An Altitude-Error Display for Height-Finder Radar

A. E. Barrios
ADMINISTRATIVE INFORMATION

The work described in this report was performed by the Tropospheric Branch (Code 543), Naval Oceans Systems Center, for the Office of Naval Technology, Office of Chief of Naval Research.

Released by
H.V. Hitney, Head
Tropospheric Branch

Under authority of
J.H. Richter, Head
Ocean and Atmospheric Sciences Division
PAGES ___ ARE MISSING IN ORIGINAL DOCUMENT
An Attitude-Error Display for Height-Finder Radar

Strong ducting conditions affect height-finder radars in giving accurate target positions. An attitude-error display has been developed to show the amount of error present.
SUMMARY

An altitude-error display for height-finder radar has been developed on the HP9000 - 500-series computer. The display looks like an ordinary ray-trace display except the color of the rays are dependent on the height difference, as compared to a standard atmosphere for the same elevation angle and range.

CONCLUSIONS

Since this effort only considered alternate displays of well-established ray-tracing theory, no attempt was made to validate the accuracy of any of the resulting displays.

RECOMMENDATION

The altitude-error displays reported here should be incorporated into the coverage diagram of the Integrated Refractive Effects Prediction System (IREPS).
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>BACKGROUND</td>
<td>1</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>5</td>
</tr>
<tr>
<td>SAMPLE DISPLAYS</td>
<td>7</td>
</tr>
<tr>
<td>CONCLUSIONS</td>
<td>25</td>
</tr>
<tr>
<td>RECOMMENDATION</td>
<td>25</td>
</tr>
<tr>
<td>APPENDIX A: COMPUTER PROGRAM LISTING</td>
<td>A-1</td>
</tr>
</tbody>
</table>
INTRODUCTION

Strong ducting conditions occur over many ocean areas. These conditions affect height-finder radars (such as the SPS-48) in giving accurate target positions. Most height-finder radars calculate altitude based on a standard atmosphere. When ducting conditions are present, large errors can occur between the calculated and the true target height, depending on the transmitter height and the target range. In many cases, these errors are greater than 50 percent. A height-finder radar altitude-error display has been developed to show the amount of error present. The display looks like an ordinary ray-trace display, except the color of the rays are dependent on the height difference, as compared to a standard atmosphere for the same elevation angle and range.

BACKGROUND

The atmosphere is considered to consist of vertical layers, each with a certain gradient of the index of refraction. The layers are assumed to be horizontally homogeneous. Each profile consists of at least two layers. Each layer is associated with a height, \( H(i) \), and modified refractivity or M-unit value, \( M(i) \). Modified refractivity, \( M \), is related to the index of refraction, \( n \), by

\[
M = \left( n - 1 + \frac{z}{a} \right) \cdot 10^6
\]

where \( a \) is the mean earth's radius, and \( z \) is the height above the earth's surface.

A ray path under standard atmospheric conditions will bend downward at a rate less than the curvature of the earth, so to an observer stationed on the earth's surface, the ray will appear to bend upward. A trapped ray is one that, because of a trapping layer, will bend downward at a rate exceeding the curvature of the earth. A trapping layer can be very easily identified by a negative M-gradient. Examples of profiles that represent a surface-based duct and an elevated duct are shown in figure 1. Two other types of refraction that describe the relation between modified refractivity and height are subrefraction and superrefraction. A subrefractive profile will cause rays to be bent less than the normal or standard, while a superrefractive profile bends rays at a rate exceeding the normal but not enough to cause trapping.

The gradient, \( \text{Dmdh} \), is the change in M-units with respect to the change in height and is defined as

\[
\text{Dmdh}(i) = \left( \frac{M(i + 1) - M(i)}{H(i + 1) - H(i)} \right) \cdot 10^{-3}
\]

Table 1 shows the relation between the M-gradient and the different types of refraction. Figure 2 gives a clear picture of the relative bending among the different types.

An individual ray trace begins with an elevation angle specified at the source height, \( H_t \), and consists of a series of calculations to determine \( H_t \); a specified range or range at a specified height. All calculations can be described by one of the following six cases. The variables are defined as
Figure 1. Examples of elevated and surface-based ducts from elevated layers.

Table 1. The relation between M-gradient and the different types of refraction.

<table>
<thead>
<tr>
<th>Types of Refraction</th>
<th>M-Gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trapping</td>
<td>$\leq 0$ M/km</td>
</tr>
<tr>
<td></td>
<td>$\leq 0$ M/kft</td>
</tr>
<tr>
<td>Superrefractive</td>
<td>0 to 79 M/km</td>
</tr>
<tr>
<td></td>
<td>0 to 24 M/kft</td>
</tr>
<tr>
<td>Standard</td>
<td>79 to 157 M/km</td>
</tr>
<tr>
<td></td>
<td>24 to 48 M/kft</td>
</tr>
<tr>
<td>Subrefractive</td>
<td>$&gt;157$ M/km</td>
</tr>
<tr>
<td></td>
<td>48 M/kft</td>
</tr>
</tbody>
</table>
Figure 2. Relative bending for the four types of refraction.

\[
\begin{align*}
\alpha &= \text{elevation angle at beginning of calculation (rad)} \\
\alpha' &= \text{elevation angle at end of calculation (rad)} \\
h &= \text{height at beginning of calculation (m)} \\
h' &= \text{height at end of calculation (m)} \\
r &= \text{range at beginning of calculation (km)} \\
r' &= \text{range at end of calculation (km)}
\end{align*}
\]

Case 1: \( \alpha > 0, h' \text{ known (figure 3)} \)

\[
\begin{align*}
\alpha' &= \sqrt{\alpha^2 + 2 \cdot 10^{-3} \text{Dmdh}(i)} (h' - h), \text{ if } H(i) \leq h \leq h' \leq H(i + 1) \\
r' &= r + \frac{\alpha' - \alpha}{\text{Dmdh}(i)}
\end{align*}
\]

If the source is in a duct, trapping may occur for some initial elevation angles. In this case, the radicand for \( \alpha' \) becomes negative, and \( h' \) reaches a maximum height. For this example,

\[
\begin{align*}
\alpha' &= 0, \\
r' &= r - \frac{\alpha}{\text{Dmdh}(i)}, \text{ and} \\
\frac{\alpha^2}{2 \cdot 10^{-3} \text{Dmdh}(i)}
\end{align*}
\]
Case 2: \( \alpha > 0, r' \) known (figure 3 applies)

\[
\alpha' = \alpha + \text{Dmdh}(i) (r' - r), \quad \text{if } h < H(i + 1),
\]
\[
h' = h + \frac{\alpha^2 - \alpha'^2}{2 \cdot 10^{-3} \text{Dmdh}(i)}
\]

