Technical Report 1576
February 1993

# A Ray Trace Model for Propagation Loss

C. P. Hattan

NRaD

Technical Report 1576
February 1993

# A Ray Trace Model for Propagation Loss

C. P. Hattan

## ADMINISTRATIVE INFORMATION

Personnel at the Tropospheric Branch, Code 543, at the Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, California 92152-5001, conducted this study. This study was funded by the Naval Research Laboratory Detachment, Stennis Space Center, Mississippi 39529-5000, under program element 0602435N and accession number DN302214.

Released by
R. A. Paulus, Head
Tropospheric Branch

Under authority of
J. H. Richter, Head
Ocean and Atmospheric
Sciences Division

DM

## EXECUTIVE SUMMARY

### OBJECTIVES

This study was conducted to determine the feasibility of replacing the optical-region propagation model currently assessing the effect of lower atmosphere refraction on electromagnetic (EM) systems performance within the Engineer's Refractive Effects Prediction System (EREPS) with a complete ray trace model. Although the EREPS optical-region propagation model works efficiently by assuming the existence of a single-layer atmosphere, it cannot account for all measured propagation effects. The ray trace model could avoid the optical-region propagation model's deficiencies because it would trace ray paths through the true atmosphere.

### RESULTS

The ray trace model uses a full ray trace solution to describe the optical interference region where two-path coherent interference between direct and sea-reflected rays dominates electromagnetic propagation. The ray trace model needs a multisegmented modified-refractivity profile to describe the propagation environment. It is implemented in a Microsoft QuickBASIC program intended for IBM-compatible personal computers. The ray trace model can determine propagation loss under certain conditions, but it resolves ray paths with increasing difficulty when the EM system frequency rises, and it produces large divergence factors.

### RECOMMENDATIONS

Since attempts to eliminate fluctuations in the ray trace divergence factors have failed, the study's results argue that the ray trace model is not suitable for EREPS.

## CONTENTS

# INTRODUCTION

This report describes a ray trace model that H.V. Hitney[1] proposed be used as a possible replacement for the optical-region propagation model currently in the Engineer's Refractive Effects Prediction System (EREPS) (Patterson et al., 1990). EREPS is a software package for personal computers. It allows engineers to assess the effects of lower atmosphere refraction on electromagnetic (EM) systems performance. The ray trace model has also been recently incorporated into the Radio Physical Optics (RPO) propagation model described by Hitney (1992).

The simplest case of EM propagation is the transmission of energy in free space between a transmitter and a receiver. Free space is defined as a region whose properties are isotropic, homogeneous, and loss-free—i.e., away from the influences of the earth's atmosphere. In free space, the EM wave front spreads uniformly from the transmitter in all directions, and the energy is distributed over an ever enlarging surface. Thus, the energy level along any one ray decreases inversely with the square of the sphere's radius. This decrease is called the free-space path loss. The free-space path loss for isotropic antennas, expressed in terms of frequency, is

$$L_{fs} = 32.44 + 20\log_{10}(r) + 20\log_{10}(f) \tag{1}$$

for $r$ in kilometers (km) and $f$ in megahertz (MHz).

If nonisotropic antenna radiation patterns are considered within the loss calculations, the loss is called propagation loss rather than path loss. The propagation loss can be described with the aid of the pattern propagation factor, which is the ratio of the actual field strength at a point in space to the field strength that would exist at the same range under free-space conditions with the maximum beam of the transmitter directed toward the point in question. For simplicity, *propagation factor* is used throughout this report to refer to the pattern propagation factor. Thus, the effects of the transmitter antenna pattern are implied in all calculations. Symbolically, the propagation factor, $F$, is given by

$$F = \left|\frac{E}{E_0}\right| \tag{2}$$

where $E_0$ is the magnitude of the electric field under free-space conditions and $E$ is the magnitude of the actual field at that same point.

The propagation factor is a desirable quantity because it is an identifiable parameter in most radar-detection-range equations. It contains the information necessary to account for such effects as sea-surface reflection, atmospheric refraction, scattering from atmospheric inhomogeneities, and diffraction from the bulge of the earth's surface. Thus, if we know the functional form of $F$, then we can determine the

[1] Private communication, September 1989.

propagation loss at any point, since the calculation of the free-space field is quite simple. The propagation loss (in decibels [dB]), including antenna patterns, is equivalent to

$$L = L_{fs} - 20\log_{10}(F) \tag{3}$$

Three regions require different methods to obtain signal strength (or, equivalently, propagation factor or loss) as a function of range. The first region is the optical interference, or optical, region, and it roughly extends from the transmitter to the radio horizon. Within the optical region, propagation is dominated by two-path coherent interference between direct and surface-reflected waves. The other distinct region is the diffraction/troposcatter region, which begins slightly beyond the radio horizon. A third region, the intermediate region, lies between the optical and diffraction regions. The propagation factor in this region is obtained by a linear interpolation between the $F$ values in the optical and diffraction regions.

The current EREPS optical-region propagation model assumes a single-layer atmosphere. The assumption of a single-gradient atmosphere is somewhat restrictive, since nature does not always provide such a simple propagation medium. The single-layer model treats refraction by assuming that the refractive bending of the EM rays can be accounted for by using an effective earth radius that is different—usually larger—from the true earth radius. Ray paths over such an oversized earth would then appear as straight lines rather than curved paths. The two-path interference calculation is thus reduced to a geometry problem, which, in the case of the present EREPS model, is computationally efficient.

While this earth-flattening procedure is useful, this method does not thoroughly account for some measured propagation effects. A chief deficiency of the effective earth radius model is that propagation loss values near the optical region's lower angular limit are too low for the higher frequencies (above 3 gigahertz [GHz]) in the presence of strong evaporation ducts. In addition, the location of optical interference maxima (peaks) and minima (nulls) is not usually correct near the lower angular limit for this same high-frequency, evaporation duct case. A complete ray trace model could correct these deficiencies because the actual paths for each ray would be traced through the true atmosphere rather than an atmosphere idealized by a single-gradient average parameter.

## RAY TRACE MODEL

The ray trace model given here requires a piecewise-linear modified refractivity profile. The modified refractivity, or $M$-unit, profile is constructed in such a manner that the $M$-unit value at the surface and a zero-meter (m) height are the first elements in the profile arrays. That is, $z_0 = 0.0$ and $M_0 = M(z_0)$. The remainder of the $M$-unit profile is specified by height and $M$-unit elements $z_i$ and $M_i$, where $z_i$ is the height in m and $M_i$ is the corresponding modified refractivity at the top of the $i$th linear segment in $M$ units. Two further requirements are that $z_{i+1} > z_i$ and $M_{i+1} \neq M_i$. If the transmitter height, $z_t$, does not equal some $z_i$, then the $z_i$, $M_i$ profile is redefined to include it by using linear interpolation to calculate the corresponding $M$-unit value. A third array, which contains the gradient between adjacent layers, can be constructed from these two arrays. The general definition for this array is

$$p_i = 10^{-6}\left(\frac{M_{i+1} - M_i}{z_{i+1} - z_i}\right) \tag{4}$$

Negative values of $p_i$ indicate trapping layers. Another useful array, $q_i$, is also defined because it minimizes the number of mathematical operations performed in the ray trace equations. This array is given by

$$q_i = 2(M_{i+1} - M_i)10^{-6} \tag{5}$$

## RAY TRACE EQUATIONS

The general ray trace equations required to calculate the spatial position of a ray that is based on some initial launch angle, $a$, are divided into two categories: upgoing and downgoing rays. The equations given are the basic "building block" equations describing the ray trajectory and position within a layer. A complete ray trace consists of a series of tests that track the position of the ray as it sequentially enters and exits each layer. The equations can be used to trace a ray to a specified range or height, provided an initial launch angle of $a$ is given. Heights and ranges are in m and all angles are in radians relative to the local horizontal. A (upgoing) ray travels into the layer above when $a > 0$ or when $a = 0$ and $p_i > 0$. A (downgoing) ray travels into the layer below when $a < 0$ or when $a = 0$ and $p_i < 0$. The ray position depends only on the $M$-unit gradient of the layer, not the absolute $M$-unit values.

### Upgoing Rays

A ray entering the ith layer of the $M$-unit profile, as shown in figure 1, has an initial ray launch angle at height $z_0 = z_i$, defined as $a_0$. The initial range at this height is $x_0$. $a_1$ is the angle at the end of the traced path, either at height $z_{i+1}$ or, if the trace is terminated in the layer, at some intermediate height $z_1$. Similarly, $x_1$ is the range at the end of the traced path. When the quantity $(a_0^2 + q_i) > 0$, then for any $z_i \leq z_{i+1}$ (refer to figure 1)

$$a_1 = a_0 + (x_1 - x_0)p_i \tag{6}$$

$$x_1 = x_0 + (a_1 - a_0)/p_i \tag{7}$$

$$z_1 = z_0 + (a_1^2 - a_0^2)/(2p_i) \tag{8}$$

If the ray will be traced to a range or height in such a manner that it will exit the top of the $i$th layer, then

$$a_1 = (a_0^2 + q_i)^{1/2} \tag{9}$$

In this case, $z_1 = z_{i+1}$, so equation 8 can be used to solve for $a_1$, but the $q_i$ array has been defined to minimize mathematical operations while tracing the ray path. If the ray trace is terminated at some specified height less than $z_{i+1}$, then equation 8 can be solved for $a_1$ and $x_1$ obtained from equation 7. Similarly, if the terminal range $x_1$ is such that the ray is still in the $i$th layer, then equation 7 can be solved for $a_1$ and $z_1$ obtained from equation 8.

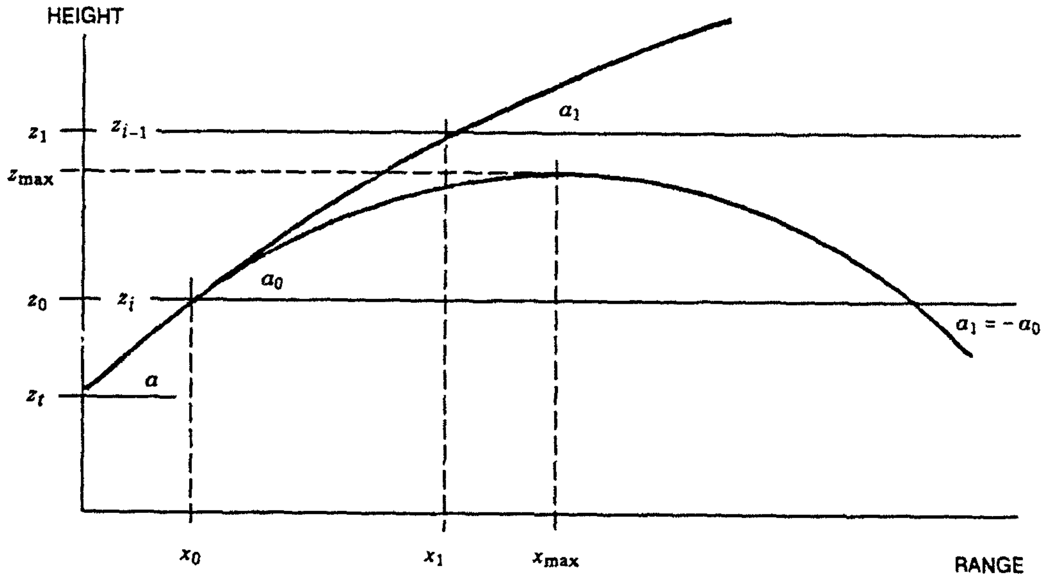


Figure 1. Upgoing ray trace.

When $(a_0^2 + q_i) < 0$, the ray reaches a maximum height and range in the $i$th layer given by

$$x_1 = x_{\max} = x_0 - a_0/p_i \tag{10}$$

$$z_1 = z_{\max} = z_0 - a_0/(2p_i) \tag{11}$$

and $a_1 = 0$ at this range and height, as shown in figure 1. By symmetry, the ray exits the layer at height $z_i$ with an angle $a_1 = -a_0$ and with a total range step (entry-maximum-exit) in the $i$th layer equal to $-2\,a_0/p_i$. If the ray trace will be terminated at some range greater than the $x_{\max}$ of equation 10 but less than the range-to-height $z_i$, then the trace equations for downgoing rays will apply for ranges beyond $x_{\max}$.

## Downgoing Rays

Equations 6 through 8 apply to downgoing rays for $(a_0^2 + q_{i-1}) > 0$, when $p_{i-1}$ is substituted for $p_i$, as shown in figure 2. If the ray will be traced to a range or height in such a manner that it will exit the layer at $z_{i-1}$, then

$$a_1 = (a_0^2 + q_{i-1})^{1/2} \tag{12}$$

In this case, $z_1 = z_{i-1}$, so equation 8 can also be used to solve for $a_1$. If the ray trace is terminated at some specified height $z_1$, which is greater than $z_{i-1}$, then equation 8 can be solved for $a_1$ and $x_1$ obtained from equation 7. Similarly, if the terminal range $x_1$ is such that the ray is still in the $i$-1 layer, then equation 7 can be solved for $a_1$ and $z_1$ obtained from equation 8.

When $(a_0^2 + q_{i-1}) < 0$, the ray reaches a minimum height in the $i$-1 layer given by

$$x_1 = x_{\min} = x_0 - a_0/p_{i-1} \tag{13}$$

$$z_1 = z_{\min} = z_0 - a_0/(2p_{i-1}) \tag{14}$$

and $a_1 = 0$ at this range and height, as shown in figure 2. By symmetry, the ray exits the layer at height $z_i$ with an angle $a_1 = -a_0$ and with a total range step (entry-maximum-exit) in the layer equal to $-2\ a_0/p_{i-1}$. If the ray trace will be terminated at some range greater than the $x_{\min}$ of equation 13 but less than the range-to-height $z_i$, then the trace equations for upgoing rays apply for ranges beyond $x_{\min}$.

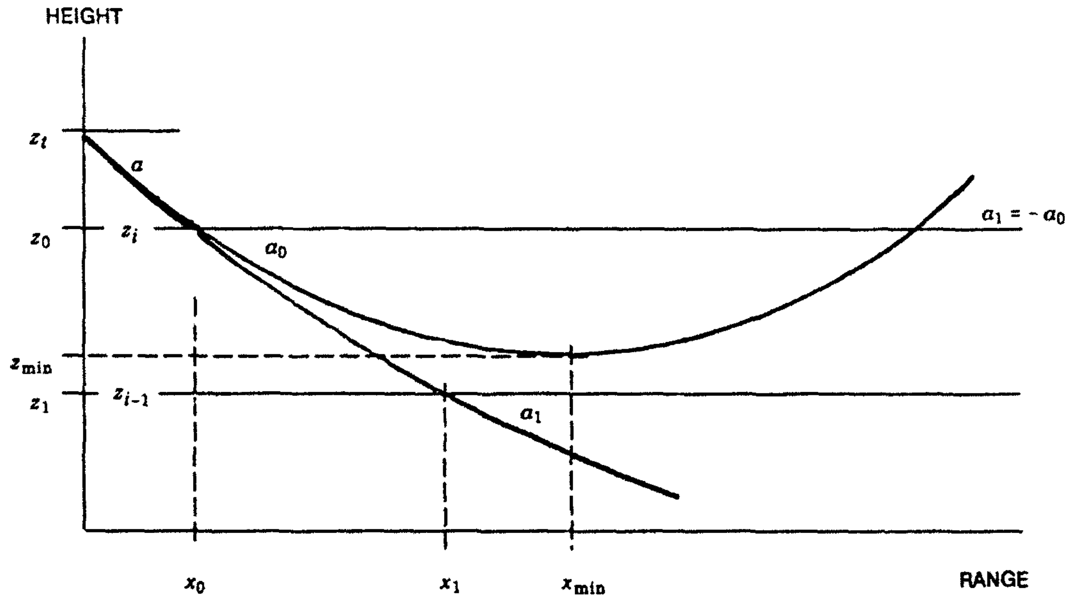


Figure 2. Downgoing ray trace.

