High-Speed Ray Tracing for Underwater Sound, A Status Report

APL-UW 8109
October 1981

High-Speed Ray Tracing for Underwater Sound, A Status Report

by

Robert P. Goddard
David W. Princehouse

APL-UW 8109
October 1981

Applied Physics Laboratory
University of Washington

Contract No. N00024-78-C-6018
N00024-81-C-6042

ACKNOWLEDGMENTS

We gratefully acknowledge the contributions made by several individuals and organizations to this project. Most of the software was written by Diane Thompson and Van Casey of APL-UW. Jim Grembi of NAVSEA PMS 406-T2 was instrumental in arranging for Advanced Development support and funding via the Advanced Lightweight Torpedo project. Bob Hearn, Bob Dukelow and Ron Dahlseid of the Hybrid Simulator group at the Naval Ocean Systems Center, San Diego, provided valuable encouragment and counsel. Comments from many people on the technical staff and in the editorial department at APL, and from Jon Lutton, resulted in substantial improvements in style and content.

ABSTRACT

Computer simulations of underwater sound are often limited by the computer time required to trace the many paths (rays) alcng which sound may propagate. We have developed a high-speed ray tracing technique that is applicable to situations in which the sound speed depends on depth only, and not on horizontal position or time. Included in the model are specular reflections from a flat surface and a flat bottom, and ray bending in a vertical plane. Diffractive effects, such as propagation through caustics and leakage from ducts, are not modeled at this time.

The technique is fast because the time-consuming parts of the calculations are done before the simulation starts, and the results are stored in tables. During the simulation, these results are retrieved from the tables and interpolated. The stored data consist of three functions of two variables, tabulated on a nonuniform two-dimensional grid. A doubly linked tree structure with extra links to neighboring leaves is used to permit rapid access to neighboring points in the grid.

Software exists to create and fill the tables and to use the tables to plot rays and find eigenrays. Good accuracy, in most cases, can be achieved using tables of less than 100 kilobytes. Modifications are planned that will reduce the table size, improve reliability, and make the software easier to use. Our goal is to include ray tracing in simulators such as REVGEN and the NOSC Hybrid Simulator that are now limited to direct straight-line propagation.

CONTENTS

1. INTRODUCTION

Computer simulations of underwater sound play an important role in the development and evaluation of sonar systems and underwater guidance and control systems for military and civilian agencies. The sound propagation (ray tracing) calculation, which is the part of the simulation program that computes the amplitude, direction, and time delay of the signal from a given source, typically is executed many thousands of times to produce a few seconds of simulated reverberation from a single sonar ping. Therefore, if the simulation is to run in real time or nearly real time, the sound propagation software must be very fast. This report addresses this problem by introducing a new ray tracing technique that is capable of drastically reducing the time required for sound propagation calculations while retaining acceptable accuracy. The method has the potential to permit simulations in real time for a non-isovelocity ocean.

An example of a current system that requires rapid ray propagation calculations is the Hybrid Simulator at NOSC,* in which the new technique described here may ultimately be used. The Hybrid Simulator is a development and evaluation tool for acoustic torpedoes. It computes reverberation, target, noise, and countermeasure signals and presents them to torpedo hardware in a closed-loop simulation. It must therefore run in real time. The essential requirement of this simulation is that it must perform an enormous amount of ray tracing for a single sound speed profile to provide enough reverberation for an entire torpedo-submarine encounter. To produce this amount of reverberation in real time, however, the present Hybrid Simulator is limited to isovelocity (i.e., straight-line) direct ray propagation.

Unfortunately, underwater sound rarely travels in straight lines. The sound speed is usually nonuniform, and therefore the sound follows curved paths. Also, reflections from the surface and bottom are often important. Therefore simulations using straight-line direct propagation fail to reproduce important phenomena such as shadow zones, caustics, and multiple images. Several sound propagation models that include these phenomena are available (Ref. 1), and a few have been used in simulations; however, all of these models are far too slow for real-time simulations involving more than a few isolated scatterers. Even non-real-time simulations including ray bending tend to use an unrealistically small number of scatterers for reverberation, simply because a more complete simulation would be too expensive. Thus, high-speed ray tracing would permit more realism in both real-time and non-real-time simulations.

*Naval Ocean Systems Center, San Diego, California

The high-speed ray tracing technique described in this report takes advantage of the observation that the sound speed profile to be used in a simulation is usually known well before the simulation starts. Therefore, we can afford a large amount of analysis before the simulation. Our method derives its speed from the fact that the time-consuming parts of the calculation are done only once for each new sound speed profile, and the results are stored in a set of tables. Thereafter, many different rays can be traced rapidly using table lookup and interpolation. The tables are stored on disk or tape; hence, they can be generated long before the actual simulation, when time is much less critical.

The sound speed is assumed to be a function of depth only, and independent of range and time. Ray bending and specular reflections from the surface and bottom are included in the model, but diffraction and other wave phenomena are not. The data stored in the tables consist of three functions of two variables, tabulated on a nonur form two-dimensional grid, plus some auxiliary data. The data structure of the tables can be described as a doubly-linked tree with additional cross-links between neighboring leaves. This structure was chosen to permit rapid access to a sequence of values at neighboring points in the grid. This capability is especially important in searches for "eigenrays" connecting two given points in the ocean (Ref. 2).

The software described in this report was designed to test the algorithm, and does not constitute a complete, useful program package. On the input side, it requires ray tracing software (which need not be fast) to compute the data to be stored in the tables. We use subroutines borrowed from the Navy's NISSM II program (Ref. 3) in our demonstration software. This choice is not essential to the technique; others probably would work also. On the output side, the software is not really useful unless it is embedded in a simulation program designed to make effective use of its speed. Such a program has not yet been written.

At this writing, software exists to create and fill the tables and to use the tables to plot rays and find eigenrays. We have found that good accuracy, in most cases, can be achieved using tables of reasonable size (less than 100 kilobytes). Speed measurements would not be meaningful at this point because the code is full of diagnostic tests and has not been optimized, but we expect that our table lookup technique will be at least an order of magnitude faster than straightforward integration. The actual speed will depend strongly on the accuracy required and on the characteristics of the hardware and software used for the simulation.

The future of research on this technique is somewhat uncertain. Major software modifications are required to improve the accuracy of the interpolation for nearly horizontal rays and to make the creation of the tables automatic (it is now interactive). We have suspended work on the ray tracing software, although we plan to return to ray tracing in late summer or fall of 1981 when work begins on Version 4.1 of the APL Reverberation Generator (REVGEN) simulation program, which will include nonisovelocity propagation. We will decide then whether to modify the existing software or to apply what we have learned to new software.

In the next section we outline the physical principles and assumptions on which the high-speed ray tracing algorithm is based. Section 3 describes the properties of the tabulated functions, which will make the reader aware of the problems inherent in the tabulation and interpolation of the functions. The structure of the tables and the relationship between the structure and the properties of the tabulated functions are given in Section 4. Section 5 contains a brief summary of the software we have written to create and use the tables; detailed documentation for the software is given in a separate informal report (unpublished). In the final two sections we summarize the present status of the project and the work that remains to be done.

?. PHYSICAL BACKGROUND AND ASSUMPTIONS

2.1 Ray Acoustics

Sound propagation can be described mathematically by solutions of the wave equation. Several methods are available to solve the wave equation directly (see Ref. 1), but these methods are impractical at high frequencies because of the large amount of computation required. Instead, high frequency sound propagation is usually approximated using a ray acoustics model, in which the sound energy is assumed to propagate along well-defined paths called "rays" (Ref. 2). A ray is roughly analogous to the path of a classical particle moving in a potential; it may be bent by a "force" proportional to the gradient of the sound speed or it may be reflected from a discontinuity in the sound speed (e.g., at the surface), but it will not interfere with neighboring rays.

Diffractive effects are ignored in the ray approximation. Therefore, phenomena that occur on a scale comparable to the wavelength of the sound, or smaller, are not reproduced accurately. Ray theory will fail if the sound speed or the sound intensity changes appreciably over a distance of one wavelength. In the vicinity of a caustic (Ref. 2, Section 6.3), ray theory predicts extremely sharp maxima in sound transmission due to focusing effects in the medium, whereas in the wave theory and in observations these maxima are much lower and wider. Ray theory also does not reproduce leakage of sound energy into shadow zones or out of ducts.

Nevertheless, simple ray theory is adequate for practical simulations in most situations, and it forms the starting point for various refinements in which the wave phenomena are introduced in an approximate way (e.g., Ref. 4). At least a theory with ray bending is more accurate than straight-line propagation.

