
        if(hh.gt.140.0) then
            hi=hh
            ea=ee
        else
            if(hh.gt.230.0) then
                h2=hh
                eb=ee
            else
                h3=hh
                ec=ee
            endif
        endif
    endif
endif
else
    hh=0.0
    ee=0.0
    hz=0.0
endif

    if(d11*d10.le.1.d-33.and.abs(d11-d10).gt.1.d-33) then
        hhs=(d11*r0-d10*r1)/(d11-d10)-re
eees=(d11*e0-d10*e1)/(d11-d10)
        if(hh.ge.100.0) then
            if(hh.ge.140.0) then
                h2=hh
                eb=ee
            else
                h1=hh
                ea=ee
            endif
        endif
    endif
endif
endif

    if(i.ge.3) then
        if(r1-re.lt.230.0) then
            if(d21*d20.le.1.d-33.and.abs(d21-d20).gt.1.d-33) then
                hz=(d21*r0-d20*r1)/(d21-d20)-re
                ez=(d21*e0-d20*e1)/(d21-d20)
            endif
            if(hz.gt.100.0) then
                if(hz.ge.140.0) then
                    h2=hz
                    eb=ez
                else
                    h1=hz
                    ea=ez
                endif
            endif
        endif
    endif
endif
endif
endif
endif
d10=d11
d11=d12
d20=d21
e0=e1
e1=e2
r0=r1
r1=r2
enddo
if(h1.lt.h0) hi=h0
if(h2.lt.h0) h2=h1
if(h3.lt.h0) h3=h2
if(ec.lt.1.0) then
    ae=2
    h3=r2-re
endif
f1=sqrt(abs(ea*80.6d-6))
f2=sqrt(abs(eb*80.6d-6))
f3=sqrt(abs(ec*80.6d-6))
return
end

***** ELECTRON ************************************************************

***** electron density and its gradient **************************************

***** input ***************************************************************
* real*8  x(*) - Cartesian coordinates (Earth centred) of sample point *

***** output ***************************************************************
* real*8  en = electron density (electrons per cubic cm) *
* dnx(*) = Cartesian derivatives of electron density *

***** subroutines required *****************************************************
* ELDENX, ELDEN, LAYERX and ELECTRONR *

***** common blocks ***********************************************************
* GRID and DAT (defined in HASEL) *

subroutine electron(x,en,dnx)
 implicit real*8 (a-h,o-z)
 dimension x(3),dnx(4)
 real*4 yp(61,961,2)
 common /grid/bhla,bhlo,rmg,dao,dlo,dtz,yp,dr,nla,nlo
 common /dat/fos,ham,ym,f01,lnf1,ymf1,f0f2,lnf2,ymf2,rey,mode
 r=sqrt(x(1)**2+x(2)**2+x(3)**2)
 scale=45./atan(1.)
 theta=acos(x(3)/r)
 phi=atan2(x(2),x(1))
 elat=90.-scale*theta
 elong=scale*phi

d=sqrt(x(1)**2+x(2)**2+x(3)**2)

 Parameter da should be less than 20% of the smallest geographic scale **

 scale (in degrees).
 da=.002
 if(model.eq.10) then
 en=eldenx(elong,elat,r,diff,r,dthet,dphi,1)
 dthet=scale*dthet
 dphi=scale*dphi
 if(dtz.gt.1.d-20) then
 enp=eldenx(elong,elat,r,1,2,3,-2)
 enm=en
 endif
 else
 if(model.eq.7) then
 Parameter dr should be less than 20% of the smallest height *****

 scale (in kilometers).
 dr=.2
 t=0.0
 enpp=elden(t,elong+dr,elat,r)
 enpm=elden(t,elong-dr,elat,r)
 enp=elden(t,elong,elat+dr,r)
 enm=elden(t,elong,elat-dr,r)
 if(dtz.gt.1.d-20) then
 enp=elden(t,elong,elat,1)
 enm=elden(t,elong,elat,-2)
 endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endi
diffz = .25 * (dpp + dpm + dtp + dtm)
endif

