

AD-A178 382

NSC
SYSTEMS CENTER San Diego, California 92152-5000

12

Technical Report 1152
January 1987

An Altitude-Error Display for Height-Finder Radar

A. E. Barrios

DTIC
ELECTE
MAR 25 1987
S D

*Original contains color
plates; All DTIC reproductions
will be in black and
white*

DTIC FILE COPY



Approved for public release, distribution is unlimited

87 3 24 0

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

E. G. SCHWEIZER, CAPT, USN
Commander

R. M. HILLYER
Technical Director

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

AD-A178382

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC/TR-1152			7a. NAME OF MONITORING ORGANIZATION		
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center		6b. OFFICE SYMBOL (if applicable) Code 543	7b. ADDRESS (City, State and ZIP Code)		
6c. ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000			9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Technology		8b. OFFICE SYMBOL (if applicable) ONT	10. SOURCE OF FUNDING NUMBERS		
8c. ADDRESS (City, State and ZIP Code) Office of Chief of Naval Research Washington, DC 22217			PROGRAM ELEMENT NO. 62759N	PROJECT NO. N02C	TASK NO. RW69
			AGENCY ACCESSION NO. DN888 715		
11. TITLE (Include Security Classification) An Altitude-Error Display for Height-Finder Radar					
12. PERSONAL AUTHOR(S) A.E. Barrios					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM <u>SEP 85</u> TO <u>APR 86</u>		14. DATE OF REPORT (Year, Month, Day) JANUARY 1987	
15. PAGE COUNT 51					
16. SUPPLEMENTARY NOTATION <i>Fig 1 - Legend include:</i>					
17. COBALT CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB GROUP	Height-finder radars Target height Ray-trace display		
			Altitude-error display HP9000-800 series computer, and IREPS <i>app</i>		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) Strong ducting conditions affect height-finder radars in giving accurate target positions. An altitude-error display has been developed to show the amount of error present.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ UIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE MONITORIAL A.E. Barrios			22b. TELEPHONE (Include Area Code) (619) 226-7247		22c. OFFICE SYMBOL Code 543

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).

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



CONTENTS

	Page
INTRODUCTION.....	1
BACKGROUND.....	1
PROGRAM.....	5
SAMPLE DISPLAYS.....	7
CONCLUSIONS.....	25
RECOMMENDATION.....	25
APPENDIX A: COMPUTER PROGRAM LISTING.....	A-1

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, Dm/dh , is the change in M-units with respect to the change in height and is defined as

$$Dm/dh(i) = \left[\frac{M(i+1) - M(i)}{H(i+1) - H(i)} \right] \cdot 10^{-3}$$

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 the height at 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

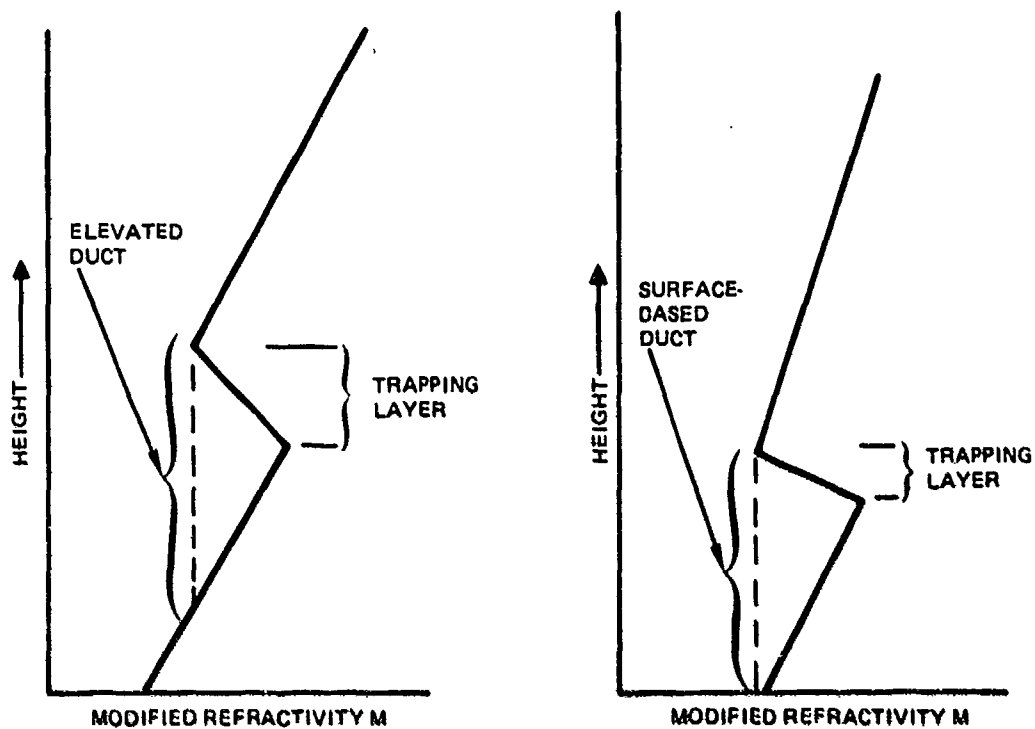


Figure 1. Examples of elevated and surface-based ducts from elevated layers.

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

Types of Refraction	M-Gradient
Trapping	≤ 0 M/km ≤ 0 M/kft
Superrefractive	0 to 79 M/km 0 to 24 M/kft
Standard	79 to 157 M/km 24 to 48 M/kft
Subrefractive	> 157 M/km 48 M/kft

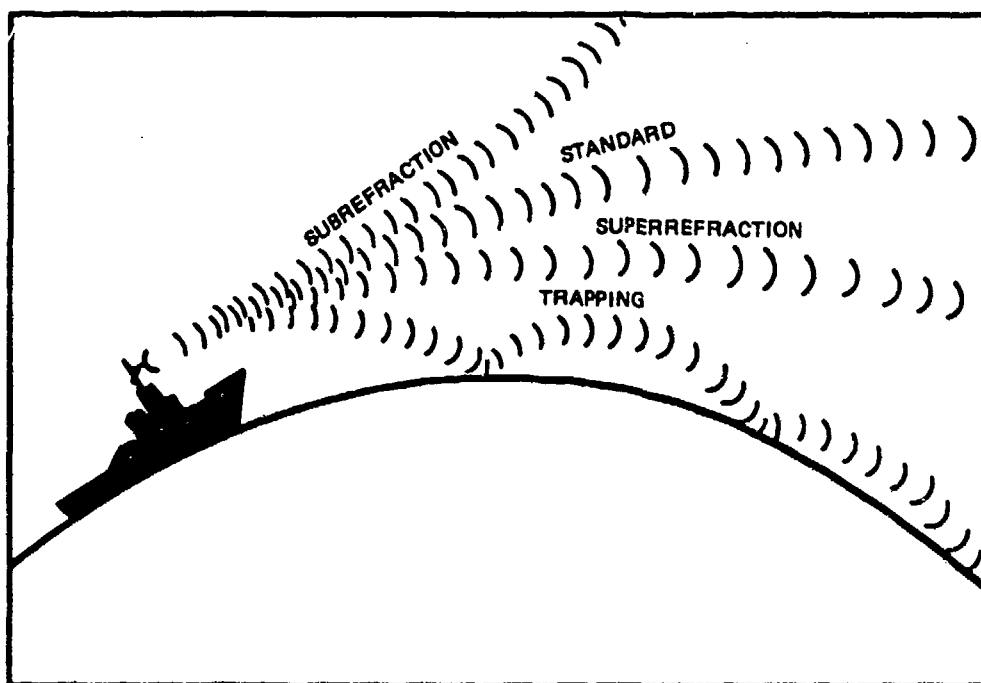


Figure 2. Relative bending for the four types of refraction.

α = elevation angle at beginning of calculation (rad)

α' = elevation angle at end of calculation (rad)

h = height at beginning of calculation (m)

h' = height at end of calculation (m)

r = range at beginning of calculation (km)

r' = range at end of calculation (km)

Case 1: $\alpha > 0$, h' known (figure 3)

$$\alpha' = \sqrt{\alpha^2 + 2 \cdot 10^{-3} D m d h(i) (h' - h)} \quad , \quad \text{if } H(i) \leq h < h' \leq H(i+1)$$

$$r' = r + \frac{\alpha' - \alpha}{D m d h(i)}$$

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

$$\alpha' = 0 \quad ,$$

$$r' = r - \frac{\alpha}{D m d h(i)} \quad , \quad \text{and}$$

$$h' = h - \frac{\alpha^2}{2 \cdot 10^{-3} D m d h(i)}$$

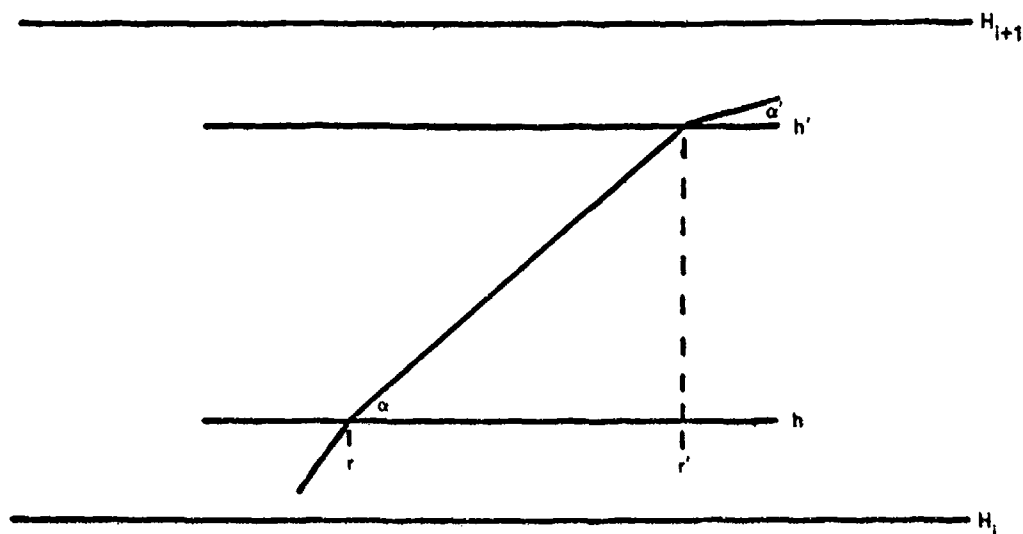


Figure 3. Ray trace variables for $\alpha > 0$.

Case 2: $\alpha > 0$, r' known (figure 3 applies)

$$\alpha' = \alpha + Dmdh(i) (r' - r) \quad , \quad \text{if } h < H(i+1),$$

$$h' = h + \frac{\alpha'^2 - \alpha^2}{2 \cdot 10^{-3} Dmdh(i)}$$

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

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

$$r' = r + \frac{\alpha' - \alpha}{Dmdh(i)}$$

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

$$\alpha' = 0 \quad ,$$

$$r' = r - \frac{\alpha}{Dmdh(i)} \quad ,$$

$$h' = h - \frac{\alpha^2}{2 \cdot 10^{-3} Dmdh(i)} \quad .$$

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

$$\alpha' = \alpha + Dmdh(i) (r' - r) \quad , \quad \text{if } h' > H(i),$$

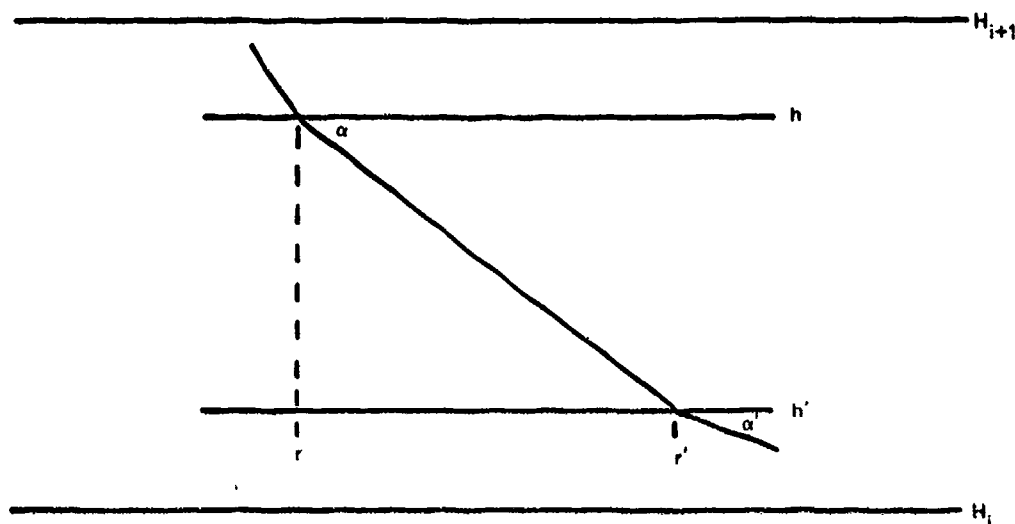


Figure 4. Ray trace variables for $\alpha < 0$.

$$h' = h + \frac{\alpha'^2 - \alpha^2}{2 \cdot 10^{-3} Dmdh(i)}$$

Case 5: $\alpha = 0$

If $Dmdh(i) > 0$, use Case 1 or Case 2 as appropriate.