2.2 Range Independence

Our second major set of assumptions is referred to collectively as "range independence." We assume that the sound speed depends only on depth, and not on horizontal position or time. We also assume that the depth of the water is the same everywhere, and that sound reflects from the surface and the bottom in the specular direction only (i.e., the angle of reflection equals the angle of incidence). These assumptions are important because they reduce the dimensionality of the problem. Any two rays launched from the same depth with the same angle from the horizontal have identical properties, except for simple translations and rotations. Hence, we can use two-dimensional tables to store the ray properties.

2.3 A Sample Ray Diagram

The behavior of acoustic rays in a range-independent medium is illustrated in Fig. 1. The left side of the figure shows the sound speed profile (SSP) on which the ray diagram is based. This is a wholly artificial (and quite unrealistic) SSP, designed to show the qualitative features expected in a real SSP. Because it has two subsurface ducts (sound speed minima), it is somewhat more complicated than the SSP's now used in most shallow water sonar simulations, but much simpler than SSP's found in studies of ocean microstructure. This same SSP is used in all of the examples in this report.

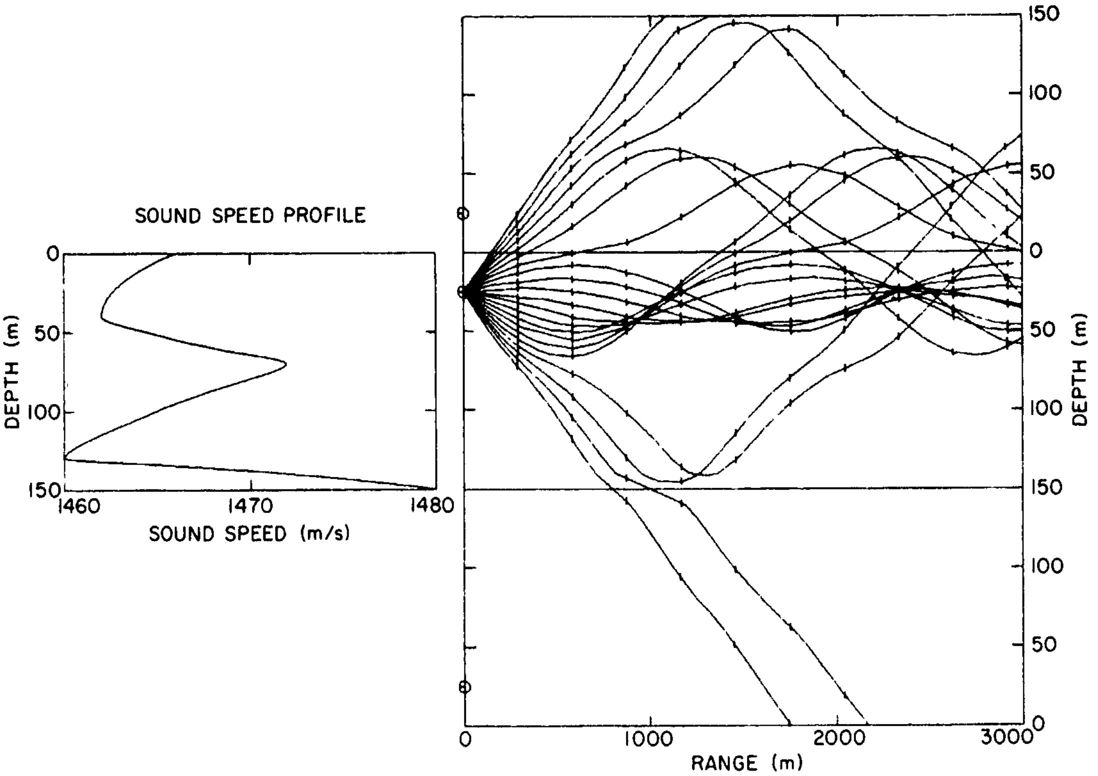

Figure 1. Sample ray diagram, shown on the right, for an artificial sound speed profile, shown on the left. Rays are launched at 1° intervals between -10° and +10° from the horizontal. Tick marks are at 0.2-second intervals. The three images represent rays which bounce once from the surface (top), those which bounce once from the bottom (bottom), and everything else (center).

The right side shows a fan of sound rays computed using this SSP. The rays represent paths along which sound energy flows to or from a sonar transducer at a depth of 25 meters. The rays are perpendicular to the wave fronts. Rays are shown at 1° intervals. The propagation time along each ray is shown by short vertical bars at 0.2-second intervals.

To avoid a confusing tangle of rays, the figure is divided into three sections representing three "images" of the ocean, in which unreflected (direct) rays are shown in the middle image and rays that are reflected once from the surface and bottom are shown upside down in the upper and lower images respectively. To reassemble the ray diagram in your mind, imagine folding the paper along the image boundaries and looking through the paper.

As the figure shows, the sound rays are bent toward the depths where the sound speed is slowest. A target located in a region where the rays are close together will seem relatively "loud" to the sonar; conversely, a target in a region where the rays are far apart or absent will seem relatively quiet, and thus difficult for the sonar to detect. In some regions there are several rays (called "multipaths") from the sonar to a given target location; for an active sonar, a target located in such a region would produce several echoes which would return to the sonar at slightly different times and from different elevations.

2.4 Decomposition of a Ray

Computation of acoustic rays in a range-independent medium is greatly simplified by three observations: there is a constant of the motion, and the rays are periodic and reversible. The constant of the motion is the vertex velocity C_V, defined by (Ref. 2)

$$C_V = \frac{C(Z)}{\cos\,\theta(Z)} \quad , \qquad (2\text{-}1)$$

where Z is the depth, C(Z) is the sound speed, and θ(Z) is the elevation (i.e., the angle of the ray from the horizontal). The vertex velocity is a constant along any given ray as a consequence of Snell's Law (Ref. 2, p. 116).

Note that C_V must be greater than or equal to C(Z) since the cosine of any angle is less than or equal to one. In fact, when $\theta = 0$, the ray reaches a vertex (an extremum in the depth of the ray), and $C_V = C(Z)$, which is the origin of the term vertex velocity. Note also that there are two distinct ray segments described by the same value of C_V, one launched upward and one launched downward, since cos (θ) is a symmetric function of θ.

The second simplifying observation is that the rays are periodic: eventually a ray will return to the starting depth at the starting angle and repeat itself. Therefore, if we know everything about one cycle of a ray, we can compute everything about the ray simply by adding more cycles (or partial cycles) end to end.

The third simplifying observation is that the rays are reversible: The downward-traveling parts of a ray are just mirror images of the upward-traveling parts, reflected through a plane perpendicular to the ray at a vertex. Therefore, we can go one step further: If we know everything about one half-cycle of a ray, say from a depth minimum to the next depth maximum, we can compute everything about the ray by alternately adding half-cycles and their mirror images end to end.

The decomposition of a ray is illustrated in Fig. 2, which shows schematically a ray that starts from a sonar depth Z_S at an elevation θ_S upward. (By convention, a positive elevation corresponds to increasing depth, so $\theta_S < 0$ and $\theta_T > 0$ in the example.) The ray reaches an upper vertex at point B, then propagates downward to a lower vertex at E, and reaches a target at point F, depth Z_T. The entire ray is characterized by a single value of the vertex velocity C_V, which is determined by the initial elevation θ_S and the sound speed $C(Z_S)$ at the sonar depth [see Equation (2-1)].

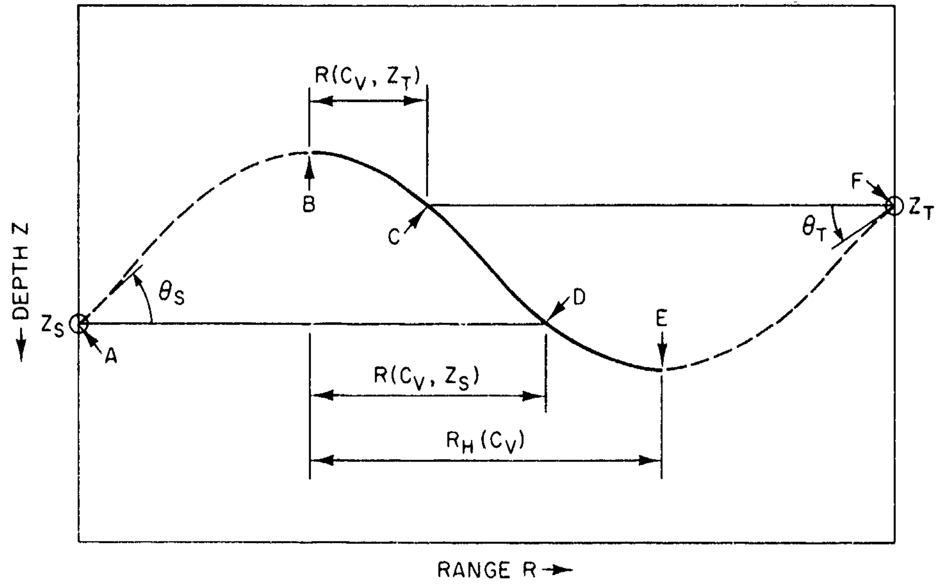

Figure 2. Decomposition of a ray into segments. The symbols are explained in the text.

