Ray Tracing for Ocean Acoustic Tomography

by Brian D. Dushaw and John A. Colosi

Technical Memorandum
APL-UW TM 3-98
December 1998

Applied Physics Laboratory University of Washington
1013 NE 40th Street Seattle, Washington 98105-6698

DARPA Grant MDA 972-93-1-003
ONR Grant N00014-97-1-0259

ACKNOWLEDGEMENTS

This work is conducted as part of the Acoustic Thermometry of Ocean Climate (ATOC) and North Pacific Acoustic Laboratory (NPAL) projects supported by DARPA (Grant MDA 972-93-1-0003) and ONR (Grant N00014-97-1-0259), respectively. John Colosi is grateful for a Young Investigator Award from the Office of Naval Research. Matthew Dzieciuch made several constructive comments on this technical report and on the development of the code. Bruce Cornuelle urged the inclusion of the effects of ocean currents in the ray calculations. Bob Odom provided most of the discussion of finite-frequency effects on rays passing near the ocean surface.

Questions concerning this report and its associated FORTRAN code may be addressed to:

Brian Dushaw
Applied Physics Laboratory
College of Ocean and Fisheries Sciences
University of Washington
1013 N.E. 40th Street
Seattle, WA 98105-6698
(206) 543-1300
dushaw@apl.washington.edu

John Colosi
Applied Ocean Physics and Engineering Department
Woods Hole Oceanographic Institution
MS #11
Woods Hole, MA 02543
(508) 289-2317
jcolosi@whoi.edu

ABSTRACT

This report describes a new, flexible computer code in the FORTRAN computer language to make ray calculations for ocean acoustic tomography. The *Numerical Recipes* software package provided the basis for much of this computer code. The ray equations are reviewed, and ray equations that include the effects of ocean current are derived. Methods are derived for rapidly integrating those equations to obtain time front and eigenray information for long-range, deep-water acoustic transmissions. These methods include a look-up table for sound speed, sound speed gradient, second derivative of sound speed, and range-dependent information. Cubic spline methods are used to interpolate sound speed with depth and to obtain the derivatives of sound speed. The choice of the step size increments used to integrate the equations is a critical aspect of the integration, affecting both the accuracy of the prediction and the speed of computation. A predetermined, user-specified step size appears to allow more efficient calculations than "adaptive step" methods. "Adaptive step" methods adjust the step size automatically to maintain a given accuracy in the integration of the ray equations, while user-specified step sizes allow one to use prior knowledge of the integration problem to achieve the desired accuracy with much less computational overhead. Several integration methods were explored, but the classical 4th order Runge-Kutta method appears to be the most efficient and best method for this integration problem. Appendices describe detailed aspects of the computer code, as well as the methods used for deriving eigenray information and for parallelizing the ray calculations. The computer code is designed to be unstable so that the user can easily modify it to his or her own porpoises.

TABLE OF CONTENTS

LIST OF FIGURES

Page

This is a blank page.

MOTIVATION

Although numerous ray tracing codes are available, none satisfy all of the present requirements of long-range ocean acoustic tomography. Ocean acoustic tomography is described by Munk, Worcester, and Wunsch (1995). What is required for long-range tomography is a fast, accurate, and flexible code. The requirement of flexibility necessitates coding that can be easily modified by the user; thus FORTRAN is the preferred computer language. Flexibility allows the user to easily implement his or her own algorithms, such as a better search for eigenrays or an alternative (perhaps faster or more accurate) integration routine. Issues of numerical accuracy in ray predictions are discussed in the section on integrating the ray equations. Finally, the requirement of speed necessitates coding that can sometimes become opaque. This report therefore also describes the methods employed to achieve computer code that is highly efficient.

The present and immediate goal of the authors of this report is to achieve fast, accurate wavefront and eigenray travel time predictions at basin scale ranges (3–5 Mm) in the North Pacific Ocean as part of the North Pacific Acoustic Laboratory and Acoustic Thermometry of Ocean Climate projects (Dushaw 1999; Dushaw et al. 1999; Worcester et al. 1998, Colosi et al. 1999). These projects require eigenray predictions for a half dozen or so time series of sound speed sections derived from acoustic data obtained at 3–5 Mm range. For a single sound speed section, calculation of the time series of ray travel times takes about 30 hours on a 200-MHz Pentium Pro computer, so the need for the fastest possible code is evident.

Many of the ideas used in implementing the code to be discussed here originated in the RAY code and associated technical report by Bowlin et al. (1992). Alas, this code is in C. The new computer code described here uses the FORTRAN cubic spline and integration routines from *Numerical Recipes* (Press et al. 1992; *Numerical Recipes* hereinafter); thus use of this code requires a license to use the *Numerical Recipes* software ($40).

CREDO: The code described here is meant to be easily modified by the user, and so it will never be a stable ray code. The code is meant to be transparent and fast. Metaphorically, if the Bowlin RAY code is a Fiat 2000 with a Fiat engine, the code described here is a '67 Chevy Impala with JATO (Jet Assisted Take-Off) propulsion.*

*Urban Legend No. 37

RAY EQUATIONS, OR EQUATIONS OF MOTION

According to Bowlin et al. (1992): *The equations of motion for a ray traveling through the ocean can be cast in Cartesian coordinates as follows:*

$$\frac{d\theta}{dr} = \frac{\partial_r c}{c}\tan\theta - \frac{\partial_z c}{c} \tag{1a}$$

$$\frac{dz}{dr} = \tan\theta \tag{1b}$$

$$\frac{dt}{dr} = \frac{sec\theta}{c} \tag{1c}$$

where θ is the angle of the ray with respect to the horizontal r *axis, and* z *is the vertical coordinate* [positive upward].

Bowlin et al. (1992) continue: *For long range ocean acoustics, the curvature of the Earth's surface makes non-Cartesian coordinates more suitable for ray tracing. Let new z' axes lie along radii passing through the center of the Earth with $z' = 0$ at sea level and $z' = R_e$ at the Earth's center, where R_e is the radius of the Earth, and let the new r' be the range measured along a circular arc at sea level. ..... The new equations of motion which include the geometrical effects due to a spherical Earth are*

$$\frac{d\theta}{dr'} = f_e\frac{\partial_{z'} c}{c} - \frac{\partial_{r'} c}{c}\tan\theta - \frac{1}{R_e} \tag{2a}$$

$$\frac{dz}{dr'} = f_e\tan\theta \tag{2b}$$

$$\frac{dt}{dr'} = \frac{f_e sec\theta}{c} \tag{2c}$$

where

$$f_e = \frac{dr}{dr'} = \frac{(R_e - z')}{R_e} \tag{3}$$

These are the equations that RAY integrates.... [These latter equations have z positive downward.] However, the RAY code actually integrates not θ, but $\cos\theta$ and $\sin\theta$, thus avoiding the calculation of the transcendental functions.

An alternate way to integrate these equations is to use the equations in Cartesian coordinates (the equations above with $R_e \to \infty$ and $f_e \to 1$) but apply the well-known Earth flattening transformation (Aki and Richards 1980). This transformation is applied once to the initial sound speeds and associated depths, and the subsequent integration in the Cartesian coordinates is then mathematically equivalent to the above equations. Physically, this transformation is a stretching of depth and sound speed equivalent to the curvature of the Earth's surface. If $\varepsilon = z/R_e$ (z positive downward) then the Earth flattening transformation is $z = z * (1 + \varepsilon/2 + \varepsilon * \varepsilon/3)$ and $c = c * (1 + \varepsilon + \varepsilon * \varepsilon)$. Since some computation can be saved if this transformation is applied once before integrating the differential equations, this transformation together with the Cartesian differential equations is preferred. Ray predictions using the two methods agree to within 1 ms at 3-Mm range, and the integration of the Cartesian equations appears to be roughly 10% faster.

If $cs = \cos\theta$, $sn = \sin\theta$ and we apply the flat Earth transformation to z and c, the equations above can be re-written as

$$\frac{d\,cs}{dr'} = -\,sn\left(\frac{\partial_{z'}c}{c} - \frac{\partial_{r'}c}{c}\,\frac{sn}{cs}\right) \tag{4a}$$

$$\frac{d\,sn}{dr'} = cs\left(\frac{\partial_{z'}c}{c} - \frac{\partial_{r'}c}{c}\,\frac{sn}{cs}\right) \tag{4b}$$

$$\frac{dz}{dr'} = \frac{sn}{cs} \tag{4c}$$

$$\frac{dt}{dr'} = \frac{1}{c\,cs} \tag{4d}$$

No trigonometric functions need to be calculated while integrating these equations. Since integration of these equations is most sensitive to the angle integration, the redundant equations for angle (the equations for *cs* and *sn*) are actually helpful for the stability of the integration. The above equations can be cast in only three equations, where the first equation is for $d(tan\theta)/dr$, but this formulation involves a term sqrt(1+tan θ*tan θ) in the integration for travel time (see Appendix B). This square root is computationally expensive (for one test case it increased computation time by about 15%), perhaps more expensive than integrating the additional differential equation. Similarly, it is important to code

these equations using the smallest number of divisions; a division is about as computationally expensive as taking a square root.