If $Dmdh(i) < 0$, 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.
5. Upper elevation angle. Upper angular limit in milliradians.
6. Number of rays. This is the number of rays to be traced.

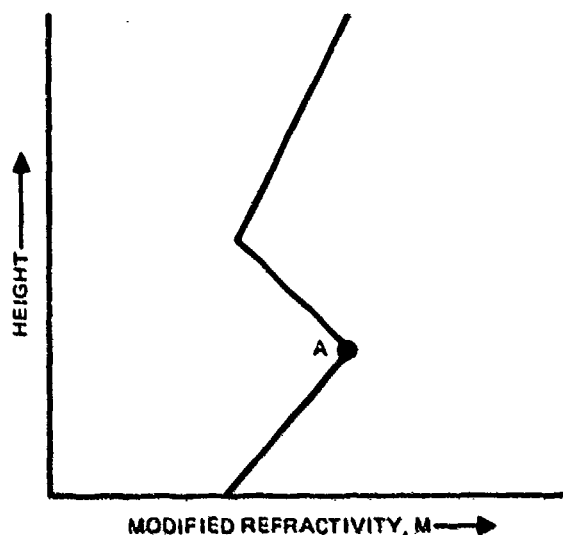


Figure 5. Breakpoint at A.

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 (Dm/dh) 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 Dm/dh 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 Dm/dh approaching zero, a value of $10e-6$ was assigned to Dm/dh if it fell in the region $-10e-6 < Dm/dh < 10e-6$.

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.

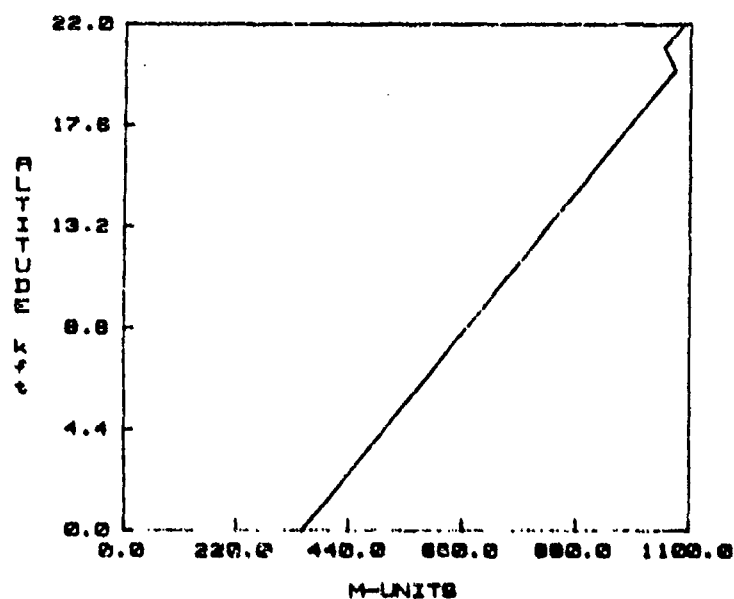
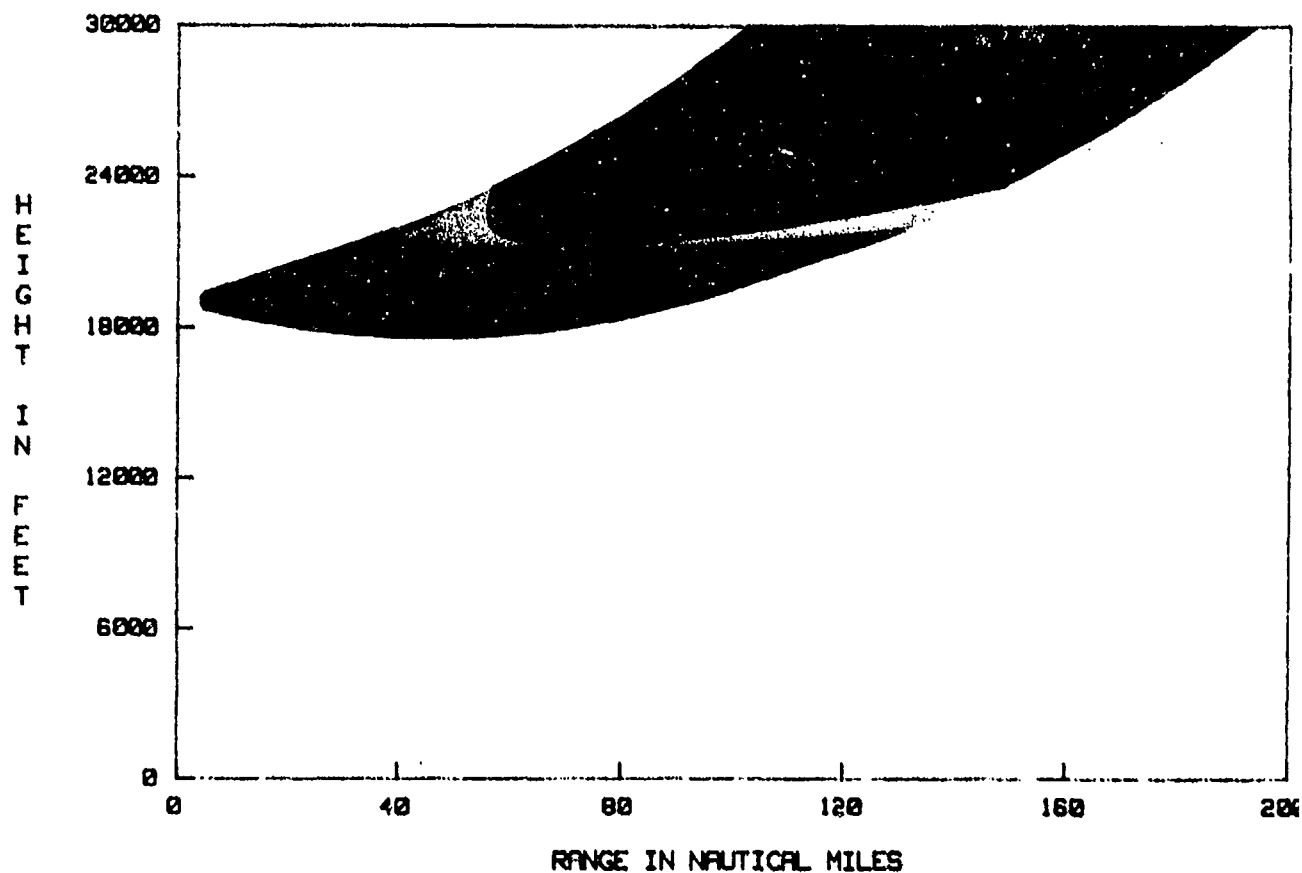
SAMPLE DISPLAYS

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.

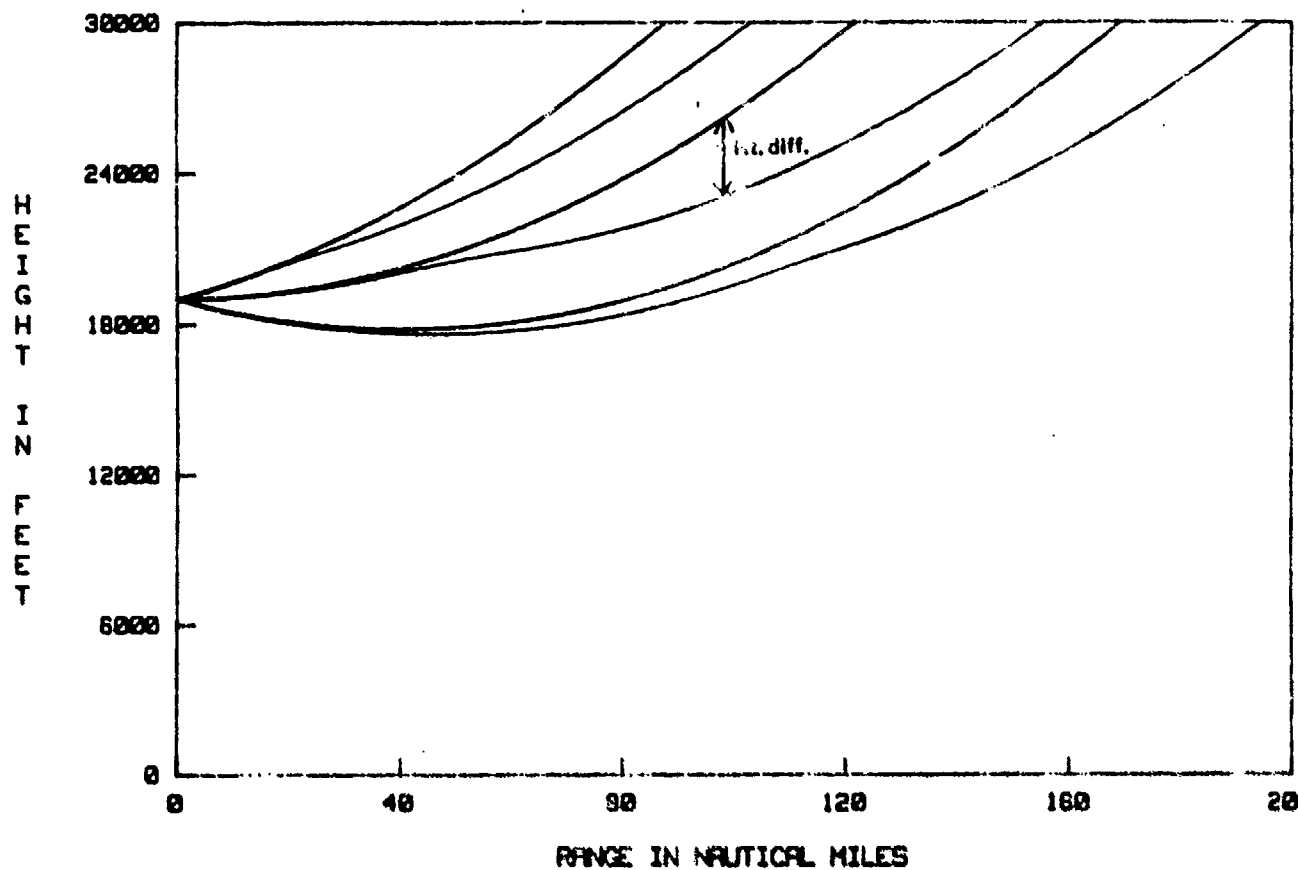


HEIGHT ERROR SCALE(ft)

- 0.0 - 500.0
- 500.1 - 1000.0
- 1000.1 - 1500.0
- 1500.1 - 2000.0
- 2000.1 - 2500.0
- 2500.1 - 3000.0
- > 3000.0

ANTENNA HEIGHT (ft): 19000.0
 LOWER ELEVATION ANGLE (in mrad): -10.0
 UPPER ELEVATION ANGLE (in mrad): 10.0
 NUMBER OF RAYS: 200