Let us focus for the moment on the solid portion of the curve, from B to E. The shape of this segment can be characterized by a single function $R(C_V,Z)$, which is defined as the horizontal range from the minimum-depth point B to the point where the ray first reaches a depth Z. As a consequence of our simplifying observations (which are valid only for "range-independent" media), the function $R(C_V,Z)$ is sufficient to characterize the shape of the entire ray. Segment AB is just a mirror image of segment BD, reflected about a vertical line through point B; hence $R_{AB} = R(C_V,Z_S)$. Segment EF is a mirror image of segment CE, reflected about a vertical line through point E; hence $R_{EF} = R_{BE} - R(C_V,Z_T)$. Segment BE is half of an oscillation, and the corresponding horizontal range is the "half-wave range" $R_H(C_V)$ defined by

$$R_H(C_V) \equiv R[C_V,\ Z_{max}(C_V)] \quad , \tag{2-2}$$

where $Z_{max}(C_V)$ is the maximum depth reached by the ray. Therefore, the total range R_{AF} is given by

$$R_{AF} = R_{AB} + R_{BE} + R_{EF} = 2R_H(C_V) + R(C_V,Z_S) - R(C_V,Z_T) \quad . \tag{2-3}$$

The result (2-3) can be generalized in a number of ways. If the ray reflects off the surface or bottom or both, instead of turning around at a vertex, Eq. (2-3) still holds provided $R(C_V,Z)$ is defined as the range from the minimum depth (upper vertex or surface intervention) and $R_H(C_V)$ is defined as the range from minimum to maximum depth (lower vertex or bottom intersection). The ray may oscillate any number of times, so an even number of $R_H(C_V)$ contributions may be added. Finally, the elevations θ_S and θ_T can be either positive or negative; this affects the signs of the second and third terms in (2-3). The general form of (2-3) is given by

$$R_{ST} = 2N_B R_H(C_V) - S_S R(C_V,Z_S) - S_T R(C_V,Z_T) \quad , \tag{2-4}$$

where R_{ST} is the horizontal range from sonar to target, Z_S and Z_T are the sonar and target depths, N_B (a nonnegative integer) is the number of lower vertices or bottom intersections, $R_H(C_V)$ is the half-wave range (2-2), and $R(C_V,Z)$ is the range function defined before. The factors S_S and S_T may have the values +1 or -1; their signs are the signs of the elevations θ_S and θ_T at the sonar and target respectively (positive = down).

2.5 Propagation Loss and Time Delay

Once we find a solution to Eq. (2-4), we need two more pieces of information: the propagation loss and the time delay. In general, the propagation loss has several components, including reflection losses at the surface and bottom and the attenuation in the medium. For the present, we are concerned only with the "spreading loss," i.e., changes in signal strength due to the distribution of the energy flux over an ever-larger area as the sound gets farther away from its source. For straight-line propagation, the spreading loss P_{loss} (expressed as a reduction in pressure amplitude) is simply D^{-1}, where D is the distance along the ray path. If the rays bend, a more general expression holds (Ref. 3, Eq. 5):

$$|P_{loss}| = \left| C_V \tan\theta_S \tan\theta_T R_{ST} \frac{\partial R_{ST}}{\partial C_V} \right|^{-\frac{1}{2}} , \tag{2-5}$$

where $\partial R_{ST}/\partial C_V$ is obtained by differentiating the right side of (2-4) with respect to C_V.

The time delay is given by a formula analogous to Eq. (2-4):

$$T_{ST} = 2N_B\, T_H(C_V) - S_S\, T(C_V,Z_S) - S_T\, T(C_V,Z_T) \quad , \tag{2-6}$$

where T_{ST} is the time delay from sonar to target, $T(C_V,Z)$ is the time required for the ray to travel from its minimum-depth point (upper vertex or surface intersection) to depth Z, and the half-wave time $T_H(C_V)$ is the time required for the ray to travel from its minimum depth to its maximum depth:

$$T_H(C_V) \equiv T[C_V,Z_{max}(C_V)] \tag{2-7}$$

2.6 The Range and Time Functions

The quantities appearing in Eqs. (2-4) through (2-7) are given by the following integrals (Ref. 3):

$$R(C_V,Z) = \int_{Z_{min}(C_V)}^{Z} \frac{C(Z')\,dZ'}{\sqrt{C_V^2 - C^2(Z')}} \tag{2-8}$$

$$R_H(C_V) = R[C_V,Z_{max}(C_V)] \tag{2-9}$$

$$T(C_V,Z) = \int_{Z_{min}(C_V)}^{Z} \frac{C_V \, dZ'}{C(Z')\sqrt{C_V^2 - C^2(Z')}} \qquad (2\text{-}10)$$

$$T_H(C_V) = T[C_V, Z_{max}(C_V)] \quad , \qquad (2\text{-}11)$$

where $Z_{min}(C_V)$ is the depth of the upper vertex or surface intersection, and $Z_{max}(C_V)$ is the depth of the lower vertex or bottom intersection. We compute these integrals using NISSM II methods (Ref. 3), although other methods could be used without major changes in the high-speed ray tracing algorithms or software.

2.7 Eigenrays

Sound can travel from a given source to a given receiver (or scatterer) by many different paths. There are always paths that bounce one or more times from the surface or bottom, and if the sound speed is nonuniform, there may be several wholly-refracted paths too. Each path is characterized by a different combination of travel time, transmit and receive directions, propagation loss, and numbers of upper and lower vertices or reflections. These paths are called "eigenrays" (Ref. 2). Each eigenray corresponds to a solution of Eq. (2-4). To find the eigenrays between two given points, we must find those values of C_V, N_B, S_S, and S_T for which (2-4) is satisfied, for given values of Z_S, Z_T, and R_{ST}. A typical simulation may involve tens or hundreds of thousands of scatterers, and as many as ten significant eigenrays per scatterer. Therefore, one of our primary objectives is to find eigenrays efficiently.

Algorithms for finding eigenrays generally involve an iterative search. One such algorithm goes roughly as follows: Starting at some minimum value of C_V, we solve Eq. (2-4) for N_B for four cases, corresponding to the four combinations of the signs S_S and S_T. The resulting values of N_B, which we denote N_{++}, N_{+-}, N_{-+}, and N_{--}, usually are not nonnegative integers. We truncate them to the next lower integer, increment C_V by some amount, and repeat the process, comparing the four N_B values with the previous values at each step. Eventually we find that one of the truncated values, say N_{++}, changes by one unit (or more) from one step to the next. At that point we know that a solution to (2-4), with an integer value of N_B and $S_S = S_T = +1$, exists for a C_V value somewhere between the last two test values. The solution can be found by inverse interpolation or by an iterative technique such as regula falsi (Ref. 5). The search then continues until all significant eigenrays are found.

The speed of this algorithm is strongly dependent on two factors: (a) minimizing the time required to evaluate the range functions $R(C_V,Z)$ and $R_H(C_V)$ at each step in the search, and (b) choosing a reasonable sequence of C_V intervals for the search. Specifically, we will need two sequences of values of $R(C_V,Z)$ at the sonar and target depths, plus a sequence of values of $R_H(C_V)$, for C_V values chosen to be as far apart as possible subject to some requirements for accuracy of the final interpolation. We also want to avoid skipping over pairs of solutions to Eq. (2-4) and thus missing them. The high-speed ray tracing software is designed both to choose reasonable C_V values and to evaluate the functions efficiently for those values of C_V.

3. PROPERTIES OF THE TABULATED FUNCTIONS

As we mentioned in the Introduction, the basic strategy of the high-speed ray tracing software is to do the time-consuming parts of the calculation in advance, store the results in a big table, and look them up and interpolate to trace rays. The discussion in the previous section identified the functions we need to store. They are:

(1) $R(C_V,Z)$ [Eq. (2-8)]

(2) $$D(C_V,Z) = \frac{\partial R(C_V,Z)}{\partial C_V}$$

(3) $T(C_V,Z)$ [Eq. (2-10)]

(4) $R_H(C_V)$ [Eq. (2-9)]

(5) $$D_H(C_V) = \frac{dR_H(C_V)}{dC_V}$$

(6) $T_H(C_V)$ [Eq. (2-11)] .

