QC
807.5
.U6W6
no. 98
c. 2

NOAA Technical Memorandum ERL WPL-98

ALGORITHMS FOR REFLECTING RAYS FROM GENERAL TOPOGRAPHIC SURFACES IN A RAY TRACING PROGRAM

R. M. Jones

Wave Propagation Laboratory
Boulder, Colorado
July 1982

UNITED STATES
DEPARTMENT OF COMMERCE
Malcolm Baldrige,
Secretary

NATIONAL OCEANIC AND
ATMOSPHERIC ADMINISTRATION
John V. Byrne,
Administrator

Environmental Research
Laboratories
George H. Ludwig
Director

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ALGORITHMS FOR REFLECTING RAYS FROM GENERAL TOPOGRAPHIC SURFACES IN A RAY TRACING PROGRAM

R.M. Jones

ABSTRACT

Algorithms for reflecting rays from general terrain models are presented. They enable a programmer to translate flow charts into computer program subroutines for three-dimensional ray-tracing computer programs that calculate ray paths by numerical integration of Hamilton's equations. The surface of the terrain can be expressed either by giving height as a function of longitude and latitude or as a function of the form $f(r, \theta, \phi) = 0$, where r, θ, and ϕ are earth-centered spherical polar coordinates. In either case, the terrain model must specify partial derivatives of the function including second derivatives, and all specified derivatives must be continuous functions of the arguments. Detailed flow charts and equations are given for the following subtasks:

1) Estimating when the ray may intersect the terrain surface.
2) Recognizing that the ray may have intersected the terrain surface between integration steps.
3) Stepping the ray to the point of intersection with the surface by numerical integration.
4) Reflecting the ray at the surface.

Although designed for atmospheric or underwater acoustic waves, most of the algorithms are general enough to apply to the propagation of other kinds of waves.

1. INTRODUCTION

When we calculate a ray path through an inhomogeneous medium, we use Hamilton's equations to calculate the bending of the ray in regions where the medium varies continuously with position. However, when the propagation properties change discontinuously across a surface (such as an air-land or air-water boundary), we must use Snell's law to calculate the discontinuous change in direction of the ray as it crosses the surface into the new medium, or we must use the appropriate reflection condition to calculate the discontinuous change in the direction of the ray as it reflects from the surface of discontinuity. An efficient, general, ray-tracing program must therefore combine a numerical integration scheme in the continuously varying regions with an algorithm to handle discontinuities at surfaces.

When calculating the ray path in one of the continuously varying regions by numerical integration, it would not be appropriate to include discontinuities at surfaces that separate continuously varying regions, because standard numerical integration algorithms could not correctly treat the discontinuity. (A correct treatment would involve the integration of a Dirac delta function, an impossible feat for numerical integration.) Rather, it is more appropriate to extend the first medium across the boundary in a smooth, continuous way. Then, as the numerical integration proceeds, step by step, it is necessary to detect when the ray has crossed the boundary. At that point, it is necessary to reverse the direction of numerical integration, and by successively smaller steps, find the intersection of the ray with the surface of discontinuity within some specified tolerance. Then the ray can be appropriately refracted to start numerical integration in the new medium, or appropriately reflected to continue numerical integration in the old medium.

Figure 1 is an example of a ray path that illustrates both refraction and reflection at discontinuities. Figure 2 is a four-part account of how the ray path is calculated in the region of such a surface of discontinuity.

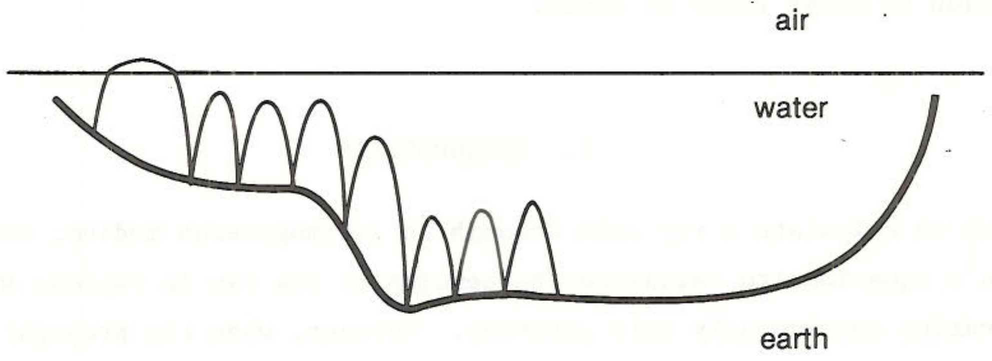


Figure 1.--An example of a ray path illustrating refraction and reflection at surfaces of discontinuity of the medium, in this example, for sound propagation in the ocean.

Detecting that the ray has crossed the surface of discontinuity is quite straightforward for the example shown in Fig. 2a. Figure 3 shows a more difficult example in which the ray crosses the discontinuity twice. It is difficult to distinguish such a ray path from the example in Fig. 4 knowing only the ray path coordinates and ray directions between the integration steps (shown by dots in Figs. 3 and 4). A further difficulty is that iteration of the numerical integration in Fig. 3 might lead to finding the wrong intersection with the surface of discontinuity, a problem shared by the example in Fig. 5.

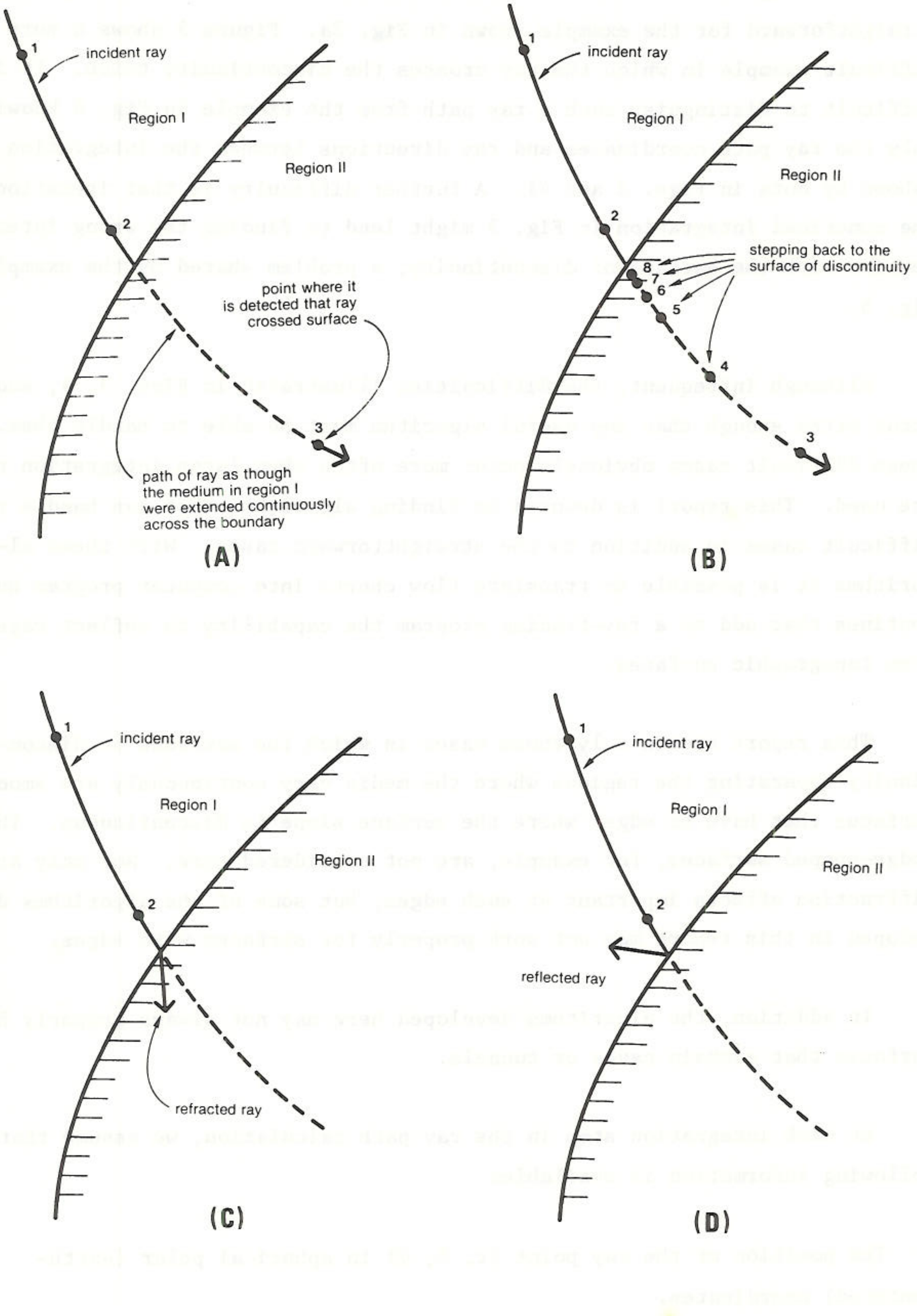
Although infrequent, the difficulties illustrated in Figs. 3, 4, and 5 occur often enough that any useful algorithm must be able to handle them. These difficult cases obviously occur more often when large integration steps are used. This report is devoted to finding algorithms that can handle these difficult cases in addition to the straightforward cases. With these algorithms it is possible to translate flow charts into computer program subroutines that add to a ray-tracing program the capability to reflect rays from topographic surfaces.

This report treats only those cases in which the surfaces of discontinuity separating the regions where the media vary continuously are smooth surfaces that have no edges where the surface slope is discontinuous. Thus wedge-shaped surfaces, for example, are not considered here. Not only are diffraction effects important at such edges, but some of the algorithms developed in this report may not work properly for surfaces with edges.

In addition, the algorithms developed here may not always properly handle surfaces that contain caves or tunnels.

At each integration step in the ray path calculation, we assume that the following information is available:

1) The position of the ray point (r, θ, ϕ) in spherical polar (earth-centered) coordinates.


1
incident ray
Region I
Region II
2
point where it is detected that ray crossed surface
3
path of ray as though the medium in region I were extended continuously across the boundary
(A)
1
incident ray
Region I
Region II
2
stepping back to the surface of discontinuity
8
7
6
5
4
3
(B)
1
incident ray
Region I
Region II
2
refracted ray
(C)
1
incident ray
Region I
Region II
2
reflected ray
(D)

← Figure 2.--A sequence showing how a ray path is calculated in the region of a surface of discontinuity. The raypaths are curved because the media are inhomogeneous. Dots indicate numerical integration steps. a) Recognition that the ray has crossed a surface of discontinuity. The numbers indicate successive positions of the ray between integration steps. b) Iteration by numerical integration to the surface of discontinuity. The numbers indicate successive positions of the ray during the iteration process. This process normally converges faster than shown. c) The refracted ray ready to start numerical integration in the new medium. d) The reflected ray ready to start numerical integration in a new direction in the original medium.

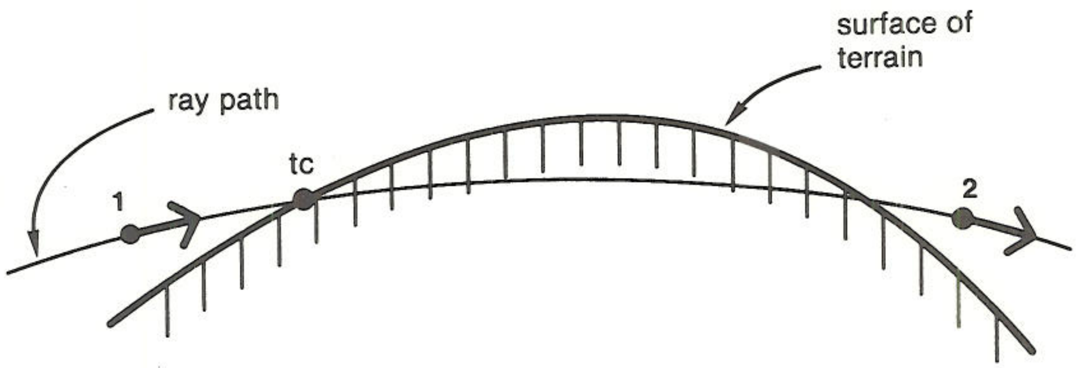


Figure 3.--A ray crossing a surface of discontinuity and crossing back again between integration steps. An algorithm that checked only if the ray point at integration steps were in a different medium would miss such intersections with the surface.

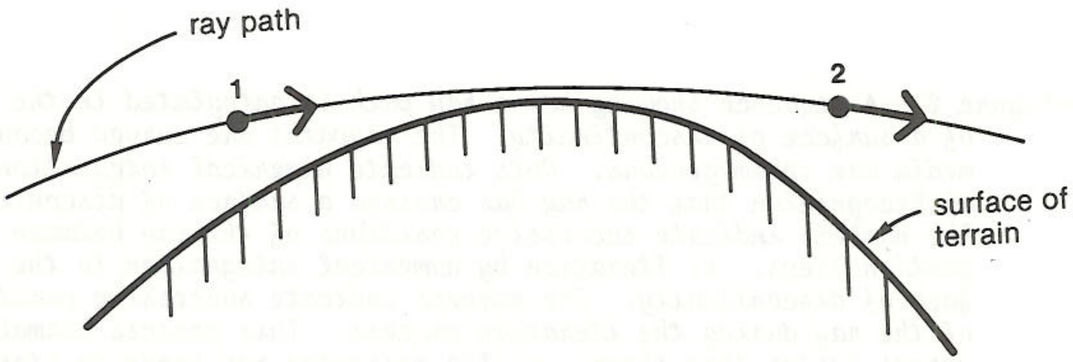


Figure 4.--A ray path that nearly intersects a surface of discontinuity. It is sometimes difficult for an algorithm to distinguish between this situation and that in Fig. 3 by examining the ray only at integration steps.

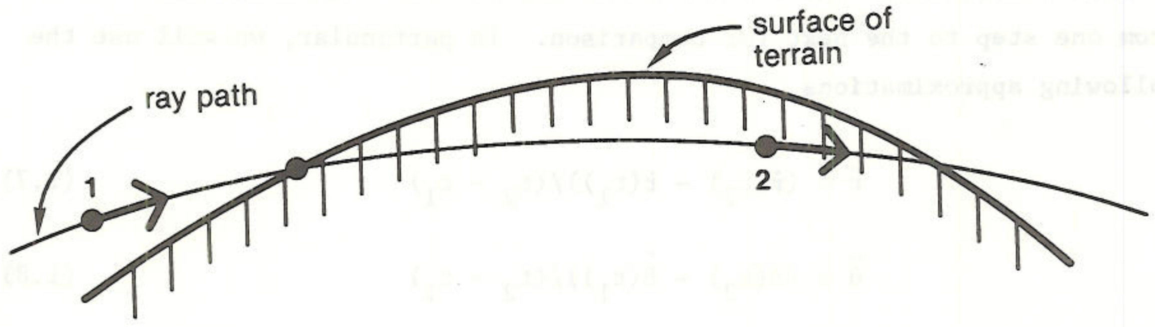


Figure 5.--A ray crossing a surface of discontinuity. An algorithm that is too simple might step the ray by numerical integration steps to the wrong intersection with the surface in order to reflect the ray.