## Optical Path Length Difference

The total phase difference between direct and sea-reflected rays is equal to the optical *path length difference* between the two paths plus the phase lag the reflected ray undergoes when reflected from the sea surface. We can obtain the optical path length difference by counting the number of wavelengths along the path, but this method quickly produces very large numbers for high frequencies. Hitney (1992) proposed an alternate method: sum the difference between the optical path length and the ground range for each ray, then subtract the direct ray sum from the reflected ray sum. This method avoids summing two very large numbers—a task that could require greater than normal computer precision. The basic quantities necessary for the calculation are shown in figure 3. $D_s$ represents the difference between the electrical path length $L_s$ and the ground range $x_s$ within a single layer $i$. In the figure, $L_s$ is the geometrical length of the arc of the ray path in the layer, which is actually the integral of the refractive index at each point times the incremental arc length. Within each layer, $D_s$ is given by

$$D_s = \frac{[(10^{-6} M_i - a_0^2/2)(a_1 - a_0) + (a_1^3 - a_0^3)/3]}{P_i} \tag{15}$$

for an upgoing ray. $D_s$ for a downgoing ray is identical, except $p_{i-1}$ replaces $p_i$ in equation 15. The total path length difference for any ray path is obtained by iteratively summing equation 15 over the entire path. If $D_r$ represents the sum of the $D_s$ elements over the reflected ray path and $D_d$ represents the direct ray sum, then the total path length difference, $\delta$, between the two rays is given in radians by

$$\delta = 2\pi(D_r - D_d)/\lambda \tag{16}$$

where $\lambda$ is the wavelength. The total phase difference, $\Theta$, between the two rays, including the phase lag due to reflection from the sea surface, is given in radians by

$$\Theta = \delta + \Phi \tag{17}$$

where $\Phi$ is the phase lag on reflection. The phase lag due to reflection from the sea surface, $\Phi$, is a function of the wavelength, the electric field polarization, and the incident angle. The phase lag for horizontal polarization varies slightly from $\pi$ radians at any frequency or incident angle, but the phase lag for vertical and circular polarization displays a Brewster angle effect. The EREPS 2.0 models are used to calculate the phase lag.

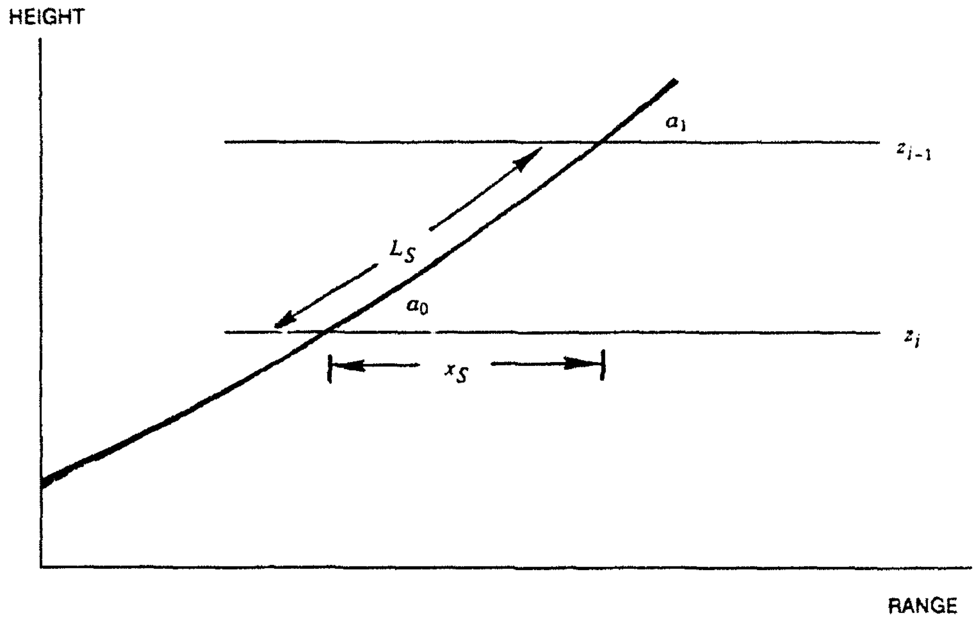


Figure 3. Electrical and geometric path length along a ray.

## Ray Divergence and Field Amplitude

To determine the electric field amplitude for direct and sea-reflected rays at a receiver located at some distant point, we use the transmitting antenna pattern weighting factors and compare the actual ray divergence at that point with the ray divergence that would occur in free space. The divergence calculation is based on the idea that field strength is inversely proportional to the area it spreads over at the receiver site. Assuming that the spreading in the azimuthal direction is identical for both free space and actual conditions, then the power (if we temporarily ignore the antenna pattern weighting factors) is given by

$$\left[\frac{E}{E_0}\right]^2 = F^2 = x\frac{da}{dz} = \frac{x}{[\beta(dx/da)]} \tag{18}$$

since power is the square of the electric field strength. Figure 4, where a small angle approximation tor the tangent of $\beta$ is implicit, illustrates this equation. Since both direct and reflected ray divergences can be determined by using this formulation, a separate calculation for spherical earth divergence is not necessary. The pattern propagation factors for the direct and sea-reflected rays are given by

$$F_d^2 = f_d^2 = \frac{x_x}{(-\beta_d(dx/da)_d)} \tag{19}$$

$$F_r^2 = f_r^2 = \frac{x_x}{(-\beta_r(dx/da)_r)} \tag{20}$$

where the $d$ subscript indicates the direct ray and $r$ represents the reflected ray. $\beta_d$ and $\beta_r$ are the local elevation angles and the $(dx/da)$ terms are the derivatives at the receiver site for the respective rays. The $f_d$ and $f_r$ terms are the (normalized to one) antenna pattern weighting factors for the direct and reflected rays, respectively. The EREPS 2.0 antenna pattern models are used for these calculations. The derivative $(dx/da)$ is computed in each step and summed over the entire ray path. The derivative for each step for an upgoing ray in a single layer is given by

$$\frac{dx}{da} = \frac{(a/a_1 - a/a_0)}{p_i} \tag{21}$$

where $x$ is the range step in the layer, $a$ is the initial ray launch angle at the transmitter, and $a_1$ and $a_0$ are the final and initial angles in the $i$th layer, respectively. The derivative for downgoing rays is identical, except $p_{i-1}$ replaces $p_i$ in equation 21. Equation 21's major disadvantage is that neither $a_1$ nor $a_0$ can equal zero. If a ray path in a layer goes through a minimum or maximum, then the summation must be taken over the whole step, and any ray path where $a_1 = 0$ for the final ray step must be avoided. Another disadvantage of this formulation is that the derivative quickly becomes very large for small $p_i$ , $a_0$, and $a_1$, resulting in a large ray divergence and a corresponding reduction of power at the receiver site. By itself this is not a shortcoming. However, closely spaced rays can have widely different divergence factor values, even when the gradients of adjacent layers are similar. This generally occurs when $p_i$ is small ($10^{-3}$, for example) and one ray penetrates a layer, goes through a minimum or maximum, then exits that layer while an adjacent ray fails to penetrate the layer.

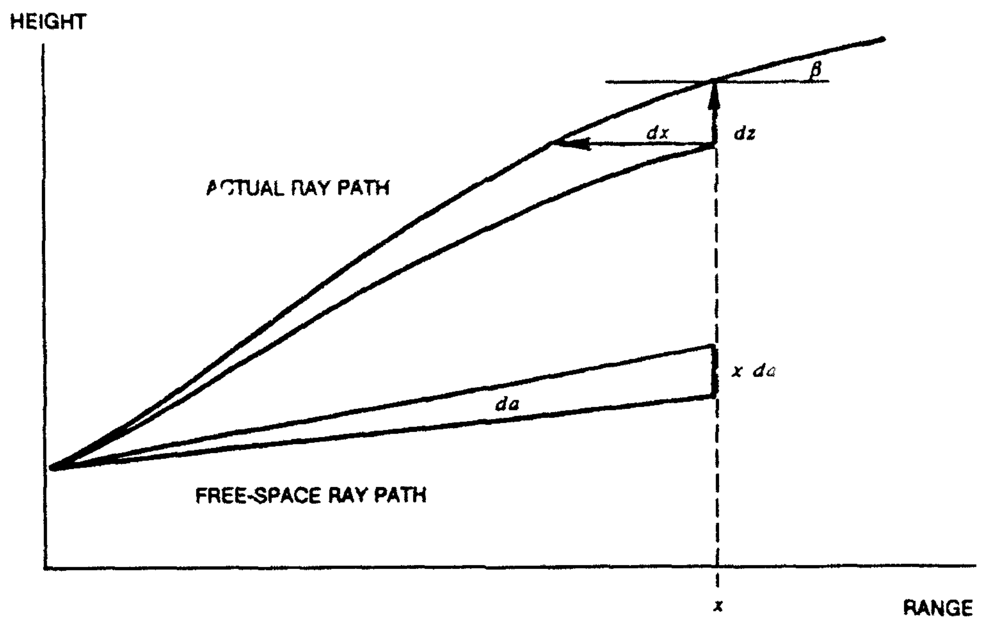


Figure 4. Ray divergence.

## Pattern Propagation Factor

Equations 17, 19, and 20 can be used to write the optical interference region pattern propagation factor for the direct and sea-reflected rays. The pattern propagation factor at the receiving antenna is the vector sum of the two ray contributions. It is given by

$$F^2 = F_d^2 + F_r^2R^2 + 2F_dF_rR\cos(\Theta) \tag{22}$$

where $R$ is the magnitude of the reflection coefficient, which is obtained by using the EREPS 2.0 models. The propagation loss is determined by using equations 3 and 22. The expression for $F$ in equation 22 is valid for all values of $\Theta$ where the path length difference $\delta$ is greater than or equal to a quarter wavelength ($\pi/2$ radians). Reed and Russell (1966) proposed a grazing angle limit that may extend the optical region to path length differences less than a quarter wavelength, but this limit is generally defined for only a single-gradient atmosphere and usually applies to lower frequencies and low-transmitting antenna heights. It is also possible that the quarter-wavelength limit is not attainable. In ducting environments, the limits may require much greater values of $\Theta$ for valid two-ray paths.

## Angular Limits

To solve equation 22, both a direct and a reflected ray path must exist over the range or height interval being considered. To simplify ray trace implementation, the transmitter height is always assumed to be the lower of the two terminal heights. To m[illegible] this condition, transmitter and receiver heights are "swapped" for ray trace purposes, if necessary. Heights are swapped because the lowest direct-ray launch angles are usually negative for most propagation conditions. Forcing the transmitter to be the lower of the two terminal heights ensures that the traced ray has only one intersection with the "receiver" height for those rays that will define the optical region. Since the principle of reciprocity holds—that is, the loss is independent of the direction of propagation along the path—this is not a restriction. (The correct launch angles for the antenna pattern factors in equations 19 and 20, which are used for swapped antenna heights, do, however, require substitution of $-a_1$ for $a$.)

When we discuss the angular limit to the optical interference region, it should be understood that the "limit" refers to the greatest range for a specified transmitter height, $z_t$, and receiver height, $z_r$, where two, and only two, rays can interfere. (0 radians is the horizontal and $\pi$ radians is the zenith angle.) This range corresponds to the absolute lower angular limit, $a_{\text{lim}}$, for the direct-ray path. Other considerations may further reduce this range, especially if the total phase difference between the two rays is greater than $2\pi$ radians. If $\Theta(a_{\text{lim}}) > 2\pi$, then the optical region limit is defined as the next optical region interference lobe peak where $[\Theta(a) = 2n\pi] > \Theta(a_{\text{lim}})$, for $n = 2$, 3, etc., and $a > a_{\text{lim}}$. The requirement to blend the end of the optical region to the beginning of the diffraction region necessitates this shortened optical region.

We consider five propagation situations when determining the angular limits, and we must determine the exact limit for each case by using an iterative ray-search scheme. It is usually possible to empirically determine an approximate angular bound for each of the direct and the sea-reflected rays, but these rays do not necessarily interfere with each other. Case 1, the simplest case, is an atmosphere with the minimum $M$-unit value of the refractivity profile, $M_{\text{min}}$, at the surface and no subrefractive layers or surface ducts below the transmitter height, $z_t$. In addition, $z_t$ must not be below a local minimum on the $M$-unit profile, i.e., not be in an elevated-duct. This is somewhat equivalent to a standard, single-gradient atmosphere, but super-refractive layers or small nonsurface-based ducts can exist below $z_t$ without much impact. Of course, since $z_t$ is restricted in EREPS to heights of less than 100 m, it is likely that any ducts below $z_t$ are surface ducts. Case 2 accounts for subrefractive layers between $z_t$ and the surface, and case 3 accounts for surface-based ducts below $z_t$. Cases 4 and 5 involve the transmitter in a surface-based duct and the receiver $z_t$ either above the duct or in the duct, respectively.

**Case 1 Angular Limits.** If the minimum $M$-unit value of the refractivity profile occurs at the surface, then the lowest possible direct ray launch angle is the ray just slightly greater than the ray that is tangent to the earth's surface. The tangent ray launch angle, $a_t$, is given by

$$a_t = -10^{-3}[2(M_{zt} - M_0)]^{1/2} \qquad (23)$$

Here $M_{zt}$ is the $M$-unit value at the transmitter height $z_t$ and $M_0$ is the $M$-unit value at the surface. Ray launch angles that are slightly more negative than $a_t$ are reflected from the surface, and rays with slightly less negative launch angles are direct rays. $a_t$ is the upper angular limit of the reflected ray set and the lower angular limit of the direct ray set. However, the actual ray paths must normally meet the quarter-wavelength requirement, so the actual limiting ray launch angles are constrained by the wavelength itself: the higher the frequency, the closer the angular limits are to $a_t$.

As mentioned before, a grazing angle limit may extend the optical region to ranges beyond the quarter-wavelength limit. This limit is given by the expression

$$\psi_{\text{lim}} = \tan^{-1}\left[\frac{10^{-3}\lambda}{2\pi a_e}\right]^{1/3} \approx \left(\frac{0.01957}{(kf)^{1/3}}\right) \tag{24}$$

where $\psi$ is the grazing angle, $\lambda$ is the wavelength in m, $a_e$ is the effective earth radius, and $f$ is the frequency in MHz. $a_e$ is defined as the mean earth radius $a$ times the effective earth radius factor $k$. Since the atmosphere of case 1 is relatively simple, we can establish a pseudo single-layer atmosphere by means of a ray trace. $k$ can be obtained by tracing a ray launched at $z_0$ with a launch angle of $a = 10^{-3}$ to 370 km (200 nautical miles) and comparing the height at that range to the height that a ray launched at the same angle in a single-gradient atmosphere would achieve. The single-layer equivalent $M$-unit gradient is

$$P = -(2a)/370 + (0.002z)/(370) \tag{25}$$

where $z$ is the terminal height of the ray at 370 km. Then $k$ is defined as

$$k = 1/(Pa) \qquad 1 \le k \le 5 \tag{26}$$

where $a$ is the mean earth radius of 6371 km. (In a standard atmosphere, $k = 4/3$ and $P = 0.000118$.) $\psi_{\text{lim}}$ is used to determine the upper angular limit at the transmitter for the reflected ray. A ray with an initial launch angle of $a = \psi_{\text{lim}}$ is traced from the surface to $z_t$. The upper angular limit corresponds to the inverse of the penetration angle at the transmitter height $a = -a_1$. If a valid direct ray that interferes with this ray at the receiver height and has a path length difference less than $\pi/2$ radians can be found, then these two rays become the limiting rays.

**Case 2 Angular Limits.** Direct-ray multipath can occur if subrefractive layers exist below $z_t$. (Subrefractive layers have gradients of $p_i \ge 0.157$ $M$/m.) A ray launched with an initial negative angle that penetrates the subrefractive layer may interfere with other direct rays (crossed paths) when traced to sufficiently long ranges, as shown in figure 5. The ray tangent to the top of the subrefractive layer is

$$a_{st} = -10^{-3}[2(M_{zt} - M_{s2})]^{1/2} \qquad M_{zt} \ge M_{s2} \tag{27}$$

where $M_{s2}$ is the $M$-unit value at the top of the subrefractive layer. The ray tangent to the bottom of the subrefractive layer is

$$a_{sb} = -10^{-3}[2(M_{zt} - M_{s1})]^{1/2} \qquad M_{zt} \geq M_{s1} \qquad (28)$$

where the $s2$ index is replaced by $s1$. Rays with launch angles of $a_{st} > a > a_{sb}$ may cross to create direct-ray interference, which violates the two-ray requirement. The lowest direct-ray angle can be found by tracing several rays to the range or height of interest, then determining if any of these rays cross. The lowest ray that does not cross a ray with a more negative launch angle becomes the lower direct-ray bound. Of course, to be valid, the ray must also meet the quarter-wavelength restriction and other restrictions.