In this section, we examine the properties of these functions. In particular, we focus on the "range" functions $R(C_V,Z)$ and $R_H(C_V)$; the properties of the "range derivatives" $D(C_V,Z)$ and $D_H(C_V)$ can be obtained from R and R_H by differentiation, and the properties of the "time" functions $T(C_V,Z)$ and $T_H(C_V)$ are very similar to the properties of the range functions.

Figure 3 shows a two-dimensional plot of the surface defined by the range function $R(C_V,Z)$. The dark curve labeled "SSP" near the left edge of the figure is the sound speed profile $C(Z)$ from which the function was calculated. This is the same example SSP used for the ray plots in Fig. 1. The part of the figure to the left of this curve has no physical significance because $C_V < C(Z)$ there [see Eq. (2-1)]. The prominent ridge extending downward from the local SSP maximum actually extends to infinite range; it was truncated arbitrarily for the plot. Our goal is to find a sufficiently accurate tabular representation of this surface, in a form suitable for rapid access. As the figure shows, this is not an easy task, because the function contains many difficult features (infinities, infinite slopes, and discontinuities in slope) where a fine grid will be necessary for accurate interpolation, and also large featureless areas where a fine grid would be wasteful. The key to this problem is that we know exactly where the difficult features lie, and we know their approximate form, so it is relatively easy to design a suitable tabulation grid for each new SSP.

Figure 4 shows in more detail the range $R(C_V,Z)$ as a function of depth Z, for four selected values of the vertex velocity C_V. The same curves are darkened for reference in Fig. 3. Similarly, the solid curves in Fig. 5 show $R(C_V,Z)$ as a function of C_V, for three selected values of Z. These curves, too, are darkened and labeled in Fig. 3. The dashed curves in Fig. 5 will be discussed later.

The relation between the range function plotted in Fig. 4 and the rays shown in Fig. 1 is very simple: If Fig. 4 is rotated 90° clockwise, then each curve is a segment of a ray like those in Fig. 1. Each ray segment starts at the minimum depth for that ray (i.e., at an upper vertex or surface intersection) and extends to the maximum depth for that ray (i.e., the next lower vertex or bottom intersection) exactly like segment BE of Fig. 2. The union of all such ray segments forms the surface shown in Fig. 3. Similar surfaces for $D(C_V,Z)$ and $T(C_V,Z)$ are defined in the same domain.

Several features of Figs. 3-5 are especially noteworthy.

(a) The asterisks in Figs. 4 and 5 mark places where the rays reach a vertex (become horizontal), and hence the range curves have infinite slope. (The range itself is finite.) The slope of the range curves is infinite whenever $C_V = C(Z)$, because $\theta = 0$ there [see Eq. (2-1)]. Because of this property, very fine grids (or special techniques) are necessary for accurate interpolation in regions of the C_V-Z plane near the SSP. Note that nothing fundamentally difficult is happening there. The problem is that we need range as a function of depth, and not vice versa. At an innocuous point where a plot of depth vs. range has zero slope, the corresponding plot of range vs. depth has infinite slope.

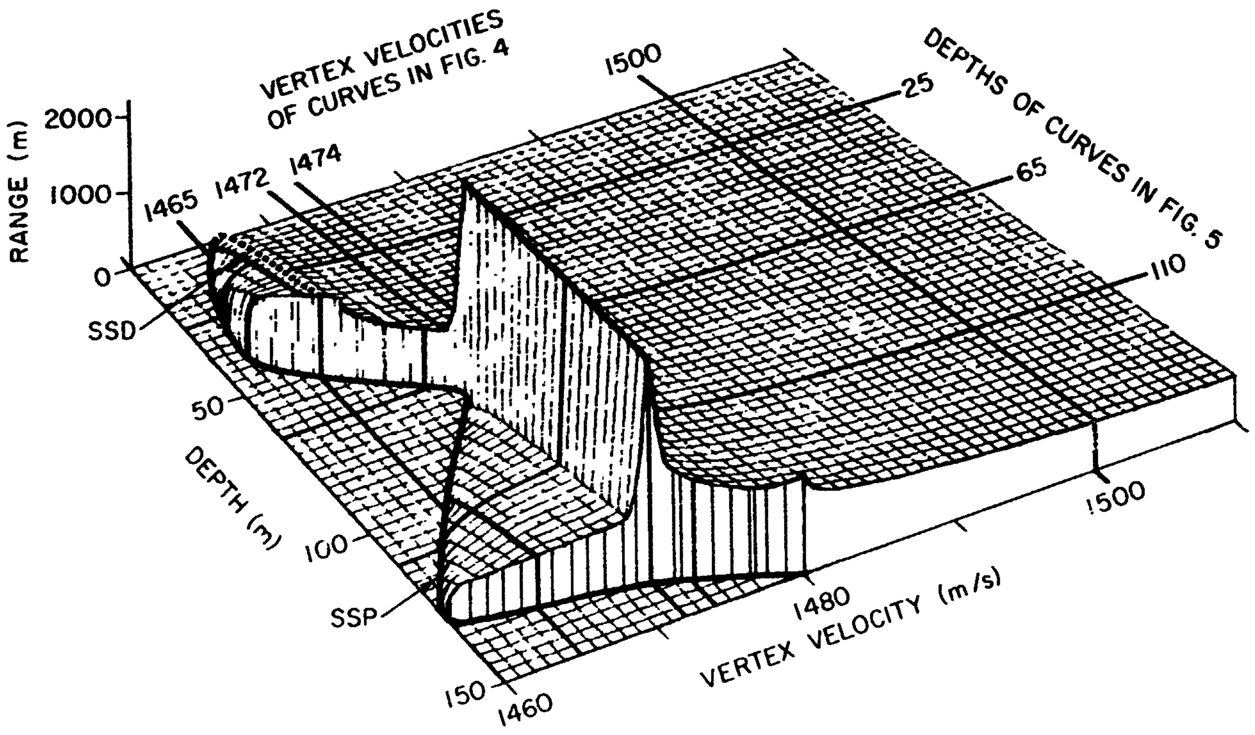

Figure 3. Horizontal range $R(C_V, Z)$ from minimum depth. The dark lines labeled at upper left and upper right indicate the positions of the curves in Figs. 4 and 5. The dark curve labeled SSP is the sound speed profile; parts of the plot to the left of the SSP have no physical significance. The dashed curve labeled SSD is the surface slope discontinuity, which marks the transition between rays that reach an upper vertex and rays that reflect from the surface.

(b) The local maximum in the sound speed (at $Z = 70$ m, $C(Z) = 1472$ m/s) divides the ocean into two sound channels. Any ray with a sufficiently small initial elevation (such that $C_V < 1472$ m/s) will stay in the channel in which it started. Therefore, Fig. 4 shows two curves for $C_V = 1465$ m/s: one for rays that start in the upper channel, and one for rays that start in the lower channel.

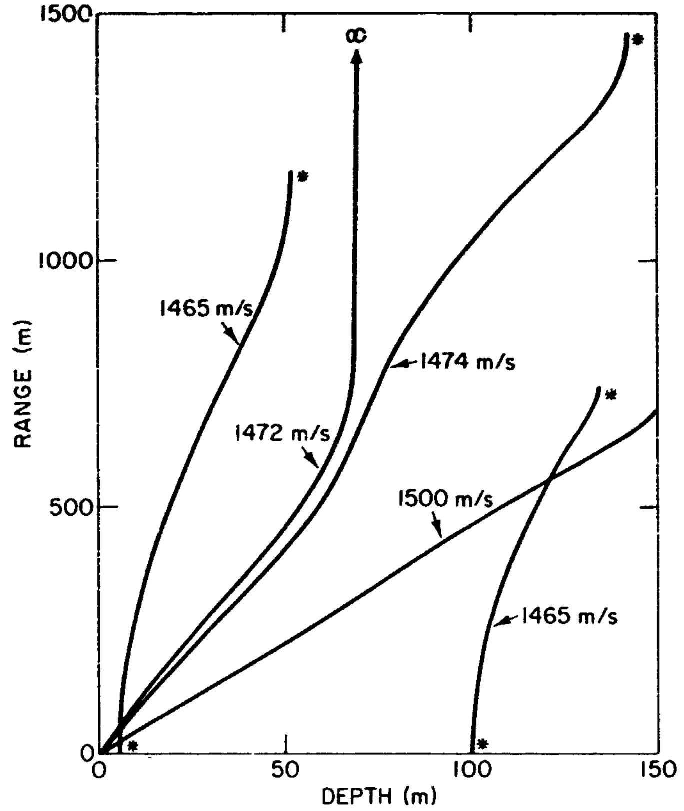

Figure 4. *Range $R(C_V, Z)$ as a function of depth Z for four values of vertex velocity C_V.*

(c) The ray at C_V = 1472 m/s (the "limiting ray") is a special case, in that C_V falls on a maximum in the SSP. That ray approaches a vertex asymptotically, but never quite reaches it; hence, the range function approaches infinity as it nears the SSP maximum. The portion of this ray lying deeper than the SSP maximum cannot be represented in this scheme at all; because the ray never reaches an upper vertex (but only approaches one), the range from the vertex is infinite. This infinity is visible in Fig. 3 as the prominent ridge (truncated arbitrarily) extending downward from the SSP maximum, in Fig. 4 at the right end of the ray at C_V = 1472 m/s, and in Fig. 5 in the curve at Z = 110 m.