2) The Cartesian components (k_r, k_θ, k_ϕ) of the wave vector (a vector pointing in the wave-normal direction and normalized so that the magnitude equals the wavenumber).

3) The accumulated group time delay, t, of the ray, (which is also the independent variable for the numerical integration).

4) The derivatives

$$\dot{r} \equiv dr/dt \tag{1.1}$$

$$\dot{\theta} \equiv d\theta/dt \tag{1.2}$$

$$\dot{\phi} \equiv d\phi/dt \tag{1.3}$$

$$\dot{k}_r \equiv dk_r/dt \tag{1.4}$$

$$\dot{k}_\theta \equiv dk_\theta/dt \tag{1.5}$$

$$\dot{k}_\phi \equiv dk_\phi/dt \ . \tag{1.6}$$

We will assume also that the values of any of these variables can be saved from one step to the next for comparison. In particular, we will use the following approximations,

$$\ddot{r} \approx (\dot{r}(t_2) - \dot{r}(t_1))/(t_2 - t_1) \tag{1.7}$$

$$\ddot{\theta} \approx (\dot{\theta}(t_2) - \dot{\theta}(t_1))/(t_2 - t_1) \tag{1.8}$$

$$\ddot{\phi} \approx (\dot{\phi}(t_2) - \dot{\phi}(t_1))/(t_2 - t_1) \ , \tag{1.9}$$

which allow us to estimate the curvature of ray. In using the algorithms presented here, it would probably be useful to incorporate a test of the validity of the above approximations.

Although not all the algorithms presented in this report have been tested in a ray-tracing program, most of them are extensions of simpler algorithms designed to deal with the intersection of rays with a spherical earth when the step length is large. These simpler versions have been extensively tested in the ray-tracing program of Jones and Stephenson (1975).

2. SUMMARY--HOW TO INCORPORATE THESE ALGORITHMS INTO A HAMILTONIAN RAY TRACING PROGRAM

Suppose you have a ray tracing program that calculates ray paths by numerically integrating Hamilton's equations. You can modify that program to handle reflections from some sort of topographic terrain model. Figure 6 is a flow chart showing how that can be done with the algorithms developed in this report. Blocks in that flow chart may represent complicated algorithms. Algorithms that are represented in more detail are identified with asterisks. The supporting detail in several instances is another flow chart, which also may contain references to additional supporting detail. The process of expanding detail continues thus until the complete algorithm is well defined.

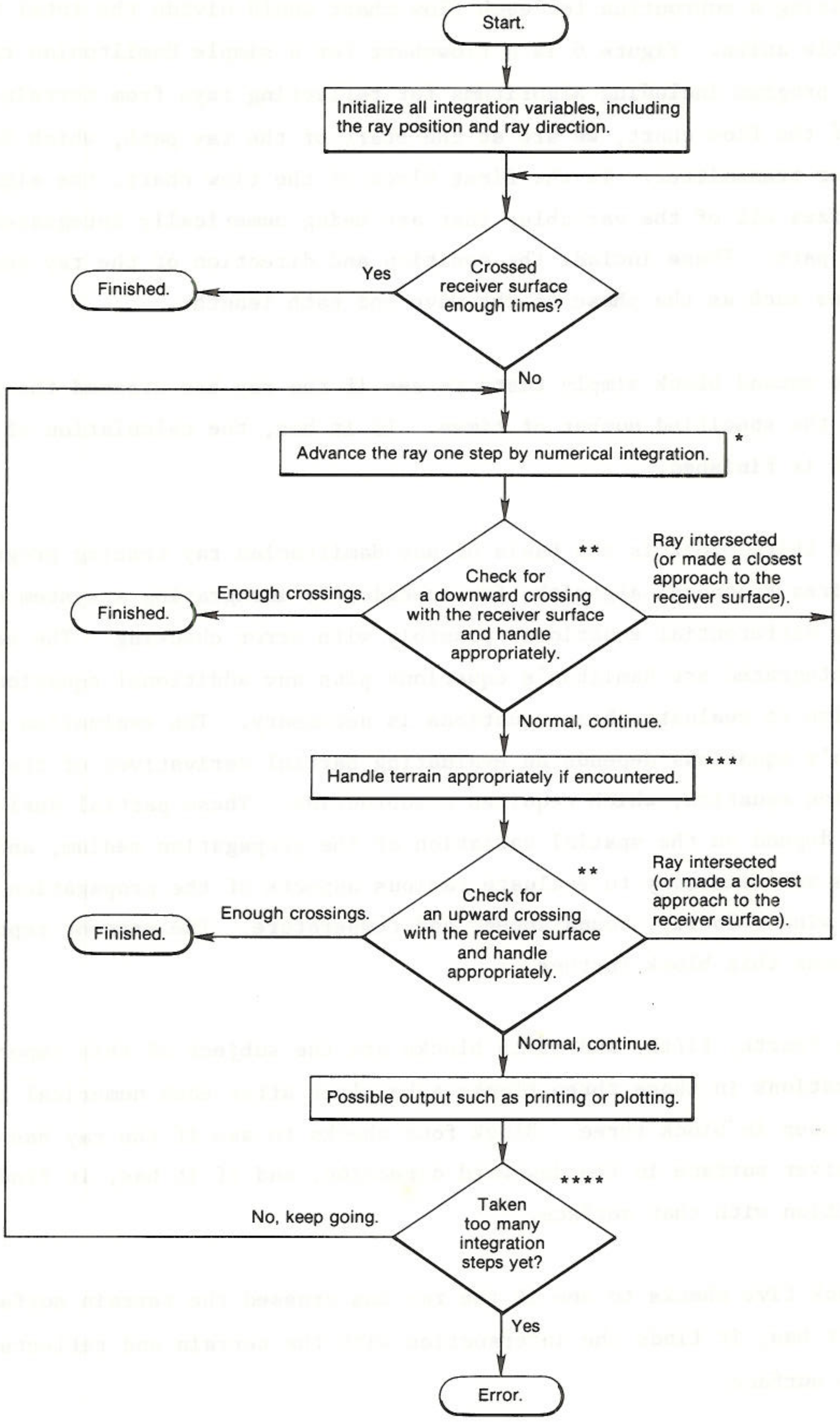
Writing a subroutine for each flow chart would divide the total task into manageable units. Figure 6 is a flowchart for a simple Hamiltonian ray-tracing program including algorithms for reflecting rays from terrain. At the start of the flow chart, we are at the start of the ray path, which is at the source or transmitter. In the first block of the flow chart, the algorithm initializes all of the variables that are being numerically integrated along the ray path. These include the position and direction of the ray and other variables such as the phase of the wave and path length.

The second block simply tests to see if the ray has crossed the receiver surface the specified number of times. If it has, the calculation of that ray path is finished.

The third block is the basis of any Hamiltonian ray tracing program. It requires a general algorithm for numerically integrating a system of ordinary differential equations, possibly with error checking. The equations to be integrated are Hamilton's equations plus any additional equations, and a subroutine to evaluate these equations is necessary. The evaluation of Hamilton's equations depends on evaluating partial derivatives of the wave dispersion equation, which requires a subroutine. These partial derivatives in turn depend on the spatial variation of the propagation medium, and subroutines are necessary to evaluate various aspects of the propagation medium, such as wind velocity, sound speed, and temperature. The present report does not discuss this block further.

The fourth, fifth, and sixth blocks are the subject of this report. The operations in those three blocks take place after each numerical integration step in block three. Block four checks to see if the ray has crossed the receiver surface in the downward direction, and if it has, it finds the intersection with that surface.

Block five checks to see if the ray has crossed the terrain surface, and if it has, it finds the intersection with the terrain and reflects the ray from the surface.


Start.
Initialize all integration variables, including the ray position and ray direction.
Crossed receiver surface enough times?
Yes
Finished.
No
Advance the ray one step by numerical integration.
*
Check for a downward crossing with the receiver surface and handle appropriately.
**
Enough crossings.
Finished.
Ray intersected (or made a closest approach to the receiver surface).
Normal, continue.
Handle terrain appropriately if encountered.
***
Check for an upward crossing with the receiver surface and handle appropriately.
**
Enough crossings.
Finished.
Ray intersected (or made a closest approach to the receiver surface).
Normal, continue.
Possible output such as printing or plotting.
Taken too many integration steps yet?
****
No, keep going.
Yes
Error.

← *Figure 6.--Flow chart for a simple ray-tracing program that uses the algorithms developed in this report.*

* *This block is the basis for any Hamiltonian ray-tracing program. It requires a general algorithm for numerically integrating a system of ordinary differential equations (Hamilton's equations), possibly with error checking. It is not the subject of the present report.*

** *See Fig. 9 for the details of the algorithm that checks for the ray crossing the receiver surface and handles it appropriately.*

*** *See Fig. 8 for the details of the algorithm that checks for the ray crossing the terrain surface and handles it appropriately.*

**** *This is to avoid spending a fortune in computer costs if something goes wrong.*

Block six checks to see if the ray has crossed the receiver surface in the upward direction, and if it has, it finds the intersection with that surface.

Any receiver surface can be chosen that contains the receiver (or observer), but it is usual to choose a simple surface such as a horizontal or vertical plane or a sphere concentric with the earth. If the observer is on the ground, it may be useful to choose the receiver surface to coincide with the terrain surface.

All the algorithms presented that handle receiver surface crossings and terrain could be incorporated similarly into other ray-tracing algorithms.

3. ALGORITHM FOR THE TERRAIN CHECK AFTER EACH NUMERICAL INTEGRATION STEP

The key to solving the problem of reflection or refraction at a boundary is to separate the problem into a hierarchy of subtasks. Figure 7 represents that hierarchy pictorially. The more primitive subtasks are represented by the blocks on the bottom.

In this section we describe the general organization of the terrain problem, and how it depends on three subtasks:

1) Recognizing that a crossing of the surface may have occurred.

2) Stepping the ray by numerical integration to reach the point of crossing.

3) Performing the process of reflection or refraction.

In so doing, we will assume (or require) that the subtasks lower in the hierarchy have certain properties. In addition to the obvious, these are the following:

1) In recognizing that a crossing of the terrain surface has occurred, the algorithm must make sure that the crossing is in the correct direction (that

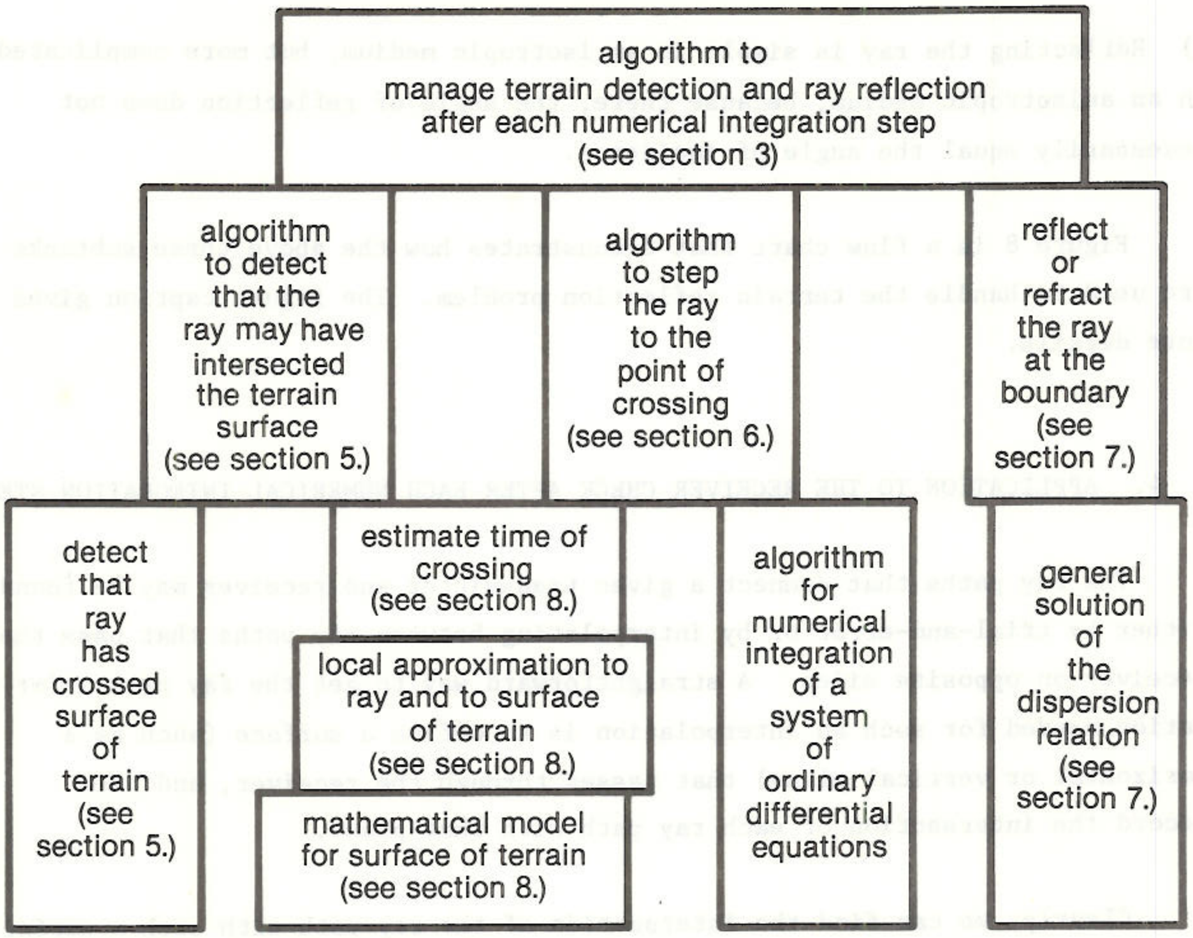

algorithm to manage terrain detection and ray reflection after each numerical integration step (see section 3)
algorithm to detect that the ray may have intersected the terrain surface (see section 5.)
algorithm to step the ray to the point of crossing (see section 6.)
reflect or refract the ray at the boundary (see section 7.)
detect that ray has crossed surface of terrain (see section 5.)
estimate time of crossing (see section 8.)
local approximation to ray and to surface of terrain (see section 8.)
mathematical model for surface of terrain (see section 8.)
algorithm for numerical integration of a system of ordinary differential equations
general solution of the dispersion relation (see section 7.)

Figure 7.--A symbolic representation of the dependence of the solution to the terrain problem on more primitive algorithms. In each case, algorithms in upper blocks depend on those in lower blocks. One could think of this diagram as an organizational chart. While this diagram presents the hierarchy in the structure, it does not present the order or sequence of operations. The sequence of operations corresponding to the top block is shown in the flow chart in Fig. 8, and that for other blocks is shown in other flow charts.

is, from above the terrain to below the terrain). In addition, the algorithm must estimate the time (group travel time, the independent variable for the integration) of crossing.

2) In stepping the ray by numerical integration to the point of crossing, it may sometimes happen that the ray actually misses the terrain. In that case, it seems most appropriate to have that algorithm find the point of closest approach.