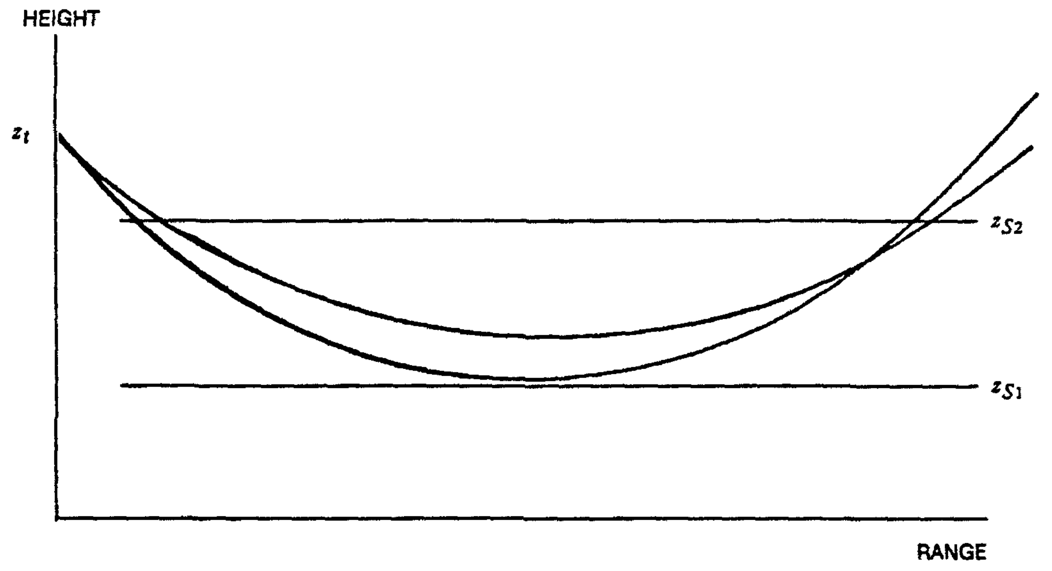


Figure 5. Direct ray interference due to a subrefractive layer.

**Case 3 Angular Limits.** If the transmitter is located above a surface-based duct, then equation 23 is not valid for $a_t$. In this case, the lowest direct ray is the ray that is tangent to the top of the duct, and its launch angle is

$$a_t = -10^{-3}[2(M_{zt} - M_{\text{min}})]^{1/2} \qquad M_{zt} \geq M_{\text{min}} \qquad (29)$$

where $M_{\text{min}}$ is the minimum $M$-unit value at some height below $z_t$. The height where $M_{\text{min}}$ occurs, $z_{\text{min}}$, is the height of the surface-based duct. There are two kinds of surface-based ducts: evaporation ducts and surface-based ducts created by elevated layers. If $z_{\text{min}}$ equals the EREPS-specified evaporation duct height, then the duct is treated as an evaporation duct for the diffraction field region calculations. This has no effect on the optical region calculations, however. Rays launched with $a < a_t$ penetrate the duct and are reflected from the surface, and rays with $a > a_t$ are direct rays. $a_t$ then defines the angular limit just as equation 23 defines it for case 1. There is, however, a crucial difference. Since the ray paths through the duct are altered, the optical region maximum range will probably be extended. It is also possible that the quarter-wavelength condition cannot be met. If two interfering rays that meet the quarter-wavelength condition cannot be found, then the optical region limit is defined by the ray pair that minimizes $\Theta$ for $\Theta \leq 2\pi$. If $\Theta > 2\pi$, then the optical region is defined by the ray pair that produces the lowest possible integer multiple of $\Theta = 2n\pi$, where $n = 1, 2, 3$, etc., because it is difficult to blend optical and diffraction calculations if, for instance, the optical region were to end in a null ($\Theta = 3\pi, 5\pi$, etc.).

**Case 4 Angular Limits.** If the transmitter is located below a local minimum of the modified refractivity profile and the receiver is above the minimum, then there is almost certainly a sufficiently high launch angle that a direct-ray path between the two exists. This critical angle to the top of the duct is defined as

$$a_c = -10^{-3}[2(M_{\min} - M_{zt})]^{1/2} \qquad M_{zt} \leq M_{\min} \tag{30}$$

while the minimum negative critical angle is equal to $-a_c$. Here $M_{\min}$ is the minimum $M$-unit value at some height greater than $z_t$. If $z_t$ is in a duct, then the duct is treated as a surface-based duct for the diffraction region calculations, even if it does not extend to the surface. The height where $M_{\min}$ occurs, $z_{\min}$, is the height of the surface-based duct.

Rays launched with angles of $a_c > a > -a_c$ are trapped within the duct. Two possible ray paths exist for the ray launched at $a_c$, because the ray reaches a maximum at the top of the duct and can split into an upgoing and a downgoing ray. Normally, when determining the lower angular direct-ray limit to the optical region, an additional $10^{-6}$ radians are added to $a_c$ to ensure the ray will penetrate the duct. Any ray launched with a more negative angle than $-a_c$ is reflected from the surface if $M_{\min} \leq M_0$. If $M_{\min} > M_0$ (an earth-detached duct), then the reflected-ray set upper boundary is given by equation 23 to be consistent with the quarter-wavelength and other phase difference limit conditions.

**Case 5 Angular Limits.** If both the transmitter and the receiver are located in a surface-based duct, then an optical interference region can exist for angles less than $a_c$. Such rays are, by definition, trapped rays and, if traced to great enough ranges, they become subject to direct-ray interference. The optical region for this case may be extended beyond the range given by $a_c$ if the trace is done in such a manner that it excludes multiple direct-ray interference rays. The ray set bounded by $\pm a_c$ always defines a valid, though possibly short, optical region, as long as the transmitter height is below the receiver height.

In the special case where $z_t = z_r < z_{\min}$, only direct rays launched with angles $a_c > a > -a_c$ can reach the receiver height. It is possible to construct a surface-based duct profile in such a manner that a two-ray optical interference region will not exist for this case. One example of such a duct would be a bilinear duct with $p_0 = -p_1$, $z_1 = z_t = z_r$, $z_2 = 2z_t = z_{\min}$, and $M_0 = M_2 = M_{\min}$. By symmetry, each negative direct ray with a launch angle of $a > -a_c$ has a corresponding, interfering, positive direct-ray launch angle with the same magnitude but opposite sign.

Case 5 angular limits are not implemented in the ray trace program. An alternate way of performing the optical interference region calculations for this case uses the current EREPS single-gradient model, effective earth radius method. To use this method it is necessary to obtain an effective earth radius factor, *k*, as described above under "Case 1 angular limits." However, the correct value for $a_c$ in equation 30 is now

$$a_c = -10^{-3}[2(M_{\min} - M_0)]^{1/2} + 10^{-6} \tag{31}$$

which is the angle necessary to escape the duct. We can then perform the optical region calculations by using the effective earth radius obtained from equations 25 and 26 and the EREPS 2.0 optical region models.

## RAY TRACE MODEL QuickBASIC PROGRAM

A QuickBASIC program that implements the ray trace model, RAYOPR, has been developed for IBM-compatible personal computers. QuickBASIC is a Microsoft Corporation extended BASIC language that provides a structured programming environment. Existing EREPS programs are also written in QuickBASIC, and many of the existing EREPS subroutines are incorporated into the program. RAYOPR provides a plot of propagation factor (in dB) versus range as the output. A complete program listing for RAYOPR is in appendix A.

### Environmental and EM System Inputs

RAYOPR requires all of the normal EREPS EM system inputs: frequency in MHz, transmitter and receiver/target heights in m, antenna type and antenna beamwidth, elevation angle, and polarization. In RAYOPR, the antenna type is fixed to a horizontally polarized Sin($x$)/$x$ type with a 4-degree beamwidth and an elevation angle of 0 degrees. These inputs are fixed because they do not affect the ray trace portion of the program.

RAYOPR also uses the six EREPS environmental inputs: surface-based duct and evaporation duct heights, humidity, wind speed, surface refractivity, and the effective earth radius factor. RAYOPR uses fixed values of 7.5 g/m for the humidity and 0 knots for the wind speed, because these quantities do not affect the ray trace portion of the program. Instead of using a specified surface-based duct or evaporation duct height, RAYOPR obtains these quantities from an ASCII input file containing the *M*-unit profile. The first line of the ASCII input file contains a label and the second contains the evaporation duct height in m. The remaining lines list the height and *M*-unit pairs described above in the "Ray Trace Model" section. The file is dimensioned for a maximum of 50 height elements; a minimum of 2 heights is required. The height and *M*-unit elements are separated by at least one space. The evaporation duct height is specified in the file since the program is unable to determine the type of surface duct by inspecting the profile. The effective earth radius factor, required by several EREPS subroutines, is obtained through the ray trace technique described above under "Case 1 angular limits." Examples of the environments used in testing the ray trace model are in appendix B.

RAYOPR also requires a maximum plot range in km and the range plot increment in m. The range plot increment determines the number of points that will be plotted for the output. Several factors determine how large this increment should be. In general, for a fixed geometry, the higher the frequency, the smaller this number must be to obtain a smooth output plot. A default set, which the user can accept or change, is provided of each of these inputs at run time.

## Program Description

RAYOPR incorporates several existing EREPS subroutines. The EREPS subroutines used by RAYOPR are the following: ANTPAR, ANTPAT, DCONST, DIFINT, DLOSS, GOFZ, HGAIN, OPCONST, REF, RUFF, and SBD. These routines are documented in the EREPS user's manual and they will not be described here in detail.

The RAYOPR MAIN program is a simple program that contains the common block INCLUDE file variables and the EM system and environmental default variables. MAIN displays and labels the default values for the operator-selected variables, and it displays the prompts to plot and label the axes and initiate and/or overplot the propagation loss curves. MAIN also allows the operator to edit operator-selected inputs.

The MAIN program displays and labels the default values for the following operator-selected variables: EM system frequency, transmitter and receiver/target heights, plot range, and plot increment. An "*" character is displayed in the first column of the screen, initially on the EM system frequency input line. The * is an editing aid, indicating which current line's input value the operator has the option to change. The line containing the * can be selected, up or down, by using the arrow keys. When the operator presses the ENTER key, the variable on the line containing the * in the first column is deleted and the operator must enter the desired numerical value, then press ENTER. MAIN also displays the prompts that allow the operator to initiate a plot and/or overplot an existing plot. To plot the axes and propagation loss curves, the operator presses special function key F10. To overplot an existing propagation loss plot, the operator presses special function key F9. MAIN calls two subroutines, PLOTAXES and CALCS. PLOTAXES scales and displays the plot axes. CALCS initiates the ray trace and propagation loss plots. MAIN is intentionally a "minimal" program, allowing the operator to quickly edit and recompile program code without the overhead of the EREPS code with its more complex edit and plot capabilities.

CALCS opens and reads the *M*-unit input file, swaps transmitter and receiver/target heights if necessary, and calls the subroutines needed to determine the propagation loss. CALCS calls, in order, the following subroutines: MPQARRAY, GETK, OPCONST, DCONST, ANTPAR, SKIPTR, OPTICAL, and DIFRACTR. MPQARRAY creates the $zt_i$, $M_i$, $p_i$, and $q_i$ arrays used in the ray trace subroutines. $zt_i$ and $M_i$ contain the height and *M*-unit arrays used in the actual ray trace and differ from the input file arrays by addition of the transmitter height and $M_{zt}$ value, if these values are not included in the input file. MPQARRAY also checks the profile to determine if a surface-based duct is present and determines the duct's height and the critical angle necessary to penetrate it. GETK is used to obtain an effective earth radius factor as described under "Case 1 angular limits." GETK also performs a ray trace for a direct-ray initial angle of 3 degrees, then calculates the effective earth radius at this relatively high angle where the effects of refraction are minimal. Our original intention was to switch to the existing EREPS single-layer model above 3 degrees where refractive effects are minimal. However, the program code to perform this task was never implemented.

After the effective earth radius factor is determined, OPCONST and DCONST are called to initialize constants for the optical and diffraction regions. Then ANTPAR is called to initialize the antenna pattern parameters for the ANTPAT antenna pattern subroutine. If a surface-based duct due to elevated layers is present, then the SKIPTR subroutine is called. SKIPTR is a ray trace routine used to determine a "skip zone" distance. SKIPTR is used only if the transmitter and receiver/target heights are both in the surface-based duct. This distance is established by tracing a ray with a launch angle of 0 radians (the critical ray) from the top of the surface duct to both the transmitter and receiver/target heights. When the total range obtained from the ray trace is greater than the maximum range in the optical region, the decrease in signal strength in the intermediate region is accounted for by using the existing EREPS models. The optical region limit for this case is described above under "Case 5 angular limits," but it is not implemented in the program code.

OPTICAL is the subroutine directing the implementation of most of the ray trace model. OPTICAL calls OPLIMITR, which determines the maximum range in the optical region. Then OPTICAL calculates and plots the propagation loss from the shortest optical region range to the maximum range. A flat-earth approximation is used to obtain a starting angle and RALOOPR is used to determine the exact angle necessary to place both rays through the target/receiver height at that range. Once the plot is started, the range is increased in steps equal to the operator-selected range increment until the optical region maximum is reached. At each range step, subroutine RALOOPR is used to determine the direct and reflected ray launch angles necessary to reach the receiver/target height. Then subroutine FFACTR is called to determine the pattern propagation factor for the two rays. This process is repeated until the optical region maximum range is reached or the maximum plot range is exceeded.

OPLIMITR determines the maximum range in the optical region; it traces both a direct and a reflected ray by using the approximate limiting angles discussed above under "Angular Limits." The shorter of the two obtained ranges becomes the starting point for determining the optical limit. While this shorter range is the initial limit, we obtain the launch angle that places the other ray through this range by using a Newton–Raphson iteration scheme. The iteration is performed in subroutine RALOOPR, which returns the launch angle necessary to place a ray through the desired range, assuming such a ray exists. Once we know the ray launch angles that correspond to this range, we calculate the optical path length difference. If these rays meet the appropriate limit—quarter-wavelength, grazing angle, etc.—this range is the optical limit; if they do not, the correct range is determined by iteration.

RALOOPR determines the ray launch angle necessary to place a ray through a particular range and height. RALOOPR is used for both direct and reflected rays. It calls subroutine RAYR to perform the actual ray trace, then it uses a Newton–Raphson iteration technique to place a ray through the desired range and height within the range tolerance limits (0.5 m for most of the test cases that were run). If the Newton–Raphson iteration fails to achieve a solution within 11 tries, then one of two other iterations is used. If the present and previous solution are on opposite sides of the desired range, then a halving technique is used. The angular difference between the two launch angles is divided into 10 equal parts, and this angular increment is added to the ray angle corresponding to the shorter range. This action is repeated until either the desired range is within the tolerance limits or exceeded. If the desired range is exceeded and the actual range is still not within the range tolerance, then the angular increment is divided in half and the process repeated. This process continues until the solution is achieved or until 25 iterations have occurred. It is possible that a solution cannot be found within the precision limits of the computer. If both the previous and present solutions from the Newton–Raphson iteration are either greater than or less than the desired range, a slightly different iteration technique is used. The angular distance between the present and the previous ray is scaled by the ratio of the present range minus the target range divided by the present range minus the previous range. This angular increment is added to the present launch angle until the range is within the range tolerance limits or 11 iterations have occurred. If the addition of the angular increment causes the ray to switch from a direct to a reflected ray, or vice versa, the angular increment is halved until the ray type is correct. Again, it is possible that a solution does not exist within the numerical precision of the computer. The last angle used in the iteration is accepted as the correct answer if the ray is the correct type. If the ray type is not correct, an error flag is returned and processing stops.