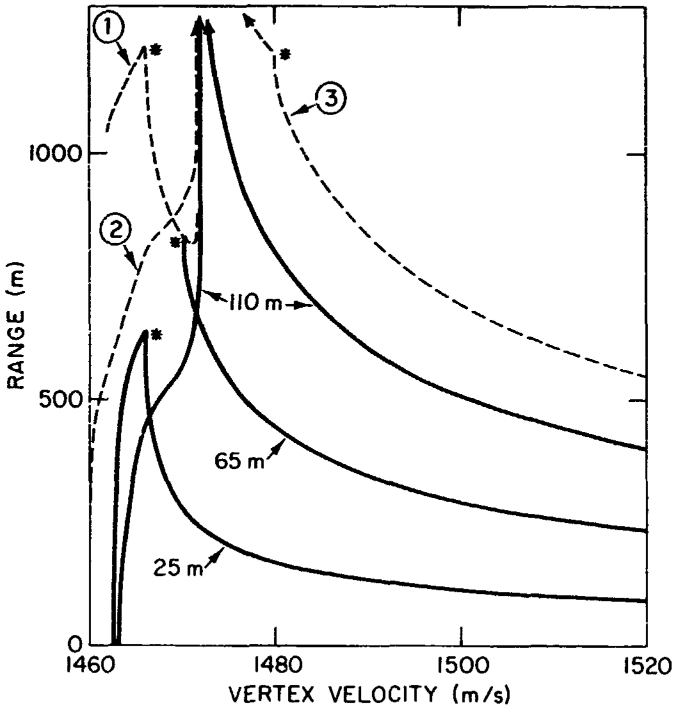

Figure 5. The solid curves are the range $R(C_V,Z)$ for three values of depth Z. The dashed curves show the half-wave range $R_H(C_V)$. For $C_V < 1472$ m/s, R_H is double-valued; curve 1 is used in the upper sound channel, and curve 2 in the lower.

(d) The range function has a discontinuity in slope in the upper sound channel at $C_V = C(0) = 1465$ m/s, where the SSP meets the surface. This "surface slope discontinuity" (SSD) is marked by a dashed line labeled SSD in Fig. 3. The SSD can be seen in the 25-meter curve in Fig. 5. The slope of this curve approaches a positive, finite value as C_V approaches the SSD from the left, and jumps to a negative infinite value on the right side of the SSD. The SSD marks the transition between rays that reach an upper vertex and rays that reflect from the surface.

(e) The nasty behavior of the range surface near the features mentioned above is in marked contrast to the smooth behavior at slightly higher C_V values. The ray at $C_V = 1500$ m/s in Fig. 4 is very close to a straight line, even though the difference C_V-C(Z) varies by a factor of two for that ray [see Eq. (2-8)]. Thus the range surface is difficult to interpolate over a narrow portion of the C_V-Z plane near the SSP, but it is essentially featureless elsewhere.

The dashed curves in Fig. 5 show the half-wave range $R_H(C_V)$ [Eq. (2-9)]. For values of C_V less than 1472 m/s (the local sound speed maximum), $R_H(C_V)$ is double valued; curve 1 is used in the upper sound channel, and curve 2 in the lower. For larger values of C_V (larger elevations), curve 3 is used at all depths. The value of $R_H(C_V)$ is the maximum value attained by the range $R(C_V,Z)$ for depths Z within the corresponding channel.

The following features of the R_H curves are noteworthy:

(a) Curves 1 and 2 start on the left at finite values, with finite slopes. This is in contrast to the range curves, which start with infinite slopes.

(b) Curves 1 and 3 have slope discontinuities, with infinite slopes on the right, at C_V values corresponding to the intersections of the SSP with the surface and bottom. Note that the "bottom slope discontinuity" is not seen in the $R(C_V,Z)$ curves.

(c) All three R_H curves are infinite at the C_V value corresponding to the maximum in the SSP. In contrast, the range curves are infinite there only for depths below (deeper than) the SSP maximum.

4. DATA STRUCTURE

We have shown in the previous section that the functions we want to tabulate are well-behaved over most of their range, but that they can get very wild indeed for values of C_V comparable to the sound speed. Obviously, we cannot simply tabulate them on a uniform grid; that would result in either an enormous waste of memory or inaccurate interpolation. Instead, the tessellation* scheme (i.e., the choice of grid points for the tabulation) must be tailored to the known properties of the tabulated functions in each of the various regions of the C_V-Z plane. In this section we describe the data structure that forms the basis for such an adaptive tessellation method.

4.1 A Sample Tessellation

The easiest way to understand the data structure is to start with an example of the end product. Figure 6 shows a tessellation of the same example SSP that we used in the previous figures. The curve shown is the SSP. The tabulated functions are defined in that part of the outermost rectangle that lies to the right of the SSP [see Eq. (2.1)]. Each of the innermost rectangles shown is called a "leaf" for reasons discussed later. Each leaf is subdivided into smaller rectangles called "regions." The numbers shown in each leaf are the number of subdivisions in the C_V and Z directions, respectively. For example, the largest leaf, marked "10 x 15," contains 150 identical regions arranged in 10 columns and 15 rows. The functions R, D, and T are tabulated at the corners of the regions; hence, the largest leaf contains 16 x 11 = 176 points on which the functions are tabulated. Function values within any region are found by bilinear interpolation (Ref. 6, Eq. 25.2.66) based on the four function values at the corners of the region. A comparison of Figs. 6 and 3 shows that the tabulated grid is dense where the range function is changing rapidly and sparse where the range function is smooth. These are the properties required to combine accurate interpolation with reasonable memory size.

The leaves shown in Fig. 6 are related to one another in two different ways. First, each leaf has a set of nearest neighbors (which may be empty) on each of its four sides. Second, each leaf is a subdivision of a larger rectangle, which in turn may be a subdivision of a still larger rectangle, and so on up to the largest rectangle defining the boundaries of the plane. Both types of relationships between leaves are useful, and both are reflected in the data structure.

*To tessellate is "to lay out, inlay, or pave in a mosaic pattern of small square blocks." Mathematicians have generalized it to include covering surfaces with complicated shapes.