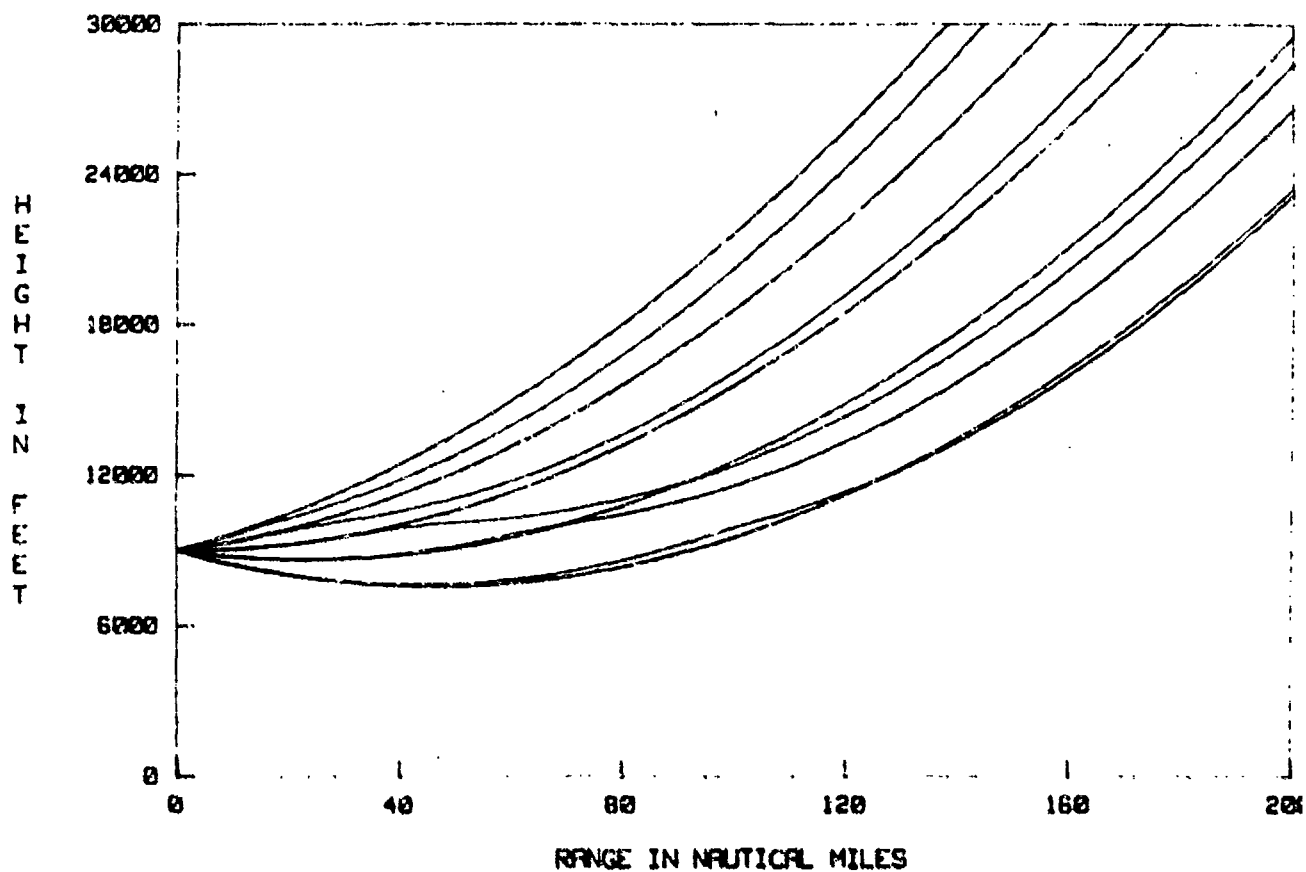
Figure 6. Error display for elevated duct - 200 rays.



HEIGHT(ft)	M-UNITS
.0	350.0
20000.0	1070.0
21000.0	1050.0
22000.0	1086.0

ANTENNA HEIGHT (ft): 19000.0
 LOWER ELEVATION ANGLE (in mrad): -10.0
 UPPER ELEVATION ANGLE (in mrad): 10.0
 NUMBER OF RAYS: 3

Figure 7. Ray-trace display for elevated duct of figure 6 (red) and standard atmosphere (black).



HEIGHT(ft)	M-UNITS
16.0	339.8
669.0	365.6
4125.0	492.3
6105.0	575.2
9418.0	702.7
9726.0	707.5
10256.0	688.7
10449.0	698.0
11008.0	715.7
12759.0	792.5
14409.0	861.5
20207.0	1109.9

ANTENNA HEIGHT (ft): 9000.0
 LOWER ELEVATION ANGLE (in mrad): -10.0
 UPPER ELEVATION ANGLE (in mrad): 10.0
 NUMBER OF RAYS: 5

Figure 8. Ray-trace display for elevated duct (red) and standard atmosphere (black).

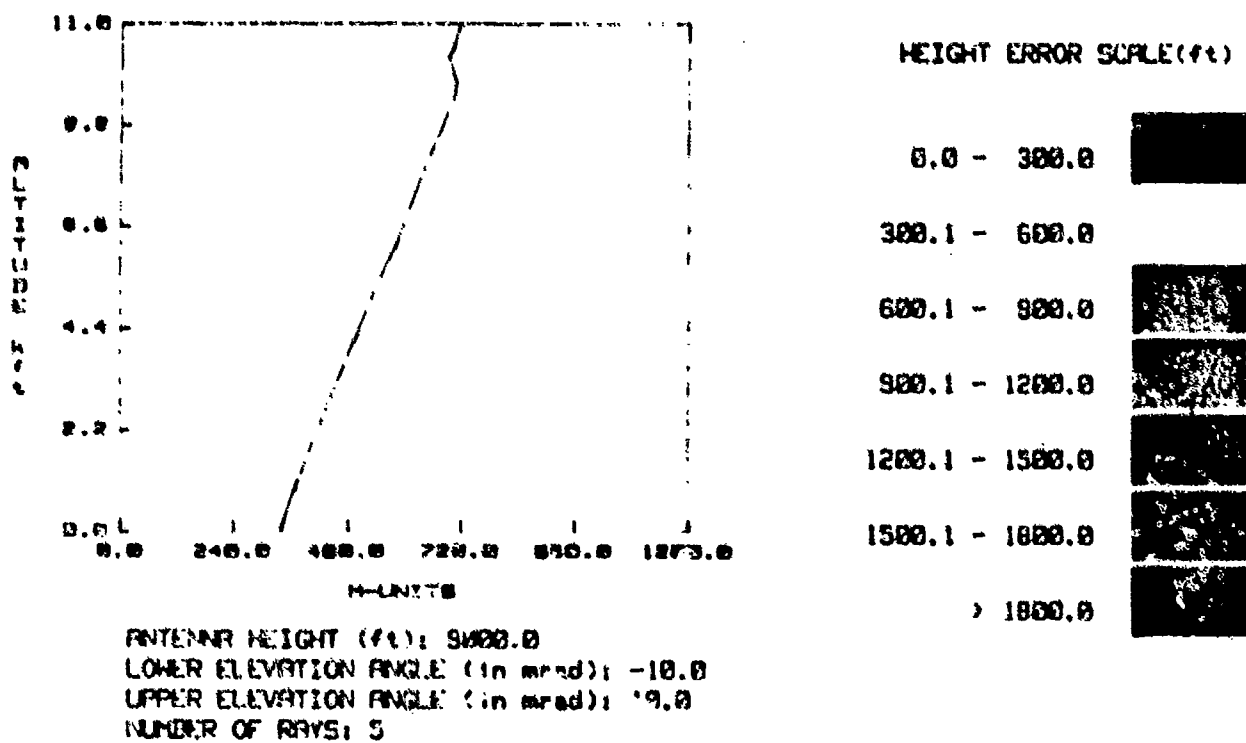
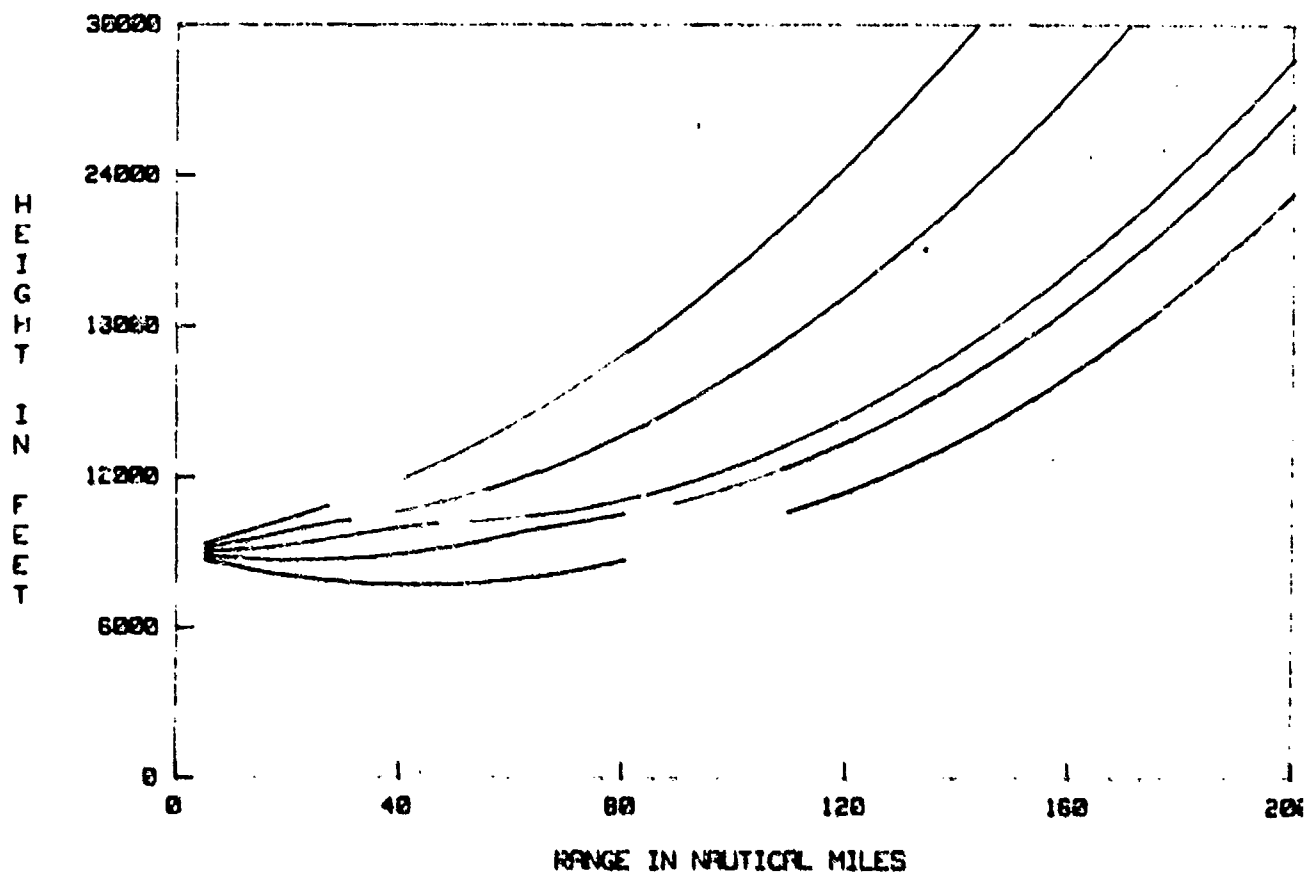
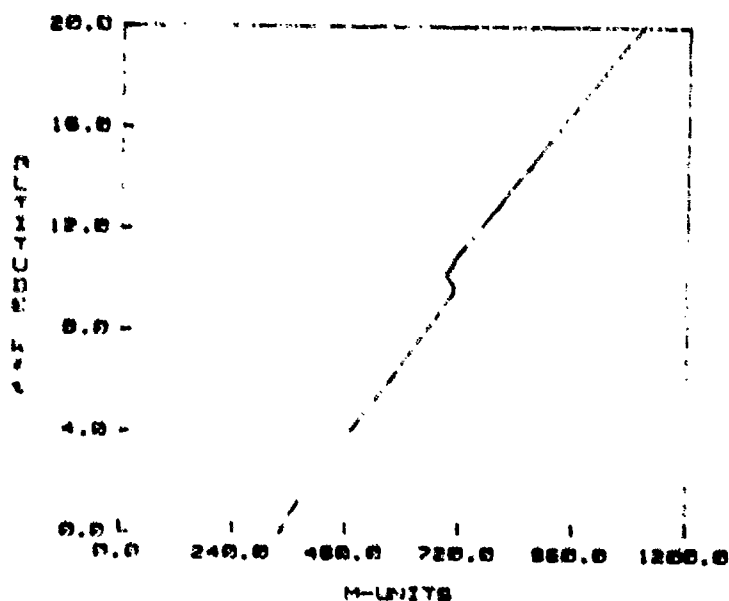
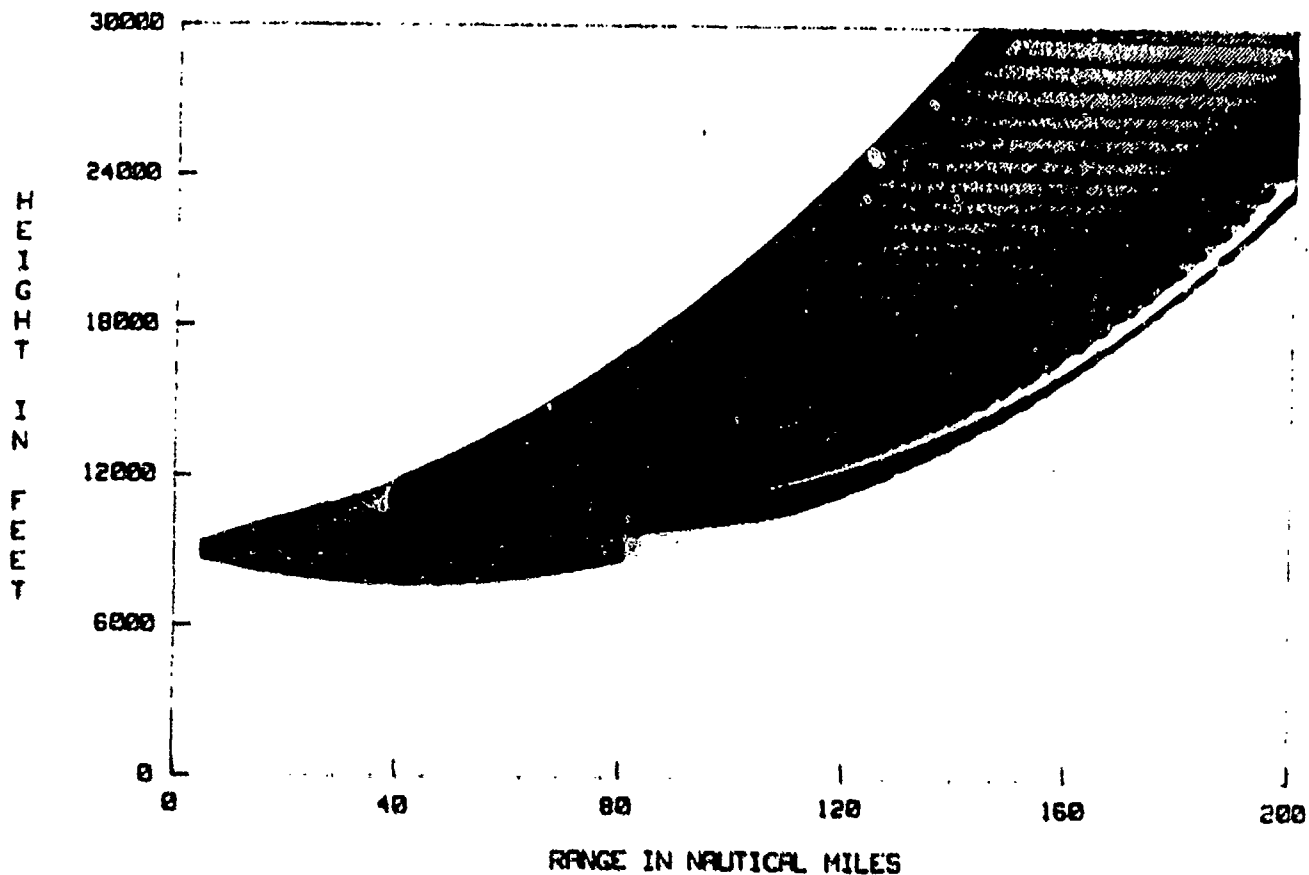