3) Reflecting the ray is simple in an isotropic medium, but more complicated in an anisotropic medium, because there, the angle of reflection does not necessarily equal the angle of incidence.

Figure 8 is a flow chart that demonstrates how the above three subtasks are used to handle the terrain reflection problem. The figure caption gives more details.

4. APPLICATION TO THE RECEIVER CHECK AFTER EACH NUMERICAL INTEGRATION STEP

The ray paths that connect a given transmitter and receiver may be found either by trial-and-error or by interpolating between ray paths that pass the receiver on opposite sides. A straightforward way to get the ray path information needed for such an interpolation is to define a surface (such as a horizontal or vertical plane) that passes through the receiver, and then record the intersection of each ray path with that plane.

Clearly, we can find the intersection of the ray path with such a surface if we can find the intersection of the ray path with a topographic surface discussed in the previous section. However, because we do not want to reflect or refract the ray at the surface, only two of the three subtasks are needed here. However, there are three differences:

1) If the ray path makes a closest approach to but does not intersect the surface, we want to record that event.

2) If the receiver is on the terrain, then we may want to choose the receiver surface to coincide with the terrain.

3) We will be interested in both upgoing and downgoing crossings of the surface (but the crossing of interest will alternate).

Figure 9 is a flow chart that includes these three differences. Although the receiver problem is not the main subject of this report, the obvious application made it worth mentioning.

The receiver check should be performed before the terrain check to handle correctly the cases where the ray crosses the receiver surface and the terrain surface in a single integration step. (This assumes the receiver is not below the terrain.)

5. RECOGNIZING WHEN THE RAY MAY HAVE INTERSECTED A SPECIFIED SURFACE

If we look again at Fig. 7, we see that recognizing that the ray may have crossed a specified surface depends on two more primitive subtasks:

1) Recognizing that the ray actually did cross the surface, and

2) Being able to estimate the time when an intersection may have recently occurred or will occur in the near future.

In the former, the ray point at an integration step has actually been found to have crossed the surface (as illustrated in Fig. 5); in the latter, only a prediction of an intersection with the surface has been made (such as the point t_c in Fig. 3). In this section, we consider three algorithms for recognizing when the ray may have intersected the surface. We shall start with the simplest, and add more complexity. The simplest algorithm will sometimes miss the occurrence of an intersection that the more complicated algorithms will recognize.

5.1 Simple Method

Figure 10 shows the flow chart for the simplest algorithm for recognizing that the ray may have intersected the surface. The algorithm can recognize the situation shown in Fig. 5, because one of the ray points is actually below the terrain. It would be able to recognize the possible intersection in Fig. 3 because the ray was going toward the surface in the earlier step, and away from the surface in the later step. (However, it would have difficulty in distinguishing between the cases in Figs. 3 and 4 when the ray path either

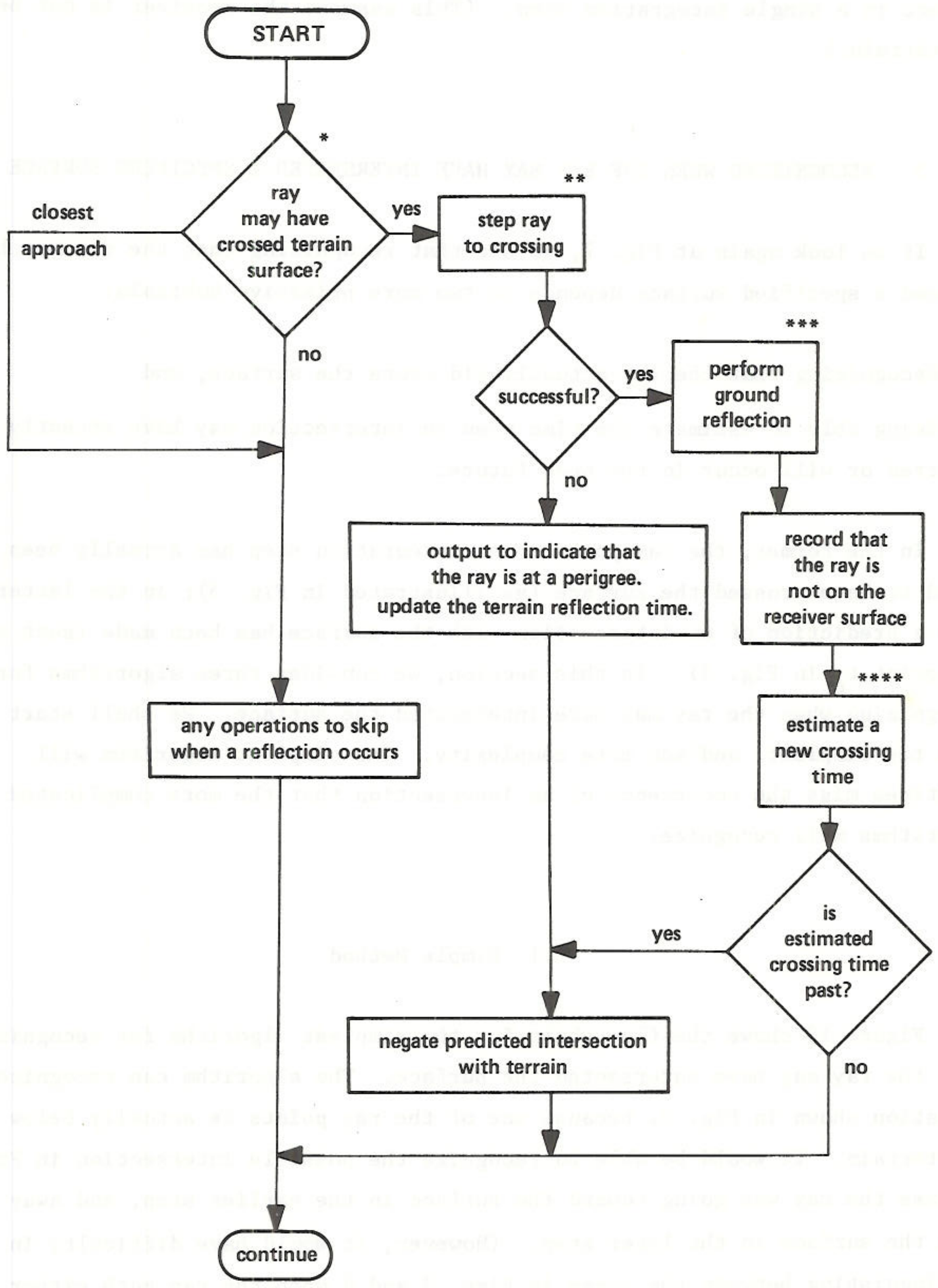

START
*
ray may have crossed terrain surface?
closest approach
yes
no
**
step ray to crossing
successful?
yes
no
***
perform ground reflection
output to indicate that the ray is at a perigree. update the terrain reflection time.
record that the ray is not on the receiver surface
****
estimate a new crossing time
any operations to skip when a reflection occurs
is estimated crossing time past?
yes
no
negate predicted intersection with terrain
continue

Table 1.--Details of performing a ground reflection

1.	Change the sign of the component of the wave vector normal to the terrain surface. Adjust the magnitude of the normal component of the wave vector to satisfy the dispersion relation. Equation (7.15) gives the components of the wave vector of the reflected acoustic wave in terms of the components of the wave vector of the incident acoustic wave.
2.	Remember that the numerical integration routine must be told to begin a new integration rather than continue a previous integration.
3.	Create output as appropriate--normally a line of printout and a record of machine-readable output to indicate that the ray is at a ground reflection.

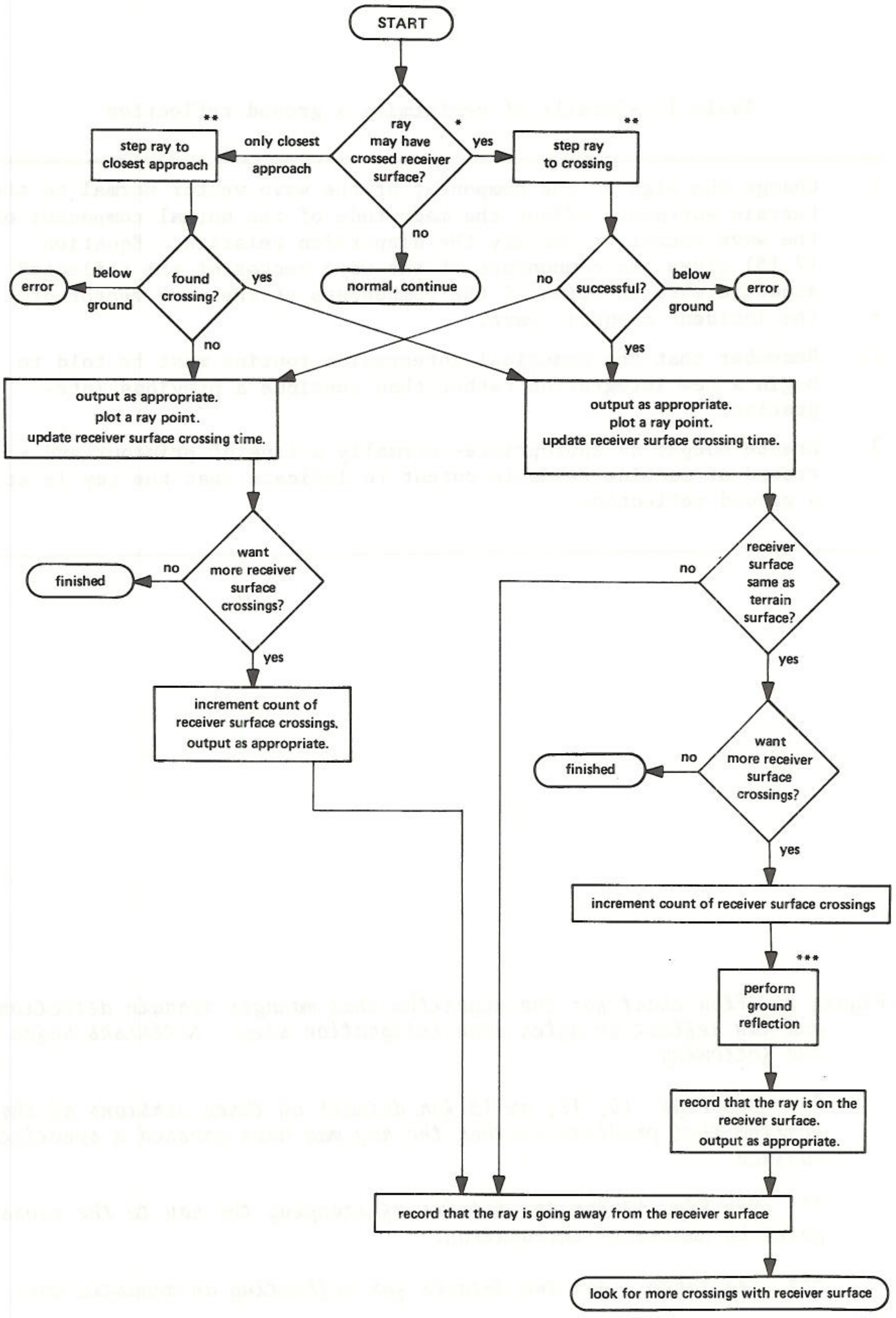
← *Figure 8.--Flow chart for the algorithm that manages terrain detection and ray reflection after each integration step. Asterisks refer to the following:*

* *See Figs. 10, 12, or 13 for details of three versions of the algorithm that predicts whether the ray may have crossed a specified surface.*

** *See Fig. 17 for the details of stepping the ray to the crossing point by numerical integration.*

*** *See Table 1 for the details for reflecting an acoustic wave.*

**** *Use equation (8.19) or (8.38) to estimate the time of crossing a specified surface.*


START
ray may have crossed receiver surface?
*
only closest approach
yes
no
normal, continue
**
step ray to closest approach
**
step ray to crossing
found crossing?
below ground
error
yes
no
successful?
no
below ground
error
yes
output as appropriate. plot a ray point. update receiver surface crossing time.
output as appropriate. plot a ray point. update receiver surface crossing time.
want more receiver surface crossings?
no
finished
yes
increment count of receiver surface crossings. output as appropriate.
receiver surface same as terrain surface?
no
yes
want more receiver surface crossings?
no
finished
yes
increment count of receiver surface crossings
***
perform ground reflection
record that the ray is on the receiver surface. output as appropriate.
record that the ray is going away from the receiver surface
look for more crossings with receiver surface

←Figure 9.--Flow chart for applying the algorithms to check for the intersection of a ray with receiver surface after each integration step. Asterisks refer to the following:

* See Figs. 10, 12, or 13 for details of three versions of the algorithm that predicts whether the ray may have crossed a specified surface.

** See Fig. 17 for the details of the algorithm that steps the ray to a crossing or closest approach by numerical integration.

*** See Table 1 for the details for reflecting an acoustic wave.

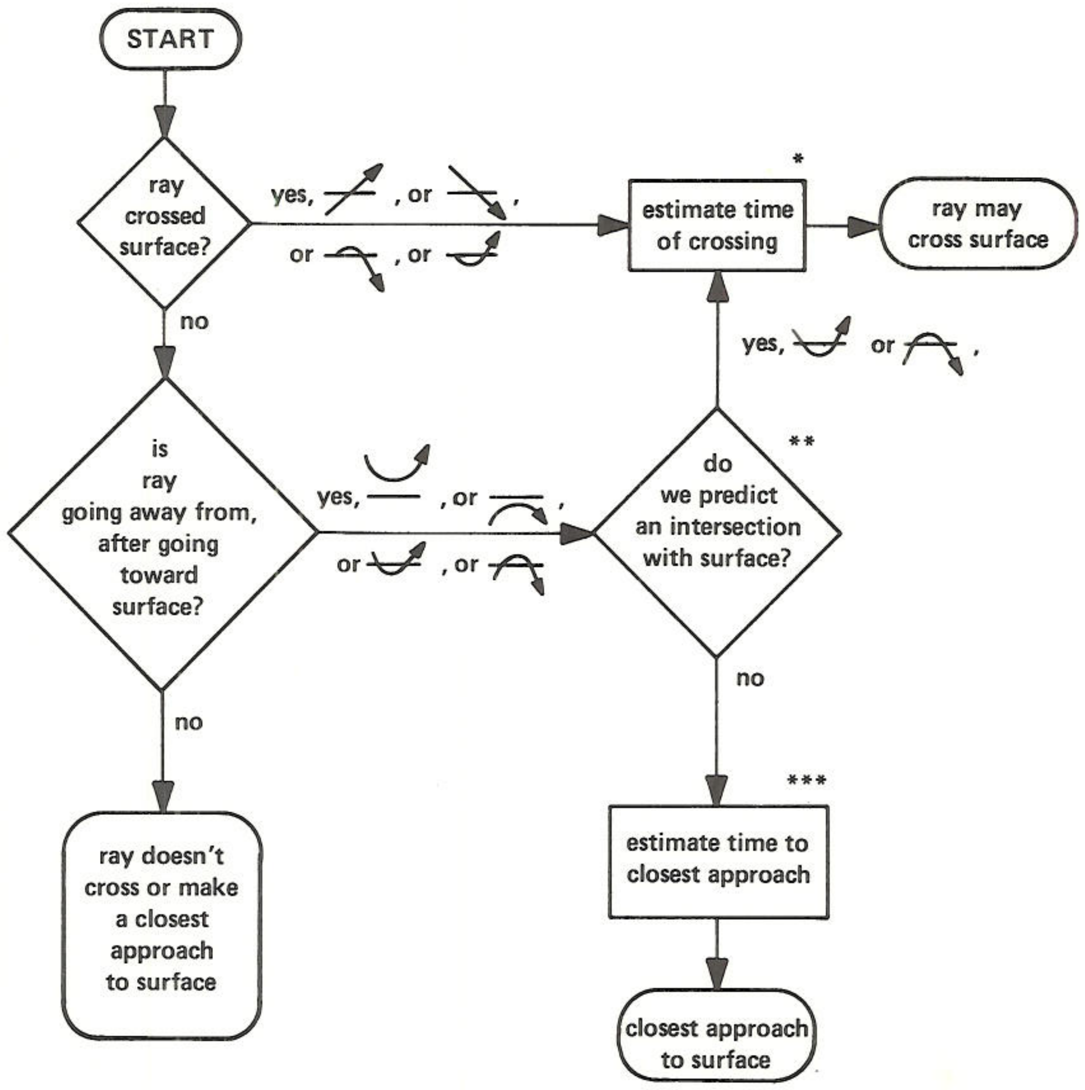

START
ray crossed surface?
yes, , or ,
or , or
estimate time of crossing
ray may cross surface
no
yes, or ,
is ray going away from, after going toward surface?
yes, , or ,
or , or
do we predict an intersection with surface?
no
no
ray doesn't cross or make a closest approach to surface
estimate time to closest approach
closest approach to surface

Figure 10.--Flow chart for the simple version of the algorithm that predicts whether the ray may have crossed a specified surface. Asterisks refer to the following:

* Use equation (8.19) or (8.38) to estimate the time of crossing in a specified direction.