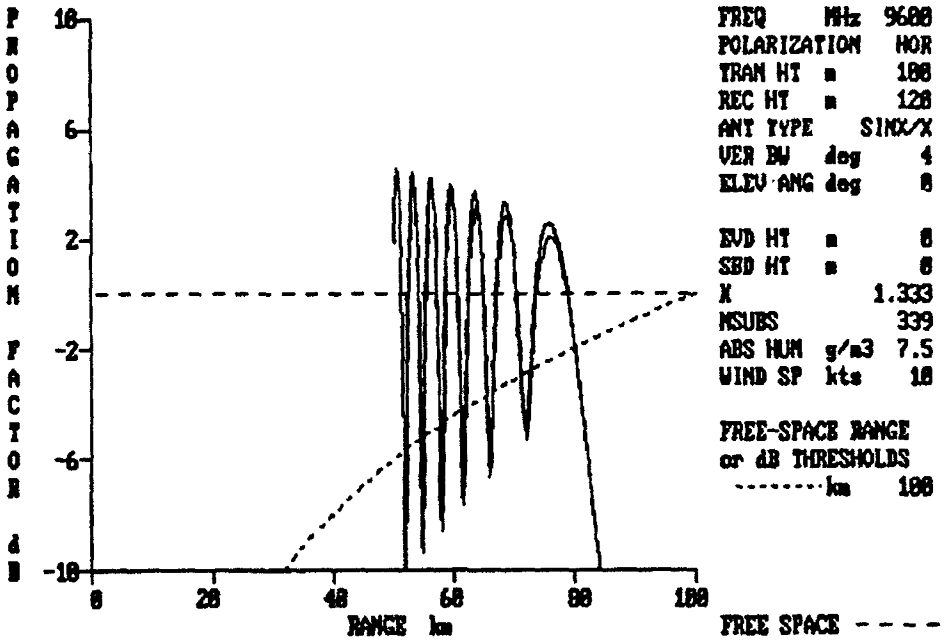
Subroutine RAYR traces a ray to the receiver/target height by using a specified initial launch angle of $a$. RAYR returns the range at this height; the terminal angle, $a_1$; the path length difference, $\delta$; and the ray divergence, $dx/da$. If the ray is a reflected ray, then the grazing angle, $\psi$, is also returned. RAYR returns an error flag if the direct or reflected ray type is not correct. A failure flag is set if the ray cannot reach the receiver/target height. RAYR is not designed to trace trapped rays and terminates when the ray intercepts the receiver/target height.

The GETK subroutine uses a special ray trace subroutine, RAYPOS, to trace a ray to 370 km and return the height at that range. RAYPOS differs from RAYR in several ways. RAYPOS initiates the trace from the surface rather than the transmitter height, traces only upgoing rays, and returns the height at a specified range rather than the range at a specified height.

## TEST CASES

A variety of test cases were run to validate the ray trace method. Test cases were compared to the RPO program for nonstandard *M*-unit profiles or the existing EREPS program for single-gradient atmospheres. The RPO program uses a split-step Fourier transform to solve a parabolic approximation to the full wave equation. RPO is believed to be one of the most accurate programs available to assess EM propagation in the 100-to-20,000-MHz regime. RPO is a larger and more complex program than RAYOPR. Comparisons were made using the EREPS binary and ASCII file plot capability, which allows loss values obtained from different programs to be displayed on the same scale.

Figure 6 compares the EREPS 2.1 output for a standard, single-layer atmosphere and a 9600-MHz EM system that has a transmitter height of 100 m and a receiver height of 120 m. Both programs give identical null placements, but the ray trace model differs from the single-layer model by about 0.5 dB at the first optical region peak (which corresponds to the greatest range in the optical region—approximately 85 km). There is a smaller, but detectable, difference at the second and third peaks. The difference is due to the divergence factor calculation. However, the overall agreement is quite good, as should be expected for such a simple atmosphere. The profile used to generate the ray trace plot is given in appendix B.



```
FREQ          MHz   9600
POLARIZATION         HOR
TRAN HT       m      100
REC HT        m      120
ANT TYPE          SINX/X
VER BW        deg      4
ELEV ANG      deg      0

EVD HT        m        0
SBD HT        m        0
K                  1.333
NSUBS                339
ABS HUM       g/m3   7.5
WIND SP       kts     10

FREE-SPACE RANGE
or dB THRESHOLDS
 --------km          100
```

Figure 6. Comparison of EREPS and ray trace models for a standard atmosphere.

Figure 7 compares the RPO program with the ray trace model for a 28-m evaporation duct profile and the same EM system parameters of figure 6. This profile, which contains 30 levels, is given in appendix B. Once again, both models agree on the location of the nulls. The loss value differences are due to the divergence factor calculation in the ray trace model. The ray trace model optical region is also shorter than the RPO model; the ray trace model can calculate valid two-ray paths to a range of only about 115 km, which does not correspond to an optical region peak, while the phase difference at this range is greater than $2\pi$ radians. In this case, the next optical region peak becomes the end of the optical region in the ray trace model. Figure 8 shows the ray trace plot alone for the same case. Note the "wiggles" in the ray trace model path-loss curves at 67, 80.5, and 94 km. These deviations mark the ranges where the ray paths begin to pass through successively lower levels in the profile. Figure 9 plots the ray trace model without corrections to the divergence factor. The ranges where the rays pass through the lower layers are clearly shown. Figure 9 also shows how far the optical region can be traced if the limits cited in the "Angular Limits" section are ignored. Ignoring those limits, the greatest range where two valid rays can be traced is 125.23 km. The total phase difference at that range is $6.29\pi$ radians, which is not quite equal to the third optical peak value of $6\pi$ radians.

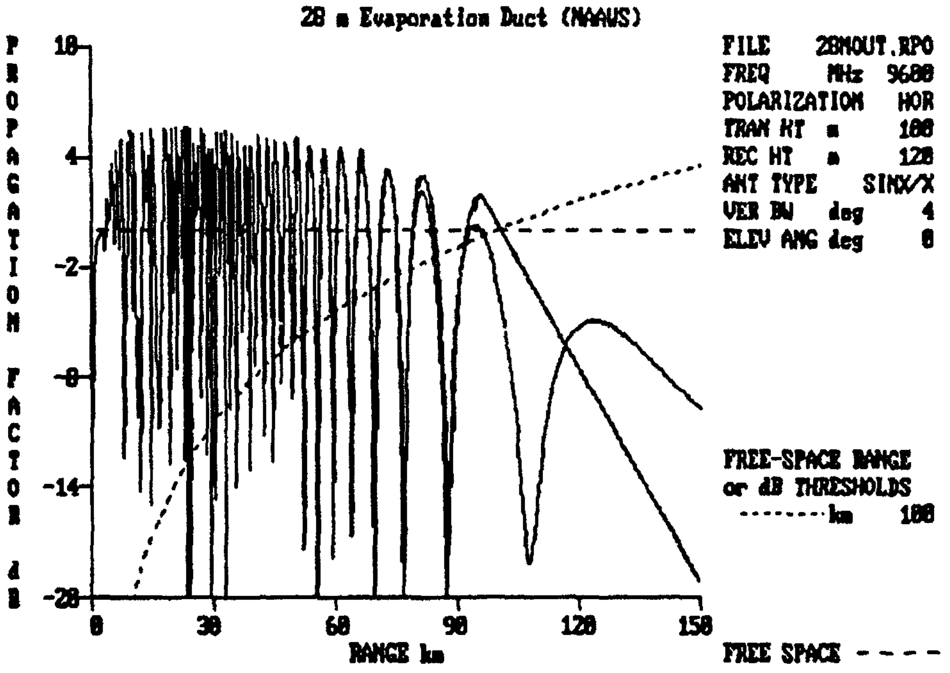


Figure 7. Comparison of RPO and ray trace models for a 28-m evaporation duct profile.

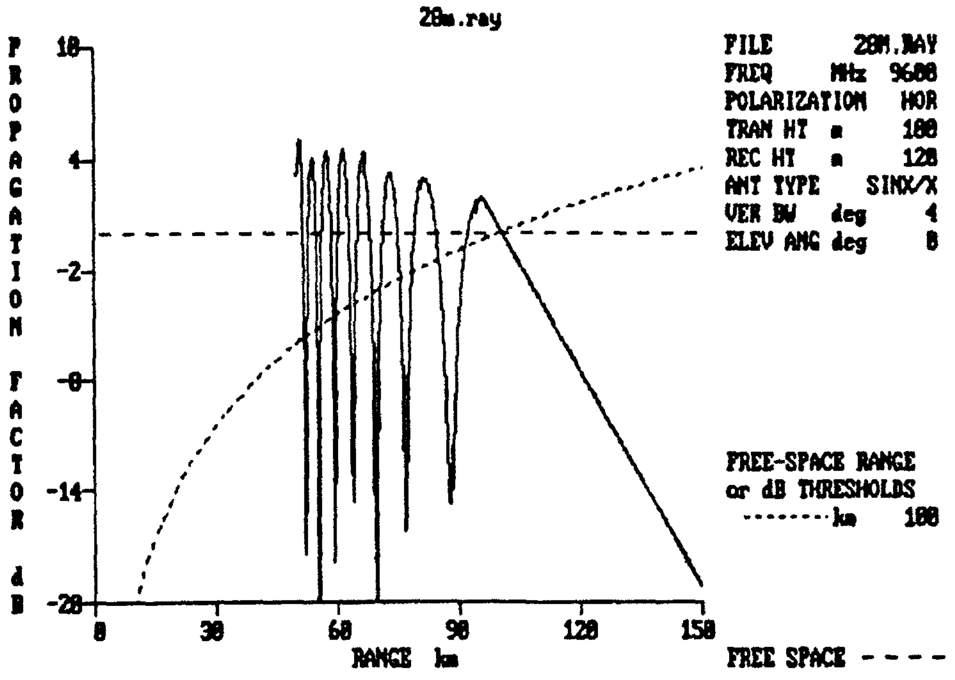


Figure 8. Ray trace model plot for the 28-m evaporation duct profile of figure 7.

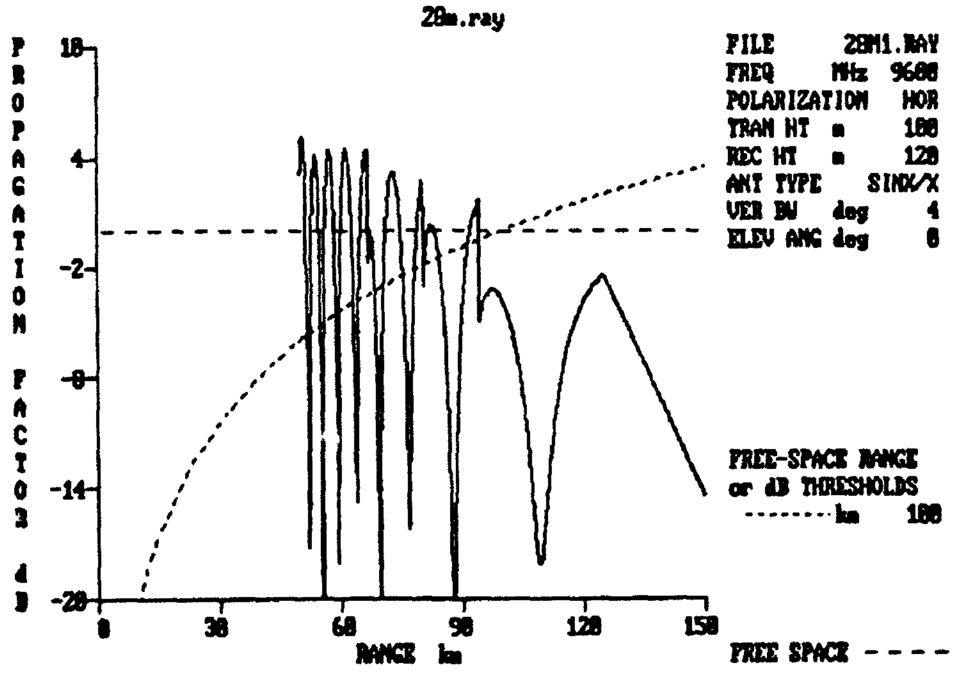


Figure 9. Divergence factor fluctuations for a 28-m evaporation duct profile.

## CONCLUSIONS AND RECOMMENDATIONS

The ray trace model can be an effective tool under certain conditions for determining propagation loss. Unfortunately, it has two significant deficiencies: ray resolution and the ray divergence calculation. It becomes more difficult to resolve ray paths to range tolerances of 0.5 m as EM system frequency increases. The resolution is also affected by the absolute ground range: the greater the range covered by the trace, the harder it is to accurately place a ray through a given range and height. Even double precision calculations cannot always resolve the ray to the desired degree.

Equally troubling is the problem of the divergence factor calculation. The problem originates in equation 21 when a ray that is initially negative barely penetrates a layer below the transmitter and reaches a minimum height before exiting the layer. Inspection of equation 21 shows that $da/dx = -2(a/a_0)/p_{i-1}$ for the $i$-1 $M$-unit layer where the ray reaches a minimum. When the initial angle, $a$, is about $10^{-3}$ radians and the entry and exit angle, $a_0$, is very small ($10^{-10}$ radians), then the resultant ray divergence in that layer is quite large. If the $p_{i-1}$ gradient is also small, the effects are exacerbated. Typically, this causes a large divergence factor difference between rays whose launch angles differ slightly. The calculated divergence value can go from 0.8 to nearly zero with little change in the launch angle or the gradient between the layers where change occurs. The RPO program output indicates the divergence factor does vary, but not to the degree given by equation 21, which typically causes a change in the propagation factor of 10 or more dB.

Several different methods to eliminate the divergence factor problem have been tried, but they have achieved only partial success. One method calculated the divergence, $da/dx$, by performing an additional ray trace at a slightly different ray launch angle, then determined the divergence directly from the terminal range of the two rays. The additional ray was traced if the direct-ray divergence factor varied by more than 0.2 between adjacent range steps. This method helped somewhat, but it created rather ragged plots of propagation loss. The additional ray traces also took longer and slowed the program down. The most successful method attempted set the direct-ray divergence factor to 0.6 if the divergence factor variation was more than 0.2 between adjacent range steps and the new value was less than 0.6. If this value of 0.6 was greater in magnitude than the value of the divergence factor at the previous range step, then the previous value was used. This allowed the divergence factor to slowly vary to values less than 0.6 and still filter out the large fluctuations that occur as rays successively penetrate lower layers. None of the methods tried removed all the fluctuations, however. The ultimate conclusion is that the ray trace model is not suitable to be included in the EREPS models.

It should be noted that RPO uses essentially the same ray optics methods as those described in this report. However, RPO restricts the application of the ray optics methods to substantially higher elevation angles, and hence no numerical problems with ray resolution or ray divergence calculations exist.

## REFERENCES

Hitney, H. V. 1992. "Hybrid Ray Optics and Parbolic Equation Methods for Radar Propagation Modeling," in *Radar 92, IEE*, IEE Conference Publication No. 365 (pp. 58-61). October 12-13, Brighton, United Kingdom.

Patterson, W. L., C. P. Hattan, H. V. Hitney, R. A. Paulus, A. E. Barrios, G. E. Lindem, and K. D. Anderson. 1990. "Engineers Refractive Effects Prediction System (EREPS) Revision 2.0," NOSC TD 1342 (February). Naval Ocean Systems Center, San Diego, CA.

Reed, H. R., and C. M. Russell. 1966. *Ultra High Frequency Propagation*. Boston Technical Publishers, Inc., Cambridge, MA.