For the open-ocean environment of the North Pacific, the term involving dc/dr in the above equations appears to be 5–6 orders of magnitude less than the other terms. Ray predictions using Levitus sound speeds with and without the dc/dr term were insignificantly different. As Bowlin et al. (1992) comment, *if ... an environment where dropping the dc/dr term [in Eq. 4] produces a significant change in the wavefront then it is very likely that the linear range dependent model is inadequate for that environment even if the dc/dr term is included.* This term is therefore commented out in the code for efficiency purposes, but the user concerned with significant range dependence should consider checking if this term is important or not.

LOOK-UP TABLES AND SOUND SPEED INTERPOLATION

It is clear that a fast code will necessarily rely on look-up tables for sound speed, sound speed gradient, and other parameters. Depth is the obvious index to these look-up tables, since with this index the sound speed values can be quickly obtained at arbitrary depths during the integration. As pointed out by Bowlin et al. (1992), the rapid, accurate calculation of sound speed, sound speed gradient, etc., at arbitrary depth and range is a critical aspect of the ray prediction.

For the present code, we adopt cubic splines for interpolation of sound speed (*Numerical Recipes*) and for the calculation of sound speed gradients. Splines have been discussed in the context of calculating acoustic rays by Moler and Solomon (1970). Cubic splines are mathematically the most natural and smoothest interpolation; their goal in life is to minimize the second derivative of the interpolation (Parker 1994). The smoothest possible interpolation is essential because of the sensitivity of the ray predictions to sound speed gradient. In addition, cubic splines allow the sound speed gradient to be calculated quickly and accurately.

We note again that it is essential that smoothly varying sound speeds be used for ray predictions. It is therefore important for the user of any ray trace code to apply any desired smoothing or interpolation—horizontally or vertically—prior to the ray prediction. The code does not implement any smoothing of the user's sound speed profiles. Such smoothing necessarily requires allowing some misfit to the sound speed values, which can lead to biases in the sound speed profiles. Since the code described by Bowlin et al. (1992) applies some smoothing to the sound speed profiles, Bowlin et al. discuss this bias problem at length; it is perhaps a problem best left to the user of the code.

The cubic spline interpolation method is described by *Numerical Recipes* and Parker (1994), so only the details appropriate to the present application are discussed here. Typically, sound speed values for ray tracing are available at a small number of depths, 30–100 say. The cubic spline method is used to interpolate those values, as well as values for the first and second derivatives, to 3-m increments throughout the water

column, and these values are used to fill a table of sound speed values. (To properly handle surface reflected rays, the value for sound speed at the surface is repeated at 3 m above the ocean surface.) For arbitrary depth z, the sound speed can be calculated by

$$c = c_j + dz\left(\frac{dc}{dz}\right)_j + \tfrac{1}{2}\, dz^2\left(\frac{d^2c}{dz^2}\right)_j \tag{5}$$

where $c_j \equiv c(z_j)$, $dz = z - int(z)$, and $j = int(z)$ gives the index in the sound speed look-up table. While this is an approximation for sound speed, it is sufficiently accurate for ray tracing.

Sound speed gradient in the ray equations must be calculated carefully, since even small errors in this quantity result in ray predictions with poor quality. With cubic splines, the sound speed gradient at arbitrary depth z between the jth and $(j+1)$th depths is (*Numerical Recipes*)

$$\frac{dc}{dz} = \frac{c_{j+1} - c_j}{z_{j+1} - z_j} - \frac{3A^2 - 1}{6}\,(z_{j+1} - z_j)\,c''_j + \frac{3B^2 - 1}{6}\,(z_{j+1} - z_j)\,c''_{j+1} \tag{6}$$

where

$$A \equiv \frac{z_{j+1} - z}{z_{j+1} - z_j}, \quad B \equiv \frac{z - z_j}{z_{j+1} - z_j} = 1 - A \tag{7}$$

However, the difference between adjacent depths in the table is a constant Δz = 3 m, and many of the terms in this equation are constant and so may be calculated prior to the integration. When a 1-m depth increment is used, the equations simplify even further at the expense of a very large look-up table. Thus, the equation for the first derivative becomes (positive z is downward)

$$\frac{dc}{dz} = \frac{A^2 c''_j}{2\Delta z} - \frac{B^2 c''_{j+1}}{2\Delta z} - const._j \tag{8}$$

where one half the second derivative and $const._j$ are calculated ahead of time in the look-up table.

$$const._j = \frac{c_{j+1} - c_j}{\Delta z} + \frac{\Delta z(c''_j - c''_{j+1})}{6} \tag{9}$$

The vertical gradient of sound speed can be calculated rapidly and accurately in this way. Note that while the Taylor expansion approximation for $c(z)$ in Eq. 5 is adequate, the Taylor expansion approximation for dc/dz is not.

Range dependence is implemented by assuming a constant horizontal sound speed gradient and a constant horizontal gradient of the vertical gradient of sound speed. Thus,

$$c(r, z) = c(r_j, z) + (r - r_j)\,\frac{c(r_{j+1}, z) - c(r_j, z)}{r_{j+1} - r_j} \tag{10}$$

for an integration step at range r between the jth and $(j + 1)$th sound speed profiles. The equation for the horizontal dependence of sound speed vertical gradient is similar. All indications (Bowlin et al. 1992) are that these approximations for range dependence are adequate. It is essential to implement at least a constant sound speed gradient between sound speed profiles, however, because jumping from one profile to another in a discontinuous fashion produces a multitude of false caustics in the time front.

The sound speed look-up table containing the information required to calculate sound speed and sound speed gradient, including their range dependence, at arbitrary depth and range may be constructed using a three-dimensional matrix of depth, range, and six variables at those depths and ranges. We use 1835 depths at 3-m increments and the ranges given by the initial set of sound speed profiles. The six variables at those depths and ranges are sound speed, sound speed gradient, sound speed second derivative with depth, horizontal derivative of sound speed, horizontal derivative of sound speed gradient (assuming a constant gradient between profiles), and the constant term described above for calculating sound speed gradient at arbitrary depth. With these variables, sound speed and sound speed gradient can be calculated at arbitrary depth and range very efficiently. This double precision table, if filled, uses about 13 Mbytes of computer memory for 150 sound speed profiles.

INTEGRATION OF THE DIFFERENTIAL EQUATIONS

There are several well-known methods for integrating a coupled set of ordinary differential equations such as Eq. 4. One of the most common is the Runge-Kutta method. Applied to Eq. 4, this method involves a series of range steps. The derivatives of ray angle, depth, and travel time with respect to range are calculated at each step. When using any integration method, it is important to monitor truncation error or to otherwise assure that the integration is accurate. Developing code that obtains acceptable errors in the integration in a timely fashion is the essence of the difficulty of developing a ray integration code. The most stringent accuracy requirements occur in the integration of ray angle. The nature of this difficulty can be shown by noting that the main term in Eq. 4,$(1/c)\,dc/dz$equals $d(\ln(c))/dz$, and that sound speed varies by only about 5% in the water column. Since $\ln(c)$ varies by even less, the errors in estimating the derivative

rapidly wreck the integration.

The *Numerical Recipes* textbook describes an integration method involving "adaptive stepping." This technique involves measuring the truncation error at each step and then modifying the step size such that the integration maintains a specified level of error. This ensures both that the integration is accurate and that step sizes are not so small as to involve unnecessary computation. Properly employed, an "adaptive step" method can result in significant improvements in computation time.

In the present case, however, we already know a great deal about the ray tracing problem, unlike a generic integration problem. For example, given the nature of the ray tracing problem, it is probable that step sizes of 1 m are too small, and step sizes of 1000 m are too large. It thus appears that using adaptive stepping will not result in an order of magnitude decrease in computation time as might be expected for an unknown, generic integration problem. It is true that adaptive stepping gives a measure of truncation error, which is quite important. However, with experience the character of a predicted wave front can be used to decide if predetermined step sizes are sufficiently small, or a few initial calculations can be made with different step sizes to test the convergence of the integration. Once the parameters of the integration are determined, speed of computation is the only remaining issue.

It is not presently known whether an adaptive stepping method can be developed that will result in speedier computations than a user-specified stepping. It is certainly true that the "adaptive stepping" method described in the *Numerical Recipes* textbook is more computationally expensive than the user-specified stepping method that has been developed for the present ray tracing code. It may be that once the nature of the problem has been well defined, adaptive step sizing is a computational overhead that is no longer required. Once the step sizes have been defined for the problem, the step sizes do not need to be recalculated every time. Figure 1 shows the step sizes determined by an adaptive step ray trace and the predetermined step size currently implemented in the code. This predetermined step size is linear from the surface to a user-specified depth (1500 m in Figure 1) and has a hyperbolic tangent (tanh) functional form below that depth. While the predetermined steps are generally considerably smaller than the adaptively sized steps, the code using the former is significantly faster than the code using the latter.