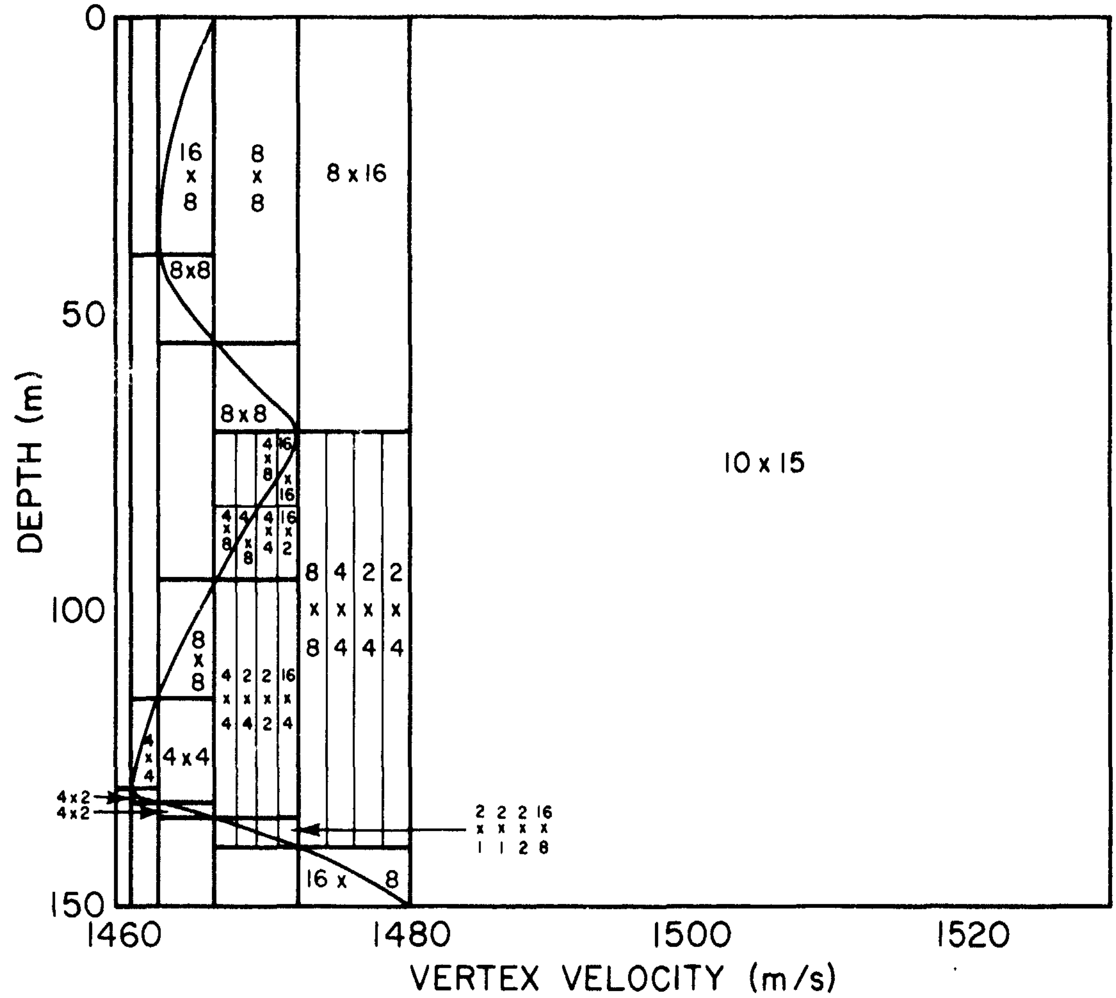

Figure 6. *A sample tessellation. Each rectangular box is the area of the C_V-Z plane represented by a single leaf in the tree-structured table. The numbers in the boxes are the numbers of subdivisions (regions) in the C_V and Z directions, respectively. The curve is the SSP.*

4.2 Tree Structure

The table used for interpolation is organized in a tree structure (Ref. 7, Section 2.3) in which each node of the tree represents one of the rectangles in Fig. 6. We will introduce the structure of the table by outlining the method used to create it. This method is illustrated in Fig. 7, using the tessellation of Fig. 6 as an example. The outermost rectangle--the entire portion of the C_V-Z plane on which the functions are to be tabulated--is represented by one node of the tree, which we call the "root" (level 0 in the figure). This node is described in the computer's memory by a collection of data specifying the rectangle's boundaries and other information.

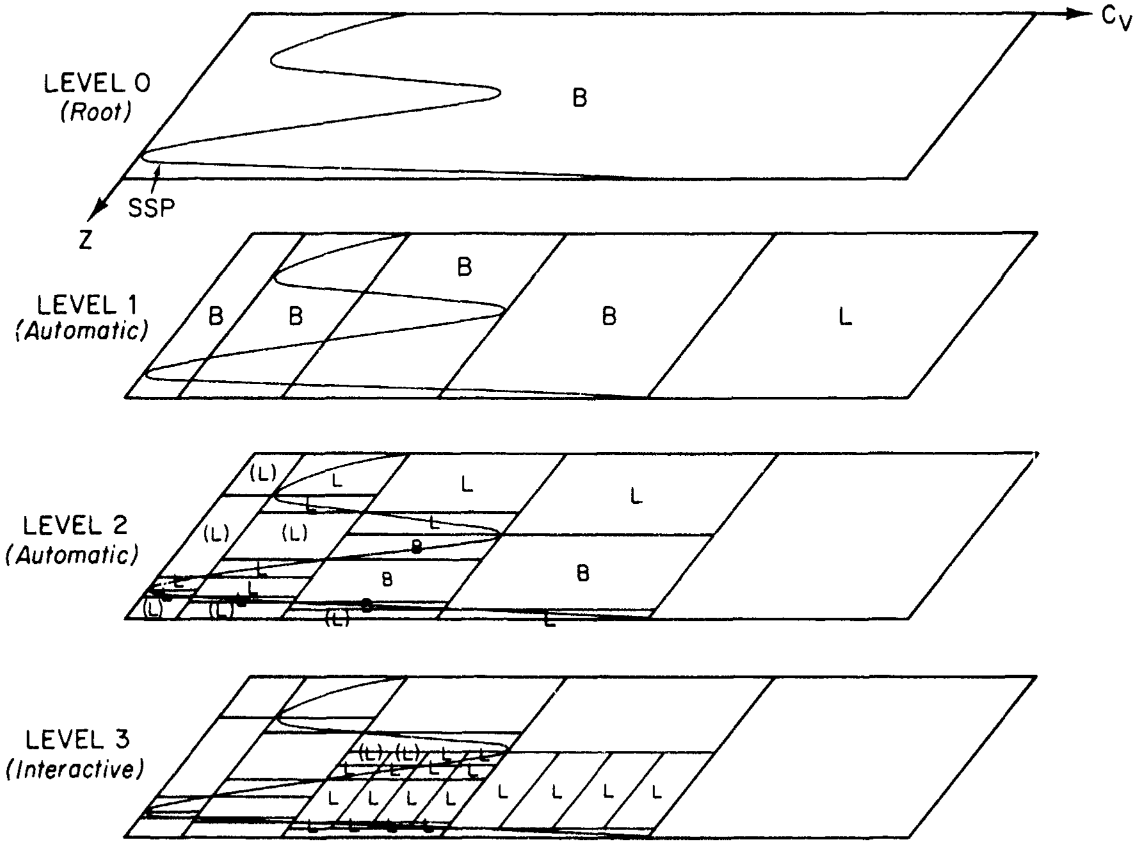

Figure 7. Creation of the tree-structured tables. Areas of the C_V-Z plane are represented by nodes at various levels. These nodes are indicated by the labeled rectangles. B indicates a branch node, and L indicates a leaf. (L) indicates a leaf lying entirely to the left of the SSP, where the tabulated functions are undefined. Empty rectangles indicate areas covered by leaves at higher levels.

The root is divided into several smaller rectangles (level 1) with boundaries at the C_V values corresponding to local extrema in the sound speed and to the intersections of the SSP with the surface and bottom. Each of these rectangles is represented by another node of the tree. Links (pointers) to the level 1 nodes are placed in the root node, and a link back to the root is placed in each level 1 node. (Because the links go both ways, the tree is "doubly linked"). The level 1 nodes are "offspring" of the root, and the root is the "parent" of the level 1 nodes. Nodes which have offspring are "branch nodes," and are marked with "B" in the figure. Nodes without offspring are "leaves," and are marked with "L" in the figure.

The process of subdivision continues on the next level. Boundaries of the level 2 nodes are placed at those depths (Z values) where the SSP touches a boundary of a level 1 node. New nodes are created as before, with links to and from the corresponding level 1 parent. By subdividing the first two levels in this fashion, we guarantee that all difficult features of the $R(C_V,Z)$ surface will lie on boundaries of a region, not in its interior.

Nodes through level 2 are created automatically. Eventually the whole tree should be built automatically, but in the present version of the software the rest of the decisions must be made by the user and entered via a terminal. The user may choose to extend the tree by subdividing any of the level 2 nodes; this was done four times in the example. This subdivision may be repeated on as many levels as necessary. The user also controls the subdivision of each leaf into an N-by-M array of identical rectangular regions. The functions $R(C_VZ)$, $D(C_V,Z)$, and $T(C_V,Z)$ are tabulated at the corners of the regions; therefore, each leaf contains 3 x (N+1) x (M+1) tabulated function values. The user's decisions regarding the subdivision of nodes into new nodes, or of leaves into regions, are guided by the results of tests done by the software to estimate the accuracy of the four-point bilinear interpolation used to find the function values in each region.

4.3 Nearest-Neighbor Links

Each leaf also contains four additional sets of links which point to the neighboring leaves on each of the four sides. These nearest-neighbor links permit the software to find a series of function values along any line in the C_V-Z plane without traversing the tree back toward the root and out the next branch at each leaf border. The resulting increase in processing speed is especially important in searches for eigenrays, for which a series of values along a line at constant depth is required (see Section 2.7). The use of these links is described more fully in Section 5.3.

4.4 Other Data

In addition to the function values, node and region boundaries in C_V and Z, and the pointers just described, the tree contains various flags and codes giving properties of each node and each region. For example, there is one flag for each region that indicates whether or not the upper boundary of the region is at the surface of the water. The details of this auxiliary information, and the arrangement of the data structure in memory, are given in the program documentation.