** Use the inequality (8.16) or (8.35) to predict the occurrence of an intersection with the surface.

*** Use equation (8.18) or (8.37) to estimate the time of closest approach to the surface.

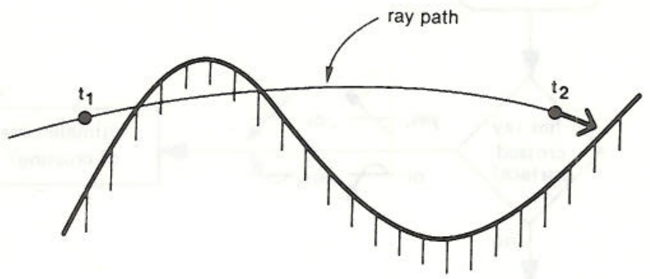


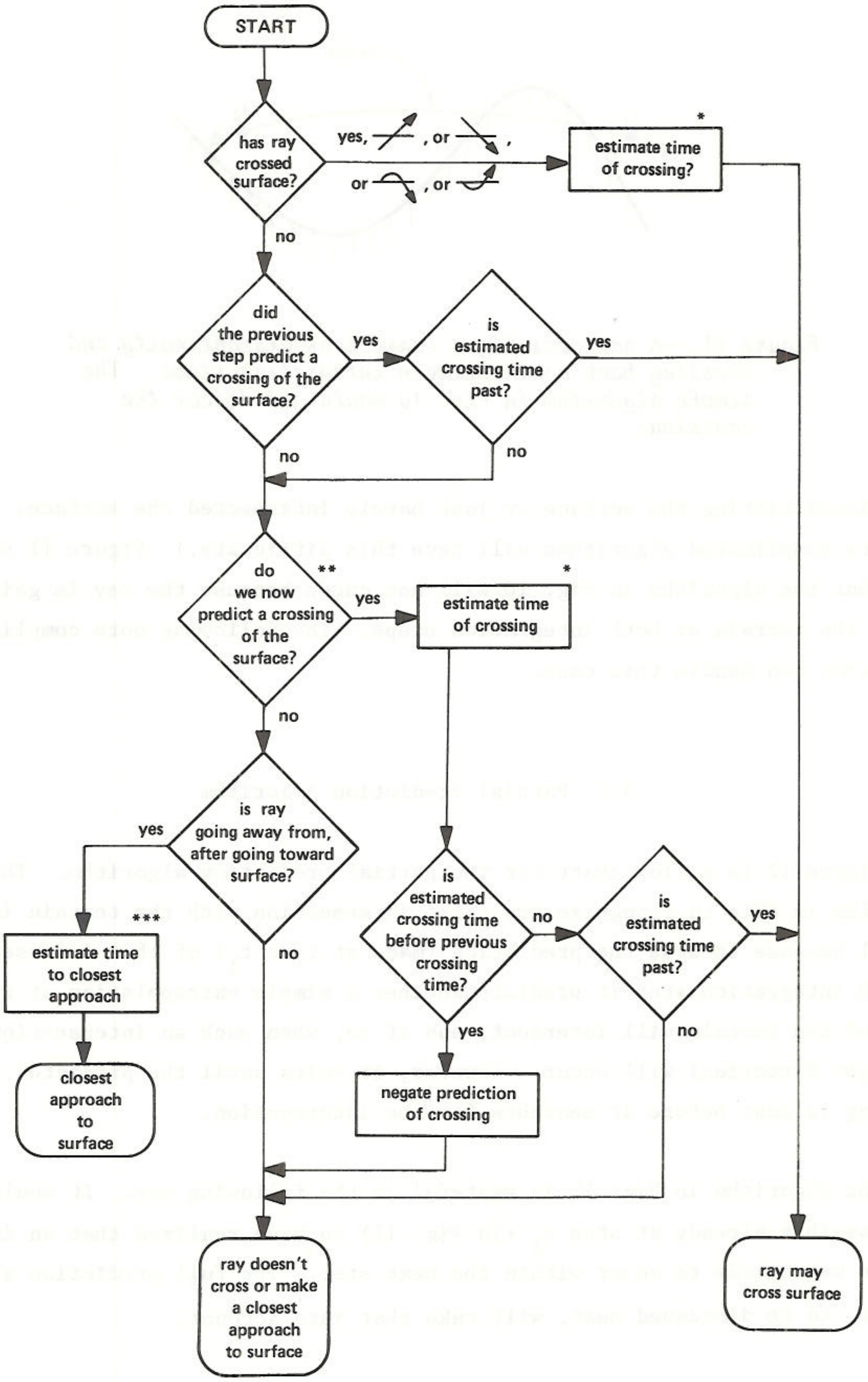
Figure 11.--A ray crossing a surface of discontinuity and crossing back again between integration steps. The simple algorithm in Fig. 10 would not detect the crossings.

just missed hitting the surface or just barely intersected the surface. Even the more complicated algorithms will have this difficulty.) Figure 11 shows a case that the algorithm in Fig. 10 will not catch because the ray is going toward the terrain at both integration steps. The following more complicated algorithms can handle this case.

5.2 Partial Prediction Algorithm

Figure 12 is a flow chart for the partial prediction algorithm. The algorithm is able to recognize the first intersection with the terrain in Fig. 11 because it uses the prediction (made at time t_1) of that intersection. At each integration step it predicts whether a simple extrapolation of the ray path and the terrain will intersect, and if so, when such an intersection (in the right direction) will occur. However, it waits until the predicted time of crossing is past before it searches for the intersection.

The algorithm in Fig. 12 is wasteful in the following way. It would have been possible already at step t_1 (in Fig. 11) to have realized that an intersection was likely to occur within the next step. The full prediction algorithm, to be discussed next, will take that into account.


START
has ray crossed surface?
yes, , or , or , or
estimate time of crossing?
*
no
did the previous step predict a crossing of the surface?
yes
is estimated crossing time past?
yes
no
no
do we now predict a crossing of the surface?
**
yes
estimate time of crossing
*
no
is ray going away from, after going toward surface?
yes
***
estimate time to closest approach
closest approach to surface
no
is estimated crossing time before previous crossing time?
no
is estimated crossing time past?
yes
yes
negate prediction of crossing
no
ray doesn't cross or make a closest approach to surface
ray may cross surface

← Figure 12.--Flow chart for the partial prediction version of the algorithm that predicts whether the ray may have crossed a specified surface. Asterisks refer to the following:

* Use equation (8.19) or (8.38) to estimate the time of crossing in a specified direction.

** Use the inequality (8.16) or (8.35) to predict the occurrence of an intersection with the surface.

*** Use equation (8.18) or (8.37) to estimate the time of closest approach to the surface.

5.3 Full Prediction Algorithm

Figure 13 is the flow chart for the full prediction algorithm. Although the full prediction algorithm does not recognize any additional intersections of the ray with the surface, as compared with the partial prediction algorithm, it does so sooner, and therefore saves computer time.

5.4 Prediction Methods for Recognizing a Closest Approach

The algorithms discussed so far have recognized not only intersections of the ray path with the surface, but also points of closest approach to the surface. In all three algorithms, however, the method was simple, and was equivalent to the algorithm represented in Fig. 14. Although the algorithm in Fig. 14 does not distinguish between an intersection with the surface and a closest approach, the assumption is that an intersection with the surface has already been looked for and not found.

However, as with trying to recognize intersections, the simple algorithm in Fig. 14 will not recognize all closest approaches. For example, Fig. 15 shows two cases that will be missed by the algorithm in Fig. 14. In the upper diagram in Fig. 15, the ray is going toward the surface in both steps, and in the lower, away from the surface in both steps.

Figure 16 shows a full prediction algorithm that would recognize closest approaches such as those in Fig. 15.

6. STEPPING THE RAY TO THE INTERSECTION BY NUMERICAL INTEGRATION

If we look again at Fig. 7, we see that stepping the ray to an intersection by numerical integration depends on two primitive subtasks.

1) Being able to estimate the time of occurrence of an intersection.

2) Having a general numerical integration method for a system of ordinary differential equations.

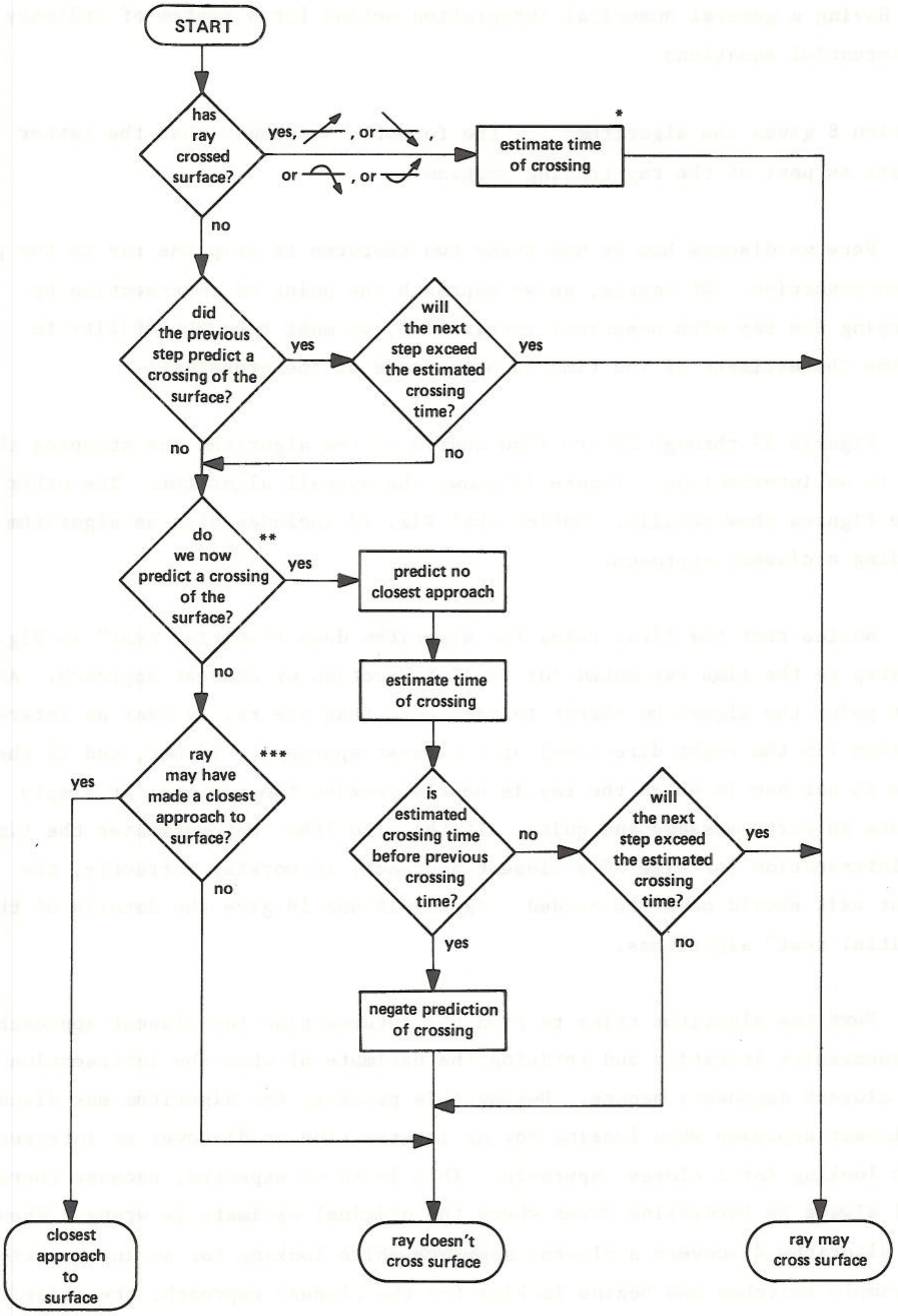
Section 8 gives the algorithms for the former. We assume that the latter exists as part of the ray tracing program.

Here we discuss how we use these two features to step the ray to the point of intersection. Of course, as we approach the point of intersection by stepping the ray with numerical integration, we must have the ability to refine the estimate of the time of occurrence of the crossing.

Figures 17 through 22 are flow charts of the algorithm for stepping the ray to an intersection. Figure 17 shows the overall algorithm. The other five figures show details. Notice that Fig. 17 includes also an algorithm for finding a closest approach.

Notice that the first thing the algorithm does ("initial test" in Fig. 17) is step to the time estimated for the intersection or closest approach. At that point the algorithm checks to make sure that the ray is near an intersection (in the right direction) or a closest approach. If not, and if the case is not one in which the ray is nearly grazing the surface, it simply prints an error message and quits. If the algorithm that estimates the time of intersection (or time of a closest approach) is working correctly, the error exit should never be needed. Figures 18 and 19 give the details of the "initial test" algorithms.

Next the algorithm tries to find the intersection (or closest approach) by successive iteration and refining the estimate of when the intersection (or closest approach) occurs. During this process, the algorithm may discover a closest approach when looking for an intersection or discover an intersection when looking for a closest approach. This is to be expected, because there will always be borderline cases where the original estimate is wrong. When the algorithm discovers a closest approach while looking for an intersection, it simply switches and begins looking for the closest approach; the algorithm switches similarly for the opposite case.