Besides the method used to derive the step sizes, a second issue is the method of integration to be employed. The *Numerical Recipes* textbook lists several methods: Classical 4th order Runge-Kutta, Cash-Karp Runge-Kutta, Bulirsch-Stoer, "predictor-corrector," and "stiff problem" integration. The Cash-Karp Runge-Kutta method is a 5th order method which also allows an estimate of truncation error using the difference between 4th and 5th order Runge-Kutta results. Since one often needs to integrate through mixed layers and other sharp features, the best integration method is probably Runge-Kutta, because many of the other methods require the integration problem to be relatively smooth. For a suitably chosen step size, the classical 4th order Runge-Kutta and the Cash-Karp Runge-Kutta methods produce nearly identical results in the ray trace. As pointed out in the *Numerical Recipes* textbook, higher order does not necessarily

mean higher accuracy. The classical 4th order Runge-Kutta is much cheaper to calculate. We tried to implement the Bulirsch-Stoer method, but it produced unsatisfactory results. We did not try to implement any "predictor-corrector" methods.

At the present time we conclude that the most efficient way to make these calculations is to use the classical 4th order Runge-Kutta with a step size that is predetermined. Several test cases of varying step sizes can be run initially to ensure that the integration is accurate before proceeding with the optimal calculations. The main issue is to balance speed and accuracy. We hold out the hope, however, that a more efficient adaptive stepping scheme may be devised. (The Bolin RAY code adopts 4th order Runge-Kutta and predetermined step size as well. However, the step sizing is determined by the depth spacing of the sound speed data that are input. As a result Bowlin RAY predictions using sound speed profiles defined by many depths take longer to compute than predictions using sound speed profiles defined by few depths. It is probably better to have the step size vs accuracy issue explicitly recognized by the user of the code.)

Step sizing is critical. Unsuitably defined step sizing causes irregular wave front predictions and can introduce biases to the time front. The user performing ray tracing for the first time on a particular problem is urged to try several step sizing definitions to ensure that the desired accuracy is obtained. It is far more difficult to derive eigenrays if the time front is irregular and rough. For adaptive step sizing the scaling of the variables in the problem is critical. High accuracy is needed at every step to avoid the accumulation of errors during the integration. This means that the scaling of the variables needs to be proportional to the step size. Small step sizes are required where $d \cos\theta / dr \approx 0$ (i.e., where $dc/dz \approx 0$) or where the sound speed changes rapidly. Thus small step sizes are usually required near the sound channel axis or near the surface.

Other methods of integration have not yet been tested. It may well be that faster, more efficient methods than Runge-Kutta integration could be found. There is surely a mathematical analysis of the ray tracing problem that would point to the optimal method, but this is beyond the means of the authors.

SURFACE AND BOTTOM REFLECTIONS

Surface reflections are handled in a simple way in the code. The code determines at every step if a surface reflection is to occur in the next range step. If h is a range step to take after reaching a depth z_n, then the depth of the ray after this step can be approximated by $z_{n+1} \approx z_n + dz/dr\, h$. If z_{n+1} is above the ocean surface, we can analytically solve for a new, shorter step size such that the ray will arrive exactly at the surface after this step. This works best (the approximate integration for the depth z_{n+1} has less error) when very small step sizes are taken near the surface. Once this has been done, we need merely change the sign of the ray angle and continue with our calculation.

Because the step size derived to arrive exactly at the surface is approximate, the integration will occasionally produce a ray that passes infinitesimally above the ocean

surface. To ensure that the calculation can proceed without mishap in this case, the sound speed table includes a value for sound speed at 3 m above the ocean surface. The user can specify a tolerance for missing the ocean surface. In general, depth errors of several centimeters or more in the surface reflection will not significantly affect the ray tracing results.

Bottom reflections are handled in a manner similar to surface reflections, with a few complications. The bottom is modeled as line segments between the depths specified in the input bathymetry file, and the ray reflects specularly. By simple geometry, if the incident angle is θ, the reflected angle is θ', and the angle of the bottom with respect to the horizontal is ϕ, then $\theta' = -\theta + 2\phi$. The angle ϕ is the arctangent of the slope of the bottom.

The complexities arise in determining when the ray is approaching the ocean bottom and in solving for the step size required for the ray to reach the bottom exactly without overshooting. Suppose the segment of ocean bottom is modeled as $B(r) = mr + b$, where m is the bottom slope and b is a line intercept, and suppose the ray path is $z(r)$. The sign of $z - mr - b$ therefore determines which side of the bottom the ray is. Similarly to the surface reflection procedure, the sign of $(z + h\, dz/dr) - m(r + h) - b$ is a test of whether the ray will cross the bottom on the next step of size h. This equation can also be used to solve for the step size h for the ray to exactly strike the bottom. However, because much larger step sizes are used near the ocean bottom than near the surface, the approximate integration by dz/dr does not work very well. The code uses the constant ray slope approximation to determine when the ray is near the bottom, and then iterates step size to obtain the step size required to put the ray at the ocean bottom within a user-specified tolerance. In practice, a look-up table of range, depth, bottom slope, twice bottom angle, and the value of an intercept of the line segment is used to determine which segment of the bottom the ray is approaching and to calculate the reflection point and new ray angle.

This code does not model the finite frequency effects on ray paths with small slope that pass near the surface (Murphy and Davis 1974; Odom 1998). The geometric ray theory described in this memorandum is an infinite frequency approximation which permits rays to make an arbitrarily close approach to the surface without being influenced by it. Only when the geometric rays actually contact the surface are they reflected. A finite frequency ray contains energy which decays exponentially above the geometric turning point, as shown by predictions using acoustic modes. The length of the exponential tail is frequency dependent with lower frequency rays having longer tails, thus more energy above the geometric turning point. Because of the exponential tail, the finite frequency ray senses the surface sooner than the corresponding geometric ray. The result is that the finite frequency ray turns at a shorter range from the source than the geometric ray with the same launch angle. This effect can cause significant changes to ray travel times and ray paths in very long range acoustic transmissions. In one example from the Eastern Pacific, the travel times were affected by 1–2 ms per surface interaction, but the rays have to be grazing the surface in just the right way for an effect of this magnitude

(M. Dzieciuch personal communication 1999, Worcester et al. 1998). However, it is difficult to model the near-surface interaction with rays in a general way, particularly for broad-band acoustic transmissions. For the time being, those concerned about this effect may make an empirical assessment of its magnitude by comparing ray and mode or parabolic equation predictions.

BENCHMARKING VS ACCURACY

The speed of a ray prediction depends on the accuracy required by the user. Thus, it is virtually impossible to compare the speeds of various ray codes, unless one has a means of ensuring that the various predictions are to the same accuracy. The table below shows computation times for 5000 rays when using an early version of the ray code. The calculations are for a 3-Mm range, range-dependent sound speed section. For these calculations, the Step Size Scale is a scaling factor that is applied to the step size function shown in Figure 1. The table shows that the computation time depends linearly on the step size.

Step Size Scale	0.5	1.0	2.0	4.0
Computation Time (min.)	32.0	16.5	8.75	4.5

Figure 2 shows that the time front predictions associated with these various step size scalings deteriorate considerably as step size is increased. By comparison, an adaptive step size prediction took about 1 hour, the Bowlin RAY prediction (Figure 2) took 34.0 minutes, and the Colosi prediction (Figure 2) (paranoid for accuracy) took over 5 hours. While for all of these predictions the location of the individual ray arrivals may vary, all of the time fronts agree within milliseconds. The shape and absolute travel times of time fronts appear to be fairly robust.

The lesson here is that the user of a ray code needs to balance his or her desire for accuracy against the speed of computation.

The astute reader will have noticed in Figure 2 occasional gaps in the time fronts, particularly in the earliest part of the arrival pattern. These gaps apparently result from the ray approximation; similar predictions using the parabolic equation do not show these gaps. In the ray approximation, rays which pass near a surface layer will either travel into that surface layer or miss it altogether, even though the launch angle in the two cases may differ only infinitesimally. This property of the ray approximation is a likely cause of the gaps in the time front prediction. When such gaps occur near the depth of a receiver, it can be difficult to obtain an associated eigenray prediction. Indeed, difficulty in obtaining the eigenray prediction for a ray known to exist probably results from such an unphysical gap in the time front.

CONCLUSIONS

We have described a method for making fast, accurate computations of acoustic rays as a tool for ocean acoustic tomography. We described ray equations, including a derivation of ray equations that include the effects of weak ocean currents (Appendix F). Methods using cubic spline interpolation and a look-up table allow sound speed and sound speed gradient to be calculated rapidly and accurately at arbitrary range and depth. The choice of the step size used in integrating the differential equations is critical, affecting both the computation time and the accuracy of the ray predictions. The best method for integrating the ray equations appears to be ordinary, classical 4th order Runge-Kutta integration. Presently known integration methods using the adaptive-step techniques to maintain a user-specified numerical accuracy appear to carry too much computational cost to be competitive. A user-specified step size greatly increases the efficiency of the computation, but a more efficient adaptive-stepping method may yet be devised. Integration accuracy can be initially checked for a particular problem by repeating the ray predictions with several trial step-size functions.