5. SOFTWARE SUMMARY

5.1 Interpolating the SSP (FITVEL)

Figure 8 shows the flow of information through the high-speed ray tracing software system. The initial input is a card-image file prepared by the user which contains a series of depth values and the corresponding values of the sound speed. This file is used as input to program FITVEL, which computes the parameters of a smooth function that goes through all of the given sound-speed data. FITVEL is from NISSM II (Ref. 3). The form of the interpolating function is chosen to make the integrals in Eqs. (2-8) and (2-10) relatively easy to compute. The parameters of the interpolating function are written onto a disk file, the SSP Parameter File.

5.2 Creating the Tables (TESSEL)

Program TESSEL creates and fills the tree-structured tables described in Section 4. TESSEL has two inputs: the SSP Parameter File and a set of commands which may come directly from the user's terminal or from a file prepared by the user. The commands direct TESSEL to perform the following operations:

- Initialize the data structure and perform level 1 and level 2 subdivision. The result is a tree with a few large, one-region leaves whose boundaries are determined by the extrema in the SSP. This automatically places the most difficult regions at the edges or corners of the leaves.

- Divide a leaf into a specified number of regions. Function values are computed at the corners of the regions using NISSM II methods (Ref. 3).

- Change a leaf to a branch node. Each of the leaf's regions becomes a new leaf.

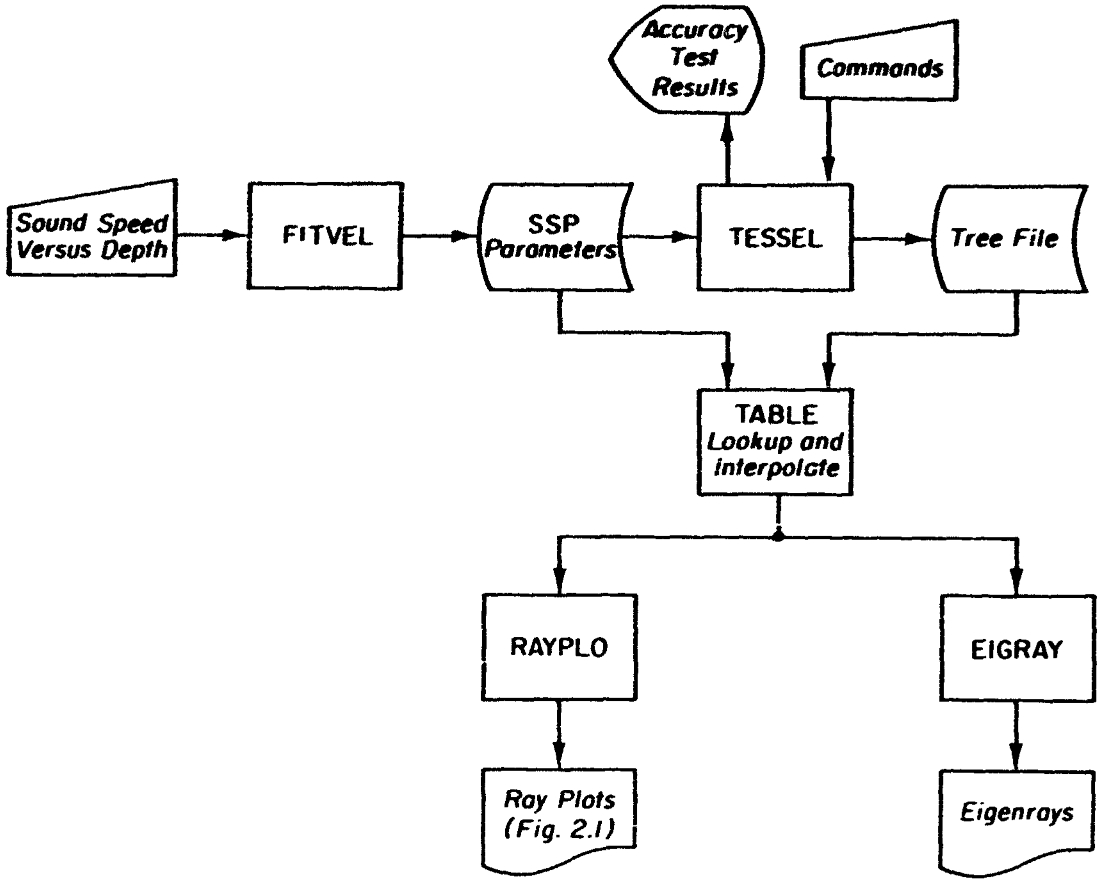

Figure 8. Data flow for the high-speed ray tracing system.

- Change a branch node to a leaf. Lower-level leaves are deleted.

- Test the accuracy of the interpolation in a given leaf. Many test points are scattered at randomly-chosen locations in each region of the leaf, and $R(C_V,Z)$ is computed at each test point using two methods: by interpolation using the tabulated function values, and by direct integration. The difference between the two results is divided by the maximum permissible error, which was arbitrarily chosen to give a maximum angle error of 0.1° and a maximum range error of 1.0 meter. The square root of the mean or squares of these normalized errors is a number which should be less than 1.0 if the region passes the test. If the number is much less than 1.0, the region is too small and is wasting memory.

- Store the tree on the disk, in the Tree File.
- Read a previous Tree File, so that it can be examined or modified. A store-read sequence also renumbers the nodes and eliminates garbage left from previous changes.
- Print various information about the tree.

5.3 Using the Tables (TABLE, RAYPLO, EIGRAY)

At this writing, there are two programs that use the tree-structured tables: RAYPLO, which plots rays as in Fig 1, and EIGRAY, which computes eigenrays and prints the results. RAYPLO uses a series of interpolated function values along a line in the (C_V,Z) plane at constant vertex velocity to plot a ray. EIGRAY uses a series of interpolated function values along lines at constant depth to search for values of the vertex velocity for which the eigenray equation (2-4) is satisfied. EIGRAY's algorithm is outlined in Section 2.7.

Both RAYPLO and EIGRAY use a common set of interface subroutines to read the Tree File and to do the table lookup and interpolation. The primary interface subroutine, TABLE, operates in three modes: "point" mode, "line" mode, and "line-continuation" mode. In the point mode, TABLE simply returns values of the functions $R(C_V,Z)$, $D(C_V,Z)$ and $T(C_V,Z)$ for a single specified point in the (C_V,Z) plane. In the line mode, the calling routine specifies both a starting point and a direction, and TABLE returns arrays of interpolated function values at a series of sample points along the specified line in the (C_V,Z) plane. The line-continuation mode is just like ine mode but the line continues where the previous line left off.

TABLE's line and line-continuation modes are designed to optimize eigenray searches and similar applications. TABLE chooses as sample points the intersections of the specified line with the region boundaries. This choice guarantees that the sample points are reasonable--i.e., close enough together to allow accurate interpolation but not so close that they are wasteful--because these are the very criteria that are used to choose the region sizes when the tree is created. Moreover, this reasonableness is achieved quickly, because TABLE uses the nearest-neighbor links (Section 4.3) to go directly from one leaf to the next; without the nearest-neighbor links, TABLE would have to traverse the tree back up one branch, often through several levels, and out the next branch wherever the line crosses a leaf boundary. Thus, as we mentioned in Section 2.7, both criteria for efficient eigenray computation are satisfied: Reasonable sample points are chosen and processing time per point is kept low.

6. STATUS

All of the computer programs mentioned in the previous section, plus test and utility routines, are written and working, and the program documentation is written. We have run tests using several sound speed profiles, including the artificial SSP that we have been using as our example in this report. Using this SSP, and the tessellation shown in Fig. 6, we achieved the results discussed below.

6.1 Table Size

The tree-structured table for our example requires 66 kilobytes of memory when the functions R, D, and T and the C_V and Z breakpoints are all stored in double precision (8 bytes). Of this, 74% of the memory is for the functions R, D, and T themselves. If single precision were used, this would be reduced to 40 kilobytes, of which 61% would be for the function values. The functions are tabulated at about 2000 points in this example. Most of the measured sound speed profiles used in sonar simulations are simpler than this one, so the corresponding tables probably would be smaller.