Case 3: \( \alpha < 0, h' \) known (figure 4)

\[
\alpha' = -\sqrt{\alpha^2 + 2 \cdot 10^{-3} \text{Dmdh}(i) (h' - h)}, \quad \text{if } H(i + 1) > h > h' > H(i),
\]
\[
r' = r + \frac{\alpha' - \alpha}{\text{Dmdh}(i)}
\]

A ray that is initially downgoing may eventually become upgoing as \( \alpha \) increases. The ray reaches a minimum height, and in this case, the radicand for \( \alpha' \) becomes negative. Therefore,

\[
\alpha' = 0
\]
\[
r' = r - \frac{\alpha}{\text{Dmdh}(i)}
\]
\[
h' = h - \frac{\alpha^2}{2 \cdot 10^{-3} \text{Dmdh}(i)}
\]

Case 4: \( \alpha < 0, r' \) known (figure 4 applies)

\[
\alpha' = \alpha + \text{Dmdh}(i) (r' - r), \quad \text{if } h' > H(i),
\]
Figure 4. Ray trace variables for \( \alpha < 0 \).

\[
h' = h + \frac{\alpha'^2 - \alpha^2}{2 \cdot 10^{-3} D_{mdh(i)}}
\]

Case 5: \( \alpha = 0 \)

If \( D_{mdh(i)} > 0 \), use Case 1 or Case 2 as appropriate.

If \( D_{mdh(i)} < 0 \), use Case 3 or Case 4 as appropriate.

Case 6: A ray launched at the inflection point (figure 5) with \( \alpha = 0 \) will stay at that height.

**PROGRAM**

The program plots height vs. range for each ray. Reflected rays are not traced. Each ray is drawn in several colors, depending on the error scale chosen by the user.

Initially, there are four options available to the user: (1) delete data file, (2) add data file, (3) edit data file, and (4) run the program for a display. If option 4 is picked, a list of available data files will appear on the screen. After a data file has been chosen, there are several parameters the user must enter into the program:

1. Maximum height for the display in feet or meters.
2. Maximum range for the display in nautical miles or kilometers.
3. Antenna height in feet or meters.
4. Lower elevation angle. Lower angular limit in milliradians.
6. Number of rays. This is the number of rays to be traced.
7. Error scale. The height-error scale can be an absolute- or percent-error scale, indicated by entering "A" or "P". The default is "A".

8. Error increment. This value determines the scale by which the colors are defined. If using an absolute-error scale, the error increment is entered in feet or meters.

Normal output for the display is the screen, but a hardcopy print can be selected.

Once the data has been read from the selected file, the program performs a linear extrapolation to find the M-unit value at the surface and at the maximum plot height, if necessary. These values are added to the height and M-unit arrays and the gradients (Dmdh) are then calculated.* The initial launch angles, calculated from the lower elevation angle, the upper elevation angle, and the number of rays, are also put into an array.

The ray trace is performed by range increments (1/50th of the maximum plot range) i.e., beginning with the initial launch angle and antenna height, a new angle and height are calculated to a specified range, plotted, the range is incremented, the next angle and height are calculated, plotted, and so forth until the maximum height or maximum range has been reached. As each new angle is calculated, the program will branch to the appropriate case (as discussed in the previous section). With each new height calculated the program will trace a ray to the same range using the same initial launch angle for a standard atmosphere. The difference between the two heights gives the height error, and the ray will be drawn in the appropriate color.

If the antenna height is at a breakpoint where Dmdh is positive beneath the point and negative above, the program will calculate through an infinite number of maximums and minimums for sufficiently small initial launch angles, resulting in a complete halt of the program. Also, because of the dot-matrix screen, some height increments may be too small to be noticed in plotting, and what the program calculates as maximums and minimums may look like a straight line across the screen. Therefore, a restriction on the initial launch angles is made. Depending on the angular limits used, the program will not allow traces of

*To avoid Dmdh approaching zero, a value of 10e-6 was assigned to Dmdh if it fell in the region -10e-6 < Dmdh < 10e-6.
rays launched within a certain angle above and below the horizontal. A single horizontal ray trajectory is drawn for launch angles within this range and the next greater angle in the launch angle array is selected upon completion of this ray path.

SAMPLE DISPLAYS

Figure 6 shows a height-finder radar altitude-error display for an elevated duct. Figure 7 shows two ray-trace displays, one for a standard atmosphere (black rays) and one for the same elevated duct (red rays). The height difference calculated for a specific range between the two sets of rays is shown. The middle ray, launched at zero radians, is refracted the most and therefore produces the most error. This can also be seen in figure 6 where the red area (for the shortest range) is associated with rays launched near zero radians.

The program does not distinguish between height difference above or below the standard. For instance, at a height of 10,000 feet and a range of 100 miles, one may get an error of 1000 feet, which could be 1000 feet above the standard or 1000 feet below the standard. As can be seen in figure 8, the ray at -10 mrad bends above the standard then crosses over to bend below (rays in standard atmosphere are in black). Again, the middle ray at zero radians gives the highest error. The error display is shown in figure 9. The error display for the same profile using 200 rays is shown in figure 10. In displays such as this, one can see rays increasing in height error (as well as height and range), then starting to decrease in error. Figure 8 shows why this is so.

Another example of this type of display is shown in figure 11 with the corresponding ray trace shown in figure 12. In figure 11, the first ray is reflected and therefore, not traced. From the third ray on, the rays bend toward the standard to produce a decrease in height error. The “full” error display (using 200 rays) is shown in figure 13.

Although the error displays are meant to be used for surface height-finder radars, figures 6 and 10 are displays for airborne height-finder radars and are shown only for demonstration purposes.
Figure 6. Error display for elevated duct - 200 rays.