For 5-Mm range acoustic transmissions across the North Pacific, eigenray travel times calculated with the code described here agreed with the travel times of the Bowlin code to within 20 ms. Similar agreement was found for other calculations at similar multimegameter range. The upper and lower turning depths of eigenray paths calculated using the code here and the Bowlin code agreed to within a few meters. Sound speeds calculated using the annual mean Levitus ’94 ocean atlas were used for these calculations. When calculating time fronts to similar accuracy, the present code appears to be comparable in speed to the Bowlin code. However, when calculating eigenrays (Appendix B), the present code appears to be 2–3 times faster than the Bowlin code because most of the time it integrates only two differential equations. In addition, the present eigenray code appears to be a little more effective at finding the eigenrays than the Bowlin code, so fewer rays need to be calculated to define the initial time front used to find the eigenrays.

The code described here is inherently unstable, yet it is highly flexible. The computer code is easily modified for particular problems. One application of this code may be to calculate the forward problem matrices used for travel time inversion while calculating the eigenrays. This calculation would result in more accurate matrices and it will not require saving the ray paths to one’s hard disk. As described in some of the appendices, the code can calculate eigenrays, can be modified to run on a parallel computer for rapidly obtaining ray predictions, or can be modified to include the effects of ocean current. The present suite of software consists of about 1800 lines (including lots of commentary) of FORTRAN code.

APPENDIX A
A TECHNICAL SUMMARY AND A FLOW CHART OF THE COMPUTER CODE

A flow chart sketching the computer code is shown in Figure 3. The main program **ray.f** first sets up the sound speed look-up tables (subroutine **speed.f**) before integrating the rays (subroutine **dodeint.f**) with the desired ray launch angles. The code is in double precision throughout. Note that for the code to be universally portable the common tables must have the double precision arrays and variables listed before the integer variables.

The sound speed look-up tables (**ctab**) are calculated using the *Numerical Recipes* cubic spline routines **dspline.f** and **dsplint.f**. The subroutine **dspline.f** calculates the second derivative of sound speed, from which the cubic splined sound speeds are obtained. The subroutine **dsplint.f** has been modified to obtain the subroutine **dsplint_both.f** which returns the values of both sound speed and sound speed gradient. While reading in the values for sound speed, the subroutine also obtains the table of profile ranges (**range**).

A step size look-up table (**steps**) is also determined in the subroutine **speed.f**. As mentioned earlier, this predetermined step size is linear from the surface to a user-specified depth and then has a tanh functional form below that depth. The user is asked to specify the transition depth from linear to tanh forms and the step size values at the surface, at the transition depth, and at 5500-m depth.

The bathymetry look-up table (**btab**) is determined in the subroutine **bathy.f**. This subroutine reads in the maximum allowable bottom bounces (**MBONK**), the tolerance for the ray to miss the ocean bottom (**btol**), and the bathymetry data. The bathymetry look-up table consists of the five variables: range, depth, slope, twice the slope angle, and a value of an intercept of the line segment.

Once the sound speed, range, step size and bathymetry look-up tables have been defined, the rays are integrated using the subroutine **dodeint.f**. Presently the code loops over a set of rays at equally incremented launch angles. The subroutine **dodeint.f** has been heavily modified from the *Numerical Recipes* subroutine example of the same name. The role of this subroutine is as a driver for taking the Runge-Kutta steps (step size **h**) with the subroutine **drk4.f**; the user wishing to change the method of integration from Runge-Kutta may merely substitute an alternate subroutine to **drk4.f**. The subroutine **drk4.f** has been only slightly modified from the *Numerical Recipes* subroutine of the same name. The subroutine **drk4.f** calls the subroutine **derivs.f** which calculates the coupled differential equations (the ray equations) at arbitrary depth and ray angle. The subroutine **dodeint.f** saves the ray paths derived during the integration in variables **xp** and **yp** and returns the cosine and sine ray angle at the receiver range, the depth of the ray at the receiver range, and the ray travel time for writing to a file.

Range dependence is implemented in the **derivs.f** subroutine using the variable **deltaX**; **deltaX** is the horizontal range between a step in the integration and the range of the sound speed profile most recently exceeded by the ray. Since range dependence is implemented using a constant sound speed gradient between profiles, the sound speed at

any step in the integration can be found by $c(r, z) = c(r_i, z) + deltaX\ dc/dr$. Here, $c(r_i, z)$ is the sound speed profile at range r_i. The value for the vertical gradient of sound speed can be found similarly. In general, the step size **h**, O(5–200) m, is much less than **deltaX**, O(10 km).

APPENDIX B
CALCULATION OF EIGENRAYS

There are several ways in which eigenrays can be computed accurately, yet with far fewer calculations than are required for a complete time front prediction. In all cases, eigenrays must be found by first tracing a fan of rays and then determining the ray launch angles that result in rays arriving at the receiver depth. However, there are a number of shortcuts that can be implemented.

In a ray fan equally spaced in launch angle, most of the rays concentrate in the cusps of the time front and so frequently do little to resolve the time front in the depth region of interest. Thus, one of the simplest ways to speed up eigenray calculations is to use an initial prediction with a small number of rays in order to define the range of ray angles that arrive near the depth of the receiver. Using these selected angle ranges, a second prediction with many rays can be done that is far more efficient at defining the time front near the depth of the reciever.

A second way to gain efficiency is to omit the calculation of travel time in the initial fan of rays. All that is required to derive eigenrays is the angle that produces a ray that arrives at the receiver depth, so that travel time is not necessary. The present incarnation of the eigenray code implements this using only two coupled differential equations (for tangent of ray angle and ray depth; see Eq. B1 below), rather than the four required for obtaining travel time as well. Once the launch angles resulting in rays that arrive at the receiver depth are found, a small number of integrations with all four differential equations can then be performed to obtain all the information about the eigenrays.

Eigenray predictions are probably more efficient if fewer, more accurate rays are used to calculate the time front rather than many, inaccurate rays. The more linear the relation between launch angle and arrival depth, the easier it will be to obtain the eigenrays.

After the two ray arrivals that span the receiver depth are determined, eigenrays are probably best found by a sequence of bisections of ray launch angle giving a sequence of rays that converge on the receiver depth. An initial interpolation to solve for the eigenray launch angle might improve efficiency, but in general the ray arrivals are not linear in launch angle and arrival depth, so the interpolation will not be accurate. The code presently uses ordinary bisection on rays that span the receiver depth. Because of nonlinearities in the relation between launch angle and receiver depth, the sequence of bisections sometimes misses the receiver depth. Various contingencies have been built into the code to account for this, but it may sometimes happen that the eigenray cannot be found at all. It appears that at 3-Mm range, usually less than five bisections are required to find a ray that arrives within 10 m of the receiver depth.

If one wants only the travel time of the eigenrays, one could interpolate the predicted time front to get the travel times associated with the depth of the receiver. Travel times determined in this way will (for benign oceans) be accurate to within a few milliseconds.

Thus, the travel times of the eigenrays can be determined without having to actually solve for the eigenray path. This technique has not been implemented.

The code here is flexible enough that the user can easily implement ideas for finding eigenrays. If someone develops code that works well for determining eigenrays, he or she is requested to forward those methods to the authors of this report.

The code for finding eigenrays is similar to that described in Appendix A, with the addition of subroutines **dodeint_short.f, drk4_short.f, and derivs_short.f**. These subroutines use only two differential equations in tangent of ray angle and ray depth. If $tn = tan(\theta)$, then these equations are

$$\frac{dtn}{dr} = (1 + tn^2)\,\frac{\partial_{z'} c}{c} \tag{B1a}$$

$$\frac{dz}{dr} = tn \tag{B1b}$$

$$\left(\frac{dt}{dr} = \frac{\sqrt{1 + tn^2}}{c} \right) \tag{B1c}$$

The subroutine used to find the eigenray ray launch angles is **find_arr.f**. This routine uses the time front information from the initial ray traces to select pairs of ray angles with depths at the receiver range that span the receiver depth. A series of bisections on ray launch angle is used to converge on the desired eigenray ray launch angle. The rays at the selected angles are then recalculated using all four differential equations to obtain all the information about the ray. Since this entire procedure relies on these two different integrations giving nearly identical ray paths, the procedure is actually one test of integration accuracy. If the eigenrays in the second integration miss the receiver depth, one problem may be that the step sizes are too big. However, the integrations are so sensitive that the series of step sizes during the two integrations must be fairly well synchronized. If even slightly different step sizes are used in the second integration, the rays can miss the the receiver depth by O(100 m).