# APPENDIX A
# PROGRAM LISTING FOR RAYOPR

```
'PROGRAM RAYOPR

'   Rayopr Computes 20LOG10(F) vs. range, where F is the pattern
'   propagation factor

'INPUTS: Profile name of height & M-unit profile
'        Source (transmitter) height (m) -- source
'        Frequency (MHz) ----------------- freq
'        Range (km) ---------------------- xkm
'        Receiver/target height (m) ------- hr
'        Range increment for output (m) --- rinc
DEFINT I-N
CONST PI = 3.14159
COMMON SHARED /comffactr/ ae, ae2, aeth, alpha, antbwr
COMMON SHARED /comffactr/ antelr, antfac, antyp$, atten
COMMON SHARED /comffactr/ bwidth, C1, C2, C3, C4, C5, C6, C7
COMMON SHARED /comffactr/ capk, del, delta, difac, dtot, elevat
COMMON SHARED /comffactr/ elmaxr, exloss, f3, fofz
COMMON SHARED /comffactr/ freq, fsloss, fsterm, h1, h2
COMMON SHARED /comffactr/ h14pil, h24pil, h24ae2, hbar
COMMON SHARED /comffactr/ hbfreq, hdif, hmin, horizn
COMMON SHARED /comffactr/ horizn1, hr, ht, humid, opmaxd
COMMON SHARED /comffactr/ opmaxl, patd, patrfac, polar$, psi
COMMON SHARED /comffactr/ psilim, r1lim, r1min, rk, rmag
COMMON SHARED /comffactr/ rn2imag, rn2real, rns2, rnsterm
COMMON SHARED /comffactr/ rnsubs, rsbd, rsbdloss, rsubd
COMMON SHARED /comffactr/ sbdht, tfac, thefac
COMMON SHARED /comffactr/ tsub1, tsub2, twoae, wind, wvatten
COMMON SHARED /comffactr/ xterm, zfac, zmax, zterm

COMMON SHARED /rayoph/  M!(1), p(1), q(1), zt(1), grdlim(1)
COMMON SHARED /rayoph/  imin, istart, iswap, levels, Mmin!, numin
COMMON SHARED /rayoph/  asubc1, colr, dtot1, pltmax
COMMON SHARED /rayoph/  rk3deg, source, twopi, wlngth
COMMON SHARED /rayoph/  xmax, zinc, ztol

COMMON SHARED /rayopr/   admin, armax, ray1, rinc, rtol, irng
COMMON SHARED /rayopr/   divdlst, divfacd, zmin, rng(1), pl(1)

DIM rng(1000),pl(1000)
DIM zin(50), Munits!(50), zt(50), M!(50), p(50), q(50), grdlim(50)
'  Set default values for plot, system and environmental variables
screen  9                 ' Set EGA mode
f$ = "28m"
source = 100.
hr = 120.
freq = 9600.
xkm = 150.
pltmax = 2500.
rinc = 100.
nlobes = 2
polar$ = "H"              '  EM system default variables
antyp$ = "SINX/X"
bwidth = 4!
elevat = 0!
delta = 0!                '  Environmental default variables
```

```
humid = 7.5
sbdht = 0!
wind = 0!

LOCATE 2,3:  PRINT "filename:"
LOCATE 2,16: PRINT f$
LOCATE 3,3:  PRINT "freq (MHz):", freq
LOCATE 4,3:  PRINT "source (m):", source
LOCATE 5,3:  PRINT "tgt (m):", hr
LOCATE 6,3:  PRINT "range (km):", xkm
LOCATE 7,3:  PRINT "rinc (m):", rinc
lin=0:star=2
LOCATE  9,3: PRINT "F9:  Overplot";
LOCATE 10,3: PRINT "F10: New plot";
LOCATE 2,1:  PRINT "*";
WHILE a$<>chr$(27)
  IF lin=2 THEN LOCATE 2,16:INPUT "", f$
  IF lin=3 THEN LOCATE 3,16:INPUT "", freq
  IF lin=4 THEN LOCATE 4,16:INPUT "", source
  IF lin=5 THEN LOCATE 5,16:INPUT "", hr
  IF lin=6 THEN LOCATE 6,16:INPUT "", xkm
  IF lin=7 THEN LOCATE 7,16:INPUT "", rinc
  lin=0: a$=""
  delta = 0.0
  IF f$ = "14m" THEN delta = 14.
  IF f$ = "28m" THEN delta = 28.
  IF f$ = "combi" THEN delta = 14.
  WHILE a$=""
    a$=inkey$
  WEND
  IF a$=chr$(0)+chr$(80) THEN
    LOCATE star,1: PRINT " ";
    star=star+1
    IF star=8 THEN star=2
    LOCATE star,1: PRINT "*";
  END IF
  IF a$=chr$(0)+chr$(72) THEN
    LOCATE star,1: PRINT " ";
    star=star-1
    IF star=1 THEN star=7
    LOCATE star,1: PRINT "*";
  END IF
  IF a$=chr$(13) THEN       ' prepare to INPUT new data item
    lin=star
    LOCATE lin,14
    PRINT spc(21)
  END IF
  IF a$=chr$(0)+chr$(68) THEN    ' F10: Draws axis & plots
    colr = 10
    pltmax = xkm*1000.0
    GOSUB plotaxes
    GOSUB CALCS
  END IF
  IF a$=chr$(0)+chr$(67) THEN     ' F9: Overplot calcs
    colr = colr + 1
    GOSUB CALCS
  END IF
WEND
END
```

```
PLOTAXES:                 ' ******* Plot and label axes *********
' Set EGA mode of operation
screen 9
view (189,2)-(635,329),,3
window (0,-40)-(pltmax,10)
cls

FOR i=0 to 4   'x tics
    line (0,-30+i*10)-(pltmax/50,-30+i*10)
NEXT i
FOR i=1 to 4   'y tics
    line (i*pltmax/5,-40)-(i*pltmax/5,-39)
NEXT i
LOCATE 25,24
IF (xkm > 999.) THEN
  PRINT "0                                                        ";xkm;
ELSE
  IF (xkm > 99.) THEN
    PRINT "0                                                        ";xkm;
  ELSE
    PRINT "0                                                         ";xkm;
  END IF
END IF
RETURN

CALCS:                   ' ********* perform loss calculations ******

'  Read input files for the Height & M-Unit profiles
OPEN f$ for INPUT as 1
INPUT #1, label$
i=0
DO                          'WHILE not eof(1)
  INPUT #1, zin(i),Munits!(i)
  i=i+1
LOOP UNTIL eof(1)       'WEND
CLOSE #1
numin=i-1

iswap = 0
IF (source > hr) THEN
  iswap = 1
  h1 = hr
  h2 = source
ELSE
  h1 = source
  h2 = hr
END IF

' ....Misc. initialization subroutines and constants
CALL mpqarray(Munits!(), zin(), asubc1, sbdht, h1, numin)
' Calculate effective-earth radius factor
CALL getk(asubc1, h1, rk, rk3deg)

twopi = 2.0 * PI
ht = source
CALL opconst     ' constants for REF sub. and water-vapor absorbtion
CALL dconst      ' constants for the diffraction/intermediate region
CALL antpar      ' initialize antenna pattern parameters
```

```
IF (sbdht > 0.0) AND (ht < sbdht) THEN
  rsbd = 0.0
  rayl = 0.0
  IF (hr < sbdht) THEN
'   calculate skipzone range to top of surface duct
    CALL skiptr(ht,rayl)
  END IF
  CALL skipzone
END IF

Phi = PI          ' phase change due to reflection temporarily set to PI
wlngth = 300./freq    ' wavelength in meters
xmax = xkm*1000.0     ' range in meters
rtol = 0.5            ' tolerance on range for Newton's method
rmin = xmax/20

'      Begin loss versus range calculations
CALL optical
CALL DiffractR(opmaxd, opmaxl, opmaxd, ff)
'      Terminate loss versus range calculations
' CREATE a binary output file for plotting in EREPS
g$ = f$ + ".ray"
OPEN g$ FOR OUTPUT AS #1
PRINT #1, "Rayopr loss vs range output file"
PRINT #1, CHR$(26)
PRINT #1, "EREPS 2.1"
PRINT #1,  f$ + ".ray"
PRINT #1, freq
PRINT #1, "HORIZONTAL"
PRINT #1, source
PRINT #1, "SINX/X"
PRINT #1, 4.0
PRINT #1, 0.0
PRINT #1, 1
PRINT #1, hr
PRINT #1, 1.0
PRINT #1, irng
PRINT #1, 50000.0
PRINT #1, rinc
FOR i = 1 TO irng
  n = CINT(pl(i)*10)
  PRINT #1, MKI$(n);
NEXT
CLOSE #1
' CREATE an ASCII output data file for plotting in EREPS
g$ = f$ + ".asc"
OPEN g$ FOR OUTPUT AS #1
PRINT #1, "#  Rayopt output"
PRINT #1, "#  Earth radius factor =";rk
PRINT #1," m       dB"
rngout = 50000.0 - rinc
FOR i = 1 TO irng
  rngout = rngout + rinc
  PRINT #1, rngout;fsterm + 8.686 * LOG(rngout/1.e3) - pl(i)
NEXT
CLOSE #1
RETURN
```

```
SUB ANTPAR STATIC

'  Process:  Initialize antenna parameters
'  Inputs from common block:  antyp$, bwidth, elevat, PI
'  Outputs to common block:  antbwr,antelr,antfac,elmaxr,patrfac
'  Subroutines called:  None
'  Subroutine called by:  FFACTR

antbwr = .01745 * bwidth
antelr = .01745 * elevat
elmaxr = 1.047
IF (antyp$ <> "OMNI") THEN
   IF (antyp$ = "GAUSS") THEN
      antfac = -LOG(2!) / (2! * SIN(antbwr / 2!)^2)
      patrfac = SIN(antelr)
      amax = SQR(10.11779 * SIN(antbwr / 2!)^2)
      elmaxr = antelr + ATN(amax / SQR(1! - amax * amax))
   ELSE
     IF (antyp$ = "CSC-SQ") THEN
       elmaxr = antelr + .78525
       antfac = SIN(antbwr)
     ELSE
       IF (antyp$ = "SINX/X") OR (antyp$ = "HT-FINDER") THEN
         antfac = 1.39157 / SIN(antbwr / 2!)
         amax = PI / antfac
         patrfac = -ATN(amax / SQR(1! - amax * amax))
         IF (antyp$ = "SINX/X") THEN elmaxr = antelr - patrfac
       END IF
     END IF
   END IF
END IF
END SUB
```

```
SUB ANTPAT(angle, patfac) STATIC

'  Process:  Calculate the normalized antenna pattern factor
'  Inputs from common block:  alpha, antbwr, antelr, antfac
'                             antyp$, patrfac
'  Inputs from argument list:  angle
'  Outputs to common block:  None
'  Outputs to argument list:  patfac
'  Subroutines called:  None
'  Subroutine called by:  OPFFAC

patfac = 1!
IF antyp$ <> "OMNI" THEN
  IF antyp$ = "HT-FINDER" AND alpha > antelr THEN
    alpha0 = alpha
  ELSE                                 ' SINX/X or CSC-SQ or GAUSS
    alpha0 = antelr
  END IF
  apat = angle - alpha0

  IF (antyp$ = "CSC-SQ") THEN
    IF apat > antbwr THEN
      patfac = SIN(antbwr) / SIN(ABS(apat))
    ELSEIF (apat < 0) THEN
      patfac = 1! + apat / antbwr
      IF patfac < .03 THEN patfac = .03
    END IF
  ELSEIF (antyp$ = "GAUSS") THEN
    IF (apat > elmaxr) OR (angle < -elmaxr) THEN
      patfac = .03
    ELSE
      patfac = EXP(antfac * (SIN(angle) - patrfac) ^ 2)
    END IF
  ELSE                              ' antyp$ is SIN(X)/X or HT-FINDER
    IF (apat <> 0!) THEN
      IF (apat <= patrfac) OR (-apat <= patrfac) THEN
        patfac = .03
      ELSE
        ufac = antfac * SIN(apat)
        patfac = SIN(ufac) / ufac
        IF (patfac > 1!) THEN patfac = 1!
        IF (patfac < .03) THEN patfac = .03
      END IF
    END IF
  END IF
END IF
END SUB
```

```
SUB DCONST STATIC

'  Process:  Initialize variables for the diffraction and
'            troposcatter region
'  Inputs from common block:  ae, delta, freq, fsterm, h1, h2,
'                             rk, rnsubs, rn2real, rn2imag
'  Outputs to common block:  atten, c1, c2, c3, c4, c5, c6, c7, capk,
'                            del, difac, dtot, f3, hmin, rnsterm, rns2,
'                            tfac, xterm, zfac, zmax
'  Subroutines called:  HGAIN
'  Subroutine called by:  FFACTR

'    Troposcatter region constants
   rnsterm = .031 - .00232 * rnsubs + 5.67E-06 * rnsubs * rnsubs
   tfac = .08984 / rk
   f3 = freq * freq * freq
   rns2 = .2 * rnsubs

'   CCIR diffraction region model constants
   f13 = freq^(1./3.)
   ae13 = ae^(1./3.)
   eps = rn2real
   sigfac = rn2imag * rn2imag
   rkfac = ((twopi / 0.300)^(-1./3.))/(ae13 * f13)
   rksubh = rkfac * (((eps - 1.0)^2 + sigfac)^(-1./4.))
   betad = 1.0
   capk = rksubh
   IF (polar$ <> "H") THEN
     rksubv = rksubh * SQR(eps * eps + sigfac)
     capk = rksubv
     IF (freq < 300.0) THEN
       capksq = rksubv * rksubv
       betad = 1.0 + 1.6 * capksq + 0.75 * capksq * capksq
       betad = betad / (1.0 + 4.5 * capksq + 1.35 * capksq * capksq)
     END IF
   END IF
   zterm = 9.6e-3 * betad * f13 * f13 / ae13
   xterm = 2.2 * betad * f13 / (ae13 * ae13)

   IF (delta = 0!) THEN              ' no evaporation duct height
      del = 0!
   ELSE

'     Following terms for NOSC evaporation duct model
      rfac = .04705 * f13
      zfac = .002214 * f13 * f13
      hmin = 1!
      del = delta * zfac
      IF (del > 23.3) THEN del = 23.3
      IF (del >= 10.25) THEN

'        Duct height greater than or equal to 10.25 meters
        C1 = -.1189 * del + 5.5495
        C3 = 3! / 2!
        C2 = 1.3291 * SIN(.218 * (del - 10!) ^ .77) + .2171 * LOG(del)
        C2 = C2 * 4.72 ^ (-C3)
        C4 = 87! - SQR(313.29 - (del - 25.3) ^ 2)
        zmax = 4! * EXP(-.31 * (del - 10!)) + 6!
        arg = C2 * zmax ^ C3
```

```
          slope = 4.72 * C1 * C2 * C3 * SQR(zmax) / TAN(arg)
          C7 = 49.4 * EXP(-.1699 * (del - 10!)) + 30!
          fmax = C1 * LOG(SIN(arg)) + C4 - C7
          C6 = (zmax / 4.72) * slope / fmax
          C5 = fmax / zmax ^ C6
        ELSE

'            Duct height less than 10.25 meters
          C2 = SQR(40623.61 - (del + 4.4961) ^ 2) - 201.0128
          C1 = (-2.2 * EXP(-.244 * del) + 17!) * 4.72 ^ (-C2)
          C4 = SQR(14301.2 - (del + 5.32545) ^ 2) - 119.569
'            The # symbol is QuickBASIC double precision notation
          C3 = (-33.9 * EXP(-.5170001# * del) - 3!) * 4.72 ^ (-C4)
          C5 = 41! * EXP(-.41 * del) + 61!
        END IF

        atten = 92.516 - SQR(8608.7593# - (del - 20.2663) ^ 2)
        IF (atten < .0009) THEN atten = .0009
        atten = atten * rfac
        IF (del <= 3.8) THEN tlm = 216.7 + del * 1.5526
        IF (del > 3.8)  THEN tlm = 222.6 - (del - 3.8) * 1.1771
        difac = 51.1 + tlm + 4.343 * LOG(rfac)
     END IF
END SUB
```

```
SUB DIFFRACTR(opmaxd, opmaxl, rnow, diff) STATIC

' Process: Calculate the diffraction and troposcatter region loss
' Inputs from common block: ae, capk, delta, freq, fsterm,
'           rk, rnsubs, rn2real, rn2imag, source, zterm, rinc
' Outputs to common block: atten, c1, c2, c3, c4, c5, c6, c7,
'           del, difac, dtot, difac, f3, hmin, horizn, hormin,
'           h1, h2, h14pil, h24pil, rkmin, rsubd, tsub1, tsub2
' Subroutines called: HGAIN
' Subroutine called by: CALCS

h1 = source
h2 = hr
' Troposcatter constants
tsub1 = SQR(h1 * ae / 500!) / ae
h14pil = h1 * .0419 * freq
tsub2 = SQR(hr * ae / 500!) / ae
h24pil = hr * .0419 * freq
horizn = 3.572 * (SQR(rk * h2) + SQR(rk * h1))

' CCIR diffraction loss constants
yt = zterm * source
CALL gofz(yt, GHT)
yr = zterm * hr
CALL gofz(yr, GHR)
dtot = fsterm - ght - ghr

' NOSC evaporation duct loss constants
CALL hgain(source, dummy, FZT)
dffac = difac - fzt
CALL hgain(hr, FOFZ, FZR)
difac = dffac - fzr

rkmin = rk
IF (rkmin < 1.3333) THEN rkmin = 1.3333
rsubd1 = 3.572 * SQR(rkmin * source) + 230.2 * (rkmin^2 / freq)^(1!/3!)
'  Minimum range for diffraction region calculations
rsubd = 3.572 * SQR(rkmin * hr) + rsubd1
opmax = opmaxd / 1000.0
pltmax = xmax / 1000.0
rnow = rnow / 1000.0
'rnginc = (rsubd - rnow) / 20.0
'***
rnginc = rinc / 1000.0
rnow = cint(rnow*10)/10.0
'***
DO WHILE (rnow < pltmax)
  rnow = rnow + rnginc
  fsloss = fsterm + 8.686 * LOG(rnow)
  IF (rnow > pltmax) THEN rnow = pltmax
  IF (rnow < rsubd) THEN
    ' Calculate propagation loss in intermediate/diffraction region
    CALL difint(opmax, opmaxl, rnow, diff)
  ELSE
    CALL dloss(rnow, diff)
    IF (sbdht > 0) AND (sbdht >= ht) THEN
      CALL sbd(rnow, sbdloss)
      IF (sbdloss < diff) THEN diff = sbdloss
    END IF
    diff = -(diff - fsloss)
  END IF

rnowm = rnow * 1000.0
  line -(rnowm, diff),colr
  '****
    IF (rnow > 50.0) AND (rnow < pltmax) THEN
       irng = irng + 1
       rng(irng) = rnowm
       pl(irng) = fsloss - diff
    END IF
LOOP
END SUB
```

```
SUB DIFINT(opmaxd, opmaxl, r, ff) STATIC

'  Process:  Calculates 20 times the (base 10) logarithm for the
'            propagation factor within the intermediate region,
'            i.e. for ranges greater than opmaxd and less than
'            rsubd.
'  Inputs from common block:  fsloss, fsterm, ht, rsubd, sbdht