Figure 9. Error display for elevated duct of figure 8 — five rays.



HEIGHT ERROR SCALE (ft)

0.0 - 300.0

300.1 - 600.0

600.1 - 900.0

900.1 - 1200.0

1200.1 - 1500.0

1500.1 - 1800.0

> 1800.0

ANTENNA HEIGHT (ft): 9000.0
 LOWER ELEVATION ANGLE (in mrad): -10.0
 UPPER ELEVATION ANGLE (in mrad): 10.0
 NUMBER OF RAYS: 200

Figure 10. Error display for elevated duct of figure 8 - 200 rays.

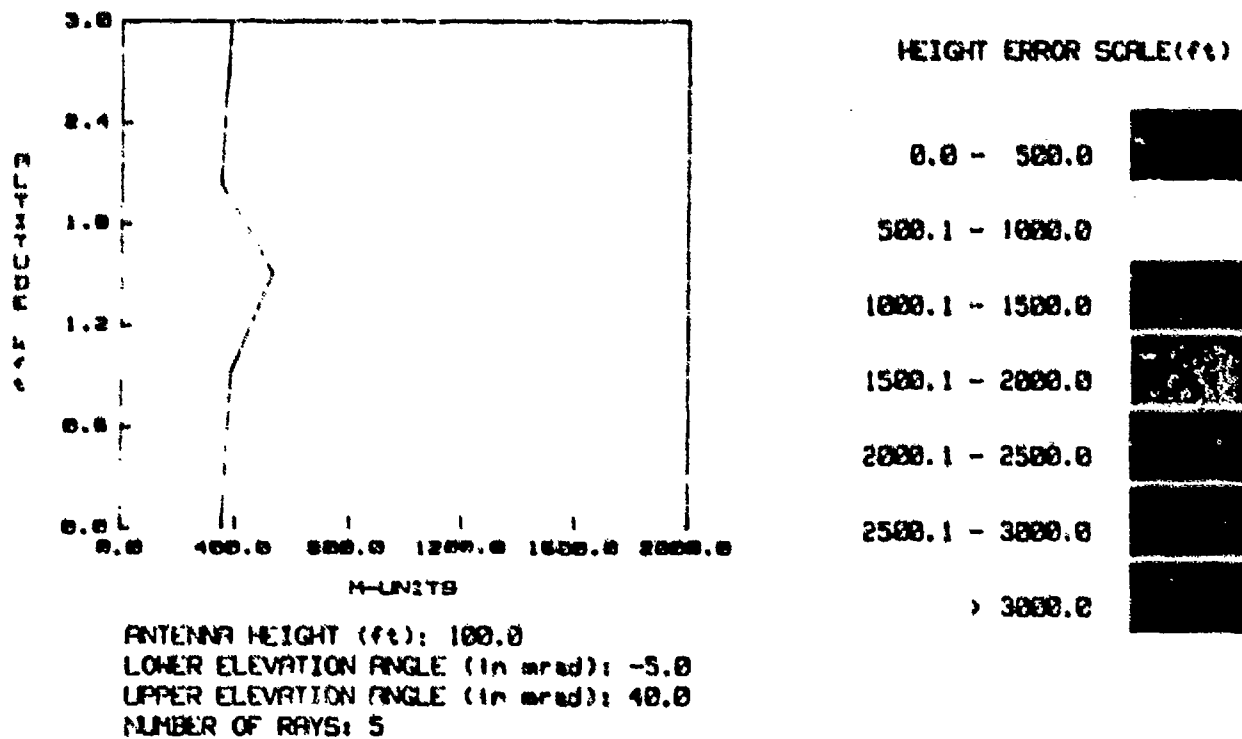
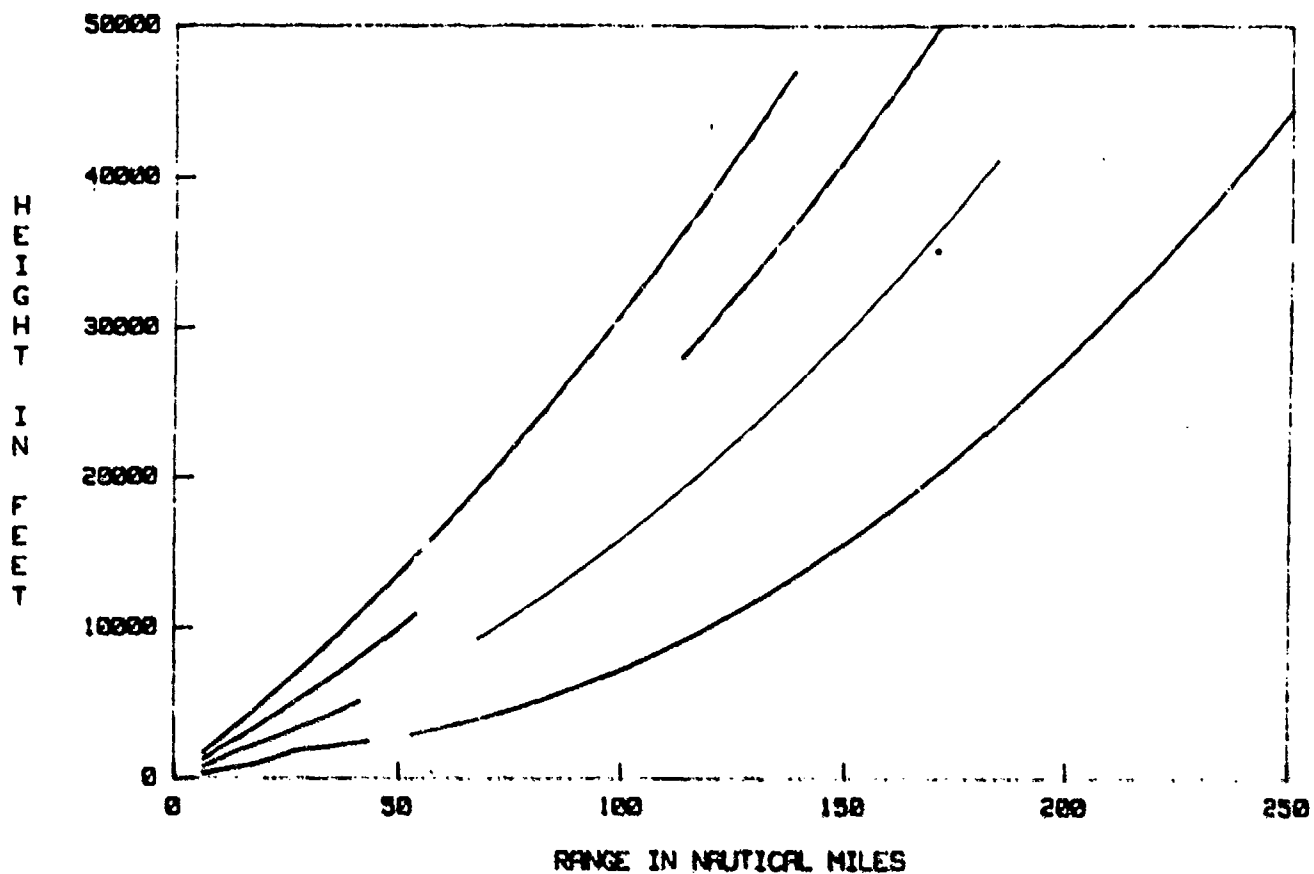
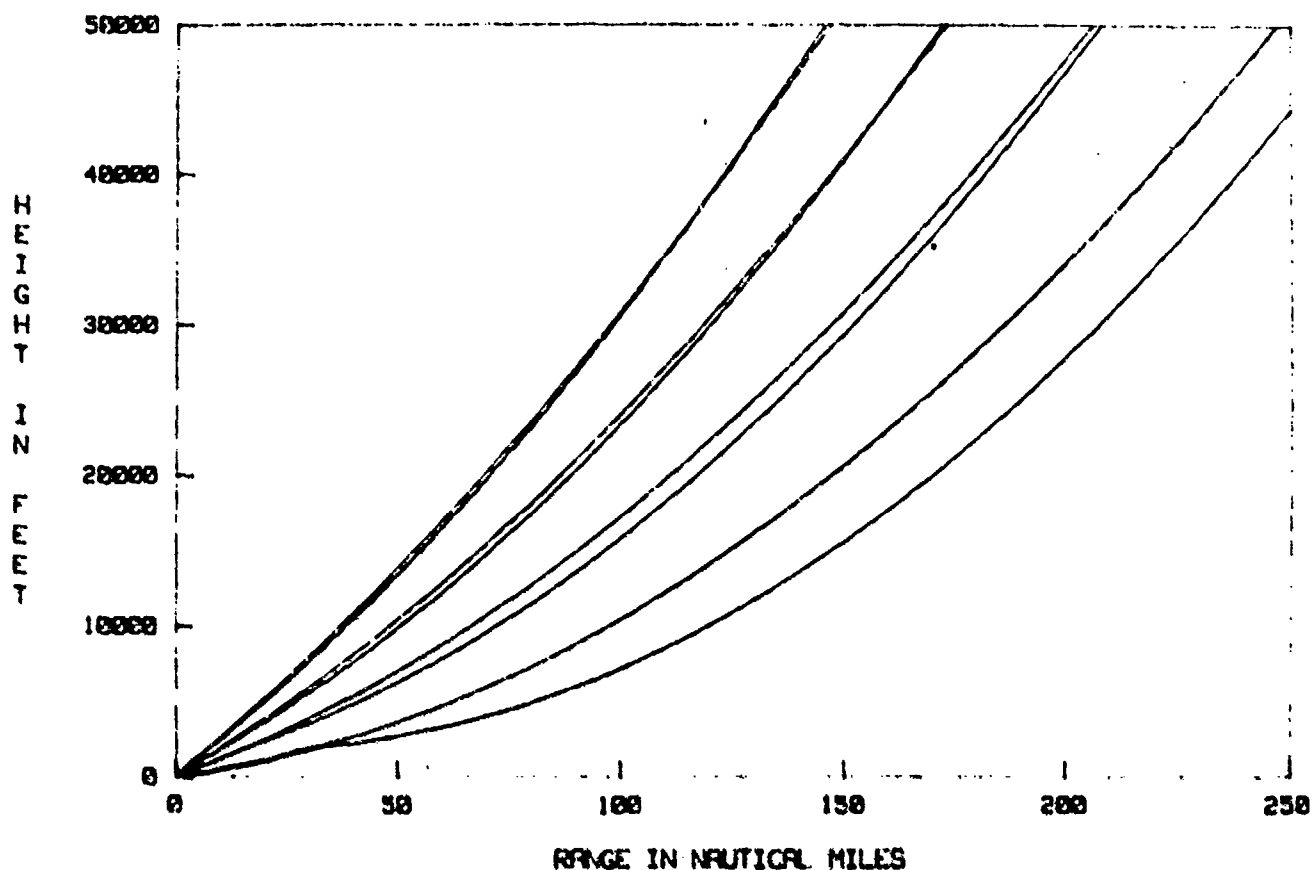


Figure 11. Error display for surface-based duct - five rays.