ANTENNA HEIGHT (ft): 30000.0
LOWER ELEVATION ANGLE (in mrad): -10.0
UPPER ELEVATION ANGLE (in mrad): 10.0
NUMBER OF RAYS: 200
Figure 7. Ray-trace display for elevated duct of figure 6 (red) and standard atmosphere (black).
Figure 8. Ray-trace display for elevated duct (red) and standard atmosphere (black).
Figure 9. Error display for elevated duct of figure 8 — five rays.
Figure 10. Error display for elevated duct of figure 8 - 200 rays.
Figure 11. Error display for surface-based duct – five rays.
<table>
<thead>
<tr>
<th>HEIGHT (ft)</th>
<th>M-UNITS</th>
<th>HEIGHT (ft)</th>
<th>M-UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.0</td>
<td>355.6</td>
<td>15639.0</td>
<td>913.4</td>
</tr>
<tr>
<td>931.0</td>
<td>385.6</td>
<td>19748.0</td>
<td>1088.9</td>
</tr>
<tr>
<td>1506.0</td>
<td>331.1</td>
<td>32300.0</td>
<td>1642.5</td>
</tr>
<tr>
<td>2034.0</td>
<td>349.3</td>
<td>38200.0</td>
<td>1906.3</td>
</tr>
<tr>
<td>2681.0</td>
<td>372.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3575.0</td>
<td>406.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4417.0</td>
<td>440.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4857.0</td>
<td>475.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5606.0</td>
<td>492.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6514.0</td>
<td>525.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6935.0</td>
<td>543.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7412.0</td>
<td>563.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8536.0</td>
<td>610.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9625.0</td>
<td>659.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13968.0</td>
<td>852.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ANTENNA HEIGHT (ft): 100.0
LOWER ELEVATION ANGLE (in mrad): -5.0
UPPER ELEVATION ANGLE (in mrad): 40.0
NUMBER OF RAYS: 5

Figure 12. Ray-trace display for surface-based duct of figure 11 (red) and standard atmosphere (black).
Figure 13. Error display for surface-based duct of figure 11 - 200 rays.
CONCLUSIONS

A height-finder radar altitude-error display has been developed on the HP9000 — 500-series computer based on traditional ray-tracing concepts. This display shows altitude error compared to a standard atmosphere using a color scale superimposed on a traditional altitude-versus-range ray trace for arbitrary, piecewise, linear refractivity-versus-altitude profiles. Since this effort only considered alternate displays of well-established ray-tracing theory, no attempt was made to validate the accuracy of any of the resulting displays.

RECOMMENDATION

The altitude-error displays reported here should be incorporated into the coverage diagram of the Integrated Refractive Effects Prediction System (IREPS).
********** Variables used in program **********

dadh: 35 element array containing the gradients of the profile.

erropt: Character string indicating if absolute ('a' or 'A') or percent ('p' or 'P') for color scale was chosen.

aber: Absolute error increment in meters.

per: Percent error increment.

fildes: File descriptor for starbase graphics routine.

index: Integer indicating color to be used for plotting (1-7).

mvfl: Logical flag indicating when to move to the next height-range point for plotting onto hardcopy when changing pen colors.

ht: Antenna height in meters.

heax: Maximum height for plot display in meters.

max: Maximum range for plot display in kilometers.

orh: 33 element array containing heights of the original profile.

orm: 33 element array containing m-units of the original profile.

h: 35 element array containing heights of the new profile.

im: Counter indicating what color is currently being used for hardcopy plotting.

plot: Logical flag indicating if plotting onto hardcopy, (T-plot, F-don't plot).

name: Character string containing the data filename to be read.

nlvl: Number of new levels after extrapolation to the surface and maximum height.

lv1: Number of levels in the original profile.

m: 33 element array containing m-units of the new profile.

f7: Character string indicating if 'AUTODUMP' is set to on or off.

angle: Array containing initial elevation angles (300 maximum).

hun: Character string indicating what height units the profile was stored in.

reun: Character string indicating if refractivity values were stored in m-units or n-units.

frstime: Logical flag indicating first height-range point calculated for new ray.

sor: Starting range for hardcopy plotting.

soh: Starting height for hardcopy plotting.

frst: Logical flag indicating first storage of height-range point for interpolation between color transitions.

ipro_print: Integer indicating type of profile printout for hardcopy; 1-graph, 2-number.

twice: Logical flag indicating if program has been run at least once.

common/errvar/dadh(35), erropt, aber, per, fildes, index, mvfl, +ht, heax, max, rinc, orh(33), orm(33), h(35), im, plot, name, +nlvl, lv1, m, f7, angle(300), hun, reun, frstime, sor, soh, frst, +ipro_print, twice
57: integer*4 fildes,index
58: real m(35)
59: character*3 f7
60: character*1 erropt,hun,reun
61: character*14 name
62: logical mvfl,plot,frstime,frst,logical,twice
PROGRAM HFERR

**Purpose:** This is the main program that will begin all procedures for ray tracing.

**Glossary:**

- **opst**: Character string indicating if the 'option' or 'backup' key was hit.
- **outst**: Character string variable.
- **optn**: Real value of option number chosen by user.
- **ioptn**: Integer value of option number chosen by user.
- **flag**: Logical flag indicating if the user has gone through the edit routine.
- **profile**: Array containing all data filenames.
- **ifil**: Number of data filenames in profile.
- **rpick**: Real value of number of environmental data file chosen.
- **ipick**: Integer value of number of environmental data file chosen.
- **rlowea**: Lower elevation angle in mrad.
- **upea**: Upper elevation angle in mrad.
- **rplot**: Number of rays to be plotted.
- **irplot**: Integer value of number of rays to be plotted.