'  Inputs from argument list: opmaxd, opmaxl, r
'  Outputs to common block:  None
'  Outputs to argument list: ff
'  Subroutines called:  DLOSS, SBD
'  Subroutine called by:  FFACTR

   CALL dloss(rsubd, diff)
   diff = -(diff - fsterm - 8.686 * LOG(rsubd))
   deltaf = (r - opmaxd) * (opmaxl - diff) / (opmaxd - rsubd)
   ff = opmaxl + deltaf
   IF (sbdht > 0!) AND (sbdht >= ht) THEN
     tenlgr = 4.343 * LOG(r)
     dtemp = ff + fsloss
     CALL sbd(r, sbdloss)
     IF (sbdloss < dtemp) THEN dtemp = sbdloss
     ff = dtemp - fsloss
   END IF
END SUB
```

```
SUB DLOSS(r, diff) STATIC

'  Process:  Calculate the diffraction region loss
'  Inputs from common block:  atten, delta, difac, dtot, fsterm
'                             xterm
'  Inputs from argument list:  r, tloss
'  Outputs to common block:  None
'  Outputs to argument list:  diff, r
'  Subroutines called:  TROPO
'  Subroutine called by:  FFACTR, DIFINT

'  Calculate diffraction region loss using CCIR model
   tenlgr = 4.343 * LOG(r)
   x = xterm * r
   fofx = 11! + 4.343 * LOG(x) - 17.6 * x
   diff = dtot + 2! * tenlgr - fofx
   IF (delta <> 0!) THEN
      diffe = difac + tenlgr + atten * r
      IF (diffe < diff) THEN

'        Use lesser of CCIR and NOSC evap. duct models
        diff = diffe
      END IF
   END IF
   diff = diff + exloss
   CALL tropo(r, tloss)

'  Add the troposcatter loss to the diffraction loss
   dif = diff - tloss
   IF (dif >= 18!) THEN
      diff = tloss
   ELSEIF (dif >= -18!) THEN
      diff = diff - 4.343 * LOG(1 + EXP(dif / 4.343))
   END IF
END SUB
```

```
SUB FFACTR(alphad,alphar,betad,betar,dxdad,dxdar,patd,psi,rnow,theta,FF)
STATIC
'  Process:  Calculates 20 times the (base 10) logarithm of the
'            propagation factor within the optical interference region,
'  Inputs from common block:  rmag
'  Inputs from argument list:  alphad,alphar,betad,betar,dxdad,dxdar,
'                              psi,rnow,theta
'  Outputs to common block:  None
'  Outputs to argument list: .ff, patd
'  Subroutines called:  ANTPAT, RUFF
'  Subroutine called by:  OPTICAL

  IF (iswap = 0) THEN
    CALL antpat(alphad, PATD)
    CALL antpat(alphar, PATR)
  ELSE
    CALL antpat(-betad, PATD)
    CALL antpat(-betar, PATR)
  END IF
  sinpsi = SIN(psi)
  CALL RUFF(sinpsi, RUF)
  dzdad = -TAN(betad) * dxdad
  dzdar = -TAN(betar) * dxdar
  divfacd = ABS(rnow * COS(betad)/dzdad)
  IF (ABS(divdlst - divfacd) > .2) THEN
    IF (divfacd < .60) THEN divfacd = .60
    IF (divfacd > divdlst) THEN divfacd = divdlst
  END IF
  fdsqd = patd * patd * divfacd
  drsq =  rmag * rmag * ruf * ruf * patr * patr
  divfacr = ABS(rnow * COS(betar)/dzdar)
  frsqd = drsq * divfacr
  ff = fdsqd + frsqd + 2.0 * SQR(fdsqd * frsqd) * COS(theta)
END SUB
```

```
SUB GETK(asubcl, hxmtr, rk, rk3deg) STATIC

'  Process:  Determine an Effective Earth Radius factor, rk, by tracing
'            a ray to 200 naut. mi. and comparing the height at that
'            range to the height that a ray would reach in a standard
'            atmosphere.  Also calculate rk3deg for a ray launch angle
'            of 3 degrees.
'  Inputs from common block:  imin, M!(*)
'  Inputs from argument list:  asubcl, hxmtr
'  Outputs to common block:  None
'  Outputs to argument list:  rk, rk3deg
'  Subroutines called:  RAYPOS
'  Subroutine called by:  CALCS

shared zt(), M!(), p(), q(), imin, istart, levels, xmax

' Trace a ray at 0 degrees (no duct) or at critical angle, to 200 nmi
xmax = 3.7e5
angle = SQR(ABS(2.e-6*(M!(imin) - M!(0)))) + 1.e-8
CALL RAYPOS(angle, xmax, z, raytype)
grad = -(2000.0 * angle)/370.0 + (2.0 * z)/136900.0
rk = 1.0/(grad * 6.3710)

IF (rk > 5.0) THEN rk = 5.0
IF (rk < 1.0) THEN rk = 1.0

' Trace a ray at 3 degrees elevation angle to 200 nmi, determine
' effective earth radius at this angle.
angle = 0.052360
xmax = 3.7e5
CALL RAYPOS(angle, xmax, z, raytype)
grad = -(2000.0 * angle)/370.0 + (2.0 * z)/136900.0
rk3deg = 1.0/(grad * 6.3710)

IF (rk3deg > 5.0) THEN rk3deg = 5.0
IF (rk3deg < 1.0) THEN rk3deg = 1.0

END SUB
```

```
SUB GOFZ(z, G) STATIC

'  Process:  Calculate the diffraction region height-gain in dB for
'            the CCIR diffraction region model

'  Inputs from common block:  capk
'  Inputs from argument list:  z
'  Outputs to common block:  None
'  Outputs to argument list:  G
'  Subroutines called:  None
'  Subroutine called by:  DCONST

  IF (z > 2!) THEN
    G = 17.6 * SQR(z - 1.1) - 2.1715 * LOG(z - 1.1) - 8!
  ELSE
    IF (z > 10! * capk) THEN
      G = 8.686 * LOG(z + .1 * z * z * z)
    ELSE
      G = 2! + 8.686 * LOG(capk)
      IF (z > capk / 10!) THEN
        zkfac = .4343 * LOG(z / capk)
        G = G + 9! * zkfac * (zkfac + 1!)
      END IF
    END IF
  END IF
END SUB
```

```
SUB HGAIN(h, fzdb1, fzdb2) STATIC

'  Process:  Calculates height-gain factor in dB for a specified
'            height
'  Inputs from common block:  c1, c2, c3, c4, c5, c6, c7, del
'            freq, h, hmin, sbdht, zfac, zmax
'  Inputs from argument list:  h
'  Outputs to common block:  None
'  Outputs to argument list: fzdb1, fzdb2
'  Subroutines called: None
'  Subroutine called by: DCONST, SKIPZONE

   fzdb1 = 0!
   fzdb2 = 0!

'  Surface-based duct height-gain factor
   IF (sbdht > 0!) THEN
      z1 = h / sbdht
      IF((freq <= 150!)AND(z1 < .8))THEN fzdb1 = -60! * (z1 - .5) ^ 2
      IF((freq <= 150!) AND (z1 >= .8))THEN
        fzdb1 = 1.14 * z1 ^ (-6.26) - 10!
      END IF
      IF((freq > 150!)AND(z1 < 1!))THEN fzdb1=10! - 200! * (z1 - .5) ^ 4
      IF((freq > 150!)AND freq <= 350!)AND z1 >= 1!))THEN
        fzdb1 = 7.5 * z1 ^ (-13.3) - 10!
      END IF
      IF (freq > 350!)AND z1 >= 1!))THEN fzdb1 = 12.5 * z1 ^ (-8!) - 15!
   END IF

'  Evaporation duct height-gain factor
   IF (del > 0!) THEN
       z2 = h * zfac
       IF (z2 < hmin) THEN z2 = hmin
       IF (del >= 10.25) THEN
         IF (z2 > zmax) THEN
           fzdb2 = C5 * (z2 ^ C6) + C7
         ELSE
           fzdb2 = C1 * LOG(SIN(C2 * (z2 ^ C3))) + C4
         END IF
       ELSE   ' del < 10.25
         fzdb2 = (C1 * z2 ^ C2) + (C3 * z2 ^ C4) + C5
       END IF
   END IF
END SUB
```

```
SUB MPQARRAY(Munits!(1), zin(1), asubcl, sbdht, hxmtr, numin) STATIC

'  Process:  Builds the M,P,Q and st arrays for use by the raytrace sub-
'            routines.  This routine also places the transmitter height
'            and M-unit value in the M!(*) and zt(*) arrays.
'  Inputs from common block:  delta
'  Inputs from argument list:  hxmtr, Munits!(*), numin, zin(*)
'  Outputs to common block:  imin, istart, levels, M!(*), rnsubs, zt(*)
'  Outputs to argument list:  asubcl, sbdht
'  Subroutines called:  None
'  Subroutine called by:  CALCS

shared zt(), M!(), p(), q(), imin, istart, levels, xmax

rnsubs = Munits!(0)     '  used in Troposcatter routines
'build zt and M! arrays to include transmitting antenna height
istart = 0
j = 0
FOR i=0 TO numin
  zt(j) = zin(i)
  M!(j) = Munits!(i)
  IF (hxmtr = zin(i)) THEN istart=i
  IF (hxmtr > zin(i)) AND (hxmtr < zin(i+1)) THEN
    j = j+1
    zt(j) = hxmtr
    M!(j)=Munits!(i)+(Munits!(i+1)-Munits!(i))*(hxmtr-zin(i))/(zin(i+1)-
zin(i))
    istart = j
  END IF
  j = j+1
NEXT i
levels = j-2

'build p and q arrays for raytrace subroutines
FOR i=0 TO levels
  p(i) = 1.e-6*(M!(i+1) - M!(i))/(zt(i+1) - zt(i))
  q(i) = 2.e-6*(M!(i+1) - M!(i))
NEXT i

' find profile minimum M-value & its index
Mmin! = M!(0)
imin = 0
FOR i=1 TO levels
  IF (M!(i) < Mmin!) THEN
    Mmin! = M!(i)
    imin = i
  END IF
NEXT i

'  IF minimum on profile is not at the surface and is not equal to the
'  evaporation duct height, it defines the surface-based duct height.
IF ((imin <> 0) AND (imin > istart)) THEN
  asubcl = SQR(ABS(2.e-6*(M!(imin)-M!(istart)))) + 1.e-6
  IF (zt(imin) <> delta) THEN sbdht = zt(imin)
ELSE
  asubcl = 0.0
  sbdht = 0.0
END IF
END SUB
```

```
SUB OPCONST STATIC

'  Process:  Initializes constants for optical region
'  Inputs from common block:   antype$, freq, humid, polar$, rk, wind
'  Outputs to common block:  ae, ae2, aeth, fsterm, hbar, hbfreq,
'                            rn2imag, rn2real, thefac, wvatten
'  Subroutines called:  None
'  Subroutine called by: FFACTR


'  Variables for reflection coefficient subroutines.  Also used
'    in the diffraction region subroutine.
   IF (freq <= 1500.0) THEN
      eps = 80.0                            ' Salt water permittivity
      sigma = 4.3                           ' Salt water conductivity
   ELSEIF (freq <= 3000.0) THEN
      eps = 80.0 - .00733 * (freq - 1500.0)
      sigma = 4.3 + .00148 * (freq - 1500.0)
   ELSEIF (freq <= 10000.0) THEN
      eps = 69.0 - .00243 * (freq - 3000.0)
      sigma = 6.52 + .001314 * (freq - 3000.0)
   ELSE  ' freq > 10000
      eps = 51.99
      sigma = 15.718
   END IF

'  Real & imaginary part of square of index of refraction
   rn2real = eps
   rn2imag = (-18000.0) * sigma / freq

'  Variables for RUFF subroutine
   hbar = .0051 * (.51477 * wind)^2         ' Rms wave height
   hbfreq = .02094 * freq * hbar            ' (hbar*2*Pi)/wavelength
'  Variables for miscellaneous subroutines
   ae = rk * 6371                         ' Effective earth radius in km
   twoae = 2! * ae
   aeth = rk * 6.371
   ae2 = aeth * 2!
   thefac = freq * 4.193E-05                ' 4*Pi / wavelength
   fsterm = 32.45 + 8.686 * LOG(freq)       ' Free space loss term

'  CCIR model water vapor attenuation rate constants
   freqg = freq / 1000!                     ' Frequency in GHz
   wv1 = 3.6 / ((freqg - 22.2)^2 + 8.5)
   wv2 = 10.6 / ((freqg - 183.3)^2 + 9!)
   wv3 = 8.9 / ((freqg - 325.4)^2 + 26.3)

'  water vapor attenuation rate in dB/km
   wvatten = (0.050 + 0.0021*humid + wv1 + wv2 + wv3)
   wvatten = wvatten * freqg * freqg * humid / 10000!

END SUB
```

```
SUB OplimitR(ad, admin, ar, armin, psimin, source, rmax, thlimit) STATIC

'  Process:  Determine the greatest range that direct & reflected rays
'            can interfere, excluding trapped rays.  This range is the
'            optical region maximum range for the two terminal heights.
'  Inputs from common block:  asubcl, freq, h2, imin, istart, PI,
'                             qwvlgth, rk, twopi
'  Inputs from argument list:  source
'  Outputs to common block:
'  Outputs to argument list:  ad, admin, ar, armin, psimin, rmax,
'                             thlimit
'  Subroutines called:  RAYR, RALOOPR, REF
'  Subroutine called by:  CALCS


qwvlgth = 1.5 * PI
rmax = 0.0
admin = 0.0
armax = 0.0

'define limits on height and angles
psilim = 0.01957/(rk*freq)^0.3333 'grazing angle limit - Reed & Russell
psimin = SQR(ABS(2.e-6*(M!(0) - Mmin!))) + 1.e-5  'minimum grazing angle
'IF (psilim < psimin) THEN psimin = psilim

IF (asubcl > 0.0) THEN
' xmtr is in a surface-based duct or below profile min. of evap. duct
  ar = -asubcl - 2.e-6
  ad = asubcl
  FLAG% = 0
  CALL RAYR(ad,h2,betad,dxdad,pldd,psi,raytype,FLAG%,ifail,rngd)
  FLAG% = 1
  CALL RAYR(ar,h2,betar,dxdar,pldr,psi,raytype,FLAG%,ifail,rngr)
ELSE
' test for sub-refractive layers below xmtr height
  IF (istart > 1) THEN
    ac1 = 0.0
    ac2 = 0.0
    isub = 0
    FOR i=0 TO istart-1
      IF (p(i) > 1.57e-7) THEN
        ac1 = -1.e-3*SQR(2.0*ABS((M!(istart)-M!(i+1))))+1.e-6
        ac2 = -1.e-3*SQR(2.0*ABS((M!(istart)-M!(i))))+1.e-6
        isub = i
      END IF
    NEXT i
  END IF
  IF (isub <> 0) THEN
    ' transmitter above a sub-refractive layer
    CALL RAYR(ac1,h2,betad,dxdad,pldd,psi,raytype,0,ifail,rmax)
    adiff = (ac1 - ac2)/10.0
    angle = ac1
   ' loop to determine approx. angular limit for direct-ray interference
    FOR j=1 TO 10
      angle = angle - adiff
      CALL RAYR(angle,h2,betad,dxdad,pldd,psi,raytype,0,ifail,rngd)
      IF (rngd < rmax) THEN rmax = rngd
    NEXT j
    dzdad = -TAN(betad) * dxdad
    admin = acl
    armax = -SQR(ABS(psimin*psimin + 2.e-6*(M!(istart) - M!(0))))
    CALL RAYR(armax,h2,betar,dxdar,pldr,psi,raytype,1,ifail,rngr)
    dzdar = -TAN(betar) * dxdar
 ELSE
    '  xmtr not in trapping layer or above sub-refractive layer
    ad = SGN(imin-istart)*SQR(ABS(2.e-6*(M!(imin) - M!(istart))))+ 1.e-6
    CALL RAYR(ad,h2,betad,dxdad,pldd,psi,raytype,0,ifail,rngd)
    IF (ifail = 1) THEN
      id = 0
      DO
        id = id + 1
        ad = ad + 1.e-8
        CALL RAYR(ad,h2,betad,dxdad,pldd,psi,raytype,0,ifail,rngd)
      LOOP UNTIL (ifail = 0) OR (id = 11)
    END IF
    ar = -SQR(ABS(psimin*psimin + 2.e-6*(M!(istart) - M!(0))))
    CALL RAYR(ar,h2,betar,dxdar,pldr,psi,raytype,1,ifail,rngr)
    IF (ifail = 1) THEN
      id = 0
      DO
        id = id + 1
        ar = ar - 1.e-8
        CALL RAYR(ar,h2,betar,dxdar,pldr,psi,raytype,1,ifail,rngr)
      LOOP UNTIL (ifail = 0) OR (id = 11)
    END IF
    psi = psimin
  END IF
END IF
IF (rngr < rngd) THEN
  rmax = rngr
  CALL RALOOPR(ad,betad,dxdad,0,rmax,rngd,dzdad,psi,rtol,pldd,raytype)
ELSE
  rmax = rngd
  CALL RALOOPR(ar,betar,dxdar,1,rmax,rngr,dzdar,psi,rtol,pldr,raytype)
END IF
armax = ar
admin = ad

'CALL REF(p$, psi, PHI)
wldif = (pldr - pldd)/wlngth
thlimit = twopi*wldif + PI
thsave = thlimit
IF (thlimit > twopi) AND (imin > 0) THEN
  thlimit = INT(thlimit/twopi + 1)*twopi
  'thlimit = thsave
ELSE
  ' Approximate quarter-wavelength limit by qwvlgth until REF subroutine
  ' is used.  Then substitute PHI for PI and set thlimit = PI/2 + PHI
  IF (wldif > 0.250) AND (psi = psilim) THEN thlimit = qwvlgth
  IF (wldif < 0.250) AND (psi <> psilim) THEN thlimit = qwvlgth
  IF (del > 10.25) THEN
    thlimit = twopi
  ELSE
    thlimit = thlimit + del/10.25 * (twopi - thlimit)
  END IF
END IF

IF (thsave < thlimit) THEN    'find shorter range, rmax, corresp. to
limit.
```

```
    delr = - rmax/10.
    ij = 0
    DO
      ij = ij + 1
      IF (-delr > rmax) THEN delr = delr/2.0
      rmax = rmax + delr
      CALL RALOOPR(ad,betad,dxdad,0,rmax,rngd,dzdad,psi,rtol,pldd,raytype)
      CALL RALOOPR(ar,betar,dxdar,1,rmax,rngr,dzdar,psi,rtol,pldr,raytype)
      'CALL REF(p$, psi, PHI)
      theta = twopi*(pldr - pldd)/wlngth + PI
      IF (theta > thlimit) THEN
        theta = thsave
        rmax = rmax - delr
        delr = delr / 2.0
      END IF
      thsave = theta
    LOOP UNTIL (ABS(delr) < 1.0) AND (thlimit <= theta) OR (ij > 25)
  END IF

  END SUB
```