6.2 Accuracy

Our sample tessellation passes the standard test described in Section 5.2 (angle error less than 0.1°, range error less than 1.0 m) for every leaf except one; for that one, the test parameter is 1.09, where 1.0 or less is a passing grade. (The exception is the small 16 x 16 leaf shown just below the SSP maximum in Fig. 6.) To verify the adequacy of this standard test, we computed eigenrays for a variety of depths and for ranges up to 3000 meters, and compared the results with eigenrays computed using NISSM II techniques directly. Any differences are attributable to interpolation errors. For almost all eigenrays, the angle errors were less than 0.2°, the errors in propagation time were less than 0.2 ms, and the errors in the spreading loss were less than 0.1 dB. The few eigenrays that were less accurate had vertex velocity values near the SSP maximum. The problems there are well understood and fixable; they are caused by the fact that the half-wave range $R_H(C_V)$ [Eq. (2-9)] is not tabulated separately, but is taken from the table for $R(C_V,Z)$ using (2-9). Separate tabulation of $R_H(C_V)$, $D_H(C_V)$, and $T_H(C_V)$ would require little more memory, and would improve both speed and accuracy.

7. DIRECTION OF FUTURE WORK

The high-speed ray tracing software, in its present form, is a research tool designed to test the concept. We have shown that it works, but we have not shown that its speed lives up to our expectations. The work that remains to be done falls into four major categories, which overlap quite a lot: evaluating the performance of the system in comparison with other options, trying new ideas to improve the performance, making the program easier to use, and installing the software in an application program such as REVGEN. The specific tasks are:

(a) Tabulate the half-wave functions $R_H(C_V)$, $D_H(C_V)$, and $T_H(C_V)$ separately. This would improve both accuracy and speed.

(b) Improve the algorithm for interpolating the tabulated functions in the difficult regions near the sound speed profile. In the present version, bilinear interpolation is used everywhere. If one or more corners of a region fall to the left of the SSP, the function values for those points are extrapolated as the tree is created in such a way that the usual bilinear interpolation will yield the correct answers at some point on the SSP. Several options for improving this scheme are being considered.

One option is based on a class of mathematical transformations that we call "regularizing transformations." Instead of tabulating the functions R, D, and T directly, we tabulate three different functions--$\hat{R}(C_V,Z)$, $\hat{D}(C_V,Z)$, and $\hat{T}(C_V,Z)$--which are related to the original functions by transformations of the form

$$\hat{R}(C_V,Z) = F[C_V,Z,\ R(C_V,Z)] \qquad (7\text{-}1)$$

and so on, where F is a simple, invertible function chosen such that the transformed function $\hat{R}(C_V,Z)$ is relatively smooth, and therefore can be interpolated accurately on a coarse grid. After interpolation, the inverse transformation F^{-1} is used to recover the original function R. Similar transformations would be used for D and T.

We have investigated several sets of regularizing transformations. All of them have serious problems: Either they fail to eliminate some of the difficult features, or they are so complicated that much of the speed advantage of the table lookup technique is lost.

A second option is to use a higher-order interpolation technique everywhere. For example, we could make use of the partial derivatives $\partial R/\partial C_V$ [i.e., $D(C_V,Z)$] and $\partial R/\partial Z$ [from (2-8)] and do bicubic interpolation, as in Ref. 8. Special methods. or extrapolation, still would be necessary for regions bordering the sound speed profile.

A third option is to continue to use bilinear interpolation wherever it is accurate enough, and switch to a more accurate technique in those few regions where it is needed. One bit in the table for each region would indicate whether or not linear interpolation was adequate for that region. If not, then all or part of the integrals in Eqs. (2-8) and (2-10), and the corresponding one for D, would be done explicitly, using NISSM II techniques (Ref. 3) or some other method. The fact that direct integration is slow is unimportant because it would be used for only a small fraction of the calculations. This option has the advantage that the compromise between memory and speed can be made explicit. If the available memory is too small to allow bilinear interpolation to be used everywhere, the calculation will be slightly slower, but accuracy need not suffer. At this writing, this appears to be the most promising option.

(c) Improve the interactive input and output in TESSEL. The present version has too many commands and inconvenient formats.

(d) Design and install an automatic tessellation algorithm. The present interactive method uses a large amount of human time to create the tree. This is acceptable in a developmental version, but in an operational version the software should do most or all of this task.

(e) Optimize the code for speed. We must measure the time required for one or more test cases, find out which parts of the code use most of the time, make changes designed to speed up those parts, and repeat the tests.

(f) Install the code in REVGEN (Refs. 9 and 10) and perhaps in other simulation programs. The most difficult part of this task is the development of an efficient buffering algorithm to make sure that all of the echoes from each point scatterer are delivered to the signal processing subroutines when they are needed, i.e., in the order in which the echoes are received. Considerable progress has been made on this task, but a discussion of the techniques is outside the scope of this report.

In conclusion, we have shown that high-speed ray tracing using tree-structured tables will work, and that the table size is reasonable for a medium-size minicomputer. Most of the major system-design problems have been solved. The task of transforming the software from a proof-of-concept vehicle to a useful simulation tool will require several more months.

8. REFERENCES

1. P. C. Etter and R. S. Flum, "A Survey of Underwater Acoustic Models and Environmental-Acoustic Data Banks," ASWSPO Report ASWR-80-115, Anti-Submarine Warfare Systems Project Office, Dept. of the Navy, Washington, D. C. 20362, September 1980.

2. R. J. Urick, *Principles of Underwater Sound,* second edition (McGraw-Hill, 1975), Section 5.6.

3. H. Weinberg, "Naval Interim Surface Ship Model (NISSM II)," NUSC Technical Report 4527, Naval Underwater Systems Center, New London, CT, 1973.

4. M. A. Pedersen, D. White, and D. W. Johnson, "Generalized expansions of ray theory in terms of phase velocity. I," J. Acoust. Soc. Am. 58, 78-96 (1975).

5. J. Mathews and R. L. Walker, *Mathematical Methods of Physics* (Benjamin, 1965), p. 338.

6. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions* (Dover, 1965).

7. D. E. Knuth, *The Art Of Computer Programming,* Vol. 1: *Fundamental Algorithms,* second edition (Addison-Wesley, 1973).

8. H. Akima, "Algorithm 474, Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures," Communications of the ACM 17, 1974, pp. 18-20 and 26-31.

9. D. W. Princehouse, "REVGEN, a Real-Time Reverberation Generator, Concept Development," APL-UW 7511, Applied Physics Laboratory, University of Washington, Seattle, WA, 1975.

10. D. W. Princehouse, "Reverberation Generator Ocean Algorithm, A Status Report," APL-UW 7806, Applied Physics Laboratory, University of Washington, Seattle, WA, 1978.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER	2. GOVT ACCESSION NO.	3 RECIPIENT'S CATALOG NUMBER

4 TITLE (and Subtitle)

HIGH-SPEED RAY TRACING FOR UNDERWATER SOUND, A STATUS REPORT

5. TYPE OF REPORT & PERIOD COVERED

Status Report
1978 to 1980

6 PERFORMING ORG. REPORT NUMBER

APL-UW 8109

7. AUTHOR(s)

Robert P. Goddard
David W. Princehouse

8 CONTRACT OR GRANT NUMBER(s)

N00024-78-C-6018
N00024-81-C-6042

9 PERFORMING ORGANIZATION NAME AND ADDRESS

Applied Physics Laboratory
University of Washington
1013 N.E. 40th St., Seattle, WA 98105

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11 CONTROLLING OFFICE NAME AND ADDRESS

Naval Sea Systems Command
Department of the Navy
Washington, D.C. 20362

12. REPORT DATE

October 1981

13 NUMBER OF PAGES

27

14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15 SECURITY CLASS. (of this report)

Unclassified

15a DECLASSIFICATION/DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer algorithms
Underwater sound
Ray tracing
Simulations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Computer simulations of underwater sound are often limited by the computer time required to trace the many paths (rays) along which sound may propagate. We have developed a high-speed ray tracing technique that is applicable to situations in which the sound speed depends on depth only, and not on horizontal position or time. Included in the model are specular reflections

(cont.)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20., cont.

from a flat surface and a flat bottom, and ray bending in a vertical plane. Diffractive effects, such as propagation through caustics and leakage from ducts, are not modeled at this time.

The technique is fast because the time-consuming parts of the calculations are done before the simulation starts, and the results are stored in tables. During the simulation, these results are retrieved from the tables and interpolated. The stored data consist of three functions of two variables, tabulated on a nonuniform two-dimensional grid. A doubly linked tree structure with extra links to neighboring leaves is used to permit rapid access to neighboring points in the grid.

Software exists to create and fill the tables and to use the tables to plot rays and find eigenrays. Good accuracy, in most cases, can be achieved using tables of less than 100 kilobytes. Modifications are planned that will reduce the table size, improve reliability, and make the software easier to use. Our goal is to include ray tracing in simulators such as REVGEN and the NOSC Hybrid Simulator that are now limited to direct straight-line propagation.