**Routines:**

```fortran
program hferr
include '/usr/include/starbase.f1.h'
include '/usr/include/starbase.f2.h'
#include 'errvar'

character*1 fun
c
character*8 opst,kgt
c
character*14 profile(50)

c
character*6 dummy,outst
c
integer*4 status
c
logical flag,duct,us_flag
c
ir=5
c
iw=6
c
call kyinit(ir,iw,15)
c
twice=.false.
c
us_flag=.false.
c
ipro_print=0
c

clear alphanumerics screen and write options list. 

10 call kysetos(0,0,-1)
c
write(iw,('(1 -- Delete an environmental data file')'))
c
write(iw,('(2 -- Add an environmental data file')'))
c
write(iw,('(3 -- Edit an environmental data file')'))
c
write(iw,('(4 -- Altitude Error Display')'))
c
if(twice) write(iw,('(5 -- Display for same profile')'))
c
write(iw,')
c
call kyenter("option number (or 'end') ",0,1..5..-1..optn,opst)
c
if(opst.eq."backup") or (opst.eq."option") goto 10
c
write(iw,')
```
ioptn=mint(optn)
if(ioptn.eq.4) twice=.false.
if(optn.eq.-1)opst='end'
if((opst.eq.'end').or.(opst.eq.'END')) goto 20
else
   if(.not.twice) call renew
end if
flag=.false.
if((ioptn.eq.1).or.(ioptn.eq.2).or.(ioptn.eq.3)) then
   call edit(ioptn,opst,flag,iw)
   if((opst.eq.'option').or.(opst.eq.'backup')) goto 10
end if
if(flag) goto 10
if(ioptn.eq.5) then
   use_flag=.true.
call putin(r1,ru,irplot,filvar,fun)
goto 15
end if
************* List existing environmental data files. *************
call lsfiles(.'prof',profile,fil1)
if(fil1.eq.0) then
   write(iw,'(a6)')
call kyenter('No data files exist. Press RETURN to create new file.'
               +'-1',opst,fil1)
   if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
   ioptn=2
   goto 30
end if
************* Begin writing prompt strings for user input. *************
call kyenter('number of environmental data file',0,1,real(fil1),
           +0.,rpick,opst)
if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
ipick=aint(rpick)
name=profile(ipick)
call kyenter('maximum height for display',1.1..50000.,0.,mmax,
             +opst)
if((opst.eq.'backup') goto 40
if((opst.eq.'option') goto 10
write(iw,'*')
call kyenter('maximum range for display',2,20.,1000.,0.,remax,
             +opst)
if((opst.eq.'backup') goto 50
if((opst.eq.'option') goto 10
write(iw,'*')
call kyenter('antenna height',1,1.,50000.,0.,ht,opst)
if((opst.eq.'backup') goto 60
if((opst.eq.'option') goto 10
write(iw,'*')
call kyenter('lower elevation angle in mrad',0.,-100.,100.,101.,rl,
**Call INITIAL to set up arrays and variables used in the ray tracing subroutines.**

```
113: +opst)
114: if(opst.eq.'backup') goto 70
115: if(opst.eq.'option') goto 10
116: write(iw,*)
117: 90 call kyenter("upper elevation angle in mrad",0.,-100.,100.,ru,
118: +opst)
119: if(opst.eq.'backup') goto 80
120: if(opst.eq.'option') goto 10
121: write(iw,*)
122: 100 call kyenter("number of rays to be plotted",0.,300.,0.,rplot,
123: +opst)
124: if(opst.eq.'backup') goto 90
125: if(opst.eq.'option') goto 10
126: irplot=nint(rplot)
127: write(iw,*)
128: 110 call kyenter("A or P for absolute or percent error",0.,'A,a,P,p',
129: +a',erropt,opst)
130: if(opst.eq.'backup') goto 100
131: if(opst.eq.'option') goto 10
132: write(iw,*)
133: if((erropt.eq.'a').or.(erropt.eq.'A')) then
134: 120 call kyenter("number for error increment",1.,5000.,0.,aber,
135: +opst)
136: if(opst.eq.'backup') goto 110
137: if(opst.eq.'option') goto 10
138: write(iw,*)
139: else
140: 130 call kyenter("percent error increment",0.,20.,5.,per,opst)
141: if(opst.eq.'backup') goto 110
142: if(opst.eq.'option') goto 10
143: write(iw,*)
144: end if
145: call print(iw,profile,run,run)
146: 15 call kyread(7,f7)
147:
148: c ****** Call INITIAL to set up arrays and variables used in
149: c the ray tracing subroutines. ****************************
150: c
151: call initial(rl,ru,irplot,duct,forbid)
152: c ****** Call GRAPH to initialize graphics screen and draw axes
153: c for plotting.  ******************************************
154: c
155: call kyread(6,kgt)
156: if((ipro_print.eq.2).or.(ipro_print.eq.1)) kgt='on'
157: if(kgt.eq.'off') then
158: plot=.false.
159: end if
160: else if(kgt.eq.'on') then
161: plot=.true.
162: if((ipro_print.ne.1).and.(ipro_print.ne.2)) then
163: call kyenter("profile printout: 1 - graph ; 2 - numbers",
164: + 0,1.,2.,1.,pro_print,opst)
165: pro_print=nint(pro_print)
166: end if
167: end if
168: 25 call graph(rl,ru,irplot)
```
169: ** Start of main calculations. **
170: call actapp(irplot, duct, forbid)
171: write(iw,*)
172: ** Begin dump. **
173: status=gcloset(fildes)
174: twice=.true.
175: if(plot) goto 10
176: call kysetent("Press 'AUTO DUMP-ON' and 'RETURN' for hardcopy",
177: -1,'*','*',outst,opst)
178: if(opst.eq.'option') goto 10
179: call kyread(6,kgt)
180: if(kgt.eq.'on') then
181: call kyenter("profile printout: 1 - graph; 2 - numbers",
182: + 0,1,2,1,.pro_print,opst)
183: if(opst.eq.'option') goto 10
184: ipro_print=nint(pro_print)
185: plot=.true.
186: goto 25
187: else
188: goto 10
189: end if
190: continue
191: if(use_flag) call system('rm usin')
192: call kyterm
193: end
SUBROUTINE ACTAPP

Purpose: This begins the main loop of the program. It checks for the transmitter height level and begins the ray-tracing subroutines.

Glossary:
- duct Logical flag indicating if the transmitter height is at a break point.
- irays Number of rays to be used in plotting.

subroutine actapp(irays, duct, forbid)

*INCLUDE 'errvar'

subroutine stand(fixang, rp, hp)

*INCLUDE 'errvar'

rf=rp
rref=0.
**Begin trace for down-going rays.**

```fortran
  hbef=ht
  alpha=fixang
  ********** Begin trace for down-going rays. **********
  if(alpha.lt.0.) then
    alphap=alpha+1.18e-4*rf
    if(alphap.ge.0.) then
      alphap=0.
      range=rbef-alpha/1.18e-4
      hs=hbef-alpha**2/2.36e-7
      alpha=alphap
      hbef=hs
      rbef=range
      goto 10
    end if
    hs=hbef+(alphap**2-alpha**2)/2.36e-7
  end if

  If the height is negative, then ray is reflected. ********
  if(hs.lt.0.) then
    alphap=-SQRT(alpha**2+2.36e-7*(-hbef))
    range=rbef+(alphap-alpha)/1.18e-4
    alphap=alphap
    alphap=alphap
    hbef=0.
    goto 10
  end if
  goto 20
end if

**Begin trace for up-going rays.**

```fortran
  10 if(alpha.ge.0.) then
    alphap=alpha+1.18e-4*(rf-rbef)
    hs=hbef+(alphap**2-alpha**2)/2.36e-7
  end if
  20 continue
  hdif=ABS(hp-hs)

  **Calling routine to determine color according to error increment scale.**
  rng=0.
  hyt=0.
  call interpol(hdif,rp,hp,rng,hyt,hs)
  first=.false.
  if(plot) then
    call stepfixplt(hdif,hs,rp,hp,rng,hyt)
  else
    call stepfixscr(hdif,hs,rp,hp,rng,hyt)
  end if
  return
end
```

---

This code snippet is a subroutine for handling ray tracing, where the main focus is on the handling of down-going and up-going rays. It includes logic for determining the new position of a ray based on its angle of incidence, the range, and other variables related to the ray's behavior. The code also includes a call to a routine for determining color according to error increment scale.
SUBROUTINE DOWN

Purpose: This routine performs a ray trace for down-going rays.

Glossary:
- `rbef`: Range before calculation.
- `rp`: Range after calculation.
- `rinc`: Range increment.
- `hbf`: Height before calculation.
- `hp`: Height after calculation.
- `ij`: Height level counter.
- `alpha`: Angle before calculation.
- `alphap`: Angle after calculation.

**BEGIN LOOP TO TRACE FROM TRANSmitter HEIGHT TO THE FIRST LEVEL OR UN1L A MINIMUM HAS BEEN REACHED.**

```fortran
do while((ij.ge.1).and.(alpha.le.0.))
  rp=rbef+rinc
  if(rp.ge.rmax) goto 10
  alphap=alpha+dmdh(ij)*(rp-rbef)
end if
```

If the new angle calculated is positive then a minimum has been reached.