HEIGHT(ft)	M-UNITS	HEIGHT(ft)	M-UNITS
13.0	355.6	15639.0	913.4
931.0	385.6	19748.0	1088.9
1506.0	531.1	32300.0	1642.5
2034.0	349.3	38209.0	1906.3
2681.0	372.3		
3575.0	406.1		
4417.0	440.3		
4857.0	475.3		
5686.0	492.1		
6514.0	525.2		
6935.0	543.9		
7412.0	563.6		
8536.0	610.5		
9625.0	659.8		
13968.0	852.3		

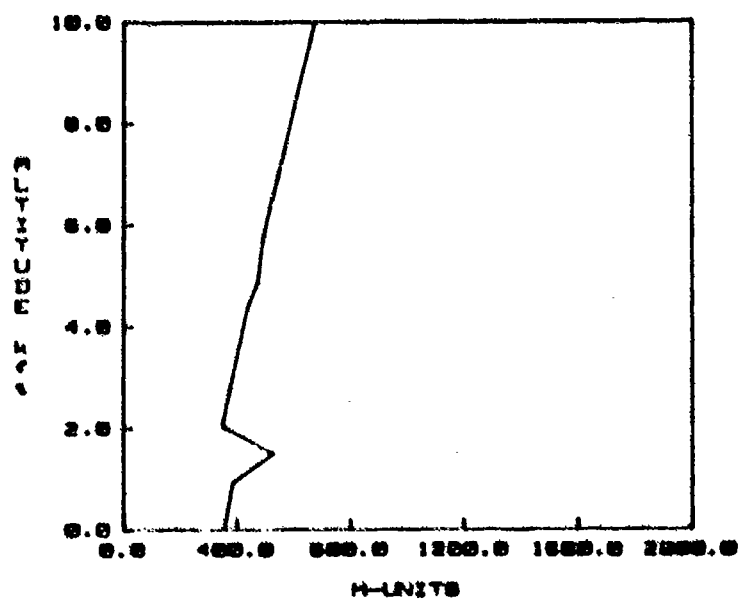
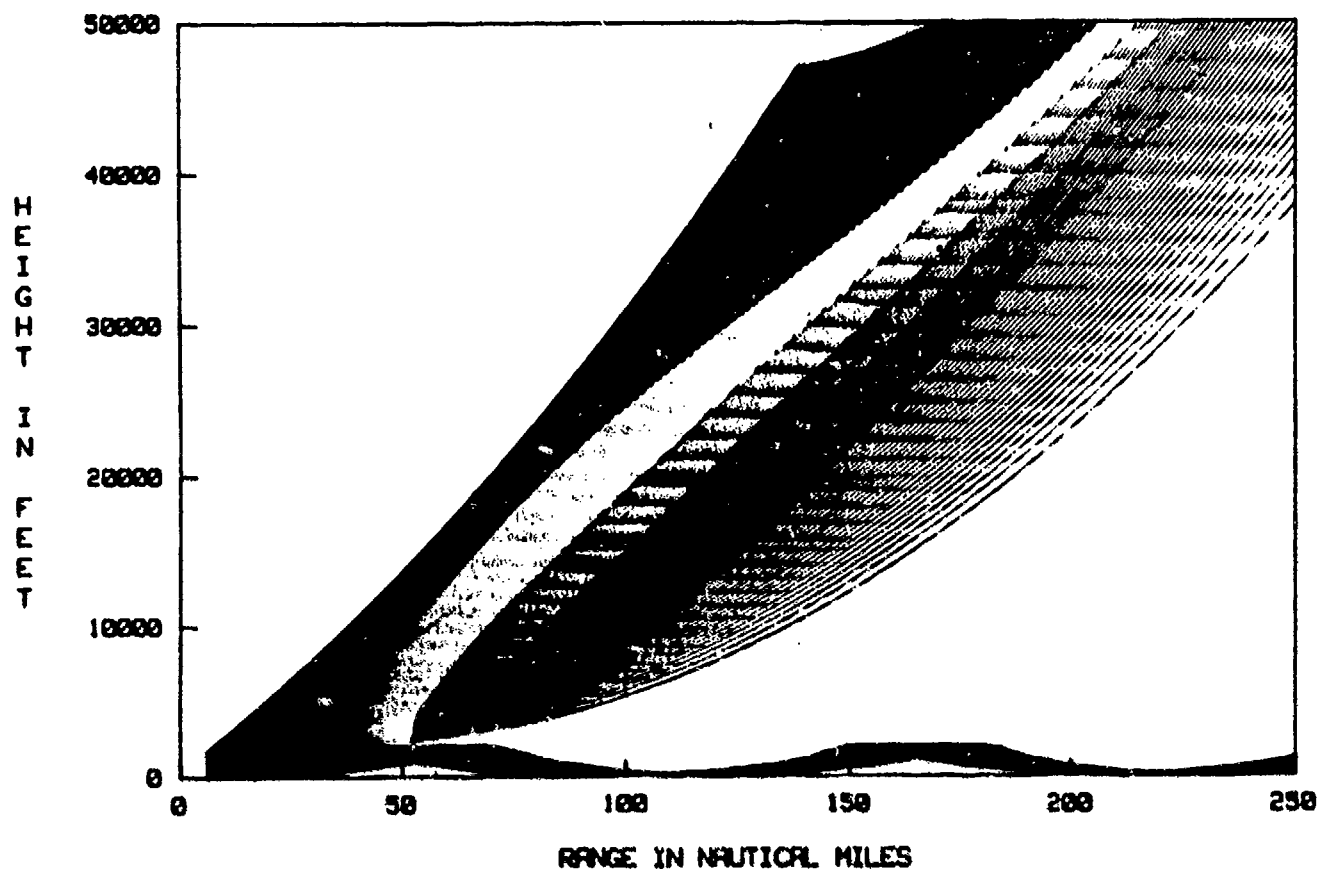
ANTENNA HEIGHT (ft): 100.0

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).



HEIGHT ERROR SCALE (ft)

0.0 - 500.0
 500.1 - 1000.0
 1000.1 - 1500.0
 1500.1 - 2000.0
 2000.1 - 2500.0
 2500.1 - 3000.0
 > 3000.0

ANTENNA HEIGHT (ft): 100.0
 LOWER ELEVATION ANGLE (in mrad): -5.0
 UPPER ELEVATION ANGLE (in mrad): 40.0
 NUMBER OF RAYS: 200

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).

APPENDIX A: COMPUTER PROGRAM LISTING

```

1:c      ***** Variables used in program *****
2:c
3:c      dadh:      35 element array containing the gradients of the
4:c                profile.
5:c      erropt:    Character string indicating if absolute ('a' or 'A')
6:c                or percent ('p' or 'P') for color scale was chosen.
7:c      aber:      Absolute error increment in meters.
8:c      per:        Percent error increment.
9:c      fildes:     File descriptor for starbase graphics routine.
10:c     index:      Integer indicating color to be used for plotting
11:c                (1-7).
12:c     mvfl:        Logical flag indicating when to move to the next
13:c                height-range point for plotting onto hardcopy when
14:c                changing pen colors.
15:c     ht:          Antenna height in meters.
16:c     hmax:        Maximum height for plot display in meters.
17:c     rmax:        Maximum range for plot display in kilometers.
18:c     orh:         33 element array containing heights of the original
19:c                profile.
20:c     orm:         33 element array containing m-units of the original
21:c                profile.
22:c     h:           35 element array containing heights of the new
23:c                profile.
24:c     im:          Counter indicating what color is currently being used
25:c                for hardcopy plotting.
26:c     plot:        Logical flag indicating if plotting onto hardcopy,
27:c                (T-plot, F-don't plot).
28:c     name:        Character string containing the data filename to be
29:c                read.
30:c     nlvl:        Number of new levels after extrapolation to the
31:c                surface and maximum height.
32:c     ilvl:        Number of levels in the original profile.
33:c     m:           33 element array containing m-units of the new
34:c                profile.
35:c     f7:          Character string indicating if 'AUTODUMP' is set to on
36:c                or off.
37:c     angle:       Array containg initial elevation angles (300 maximum).
38:c     hun:         Character string indicating what height units the
39:c                profile was stored in.
40:c     reun:        Character string indicating if refractivity values
41:c                were stored in m-units or n-units.
42:c     frstime:     Logical flag indicating first height-range point
43:c                calculated for new ray.
44:c     sor:         Starting range for hardcopy plotting.
45:c     soh:         Starting height for hardcopy plotting.
46:c     frst:        Logical flag indicating first storage of height-range
47:c                point for interpolation between color transitions.
48:c     ipro_print:  Integer indicating type of profile printout for
49:c                hardcopy; 1-graph, 2-number.
50:c     twice:       Logical flag indicating if program has been run at
51:c                least once.
52:c
53:c     common/errvar/dadh(35),erropt,aber,per,fildes,index,mvfl,
54:c     +ht,hmax,rmax,rinc,orh(33),orm(33),h(35),im,plot,name,
55:c     +nlvl,ilvl,m,f7,angle(300),hun,reun,frstime,sor,soh,frst,
56:c     +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
```

```

1:c ***** PROGRAM HFERR *****
2:c
3:c Purpose: This is the main program that will begin all procedures
4:c           for ray tracing.
5:c
6:c Glossary:
7:c     opst      Character string indicating if the 'option' or
8:c              'backup' key was hit.
9:c     outst     Character string variable.
10:c    optn      Real value of option number chosen by user.
11:c    ioptn     Integer value of option number chosen by user.
12:c    flag      Logical flag indicating if the user has gone
13:c             through the edit routine.
14:c    profile   Array containing all data filenames.
15:c    ifil      Number of data filenames in profile.
16:c    rpick     Real value of number of environmental data file
17:c             chosen.
18:c    ipick     Integer value of number of environmental data file
19:c             chosen.
20:c    rlouea    Lower elevation angle in mrad.
21:c    upea     Upper elevation angle in mrad.
22:c    rplot     Number of rays to be plotted.
23:c    irplot    Integer value of number of rays to be plotted.
24:c
25:c *****
26:c
27:c program hferr
28:c include '/usr/include/starbase.f1.h'
29:c include '/usr/include/starbase.f2.h'
30:c *INCLUDE 'errvar'
31:c character*3 fun
32:c character*8 opst,kgt
33:c character*14 profile(50)
34:c character*16 dapfrm
35:c character*80 dummy,outst
36:c integer*4 status
37:c logical flag,duct,us_flag
38:c ir=5
39:c iw=6
40:c call kyinit(ir,iw,15)
41:c twice=.false.
42:c us_flag=.false.
43:c ipro_print=0
44:c
45:c ***** Clear alphanumerics screen and write options list. *****
46:c
47:c 10 call kycrsor(0,0,-1)
48:c write(iw,('1 -- Delete an environmental data file'))
49:c write(iw,('2 -- Add an environmental data file'))
50:c write(iw,('3 -- Edit an environmental data file'))
51:c write(iw,('4 -- Altitude Error Display'))
52:c if(twice) write(iw,('5 -- Display for same profile'))
53:c write(iw,e)
54:c call kyenter('option number (or 'end') ',0,1,,5,,-1,,optn,optst)
55:c if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
56:c write(iw,e)

```

```

57:      ioptn=nint(optim)
58:      if(ioptn.eq.4) twice=.false.
59:      if(option.eq.-1)opst='end'
60:      if((opst.eq.'end').or.(opst.eq.'END')) then
61:        goto 20
62:      else
63:        if(.not.twice) call renew
64:      end if
65: 30    flag=.false.
66:      if((ioptn.eq.1).or.(ioptn.eq.2).or.(ioptn.eq.3)) then
67:        call edit(ioptn,opst,flag,iw)
68:        if((opst.eq.'option').or.(opst.eq.'backup')) goto 10
69:      end if
70:      if(flag) goto 10
71:      if(ioptn.eq.5) then
72:        us_flag=.true.
73:        call putin(rl,ru,irplot,filvar,fun)
74:        goto 15
75:      end if
76: c
77: c ***** List existing environmental data files. *****
78: c
79:      call lsfiles('.prof',profile,ifil)
80:      if(ifil.eq.0) then
81:        write(iw, '(a1)' )char(7)
82:        call kysntent("** No data files exist. Press 'RETURN' to create
83: + new file. **",-1,'*', ' ',outst,opst)
84:        if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
85:        ioptn=2
86:        goto 30
87:      end if
88:      write(iw,*)
89: c
90: c ***** Begin writing prompt strings for user input. *****
91: c
92: 40    call kynter("number of environmental data file ",0,1.,real(ifil),
93: +0.,rpick,opst)
94:      if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
95:      ipick=nint(rpick)
96:      name=profile(ipick)
97:      write(iw,*)
98: 50    call kynter("maximum height for display",1,1.,50000.,0.,hmax,
99: +opst)
100:      if(opst.eq.'backup') goto 40
101:      if(opst.eq.'option') goto 10
102:      write(iw,*)
103: 60    call kynter("maximum range for display",2,20.,1000.,0.,rmax,
104: +opst)
105:      if(opst.eq.'backup') goto 50
106:      if(opst.eq.'option') goto 10
107:      write(iw,*)
108: 70    call kynter("antenna height",1,1.,50000.,0.,ht,opst)
109:      if(opst.eq.'backup') goto 60
110:      if(opst.eq.'option') goto 10
111:      write(iw,*)
112: 80    call kynter("lower elevation angle in erad",0,-100.,100.,101.,rl,