Once the eigenrays have been calculated, frequently the next step for the tomographer is to calculate the forward problem matrix,

$$G_{ij} = \int_{\Gamma_i} \frac{\eta_j(\mathbf{x})\,ds}{c(\mathbf{x})^2} \tag{B2}$$

Here Γ_i is the ith ray path, η_j is one of a set of functions used to model the ocean, ds is an element of ray path length (equal to $dr/cos(\theta)$), and c is sound speed. Traditionally, to facilitate this calculation the parameter ds/c^2 is calculated for each saved step in the ray trace. With this parameter, the above integral can be calculated with relative ease. However, ds/c^2 is not entirely easy to calculate with rigorous accuracy.

The problem of calculating ds/c^2 is illustrated in Figure 4. While a ray path is calculated using very small horizontal step sizes, O(5–200 m), the ray path is saved at a much large range increment, O(1000 m). What is required, then, is a value for ds/c^2 appropriate for this much larger horizontal range increment. It is not obvious how this parameter is to be calculated rigorously. It is easy, of course, to approximate ds by summing the small ds_i that occur during the integration, but what of $1/c^2$? Should the values of c, $1/c$, or $1/c^2$ be averaged over the several small steps? Sound speed varies so little over a path increment that it probably does not matter too much. The present code calculates ds_i/c^2 at each integration step and sums these values over the several integration steps that comprise the saved range increment. The value of ds/c^2 is saved together with the ray path range and depth pairs. Possible errors may arise at depths where the sound speed or mode functions are varying rapidly, however.

With the present set of subroutines, it would be relatively easy to merge the ray tracing routines with the calculation of the forward problem. In this way, the forward problem matrices can be calculated at the very small step sizes of the integration on the fly, as it were, without having to fill one's hard disk up with the stored ray paths. This is the ideal way to calculate the forward problem matrices.

The Bowlin et al. (1992) RAY code appears to address the problem of ds/c^2 by integrating a variable $q = s/c^2$. With this variable, ds/c^2 may presumably be calculated at the nth step by $q_n - q_{n-1}$. This procedure is not correct, since $d(s/c^2) = ds/c^2 - 2\, s\, dc/c^3$. The second term has a dangerous dependence on the path length itself. The second term oscillates in sign because dc changes sign from upward-going to downward-going rays. Calculation of this variable requires the integration of an additional differential equation. For small values of s, the values of ds/c^2 derived from present code and from the Bowlin code appear agree to 1 part in 10,000 for the North Pacific environment modeled by the annual mean Levitus atlas.

APPENDIX C
MODIFYING THE CODE FOR COMPUTATIONS IN PARALLEL

Calculating ray predictions for long-range transmissions is a natural application for parallel computation. Each ray trace is independent of all the others, so several computer processors can calculate subsets of rays separately. A parallel version of the ray code described here has been developed using the MPI suite of subroutines (Gropp et al. 1994).

In order for several processors to be able to trace rays, all that is required is that they have access to the sound speed look-up table. Thus, the parallel version of the code broadcasts this look-up table, together with a few other variables, to all available processorss. With this information, the available processors can trace subsets of the desired rays. When they have completed these calculations, they then communicate the results (travel times, ray paths, etc.) back to the master process for printing or other use. The MPI routines perform these communications tasks (broadcasting to all processors or sending and receiving information between processors) relatively easily. The speed of the ray calculations scales well with the number of available processors.

With a homogeneous set of processors, the complete set of rays to be calculated can be broken up into equally sized subsets. Each processor then gets a subset of rays to calculate. The present parallelized version of the ray code adopts this strategy.

With an inhomogeneous set of processors, "load balancing" becomes an issue. Fewer rays should be sent to slower processors. It may be better to assign rays singly to processors and assign additional rays to a particular processor as its ray calculations are completed. MPI includes "non-blocking" send and receive communication subroutines which may be used for this purpose. That is, the code will not wait for a particular ray trace to be completed before assigning rays to other processors. In this way, a processor that is four times faster than another processor will complete four times as many rays without requiring the user to preallocate unequally sized subsets of rays to the processors. This strategy has not been implemented, but it would not be difficult to do so.

APPENDIX D
INPUT AND OUTPUT FILES AND OTHER OPERATIONAL INFORMATION

As noted often earlier, this code is meant to be modified by the user for his or her own purposes. This includes the formats of the sound speed data files and the means to input other information such as receiver depth, range, and other parameters of the ray trace as well as the output formats. The present setup of the code is described here as an example of a way to implement the code.

The code is presently run by executing the command

time eigenray < in.ray

"time", of course, reports the time it takes to execute the raytrace; "eigenray" is the executable, and the file "in.ray" contains all of the relevant parameters that "eigenray" asks for.

An example of a file for input to eigenray:

8. 12.	Starting and ending values for ray angle (degrees)
100 1500.	# rays to sketch out time front, and depth below which to omit rays in the second raytrace. The second ray trace is not performed for time front calculations.
100	# rays in second ray trace.
1000 1000	Source and receiver depths (meters)
0.0 3000000	Start and end ranges (meters)
1000.0	The range increment to save the ray paths (meters)
10.0	Tolerance for missing receiver depth (meters)
1	Save eigenray paths? 0=no, 1=yes
0	Calculate either eigenrays (=0) or a time front (=1)
1	Enable ocean bottom? 0=no, 1=yes
test.ssp	The sound speed filename
1500.	Transistion depth for step size, linear to tanh (meters)
5. 100. 500.	Values of step size at the surface, the transition depth, and at the bottom (all in meters)
1.0	Step size scaling e.g., 0.5,1.0,2.0; for testing accuracy
30	The maximum number of bottom bounces before dropping a ray
5.0	The tolerance for missing the ocean bottom (meters)
test.bth	The bathymetry file name. These last three lines are not required if the flag disabling bathymetry is set.

The file of sound speeds has the same format as that of the Bowlin RAY code, with "-1 range" marking the start of a profile, followed by depth and sound speed.

An example sound speed file:

```
 -1     0.000
  0  1500.597
 10  1500.465
 20  1499.793
 30  1498.795
 50  1495.540
 75  1491.698
100  1489.109
125  1487.366
150  1486.174
200  1484.450
250  1483.021
300  1481.752
400  1479.939
500  1479.103
600  1479.168
700  1479.679
800  1479.737
900  1480.225
1000  1480.835
1100  1481.586
1200  1482.276
1300  1483.210
1400  1484.137
1500  1485.137
1750  1487.678
2000  1491.187
2500  1498.126
3000  1506.161
3500  1514.534
4000  1523.252
4500  1532.259
5000  1541.404
5500  1550.529
 -1    50.963   A new sound speed profile at 50.963 km range.
  0  1500.751
 10  1500.612
 20  1499.982
 30  1499.038  (etc., repeated as necessary)
```

When the flag enabling bathymetry is set, the code requires a bathymetry file consisting of simply two columns of range and depth, both in meters.

The time front or eigenray data are written to the file "ray.info". This file consists of the travel time, ray angle at the source, ray angle at the receiver, ray upper and lower turning depths, depth of ray at receiver range, and number of turning points. The ray paths are written to the file "ray.data"; this is not recommended when calculating a time front. The columns of this file are range (meters) and depth (meters) along the ray path and the parameter ds/c^2 along the ray path which is used for calculating the matrices for inversion of travel times.

APPENDIX E
THE RAY EQUATIONS IN TERMS OF SOUND SLOWNESS

The ray equations (4a–4d) can be considerably simplified if they are formulated in terms of sound slowness. Sound slowness is the reciprocal of sound speed, $p = 1/c$. The following equations are equivalent to Eqns. (4a–4d), but they are not currently implemented in the code. They are included here for reference. If $A = p\,cos(\theta)$ and $B = p\,sin(\theta)$, then

$$\frac{dA}{dr} = \frac{p}{A}\frac{\partial p}{\partial r} \tag{E1a}$$

$$\frac{dB}{dr} = \frac{p}{A}\frac{\partial p}{\partial z} \tag{E1b}$$

$$\frac{dz}{dr} = \frac{B}{A} \tag{E1c}$$

$$\frac{dt}{dr} = \frac{p^2}{A} \tag{E1d}$$

These equations are ordinary differential equations for A, B, z, and t. One advantage of using these equations is that one division is saved, and so these equations may be about 15% faster to compute than Eqns. (4a–4d). However, in Eqns. (4a–4d), the terms involving $\partial c/\partial r$ are neglected, while this is a term essential to equation (E1a). It is unclear to us what the implications of this fact are, e.g., whether small errors in the calculation of $\partial c/\partial r$ may prove to be a difficulty in implementing Eqns. (E1a–E1d). The approximation that $dA/dr \approx 0$ is probably not adequate. In addition, there are certain logistical difficulties with using a look-up table for sound slowness.

APPENDIX F
THE RAY EQUATIONS WITH CURRENT

The code does not presently offer the option of including the effects of ocean current. We derive here the ray equations that include the effects of ocean current. The user interested in calculating the effects of current can modify the code to include the current effects.