en = .25 * (enpp + enpm + entp + entm)
dphi = .5 * scale * (enpp - enpm) / da
dthet = .5 * scale * (entp - entm) / da
endif

if (dtz < 1.d - 20) then
dtime = dphi / (240. * scale)
else
dtime = (enp - ein) / dtz
endif

dnx(l) = (diffx(l) + dthet * cos(theta) * cos(phi) & - dphi * sin(phi) / sin(theta)) / r
dnx(2) = (diffx(2) + dthet * cos(theta) * sin(phi) & + dphi * cos(phi) / sin(theta)) / r
dnx(3) = (diffx(3) - dthet * sin(theta)) / r
dnxx(4) = dtime
return
end

***** LAYERX ***********************************************
***** layer parameters from grid values ***********************
***** input ***********************************************
* real *8
* elong = longitude (deg) of sample point
* elat = latitude (deg) of sample point
* integer
* iuu = time slice label (1 or 2)
***** subroutine required *************************************
* TERP
***** common blocks ********************************************
* GRID and DAT (defined in HASEL)
***** note ****************************************************
* The output is placed in the common block DAT

subroutine layerx(elong, elat, iuu)
implicit real *8 (a-h, o-z)
real *4 yp(61, 961, 2)
dimension yz(4), yr(9)
common /grid/blhla, blhlo, rmg, dla, dlo, drg, dtz, yp, nr, nla, nlo
common /dat/yy, re, model
lip = abs(iuu)
dlong = elong - blhlo
if (dlong .ge. 360.) dlong = dlong - 360.
la = min(max(2, int((elat - blhla) / dla) + 1), nla - 2)
lo = int(dlong / dlo) + 1
lo = lo - int((lo - 1) / nlo) * nlo
if (lo .lt. 1) lo = lo + nlo
nex = (elat - blhla) / dla - real(1a - 1)
dox = dlong / dlo - real(lo - 1)
n0 = (la - 2) * nlo
n1 = n0 + nlo
n2 = n1 + nlo
n3 = n2 + nlo
do = 1, 9
do k = 0, 3
kk = lo - 1 + k
if (kk .lt. 1) kk = kk + nlo
if (kk .gt. nlo) kk = kk - nlo
x0 = yp(l, n0 + kk, lip)
x1 = yp(1, n1 + kk, lip)
x2 = yp(1, n2 + kk, lip)
x3 = yp(1, n3 + kk, lip)
yz(k + 1) = terp(dax, x0, x1, x2, x3)
enddo
yy(1) = terp(dox, yz(1), yz(2), yz(3), yz(4))
enddo
return
end
electron density and gradients from grid values

input

real*8
* along = longitude (deg) of sample point
* elat = latitude (deg) of sample point
* r = distance from Earth centre (Kms)
* integer
* iuu = time slice label (1 or 2)

output

real*8
* eldenx = electron density (electrons per cubic cm)
* differ = derivative in vertical direction
* dthet = derivative in counter latitudinal direction
* dphi = derivative in longitudinal direction

subroutines required

GRID (defined in HASEL)

real*8 function eldenx(elong,elat,r,differ,dtheta,dphi,iiu)
implicit real*8 (a-h,o-z)
real*4 yp(61,961,2)
dimension yz(4),yzt(4),e(4),dt(4),dp(4)
common /grid/bhlal,bhlao,rmg,dla,dlc,drg,dtz,yp,nn,nnl,nnlo
lip=abs(iuu)
if(r.lt.rmg.or.r.gt.rmg+(nr-l)*drg) then
eldenx=0.0
differ=0.0
dtheta=0.0
dphi=0.0
else
enddo

eldenx=eldenx+dterp(dax,yzt(1),yzt(2),yzt(3),yzt(4))
enddo

yyyy=yyyy+dterp(dox,yz(1),yz(2),yz(3),yz(4))
enddo

eldenx=eldenx+yyyy
else
endif
endif
yp=0.
endif
  e(1+2)=yy
dt(1+2)=yyt
dp(1)=yy
enddo
if(lr.lt.2) then
e(1)=0.0
dt(1)=0.0
dp(1)=0.0
endif
e1denx=terp(drx,e(1),e(2),e(3),e(4))
if(e1denx.lt.0.0) then
e1denx=0.0
diffr=0.0
dtet=0.0
dphi=0.0
else
  if(iuu.gt.0) then
diffr=terp(drx,e(1),e(2),e(3),e(4))
dtet=terp(drx,dt(1),dt(2),dt(3),dt(4))
dphi=terp(drx,dp(1),dp(2),dp(3),dp(4))
else
diffr=0.0
dtet=0.0
dphi=0.0
endif
e1denx=e1denx/dla
dtet=dtet/db
endif
dphi=dphi/dlo
diffr=diffr/drg
return
end

***** TERP ******************************************************
***** cubic interpolation ****************************************
***** input ********************************************************
* real*8
* p0, p1, p2 and p3 = samples (in order) at points spaced a unit
* distance apart.
* d2 = distance of test point from second sample point.
***** output *********************************************************
* real*8
* terp = value at test point
***** common block ********************************************************
* APPROX consists of a single character variable that has value 'L'
* for Lagrange interpolation and 'S' for smooth jointed interpolation.
***** real*8 function terp(d2,p0,p1,p2,p3)
implicit real*8 (a-h,o-z)
character app
common /approx/app
  if(app.eq.'L') then
d1=d2+1.
d3=d2-1.
d6=d2-2.
terp=(p3*d1*d2*d3-p0*d2*d3*d4)/6.+(p1*d1*d3*d4-p2*d1*d2*d4)/2.
else
terp=p1+.5*(p2-p0)*d2+(p0-2.5*p1+2.*p2-.5*p3)*d2**2
&+(.5*p3-1.5*p2+1.5*p1-.5*p0)*d2**3
endif
return
end
***** DTERP ****************************
***** derivative of cubic interpolation ****************************
***** input ****************************
* real*8
* p0, p1, p2 and p3 = samples (in order) at points spaced a unit
distance apart.
* d2 = distance of test point from second sample point.
***** output ****************************
* real*8
* dterp = derivative at test point
***** common block *******************************
* APPROX consists of single character variable that has value 'L'
* for Lagrange interpolation and 'S' for smooth jointed interpolation.

real*8 function dterp(d2,p0,p1,p2,p3)
implicit real*8 (a-h,o-z)
character app
common /approx/app
if(app.eq.'L') then
d1=d2+1.
d3=d2-1.
d4=d2-2.
dterp=(p3*(d2*d3+d1*d3+d1*d2)-p0* (d3*d4+d2*d4+d2*d3))/6. &+(p1*(d3*d4+d1*d4+d1*d3)- p2*(d2*d4+d1*d4+d1*d2))/2.
else
dterp=.5*(p2-p0)+(2.*p0-5.*p1+4.*p2-p3)*d2 &+(1.5*p3-4.5*p2+4.5*p1-1.5*p0)*d2**2
endif
return
end

***** DET ****************************
***** 3D determinant ****************************
***** input ****************************
* real*8
* a1(*) = 1st column of matrix
* a2(*) = 2nd column of matrix
* a3(*) = 3rd column of matrix

***** output ****************************
* real*8
* det = determinant

real*8 function det(a1,a2,a3)
implicit real*8 (a-h,o-z)
dimension a1(3),a2(3),a3(3)
det=a1(1)*(a2(2)*a3(3)-a3(2)*a2(3))-a2(1)*(a1(2)*a3(3) &-a1(3)*a2(2))+a3(1)*(a1(2)*a2(3)-a1(3)*a2(2))
return
end
***** ELECTRONR ******************************************************
***** electron density and gradient for Chapman layer model **********
***** input  ****************************************************************
 real*8 * r = distance from Earth centre
***** output  ******************************************************
 real*8
*** " electron density (electrons per cubic cm)
** diffr = derivative in vertical direction

**** subroutines required  ****************************
* CHAP, DCHAP and DET

***** common block  ****************************
* DAT (defined in HASDL)

subroutine electronr(r,en,diffr)
implicit real*8 (a-h,o-z)
dimension a(3),a(2),a(3),b(3)
common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foP2,hmP2,ymP2,foE,mode
if(r.lt.re) then
 eN=0.0 
endif 
return
end

***** CHAP  ************************************************************

chap(x)
real*8 function chap(C,x)

subroutine electron(r,en,diffr)
implicit real*8 (a-h,o-z)
dimension a(3),a(2),a(3),b(3)
common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foP2,hmP2,ymP2,foE,mode
if(r.lt.re) then
 eN=0.0 
endif 
return
end

***** chap  ****************************************************************

***** chapman layer function  ****************************
***** input  ****************************
 real*8
- C = 1 in P2 layer
  = 1/2 in E and F1 layers
- x = height above peak in units of scale height
***** output  ****************************
 real*8
- chap = value of Chapman function

real*8 function chap(C,x)
implicit real*8 (a-h,o-z)
xx=exp(-x)
if(abs(xx).gt.1.d30) then
chap=0.0
else
chap=exp(C*(1.d0-x.xx))
endif
return
end

***** DCHAP ****************************
***** derivative of chapman layer function ****************************
***** input ****************************
* real*8
* C = 1 in F2 layer
* = 1/2 in E and F1 layers
* x = height above peak in units of scale height
***** output ****************************
* real*8
* dchap = derivative of Chapman function

real*8 function dchap(C,x)
implicit real*8 (a-h,o-z)
xx=exp(-x)
if(abs(xx).gt.1.d30) then
dchap=0.0
else
dchap=C*(exp(-x)-1.d0)*exp(C*(1.d0-x.xx))
endif
return
end

***** MAGNETIC ****************************
***** magnetic fields and derivatives for an offset dipole model ****************************
***** input ****************************
* real*8
* x(*) = Cartesian coordinates (x(3) axis in the Earths axis of rotation and x(1) axis in the plane of zero longitude).
***** output ****************************
* real*8
* v(*) = magnetic field components in Cartesian system
* rxy(*,*') = derivatives of magnetic field components
* ha = magnitude of magnetic field
* dip = magnetic dip angle
* dcl = declination of magnetic field

subroutine magnetic(x,v,ryx,ha,dip,dcl)
  implicit real*8 (a-h,o-z)
  dimension x(3),v(3),ryx(3,3)
  common /mag/bhs,fr,mm,ym,zm,px,py,pz,ir
  xmt=x(1)-xm
  ymt=x(2)-ym
  zmt=x(3)-zm
  pm=xtm*px+ymt*py+zmt*pz
  rr=xmt*xmt+ymt*ymt+zmt*zmt
  r=sqrt(rr)
  rp=1.0/sqrt(rr)
  v(1)=(px-3.*pm*xmt/rr)*rp
  v(2)=(py-3.*pm*ymt/rr)*rp
  v(3)=(pz-3.*pm*zmt/rr)*rp
  ra=sqrt(x(1)*x(1)+x(2)*x(2)+x(3)*x(3))
  ha=sqrt(v(1)**2+v(2)**2+v(3)**2)
  vrv(1)=v(1)*v(1)+v(2)**2+v(3)**2)
  dip=45.*atan(v(2)*x(1)-v(1)*x(2))/(v(3)*ra-vr*x(3))/atan(1.)
  dcl=atan((v(2)**2-v(1)*x(2))/v(3)**2)/atan(1.)
  rxy(1,1)=(-3.*pm/rr+6.*pm*xmt/rr*rr-3.*px*xmt/rr)*rp
  return
end
&-3.*xmt*(px-3.*pm*xmt/rr)*rp/rr
ryx(1,2)=(-3.*py*xmt/rr+6.*pm*xmt*(rr*rr)*rp
&-3.*ymt*(px-3.*pm*xmt/rr)*rp/rr
ryx(1,3)=(-3.*pz*xmt/rr+6.*pm*zmt*xmt/(rr*rr))*rp
&-3.*zmt*(px-3.*pm*xmt/rr)*rp/rr
ryx(2,2)=(-3.*pm/rr+6.*pm*xmt/rr-3.*py*ymt/rr)*rp
&-3.*ymt*(py-3.*pm*xmt/rr)*rp/rr
ryx(2,3)=(-3.*pz*ymt/rr+6.*pm*xmt/(rr*rr))*rp
&-3.*zmt*(py-3.*pm*xmt/rr)*rp/rr
ryx(3,2)=(-3.*pm/rr+6.*pm*zmt*xmt/(rr*rr)-3.*pz*zmt/rr)*rp
&-3.*zmt*(px-3.*pm*xmt/rr)*rp/rr
ryx(3,3)=(-3.*pm/rr+6.*pm*xmt/rr-3.*py*ymt/rr)*rp
ryx(2,1)=ryx(l,2)
ryx(3,2)=ryx(2,3)
ryx(3,1)=ryx(1,3)
return
end

**** FUNC ****************************************************

**** right hand side of Haselgrove equations ********************

**** input ***********************************************

* real*6
* vec(*) = value of solution vector at which right hand required
* integer
* n = number of components in vectors vec and f

**** output ***********************************************

* real*6
* f(*) = right hand side of Haselgrove equations

**** subroutines required *************************************

* ELECTRON, MAGNETIC

**** common block ***************************************

* MAG (defined in HASEL)

**** notes *************************************************

* In the current code (HASEL), y has nine components. In order, these
* are:
* the Cartesian coordinates at the current point,
* the components of the wave vector,
* the group path,
* the Doppler shift,
* and the phase path.

subroutine func(vec,f,n)
imPLICIT REAL*6 (A-H,O-Z)
dIMENSION f(n),vec(n),v(3),ryx(3,3),dnx(4)
COMMON /MAG/FHS,FR,XM,YM,ZM,PM,PM,PM,PM,PM,PM
CALL ELECTRON(vec,EN,DXN)
RX=8.06D-5/(FR*FR)
RY=2.2D0
MAIL=0
VU=V(4)**2+V(5)**2+V(6)**2
IF(Abs(FHS).GT.1.0E-20)THEN
CALL MAGNETIC(VEK,VYX,HA,DIP,DL)
VU=(V(4)**2+V(5)**2+V(6)**2)/HA
RY=HA
RY2=RY**2
GA=GA/GV
AL=(-1.-RX-RY)
BEW.5*RY**(-1.-GA)
FAC=SQRT(ABS(BE+2.-AL*GA))
IF(IR.EQ.1)THEN
RP=-GA/BE+FAC
RJ=4.*FAC
ELSE
RP=(BE+FAC)/AL
RJ=4.*FAC
ENDIF
RP2=RP*RP
RK=-2.*(RX*VU*(RP*RY-L.1.)
R=RY*(1.-RY)**2+RP2+2.*((1.-RX-RY)**2*RP-R)
RM=2.*RX*RP*(RP*RY-L.1.)
UNCLASSIFIED 25
do i=1,3
   f(i)=vec(i+3)-2.06e-5*dx(i)/(fr*fr)
enddo
}

*** Runge-Kutta-Fehlberg scheme to solve dy/dt = f *************

***** input ****************************

* real*8
  * t = initial value of solution parameter (changed on exit)
  * y(*) = initial value of solution vector (changed on exit)
  * h = initial estimate of change in t
  * tol = maxim error that can be tolerated in this step
  * hmin= minimum h that can be tolerated
  * hmax= maximum h that can be tolerated
  * integer
    * n = number of components in vector y
***** output ******************************

* real*8
  * t = exit value of solution parameter
  * y(*) = estimate of solution vector at exit t
  * h = estimate of h for next step

***** subroutine required **************

* FUNC
  * This calculates the value of f corresponding to y. There are three
    * arguments. The first is y (real*8), the second is f (real*8) and
    * the third is the number of components in arrays y and f (integer).

26 UNCLASSIFIED
subroutine rkf(t,y,n,h,tol,hmin,hmax)
implicit real*8 (a-h.o-z)
dimension a1(11),a2(11),a3(11),a4(11),a5(11),a6(11),y(n),yt(11)
iend=0

iend=iend+1
call func(y,al,n)
alp=0.0
    do i=1,n
     alp=alp+al(i)**2
     yt(i)=y(i)+.25*al(i)*h
    enddo
alp=max(sqrt(abs(alp)),1.0d-32)
call func(yt,a2,n)
do i=1,n
    yt(i)=y(i)+(.09375*al(i)+.28125*a2(i))*h
endo
call func(yt,a3,n)
do i=1,n
    yt(i)=y(i)+(-.87938097406d0*al(i)-3.2771961766d0+a2(i)
&+3.32089212563d0*a3(i))*h
endo
call func(yt,a4,n)
do i=1,n
    yt(i)=y(i)+(.20589688616d0*al(i)+1.7348927875d0*a2(i)
&-.2592720955d0*a4(i))*h
endo
call func(yt,a5,n)
do i=1,n
    yt(i)=y(i)+(-.296296297d0+a2(i)-1.38167641326d0*a3(i)
&+.45297270955d0*a4(i)+.275*a5(i))*h
endo
call func(yt,a6,n)
erx=0.0
    do i=1,3
     erz=-.2777777777777778d-2*al(i)-.299415204678363d-1*a3(i)
&-.291998936735779d-1*a4(i)+.2D-1*a5(i)+.3636363636364D-1*a6(i)
erx=erx+erz**2
endo
erz=sqrt(erz)
hold=h
    if(erx.gt.0.001*tol) then
     s=.88*sqrt(sqrt(abs(tol/erx)))
h=s*h
    else
     h=2.*h
    endif
if(h.gt.hmin/alp) h=hmin/alp
if(h.lt.hmax/alp) h=hmax/alp
if(erx.gt.tol.and.iend.lt.8) then
    go to 99
else
   do i=1,n
     y(i)=y(i)+(.11851851852*al(i)+.51898635478d0*a3(i)
&+.50613149034d0*a4(i)+.18*a5(i)+.03636363636*a6(i))*hold
endo
t=t+hold
return
endif
end
***** ELDEN *****************************************************************************

***** general electron density *****************************************************************************

***** input **************************************************************************

* real*8
* 
* t = time from start (sec)
* along = longitude (deg) of sample point
* elat = latitude (deg) of sample point
* r = height above Earth centre (Kms)
* 
* Note that hmE, hmF1 and hmF2 should be set to suitable values to
* allow the labelling of rays

***** output **************************************************************************

* real*6
* 
* elden = electron density (electrons per cubic cm)

***** common block (optional) **************************************************************************

DAT (defined in HASEL)

real*8 function elden(t,along,elat,r)
implicit real*8 (a-h,o-z)
common /dat/foE,hmE,ymE,foF1,hmF1,ymF1,foF2,hmF2,ymF2,re,model
foF2=12.
hmF2=350.
ymF2=110.
vel=0.04
x=(r-re-hmF2-vel*t)/ymF2
elden=(foF2**2/80.6d-6)*chap(1.d0,1.4142d0*x)+0.*along*elat
return
end

*****************************************************************************

*****************************************************************************
### DISTRIBUTION

<table>
<thead>
<tr>
<th>Defence Science and Technology Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chief Defence Scientist</td>
</tr>
<tr>
<td>Central Office Executive</td>
</tr>
<tr>
<td>Counsellor, Defence Science, London</td>
</tr>
<tr>
<td>Counsellor, Defence Science, Washington</td>
</tr>
<tr>
<td>Scientific Adviser, Defence Central</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Navy Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navy Scientific Adviser</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Air Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air Force Scientific Adviser</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Army Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scientific Adviser, Army</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defence Intelligence Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scientific Adviser</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Surveillance Research Laboratory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Director, Surveillance Research Laboratory</td>
</tr>
<tr>
<td>Chief High Frequency Radar Division</td>
</tr>
<tr>
<td>Research Leader, Jindalee Operational Radar Network</td>
</tr>
<tr>
<td>Head, Radar Processing and Tracking</td>
</tr>
<tr>
<td>Head, Ionospheric Effects</td>
</tr>
<tr>
<td>Head, Radar Technology and Systems</td>
</tr>
<tr>
<td>Head, HF Radar Engineering</td>
</tr>
<tr>
<td>Graphics and Documentation, High Frequency Division</td>
</tr>
<tr>
<td>Dr J.A Bennett, Ionospheric Effects</td>
</tr>
<tr>
<td>Chief Microwave Radar Division</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Research Leader Microwave Radar</td>
</tr>
<tr>
<td>Research Leader Radar Projects</td>
</tr>
<tr>
<td>Author</td>
</tr>
</tbody>
</table>

**Electronics Research Laboratory**

<table>
<thead>
<tr>
<th>Chief Communications Division</th>
<th>Cnt Sht Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research Leader Military Communications</td>
<td>1</td>
</tr>
<tr>
<td>Research Leader Communications Countermeasures</td>
<td>1</td>
</tr>
<tr>
<td>Dr K.J.W Lynn, Radiowave Propagation</td>
<td>1</td>
</tr>
<tr>
<td>Dr R.H Clarke, Radiowave Propagation</td>
<td>1</td>
</tr>
</tbody>
</table>

**Libraries and Information Services**

| Australian Government Publishing Service         | 1            |
| Defence Central Library - Technical Reports Centre| 1            |
| Manager, Document Exchange Centre (For Retention) | 1            |
| National Technical Information Service, United States | 2          |
| Defence Research Information Centre, United Kingdom | 2          |
| Director Scientific Information Services, Canada  | 1            |
| Ministry of Defence, New Zealand                  | 1            |
| National Library of Australia                     | 1            |
| Defence Science and Technology Organisation Salisbury, Research Library | 2 |
| British Library Document Supply Centre            | 1            |
| Library Defence Signals Directorate, Melbourne    | 1            |

**Spores**

| Defence Science and Technology Organisation Salisbury, Research Library | 6 |

---

No additional text provided for the page.
A general purpose ionospheric ray tracing procedure is developed. The procedure is based on a numerical solution to the Haselgrove ray tracing equations and includes calculations of ionospherically induced Doppler shift. The procedure can handle a wide variety of ionospheric descriptions and includes the effect of the Earth's magnetic field.
<table>
<thead>
<tr>
<th>16. Abstract (CONT.)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>17. Imprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surveillance Research Laboratory</td>
</tr>
<tr>
<td>PO Box 1500</td>
</tr>
<tr>
<td>SALISBURY SA 5108</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SRL-0131-TR</td>
<td>440/828309</td>
<td>TECHNICAL REPORT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>21. Computer Programs Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>22. Establishment File Reference(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>23. Additional information (if required)</th>
</tr>
</thead>
</table>