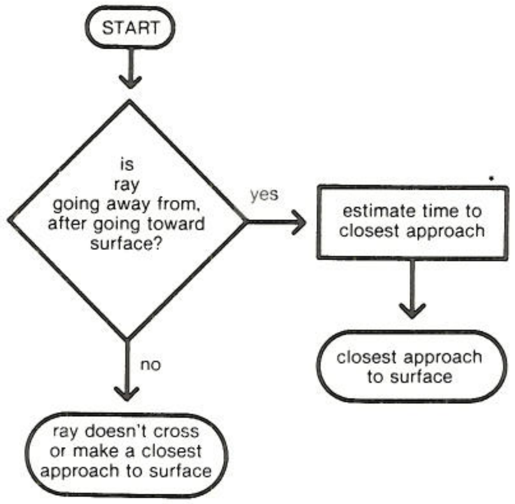
START
has ray crossed surface?
yes, , or , or , or
estimate time of crossing
*
no
did the previous step predict a crossing of the surface?
yes
will the next step exceed the estimated crossing time?
yes
no
no
do we now predict a crossing of the surface?
**
yes
predict no closest approach
estimate time of crossing
no
ray may have made a closest approach to surface?
***
yes
no
is estimated crossing time before previous crossing time?
no
will the next step exceed the estimated crossing time?
yes
yes
no
negate prediction of crossing
closest approach to surface
ray doesn't cross surface
ray may cross surface

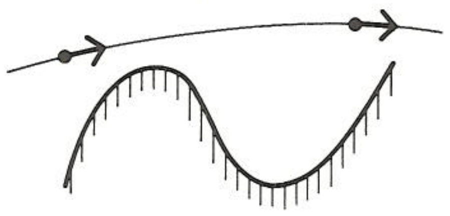
← *Figure 13.--Flow chart for the full prediction version of the algorithm that predicts whether the ray may have crossed a specified surface. Asterisks refer to the following:*

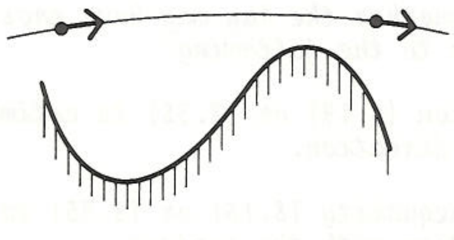
* *Use equation (8.19) or (8.38) to estimate the time of crossing in a specified direction.*

** *Use the inequality (8.16) or (8.35) to predict the occurrence of an intersection with the surface.*

*** *See Figs. 14 or 16 for two versions of the algorithm that predicts whether the ray may have made a closest approach to a specified surface.*


START
is ray going away from, after going toward surface?
yes
estimate time to closest approach
closest approach to surface
no
ray doesn't cross or make a closest approach to surface

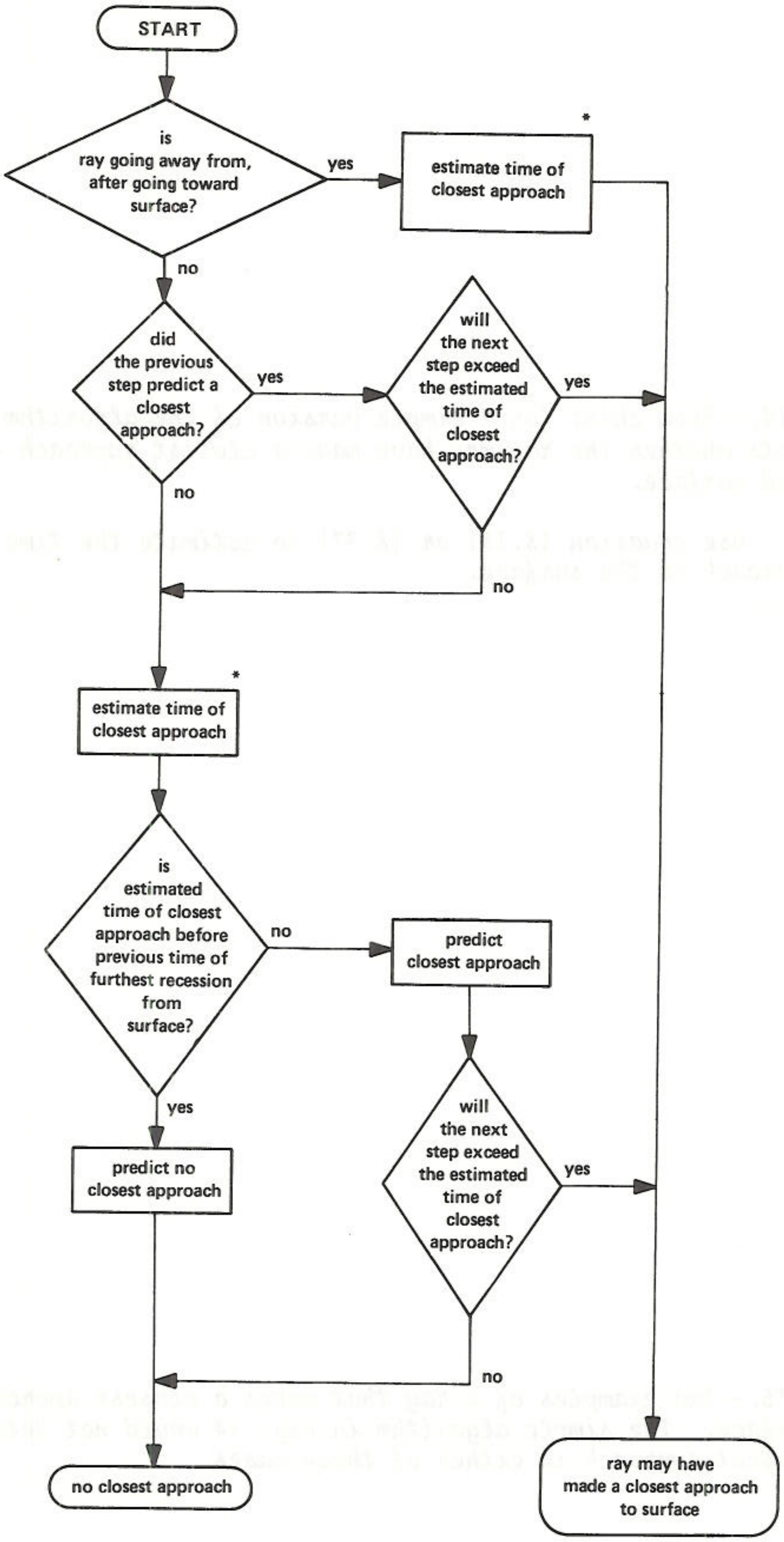




← *Figure 14.--Flow chart for a simple version of the algorithm that predicts whether the ray may have made a closest approach to a specified surface.*

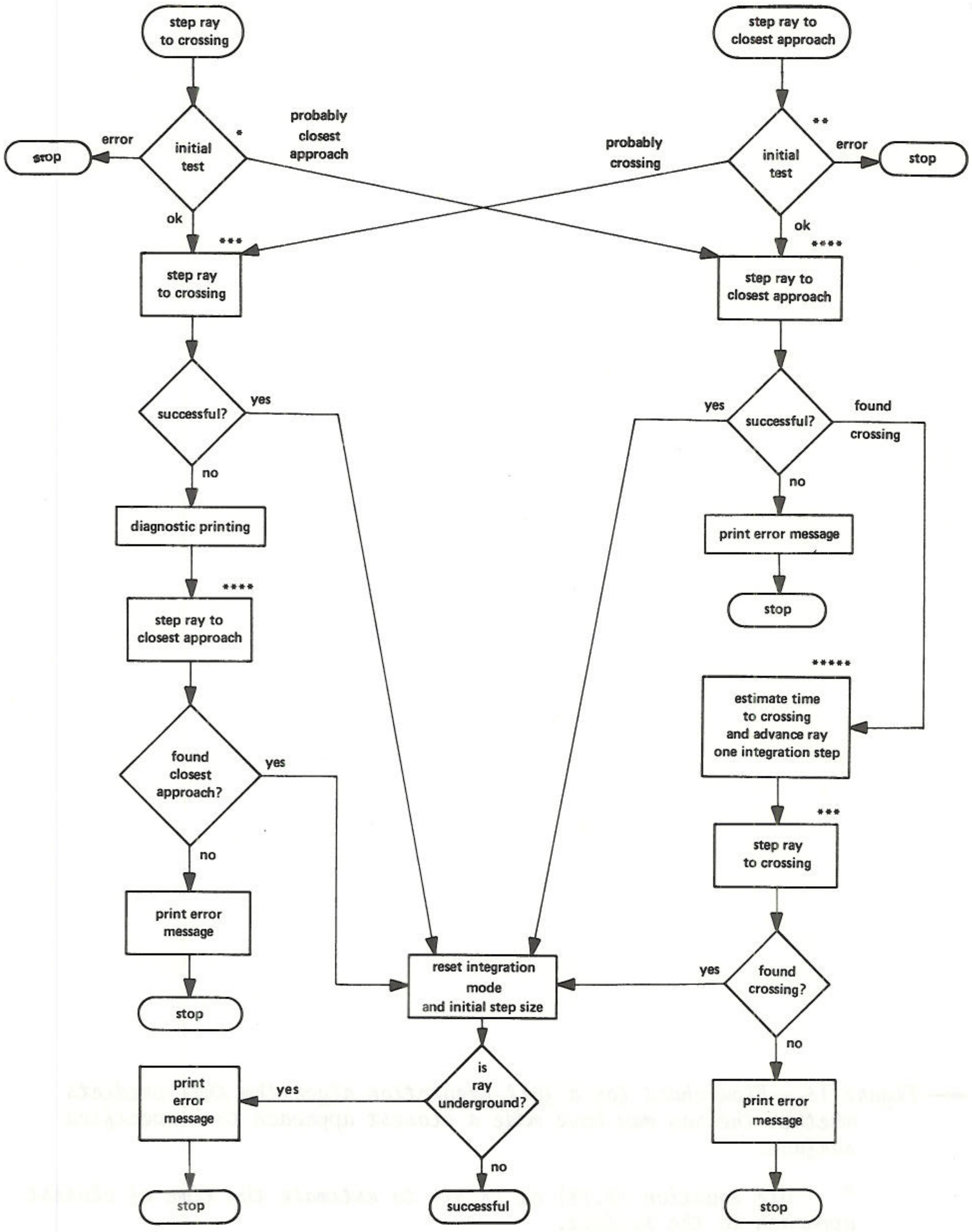
* *Use equation (8.18) or (8.37) to estimate the time of closest approach to the surface.*

← *Figure 15.--Two examples of a ray that makes a closest approach to a surface. The simple algorithm in Fig. 14 would not detect the closest approach in either of these cases.*


START
is ray going away from, after going toward surface?
yes
no
estimate time of closest approach
*
did the previous step predict a closest approach?
yes
no
will the next step exceed the estimated time of closest approach?
yes
no
estimate time of closest approach
*
is estimated time of closest approach before previous time of furthest recession from surface?
no
yes
predict closest approach
will the next step exceed the estimated time of closest approach?
yes
no
predict no closest approach
no closest approach
ray may have made a closest approach to surface

← Figure 16.--Flow chart for a full prediction algorithm that predicts whether the ray may have made a closest approach to a specified surface.

* Use equation (8.18) or (8.37) to estimate the time of closest approach to the surface.


step ray to crossing
initial test
error
stop
*
probably closest approach
ok
***
step ray to crossing
successful?
yes
no
diagnostic printing
****
step ray to closest approach
found closest approach?
yes
no
print error message
stop
step ray to closest approach
initial test
**
error
stop
probably crossing
ok
****
step ray to closest approach
successful?
yes
found crossing
no
print error message
stop
*****
estimate time to crossing and advance ray one integration step
***
step ray to crossing
found crossing?
yes
no
print error message
stop
reset integration mode and initial step size
is ray underground?
yes
no
print error message
stop
successful

←Figure 17.--Flow chart for the algorithms to step the ray to a crossing of or closest approach to a specified surface. Asterisks refer to the following:

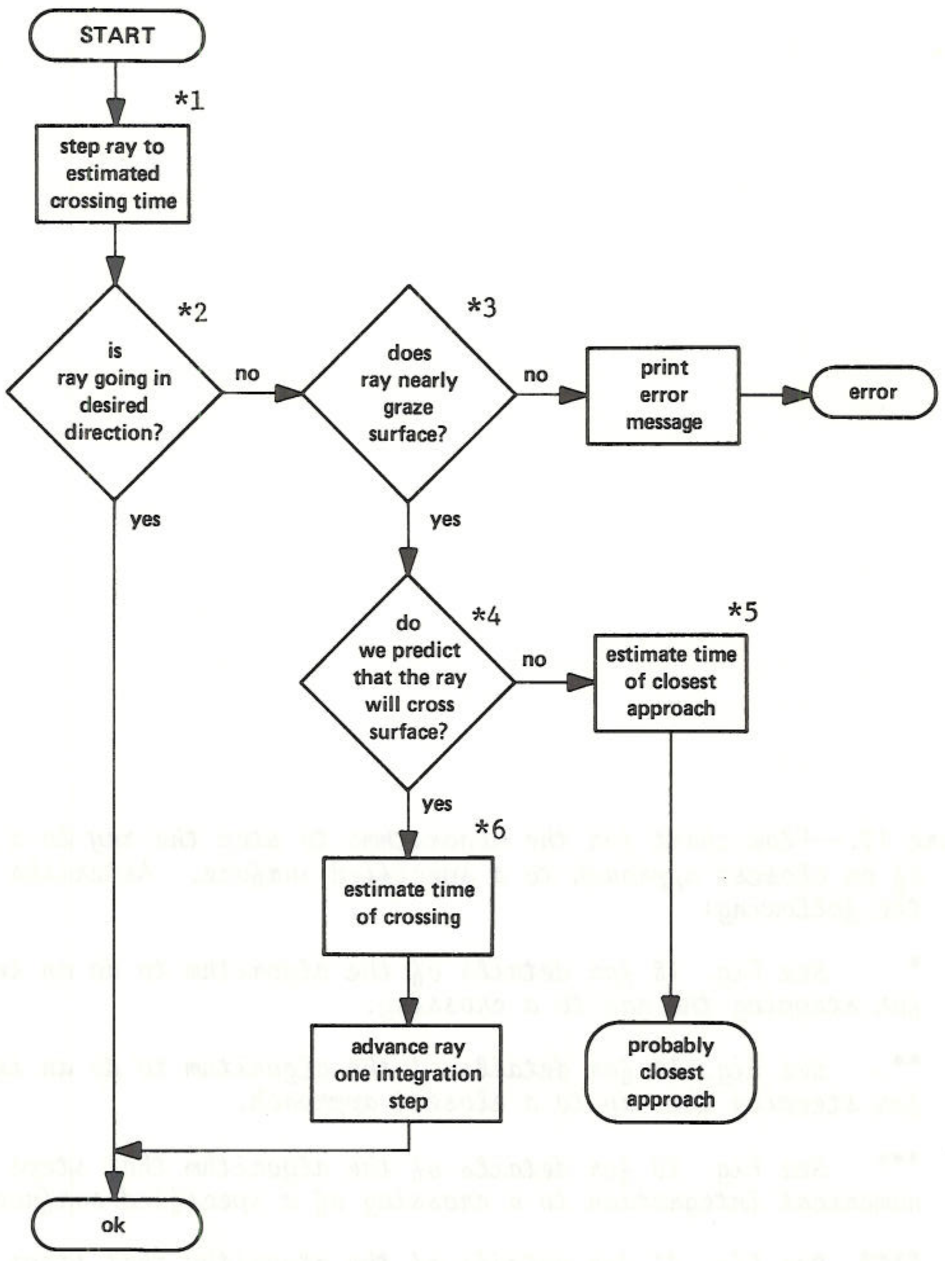
* See Fig. 18 for details of the algorithm to do an initial test for stepping the ray to a crossing.