**SUB OPTICAL STATIC**

```
'  Process:  Calculate loss versus range in the optical interference
'            region.
'  Inputs from common block:  ae, fs*erm, h1, h2, PI, twopi, xmax
'  Inputs from argument list:  None
'  Outputs to common block:  exloss, opmaxd, opmaxl
'  Outputs to argument list:  None
'  Subroutines called:  FFACTR, OPLIMITR, RAYR, RALOOPR, RE
'  Subroutine called by:  CALCS

CALL OplimitR(ad, admin, ar, armin, psimin, source, opmax, thlimit)
rmin = xmax/20.0
'rmin = xmax/4.0    ' temporary rmin to shorten run time
IF (opmax > rmin) THEN
  ad = (h2 - h1)/rmin - rmin/(2000.*ae)  'flat-earth approx's to start
  ar =-(h2 + h1)/rmin
  CALL RAYR(ad,h2,betad,dxdad,pldd,psi,raytype,0,ifail,rngd)
  dzdad = -TAN(betad) * dxdad
  divdlst = ABS(rngd * COS(betad)/dzdad)
  CALL RALOOPR(ad,betad,dxdad,0,rmin,rngd,dzdad,psi,rtol,pldd,raytype)
  CALL RAYR(ar,h2,betar,dxdar,pldr,psi,raytype,1,ifail,rngr)
  dzdar = -TAN(betar) * dxdar
  CALL RALOOPR(ar,betar,dxdar,1,rmin,rngr,dzdar,psi,rtol,pldr,raytype)
ELSE
  CALL RAYR(ad,h2,betad,dxdad,pldd,psi,raytype,0,ifail,rngd)
  dzdad = -TAN(betad) * dxdad
  divdlst = ABS(rngd * COS(betad)/dzdad)
  CALL RAYR(ar,h2,betar,dxdar,pldr,psi,raytype,1,ifail,rngr)
  dzdar = -TAN(betar) * dxdar
  rmin = opmax
END IF
wldif = (pldr - pldd)/wlngth
theta = twopi * wldif + PI
rnow = rmin
rnow = cint(rmin) * 1.0

rlast = rngd

' Main optical region plot LOOP
irng = 0
DO
  CALL RALOOPR(ad,betad,dxdad,0,rnow,rngd,dzdad,psi,rtol,pldd,raytype)
  CALL RALOOPR(ar,betar,dxdar,1,rnow,rngr,dzdar,psi,rtol,pldr,raytype)
  wldif = (pldr - pldd)/wlngth
  'CALL REF(p$, psi, PHI)
  phi = PI        ' temporary value remove when call to Ref is used
  rmag = 1.0      ' temporary value remove when call to Ref is used
  theta = wldif * twopi + phi
  IF (theta >= thlimit) OR (rnow < opmax) THEN
    CALL ffactr(ad,ar,betad,betar,dxdad,dxdar,patd,psi,rnow,theta,FF)
    ffac = -4.343*LOG(ff)                          ' 20*LOG10(F-factor)
    IF (rnow = rmin) THEN
      PSET (rnow,-ffac),colr
    ELSE
      LINE -(rnow,-ffac),colr
    END IF
    IF (rnow > 50000.0) THEN
       irng = irng + 1
```

```
            rng(irng) = rnow
            pl(irng) = fsterm + 8.686 * LOG(rnow/1.e3) + ffac
        END IF
        divdlst = divfacd
      ELSE
        rnow = rlast
        theta = thlimit
      END IF
      rlast = rnow
      IF (rnow = opmax) THEN theta = thlimit
      rnow = rnow + rinc
      IF (rnow > opmax) THEN
        rnow = opmax
        theta = thlimit
      END IF
      a$ = inkey$
    LOOP WHILE (theta > thlimit) AND (a$ <> chr$(27)) AND (rnow <= xmax)

    IF (rnow < xmax) THEN
      exloss = 8.686 * LOG(patd)
      opmaxd = rnow
      opmaxl = -ffac
    END IF

    END SUB
```

```
SUB
RALOOPR(alpha,beta,dxda,flag%,rtgt,rnow,dzda,psi,rtol,pld,raytype)STATIC

' Process:  Raloopr determines the launch angle necessary to place a ray
'           thru the target/rcvr height (hr) at the range rtgt.  flag% =
'           1 if the ray is a reflected ray and flag% = 0 if the ray is
'           a direct ray.  Raytype must be equal to flag%.  z is actual
'           ray height in iteration loop.
'  Inputs from arguement list:  alpha, h2, raytype, rnow, rtrgt, rtol,
'                               zr, flag%
'  Inputs from common block:   zt(),M!(),p(),q(),istart,levels
'  Outputs to common block:  none
'  Outputs to arguement list: beta, dxda, pld, psi, pld
'  Subroutines called:  None
'  Subroutine called by:  OPLIMTR, OPTICAL

  jcount = 0
  DO
    jcount = jcount + 1
    anglast = alpha
    rlast = rnow
    IF (dxda = 0.0) THEN dxda = 1.e-7
    delang = (rnow - rtgt) / dxda
    IF(ABS(delang)>.17453) THEN delang=SGN(delang)*.17453 'limit to 10°
    alpha = alpha - delang
    IF (flag% = 1) AND (alpha >= 0.0) THEN alpha = -alpha
    CALL RAYR(alpha,h2,beta,dxda,pld,psi,raytype,flag%,ifail,rnow)
    IF (raytype <> flag%) OR (ifail = 1) THEN
      iloop = 0
      DO
        iloop = iloop + 1
        delang = delang / 2.
        IF (delang = 0.0) THEN delang = 1.e-8
        alpha = alpha + delang
        IF (flag% = 1) AND (alpha >= 0.0) THEN alpha = -alpha
        CALL RAYR(alpha,h2,beta,dxda,pld,psi,raytype,flag%,ifail,rnow)
      LOOP WHILE (raytype <> flag%) AND (iloop < 15)
    END IF
  LOOP WHILE((ABS(rnow-rtgt)>rtol)AND(raytype=flag%)AND(jcount<11))
  IF (jcount = 11) THEN
    IF((rnow<rtgt)AND(rlast>rtgt))OR((rnow>rtgt)AND(rlast<rtgt))THEN
'   loop executed when bouncing on either side of desired range, rtgt
      delang = ABS(delang)/10.0
      IF (rnow > rtgt) THEN alpha = anglast
      ij = 0
      DO
        ij = ij + 1
        alpha = alpha - delang
        CALL RAYR(alpha,h2,beta,dxda,pld,psi,raytype,flag%,ifail,rnow)
        IF (rnow > rtgt) THEN
          IF (ABS(rnow-rtgt) > rtol) THEN
            delang = delang / 2.0
            alpha = anglast
          END IF
        END IF
        anglast = alpha
      LOOP WHILE((ABS(rnow-rtgt)>rtol)AND(raytype=flag%)AND(ij<25))
      LOCATE 20,1
      PRINT  "A"
```

```
        tim = timer
        do
        loop until timer-tim > .5
        LOCATE 20,1
        PRINT " "
     ELSE      '(rnow and rlast are both greater than or less than rtgt)
       ij = 0
       DO
         ij = ij + 1
         IF (rnow <> rlast) THEN
           delang = (rnow - rtgt) * (alpha - anglast) / (rnow - rlast)
         ELSE
           delang = 1.e-8
         END IF
anglast = alpha
rlast = rnow
IF (ABS(delang)>.17453) THEN delang = SGN(delang)*.17453  '10° limit
  alpha = alpha - delang
  IF (raytype = 1) THEN
    IF (alpha > armax) THEN alpha = armax
    ELSE    ' direct ray: raytype = 0
    IF (alpha < admin) THEN alpha = admin
  END IF
  CALL RAYR(alpha,h2,beta,dxda,pld,psi,raytype,flag%,ifail,rnow)
  IF (raytype <> flag%) OR (ifail = 1) THEN
  iloop = 0
  DO
    iloop = iloop + 1
    delang = delang / 2.
    IF (delang = 0.0) THEN delang = 1.e-8
    alpha = alpha + delang
    CALL RAYR(alpha,h2,beta,dxda,pld,psi,raytype,flag%,ifail,rnow)
    IF (flag% = 1) AND (alpha >= 0.0) THEN alpha = -alpha
  LOOP WHILE (raytype <> flag%) AND (iloop < 15)
END IF
      LOOP WHILE ((ABS(rnow-rtgt)>rtol)AND(raytype=flag%)AND(ij<11))
      LOCATE 20,1
      PRINT  "B"
      tim = timer
      DO
      LOOP until timer-tim > .5
      LOCATE 20,1
      PRINT " "
    END IF
    IF (ABS(rnow-rtgt) > rtol) OR (raytype <> flag%) THEN
      LOCATE 18,1     ' solution does not converge!
      PRINT "rnow ";rnow
      LOCATE 19,1
      PRINT "rtgt ";rtgt
    END IF
  END IF
  dzda = -TAN(beta)*dxda
END SUB
```

```
SUB RAYPOS(a, xmax, zr, raytype) STATIC

'  Process:  Traces a ray  with a launch angle, a, from range = 0. and
'            height = 0. to range xmax.
'  Inputs from arguement list:  a, raytype, xmax
'  Inputs from common block:   zt(),M!(),p(),q(),istart,levels
'  Outputs to common block:  none
'  Outputs to arguement list: zr
'  Subroutines called:  None
'  Subroutine called by:  MPQARRAY

shared zt(),M!(),p(),q(),istart,levels

'initial conditions at start of ray
a0 = a
i = 0
x = 0.0
raytype = 0

DO WHILE (x < xmax)
  IF i=levels THEN                        ' Final step in upper level
    a1 = a0 + (xmax - x)*p(i)
    zr = zt(i) + (a1*a1 - a0*a0)/(2.*p(i))
    x = xmax
  ELSE
    rad = a0*a0 + q(i)
    IF (rad < 0.0) THEN                ' Ray doesn't escape trapping layer
      raytype = 1
      x = xmax
    ELSE
      a1 = SQR(rad)
      xtemp = x + (a1 - a0)/p(i)
      IF (xtemp < xmax) THEN              ' Full step thru layer
        x = xtemp
        a0 = a1
        i = i+1
      ELSE                                ' Final step in layer
        a1 = a0 + (xmax - x)*p(i)
        zr = zt(i) + (a1*a1 - a0*a0)/(2.*p(i))
        x = xmax
      END IF
  END IF
END IF
LOOP   ' END main loop
END SUB              ' ***** end raypos subroutine *****
```

```
SUB RAYR(a,zr,a1,dxda,pld,psi,raytype,FLAG%,ifail,x) STATIC

'  Process:  RAYR traces a ray to a specified height, zr, using initial
'            ray launch angle a.   RAYR returns the range to height zr,
'            x.  It also returns the ending angle, a1, the path-length
'            difference between the ray path and the ground range in
'            radians, pld, the grazing angle, psi, and the derivative of
'            range with respect to angle, dxda.  RAYR also sets the
'            failure flag, ifail = 1, to indicate that the ray could not
'            reach zr.
'  Inputs from arguement list:  a, raytype, zr, flag%
'  Inputs from common block:   zt(),M!(),p(),q(),istart,levels
'  Outputs to common block:  none
'  Outputs to arguement list: a1, dxda, pld, psi, ifail, x
'  Subroutines called:  None
'  Subroutine called by:  OPLIMTR, OPTICAL, RALOOPR

shared zt(),M!(),p(),q(),istart,levels
'  set initial conditions at start of ray
ifail = 0                               ' ifail = 1 if ray doesn't reach zr
a0 = a
IF (a0 = 0.0) THEN a0 = 1.e-10
i = istart
x = 0.0
xtemp = 0.0
dxda = 0.0
dxtemp = 0.0
pld = 0.0
pltemp = 0.0
z = zt(istart)
raytype = 0
zmin = 0.0
DO            'WHILE (i <= levels) AND (z <> zr) AND (ifail = 0)
  IF (a0 >= 0.) THEN
'            ***************   Upgoing rays, z < zr  *************
    a0sq = a0 * a0
    IF (i < levels) THEN
      rad = a0sq + q(i)
      IF (rad < 0.) THEN
        ifail = 1             ' don't trace trapped rays
        EXIT DO
      ELSE
        IF (zt(i+1) < zr) THEN
          a1 = SQR(rad)
          x = x + (a1 - a0)/p(i)
          z = zt(i+1)
        ELSE
          a1 = SQR(a0sq + 2.0*p(i)*(zr - z))
          x = x + (a1 - a0)/p(i)
          z = zr
        END IF
      END IF
    ELSE
      rad = a0sq + 2.0*p(i)*(zr - z)
      IF (rad < 0.) THEN
        ifail = 1             ' don't trace trapped rays
        EXIT DO
      ELSE
        a1 = SQR(rad)
```

```
          x = x + (a1 - a0)/p(i)
          z = zr
        END IF
      END IF
      temp = (a/a1 - a/a0)/p(i)
      dxda = dxda + temp
      pld = pld + ((1.e-6*M!(i)-a0sq/2.)*(a1-a0)+(a1^3-a0sq*a0)/3.)/p(i)
      a0 = a1
      i = i+1
    ELSE
'       *********************** downgoing rays *********************
      a0sq = a0 * a0
      rad = a0sq - q(i-1)
      IF (rad > 0!) THEN
        a1 = -SQR(rad)
        xtemp = (a1 - a0)/p(i-1)
        x = x + xtemp
        z = zt(i-1)
      ELSE                                ' Ray reaches minimum in layer
      ' ray reaches minimum in layer and exits at top of layer
        xtemp = - 2.0 * a0 / p(i-1)
        x = x + xtemp                     'Range to top of layer i-1
        a1 = - a0
        zmin = zt(i) - a0 * a0 / (2.0*p(i-1))
      END IF
      IF (z = 0!) THEN
        IF (FLAG% = 0) THEN
          ifail = 1
          EXIT DO
        END IF
      END IF
      dxtemp = (a/a1 - a/a0)/p(i-1)
      dxda = dxda + dxtemp
      pltemp = ((1.e-6*M!(i)-a0sq/2.)*(a1-a0)+(a1^3-a0sq*a0)/3.)/p(i-1)
      pld = pld + pltemp
      a0 = a1
      IF (a0 < 0.) THEN i = i-1
      IF (z = 0!) THEN
        psi = - a1
        raytype = 1
        dxda = 2.0 * dxda
        pld = 2.0 * pld
        a0 = - a
        i = istart
        x = 2.0 * x
        z = zt(istart)
      END IF
      IF (zmin > 0.0) THEN
      '  Use symmetry to eliminate tracing ray back to starting height.
        pld = 2.0 * (pld - pltemp) + pltemp
        dxda = 2.0 * (dxda - dxtemp) + dxtemp
        x = 2.0 * (x - xtemp) + xtemp
        a0 = -a
        i = istart
        z = zt(istart)
      END IF
    END IF
  LOOP WHILE ((i <= levels) AND (z <> zr))
  END SUB
```

```
SUB REF(p$, psi, phi) STATIC

'  Process:  Calculates magnitude, rmag, and phase lag, phi, of
'            the reflection coefficient
'  Inputs from common block:  pi, rn2imag, rn2real
'  Inputs from argument list:  p$, psi
'  Outputs to common block: rmag
'  Outputs to argument list:  phi
'  Subroutines called:  None
'  Subroutine called by:  GTHETA, OPTICF

   rmag = 1!
   phi = PI
   rch = rmag
   phih = phi
   IF (p$ <> "H") THEN
      sinpsi = SIN(psi)
      y = rn2imag
      x = rn2real - COS(psi)^2
      rmagroot = (x * x + y * y)^.25
      angroot = ATN(y / x) / 2!
      rootreal = rmagroot * COS(angroot)
      rootimag = rmagroot * SIN(angroot)
      at = rn2real * sinpsi - rootreal
      ct = rn2real * sinpsi + rootreal
      bt = rn2imag * sinpsi - rootimag
      dt = rn2imag * sinpsi + rootimag
      refvreal = (at * ct + bt * dt) / (ct * ct + dt * dt)
      refvimag = (bt * ct - at * dt) / (ct * ct + dt * dt)
      rcv = SQR(refvreal * refvreal + refvimag * refvimag)
      IF (refvreal <> 0!) THEN
        phiv = ATN(refvimag / refvreal)
        IF (refvreal < 0!) THEN phiv = phiv + PI
      ELSE
        IF (refvimag < 0!) THEN phiv = -PI / 2!
        IF (refvimag > 0!) THEN phiv = PI / 2!
        IF (refvimag = 0!) THEN phiv = 0!
      END IF
      phiv = -phiv
      IF (phiv < 0!) THEN phiv = phiv + 2! * PI
      rmag = rcv
      phi = phiv
      IF (p$ = "C") THEN
         rx= rch * rch + rcv * rcv + (2! * rcv * rch * COS(phih - phiv))
         rx = SQR(rx)
         rmag = rx / 2!
         a = rcv * SIN(phiv + phih) / rx
         a = ATN(a / SQR(1 - a * a))
         phi = phih - a
         phi = -phi
         IF (phi < 0!) THEN phi = phi + 2! * PI
       END IF
   END IF
END SUB
```

```
SUB RUFF(sinpsi, ruf) STATIC

'  Process:  Calculates the surface-roughness coefficient as a
'            function of psi
'  Inputs from common block:  hbar, hbfreq, psi
'  Inputs from argument list:  sinpsi
'  Outputs to common block:  None
'  Outputs to argument list:  ruf
'  Subroutines called:  None
'  Subroutine called by:  OPFFAC

   ruf = 1!
   IF (hbar <> 0!) THEN
      hfpsi = hbfreq * psi * .159155
      IF (hfpsi <= 0.11) THEN
        ruf = EXP( -2! * hbfreq * sinpsi * hbfreq * sinpsi )
      ELSEIF (hfpsi <= 0.26) THEN
        ruf = .5018913# - SQR(.2090248# - (hfpsi - .55189)^2)
      ELSE
        ruf = .15
      END IF
   END IF
END SUB
```

```
SUB SBD(r, sbdloss) STATIC

'  Process:  Calculate surface-based duct loss
'  Inputs from common block:  exloss, fofz, rsbd, rsbdloss, sbdht
'  Inputs from argument list:  r
'  Outputs to common block:  None
'  Outputs to argument list:  sbdloss
'  Subroutines called:  None
'  Subroutine called by:  DIFINT, FFACTR

   IF (sbdht = 0!) THEN
      sbdloss = 1000!
   ELSE
      IF (r < rsbd) THEN
        sbdloss = rsbdloss + (rsbd - r) + exloss
      ELSE
        sbdloss = fsterm + 2! * tenlgr - fofz + exloss
      END IF
   END IF
END SUB
```

```
SUB SKIPTR(zr,x) STATIC

' Process:  SKIPTR traces a ray to a specified height, zr, using initial
'           ray launch angle 0.0.   RAYR returns the range to height zr,
'           x.  This routine is used to determine the skipzone ranges
'           for cases where the transmitter and target heights are both
'           below the top of the surface-based duct.
'  Inputs from arguement list:  zr
'  Inputs from common block:  zt(), p(), q()
'  Outputs to common block:  None
'  Oututs to arguement list:  x
'  Subroutines called:  None
'  Subroutine called by: SKIPZONE


shared zt(),M!(),p(),q(),istart,levels

' set initial conditions at start of ray

a0 = 0.0
i = imin
x = 0.0
xtemp = 0.0
DO
  a0sq = a0 * a0
  IF (zr > zt(i-1)) THEN    ' Trace to zr
    a1 = -SQR(a0sq + 2.0*p(i-1)*(zr - z))
    x = x + (a1 - a0)/p(i-1)
    z = zr
  ELSE                                    ' Full step thru layer
    a1 = -SQR(a0sq - q(i-1))
    x = x + (a1 - a0)/p(i-1)
    z = zt(i-1)
  END IF
  a0 = a1
  i = i-1
LOOP WHILE (i >= 1) OR (z > zr)
END SUB
```

```
SUB SKIPZONE STATIC

'  Process:  Calculates skip-zone range if a surface-based duct
'            is present and calculates the range to the start
'            of the diffraction region.
'  Inputs from common block:  ae, fsterm, r1min, sbdht, h2, hr
'  Inputs from argument list:  fofz
'  Outputs to common block:  rsbd, rsbdloss, rsubd
'  Outputs to argument list:  h2
'  Subroutines called: HGAIN, SKIPTR
'  Subroutine called by:  FFACTR

'     determine the height-gain function for surface-based
'     duct.  Note!  The variable "DUMMY" contains the height-
'     gain function for an evaporation duct which is not used
'     in this subroutine.

  CALL hgain(hr, fofz, DUMMY)
  IF (hr < sbdht) THEN
    CALL skiptr(hr,ray2)
    rsbd = (ray1 + ray2)/1000.0            'SBD start range, km
    rsbdloss = fsterm + 8.686 * LOG(rsbd) - fofz
  END IF
END SUB
```

```
SUB TROPO(r, tloss) STATIC

'  Process:  Calculate the troposcatter loss based upon Yeh with
'            frequency-gain factor, h0, from NBS 101
'  Inputs from common block:  ae, exloss, f3, h1, h14pil, h2
'            h24pil, horizn, rns2, rnsterm, tfac, tsub1, tsub2
'  Inputs from argument list:  r
'  Outputs to argument list:  tloss
'  Subroutines called:  None
'  Subroutine called by:  DLOSS

   tsub0 = r / ae
   ttot = tsub0 - tsub1 - tsub2
   zeta = ttot / 2! + tsub1 + (h1 - h2) / (1000! * r)
   chi  = ttot / 2! + tsub2 + (h2 - h1) / (1000! * r)
   rsub1 = h14pil * ttot
   rsub2 = h24pil * ttot
   IF (rsub1 < .1) THEN rsub1 = .1
   IF (rsub2 < .1) THEN rsub2 = .1
   s = zeta / chi
   IF (s > 10!) THEN s = 10!
   IF (s < .1)  THEN s = .1
   q = rsub2 / (s * rsub1)
   IF (q > 10!) THEN q = 10!
   IF (q < .1)  THEN q = .1
   hsub0 = (s * r * ttot) / ((1! + s) * (1! + s))
   etas = .5696 * hsub0 * (1! + rnsterm * EXP(-.0000033 * hsub0^6))
   IF (etas > 5!)  THEN etas = 5!
   IF (etas < .01) THEN etas = .01
   csub1 = 16.3 + 13.3 * etas
   csub2 = .4 + .16 * etas
   h0r1 = csub1 * (rsub1 + csub2)^-1.333
   h0r2 = csub1 * (rsub2 + csub2)^-1.333
   h0 = (h0r1 + h0r2) / 2!
   delh0 = 1.13 * (.6 - .4343 * LOG(etas)) * LOG(s) * LOG(q)
   IF (delh0 > h0) THEN h0 = 2! * h0 ELSE h0 = h0 + delh0
   IF (h0 < 0!) THEN h0 = 0!
   tloss = 114.9 + tfac*(r - horizn) + 4.343*LOG(r * r * f3) - rns2 + h0
   tloss = tloss + exloss
END SUB
```