A quick review of the literature (i.e., Franchi and Jacobson 1972; Munk et al. 1995) found no ray equations that included the effects of current and could be easily implemented for numerical calculation. There seems to be a tendency in the literature toward the cryptic and toward obscure notation. We therefore include here a derivation which may prove to be practically useful.

The textbook by Pierce (1989) gives ray equations with current in terms of the derivative with respect to travel time. These equations are

$$\frac{d\mathbf{r}}{dt} = \frac{c^2\mathbf{p}}{\Omega} + \mathbf{v} \tag{F1a}$$

$$\frac{d\mathbf{p}}{dt} = -\frac{\Omega}{c}\nabla c - \mathbf{p}\times(\nabla\times\mathbf{v}) - (\mathbf{p}\cdot\nabla)\mathbf{v} \tag{F1b}$$

or (in Cartesian coordinates)

$$\frac{dp_i}{dt} = -\frac{\Omega}{c}\frac{\partial c}{\partial r_i} - \sum_{j=1}^{3} p_j \frac{\partial}{\partial r_i} v_j \tag{F2}$$

Here $\mathbf{r}$ is ray spatial coordinate, $\mathbf{p} = \mathbf{k}/\omega$ is wave-slowness vector, $\mathbf{v}$ is current, and

$$\Omega = \frac{c}{c + \mathbf{v}\cdot\mathbf{n}} \tag{F3}$$

where $\mathbf{n}$ is a unit normal that is parallel to the ray. We can immediately make a number of simplifications to these equations. We assume that $\mathbf{v} = (v, 0, 0)$, i.e., current is only in the direction of the plane of acoustic propagation. We assume that the user of the code will project current onto the path between the source and receiver and that currents in the vertical are negligible. Further, the wave-slowness vector is

$$\mathbf{p} = (cos\theta, 0, \sin\theta)/c \tag{F4}$$

and

$$\Omega = \frac{c}{c + v\cos\theta} \tag{F5}$$

These simplifications result in the following equations:

$$\frac{dr}{dt} = \frac{c\cos\theta}{\Omega} + v \tag{F6a}$$

$$\frac{dz}{dt} = \frac{c\sin\theta}{\Omega} \tag{F6b}$$

$$\frac{d\left(\frac{\cos\theta}{c}\right)}{dt} = -\frac{\Omega}{c}\frac{\partial c}{\partial r} - \frac{\cos\theta}{c}\frac{\partial v}{\partial r} \tag{F6c}$$

$$\frac{d\left(\frac{\sin\theta}{c}\right)}{dt} = -\frac{\Omega}{c}\frac{\partial c}{\partial z} - \frac{\cos\theta}{c}\frac{\partial v}{\partial z} \tag{F6d}$$

While it may make for an interesting exercise to integrate these equations using small steps of travel time, this is not particularly useful in the present context. We therefore seek to use the magic of mathematics to transform these equations into something useful. We will assume that $v/c \ll 1$ and keep terms only to first order in v/c. For example,

$$\Omega \approx 1 - \frac{v\cos\theta}{c} \tag{F7}$$

The equation for dt/dr can be found by taking the reciprocal of the equation for dr/dt:

$$\frac{dt}{dr} = \frac{1}{c\cos\theta} - \frac{v}{c^2}(2 + \tan^2\theta) \tag{F8a}$$

The equation for dz/dr can be similarly found by $dz/dt/dr/dt$:

$$\frac{dz}{dr} = \tan\theta\left(1 - \frac{v}{c\cos\theta}\right) \tag{F8b}$$

The differential equations for $d\cos\theta/dr$ and $d\sin\theta/dr$ can be derived similarly by noting that

$$\frac{d\left(\frac{\cos\theta}{c}\right)}{dr} = \frac{1}{c}\frac{d\cos\theta}{dr} - \frac{1}{c^2}\cos\theta\frac{dc}{dr} \tag{F9}$$

and

$$\frac{dc}{dr} = \frac{\partial c}{\partial r} + \frac{\partial z}{\partial r}\frac{\partial c}{\partial z} \tag{F10}$$

After six pages of algebra (which interested readers should probably do for themselves to check), the set of differential equations, including those for ray angles, is

$$\frac{d\cos\theta}{dr} = -\sin\theta\left(\frac{1}{c}\frac{\partial c}{\partial r}\tan\theta - \frac{1}{c}\frac{\partial c}{\partial z}\right) \tag{F11a}$$

$$+ (3 + \tan^2\theta)\frac{v}{c^2}\frac{\partial c}{\partial r} - \frac{1}{c}\frac{\partial v}{\partial r} - \tan\theta\frac{v}{c^2}\frac{\partial c}{\partial z}$$

$$\frac{d\sin\theta}{dr} = \cos\theta\left(\frac{1}{c}\frac{\partial c}{\partial r}\tan\theta - \frac{1}{c}\frac{\partial c}{\partial z}\right) + 3\frac{v}{c^2}\frac{\partial c}{\partial z} - \frac{1}{c}\frac{\partial v}{\partial z} \tag{F11b}$$

$$\frac{dz}{dr} = \tan\theta\left(1 - \frac{v}{c\cos\theta}\right) \tag{F11c}$$

$$\frac{dt}{dr} = \frac{1}{c\cos\theta} - \frac{v}{c^2}(2 + \tan^2\theta) \tag{F11d}$$

Equations E11a–E11d constitute the ray equations, modified to account for ocean current to first order. Note that when $v = 0$ these equations satisfactorily reduce to the original ray equations, Eqs. 4a–4d.

In practice, it would be easy to include the current terms in the subroutine **derivs.f** defining the differential equations. The second modification would be in the subroutine **speed.f** to create a look-up table for current analogous to the table for sound speed.

REFERENCES

Aki, K., and P. Richards, *Quantitative Seismology, Theory and Methods*, 2 vols., Freeman, San Francisco, 1980, 932 pp.

Bowlin, J. B., J. L. Spiesberger, T. F. Duda, and L. F. Freitag, Ocean Acoustical Ray-Tracing Software RAY, Tech. Rep. WHOI-93-10, Woods Hole Oceanographic Institution, Woods Hole, Mass., 1992, 49 pp.

Colosi, J. A., E. K. Scheer, S. M. Flatté, B. D. Cornuelle, M. A. Dzieciuch, W. Munk, P. F. Worcester, and A. B. Baggeroer, "Comparisons of measured and predicted acoustic fluctuations for a 3250-km propagation experiment in the eastern North Pacific," *J. Acoust. Soc. Am.*, in press, 1999.

Dushaw, B. D., Inversion of multimegameter range acoustic data for ocean temperature, *IEEE J. Ocean. Eng.*, in press, 1999.

Dushaw, B. D., B. M. Howe, J. A. Mercer, R. C. Spindel, and the ATOC Group, Multimegameter range acoustic data obtained by bottom-mounted hydrophone arrays for measurement of ocean temperature, *IEEE J. Ocean. Eng.*, in press, 1999.

Franchi, E. R., and M. J. Jacobson, Ray propagation in a channel with depth-variable sound speed and current, *J. Acoust. Soc. Am.*, **52**, 316–331, 1972.

Gropp, W., E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, The MIT Press, Cambridge, Massachusetts, 1994, 307 pp.

Moler, C. B., and L. P. Solomon, Use of splines and numerical integration in geometrical acoustics, *J. Acoust. Soc. Am.*, **48**, 739–744, 1970.

Munk, W., P. Worcester, and C. Wunsch, *Ocean Acoustic Tomography*, Cambridge University Press, New York, 1995, 433 pp.

Murphy, E. L., and J. A. Davis, Modified ray theory for bounded media, *J. Acoust. Soc. Am.*, **56**, 1747–1760, 1974.

Odom, R., Near Surface Turning Rays, Technical Memorandum, Applied Physics Laboratory, University of Washington, in preparation, 1998.

Parker, R., *Geophysical Inverse Theory*, Princeton University Press, Princeton, New Jersey, 1994, 386 pp.

Pierce, A. D., *Acoustics: An Introduction to Its Physical Principles and Applications*, Acoustical Society of America, Woodbury, New York, 1989, 678 pp.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd Ed., Cambridge Univ. Press., 1992, 963 pp.

Worcester, P. W., B. D. Cornuelle, M. A. Dzieciuch, W. H. Munk, B. M. Howe, J. A. Mercer, R. C. Spindel, J. A. Colosi, K. Metzger, and T. G. Birdsall, A test of basin-scale thermometry using a large-aperture vertical array at 3250-km range in the eastern North Pacific, *J. Acoust. Soc. Am.*, in press, 1998.