** See Fig. 19 for details of the algorithm to do an initial test for stepping the ray to a closest approach.

*** See Fig. 20 for details of the algorithm that steps the ray by numerical integration to a crossing of a specified surface.

**** See Fig. 21 for details of the algorithm that steps the ray by numerical integration to a closest approach to a specified surface.

***** Use equation (8.15) or (8.34) to estimate the time of crossing in a specified direction.


START
*1
step ray to estimated crossing time
*2
is ray going in desired direction?
no
yes
*3
does ray nearly graze surface?
no
print error message
error
yes
*4
do we predict that the ray will cross surface?
no
*5
estimate time of closest approach
yes
*6
estimate time of crossing
advance ray one integration step
probably closest approach
ok

← *Figure 18.--Flow chart for the algorithm to do an initial test for stepping the ray to a crossing. Asterisks refer to the following:*

**1 See Fig. 22 for details of the algorithm that steps the ray to a specified time.*

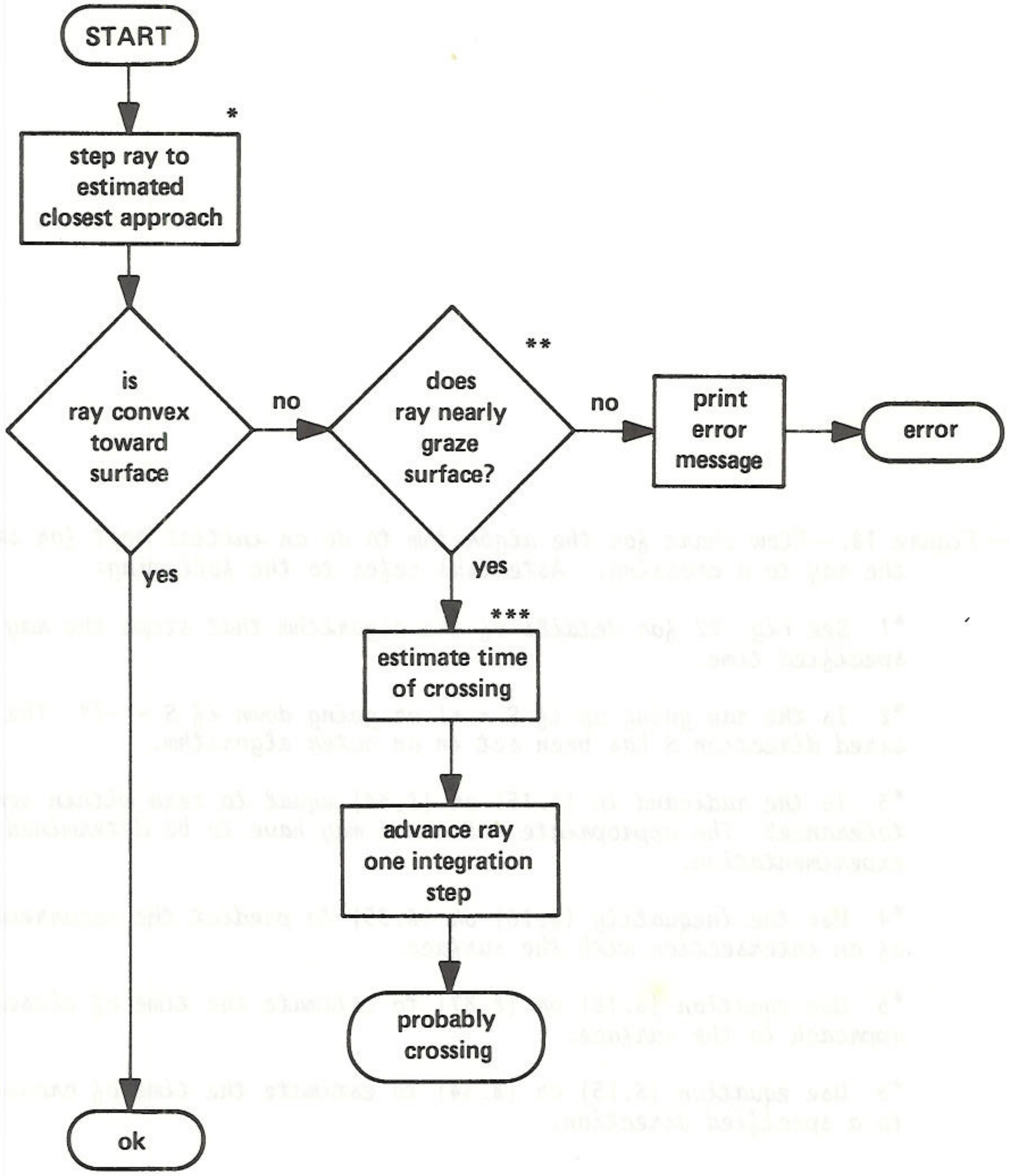
**2 Is the ray going up if S = +1 or going down if S = -1? The desired direction S has been set in an outer algorithm.*

**3 Is the radicand in (8.15) or (8.34) equal to zero within some tolerance? The appropriate tolerance may have to be determined by experimentation.*

**4 Use the inequality (8.16) or (8.35) to predict the occurrence of an intersection with the surface.*

**5 Use equation (8.18) or (8.37) to estimate the time of closest approach to the surface.*

**6 Use equation (8.15) or (8.34) to estimate the time of crossing in a specified direction.*

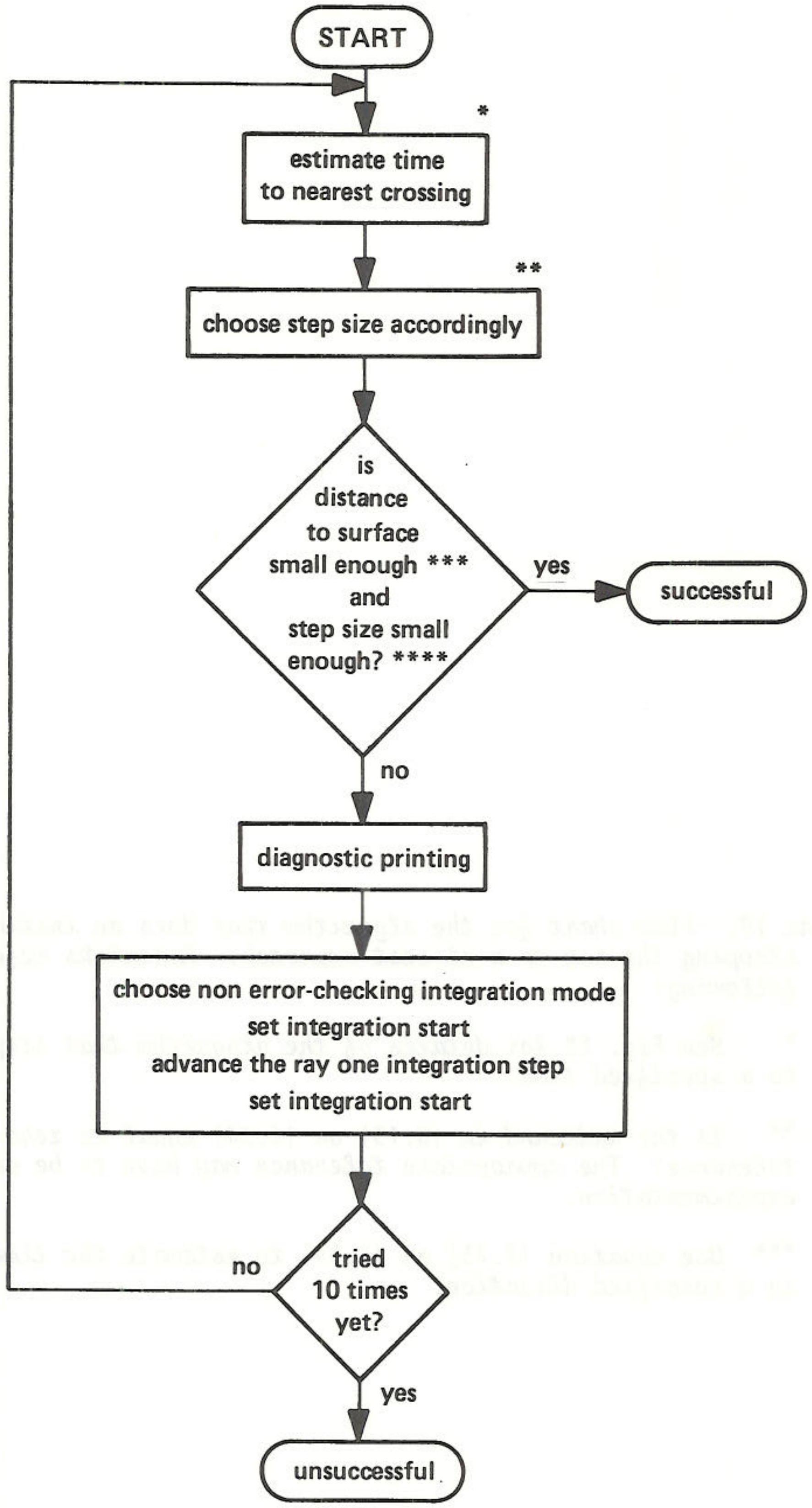

START
*
step ray to estimated closest approach
is ray convex toward surface
no
**
does ray nearly graze surface?
no
print error message
error
yes
yes
***
estimate time of crossing
advance ray one integration step
probably crossing
ok

← Figure 19.--Flow chart for the algorithm that does an initial test for stepping the ray to a closest approach. Asterisks refer to the following:

* See Fig. 22 for details of the algorithm that steps the ray to a specified time.

** Is the radicand in (8.15) or (8.34) equal to zero within some tolerance? The appropriate tolerance may have to be determined by experimentation.

*** Use equation (8.15) or (8.34) to estimate the time of crossing in a specified direction.


START
*
estimate time to nearest crossing
**
choose step size accordingly
is distance to surface small enough *** and step size small enough? ****
yes
successful
no
diagnostic printing
choose non error-checking integration mode
set integration start
advance the ray one integration step
set integration start
tried 10 times yet?
no
yes
unsuccessful

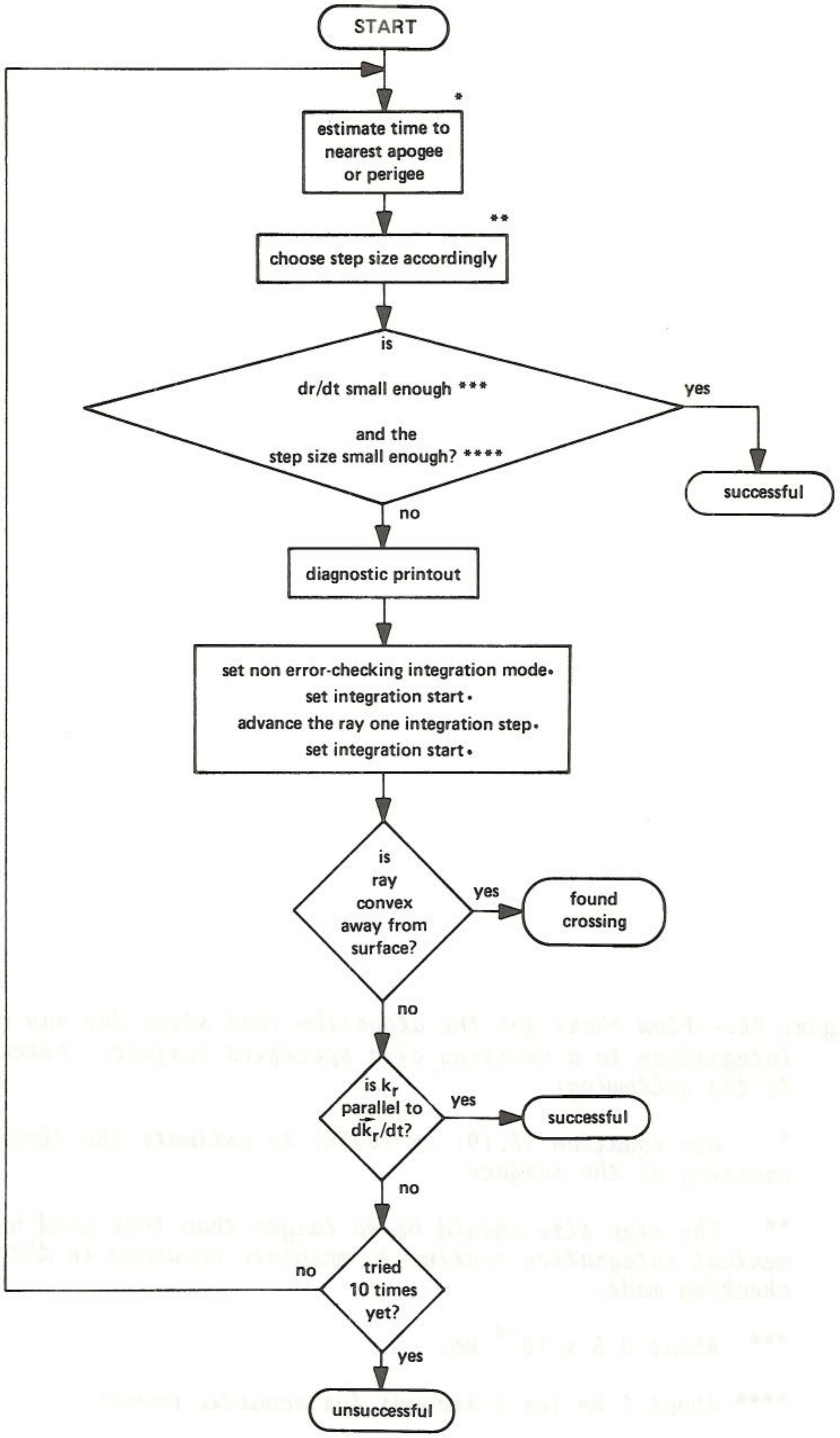
←Figure 20.--Flow chart for the algorithm that steps the ray by numerical integration to a crossing of a specified surface. Asterisks refer to the following:

* Use equation (8.19) or (8.38) to estimate the time of the crossing of the surface.

** The step size should be no larger than that used by the numerical integration routine to maintain accuracy in the error-checking mode.

*** About 0.5×10^{-4} km.

**** About 1 km (or 3 seconds for acoustic waves).


START
*
estimate time to nearest apogee or perigee
**
choose step size accordingly
is dr/dt small enough *** and the step size small enough? ****
yes
successful
no
diagnostic printout
set non error-checking integration mode.
set integration start.
advance the ray one integration step.
set integration start.
is ray convex away from surface?
yes
found crossing
no
is k_r parallel to $d\vec{k}_r/dt$?
yes
successful
no
tried 10 times yet?
no
yes
unsuccessful

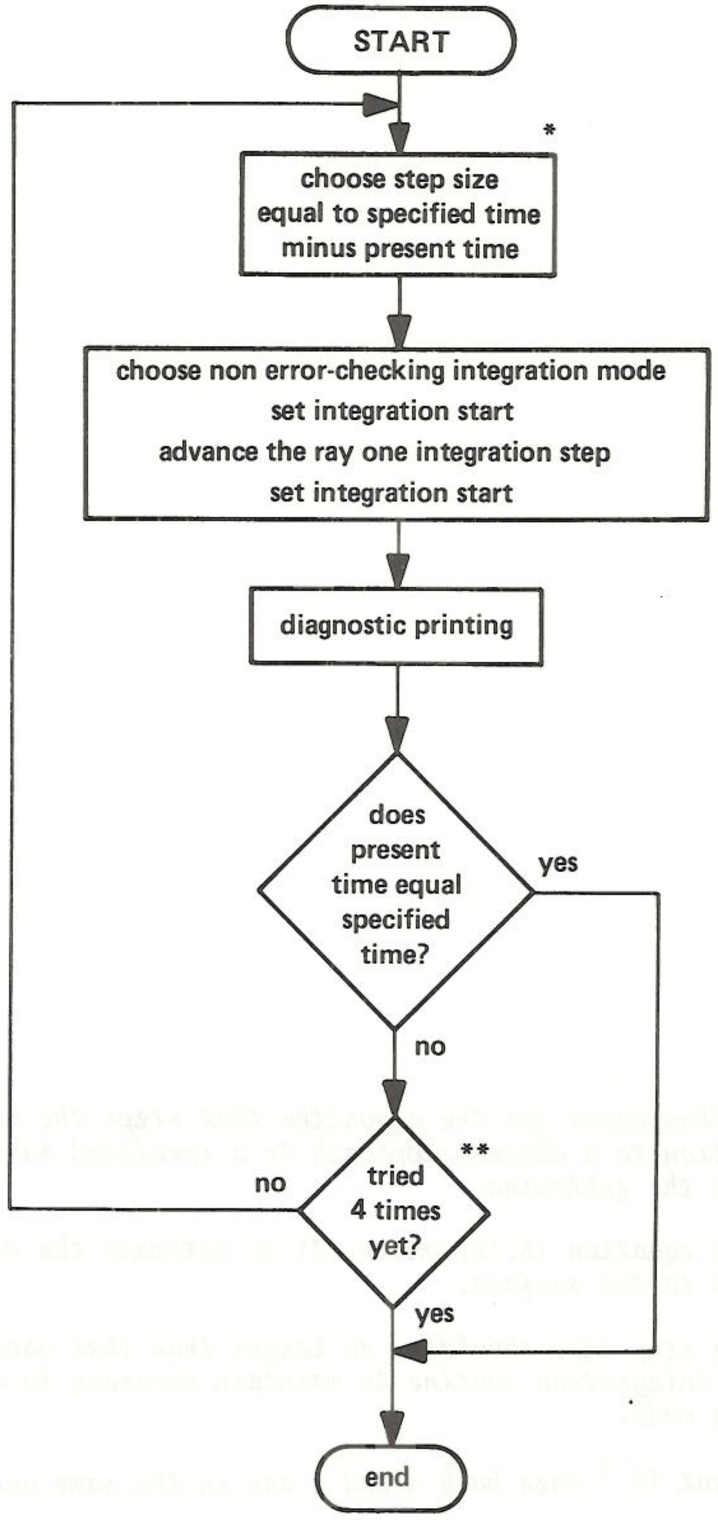
←Figure 21.--Flow chart for the algorithm that steps the ray by numerical integration to a closest approach to a specified surface. Asterisks refer to the following:

* Use equation (8.18) or (8.37) to estimate the time of closest approach to the surface.

** The step size should be no larger than that used by the numerical integration routine to maintain accuracy in the error-checking mode.

*** About 10^{-6} when both r and t are in the same units (usually km).

**** About 1 km (or 3 seconds for acoustic waves).


START
*
choose step size equal to specified time minus present time
choose non error-checking integration mode
set integration start
advance the ray one integration step
set integration start
diagnostic printing
does present time equal specified time?
yes
no
**
tried 4 times yet?
no
yes
end

← Figure 22.--Flow chart for the algorithm that steps the ray by numerical integration to a specified time. Asterisks refer to the following:

* The step size should be no larger than that being used by the numerical integration routine to maintain accuracy in the error-checking mode.

** During error checking, the Adams-Moulton predictor-corrector numerical integration method sometimes advances the ray 4 steps at once. It might require 4 steps to move the ray back to the place where the ray crossed the surface.

7. REFLECTING OR REFRACTING THE RAY

Once the intersection with the terrain has been found, the ray must be properly reflected. For an isotropic medium, this is straightforward. The algorithm must first project the wave vector into two components parallel to the surface and the component perpendicular to the surface. It then changes sign on the component perpendicular to the surface.

An anisotropic medium is more difficult. The two components parallel to the surface remain unchanged, as before, but the component perpendicular to the surface must be changed so that the dispersion relation is satisfied. Although this principle is the same for all media, the solution depends on the dispersion relation. At this point, we must specialize to the particular medium of interest, namely, acoustic waves in the presence of winds (or currents in the case of underwater acoustics).

We need to first separate the wave vector k into components perpendicular and parallel to the surface. Let $\vec{n}$ be a unit vector pointing out of the surface. Then the component of $\vec{k}$ normal to the surface is

$$k_\perp = \vec{k} \cdot \vec{n} \qquad , \qquad \vec{k}_\perp = (\vec{k} \cdot \vec{n})\, \vec{n} \tag{7.1}$$

and the part parallel to the surface is

$$\vec{k}_\parallel = \vec{k} - (\vec{k} \cdot \vec{n})\, \vec{n}\ . \tag{7.2}$$

The dispersion relation for acoustic waves in the presence of winds is

$$-\,\Omega^2 + C^2\, k^2 = 0 \tag{7.3}$$

where C is the sound speed and

$$\Omega \equiv \omega - \vec{k} \cdot \vec{v} \tag{7.4}$$

is the intrinsic frequency. ω is the wave frequency and $\vec{v}$ is the wind velocity.

With the help of (7.4), we can separate (7.3) as follows:

$$-(\omega - \vec{k}_\perp \cdot \vec{v} - \vec{k}_{\parallel} \cdot \vec{v})^2 + C^2\, k_\perp{}^2 + C^2\, k_{\parallel}{}^2 \,. \tag{7.5}$$

We want to solve (7.5) for $k_\perp$, assuming $\vec{k}_{\parallel}$ to be known. We can rewrite (7.5) as

$$(C^2 - v_\perp{}^2)\, k_\perp{}^2 + 2\, \Omega_{\parallel}\, v_\perp\, k_\perp - \Omega_{\parallel}{}^2 + C^2\, k_{\parallel}{}^2 = 0 \tag{7.6}$$

where

$$\Omega_{\parallel} \equiv \omega - \vec{k}_{\parallel} \cdot \vec{v} \tag{7.7}$$

and

$$\vec{v}_\perp \equiv (\vec{v} \cdot \vec{n})\, \vec{n} \qquad , \qquad v_\perp \equiv \vec{v} \cdot \vec{n} \tag{7.8}$$

is the component of wind normal to the surface. The quadratic formula gives the solution to (7.6) as

$$k_\perp = \frac{-\,\Omega_{\parallel}\, v_\perp \pm C\, \sqrt{\Omega_{\parallel}{}^2 - k_{\parallel}{}^2\, (C^2 - v_\perp{}^2)}}{C^2 - v_\perp{}^2} \,. \tag{7.9}$$

One solution of (7.9) should be the normal component of $\vec{k}$ for the incident wave, the other the normal component of $\vec{k}$ for the reflected wave. To convert from the incident wave to the reflected wave, it is necessary simply to change the sign of

$$k_\perp + \frac{\Omega_{\parallel}\, v_\perp}{C^2 - v_\perp{}^2} \,. \tag{7.10}$$

To do this, we can use (7.1) and (7.2) to write

$$\vec{k} = \vec{k} - (\vec{k} \cdot \vec{n})\vec{n} + (\vec{k} \cdot \vec{n})\vec{n} \,. \tag{7.11}$$

This is equivalent to

$$\vec{k} = \vec{k} - (\vec{k} \cdot \vec{n})\vec{n} - \frac{\Omega_{\parallel}\, v_{\perp}}{C^2 - v_{\perp}^{\,2}}\,\vec{n} + (\vec{k} \cdot \vec{n})\vec{n} + \frac{\Omega_{\parallel}\, v_{\perp}\,\vec{n}}{C^2 - v_{\perp}^{\,2}}\ . \tag{7.12}$$

Let us assume that (7.12) applies to the incident wave. That is

$$\vec{k}_{inc} = \vec{k}_{inc} - (\vec{k}_{inc} \cdot \vec{n})\vec{n} - \frac{\Omega_{\parallel}\, v_{\perp}}{C^2 - v_{\perp}^{\,2}}\,\vec{n} + (\vec{k}_{inc} \cdot \vec{n})\vec{n} + \frac{\Omega_{\parallel}\, v_{\perp}\,\vec{n}}{C^2 - v_{\perp}^{\,2}}\ , \tag{7.13}$$

where the subscript inc signifies the incident wave. To get the wave vector for the reflected wave, we need only reverse the sign of the last two terms in (7.13). That is,

$$\vec{k}_{ref} = \vec{k}_{inc} - 2(\vec{k}_{inc} \cdot \vec{n})\vec{n} - \frac{2\Omega_{\parallel}\, v_{\perp}\,\vec{n}}{C^2 - v_{\perp}^{\,2}}\quad , \tag{7.14}$$

where the subscript ref signifies the reflected wave.

To be more explicit, we can write (7.14) in terms of components. We assume an earth-centered spherical polar coordinate system (r, θ, ϕ). We then consider Cartesian components of $\vec{k}$, (k_r, k_θ, k_ϕ) in the r, θ, and ϕ directions. We also consider components of n, (n_r, n_θ, n_ϕ) in the r, θ, and ϕ directions. Then (7.14) is equivalent to

$$\begin{aligned}
k_{r\ ref} &= k_{r\ inc} - 2(\vec{k}_{inc} \cdot \vec{n})n_r - \frac{2\Omega_{\parallel}\, v_{\perp}\, n_r}{C^2 - v_{\perp}^{\,2}} && \text{(a)}\\
k_{\theta\ ref} &= k_{\theta\ inc} - 2(\vec{k}_{inc} \cdot \vec{n})n_\theta - \frac{2\Omega_{\parallel}\, v_{\perp}}{C^2 - v_{\perp}^{\,2}}\, n_\theta && \text{(b)}\quad (7.15)\\
k_{\phi\ ref} &= k_{\phi\ inc} - 2(\vec{k}_{inc} \cdot \vec{n})n_\phi - \frac{2\Omega_{\parallel}\, v_{\perp}}{C^2 - v_{\perp}^{\,2}}\, n_\phi\quad . && \text{(c)}
\end{aligned}$$

We might wonder whether it is realistic to allow a component of the wind normal to the surface of the terrain. Rather than debate this point here, I will simply point out that the ray tracing program in which we plan to apply these algorithms has no automatic provision that requires wind velocity models to have no component of the wind normal to the terrain. Perhaps such a provision should be added in the future. (Perhaps the tangential wind should also be required to be zero at the surface.) At present, however, the last term in (7.15) is needed to guarantee that the dispersion relation be satisfied for the reflected wave.

Let us briefly consider the case of refraction for a wave that is transmitted across the boundary surface where the sound speed and wind velocity change discontinuously. As with reflection, the parallel components of $\vec{k}$ are continuous across the boundary. The normal component of $\vec{k}$ is found by taking the appropriate solution of (7.9), evaluated by using the sound speed and wind velocity for the medium of the transmitted wave. Under ordinary conditions (the wind speed smaller than the sound speed) and with the convention that n is positive when pointing into the medium of the incident wave, the appropriate sign to choose for the transmitted wave in (7.9) is the negative sign. Let us now return to (7.11). This can be written

$$\vec{k} = \vec{k} - (\vec{k} \cdot \vec{n})\vec{n} + k_\perp \vec{n} \quad . \tag{7.16}$$

Equation (7.16) is valid for both the incident wave and the transmitted wave. However, the value of $k_\perp$ is different for the two. For the incident wave, $k_\perp$ is given by (7.1). For the transmitted wave, it is (from (7.9))

$$k_{\perp T} = \frac{-\Omega_{\parallel T}\, v_{\perp T} - C_T \sqrt{\Omega_{\parallel T}^{\,2} - k_\parallel^{\,2}\,(C_T^{\,2} - v_{\perp T}^{\,2})}}{C_T^{\,2} - v_{\perp T}^{\,2}} \tag{7.17}$$

where the subscript T refers to the medium of the transmitted wave. Thus, for the transmitted wave, we have (from (7.16))

$$\vec{k}_T = \vec{k}_{inc} - (\vec{k}_{inc} \cdot \vec{n})\vec{n} + k_{\perp T}\,\vec{n} \quad . \tag{7.18}$$

Explicitly, the parameters for the medium of the transmitted wave are

$$\Omega_{\parallel T} = \omega - \vec{k}_{\parallel} \cdot \vec{v}_T \tag{7.19}$$

and

$$v_{\perp T} = \vec{v}_T \cdot \vec{n} \ . \tag{7.20}$$

8. ESTIMATING THE TIME OF NEAREST INTERSECTION

The simplest method for predicting whether an extension of the ray path from a point will intersect the terrain surface (or the receiver surface) is to extend the ray in a straight line and see if it meets the terrain when the surface is extended as a plane by using the local slope. However, whenever the curvature of the ray and the terrain are small enough for that approximation to be useful, the ray would probably eventually go below the terrain on one of the integration steps and the simplest algorithm would recognize the intersection.

Because we want an algorithm sophisticated enough to estimate the time of nearest intersection (if one occurs) in difficult cases like those in Figs. 3, 4, 5, 11, or 15, we should include at least the local curvature in any approximations. Also, because it would be difficult to deal with a higher order approximation, a quadratic approximation to both the ray and the terrain seems to be the best compromise.

In addition, when searching for an intersection with the terrain (or receiver) surface, the intersection must be in the correct direction. In the case of the terrain, the intersection will always be downward. In the case of the receiver surface, the correct direction of crossing will alternate from one crossing to the next, but for each crossing, the direction will be determined. For the purposes of the present development, it is useful to define a parameter S that is equal to +1 if the wanted crossing is upward and -1 if the wanted crossing is downward. Thus the value of S will always be -1 for a terrain crossing.

8.1 For Single-Valued Height Function Models of the Topography

Assume that the topography (or the receiver surface) is specified in terms of a height $z(\theta, \phi)$ that depends on colatitude θ and longitude ϕ. We will require the function $z(\theta, \phi)$ to be continuous through the second derivative. We will require that the subroutine defining the surface will furnish first derivatives

$$z_\theta \equiv \partial z/\partial\theta \tag{8.1}$$

$$z_\phi \equiv \partial z/\partial\phi \tag{8.2}$$

and second derivatives

$$z_{\theta\theta} \equiv \partial^2 z/\partial\theta^2 \tag{8.3}$$

$$z_{\phi\phi} \equiv \partial^2 z/\partial\phi^2 \tag{8.4}$$

$$z_{\theta\phi} = z_{\phi\theta} \equiv \partial^2 z/\partial\theta\,\partial\phi \tag{8.5}$$

that are continuous functions of θ and ϕ. Furthermore, the first and second derivatives furnished by the subroutine must be mathematically consistent with the variation of z with θ and ϕ.

The coordinates of the ray in spherical polar coordinates are (r, θ, ϕ). If r_e is the radius of the earth, then the height of the ray above the surface is

$$h = r - r_e - z\ . \tag{8.6}$$

Let us assume that the ray path has approximately constant second derivatives with respect to the group delay time t. That is,

$$r = r_1 + \dot{r}_1(t - t_1) + \frac{1}{2}\,\ddot{r}_1(t - t_1)^2 \tag{8.7}$$

$$\theta = \theta_1 + \dot{\theta}_1(t - t_1) + \frac{1}{2}\ddot{\theta}_1(t - t_1)^2 \tag{8.8}$$

$$\phi = \phi_1 + \dot{\phi}_1(t - t_1) + \frac{1}{2}\ddot{\phi}_1(t - t_1)^2 \ . \tag{8.9}$$

The first derivatives to use in (8.7) through (8.9) are furnished by the ray tracing program. The second derivatives are found by the approximations in (1.7) through (1.9). The subscript 1 in (8.7) through (8.9) refers to the time t_1 where the ray path is assumed to have known direction and approximately known second derivatives.

Consider the derivative of (8.6) with respect to group time.

$$\dot{h} = \dot{r} - \dot{z} = \dot{r} - z_\theta \dot{\theta} - z_\phi \dot{\phi} \ . \tag{8.10}$$

Then the second derivative with respect to time is

$$\ddot{h} = \ddot{r} - z_\theta \ddot{\theta} - z_\phi \ddot{\phi} - z_{\theta\theta} \dot{\theta}^2 - z_{\phi\phi} \dot{\phi}^2 - 2z_{\theta\phi} \dot{\theta}\dot{\phi} \ . \tag{8.11}$$

If we assume that the second derivative in (8.11) is nearly constant for small time intervals, then we can write

$$h = h_1 + \dot{h}_1(t - t_1) + \frac{1}{2}\ddot{h}_1(t - t_1)^2 \tag{8.12}$$

in analogy with (8.7) through (8.9). We want to find the value of t for which the ray intersects the surface, that is, where h = 0. Let t_c be the value for which h = 0. Then

$$0 = h_1 + \dot{h}_1(t_c - t_1) + \frac{1}{2}\ddot{h}_1(t_c - t_1)^2 \ . \tag{8.13}$$

Let us now for simplicity drop the subscript 1 in (8.13), and simply remember that h and t without subscript refer to the particular point on the ray path that corresponds to the integration step where we are trying to estimate the time t_c where the ray will intersect the surface. Then (8.13) becomes

$$0 = h + \dot{h}(t_c - t) + \frac{1}{2}\ddot{h}(t_c - t)^2 \ . \tag{8.14}$$

The solution of (8.14), for which the ray is going up when S = +1 and going down when S = -1, is

$$t_c - t = \frac{-\dot{h} + S\sqrt{\dot{h}^2 - 2h\ddot{h}}}{\ddot{h}} \ . \tag{8.15}$$

Within the approximation being made here, the ray will intersect the surface if

$$\dot{h}^2 - 2h\,\ddot{h} > 0 \tag{8.16}$$

but will make a closest approach to the surface if

$$\dot{h}^2 - 2h\,\ddot{h} < 0 \ . \tag{8.17}$$

In the latter case, we can estimate the time t_p at which h is a minimum.

$$t_p - t = -\frac{\dot{h}}{\ddot{h}} \ . \tag{8.18}$$

This is close to the time when the ray makes a closest approach to the surface. If the second derivative in (8.11) is very small, then the formula in (8.15) may be impractical. In that case, the solution

$$t_c - t = \frac{-2\ h}{\dot{h} + S\sqrt{\dot{h}^2 - 2h\ddot{h}}} \tag{8.19}$$

is more useful. The advantage of (8.19) is that it is uniformly valid as $\ddot{h}$ approaches zero.

Unit normal directions from the surface

Section 7 requires unit normal directions to the surface. It is appropriate here to calculate the components of the unit normal in terms of the local derivatives. The connecting relations are

$$n_\theta = -\frac{n_r z_\theta}{r} \tag{8.20}$$

$$n_\phi = -\frac{n_r z_\phi}{r \sin \theta} \tag{8.21}$$

$$n_r^2 + n_\theta^2 + n_\phi^2 = 1 . \tag{8.22}$$

The solution of (8.20) through (8.22) is

$$n_r = \frac{1}{\sqrt{1 + \frac{z_\theta^2}{r^2} + \frac{z_\phi^2}{r^2 \sin^2 \theta}}} \tag{8.23}$$

$$n_\theta = -\frac{z_\theta}{\sqrt{r^2 + z_\theta^2 + \frac{z_\phi^2}{\sin^2 \theta}}} \tag{8.24}$$

$$n_\phi = -\frac{z_\phi}{\sqrt{r^2 \sin^2\theta + z_\theta^2 \sin^2\theta + z_\phi^2}} , \tag{8.25}$$

where the unit normal pointing upward has been chosen.

8.2 For Arbitrary Surface Models of the Topography

An arbitrary model of a surface can be expressed in the form

$$f(r, \theta, \phi) = 0 . \tag{8.26}$$

Because f is zero only on the surface, it always has one sign on one side of the surface and the opposite sign on the other side. Let us call the side of the surface that is underground the inside and the other side the outside. Then we can arbitrarily require that f be positive outside of the surface and negative inside the surface. We can similarly designate an inside and an outside for the receiver surface and make the same requirement on f.

Thus, we have

$$f(r,\ \theta,\ \phi) > 0 \tag{8.27}$$

outside the surface and

$$f(r,\ \theta,\ \phi) < 0 \tag{8.28}$$

inside the surface. The time derivative of f is

$$\dot{f} = f_r\,\dot{r} + f_\theta\,\dot{\theta} + f_\phi\,\dot{\phi} \tag{8.29}$$

where a subscript indicates a partial derivative as in (8.1) through (8.5). The time derivative of (8.29) is

$$\ddot{f} = f_r\,\ddot{r} + f_\theta\,\ddot{\theta} + f_\phi\,\ddot{\phi} + f_{rr}\,\dot{r}^2 + f_{\theta\theta}\,\dot{\theta}^2 + f_{\phi\phi}\,\dot{\phi}^2 + 2f_{r\theta}\,\dot{r}\dot{\theta} + 2f_{r\phi}\,\dot{r}\dot{\phi} + 2f_{\theta\phi}\,\dot{\theta}\dot{\phi} \tag{8.30}$$

Assuming that (8.30) is nearly constant locally, we have

$$f = f_1 + \dot{f}_1(t - t_1) + \frac{1}{2}\ddot{f}_1(t - t_1)^2\ , \tag{8.31}$$

where the subscript 1 refers to the values at the time t_1. We want to find the value of t for which the ray intersects the surface, that is where f = 0. Let t_c be the value for which f = 0. Then

$$0 = f_1 + \dot{f}_1(t_c - t_1) + \frac{1}{2}\ddot{f}_1(t_c - t_1)^2\ . \tag{8.32}$$

For simplicity, let us drop the subscript 1 in (8.32), but remember that f and t without subscripts refer to the integration step on the ray path where we are trying to estimate the time t_c where the ray will intersect the surface. Then (8.32) becomes

$$0 = f + \dot{f}(t_c - t) + \frac{1}{2} \ddot{f}(t_c - t)^2 \ . \tag{8.33}$$

The solution of (8.33), for which the ray is crossing from inside to outside when S = +1 and crossing from outside to inside when S = -1, is

$$t_c - t = \frac{-\dot{f} + S\sqrt{\dot{f}^2 - 2f\ddot{f}}}{\ddot{f}} \ . \tag{8.34}$$

Within the approximation made here, the ray will intersect the surface if

$$\dot{f}^2 - 2f\ddot{f} > 0 \tag{8.35}$$

but will make a closest approach to the surface if

$$\dot{f}^2 - 2f\ddot{f} < 0 \ . \tag{8.36}$$

In the latter case, we can estimate the time t_p at which f is a minimum.

$$t_p - t = -\frac{\dot{f}}{\ddot{f}} \ . \tag{8.37}$$

This is close to the time when the ray makes a closest approach to the surface. If the second derivative in (8.30) is very small, then the formula in (8.34) may be impractical. In that case, the solution

$$t_c - t = \frac{-2f}{\dot{f} + S\sqrt{\dot{f}^2 - 2f\ddot{f}}} \tag{8.38}$$

is more useful. The advantage of (8.38) is that it is uniformly valid as $\ddot{f}$ approaches zero.

Notice that the representation in Section 8.1 is a special case of the present representation where

$$f(r, \theta, \phi) = r - r_e - z(\theta, \phi) . \tag{8.39}$$

<u>Unit normal directions from the surface</u>

As in Section 8.1, we need to derive the formulas for the components of the unit normal to the surface. Because f is a constant along the surface, the gradient of f is in the same direction as the unit normal. That is

$$\vec{\nabla} f = f_r \hat{i}_r + \frac{f_\theta}{r} \hat{i}_\theta + \frac{f_\phi}{r \sin\theta} \hat{i}_\phi \propto \vec{n} . \tag{8.40}$$

Taking the ratio of components gives

$$n_\theta = \frac{f_\theta}{r f_r} n_r \tag{8.41}$$

and

$$n_\phi = \frac{f_\phi}{f_r r \sin\theta} n_r . \tag{8.42}$$

The solution of (8.41), (8.42), and (8.22) is

$$n_r = \frac{f_r}{\sqrt{f_r^2 + \frac{f_\theta^2}{r^2} + \frac{f_\phi^2}{r^2 \sin^2\theta}}} , \tag{8.43}$$

$$n_\theta = \frac{f_\theta}{r\sqrt{f_r^2 + \frac{f_\theta^2}{r^2} + \frac{f_\phi^2}{r^2 \sin^2\theta}}} , \tag{8.44}$$

and

$$n_\phi = \frac{f_\phi}{r \sin\theta \sqrt{f_r^2 + \frac{f_\theta^2}{r^2} + \frac{f_\phi^2}{r^2 \sin^2\theta}}} \quad . \tag{8.45}$$

As a check for consistency with Section 8.1, we can take the partial derivatives of (8.39) to give

$$f_r = 1 \ , \tag{8.46}$$

$$f_\theta = - z_\theta \ , \tag{8.47}$$

and

$$f_\phi = - z_\phi \ . \tag{8.48}$$

Substituting (8.46) through (8.48) into (8.43) through (8.45) gives (8.23) through (8.25), in agreement with Section 8.1.

9. INTEGRATING THE HEIGHT ABOVE THE TERRAIN

If the numerical integration of the ray path took small enough steps it would not be necessary to use a sophisticated algorithm to look for intersections of the ray path with the terrain. With small enough steps, all intersections of the ray path with the terrain would automatically be found.

However, it would be too expensive to use such small integration steps. We do not want to use integration steps any smaller than necessary to give the desired accuracy to the ray path calculation. For that reason, I have developed the sophisticated algorithms presented here.

As pointed out previously, however, these algorithms are not foolproof. It is possible to use numerical integration steps large enough that some intersections with the terrain would still be missed. This can happen when the steps are much larger than the scale of the structure of the terrain (if in one integration step, the ray passes by several ridges or hills, for example).

I cannot think of simple modifications to the algorithms to take care of these kinds of difficult cases. There is a compromise solution. It is to use an integration step size small enough that these very difficult cases do not occur. That will allow the algorithms developed here to be adequate. It would be desirable, however, to have the step size reduced only near regions where such difficulties arise.

This is partly possible in the following way. The numerical integration method used in our ray-tracing program (Jones and Stephenson, 1975) varies the integration step length as needed to maintain the requested accuracy. When the equations being integrated are changing rapidly, the integration routine will decrease the step size. When the equations being integrated are changing slowly, the integration routine will increase the step length to save computer time. If we add an equation to integrate the height of the ray above the terrain to the system of differential equations already being integrated, then the integration routine will not let the step size become larger than the scale of the structure of the terrain below the ray.

Unfortunately, the step size will not be affected by how close the ray is to the terrain. That is, when the ray is far above the terrain where there is no possibility of intersecting the terrain, the step size will be kept just as small as when the ray is close to the terrain. However, if some parts of the terrain are smoother than others, the step size over the smoother parts will be larger.

For single-valued height function models of the topography, the equation to add to the system of differential equations is (8.10). The partial derivatives are furnished by the terrain model, the time derivatives by the differential equations for the ray.

For arbitrary surface models of the topography, the equation to add to the system of differential equations is (8.29). Again, the partial derivatives are furnished by the terrain model, the time derivatives by the differential equations for the ray.

In the ray-tracing program we use (Jones and Stephenson, 1975), whether to integrate any of the auxiliary differential equations is optional (and controlled by input data). Thus, once the above differential equation has been added to the system, the user of the ray-tracing program can decide whether to integrate the equation, depending on how complicated his terrain model is. Most terrain models would probably not require this integration to keep from missing intersections with the terrain.

10. ACKNOWLEDGMENTS

I thank Dr. T.M. Georges for his comments on the manuscript, especially concerning the completeness and unambiguity of the algorithms. I thank J.P. Riley for his comments on the manuscript, especially concerning the direct conversion of the algorithms to computer subroutines and for suggestions for improving the formulas to estimate the time of intersection of the ray path with the terrain. I thank the editorial staff of Publication Services for helping me clarify the expression of my ideas. Such extensive help is difficult not to understate. I thank Dr. C.W. Spofford of Science Applications, Inc., McLean, Virginia, for pointing out a situation (Fig. 11) where my original algorithms would have failed. This led me to modify the algorithms to handle that case. I especially thank Ms. Mildred Birchfield for her fast accurate typing of the manuscript, for making the equations so readable, for correcting my misspellings, for returning my manuscripts promptly, and staying so friendly through it all.

11. REFERENCE

Jones, R. Michael and Judith J. Stephenson, A versatile three-dimensional ray tracing computer program for radio waves in the ionosphere, OT Report 75-76, Office of Telecommunications, U.S. Department of Commerce, October 1975.