## APPENDIX B
## *M*-UNIT PROFILES

| Standard Atmosphere *M*-unit Profile | |
|---|---|
| HEIGHT (m) | *M* UNITS |
| 0.00 | 350.00 |
| 500.00 | 409.00 |
| 1000.00 | 468.00 |

| 28-Meter Evaporation Duct *M*-unit Profile | |
|---|---|
| HEIGHT (m) | *M* UNITS |
| 0.00 | 339.00 |
| 0.04 | 331.19 |
| 0.10 | 328.31 |
| 0.20 | 326.17 |
| 0.316 | 324.76 |
| 0.501 | 323.33 |
| 0.794 | 321.92 |
| 1.259 | 320.54 |
| 1.259 | 320.54 |
| 1.995 | 319.19 |
| 3.162 | 317.90 |
| 5.012 | 316.69 |
| 7.943 | 315.62 |
| 12.589 | 314.76 |
| 19.953 | 314.24 |
| 28.000 | 314.16 |
| 39.811 | 314.52 |
| 50.119 | 315.09 |
| 63.096 | 315.99 |
| 79.433 | 317.31 |
| 100.000 | 319.16 |
| 125.893 | 321.68 |
| 158.489 | 325.03 |
| 199.526 | 329.44 |
| 209.526 | 330.62 |

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

| | | |
|---|---|---|
| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE<br>February 1993 | 3. REPORT TYPE AND DATES COVERED<br>Final |
| 4. TITLE AND SUBTITLE<br>A RAY TRACE MODEL FOR PROPAGATION LOSS | | 5. FUNDING NUMBERS<br>PE: 0602435N<br>WU: DN302214<br>Subproj: R035E81 |
| 6. AUTHOR(S)<br>C. P. Hattan | | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Naval Command, Control and Ocean Surveillance Center (NCCOSC)<br>RDT&E Division<br>San Diego, CA 92152-5001 | | 8. PERFORMING ORGANIZATION REPORT NUMBER<br>TR 1576 |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>Naval Research Laboratory Detachment<br>Stennis Space Center, MS 39529-5000 | | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
| 11. SUPPLEMENTARY NOTES | | |
| 12a. DISTRIBUTION/AVAILABILITY STATEMENT<br>Approved for public release; distribution is unlimited. | | 12b. DISTRIBUTION CODE |

13. ABSTRACT *(Maximum 200 words)*

This study was conducted to determine the feasibility of replacing the optical-region propagation model currently assessing the effect of lower atmosphere refraction on electromagnetic (EM) systems performance within the Engineer's Refractive Effects Prediction System (EREPS) with a complete ray trace model. The ray trace model uses a full ray trace solution to describe the optical interference region where two-path coherent interference between direct and sea-reflected rays dominates electromagnetic propagation. The ray trace model needs a multisegmented modified-refractivity profile to describe the propagation environment. It is implemented in a Microsoft QuickBASIC program intended for IBM-compatible personal computers. The ray trace model can determine propagation loss under certain conditions, but it resolves ray paths with increasing difficulty when the EM system frequency rises, and it produces large divergence factors. Since attempts to eliminate fluctuations in the ray trace divergence factors have failed, the study's results argue that the ray trace model is not suitable for EREPS.

| | | | |
|---|---|---|---|
| 14. SUBJECT TERMS<br>ray trace<br>ray launch angle<br>angular limits | pattern propagation factor<br>modified refractivity (M unit)<br>surface-based ducts | | 15. NUMBER OF PAGES<br>66 |
| | | | 16. PRICE CODE |
| 17. SECURITY CLASSIFICATION OF REPORT<br>UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>UNCLASSIFIED | 20. LIMITATION OF ABSTRACT<br>SAME AS REPORT |

UNCLASSIFIED

| 21a. NAME OF RESPONSIBLE INDIVIDUAL | 21b TELEPHONE *(Include Area Code)* | 21c OFFICE SYMBOL |
|---|---|---|
| C. P. Hattan | (619) 553-1427 | Code 543 |

NSN 7540-01-280-5500

Standard form 298 (BACK)

UNCLASSIFIED

## INITIAL DISTRIBUTION

| | | |
|---|---|---|
| Code 0012 | Patent Counsel | (1) |
| Code 0244 | V. Ware | (1) |
| Code 50 | H. O. Porter | (1) |
| Code 54 | J. H. Richter | (1) |
| Code 543 | R. Paulus | (1) |
| Code 543 | C. P. Hattan | (10) |
| Code 961 | Archive/Stock | (6) |
| Code 964B | Library | (2) |

Defense Technical Information Center
Alexandria, VA 22304–6145 (Master Microfiche)

NCCOSC Washington Liaison Office
Washington, DC 20363–5100

NCCOSC Division Detachment
Warminster, PA 18974–5000

Center for Naval Analyses
Alexandria, VA 22302–0268

Navy Acquisition, Research & Development
Information Center (NARDIC)
Washington, DC 20360–5000

GIDEP Operations Center
Corona, CA 91718–8000