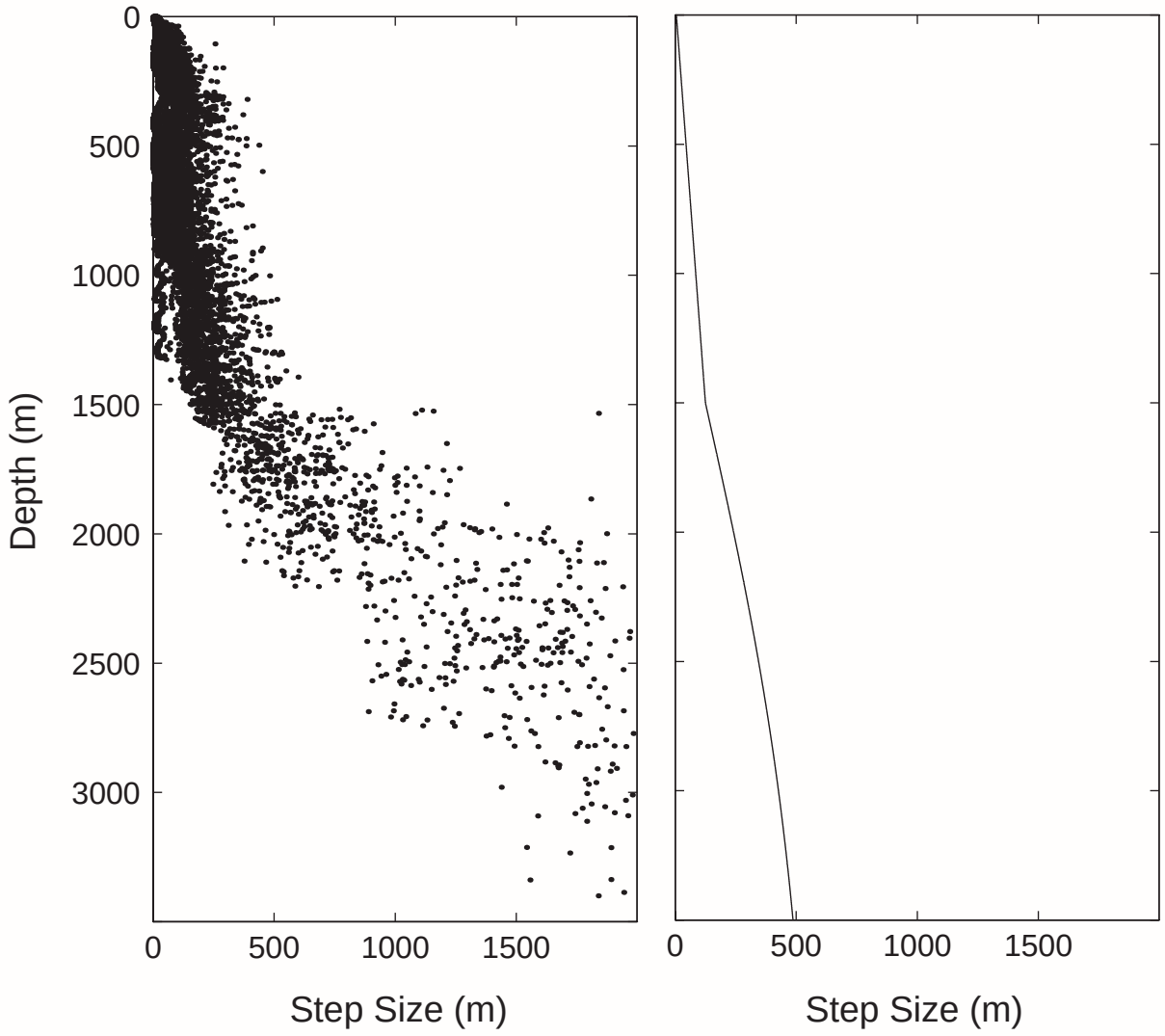

Figure 1. Step sizes determined by an adaptive ray trace and the predetermined step size presently implemented in the code. The left panel shows the step sizes as a function of depthderived by an adaptive step size algorithm used to calculate a single ray with a fairly small error tolerance. The right panel shows an analytic step size function that produces ray predictions as accurate as the adaptive step sizing but that require much less computation time.

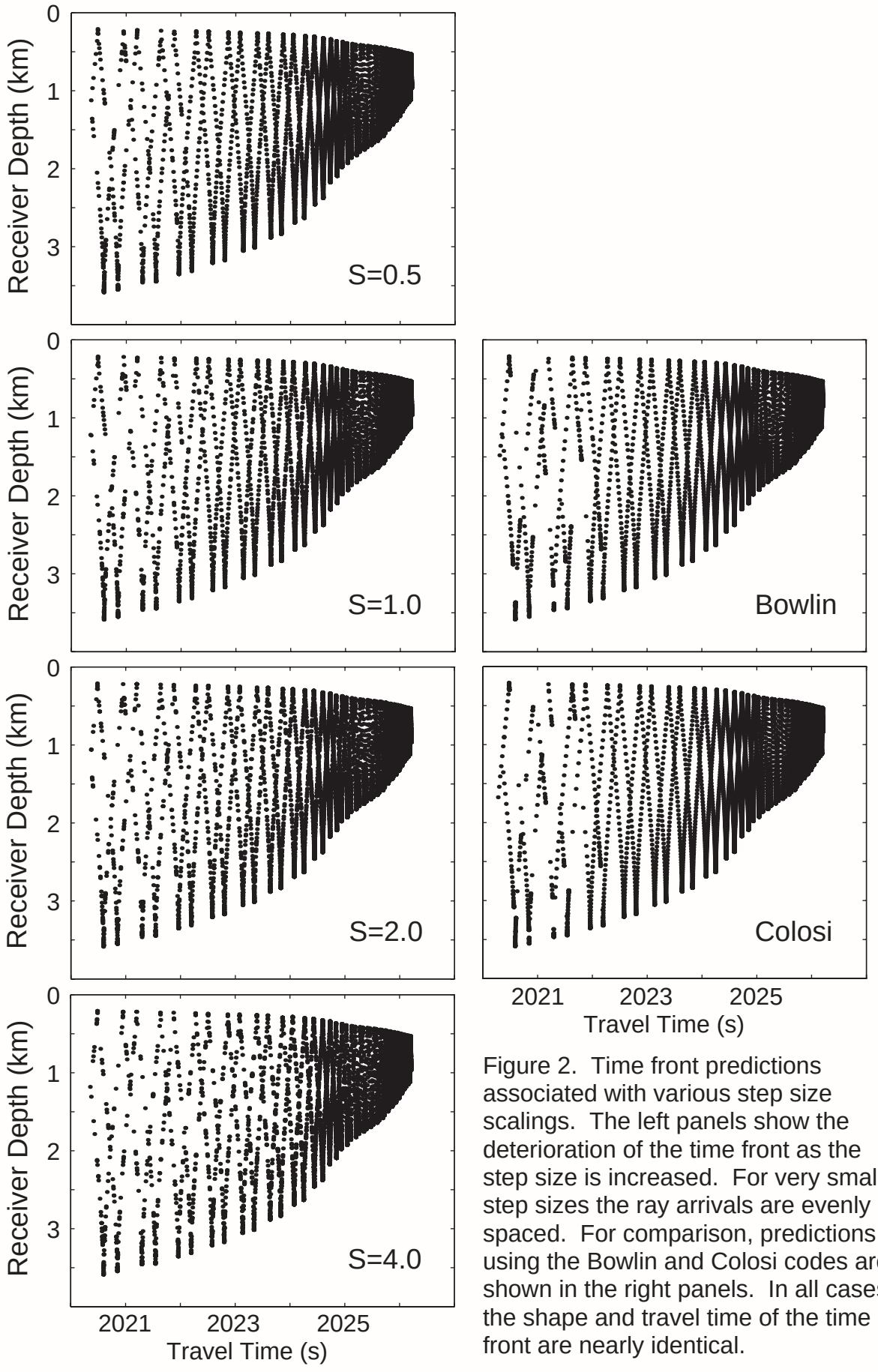

Figure 2. Time front predictions associated with various step size scalings. The left panels show the deterioration of the time front as the step size is increased. For very small step sizes the ray arrivals are evenly spaced. For comparison, predictions using the Bowlin and Colosi codes are shown in the right panels. In all cases the shape and travel time of the time front are nearly identical.