```

```

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.,101.,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",0,1.,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 kystent("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",1,1.,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",0,1.,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 readda(ipick,profile,hun,reun)
146: 15    call kyread(7,f7)
147:c
148:c ***** Call INITIAL to set up arrays and variables used in
149:c the ray tracing subroutines. *****
150:c
151:c      call initial(rl,ru,irplot,duct,forbid)
152:c
153:c ***** Call GRAPH to initialize graphics screen and draw axes
154:c for plotting. *****
155:c
156:      call kyread(6,kgt)
157:      if((ipro_print.eq.2).or.(ipro_print.eq.1)) kgt='on'
158:      if(kgt.eq.'off') then
159:          plot=.false.
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:              ipro_print=nint(pro_print)
166:          end if
167:      end if
168: 25    call graph(rl,ru,irplot)

```

```

169:c
170:c ***** Start of main calculations. *****
171:c
172:c     call actapp(irplot,duct,forbid)
173:c     write(iw,*)
174:c
175:c ***** Begin dump. *****
176:c
177:c     status=gclose(fildes)
178:c     twice=.true.
179:c     if(plot) goto 10
180:c     call kystent("Press 'AUTODUMP-ON' and 'RETURN' for hardcopy",
181:c + -1,'*',',',outst,opst)
182:c     if(opst.eq.'option') goto 10
183:c     call kyread(6,kgt)
184:c     if(kgt.eq.'on') then
185:c         call kyenter("profile printout:  1 - graph ; 2 - numbers",
186:c + 0,1.,2.,1.,pro_print,opst)
187:c         if(opst.eq.'option') goto 10
188:c         ipro_print=nint(pro_print)
189:c         plot=.true.
190:c         goto 25
191:c     else
192:c         goto 10
193:c     end if
194:c 20  continue
195:c     if(us_flag) call system('rm usin')
196:c     call kyterm
197:c     end

```

```

1:c ***** SUBROUTINE ACTAPP *****
2:c
3:c      Purpose:  This begins the main loop of the program.  It checks for
4:c                the transmitter height level and begins the ray-tracing
5:c                subroutines.
6:c
7:c      Glossary:
8:c
9:c          duct    Logical flag indicating if the transmitter height is
10:c                at a break point.
11:c          irays   Number of rays to be used in plotting.
12:c
13:c *****
14:c
15:c      subroutine actapp(irays,duct,forbid)
16:$INCLUDE 'errvar'
17:c      logical duct
18:c
19:c ***** Check what level the transmitter height is at. *****
20:c
21:c      do ix=1,nlvl
22:c          if(((h(ix).le.ht).and.(ht.le.h(ix+1)))) goto 40
23:c      end do
24:c 40      continue
25:c
26:c ***** Begin main loop. *****
27:c
28:c      im=1
29:c      if(plot) then
30:c          call seven(irays,duct,forbid,ix)
31:c      else
32:c          call regular(irays,duct,forbid,ix)
33:c      end if
34:c      return
35:c      end
36:c
37:c ***** SUBROUTINE STAND *****
38:c
39:c      Purpose:  This routine performs a ray trace for a standard atmos-
40:c                phere.  It begins at the transmitter height and traces
41:c                to the specified range given by the current non-standard
42:c                atmosphere being used with the same initial elevation
43:c                angle.
44:c
45:c      Glossary:
46:c          hs      Height traced to at range rp.
47:c          hdiff   Difference in height between the ray traced in a stan-
48:c                dard atmosphere and that traced in a non-standard atmos-
49:c                phere at the same range.
50:c
51:c *****
52:c
53:c      subroutine stand(fixang,rp,hp)
54:$INCLUDE 'errvar'
55:c      rf=rp
56:c      rbef=0.

```

```

57:      hbef=ht
58:      alpha=fixang
59:c
60:c      ***** Begin trace for down-going rays. *****
61:c
62:      if(alpha.lt.0.) then
63:          alphap=alpha+1.18e-4*rf
64:          if(alphap.ge.0.) then
65:              alphap=0.
66:              range=rbef-alpha/1.18e-4
67:              hs=hbef-alphap**2/2.36e-7
68:              alpha=alphap
69:              hbef=hs
70:              rbef=range
71:              goto 10
72:          end if
73:          hs=hbef+(alphap**2-alpha**2)/2.36e-7
74:c
75:c      ***** If the height is negative, then ray is reflected. *****
76:c
77:          if(hs.lt.0.) then
78:              alphap=-SQRT(alpha**2+2.36e-7*(-hbef))
79:              range=rbef+(alphap-alpha)/1.18e-4
80:              alphap=-alphap
81:              alpha=alphap
82:              rbef=range
83:              hbef=0.
84:              goto 10
85:          end if
86:          goto 20
87:      end if
88:c
89:c      ***** Begin trace for up-going rays. *****
90:c
91: 10      if(alpha.ge.0.) then
92:          alphap=alpha+1.18e-4*(rf-rbef)
93:          hs=hbef+(alphap**2-alpha**2)/2.36e-7
94:      end if
95: 20      continue
96:      hdif=ABS(hp-hs)
97:c
98:c      ***** Calling routine to determine color according to error
99:c      increment scale. *****
100:c
101:      rng=0.
102:      hyt=0.
103:c
104:      call interpol(hdif, rp, hp, rng, hyt, hs)
105:      frst=.false.
106:      if(plot) then
107:          call stepfixplt(hdif, hs, rp, hp, rng, hyt)
108:      else
109:          call stepfixscr(hdif, hs, rp, hp, rng, hyt)
110:      end if
111:      return
112:      end

```

```

1:c ***** SUBROUTINE DOWN *****
2:c
3:c      Purpose: This routine performs a ray trace for down-going rays.
4:c
5:c      Glossary:
6:c
7:c          rbf      Range before calculation.
8:c          rp       Range after calculation.
9:c          rinc     Range increment.
10:c         hbef     Height before calculation.
11:c         hp        Height after calculation.
12:c         ij        Height level counter.
13:c         alpha     Angle before calculation.
14:c         alphap    Angle after calculation.
15:c
16:c *****
17:c
18:c $OPTION ONEYRIP
19:c      subroutine down(alpha,hbef,ij,rbef,rp,hp,fixang)
20:c      include '/usr/include/starbase.f1.h'
21:c      include '/usr/include/starbase.f2.h'
22:c $INCLUDE 'errvar'
23:c
24:c      ***** Begin loop to trace from transmitter height to the
25:c      first level or until a minimum has been reached. *****
26:c
27:c      do while((ij.ge.1).and.(alpha.le.0.))
28:c          rp=rbef+rinc
29:c          if(rp.ge.rmax) goto 10
30:c          alphap=alpha+dadh(ij)*(rp-rbef)
31:c
32:c      ***** If the new angle calculated is positive then a minimum
33:c      has been reached. ALPHAP is set to 0. *****
34:c
35:c          if(alphap.ge.0.) then
36:c              alphap=0.
37:c              rp=rbef-alpha/dadh(ij)
38:c              hp=hbef-alpha**2/2.e-3/dadh(ij)
39:c          end if
40:c          hp=hbef+(alphap**2-alpha**2)/2.e-3/dadh(ij)
41:c          if(hp.lt.h(ij)) then
42:c              hp=h(ij)
43:c              rad=alpha**2+2.e-3*dadh(ij)*(hp-hbef)
44:c
45:c      ***** If RAD is negative then a minimum has been reached.
46:c      ALPHAP is set to 0. *****
47:c
48:c          if(rad.le.0.) then
49:c              alphap=0.
50:c              rp=rbef-alpha/dadh(ij)
51:c              hp=hbef-alpha**2/2.e-3/dadh(ij)
52:c          else
53:c              alphap=-SQRT(rad)
54:c              rp=rbef+(alphap-alpha)/dadh(ij)
55:c          end if
56:c          ij=ij-1

```

```
57:         end if
58:c
59:c ***** Once a specific height and range have been calculated
60:c subroutine STAND is called to calculate the height at the same range
61:c RP for a standard atmosphere. *****
62:c
63:         if(rp.ge.rmax) goto 10
64:         call stand(fixang,rp,hp)
65:         rbef=rp
66:         hbef=hp
67:         alphap=alphap
68:         if(alpha.eq.0.) goto 10
69:     end do
70:     if(ij.eq.0.) goto 30
71:c
72:c ***** Trace to exactly the maximum range. *****
73:c
74: 10     if(rp.ge.rmax) then
75:         rp=rmax
76:         alphap=alpha+dmdh(ij)*(rp-rbef)
77:         hp=hbef+(alphap**2-alpha**2)/2.e-3/dmdh(ij)
78:         call stand(fixang,rp,hp)
79:     end if
80: 30     return
81: end
```

```

1:c      ***** SUBROUTINE EDIT *****
2:c
3:c      This subroutine performs all the editing operations of the program.
4:c
5:c      subroutine edit(ioptn,opst,flag,iw)
6:c      character*8 opst,outst
7:c      character*14 filena(50),dumname
8:c      character*80 dum
9:c      logical flag
10:c     flag=.true.
11:c
12:c     ***** Specified files are deleted or put into the vi
13:c     editor for editing. *****
14:c
15:c     if((ioptn.eq.1).or.(ioptn.eq.3)) then
16:c         call lsfiles('.prof',filena,ifil)
17:c         if(ifil.eq.0) then
18:c             write(iw,'(a1)')char(7)
19:c             call kystent("*** No data files exist. Press 'RETURN' to cr
20:c +eate new file. ***",-1,'*',',',outst,opst)
21:c             if((opst.eq.'backup').or.(opst.eq.'option')) goto 10
22:c             ioptn=2
23:c             goto 20
24:c         end if
25:c         write(iw,*)
26:c         call kyenter("file number",0,1.,real(ifil),0.,rnum,opst)
27:c         if((opst.eq.'option').or.(opst.eq.'backup')) goto 10
28:c         inum=nint(rnum)
29:c     end if
30:c 20    if(ifil.eq.0) ifil=1
31:c
32:c     ***** For adding an environmental file the user is put into
33:c     the vi editor. *****
34:c
35:c     if(ioptn.eq.2) then
36:c         call system('cp .dprof dprof')
37:c         call system('vi dprof')
38:c         open(3,FILE='dprof')
39:c         read(3,'(3(a80/),a80)')(dum,i=1,4)
40:c         read(3,'(a14)') filena(ifil)
41:c         close(3)
42:c         call system('mv dprof .prof///filena(ifil)//char(0)')
43:c     else if(ioptn.eq.1) then
44:c         call system('rm .prof///filena(inum)//char(0)')
45:c     else
46:c         call system('vi .prof///filena(inum)//char(0)')
47:c         open(3,FILE='.prof///filena(inum)')
48:c         read(3,'(3(a80/),a80)')(dum,i=1,4)
49:c         read(3,'(a14)') dumname
50:c         if(dumname.ne.filena(inum)) call system('mv .prof///filena
51:c + (inum)//' .prof///dumname//char(0)')
52:c         close(3)
53:c     end if
54:c 10    return
55:c     end

```

```

1:c ***** SUBROUTINE INITIAL *****
2:c
3:c Purpose: This subroutine calculates all the necessary constants
4:c and arrays for use in actual ray trace calculations.
5:c
6:c Glossary:
7:c     rplot      Real value of the number of rays to be plotted.
8:c     irplot     Integer value of the number of rays to be plotted.
9:c     forbid     Angular region of unallowable ray traces.
10:c     duct       Flag indicating if the transmitter height is at
11:c               a breakpoint.
12:c     check      Flag indicating if hmax is equal to any of the
13:c               heights in the profile.
14:c     rl         Lower elevation angle in rad.
15:c     rfirst     Lower elevation angle in rad.
16:c     ru         Upper elevation angle in rad.
17:c     rlast     Upper elevation angle in rad.
18:c     ainc       Incremental angle value for alpha array.
19:c     alpha      Array containing all initial launch angles.
20:c     rinc       Range increment.
21:c     rmetperdot Meters per dot on screen.
22:c
23:c *****
24:c
25:   subroutine initial(rl,ru,irplot,duct,forbid)
26: *INCLUDE 'errvar'
27:   logical duct,check
28:c
29:c ***** Convert lower and upper angles to radians and store in
30:c an array in incremental values corresponding to the number of rays t
31:c by plotted. *****
32:c
33:   rfirst=rl*.15708
34:   rlast=ru*.15708
35:   ainc=(rlast-rfirst)/(irplot-1)
36:   angle(1)=rfirst
37:   do i=2,irplot
38:     angle(i)=angle(i-1)+ainc
39:   end do
40:   angle(irplot)=rlast
41:   rinc=rmax/50.
42:   sor=.025*rmax
43:c
44:c ***** Convert height and index array to meters and m-units
45:c if necessary. *****
46:c
47:   do i=1,ilvl
48:     if(hun.eq.'F') then
49:       h(i)=orh(i)/3.280839
50:     else
51:       h(i)=orh(i)
52:     end if
53:     if(reun.eq.'N') then
54:       m(i)=orm(i)/6.371
55:     else
56:       m(i)=orm(i)

```

```

57:         end if
58:     end do
59:     nlvl=ilvl
60:c
61:c ***** If the first height level is not 0. then an extrapola-
62:c tion is done to find the m-unit value at the surface. *****
63:c
64:     if(h(1).ne.0.) then
65:         nlvl=nlvl+1
66:         surf=h(1)/(h(2)-h(1))*(m(1)-m(2))
67:         m(nlvl)=m(1)+surf
68:     end if
69:c
70:c ***** Each height level is checked to see if it matches the
71:c maximum height. If not, then the maximum height is included in the
72:c array and the m-unit value at that height is found. *****
73:c
74:     check=.false.
75:     do i=1,ilvl
76:         if((h(i).le.hmax+1.e-5).and.(h(i).ge.hmax-1.e-5)) check=.true.
77:     end do
78:     if(.not.check) then
79:         nlvl=nlvl+1
80:         h(nlvl)=hmax
81:     end if
82:c
83:c ***** The array is sorted. *****
84:c
85:     do 10 i=1,nlvl-1
86:         do 20 j=i+1,nlvl
87:             if(h(j).gt.h(i)) goto 20
88:             dum=h(i)
89:             h(i)=h(j)
90:             h(j)=dum
91:             dum=m(i)
92:             m(i)=m(j)
93:             m(j)=dum
94: 20         end do
95: 10     end do
96:c
97:c ***** If HMAX is between two height levels then the m-unit
98:c value is found at HMAX. *****
99:c
100:    if(hmax.ne.h(nlvl)) then
101:        do j=1,nlvl
102:            if(h(j).eq.hmax) then
103:                bet=(h(j)-h(j-1))/(h(j+1)-h(j-1))*(m(j+1)-m(j-1))
104:                m(j)=m(j-1)+bet
105:            end if
106:        end do
107:c
108:c ***** The gradient at each level is calculated and stored. *****
109:c
110:    else
111:        rint=(h(nlvl)-h(nlvl-1))/(h(nlvl-1)-h(nlvl-2))*(m(nlvl-1)-
112:        * m(nlvl-2))

```

```
113:      m(nlvl)=m(nlvl-1)+rint
114:    end if
115:    do i=1,nlvl-1
116:      dmdh(i)=(m(i+1)-m(i))/(h(i+1)-h(i))*1.e-3
117:      if (ABS(dmdh(i)).lt.1.e-6) dmdh(i)=1.e-6
118:    end do
119:    rmetperdot=rmax/390.
120:c
121:c ***** The transmitter height is checked to see if it is
122:c at a breakpoint. If so, the unallowable angular range is
123:c calculated. *****
124:c
125:    forbid=0.
126:    duct=.false.
127:    do i=1,nlvl
128:      if ((h(i).le.ht+1.e-6).and.(h(i).ge.ht-1.e-6)) then
129:        if ((dmdh(i-1).gt.0.).and.(dmdh(i).lt.0.)) forbid=
130:      +      SQRT(-2.e-3*dmdh(i)*rmetperdot)
131:      end if
132:    end do
133:    if (forbid.ne.0.) duct=.true.
134:    return
135:  end
```

```

1:c
2:c ***** SUBROUTINE INTERPOL *****
3:c
4:c Purpose: This subroutine checks if a transition from one color
5:c           to the next has been reached.
6:c
7:c *****
8:c
9:   subroutine interpol(dif_2,rng_2,hp_2,rng,hyt,hs_2)
10:$INCLUDE 'errvar'
11:   save dif_1,rng_1,hp_1,hs_1
12:   if(frst) then
13:     dif_1=dif_2
14:     rng_1=rng_2
15:     hp_1=hp_2
16:     hs_1=hs_2
17:     goto 10
18:   end if
19:   if((erropt.eq.'a').or.(erropt.eq.'A')) then
20:     if(((dif_1.le.aber).and.(dif_2.gt.aber)).or.((dif_2.le.aber)
21: + .and.(dif_1.gt.aber))) then
22:       call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,aber)
23:     else if(((dif_1.le.2.*aber).and.(dif_2.gt.2.*aber)).or.
24: + ((dif_2.le.2.*aber).and.(dif_1.gt.2.*aber))) then
25:       call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,2.*aber)
26:     else if(((dif_1.le.3.*aber).and.(dif_2.gt.3.*aber)).or.
27: + ((dif_2.le.3.*aber).and.(dif_1.gt.3.*aber))) then
28:       call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,3.*aber)
29:     else if(((dif_1.le.4.*aber).and.(dif_2.gt.4.*aber)).or.
30: + ((dif_2.le.4.*aber).and.(dif_1.gt.4.*aber))) then
31:       call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,4.*aber)
32:     else if(((dif_1.le.5.*aber).and.(dif_2.gt.5.*aber)).or.
33: + ((dif_2.le.5.*aber).and.(dif_1.gt.5.*aber))) then
34:       call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,5.*aber)
35:     else if(((dif_1.le.6.*aber).and.(dif_2.gt.6.*aber)).or.
36: + ((dif_2.le.6.*aber).and.(dif_1.gt.6.*aber))) then
37:       call int(dif_1,dif_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,6.*aber)
38:   end if
39:   else if((erropt.eq.'p').or.(erropt.eq.'P')) then
40:     if((hs_1.eq.0.).or.(hs_2.eq.0.)) then
41:       hs_1=1.
42:       hs_2=1.
43:     end if
44:     perc_1=(dif_1/hs_1)*100.
45:     perc_2=(dif_2/hs_2)*100.
46:     if(((perc_1.le.per).and.(perc_2.gt.per)).or.((perc_2.le.per)
47: + .and.(perc_1.gt.per))) then
48:       call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,per)
49:     else if(((perc_1.le.2.*per).and.(perc_2.gt.2.*per)).or.
50: + ((perc_2.le.2.*per).and.(perc_1.gt.2.*per))) then
51:       call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,2.*per)
52:     else if(((perc_1.le.3.*per).and.(perc_2.gt.3.*per)).or.
53: + ((perc_2.le.3.*per).and.(perc_1.gt.3.*per))) then
54:       call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,3.*per)
55:     else if(((perc_1.le.4.*per).and.(perc_2.gt.4.*per)).or.
56: + ((perc_2.le.4.*per).and.(perc_1.gt.4.*per))) then

```

```

57:      call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,4.*per)
58:      else if(((perc_1.le.5.*per).and.(perc_2.gt.5.*per)).or.
59: + ((perc_2.le.5.*per).and.(perc_1.gt.5.*per))) then
60:      call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,5.*per)
61:      else if(((perc_1.le.6.*per).and.(perc_2.gt.6.*per)).or.
62: + ((perc_2.le.6.*per).and.(perc_1.gt.6.*per))) then
63:      call int(perc_1,perc_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,6.*per)
64:      end if
65:    end if
66:    dif_1=dif_2
67:    rng_1=rng_2
68:    hp_1=hp_2
69:    hs_1=hs_2
70:  10  return
71:  end
72: c
73: c ***** SUBROUTINE INT *****
74: c
75: c Purpose: This subroutine interpolates the transition from one
76: c          height error increment (color) to the next.
77: c
78: c *****
79: c
80: c subroutine int(dum_1,dum_2,rng_1,rng_2,hp_1,hp_2,rng,hyt,fxdif)
81: c rin=(fxdif-dum_1)/(dum_2-dum_1)*(rng_2-rng_1)
82: c rng=rng_1+rin
83: c hin=(rng-rng_1)/(rng_2-rng_1)*(hp_2-hp_1)
84: c hyt=hp_1+hin
85: c return
86: c end

```

```

1:c      ***** Subroutine LSFILES produces a two-column list, preceded
2:c      by a number, of all data files stored in directory 'DIR'.
3:c      The maximum number of data files 46. *****
4:c
5:c      *****
6:c
7:c      Glossary:
8:c          ifil    Counter for number of files in directory. Will be
9:c                  zero if no files in that directory.
10:c          ire     One-half of ifil to print out on screen a two-column
11:c                  list of filenames
12:c          name     46 element array - stores filenames
13:c
14:c      *****
15:c
16:      subroutine lsfiles(dir,name,ifil)
17:      character*14 name(46)
18:      character*(*) dir
19:      call system('ls '//dir//')@datafiles@'//char(0))
20:      open(1,FILE='@datafiles@')
21:      ifil=0
22:      do j=1,46
23:          read(1,'(a14)',END=10)name(j)
24:          ifil=ifil+1
25:      end do
26: 10      continue
27:      close(1)
28:      ire=nint(ifil/2.)
29:      do j=1,ire
30:          if((mod(ifil,2).ne.0).and.(j.eq.ire)) then
31:              write(6,'(i2,3x,a14)')j,name(j)
32:          else
33:              write(6,'(i2,3x,a14,10x,i2,3x,a14)')j,name(j),j+ire,
34:  *          name(j+ire)
35:          end if
36:      end do
37:      call system('re @datafiles@')
38:      return
39:      end

```

```

1:c
2:c ***** SUBROUTINE PUTIN *****
3:c
4:c Purpose: This subroutine puts in all user information into a file
5:c           for later retrieval. This allows the user to run a
6:c           second display for the same profile (with minor
7:c           changes if so desired) without going
8:c           through each prompt again.
9:c
10:c *****
11:c
12: subroutine putin(rl,ru,irplot,filvar,fun)
13: include 'errvar'
14: character*3 fun
15: open(2,FILE='usin')
16: if((erropt.eq.'a').or.(erropt.eq.'A')) then
17:     filvar=aber
18: else
19:     filvar=per
20: end if
21: call kyread(7,fun)
22: write(2,'(25x,"** USER INPUT PARAMETERS **")')
23: write(2,*)
24: write(2,'(5x,"You are now in ''vi''. If there are any changes",
25: * " desired use ''vi'' commands")')
26: write(2,'(5x,"for editing. Exit by typing ''ZZ''.")')
27: write(2,*)
28: write(2,*)
29: if(fun.eq.'fps') then
30:     write(2,'(f7.0,10x,"| Maximum height for display in feet")')
31:     * hmax=3.280839
32:     write(2,'(f4.0,13x,"| Maximum range for display in nm")')
33:     * rmax=1.85318
34:     write(2,'(f6.0,11x,"| Antenna height in feet")')ht=3.280839
35: else
36:     write(2,'(f7.0,10x,"| Maximum height for display in meters")')
37:     * hmax
38:     write(2,'(f4.0,13x,"| Maximum range for display in km")')
39:     * rmax
40:     write(2,'(f6.0,13x,"| Antenna height in meters")')ht
41: end if
42: write(2,'(f5.0,12x,"| Lower elevation angle (arad)")')rl
43: write(2,'(f5.0,12x,"| Upper elevation angle (arad)")')ru
44: write(2,'(i4,13x,"| Number of rays to be plotted (integer)")')
45: *irplot
46: write(2,'(a1,16x,"| A or P -- Absolute or Percent error")')erropt
47: write(2,'(i1,16x,"| 1 - graph ; 2 - numbers : for profile printout
48: * (integer)")')ipro_print
49: write(2,*)
50: write(2,'(10x,"** Fill in only the line that applies **")')
51: write(2,*)
52: if((erropt.eq.'a').or.(erropt.eq.'A')) then
53:     if(fun.eq.'fps') then
54:         write(2,'(f5.0,12x,"| Error increment in feet")')
55:         * filvar=3.280839
56:     else

```

```

57:         write(2,'(f5.0,12x,"| Error increment in meters")')filvar
58:     end if
59:     write(2,'(17x,"| Percent error increment")')
60:     else if((erropt.eq.'p').or.(erropt.eq.'P')) then
61:         write(2,'(17x,"| Error increment")')
62:         write(2,'(f3.0,14x,"| Percent error increment")')filvar
63:     end if
64:     close(2)
65:     call rdffl(rl,ru,irplot,filvar,fun)
66:     return
67: end
68:c
69:c ***** SUBROUTINE RDFFL *****
70:c
71:c Purpose: This subroutine reads all user input parameters from a
72:c          file.
73:c
74:c *****
75:c
76: subroutine rdffl(rl,ru,irplot,filvar,fun)
77: include 'errvar'
78: character*3 fun
79: character*80 dum
80: call system('vi usin')
81: open(2,FILE = 'usin')
82: read(2,'(5(a80/),a80)')(dum,i=1,6)
83: read(2,'((f7.0/),(f4.0/),f6.0)')hmax,rmax,ht
84: read(2,'(2(f5.0/),(i4/),(a1/),i1)')rl,ru,irplot,erropt,
85: +ipro_print
86: if(fun.eq.'fps')then
87:     hmax=hmax/3.280839
88:     rmax=rmax*1.85318
89:     ht=ht/3.280839
90: end if
91: read(2,'(2(a80/),a80)')(dum,i=1,3)
92: if((erropt.eq.'a').or.(erropt.eq.'A')) then
93:     read(2,'(f5.0)')filvar
94:     aber=filvar
95:     if(fun.eq.'fps') aber=aber/3.280839
96: else if((erropt.eq.'p').or.(erropt.eq.'P')) then
97:     read(2,'((a80/),f5.0)')dum,filvar
98:     per=filvar
99: end if
100: close(2)
101: do j=1,35
102:     h(j)=0.
103:     m(j)=0.
104:     dmdh(j)=0.
105: end do
106: do j=1,300
107:     angle(j)=0.
108: end do
109: return
110: end

```

```

1:c ***** SUBROUTINE READDA *****
2:c
3:c      Purpose:  Subroutine READDA reads environmental data from data
4:c                files the user specifies.
5:c
6:c      Glossary:
7:c
8:c          profile      Character array containing environmental data
9:c                      filenames.
10:c          ipick        Counter for "profile" array indicating number
11:c                      of environmental data file.
12:c
13:c *****
14:c
15:c      subroutine readda(ipick,profile)
16:c $INCLUDE 'errvar'
17:c      character*14 profile(50)
18:c      character*80 dum
19:c
20:c ***** Open data file; read data *****
21:c
22:c      open(10,FILE='.prof//'/profile(ipick))
23:c      read(10,'(3(a80/),a80)')(dum,i=1,4)
24:c      read(10,'((a14/),(a1/),a1)')profile(ipick),hun,neun
25:c      read(10,'(4(a80/),a80)')(dum,i=1,5)
26:c      ilvl=0
27:c
28:c ***** Read H and H arrays. *****
29:c
30:c      do i=1,33
31:c          if(neun.eq.'H') then
32:c              read(10,'(f7.1,5x,f6.1)',END=10)orh(i),ora(i)
33:c          else
34:c              read(10,'(f7.1,18x,f6.1)',END=10)orh(i),ora(i)
35:c          end if
36:c          ilvl=ilvl+1
37:c      end do
38:c 10  continue
39:c      close(10)
40:c      return
41:c      end

```

```

1:c ***** SUBROUTINE REGULAR *****
2:c
3:c      Purpose:  This begins the main calculations of the program.  It
4:c                does not allow rays to be traced within an angular limit
5:c                if the transmitter height is at a break point.
6:c
7:c      Glossary:
8:c
9:c          duct    Logical flag indicating if the transmitter height is
10:c                at a break point.
11:c          irplot  Number of rays to be plotted.
12:c          alpha   Beginning angle.
13:c          fixang  Initial elevation angle currently being used.
14:c          ij      Counter.
15:c          hbef    Beginning height.
16:c          rbef    Beginning range.
17:c          forbid  Rays are not traced in this angular range.
18:c
19:c *****
20:c
21:c      subroutine regular(irays,duct,forbid,ix)
22:c      include '/usr/include/starbase.f2.h'
23:c      include '/usr/include/starbase.f1.h'
24:c $INCLUDE 'errvar'
25:c      logical duct
26:c
27:c ***** Begin main loop. *****
28:c
29:c      do ik=1,irays
30:c          frst=.true.
31:c          if(plot) frstime=.true.
32:c          if(im.ne.1) then
33:c              mvfl=.false.
34:c          else
35:c              if(plot) then
36:c                  mvfl=.true.
37:c              else
38:c                  if(f7.eq.'fps') then
39:c                      call move2d(fildes,0.,ht+3.280839)
40:c                  else
41:c                      call move2d(fildes,0.,ht)
42:c                  end if
43:c              end if
44:c          end if
45:c
46:c ***** Initialize variables for ray tracing. *****
47:c
48:c          if(plot) call start(ik,ix)
49:c          alpha=angle(ik)
50:c          fixang=angle(ik)
51:c          ij=ix
52:c          hbef=ht
53:c          rbef=0.
54:c
55:c ***** Check if transmitter is at a breakpoint.  If so, skip
56:c angles within forbidden range. *****

```

```

57:c
58:      if(duct) then
59:          if(ABS(alpha).le.forbid) then
60:              rp=rmax
61:              call line_color_index(fildes,1)
62:              if(f7.eq.'fps') then
63:                  call draw2d(fildes,rp*.5396117,ht*3.280839)
64:              else
65:                  call draw2d(fildes,rp,ht)
66:              end if
67:              goto 50
68:          end if
69:      end if
70:c
71:c ***** If initial elevation angle is negative then call routine
72:c for down-going rays. If it is positive, then call routine for up-
73:c going rays. If it is 0 then call routine to check on the value of
74:c the gradient. *****
75:c
76: 30      if(alpha.eq.0.) then
77:          call zero(alpha,hbef,ij,rbef,rp,hp,fixang)
78:      else if(alpha.gt.0.) then
79:          call up(alpha,hbef,ij,rbef,rp,hp,fixang)
80:      else
81:          call down(alpha,hbef,ij,rbef,rp,hp,fixang)
82:      end if
83:      if(hp.eq.h(1)) goto 50
84:      if((rp.lt.rmax).and.(hp.lt.hmax)) goto 30
85: 50      continue
86:      end do
87:      return
88:      end
89:c
90:c ***** SUBROUTINE SEVEN *****
91:c
92:c Purpose: This routine performs the ray-tracing procedure 7 times
93:c (once for each color) for ease in plotting onto the
94:c graphics plotter.
95:c
96:c *****
97:c
98:      subroutine seven(irays,duct,forbid,ix)
99: $INCLUDE 'errvar'
100:      logical duct
101:      do ia=1,7
102:          mvfl=.true.
103:          call regular(irays,duct,forbid,ix)
104:      end do
105:      return
106:      end

```

```

1:c ***** SUBROUTINE STEPFIXSCR *****
2:c
3:c      Purpose:  STEPFIXSCR defines the color scale for plotting height
4:c                and range onto the screen in the correct color.
5:c
6:c      Glossary:
7:c
8:c                percerr Percentage error between height of non-standard
9:c                      atmosphere ray and height of standard atmosphere
10:c                     ray.
11:c                rng      Interpolated range between color transitions.
12:c                hyt      Interpolated height between color transitions.
13:c
14:c *****
15:c
16:c      subroutine stepfixscr(hdif,hs,rp,hp,rng,hyt)
17:c      include '/usr/include/starbase.f1.h'
18:c      include '/usr/include/starbase.f2.h'
19:c $INCLUDE 'errvar'
20:c      if(rng.ne.0.) then
21:c          if(f7.eq.'fps') then
22:c              call draw2d(fildes,rng/1.85318,hyt*3.280839)
23:c          else
24:c              call draw2d(fildes,rng,hyt)
25:c          end if
26:c      end if
27:c
28:c ***** Define color scale for absolute height error. *****
29:c
30:c      if((erropt.eq.'a').or.(erropt.eq.'A')) then
31:c          if(hdif.le.aber) then
32:c              call line_color_index(fildes,1)
33:c          else if((hdif.gt.aber).and.(hdif.le.2.*aber)) then
34:c              call line_color_index(fildes,3)
35:c          else if((hdif.gt.2.*aber).and.(hdif.le.3.*aber)) then
36:c              call line_color_index(fildes,4)
37:c          else if((hdif.gt.3.*aber).and.(hdif.le.4.*aber)) then
38:c              call line_color_index(fildes,5)
39:c          else if((hdif.gt.4.*aber).and.(hdif.le.5.*aber)) then
40:c              call line_color_index(fildes,6)
41:c          else if((hdif.gt.5.*aber).and.(hdif.le.6.*aber)) then
42:c              call line_color_index(fildes,7)
43:c          else if(hdif.gt.6.*aber) then
44:c              call line_color_index(fildes,2)
45:c          end if
46:c
47:c ***** Define color scale for percentage height error. *****
48:c
49:c      else if((erropt.eq.'p').or.(erropt.eq.'P')) then
50:c          if(hs.eq.1.) then
51:c              percerr=200.
52:c          else
53:c              percerr=(hdif/hs)*100.
54:c          end if
55:c          if(percerr.le.per) then
56:c              call line_color_index(fildes,1)

```

```

57:         else if((percerr.gt.per).and.(percerr.le.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) then
68:             call line_color_index(fildes,2)
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
78:c
79:c ***** SUBROUTINE RENEW *****
80:c
81:c Purpose: RENEW re-initializes arrays used in ray tracing.
82:c
83:c *****
84:c
85:     subroutine renew
86: $INCLUDE 'errvar'
87:     do j=1,35
88:         dadh(j)=0.
89:         h(j)=0.
90:         m(j)=0.
91:     end do
92:     do j=1,33
93:         orh(j)=0.
94:         orm(j)=0.
95:     end do
96:     do j=1,300
97:         angle(j)=0.
98:     end do
99:     return
100: end

```

```

1:c ***** SUBROUTINE UP *****
2:c
3:c     Purpose: This subroutine performs a ray trace for up-going rays.
4:c
5:c *****
6:c
7:c     subroutine up(alpha,hbef,ij,rbef,rp,hp,fixang)
8:c     include '/usr/include/starbase.f1.h'
9:c     include '/usr/include/starbase.f2.h'
10:c $INCLUDE 'errvar'
11:c
12:c ***** Begin at transmitter height and calculate until the ray
13:c reaches a maximum or maximum height. *****
14:c
15:c     do while((ij.lt.nlvl).and.(alpha.ge.0.))
16:c         rp=rbe+rninc
17:c         if(rp.ge.rmax) goto 20
18:c         alphap=alpha+dadh(ij)*(rp-rbef)
19:c
20:c ***** If the ray reaches a maximum then set ALPHAP to 0. and
21:c calculate height and range. Call DOWN. *****
22:c
23:c         if(alphap.le.0.) then
24:c             alphap=0.
25:c             rp=rbe-alpha/dadh(ij)
26:c             hp=hbe-alpha**2/2.e-3/dadh(ij)
27:c         end if
28:c         hp=hbe+(alphap**2-alpha**2)/2.e-3/dadh(ij)
29:c         if(hp.gt.h(ij+1)) then
30:c             hp=h(ij+1)
31:c             rad=alpha**2+2.e-3*dadh(ij)*(hp-hbe)
32:c
33:c ***** If the ray reaches a maximum then set ALPHAP to 0. and
34:c calculate height and range. Call DOWN. *****
35:c
36:c             if(rad.le.0.) then
37:c                 alphap=0.
38:c                 rp=rbe-alpha/dadh(ij)
39:c                 hp=hbe-alpha**2/2.e-3/dadh(ij)
40:c             else
41:c                 alphap=SQRT(rad)
42:c                 rp=rbe+(alphap-alpha)/dadh(ij)
43:c             end if
44:c             ij=ij+1
45:c         end if
46:c         if((rp.ge.rmax).or.(hp.ge.hmax)) goto 10
47:c
48:c ***** Call STAND to determine height of ray for standard
49:c atmosphere at same range. *****
50:c
51:c         call stand(fixang,rp,hp)
52:c         hbef=hp
53:c         rbef=rp
54:c         alpha=alphap
55:c         if(alpha.le.0.) goto 10
56:c     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=SQRT(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

```

```

1:c ***** SUBROUTINE ZERO *****
2:c
3:c      Purpose:  This subroutine determines which subroutine will be
4:c                called when a ray reaches a maximum or minimum.  If
5:c                a ray reaches a maximum, then depending on the value
6:c                of the gradient, DOWN is called.  Likewise, if a ray
7:c                reaches a minimum, UP is called.
8:c
9:c *****
10:c
11:      subroutine zero(alpha,hbef,ij,rbef,rp,hp,fixang)
12:$INCLUDE 'errvar'
13:      if(dmdh(ij).lt.0.) then
14:        call down(alpha,hbef,ij,rbef,rp,hp,fixang)
15:      else
16:        call up(alpha,hbef,ij,rbef,rp,hp,fixang)
17:      end if
18:      return
19:      end

```