```fortran
if(alphap.ge.0.) then
  alphap=0.
  rp=rbef-alpha/dmdh(ij)
  hp=hbf-alpha*2/2.E-3/dmdh(ij)
end if
```

If RAD is negative then a minimum has been reached.

```fortran
if(rad.le.0.) then
  alphap=0.
  rp=rbef-alpha/dmdh(ij)
  hp=hbf-alpha*2/2.E-3/dmdh(ij)
else
  alphap=-SQRT(rad)
  rp=rbef+(alphap-alpha)/dmdh(ij)
end if
```

**END LOOP**

```fortran
return
end subroutine down
```
end if

** Once a specific height and range have been calculated

subroutine STAND is called to calculate the height at the same range

RP for a standard atmosphere.

if(rp.ge.rmax) goto 10

call stand(fixang,rp,hp)

rbef=rp

hbef=hp

alpha=alphap

if(alpha.eq.0.) goto 10

end do

if(ij.eq.0.) goto 30

iftrp.ge.rmax) then

rpmrsax

alphapwalpha&dmdh(ij)*(rp-rbef)

hpahbef.(alphap**R-alpha&e2)/a*.-3/dmdh(1j)

call stand(fixang,rp,hp)

end if

D0:

30 return

end
SUBROUTINE EDIT

This subroutine performs all the editing operations of the program.

subroutine edit(ioptn,opst,flag,iw)

character*8 opst,outst
character*14 filena(50),dumname
character*80 dum
logical flag

************* Specified files are deleted or put into the vi editor for editing. *************

if((ioptn.eq,1).or,(ioptn.eq.3)) then
    call lsfiles('.prof',filena,ifil)
    if(ifil.eq.0) then
        write(iw,'(alp))char(7)
        call kystent('*** No data files exist. Press "RETURN" to create new file. ***',-1,'.',outst,opst)
    endif
    goto 20
endif

if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
ioptn=2
endif

if(ioptn eq 2) then
    call system('cp .dprof dprof')
    call system('vi dprof')
    open(3,FILE='dprof')
    read(3,'(3(a80),a80)') (dum,i=1,4)
    read(3,':a14') filena(ifil)
    close(3)
    call system('mv .dprof//'filena(ifil)'/char(0))
    else if(ioptn eq 1) then
        call system('mv .prof//'filena(ifil)'/char(0))
    else
        call system('vi .prof//'filena(ifil)'/char(0))
        open(3,FILE='prof//'filena(ifil))
        read(3,'(3(a80),a80)')(dum,i=1,4)
        read(3,':a14')dumname
        if(dumname.ne.filena(ifil)) call system('mv .prof//'filena
        + (ifil)'/char(0))
        close(3)
    end if
endif

return
*********************** SUBROUTINE INITIAL **********************

Purpose: This subroutine calculates all the necessary constants and arrays for use in actual ray trace calculations.

Glossary:
- rplot: Real value of the number of rays to be plotted.
- irplot: Integer value of the number of rays to be plotted.
- forbid: Angular region of unallowable ray traces.
- duct: Flag indicating if the transmitter height is at a breakpoint.
- check: Flag indicating if hmax is equal to any of the heights in the profile.
- rl: Lower elevation angle in mrad.
- rfirst: Lower elevation angle in rad.
- ru: Upper elevation angle in mrad.
- rlast: Upper elevation angle in rad.
- ainc: Incremental angle value for alpha array.
- alpha: Array containing all initial launch angles.
- rinc: Range increment.
- rmetperdot: Meters per dot on screen.

Subroutine initial(rl,ru,irplot,duct,forbid)

*INCLUDE 'errvar'
logical duct,check

Convert lower and upper angles to radians and store in an array in incremental values corresponding to the number of rays to be plotted.

rfirst=rl*1.e-3
rlast=ru*1.e-3
ainc=(rlast-rfirst)/(irplot-1)

angle(1)=rfirst
do i=2,irplot
angle(i)=angle(i-1)+ainc
end do

gle(IRplot)=rlast
rinc=rmax/50.
sor=.025*rmax

Convert height and index array to meters and m-units if necessary.

if(hun.eq.'F') then
h(i)=orh(i)/3.280839
else
h(i)=orh(i)
end if
if(hun.eq.'N') then
s(i)=orm(i)/6.371
else
s(i)=orm(i)

*************************************************
57:       end if
58:   end do
59:   nlvl=nlvl+1
60: if(h(1).ne.0.) then
61:     nlvl=nlvl+1
62:     surf=h(1)/(h(2)-h(1))*(m(1)-m(2))
63:     m(nlvl)=m(1)+surf
64:     end if
65: if(hmax.ne.0.) then
66:     eachhmax.
67: end if
68: if(check=.false.) then
69:     do i=1,ilvl
70:       if((h(i).le.hmax+5.e-5).and.(h(i).ge.hmax-5.e-5)) check=.true.
71:       end do
72:     end if
73:     if(check) then
74:       h(nlvl)=hmax
75:     end if
76:     The array is sorted.
77:     do 10 im=1,nlvl-1
78:       do 20 j=i+nlvl
79:         if(h(j).gt.h(i)) goto 20
80:         dum=h(i)
81:         h(i)=h(j)
82:         h(j)=dum
83:     end do
84:     end do
85: if(hmax.ne.h(nlvl)) then
86:   do j=1,nlvl
87:     if(h(j).eq.hmax) then
88:       bet=(h(j)-h(j-1))/(h(j+1)-h(j-1))*(m(j+1)-m(j-1))
89:     end if
90:   end do
91: else
92:     rint=(h(nlvl)-h(nlvl-1))/(h(nlvl-1)-h(nlvl-2))*(m(nlvl-1)-
93:       m(nlvl-2))
m(nlv1) = m(nlv1-1) + rint
end if

do i=1, nlvl-1
   dmdh(i) = (m(i+1) - m(i)) / (h(i+1) - h(i)) * 1.e-3
   if(ABS(dmdh(i)).lt.1.e-6) dmdh(i) = 1.e-6
end do

rmetperdot = rmax/390.
The transmitter height is checked to see if it is
at a breakpoint. If so, the unallowable angular range is
calculated.

forbid = 0.
duct = .false.
do i=1, nlvl
   if((h(i) .le. ht+1.e-6).and.(h(i) .ge. ht-1.e-6)) then
      if((dmdh(i-1) .gt. 0.).and.(dmdh(i) .lt. 0.)) forbid =
         SQRT(-2.e-3 * dmdh(i) * rmetperdot)
   end if
end do
if(forbid .ne. 0.) duct = .true.
return
end
SUBROUTINE INTERPOL

Purpose: This subroutine checks if a transition from one color to the next has been reached.

subroutine interpol(dif_2, rng_2, hp_2, rng, hyt, hs_2)
$INCLUDE 'errvar'
save dif_1, rng_1, hp_1, hs_1
if(frst) then
  dif_1=dif_2
  rng_1=rng_2
  hp_1=hp_2
  hs_1=hs_2
  goto 10
end if
if(erropt.eq.'a').or.(erropt.eq.'A')) then
  if(((dif_1.le.aber).and.(dif_2.gt.aber)).or.((dif_2.le.aber) .and.(dif_1.gt.aber))) then
    call inter(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,aber)
  else if(((dif_1.le.2.*aber).and.(dif_2.gt.2.*aber)).or.((dif_2.le.2.*aber).and.(dif_1.gt.2.*aber))) then
    call inter(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,2.*aber)
  else if(((dif_1.le.3.*aber).and.(dif_2.gt.3.*aber)).or.((dif_2.le.3.*aber).and.(dif_1.gt.3.*aber))) then
    call inter(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,3.*aber)
  else if(((dif_1.le.4.*aber).and.(dif_2.gt.4.*aber)).or.((dif_2.le.4.*aber).and.(dif_1.gt.4.*aber))) then
    call inter(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,4.*aber)
  else if(((dif_1.le.5.*aber).and.(dif_2.gt.5.*aber)).or.((dif_2.le.5.*aber).and.(dif_1.gt.5.*aber))) then
    call inter(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,5.*aber)
  else if(((dif_1.le.6.*aber).and.(dif_2.gt.6.*aber)).or.((dif_2.le.6.*aber).and.(dif_1.gt.6.*aber))) then
    call inter(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,6.*aber)
  end if
else if((erropt.eq.'p').or.(erropt.eq.'P')) then
  if((hs_1.eq.0.).or.(hs_2.eq.0.)) then
    hs_1=1.
    hs_2=1.
  end if
perc_1=(dif_1/hs_1)*100.
perc_2=(dif_2/hs_2)*100.
if(((perc_1.le.per).and.(perc_2.gt.per)).or.((perc_2.le.per) .and.(perc_1.gt.per))) then
  call inter(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,per)
else if(((perc_1.le.2.*per).and.(perc_2.gt.2.*per)).or.((perc_2.le.2.*per).and.(perc_1.gt.2.*per))) then
  call inter(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,2.*per)
else if(((perc_1.le.3.*per).and.(perc_2.gt.3.*per)).or.((perc_2.le.3.*per).and.(perc_1.gt.3.*per))) then
  call inter(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,3.*per)
else if(((perc_1.le.4.*per).and.(perc_2.gt.4.*per)).or.((perc_2.le.4.*per).and.(perc_1.gt.4.*per))) then
call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,4.*per)
else if((perc_1.1.e.5.*per).and.(perc_2.gt.5.*per)).or.
  + ((perc_2.1.e.5.*per).and.(perc_1.gt.5.*per))) then
  call int(perc_1,perc_2,rng_1,rng_1,hp_1,hp_2,rng,hyt,5.*per)
else if(((perc_1.1.e.6.*per).and.(perc_2.gt.6.*per)).or.
  + ((perc_2.1.e.6.*per).and.(perc_1.gt.6.*per))) then
  call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,6.*per)
end if
end if
dif_1=dif_2
dif_2=rng_1=rng_2
 hp_1=hp_2
hsp_1=hs_2
return
end

******** SUBROUTINE INT **********

Purpose: This subroutine interpolates the transition from one
  height error increment (color) to the next.

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

subroutine int(dum_1,dum_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,fxdif)
rin=(fxdif-dum_1)/(dum_2-dum_1)*(rng_2-rng_1)
rng=rng_1+rin
hin=(rng-rng_1)/(rng_2-rng_1)*(hp_2-hp_1)
hyt=hp_1+hin
return
end

A-18
Subroutine LSFILES produces a two-column list, preceded by a number, of all data files stored in directory 'DIR'. The maximum number of data files 46.

**Glossary:**
- ifil : Counter for number of files in directory. Will be zero if no files in that directory.
- ire : One-half of ifil to print out on screen a two-column list of filenames
- name : 46 element array - stores filenames

```
subroutine lsfiles(dir, name, ifil)
    character*14 name(46)
    character(*) dir
    call system('ls //dir//'@datafiles@'//char(0))
    open(1,FILE='@datafiles@')
    ifil=0
    do j=1,46
        read(1,'(a14)',END=10)name(j)
        ifil=ifi1+1
    end do
    continue
    close(1)
    ire=integer(ifil/2.)
    do j=1,ire
        if(mod(ifil,2).ne.0).and.(j.eq.ire) then
            write(6,'(i2,3x,a14)')j,name(j)
        else
            write(6,'(i2,3x,a14,10x,i2,3x,a14)')j,name(j),j+ire.
            name(j+ire)
        end if
    end do
    call system('re @datafiles@')
    return
end
```
SUBROUTINE PUTIN

Purpose: This subroutine puts in all user information into a file for later retrieval. This allows the user to run a second display for the same profile (with minor changes if so desired) without going through each prompt again.

subroutine putin(rl,ru,irplot,filvar,fun)
include 'errvar'
character*3 fun
open(2,FILE='usin')
if((erropt.eq.'A').or.(erropt.eq.'A')) then
   filvar=uber
else
   filvar=per
end if
!
call keyread(rl,fun)
write(2,'(25x,="USER INPUT PARAMETERS ")')
write(2,*)
write(2,'(5x,"You are now in "vi". If there are any changes,"
* desired use "vi" commands")')
write(2,'(5x,"for editing. Exit by typing ",")')
write(2,*)
write(2,*)
write(2,*)
write(2,*)
if(fun.eq.'fps') then
   write(2,'(f7.0,10x,"Maximum height for display in feet")')
   hmax=3.280839
   write(2,'(f4.0,13x,"Maximum range for display in km")')
   rmax=1.85318
   write(2,'(f6.0,11x,"Antenna height in feet")')ht=3.280839
else
   write(2,'(f7.0,10x,"Maximum height for display in meters")')
   hmax
   write(2,'(f4.0,13x,"Maximum range for display in km")')
   rmax
   write(2,'(f6.0,11x,"Antenna height in meters")')ht
end if
!
write(2,'(f5.0,12x,"Lower elevation angle (\(\text{rad}\))")')rl
write(2,'(f5.0,12x,"Upper elevation angle (\(\text{rad}\))")')ru
write(2,'(i4,13x,"Number of rays to be plotted (integer)")')
irplot
write(2,'(a1,16x,"A or P -- Absolute or Percent error")')erropt
write(2,'(i1,16x,"1 - graph ; 2 - numbers : for profile printout
* integer")')ipro_print
write(2,*)
write(2,'(10x,"** Fill in only the line that applies **")')
write(2,*)
if((erropt.eq.'A').or.(erropt.eq.'A')) then
   if(fun.eq.'fps') then
      write(2,'(f5.0,12x,"Error increment in feet")')
   else
      filvar=3.280839
   else

A-20
S7: write(2,'(f5.0,12x,"Error increment in meters")') filvar
59: end if
60: write(2,'(17x,"Percent error increment")')
61: else if((erropt eq.'p').or.(erropt eq.'P')) then
62: write(2,'(f3.0,14x,"Percent error increment")') filvar
63: end if
64: close(2)
65: call rdffl(r1,ru,irplot,filvar,fun)
66: return
67: end

S8: subroutine rdffl(r1,ru,irplot,filvar,fun)
70: include 'errvar'
71: character*3 fun
72: character*80 dum
73: call system('vi usin')
74: open(2,FILE = 'usin')
75: read(2,'(5(a80/),a80)')(dum,i-1,6)
76: read(2,'(f7.0/),(f4.0/),(f6.0/)')(hmax,rmax,ht)
77: if(fun eq.'fps')then
78: hmax=hmax/3.280839
79: rmax=rmax*1.85318
80: ht=ht/3.280839
81: end if
82: read(2,'(2(a80/),a80)')(dum,i=1,3)
83: if((erropt eq.'a').or.(erropt eq.'A')) then
84: read(2,'(f5.0/)') filvar
85: aber=filvar
86: if(fun eq.'fps') aber=aber/3.280839
87: end if
88: if((erropt eq.'p').or.(erropt eq.'P')) then
89: read(2,'((a80/),f5.0/)')(dum,filvar)
90: per=filvar
91: end if
92: close(2)
93: do j=1,35
94: h(j)=0.
95: m(j)=0.
96: dmdh(j)=0.
97: end do
98: do j=1,300
99: angle(j)=0.
100: end do
101: return
SUBROUTINE READDA

Purpose: Subroutine READDA reads environmental data from data files the user specifies.

Glossary:
- profile: Character array containing environmental data filenames.
- ipick: Counter for "profile" array indicating number of environmental data file.

SUBROUTINE READDA(ipick,profile)

INCLUDE 'errvar'
character*14 profile(50)
character*60 dum

Open data file; read data

open(10,FILE=' prof//profile(ipick))
read(10,'(3(a80/),a80)')(dum,ipick)
read(10,'((a14/),a1/)/profile(ipick),hun,reun
read(10,'(4(a80/),a80)')(dum,i=1,5)

ivm=0

Read H and M arrays.

do i=1,33
  if(reun.eq.'M') then
    read(10,'((f7.1,5x,f6.1)',END=10)orr(i),orm(i)
  else
    read(10,'((f7.1,18x,f6.1)',END=10)orr(i),orm(i)
  end if
  ivm=ivm+1
end do

Continue

close(10)
return
end
SUBROUTINE REGULAR

Purpose: This begins the main calculations of the program. It does not allow rays to be traced within an angular limit if the transmitter height is at a break point.

Glossary:

duct Logical flag indicating if the transmitter height is at a break point.
irplot Number of rays to be plotted.
alpha Beginning angle.
fixang Initial elevation angle currently being used.
jf Counter.
hbef Beginning height.
rbef Beginning range.
forbid Rays are not traced in this angular range.

---

subroutine regular(irays,duct,forbid,ix)

include '/usr/include/starbase.f2.h'
include '/usr/include/starbase.f1.h'

INCLUDE 'errvar'

logical duct

********** Begin main loop. **********

do ik=1,irays
  frst=.true.
  if(plot)frst=.true.
  if(m.ne.1) then
    mvfl=.false.
  else
    if(plot) then
      mvfl=.true.
    else
      if(f7.eq.'fps') then
        call move2d(fildes,0.,ht*3.280839)
      else
        call move2d(fildes,0.,ht)
      end if
    end if
  end if
end if

********** Initialize variables for ray tracing. **********

if(plot) call start(ik,ix)
alpha=angle(ik)
fixang=angle(ik)
jf=ix
hbef=ht
rbef=0.

********** Check if transmitter is at a breakpoint. If so, skip angles within forbidden range. **********
if (duct) then
  if (ABS(alpha).le.forbtd) then
    rpmrmax
    call line_color_index(fildes,1)
  else
    call draw2d(fildes, rp*.5396117, ht*3.280839)
  end if
else
  call draw2d(fildes, rp, ht)
end if
end if

if (f7.eq.'fps') then
  call draw2d(fildes, rp*.5396117, ht*3.280839)
else
  call draw2d(fildes, rp, ht)
end if

If initial elevation angle is negative then call routine for down-going rays. If it is positive, then call routine for up-going rays. If it is 0 then call routine to check on the value of the gradient. ***********

if (alpha.eq.0) then
  call zero(alphabef,ij,rbef,rp, hp, fixang)
else if (alpha.gt.0) then
  call up(alpha, hbe, i, rbe, rp, hp, fixang)
else
  call down(alpha, hbe, i, rbe, rp, hp, fixang)
end if

if((rp.lt.rmax). and.(hp.lt.hmax)) goto 30
continue
end do
return
end

***************************************************************************** SEVEN*****************************************************************************

Purpose: This routine performs the ray-tracing procedure 7 times (once for each color) for ease in plotting onto the graphics plotter.

***************************************************************************** SEVEN*****************************************************************************

subroutine seven(irays,duct,forbid,ix)

INCLUDE 'errvar'

logical duct
do i=1,7
  mvfl=true.
call regular(irays,duct,forbid,ix)
end do
return
end
SUBROUTINE STEPFIXSCR

Purpose: STEPFIXSCR defines the color scale for plotting height and range onto the screen in the correct color.

Glossary:
- **percerr**: Percentage error between height of non-standard atmosphere ray and height of standard atmosphere ray.
- **rng**: Interpolated range between color transitions.
- **hyt**: Interpolated height between color transitions.

```
subroutine stepfixscr(hdif,hs,rp,hp,rng,hyt)
   include '/usr/include/starbase.f1.h'
   include '/usr/include/starbase.f2.h'
end include 'errvar'
if(rng.ne.0.) then
   if(f7.eq.'fps') then
      call draw2d(fildes,rng/1.86318,hyt=3.280839)
   else
      call draw2d(fildes,rng,hyt)
   end if
end if

define color scale for absolute height error.

if((erropt.eq.'a').or.(erropt.eq.'A')) then
   if(hdif.le.aber) then
      call linecolorindex(fildes,1)
   else
      if(hdif.gt.aber) then
         call linecolorindex(fildes,2)
      else
         call linecolorindex(fildes,3)
      end if
   else if((hdif.gt.2.*aber).and.(hdif.le.3.*aber)) then
      call linecolorindex(fildes,4)
   else if((hdif.gt.3.*aber).and.(hdif.le.4.*aber)) then
      call linecolorindex(fildes,5)
   else if((hdif.gt.4.*aber).and.(hdif.le.5.*aber)) then
      call linecolorindex(fildes,6)
   else if((hdif.gt.5.*aber).and.(hdif.le.6.*aber)) then
      call linecolorindex(fildes,7)
   end if
end if

define color scale for percentage height error.

else if((erropt.eq.'p').or.(erropt.eq.'P')) then
   if(hs.eq.1.) then
      percerr=200.
   else
      percerr=(hdif/hs)*100.
   end if
   if(percrr.le.per) then
      call linecolorindex(fildes,1)
end subroutine stepfixscr
```
57:     else if((percerr.gt.per).and.(percerr.ge.2.*per)) then
58:       call line_color_index(fildes,3)
59:     else if((percerr.gt.2.*per).and.(percerr.le.3.*per)) then
60:       call line_color_index(fildes,4)
61:     else if((percerr.gt.3.*per).and.(percerr.le.4.*per)) then
62:       call line_color_index(fildes,5)
63:     else if((percerr.gt.4.*per).and.(percerr.le.5.*per)) then
64:       call line_color_index(fildes,6)
65:     else if((percerr.gt.5.*per).and.(percerr.le.6.*per)) then
66:       call line_color_index(fildes,7)
67:     else if((percerr.gt.6.*per).and.(percerr.le.7.*per)) then
68:       call line_color_index(fildes,8)
69:     end if
70:     end if
71:     if(f7.eq.'fps') then
72:       call draw2d(fildes,rp/1.85318,hp*3.280839)
73:     else
74:       call draw2d(fildes,rp,hp)
75:     end if
76:     return
77:   end if
78:
79:   subroutine renew
80:   SUBROUTINE RENEW
81:   Purpose: RENEW re-initializes arrays used in ray tracing.
82:   SUBROUTINE RENEW
83:   subroutine renew
84:   $INCLUDE lerrvar$
85:   do j=1,35
86:     dadj(j)=0.
87:   do j=1,33
88:     h(j)=0.
89:   do j=1,300
90:     m(j)=0.
91:     end do
92:   do j=1,33
93:     orh(j)=0.
94:     end do
95:   do j=1,300
96:     orh(j)=0.
97:     end do
98:   return
99:   end
SUBROUTINE UP

Purpose: This subroutine performs a ray trace for up-going rays.

subroutine up(alpha, hbegin, i, rbef, rp, hp, fixang)
  include '/usr/include/starbase.f1.h'
  include '/usr/include/starbase.f2.h'
  INCLUDE 'errvar'

Begin at transmitter height and calculate until the ray reaches a maximum or maximum height.

  do while((i.lt.nlvl).and.(alpha.ge.0.))
    rp=rbef+incr
    if(rp.ge.rrmax) goto 20
    alphap=alpha+dalph(iJ)*(rp-rbef)
  end do

  if(alphap.le.0.) then
    alphap=0.
    rp=rbef-alphap/dalph(iJ)
    hp=rbef+alphap*2/2.e-3/dalph(iJ)
  else
    alphap=SORT(rp)
    rp=rbef+(alphap-alpha)/dalph(iJ)
  end if
  end if

  if(i.ge.nlvl) then
    alphap=0.
    rp=rbef-alphap/dalph(iJ)
    hp=rbef+alphap*2/2.e-3/dalph(iJ)
    else
      alphap=SORT(rp)
      rp=rbef+(alphap-alpha)/dalph(iJ)
  end if
  end if
  if((rp.ge.rrmax).or.(hp.ge.hhmax)) goto 10

  call stand(fixang, rp, hp)
  hbeff=hp
  rbef=rp
  alpha=alphap
  if(alpha.ge.0.) goto 10
end do
57: 10 if(ij.eq.nlvl) ij=nlvl-1
58: c
59: c ********** If ray is calculated past the maximum height, then set
60: c HP equal to HMAX and calculate range at exactly the maximum height. *
61: c
62: if(hp.ge.hmax) then
63: hp=hmax
64: alphap=SGRT(alpha**2 + 2.e-3*dmdh(ij)*(hp-hbef))
65: rp=rbef+(alphap-alpha)/dmdh(ij)
66: if(rp.ge.rmax) goto 20
67: call stand(fixang,rp,hp)
68: end if
69: c
70: c ********** If ray reaches beyond maximum range then calculate height
71: c at exactly maximum range. *******************************
72: c
73: 20 if(rp.ge.rmax) then
74: rp=rmax
75: alphap=alpha+dmdh(ij)*(rp-rbef)
76: hp=hbef+(alphap**2-alpha**2)/2.e-3/dmdh(ij)
77: call stand(fixang,rp,hp)
78: end if
79: return
80: end
SUBROUTINE ZERO

Purpose: This subroutine determines which subroutine will be called when a ray reaches a maximum or minimum. If a ray reaches a maximum, then depending on the value of the gradient, DOWN is called. Likewise, if a ray reaches a minimum, UP is called.

SUBROUTINE zero(alpha,hbef,ij,rbef, rp, hp, fixang)

$INCLUDE 'errvar'

if(dmdh(ij).lt.0.) then
  call down(alpha,hbef,ij,rbef, rp, hp, fixang)
else
  call up(alpha,hbef,ij,rbef, rp, hp, fixang)
end if

return

end