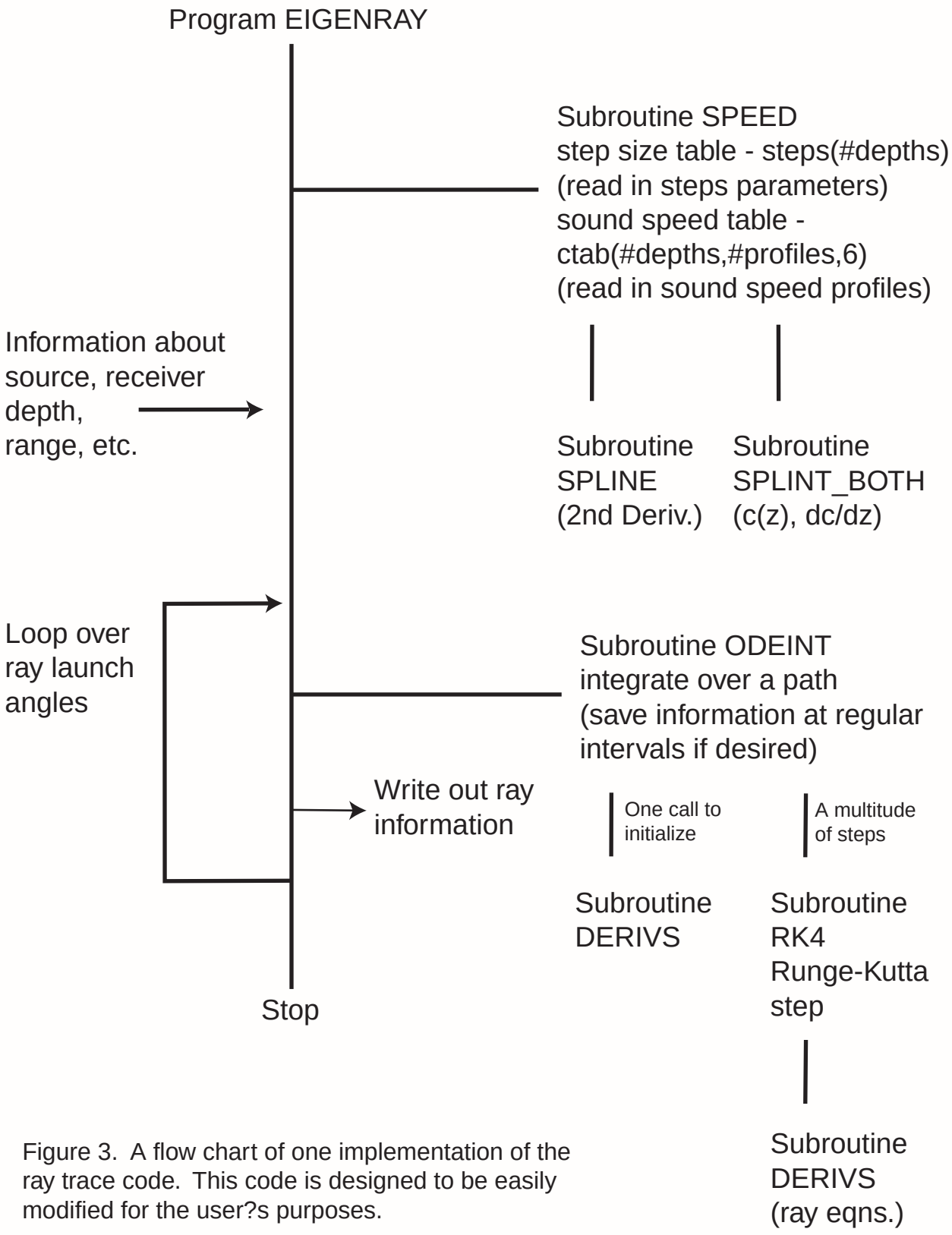

Figure 3. A flow chart of one implementation of the ray trace code. This code is designed to be easily modified for the user?s purposes.

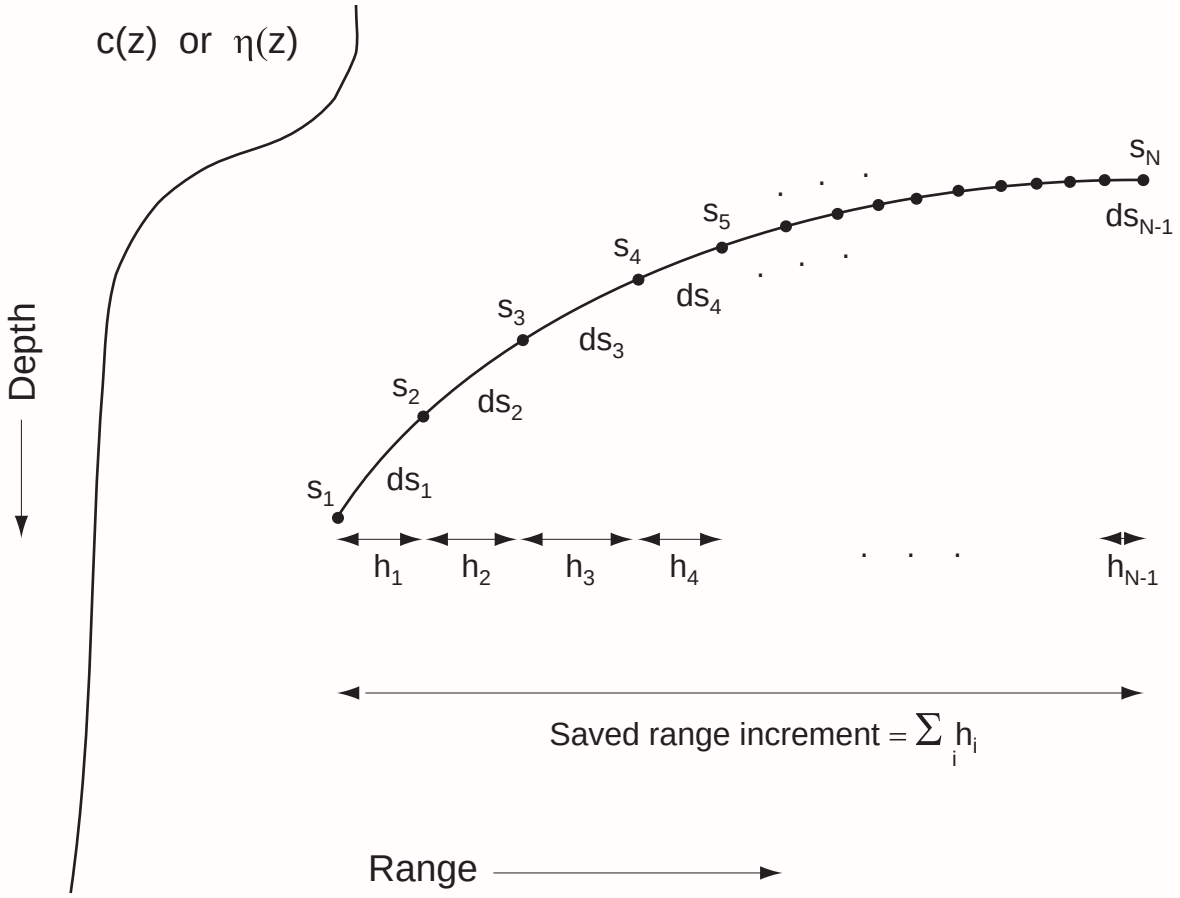

Figure 4. A schematic figure showing the relation of increments of ray path, s, to the sound speed profile or mode function. The ray path is calculated at small range increments h_i (e.g., 5-200 m) but saved at a much larger increments (e.g., 1000 m). A value of ds/c^2 associated with the saved path increment is required to calculate the matrices used in the inversion of travel times. It is probably better to calculate the matrices while performing a ray trace; better accuracy may be obtained by using the small step sizes.

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
December 1998

3. REPORT TYPE AND DATES COVERED
Technical

4. TITLE AND SUBTITLE
Ray Tracing for Ocean Acoustic Tomography

5. FUNDING NUMBERS
DARPA Grant MDA 972-93-1-003
ONR Grant N00014-97-1-0259

6. AUTHOR(S)
Brian D. Dushaw and John A. Colosi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Applied Physics Laboratory
University of Washington
1013 NE 40th Street
Seattle, WA 98105-6698

8. PERFORMING ORGANIZATION REPORT NUMBER
APL-UW TM 3-98

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Ballston Tower 1
800 N. Quincy Street
Arlington, VA 22217

Defence Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes a new, flexible computer code in the FORTRAN computer language to make ray calculations for ocean acoustic tomography. The *Numerical Recipes* software package provided the basis for much of this computer code. The ray equations are reviewed, and ray equations that include the effects of ocean current are derived. Methods are derived for rapidly integrating those equations to obtain time front and eigenray information for long-range, deep-water acoustic transmissions. These methods include a look-up table for sound speed, sound speed gradient, second derivative of sound speed, and range-dependent information. Cubic spline methods are used to interpolate sound speed with depth and to obtain the derivatives of sound speed. The choice of the step size increments used to integrate the equations is a critical aspect of the integration, affecting both the accuracy of the prediction and the speed of computation. A predetermined, user-specified step size appears to allow more efficient calculations than"adaptive step" methods. "Adaptive step" methods adjust the step size automatically to maintain a given accuracy in the integration of the ray equations, while user-specified step sizes allow one to use prior knowledge of the integration problem to achieve the desired accuracy with much less computational overhead. Several integration methods were explored, but the classical 4th order Runge-Kutta method appears to be the most efficient and best method for this integration problem. Appendices describe detailed aspects of the computer code, as well as the methods used for deriving eigenray information and for parallelizing the ray calculations. The computer code is designed to be unstable so that the user can easily modify it to his or her own purposes.

14. SUBJECT TERMS
Ray tracing, ocean acoustics, geometrical optics

15. NUMBER OF PAGES
36

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR