

AD-A072 201

NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA
UNDERWATER SOUND PROPAGATION-LOSS PROGRAM. COMPUTATION BY NORMA--ETC(U)
MAY 79 D F GORDON

F/G 17/1

UNCLASSIFIED

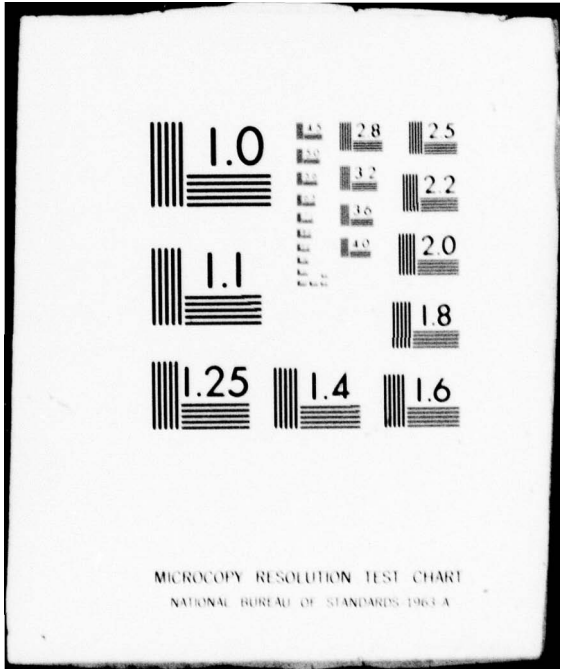
NOSC-TR-393

NL

1 of 2

AD
A072001





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL

12
B.S.

NOSC

NOSC TR 393

ADA072201

D D C
RECEIVED
AUG 1 1979
C

NOSC TR 393

Technical Report 393

UNDERWATER SOUND PROPAGATION- LOSS PROGRAM

Computation by normal modes for layered oceans and
sediments

DF Gordon

17 May 1979

Final Report for Period 1976 — 1978

Prepared for
Naval Sea Systems Command
(NSEA 63R-23) Washington DC 20362

DDC FILE COPY

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

NAVAL OCEAN SYSTEMS CENTER
SAN DIEGO, CALIFORNIA 92152

79 08 1 024



NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA 92162

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

RR GAVAZZI, CAPT, USN

Commander

HL BLOOD

Technical Director

ADMINISTRATIVE INFORMATION

The computer program described in this report was developed in the course of work sponsored by Naval Sea Systems Command, Sonar Technology Office (NSEA 63R-23), under Problem SF 52-552-602, NOSC work unit 714-SU10. Some elements of this program have been in development since 1965, but the final modifications, to achieve the current capabilities, and the reporting were done from 1976 to 1978. The computer program is derived from an earlier program developed by MA Pedersen. D White did significant parts of the mathematical analysis. This report was approved for publication 17 May 1979.

Released by
MR Akers, Head
Systems Concepts and Analysis
Division

Under authority of
GU Rogers, Head
Ocean Surveillance Systems
Department

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NOSC Technical Report 393 (TR 393)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) UNDERWATER SOUND PROPAGATION-LOSS PROGRAM - Computation by normal modes for layered oceans and sediments		5. TYPE OF REPORT & PERIOD COVERED Final Report for Period 1976-1978
7. AUTHOR(s) DF/Gordon		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Ocean Systems Center San Diego CA 92152		8. CONTRACT OR GRANT NUMBER(s) F52552
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command (NSEA 63R-23) Washington DC 20362		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NSEA 06HI SF52-552-602 724-SU10
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE 17 May 1979
		13. NUMBER OF PAGES 100
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) 62759 N		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Ducts Computer programs Propagation Numerical analysis Acoustics Continued fractions Layers Iterations		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A normal-mode program for a sound-speed profile of an arbitrary number of layers has been constructed. It has been used extensively and successfully for 12 years to compute sound propagation in idealized underwater acoustic ducts. Documentation is contained for those who wish to use or modify this program. The FORTRAN statements are given for both the normal-mode and a mode-follower program. The numerical analysis necessary for computing modified Hankel functions of order 1/3 is given. The analysis includes a continued fraction technique.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

393 159

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

OBJECTIVE

Translate well-known differential equation solutions into a working program to compute propagation in underwater acoustic ducts. Document the program methods, to assist users of this and similar programs.

RESULTS

1. An effective program for computing propagation loss in a layered ocean by normal modes has been developed. Complete documentation for the program is contained herein.
2. Sediment layers are modeled as fluids in which densities, sound speeds, and absorption can be specified. This permits a complete wave solution for bottom reflected sound energy.
3. A continued fraction technique for evaluating asymptotic series is shown to give superior results in evaluating the auxiliary functions required in this program, the modified Hankel functions of order $1/3$.
4. A mode follower program given here is useful in tracing eigenvalues. Such traces are needed to understand the eigenvalue structure.

RECOMMENDATIONS

1. Improve the mode locating ability of this normal-mode program to make it self-contained. It currently requires user interaction to locate eigenvalues.
2. Investigate methods to incorporate the effect of rough boundaries into this program.

79 08 1 024
1

CONTENTS

INTRODUCTION . . .	page 3
GENERAL SOLUTION . . .	4
DETERMINANT . . .	8
FINDING EIGENVALUES . . .	10
Control cards . . .	10
Iteration termination . . .	11
SOUND SPEED PROFILE . . .	11
NUMERICAL BREAKDOWN . . .	15
Program modifications . . .	18
Preventing zeroes in the determinant . . .	19
REFLECTION COEFFICIENTS AND OTHER AUXILIARY OUTPUTS . . .	20
COMPUTATION OF THE MODIFIED HANKEL FUNCTIONS . . .	22
Power series expansion . . .	23
Asymptotic series expansion using continued fractions . . .	25
Comparative accuracy . . .	34
MODE FOLLOWER PROGRAM . . .	36
Implementation of the mode follower . . .	38
Input and output . . .	40
CONCLUSIONS . . .	41
RECOMMENDATIONS . . .	41
REFERENCES . . .	41
APPENDIX A: NORMAL MODE PROGRAM IN FORTRAN . . .	43
APPENDIX B: SAMPLE RUN . . .	74
APPENDIX C: HANKEL FUNCTION PARAMETERS . . .	89
APPENDIX D: MODE FOLLOWER PROGRAM IN FORTRAN . . .	93

INTRODUCTION

This report describes a normal-mode program that has been used successfully for 12 years to compute sound propagation in idealized underwater acoustic ducts. The theory and considerations used in developing the program are discussed here, and a copy of the FORTRAN statements are included as appendix A. Appendix B consists of sample inputs and outputs to assist users in gaining familiarity with the program. It is hoped that this report contains sufficient information to allow a user to run the program and to modify it as desired.

This program follows the methods developed by Furry and Freehoffer (ref 1) to compute electromagnetic propagation in the 1940s. Marsh adapted these methods to underwater sound in his doctoral thesis (ref 2). Using this material, Pedersen, at NOSC in the late 1950s, adapted the method to digital computers and developed the programs to compute the auxiliary functions. This original program used two layers to define the sound-speed profile (ref 3). This program was expanded to three layers by DF Gordon and RF Hosmer and finally to the multiple-layer program reported here. In this program the only constraints on the number of layers are computer space and running time. The program is normally configured to permit up to 12 layers.

The earlier programs were used to study sound propagation in ocean surface ducts. Programs that permit more layers have proven useful also for studying propagation in the deep ocean, although the number of modes required generally limits computations to frequencies below 300 Hz. The multiple-layer program has also proven useful in modeling sediment layers and thus in computing shallow-water propagation.

The principal limitation in the application of this program to real-world situations is the requirement of ideal conditions: boundaries must be smooth and horizontal, and no variation of boundary conditions with range is permitted. Despite this limitation, the program has proven useful in predicting and explaining acoustic propagation and has applications in a number of related areas. These include checking other types of wave-theory models or corrections such as caustic corrections; determining group velocities, dispersion curves, and reflection coefficients; and determining acoustic coupling between ducts.

The following paragraphs describe the specific topics covered by the sections in this report. In GENERAL SOLUTION are the equations required to solve the wave equation with the boundary conditions used here. DETERMINANT is part of the basic solution but is concerned with the particular numerical method used in this program to evaluate the conditions imposed by the boundaries. Other approaches could be used instead. A later section, NUMERICAL BREAKDOWN, is also part of the basic solution, but deals with special numerical problems that have arisen but are not apparent from the basic equations.

1. The Bilinear Modified-Index Profile, by WH Furry, in Propagation of Short Radio Waves, DE Kerr, ed; MIT Rad Lab series, vol 13, p 140-168, McGraw-Hill, New York, 1951.
2. Navy Underwater Sound Laboratory Report 111, Theory of the Anomalous Propagation of Acoustic Waves in the Ocean, by HW Marsh, 1950.
3. Normal-Mode Theory Applied to Short-Range Propagation in an Underwater Acoustic Surface Duct, by MA Pedersen and DF Gordon; J Acoust Soc Am, vol 37, p 105-118, January 1965.

FINDING EIGENVALUES deals with the philosophy of eigenvalue location employed by this program, which essentially leaves this function to the user, the program only serving as a tool. It shows how the program is used to make computations.

Several "automatic" mode finding versions of this program have been developed to the point of accommodating certain classes of profiles. However, they need further development and have not yet been reported.

SOUND SPEED PROFILE indicates the required equations for curve fitting and the various ways the sound speed can be read in on cards. A continuous water profile can be entered quite simply, but sediment layers with sound speed discontinuities and absorption gradients can become complicated.

REFLECTION COEFFICIENTS AND OTHER AUXILIARY OUTPUTS describes a short subroutine that computes reflection coefficients for any mode at a given profile interface. Intermode interference lengths and mode damping coefficients are also discussed.

COMPUTATION OF THE MODIFIED HANKEL FUNCTIONS gives the analysis necessary for computing these functions. The use of continued fractions to evaluate an asymptotic series is discussed. To facilitate running the program on computers of different word length, this section provides the information required to optimize the functions for the different word lengths.

MODE FOLLOWER PROGRAM describes a separate but related program for investigating the eigenvalues themselves rather than using them to compute propagation losses.

GENERAL SOLUTION

The derivation of the normal-mode solution has been discussed from various points of view (eg ref 1, 4, 5). Only an outline is given here. In general, the time-independent wave equation is written in polar coordinates and the azimuthal coordinate is dropped under the assumption that the field is independent of azimuthal direction. Thus

$$(1/r) (\partial/\partial r) [r(\partial\psi/\partial r)] + (\partial^2\psi/\partial z^2) + (\omega^2/c^2) \psi = 0, \quad (1)$$

where ψ is the velocity potential, c the sound speed, and the independent variables are depth, z , and range, r .

Equation (1) is then separated into range- and depth-dependent parts with a separation constant λ . The separation is possible when the sound speed is a function of depth only. After accounting for the source discontinuity and the outgoing radiation condition, integrating over all real values of the separation constant, and normalizing, one can find the solution for a field point in terms of propagation loss H as follows:

4. Naval Air Development Center Report NADC-72002-AE, Normal Mode Solutions and Computer Programs for Underwater Sound Propagation, by CL Bartberger and LL Ackler, 4 April 1973.
5. A Normal Mode Theory of an Underwater Acoustic Duct by Means of Green's Functions, by RL Deavenport; Radio Sci, vol 1, p 709-724, 1966.

$$H = -10 \log \left| \rho_s \rho_h \pi \sum_{n=1}^N H_0^2(\lambda_n r) U_n(z) U_n(z_0) \right|^2 + \alpha_A r, \quad (2)$$

where r is the range, z_0 is the source depth, z is the receiver depth, H_0^2 is the Hankel function of order zero, second type, λ_n is the n th eigenvalue, U_n is the depth function for mode n , and ρ_s and ρ_h are the densities at source and receiver. The sum is over the number of modes, N , making a significant contribution. The final term contains the volume attenuation coefficient, α_A . From Thorp (ref 6), α_A in dB/m is computed by the relationship

$$0.9144 \alpha_A = 0.0001 F^2 / (1 + F^2) + 0.04 F^2 / (4100 + F^2), \quad (3)$$

where F is the frequency in kHz. Improved equations or those for specific ocean areas can be easily substituted. The depth function, U_n , is a solution to the depth-dependent part of the separated wave equation

$$d^2 U / dz^2 + [\omega^2 / c^2(z) - \lambda^2] U = 0, \quad (4)$$

where

$$\omega = 2\pi f$$

and f is the frequency, in Hz.

A closed-form solution to eq (4) can be obtained when the reciprocal sound speed squared or squared index of refraction is a linear function of depth. That form is used in this program, and sound speed in each layer is expressed as follows:

$$[c_i / c(z)]^2 = 1 - 2 \gamma_i (z - z_i) / c_i, \quad (5)$$

where c_i , z_i , and γ_i are the sound speed, depth, and sound-speed gradient, respectively, at the top of layer i . Up to 12 such layers are permitted by the program, for modeling the sound-speed profile.

With this expression for sound speed, solutions to eq (4) can be expressed in terms of solutions to Stokes' equation

$$h'' + zh = 0. \quad (6)$$

Only a simple change in independent variable is required from z to ξ , where

$$\xi_i(z) = \left[a_i^3 (z - z_i) + \omega^2 / c_i^2 - \lambda^2 \right] / a_i^2 \quad (7)$$

and

$$a_i^3 = -2\gamma_i \omega^2 / c_i^3. \quad (8)$$

6. Analytic Description of the Low-Frequency Attenuation Coefficient, by WH Thorp; J Acoust Soc Am, vol 42, p 270, 1967.

The solutions to Stokes' equation that are used are the modified Hankel functions of order $1/3$, $h_1(\xi)$ and $h_2(\xi)$. The depth function is a linear combination of these two independent solutions:

$$F_{n,i}(z) = A_{n,i}h_1(\xi_n) + B_{n,i}h_2(\xi_n), \quad (9)$$

where F_n is the unnormalized form of U_n . The coefficients $a_{n,i}$ and $B_{n,i}$ for mode n in layer i are determined to satisfy boundary conditions, which will be listed below. Values of λ_n for which the boundary conditions can be satisfied are the eigenvalues.

The first boundary condition is the radiation condition. It is satisfied by using a negative sound-speed gradient in the deepest layer, which extends to infinite depth, and by letting the depth function there be proportional to h_2 only. That is,

$$F_n(z) = B_n h_2(\xi_n). \quad (10)$$

At the surface the depth function is zero:

$$F_n(0) = 0, \quad (11)$$

and at layer interfaces, ρU and its depth derivative are continuous:

$$\rho_i F_{n,i}(z) = \rho_{i+1} F_{n,i+1}(z); \quad (12)$$

$$dF_{n,i}(z)/dz = dF_{n,i+1}(z)/dz. \quad (13)$$

Here ρ_i is the density in layer i , and the excess acoustic pressure, p , is given by

$$p = \rho U.$$

If U is assumed to be the vertical component of the velocity potential, eq (12) and (13) are equivalent to requiring that the pressure and the vertical component of particle velocity be continuous across the layer interface.

Applying these boundary conditions to a sound-speed profile consisting of M layers results in $2M - 1$ linear equations in h_1 and h_2 . They are homogeneous in that the constant is zero in each equation. There are $M-1$ coefficients A_i to be determined and M coefficients B_i . These coefficients can therefore be determined within a constant of proportionality D , provided the system of equations is linearly dependent. That is, the $2M - 1$ square matrix of coefficients of A_i and B_i must be of rank $2M - 2$ or less. Its determinant will then be zero. This is the eigenvalue condition. Values of λ must be found which make the determinant zero. This determinant, G , is discussed in more detail in a later section.

Zeros of the determinant, G , are found by using the secant method. The variable in this iterative method can as well be some function of λ as λ itself, and we use the following complex phase velocity (v):

$$\lambda_n = \omega/v_n. \quad (14)$$

To find a v that is a root of G requires an initial guess, v_1 , where the subscript 1 refers to the step in the iteration and a small increment, δ_1 . Each succeeding estimate is given by the relationship

$$v_{j+1} = v_j + \delta_j,$$

where

$$\delta_j = -(v_j - v_{j-1}) G_j / (G_j - G_{j-1}). \quad (15)$$

The details of this iterative process are given in a later section.

When an eigenvalue v_n is found, the coefficients are then evaluated. One coefficient can be given an arbitrary value, so A_1 is set to $\rho_1 h_2 [\xi_1(0)]$. From eq (11), B_1 is then $-\rho_1 h_1 [\xi_1(0)]$. Pairs of equations (eg (12) and (13)) for each successive interface can then be used to evaluate the next A_i and B_i as discussed later.

Finally the normalizing factor, D_n , for mode n is obtained by the relationship

$$D_n = \int_0^{\infty} \rho F_n^2(\xi) dz. \quad (16)$$

This equation follows from the orthogonality of the depth functions. It is not the pressure, however, which is proportional to ρU , but $\rho^{1/2} U$ that is orthogonal (ref 7). Therefore, D_n must be determined such that the integral of ρU^2 is 1.

From Stokes' equation (eq (6)) and eq (7-9), the integral of F^2 takes the form

$$\int_{z_i}^{z_{i+1}} F^2(\xi) dz = \left[\xi_i(z) F^2(\xi) / a_i + F'^2(\xi) / a_i^3 \right] \Big|_{z_i}^{z_{i+1}}. \quad (17)$$

Therefore

$$D_n = -\rho_1^3 W^2 / a_1 + \sum_{i=1}^{n-1} \left\{ \rho_i [\xi_i(z_{i+1}) / a_i - \rho_i \xi_{i+1}(z_{i+1}) / (a_{i+1} \rho_{i+1})] F_i^2(z_{i+1}) + \left(\rho_i / a_i^3 - \rho_{i+1} / a_{i+1}^3 \right) F_i'^2(z_{i+1}) \right\}, \quad (18)$$

where eq (12) and (13) have been used to combine terms at each interface. The derivative of F takes the form

$$F_i'(z_{i+1}) = a_i \left\{ A_i h_1' [\xi_i(z_{i+1})] + B_i h_2' [\xi_i(z_{i+1})] \right\}. \quad (19)$$

The Wronskian, W , is an imaginary constant (see eq (85)) and is the contribution of eq (17) at the surface:

$$W = -1.45749544104i.$$

7. Some Effects of Velocity Structure on Low-Frequency Propagation in Shallow Water, by AO Williams; J Acoust Soc Am, vol 32, p 363-365, March 1960.

The depth functions are normalized by the relationship

$$U_n(z_0) U_n(z) = F_n(z_0) F_n(z) / D_n. \quad (20)$$

The functions F and F' used in computing D_n are conveniently assembled from the elements of the determinant and the coefficients A_i and B_i . This requires care in developing the computer code, because F is always multiplied by ρ and F' has the term a_i in it. The surface differs from the other layers in that F_1 is zero there and F'_1 , by eq (19), is $a_1 W$. However, because ρ_1 appears as a factor in the coefficients of F_1 , the actual value of F'_1 at the surface in the computation is $\rho_1 a_1 W$. This factor of ρ_1 together with the $\rho_1^{1/2}$ needed for orthogonality, when squared, gives the ρ_1^3 of eq (18).

DETERMINANT

Normal modes are determined by finding the eigenvalues of a characteristic equation which, in turn, is obtained by setting a determinant to zero. The determinant is obtained from the coefficient matrix of a set of linear, homogeneous equations expressing the boundary conditions as given by eq (10) – (13). Since the method of handling this determinant is a central feature of this normal-mode program, it is given in detail here.

The first line of the matrix is taken from eq (11) as

$$B_1 \rho_1 h_2 [\xi_1(0)] + A_1 \rho_1 h_1 [\xi_1(0)] = 0. \quad (21)$$

At each profile interface, i , where i numbers the interfaces below the surface from 1 to $N-1$, the two boundary conditions given by eq (12) and (13) are

$$\begin{aligned} B_i \rho_i h_2 [\xi_i(z_{i+1})] + A_i \rho_i h_1 [\xi_i(z_{i+1})] - B_{i+1} \rho_{i+1} h_2 [\xi_{i+1}(z_{i+1})] \\ - A_{i+1} \rho_{i+1} h_1 [\xi_{i+1}(z_{i+1})] = 0 \end{aligned} \quad (22)$$

and

$$\begin{aligned} B_i a_i h'_2 [\xi_i(z_{i+1})] + A_i a_i h'_1 [\xi_i(z_{i+1})] - B_{i+1} a_{i+1} h'_2 [\xi_{i+1}(z_{i+1})] \\ - A_{i+1} a_{i+1} h'_1 [\xi_{i+1}(z_{i+1})] = 0. \end{aligned} \quad (23)$$

The coefficients of A_i in the first equation and B_{i+1} in the second will be the diagonal elements of the matrix. The nonzero elements of the matrix will therefore be no more than two places from the diagonal. The matrix can be stored in the computer in an array of size $(2M-1) \times 4$, where M is the maximum number of layers in the sound-speed profile. In the final layer, $A_N h_1$ is omitted, as in eq (10). In the program, the real and imaginary parts are stored in separate arrays.

The sparseness of the matrix permits efficient evaluation by a triangularization process of row reduction. For each pair of rows representing a pair of equations given by eq (22) and (23), the first element from the first equation and the first two from the second equation must be set to zero by subtracting the proper multiple of preceding rows. The determinant is then the product of the diagonal elements of the triangularized matrix. The value of the determinant, G , is used in eq (15) to find the roots by iteration.

Note that a value of v that makes this determinant zero, or near zero, ordinarily is zero because only one diagonal element is very small. For trapped modes this element is at the row representing the first interface below the mode, ie the interface just below the layer of positive gradient in which the sound speed is equal to the mode phase velocity. For untrapped modes it is usually the final diagonal element that is small. Thus the layers in which the sound speed is greater than the phase velocity of a mode do not greatly affect the eigenvalue. Eigenvalues are determined mainly by those parts of the sound-speed profile that are less than the phase velocity.

When an eigenvalue is found, the coefficients A_i and B_i must next be evaluated. As mentioned earlier, one coefficient can be arbitrarily chosen. This is done, and eq (21) is satisfied by letting

$$A_1 = \rho_1 h_2 [\zeta_1(0)]$$

and

$$B_1 = -\rho_1 h_1 [\zeta_1(0)]. \quad (24)$$

The factor ρ_1 is used simply because the number containing it is easily available in the program. It is divided out by the normalizing factor, D . Eq (22) and (23) can then be used to evaluate the remaining coefficients, but the triangularized form of the matrix yields the coefficients with less computation. If g_{ij} is the element in the i th row and j th column of the triangularized matrix, then by Cramer's rule,

$$B_i = A_{i-1} g_{2i-2, 2i-2} g_{2i-1, 2i} / E_i$$

and

$$A_i = -A_{i-1} g_{2i-2, 2i-2} g_{2i-1, 2i-1} / E_i,$$

where

$$E_i = g_{2i-2, 2i-1} g_{2i-1, 2i} - g_{2i-2, 2i} g_{2i-1, 2i-1}. \quad (25)$$

A simpler form is used for B_N in the final layer since there is no A_N there.

In certain situations numerical problems can arise in evaluating the determinant. These require some extra tests in the subroutine that makes the evaluation. The extra tests will be discussed in the section, **NUMERICAL BREAKDOWN**. A more routine problem is the loss of accuracy that can arise in subtractions in the row reduction of the matrix. This loss results in less sharpness of convergence to a root. The size of the determinant, G , can be 14 orders of magnitude less at a root than at the general background near the root. This variation occurs because the modified Hankel functions can be computed to about 14-place accuracy in a computer with 18 decimal places available. Modes usually converge to 10 or 12 places; thus a few places are lost in evaluating the determinant. In some profiles, usually those with multiple ducts or those in which propagation through bottom sediments plays a large part, the convergence can be much poorer. Modes need to converge to about 4 places to be reliable for computing losses, and convergence occasionally fails to meet this requirement. The only current cure for this loss in accuracy is to go to higher-precision arithmetic or to compute the modified Hankel functions to greater accuracy. For instance, a standard

matrix triangularization routine that uses full row and column pivoting has been tried with no resultant increase in accuracy.

FINDING EIGENVALUES

There are versions of this program under development that will locate the eigenvalues and do the entire computation without user intervention. Currently, however, these versions are reliable only for the simpler types of profiles – usually those with only one duct – and are not ready to be reported. Locating eigenvalues with the standard version of the program is discussed here.

The standard version of the program requires the user to find the eigenvalues. In this version, each time an eigenvalue is determined by iteration, the resulting value is stored and counted as an eigenvalue. Therefore, the user must ensure that all iterations result in good roots, that all required modes have been determined, and that no modes are present more than once. In most cases the user must expect to make more than one computer run to obtain this result.

CONTROL CARDS

The user controls the eigenvalue determination by using any of four different types of control cards. The first type specifies an initial value for v and an initial step size, Δv . These are both complex numbers with a real and an imaginary part. G is then evaluated at v and at $v + \Delta v$ to start the iteration. These are essentially the v_j and v_{j+1} of eq (15). If these two trial eigenvalues are in the vicinity of a root, the iteration will converge to that root.

The second type of card specifies a line segment in the complex plane, along which a search for eigenvalues, v , is made. The end points of the line are given along with the number of equally spaced points at which the line is to be divided. G is then evaluated at each successive division point along the line until a relative minimum in $|G^2|$ is found, indicating that a root is nearby. The iterative process is applied to find the root. The initial step size, Δv , is first computed to bring the second evaluation at $v + \Delta v$ as close as possible to the true root. This is done by using the point which resulted in minimum $|G^2|$ and the points on either side of it to determine the minimum of the parabola passing through them. If $v - h$, v , and $v + h$ are the three points at which G was evaluated, it follows that the distance from v to the minimum of the parabola

$$\Delta v = h[G(v + h) - G(v - h)] / 2[2G(v) - G(v + h) - G(v - h)]. \quad (26)$$

When the iteration is complete, the eigenvalue is recorded and the program continues to step along in the direction of the given line, checking again for a minimum. However, the stepping is resumed from the newly located root rather than from the approximate location where the minimum was detected. With this correction in position, the designated line does not have to hug the curve on which the eigenvalues are located because it is corrected at each eigenvalue.

This method of finding eigenvalues has proven very successful. Its main utility arises, though, because the eigenvalues of the trapped modes have negligible imaginary parts and the

search can be made along the real line. In simple profiles this can often give a successful set of modes on the first try. Usually, only the three initial eigenvalues need to be located by this means because further eigenvalues can be located by extrapolation on the previous three. This is the function of the third type of control card.

The third type of card specifies the number of additional modes to be determined by extrapolation. The starting value of each eigenvalue is determined by extrapolating from the three most recently determined eigenvalues to find v . The step size, Δv , is chosen as 0.0001 times the distance between the last two eigenvalues. The exact eigenvalue is then determined by iteration. The extrapolation is the simple parabolic form for equal steps:

$$v = 3v_n - 3v_{n-1} + v_{n-2}. \quad (27)$$

This method of locating modes works well when the modes lie along a smooth curve, as usually occurs for single ducts. But this relationship does not always occur for profiles with multiple ducts.

The final control card is punched by the program when requested and contains the correct eigenvalue to full precision. Upon encountering this card, the program does not iterate, but instead evaluates G for this eigenvalue and stores this value of G as the next eigenvalue. A deck of such completed eigenvalues can be stored, saving the expense of recomputing the eigenvalues for a given profile and frequency.

ITERATION TERMINATION

A full description of the iteration of eq (15) should include the method of termination. The usual criterion for stopping is that G fails to become smaller. As G approaches minimum size, however, round-off error can act as noise so that G is no longer a predictable function of v . The denominator of eq (15) can then be very small by chance, resulting in a large value for δ_i . If this happens, the next value of v , which was as near to the root as possible, will be far away. A much better convergence criterion is that δ_i has reached a minimum in absolute value. In the program, iteration is stopped when $|\delta^2|$ exceeds the previous value by a factor of 2. However, this criterion is not applied until three iterative steps have been completed, to permit the process to become well established. An upper limit of 15 iterative steps is permitted. We have not found an improvement on the root after 15 steps.

SOUND SPEED PROFILE

The normal mode program requires as inputs the depth of each layer and the sound speed and sound speed gradient at the top of each layer. These variables are mapped into the dimensionless internal variables of the program by eq (7). The purpose of the sound speed profile processing portion of the program is to accept the profile parameters in a form convenient for the user and to translate them into the required sound speeds and gradients.

The first function of the processing program is to make the sound speed continuous at interfaces. This is done simply by using the sound speed at the bottom of one layer as the sound speed at the top of the next. It may be necessary to compute the sound speed at the bottom of the layer. The necessary parameters will have been given. Occasionally a

discontinuity in sound speed is required, as when modeling an interface between water and sediment. The user indicates this by specifying the sound speed at the top of the layer. If left blank, the program provides the sound speed necessary for continuity.

A second function of the processing program is to permit a layer to be defined by the sound speed at top and bottom of the layer rather than by one sound speed and one gradient. Note that the profile form as given by eq (5) is a two-parameter curve.

The last layer extends to infinite depth, so a gradient must be specified at the top of it. However, this gradient can be specified by giving a depth and sound speed point below the last layer. The program handles this by checking to see if the gradient of the last given layer is unspecified. If it is, the number of layers is reduced by one, which causes the last layer to be only the required extra point determining the final gradient. This final gradient must always be negative, as is required by the boundary conditions. The program user must ensure that this gradient is negative and that no gradient is zero. A zero gradient will appear in the denominator of eq (7).

These functions of the profile processing program are relatively simple, but an additional capability used to model sediment bottoms greatly increases the complexity of the program. The capability required is to specify the absorption in a layer by adding an imaginary part to the sound speed. In older versions of this normal mode program an imaginary part, expressed as an absorption coefficient, could be added to the sound speed at the top of the layer. This imaginary part is small compared to the real part. Since the gradient was assumed real at the top of the layer, the imaginary part was initially not changing with depth and it usually changed only a minor amount through the depth of the layer. However, this small change could not always be relied upon. Also Hamilton (ref 8) has published data on absorption gradients in sediment layers, so more precise control of this part of the sound speed function is needed to model sediment layers. Therefore, a more comprehensive profile processing routine has been incorporated in the normal mode program. This curve-fitting process is described below.

The following quantities can be input for each layer depth starting at the surface:

Depth of top of the layer

Sound speed at top of layer

Sound speed at bottom of layer

Real part of sound speed gradient at top of layer

Attenuation in loss per km at the top of the layer

A similar attenuation at the bottom of the layer

Density in the layer

The density is a constant in the layer and as such requires no further curve fitting. Redundant parameters are left blank on input cards. In some cases negative values serve as flags to indicate specific treatment. For instance a negative value of absorption at the top of a layer

8. Sound Attenuation as a Function of Depth in the Sea Floor, by EL Hamilton; J Acoust Soc Am, vol 59, p 528-535, March 1976.

directs the program to use the same imaginary part of sound speed as occurred at the bottom of the previous layer. Similar flags at the bottom of a layer are discussed later.

Absorption per Hz is given in units of decibels per km (or kiloyard). The quotient of absorption over frequency is used because Hamilton (ref 8) usually considers absorption (or attenuation) as proportional to frequency with a coefficient k. We use the symbol h instead. That is,

$$\alpha = hf.$$

We interpret α to be in units of dB per km and f in Hz, whereas Hamilton uses dB per m and kHz; but the coefficients h and k remain equal.

The complex wave number in layer i is represented as

$$\begin{aligned} k_i &= \omega/C_i \\ &= \omega \operatorname{Re}C_i^{-1} - i\omega \operatorname{Im}C_i^{-1}. \end{aligned} \quad (28)$$

A plane wave will be attenuated α dB per km if

$$\begin{aligned} \operatorname{Im}k_i &= -\alpha/(20\,000 \log e) \\ &= -\pi A f, \end{aligned} \quad (29)$$

where

$$A = h/(20\,000 \pi \log e).$$

By equating the imaginary part of k_i in eq (28) and (29), the imaginary part of C_i is found to be as follows:

$$\operatorname{Im}C_i = 1/A - [1/A^2 - (\operatorname{Re}C_i)^2]^{1/2}. \quad (30)$$

If α is zero, which is the case usually used in water layers, eq (30) cannot be used; but the imaginary part of C is then simply zero. These two cases are treated separately in the program.

When sound speed is given at the top and bottom of layer i , the imaginary parts of the sound speeds are determined by eq (30) and the only curve fitting task is to determine the gradient γ_i . Solving eq (5) for γ_i ,

$$\gamma_i = C_i(C_{i+1}^2 - C_i^2)/2C_{i+1}^2(z_{i+1} - z_i). \quad (31)$$

The gradient is a complex number since the C 's here are complex. The z 's are real.

A second version of this computation arises if the gradient is required to be a real number. In this case, which is used to match older versions of the program, an additional parameter must be left unspecified and this parameter is $\operatorname{Im} C_{i+1}$. This is equivalent to having the sound absorption at the bottom of the layer unspecified. Therefore, a negative number input for this parameter is used as a flag to call for this particular fitting procedure.

For this situation, given $\text{Re } C_i$, $\text{Im } C_i$, $\text{Re } C_{i+1}$, and making γ_i real, the determination of γ_i and $\text{Im } C_{i+1}$ is not simple. When γ_i is eliminated from the real and imaginary parts of eq (31), a quartic equation in $\text{Im } C_{i+1}$ results. Rather than derive an algebraic solution to this equation, it is solved by iteration under Newton's method. A good first guess at the solution is $\text{Im } C_{i+1} \cong \text{Im } C_i$. Four iterations usually give an accurate root. The equation is

$$\begin{aligned} \text{Im } C_i (\text{Im } C_{i+1})^4 + \left[\text{Im}(C_i)^3 + 2(\text{Re } C_{i+1})^2 \text{Im } C_i \right] (\text{Im } C_{i+1})^2 \\ + 2\text{Re } C_{i+1} \text{Re}(C_i)^3 \text{Im } C_{i+1} \\ + \text{Im } C_i (\text{Re } C_{i+1})^4 - (\text{Re } C_{i+1})^2 \text{Im}(C_i^3) = f(\text{Im } C_{i+1}). \end{aligned} \quad (32)$$

The root is then found:

$$(\text{Im } C_{i+1})_j = (\text{Im } C_{i+1})_{j-1} - f/f'.$$

The gradient, γ , is next given by the relationship

$$\gamma_i = \left\{ \text{Im } C_i \left[(\text{Re } C_{i+1})^2 - (\text{Im } C_{i+1})^2 \right] + 2\text{Re } C_i \text{Re } C_{i+1} \text{Im } C_{i+1} - \text{Im}(C_i^3) \right\} / \left[4 \text{Re } C_{i+1} \text{Im } C_{i+1} (z_{i+1} - z_i) \right]. \quad (33)$$

Because the root of eq (32) may not be exact, $\text{Im } \gamma_i$ may not be exactly zero. This slight error can be transferred to C_{i+1} by using the computed real γ_i to recompute C_{i+1} . This is done in the program by transferring to a portion of the program already designed to do this.

When sound speed and gradient at the top of the layer are given, the parameters required by the program are all given. The sound speed at the bottom of the layer is routinely computed, however, because it may be required to make the next layer continuous. Equation (5) is used to determine the sound speed at depth z_{i+1} , which is the depth of the bottom of the layer. This is straightforward, but several complications arise. Only the real part of the gradient at the top of the layer is used as an input because situations have not arisen that require that the imaginary part of the gradient be specified. Often the attenuation is given at both top and bottom of the layer. That is, $\text{Re } C_i$, $\text{Im } C_i$ and $\text{Re } \gamma_i$ are given, plus a relationship between $\text{Re } C_{i+1}$ and $\text{Im } C_{i+1}$. The imaginary part of the gradient, $\text{Im } \gamma_i$, must be determined as well as both real and imaginary parts of the sound speed at the layer bottom. The derivation of this case is not trivial.

One relationship between the real and imaginary parts of the sound speed is given by eq (28) and (29). From these equations at C_{i+1} we derive

$$A(T - i) = 2/C_{i+1}, \quad (34)$$

where

$$T = \text{Re } C_{i+1} / \text{Im } C_{i+1}.$$

Substituting this expression for C_{i+1} into eq (31) and equating real parts gives a quadratic expression for T which has a usable root of

$$\operatorname{Re}(C_i^3)T = -\operatorname{Im}(C_i^3) - \left\{ [\operatorname{Im}(C_i^3)]^2 + \operatorname{Re}(C_i^3)B \right\}^{1/2}, \quad (35)$$

where

$$B = \operatorname{Re}(C_i^3) - 8 \operatorname{Re} \gamma_i(z_{i+1} - z_i)/A^2 + 4 \operatorname{Re} C_i/A^2.$$

From eq (34),

$$\operatorname{Re} C_{i+1} = 2T/A(T^2 + 1) \quad (36)$$

and

$$\operatorname{Im} C_{i+1} = \operatorname{Re} C_{i+1}/T.$$

The gradient can now be evaluated by eq (31) to find its imaginary part.

Equations (34) and (35) cannot be used if the attenuation at the bottom of the layer is given as zero. Therefore an alternate form must be used. This form is much simpler than the previous case, since C_{i+1} is real.

$$C_{i+1} = \left\{ \operatorname{Re}(C_i^3) / [\operatorname{Re} C_i - 2 \operatorname{Re} \gamma_i(z_{i+1} - z_i)] \right\}^{1/2} \quad (37)$$

$$\operatorname{Im} \gamma_i = [\operatorname{Im} C_i - \operatorname{Im}(C_i^3)/C_{i+1}^2] [2(z_{i+1} - z_i)]^{-1} \quad (38)$$

Finally, if the special case, γ_i real, is specified by inputting a negative value for absorption, eq (31) can be used directly to give

$$C_{i+1}^2 = C_i^3 / [C_i - 2\gamma_i(z_{i+1} - z_i)]. \quad (39)$$

To evaluate the square root, let

$$C_{i+1}^2 = a + bi.$$

Then

$$\operatorname{Re} C_{i+1} = \left\{ \left[a + (a^2 + b^2)^{1/2} \right] / 2 \right\}^{1/2} \quad (40)$$

and

$$\operatorname{Im} C_{i+1} = b/2 \operatorname{Re} C_{i+1}. \quad (41)$$

NUMERICAL BREAKDOWN

A situation arises frequently in which a very small depth function must be computed from the difference of two large numbers. A wrong answer results if this accuracy loss exceeds the word size of the computer. The best way that has been found to avoid this is to check for it within the program and arbitrarily replace the wrong number. In checking for this, a constant, called T-lim in the computer program, is compared to the argument of the

modified Hankel functions or to the argument of the exponential function within modified Hankel functions. A T-lim value of 25.0 is used in the program, but a smaller number occasionally is required. The program user can alter T-lim by appropriate input cards (Key 8 = 1 followed by a new value of T-lim). The next few paragraphs demonstrate the symptoms of this problem, so as to assist a user in recognizing the problem. The remainder of this section describes the modifications that have been made to the computer program to correct this loss of accuracy.

The solid line of figure 1 shows a simple surface duct and the phase velocities of the first three modes at 3 kHz. For this profile, the depth function of mode 1 is shown in figure 2. The solid line is the depth function as computed by a program that does not correct for numerical breakdown. The dashed line shows the correct depth function below a depth of 71 m. This result was determined from Airy functions, not from the program. Between depths of 71 to 100 m, the program cannot compute the depth function accurately. In the second layer, which starts at a depth of 100 m, the function can be computed accurately but it is incorrectly placed by the boundary condition that requires the depth functions to be continuous at interfaces. The slope of the depth function was correctly computed as indicated by the identical shape of the three depth functions in the second layer. The shape is such as to make the correct depth function continuous in slope across the interface.

The breakdown in accuracy at a depth of 71 m occurred when ζ had a value of -8.4 . (ζ is given by eq (7) and is the argument of the modified Hankel functions.) A negative value of ζ occurs when the mode phase velocity is less than the speed of sound. Since the ray of the same phase velocity cannot reach such a region, the sound field there is a diffracted field. The mode depth function is therefore small at such depths. In the figure, the depth function amplitude at the breakdown point is about 7 orders of magnitude (or in terms of propagation loss, 140 dB) down from its maximum. Equations (62), (66), and (68), which will be given for the modified Hankel functions, indicate that the argument of the exponential term is $2/3(8.4)^{3/2}$, or 16.2. The functions h_1 and h_2 will thus be about 10^7 in magnitude at a depth of 71 m. These large values and their small difference account for the approximate accuracy loss of 14 decimal places, which is the general accuracy of the modified Hankel functions.

Incorrect behavior in the depth function usually occurs when ζ is about -8.4 . In some more complicated profiles, however, where accuracy is lost in row reduction of the determinant, the depth functions may become incorrect at values of ζ that are less in absolute value. When this problem occurs it can be diagnosed by plotting the depth function of the mode and noting the steep positive slope through some depth interval as in figure 2. When that occurs, the value of T-lim should be decreased.

Incorrect depth functions can cause errors in propagation loss computations in two ways. In figure 2, the solid-line depth function, because of its large size, can cause losses to be too low at a depth of around 100 m. The second error would occur if the duct were deeper, say 110 m. At this depth the erroneous segment of depth function in figure 2 would reach a value of about 10^{-1} , where it would be larger than the correct lobe of the depth function near the surface. With this extra area under the curve, the normalizing factor would be increased significantly and would reduce the size of this entire depth function. Thus, losses near the surface would be larger because of the loss in size of mode 1.

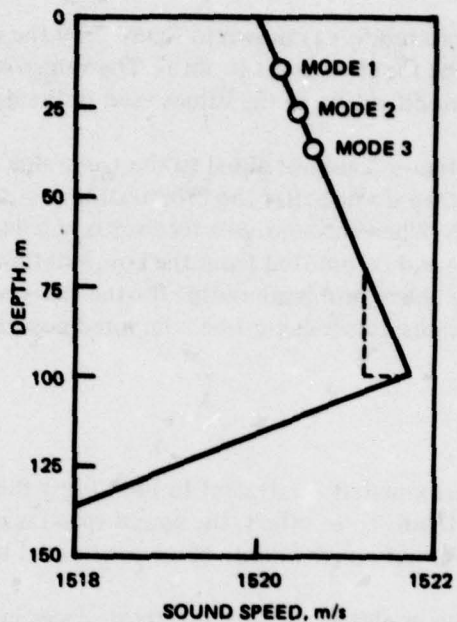


Figure 1. A two-layer sound speed profile for a surface duct. The phase velocities of the first three modes at 3 kHz are marked. The broken line shows a modification of the upper layer to prevent numerical breakdown in mode 1.

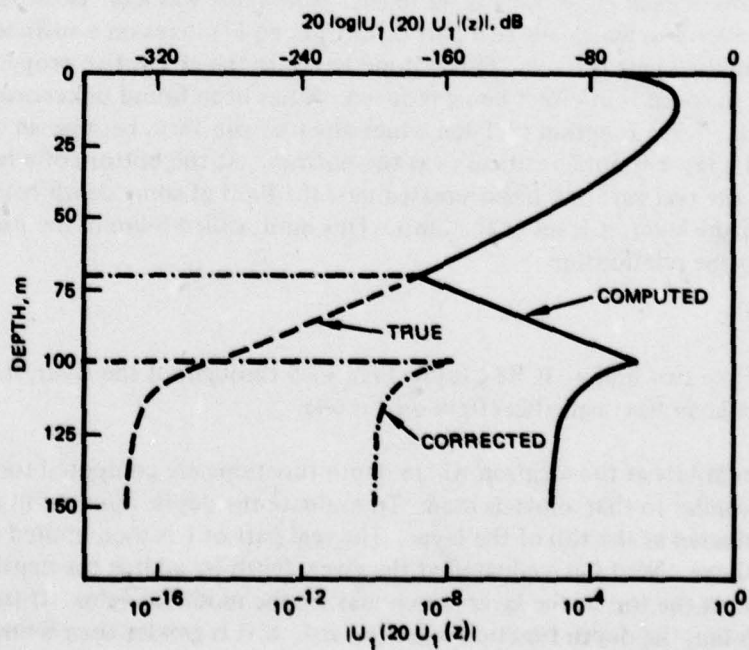


Figure 2. Depth function of mode 1 at 3 kHz, showing error in the computed function. The true function cannot be computed without increasing computer word length, but the corrected value can and it will not cause a large error in the mode sum.

The standard correction to mode 1 is shown in figure 2 by the dot-dashed line. In the depth interval where $\zeta < -8.4$, the function is set to zero. The values of the depth function at greater depths result from a modification in the values used in the determinant.

The corrected values in figure 2 are not equal to the true value of the depth function, but they are small enough that they do not alter the propagation loss to a tenth of a decibel when a full set of modes is used. When the source or receiver is at a depth where such corrections are necessary, the mode can be omitted from the computation. Thus, properly omitting modes would solve the above problems except for the cases where the normalizing factor, D , is affected. In these cases, losses cannot be computed accurately without the corrections.

PROGRAM MODIFICATIONS

The modification is approximately equivalent to modifying the sound speed profile as shown by the broken line in figure 1. In effect, the sound speed is not allowed to become enough greater than the phase velocity of the mode being considered to cause problems.

The limitation on ζ is accomplished at three different places in the normal mode program. It is not clear that this is the best way to handle the problem and it may be redundant, but it appears to be an adequate solution. These three corrections will be described next. Finally a correction to the determinant program is described which is necessary because the limiting of ζ can cause false zeroes in the determinant.

In the subroutine SETUP the elements of the determinant are computed by determining ζ at the top and bottom of each layer and then calling the modified Hankel function program. At the top of each layer, $\text{Re } \zeta$ is set to -7.5 if its value was less. However, this is done in an iterative loop in which the real part of ω/C_i in eq (7) takes on a sufficiently larger value while its imaginary part is fixed. This is done to retain the absorptive properties of a layer when its sound speed is in effect being reduced. It has been found unnecessary to make the above constant, -7.5 , a function of $T\text{-lim}$ which the user can vary, because an oversized value at the top of a layer is not as critical as at the bottom. At the bottom of a layer, several tests are made. If the real part of ζ has decreased past the limit at some depth between the top and bottom of the layer, it is set at the limit. This limit, called $S\text{-lim}$ in the program, is related to $T\text{-lim}$ by the relationship

$$S = -(T)^{2/3} \quad (42)$$

where S and T are the two limits. If $\text{Re } \zeta$ is less than -7.5 throughout the layer, it is simply set at -7.5 . Such a layer has negligible effect on a mode.

In program MAIN at the location where depth functions are computed for given depths, a process similar to that above is used. To evaluate the depth function in a given layer, ζ is first evaluated at the top of the layer. The real part of ζ is then limited as in the program SETUP above. Next ζ is evaluated at the given depth by adding the depth-dependent part onto the value at the top of the layer which may be the modified value. If this final value is less than $S\text{-lim}$, the depth function is set to zero. If it is greater than $S\text{-lim}$, the function is computed in the usual way.

The imaginary part of ζ can be large if the eigenvalue has a large imaginary part or if the speed of sound in the layer has a large imaginary part. When this happens the imaginary part of $2/3 \zeta^{3/2}$, which appears as an exponential in the modified Hankel functions, may become large in absolute value even though $\text{Re } \zeta$ has been limited. A final check is therefore made before the exponential is computed. If $\text{Im } \zeta^{3/2}$ is greater than T-lim, ζ is reduced in amplitude to the size at which it will equal T-lim. The angle of ζ in the complex plane is preserved.

This limitation of the exponential can be viewed in another way. In a following section the two components of the modified Hankel functions, F_1 and F_2 , eq (68) and (69), have exponential terms whose arguments are equal and opposite in sign. When these arguments have magnitude of $2/3$ T-lim, they differ in size by 15 decimal places, which is near the 18-decimal-place word size of the machine. The ability to compute the difference in these two terms is essentially the same as the ability to compute the depth function accurately.

PREVENTING ZEROES IN THE DETERMINANT

Placing limits on ζ can cause problems in the determinant because ζ may be set equal to S-lim at several interfaces. The equations that arise for matching boundary conditions may then be identical for these interfaces and may therefore fail to be linearly independent. The triangularized determinant will thus have zeroes on the diagonal at positions equivalent to interfaces that do not have real physical importance for the mode. These will prevent location of the significant "zeroes" or roots. These artificial zeroes must be removed.

The artificial zeroes are detected and removed in the subroutine DET, which evaluates the determinant. If four elements from the matrix have the configuration

$$\begin{array}{cc} a & b \\ c & d \end{array}$$

and c is to be set to zero by row reduction, d will be replaced by a value, x , as follows:

$$x = d - bc/a.$$

If d is located on the diagonal, complete loss of accuracy is checked for by computing

$$s = |x^2|/|d^2|.$$

If s is less than 10^{-34} , x is not used; instead, d is replaced by $10^{-17}d$. Note that this substitution will occur when x is zero, thus preventing zeroes on the diagonal. The power of ten, -17 , is chosen to be near the total word size of 18 decimal places.

The above substitution prevents sudden jumps in the value of the determinant when all precision is lost at one step in the evaluation. This is important for the mode search routine which detects roots by looking for minima in a series of values of the determinant while one parameter is incremented slowly. A sudden jump will often produce a relative minimum which will be falsely interpreted as a root. At true roots, one or more elements along the diagonal are small, but not as small as those checked for here.

REFLECTION COEFFICIENTS AND OTHER AUXILIARY OUTPUTS

Once the depth functions of a mode have been determined, it is relatively easy to compute reflection coefficients at any interface. Therefore, a subroutine called RCOEF has been added to the program which will compute and print out reflection coefficients if requested by the use of control key 3. If key 3 is set to 1, the reflection coefficients at all interfaces are computed. If set to a number, n , greater than 1, the coefficient is computed at the n th interface only, where the surface is the first interface.

The printout includes the phase as well as the amplitude of the reflection coefficient and the grazing angle. The grazing angle, θ , of the equivalent rays is computed from the mode phase velocity and the sound speed, c , at the bottom of the layer, by Snell's law:

$$\theta = \cos^{-1}(c/v).$$

The grazing angle is computed only if the phase velocity is greater than the sound speed at the interface, since otherwise the equivalent ray does not reach the interface.

The reflection coefficient is derived, following Bucker (ref 9), by assuming that an isospeed layer exists for a small depth just above the interface. In this layer the depth function can be written as

$$f(z) = Ae^{ilz} + Be^{-ilz}, \quad (43)$$

where l , the vertical component of the mode wave number, is given for mode n by

$$l_n^2 = k_i^2 - \lambda_n^2 \quad (44)$$

and

$$k_i = \omega/c_{bi},$$

where c_{bi} is the sound speed at the bottom of layer i . The derivation now consists of identifying A and B as the pressures of the upgoing and downgoing waves at the bottom of the layer; thus the reflection coefficient

$$R = A/B.$$

A and B are evaluated by making f and its derivative at the interface between this small isospeed layer and the regular profile continuous with the normal mode depth functions. The thickness of the isospeed layer is then allowed to approach zero, giving the desired value of R . If F and F' are the normal mode function and its depth derivative at the interface depth defined by eq (9) and (19), the reflection coefficient resulting from the above derivation is as follows:

$$R = (ilF + F')/(ilF - F'). \quad (45)$$

This coefficient is a complex number. Loss per reflection is given by 20 times the log of the absolute value. The phase gives the phase shift that an equivalent ray would

9. Sound Propagation in a Channel with Lossy Boundaries, by HP Bucker; J Acoust Soc Am, vol 48, p 1187-1194, November 1970.

experience upon reflection. Figure 3 is an example of the use of this computation. It shows phase and amplitude of the reflection coefficient in shallow water over a sandy-silt sediment lying over rock. The frequency is 1500 Hz. Reflections are given only at discrete points determined by the individual modes.

The model in figure 3 is for a liquid bottom. That is, no rigidity is supplied in this program and the sound speed, density, and attenuation determine the reflection coefficients.

The reflection coefficients computed by eq (45) can be closely approximated by dividing the mode attenuation by the loop length of the corresponding ray. The loop length must be determined from ray theory for the ray of the same phase velocity or vertexing velocity. However, an interesting analog of the ray loop length is the intermode interference length. This is discussed by Guthrie (ref 10). Specifically, if the difference between eigenvalues, $\text{Re } \lambda_i$, for two adjacent modes is $\Delta\lambda$, the interference length $l = 2\pi/\Delta\lambda$. This distance will usually equal the ray loop length for some ray with phase velocity between that of the two modes.

As each mode after the first is computed, the length, l , is computed and printed out. Also routinely printed out for each mode is the mode damping or mode attenuation coefficient, in units of dB per km. This attenuation, α_i , is computed from the relationship

$$\begin{aligned}\alpha_i &= -1000 \text{Im } \lambda_i \log_{10} e \\ &= -8686 \text{Im } \lambda_i.\end{aligned}$$

This quantity multiplied by range gives the damping of mode i , in dB.

10. The Connection Between Normal Modes and Rays in Underwater Acoustics, by KM Guthrie; J of Sound and Vibration, vol 32, no 2, p 289-293, 1974.

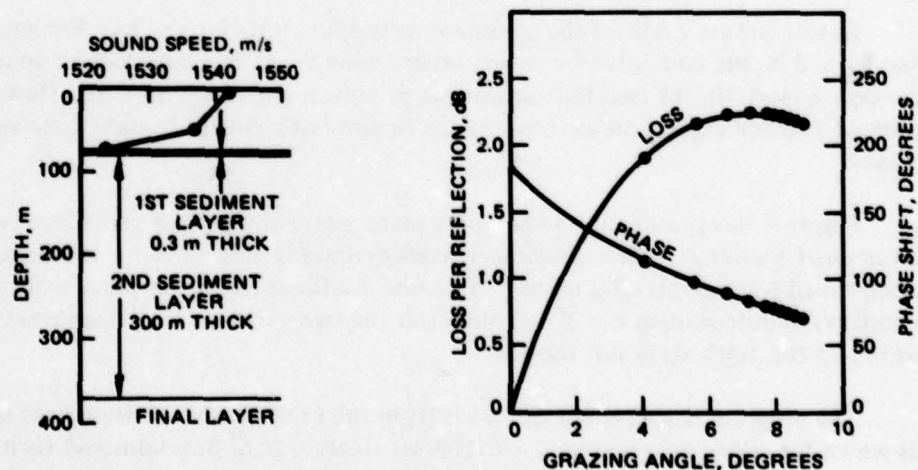


Figure 3. A shallow-water profile with resulting phase and amplitude of the reflection coefficient at 1.5 kHz. Parameters at the top of the sediment layers are as follows: 1st layer - $c = 1606.45$ m/s, $\gamma = 1.5s^{-1}$, $\alpha = 0.18$ dB/m, $\rho = 1.68$; 2nd layer - $c = 1684.0$ m/s, $\gamma = 1.5s^{-1}$, $\alpha = 1.10$ dB/m, $\rho = 1.91$; final layer - $\gamma = -0.1$.

COMPUTATION OF THE MODIFIED HANKEL FUNCTIONS

Most of the computer time required to determine eigenvalues and compute depth functions is spent in evaluating the modified Hankel functions of order $1/3$. For this reason, minimizing computer time in evaluating these functions is desirable. Gaining as many places of accuracy as possible is even more important. The average normal mode computation will have many modes that can be determined to far greater accuracy than is required to obtain 0.1 dB accuracy in the propagation loss. However, there are usually some and often many modes in which many places of accuracy are lost in evaluating the determinant. Therefore, maximum accuracy in the modified Hankel functions is required to extend the range of cases for which computations can be carried out successfully.

Optimization of the program is a function of the computer word length. The program given in this report is for the UNIVAC 1110 with 60 bits word length in double precision or 18.1 decimal places. This section gives the equations and computational techniques that are required to optimize this program for different computer word lengths. Complete details of the functions are given in reference 11.

The Airy functions $Ai(Z)$ and $Bi(Z)$ can be used instead of the modified Hankel functions h_1 and h_2 . However, since h_2 is ideally suited to matching the boundary conditions at great depth as formulated in this normal mode program, h_1 and h_2 are used here. The relationship between them is as follows:

$$h_1(z) = k [Ai(-z) - i Bi(-z)] \quad (46)$$

$$h_2(z) = k^* [Ai(-z) + i Bi(-z)] \quad (47)$$

where

$$k = (3/2)^{2/3} (1 - i\sqrt{3}/3), \text{ and } k^* \text{ is the complex conjugate of } k.$$

In this section z will be the argument of the functions h_1 and h_2 . For small values of $|z|$, h_1 and h_2 are computed by power series expansions. For large values, an asymptotic expansion is used. In the past the asymptotic series was expanded directly. However, a continued fraction expansion has been found to give both shorter running time and better accuracy.

Figure 4 shows a line in the complex plane which divides the plane into two parts. For values of z within the line, the power series method is used. When z is outside the line, the continued fraction method is used. This line is a function of computer word length, and the method of determining it will be given after the two methods have been treated. The accuracy of the methods is also treated.

The program has a parameter called IH in the FORTRAN call statement which controls which functions are computed. If IH is set to zero, both functions and their derivatives are computed. If IH is set to 1, only the functions are computed. If it is set to 2, only h_2 and its derivative are computed.

11. Tables of the Modified Hankel Functions of Order One-Third and their Derivatives, Harvard University Computation Laboratory; Harvard University Press, Cambridge MA, 1945.

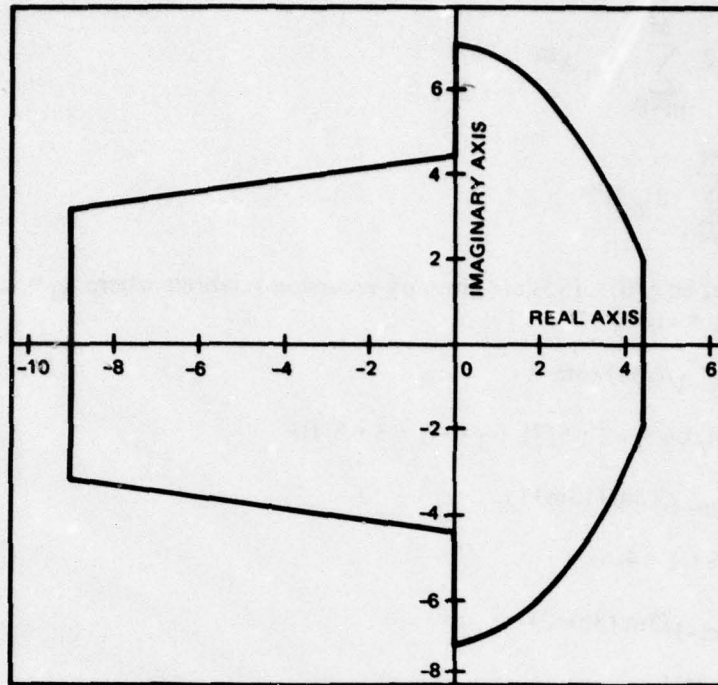


Figure 4. Line in complex plane dividing the arguments for which the modified Hankel functions are computed by (1) power series (inside) and (2) asymptotic expansion evaluated by continued fractions (outside).

POWER SERIES EXPANSION

In this expansion h_1 and h_2 are given by two auxiliary functions f and g as

$$h_1(z) = g + i(3)^{-1/2}(g - 2f) \quad (48)$$

$$h_2(z) = g - i(3)^{-1/2}(g - 2f) . \quad (49)$$

The auxiliary functions are given by the expressions

$$f = A \sum_{m=0}^M a_m X^m \quad (50)$$

$$g = Bz \sum_{m=0}^M b_m X^m , \quad (51)$$

where $X = z^3$, $A = 2^{1/3}/[\Gamma(2/3)]$ and $B = 2^{1/3}/[3^{2/3} \Gamma(4/3)]$. The derivatives $h'_1(z)$ and $h'_2(z)$ can be derived by straightforward differentiation of eq (50) and (51) to give

$$f' = -A z^2 \sum_{m=0}^M c_m X^m \quad (52)$$

$$g' = B \sum_{m=0}^M d_m X^m \quad (53)$$

The coefficients of eq (50) - (53) are given by recursion relations where $a_0 = 1$, $a_1 = 1/3!$, $a_2 = +1 \cdot 4/6!$, $a_3 = -1 \cdot 4 \cdot 7/9!$

$$a_m = -a_{m-1}/(3m)(3m-1) \quad (54)$$

$$b_0 = 1, b_1 = -2/4!, b_2 = +2 \cdot 5/7!, b_3 = -2 \cdot 5 \cdot 8/10!$$

$$b_m = -b_{m-1}/(3m)(3m+1) \quad (55)$$

$$c_0 = 3/3!, c_1 = -6 \cdot 1 \cdot 4/6!$$

$$c_m = -c_{m-1}/3m(3m+2) \quad (56)$$

$$d_0 = 1, d_1 = -4 \cdot 2/4!$$

$$d_m = -d_{m-1}/3m(3m-2) \quad (57)$$

It is important for efficient computation that the number of terms M be no larger than necessary. In the current program the same value of M is used in all four sums. This is done because the optimum number never differs by more than one in the four cases and the determination by table look-up of four M 's often would take longer than computing any unnecessary terms. M for each series is determined so that adding additional terms will not change the answer. Then the most stringent of the four conditions is tabulated and used.

A precise determination of the number of terms to use requires a knowledge of the size of the largest single term in the sum. When a term is smaller than this by a factor which is the power of 10 equal to the number of decimal places in the computer word size, it cannot affect the sum. We ignore the fact that a sum of small terms might be significant. This, then, defines the truncation point. Let m be the number of the largest term in the sum, k the number of terms to be used, and h the number of decimal digits in the machine word. Then for a given k , the largest absolute value of the argument z that can be used to compute g' is given as

$$|z^3|^m d_m = |z^3|^k d_k \cdot 10^h \quad (58)$$

The power of ten can be replaced by 2 raised to a power of the number of binary bits in the computer word if preferred. The coefficient d of eq (57) is used. Each of the other three should also be tried, to find the smallest number of the group for a given k . Equation (58) can be solved for $|z|$, giving

$$\log |z| = (\log d_k - \log d_m + h)/(3m - 3k) \quad (59)$$

A simple computer program given in appendix C will find $|z|$ for each value of k from 1 up to the maximum number of terms desired. The largest term, m , is easily determined because from one k to the next m will remain the same or increase by 1, so it is only necessary at each step to check term $m+1$ to see if it is larger than term m .

The FORTRAN subroutine HANKEL given in appendix A uses the above power series method to compute h_1 and h_2 for small arguments. The coefficients a , b , c , and d are given in lists by that name. The truncation points are given in the list called ZMLA2, which lists values of $|z|^2$ determined by eq (59) or the three similar equations.

ASYMPTOTIC SERIES EXPANSION USING CONTINUED FRACTIONS

When the argument z falls outside the curve in figure 4, h_1 and h_2 can be computed more efficiently or more accurately by asymptotic series than by power series methods. Reference 11 gives information on branch cuts and regions of validity of the two forms of the asymptotic solution (Stokes' phenomenon). Here we will give computing formulas that comply with these requirements, without discussing them further.

Since a given expansion is valid in one or more quadrants, we choose complete quadrants as regions. For z in quadrants 1, 3, or 4 use

$$h_2(z) \sim \exp(5\pi i/12) F_2(z) \quad (60)$$

$$h'_2(z) \sim \exp(-\pi i/12) G_2(z) \quad (61)$$

For z in quadrant 2 use

$$h_2(z) \sim \exp(5\pi i/12) F_2(z) + \exp(11\pi i/12) F_1(z) \quad (62)$$

$$h'_2(z) \sim \exp(-\pi i/12) G_2(z) + \exp(-7\pi i/12) G_1(z) \quad (63)$$

For z in quadrants 1, 2, or 4 use

$$h_1(z) \sim \exp(-5\pi i/12) F_1(z) \quad (64)$$

$$h'_1(z) \sim \exp(\pi i/12) G_1(z) \quad (65)$$

For z in quadrant 3 use

$$h_1(z) \sim \exp(-5\pi i/12) F_1(z) + \exp(-11\pi i/12) F_2(z) \quad (66)$$

$$h'_1(z) \sim \exp(\pi i/12) G_1(z) + \exp(7\pi i/12) G_2(z) \quad (67)$$

The four auxiliary functions follow:

$$F_1(z) = K z^{-1/4} \exp(2i z^{3/2}/3) \sum_{m=0}^M C_M X^m \quad (68)$$

$$F_2(z) = k z^{-1/4} \exp(-2i z^{3/2/3}) \sum_{m=0}^M C_m Y^m \quad (69)$$

$$G_1(z) = k z^{1/4} \exp(2i z^{3/2/3}) \sum_{m=0}^M D_m X^m \quad (70)$$

$$G_2(z) = k z^{1/4} \exp(-2i z^{3/2/3}) \sum_{m=0}^M D_m Y^m \quad (71)$$

where X and Y equal $\mp i z^{-3/2}$ respectively, and

$$K = 2^{1/3} 3^{1/6} \pi^{-1/2} = 0.853\ 667\ 218\ 838\ 951$$

The coefficients C_m and D_m are again computed by recursion relations where $C_0 = D_0 = 1$:

$$C_m = C_{m-1} [9(2m-1)^2 - 4]/48m \quad (72)$$

and

$$D_m = D_{m-1} [9(2m-1)^2 - 16]/48m \quad (73)$$

Square roots of z are to be taken so that the real part of the root is always positive and the imaginary part has the same sign as the imaginary part of z . This applies also to fourth roots. The three-halves power is obtained as the product of z and its square root.

The summations in eq (68) - (71) can be done as indicated or evaluated by continued fractions. When done as indicated they are asymptotic series, and care must be taken to truncate them at the term of smallest magnitude, if this term is reached, because adding more terms will reduce the accuracy. Since the largest term in these series will always be 1, the series can be truncated if the terms become less than 10^{-h} in magnitude, where h is the number of decimal digits in the computer word.

Continued Fraction Expansion

The method of continued fractions is more effective in evaluating these asymptotic series, and it is used in subroutine Hankel in the FORTRAN program in this report. The coefficients are stored in lists entitled C4, C5, D4, and D5. In the remainder of this section the continued fraction technique is presented, along with the method of determining coefficients.

The continued fraction has the form

$$F(x) = b_0 + \frac{a_1}{x + b_1 + \frac{a_2}{x + b_2 + \dots}} \quad (74)$$

It is to be used to evaluate a polynomial

$$P(x) = \sum_{m=0}^M C_m X^m . \quad (75)$$

This polynomial can represent any of eq (68) - (71). One of three standard forms for continued fractions, this form is used because it has two coefficients at each stage and therefore is equivalent to an asymptotic series of twice as many terms. This reduces by half the number of divisions required. Since complex divisions are lengthy, requiring six real multiplications and two divisions, this is the only standard form of the continued fraction that can compete in computer time with the asymptotic series.

The coefficients a_i and b_i of eq (74) must be determined from the coefficients C_m . The usual technique is to express P as a rational function, then use the continued fraction to evaluate the rational function. The determination of the coefficients can be done in these two steps or by a second method which goes directly from power series to continued fraction coefficients. The second method is preferable because the loss of accuracy is more in the first. But since the first method is more easily understood, each method will be given; a computer program is included in appendix C which will determine coefficients by the second method.

Let M in eq (75) be an even number so that $2N = M$. (An additional unnecessary term of the series can always be used.) The rational function will have the form

$$R(x) \approx k \frac{\sum_{i=0}^N \hat{e}_i x^i}{\sum_{i=0}^N \hat{f}_i x^i} , \quad (76)$$

where $\hat{e}_0 = \hat{f}_0 = 1$ and $k = C_0$. The coefficients \hat{e}_i and \hat{f}_i are evaluated from a set of linear equations which can be described by displaying a particular case. For $N = 3$ they are as follows:

$$\begin{bmatrix} -1 & 0 & 0 & C_0 & 0 & 0 \\ 0 & -1 & 0 & C_1 & C_0 & 0 \\ 0 & 0 & -1 & C_2 & C_1 & C_0 \\ 0 & 0 & 0 & C_3 & C_2 & C_1 \\ 0 & 0 & 0 & C_4 & C_3 & C_2 \\ 0 & 0 & 0 & C_5 & C_4 & C_3 \end{bmatrix} \begin{bmatrix} k \hat{e}_1 \\ k \hat{e}_2 \\ k \hat{e}_3 \\ \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \end{bmatrix} = - \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{bmatrix} \quad (77)$$

With e_i and f_i thus determined, $R(x)$ is equivalent to $P(x)$ through the first $M + 1$ terms. $R(x)$ can now be evaluated exactly (except for round-off error) using a continued fraction of the form $F(x)$ of eq (74).

Rather than $R(x)$, however, a similar expression in $y = 1/x$ is the form that is well suited to evaluating asymptotic series. This expression is obtained by dividing each term of $R(x)$ by X^N . The order of the coefficients can now be reversed and a simple algebraic operation can yield a value of 1 for each of the two initial coefficients and a new value for k . We will call this new rational function with renamed coefficients $R(y)$. It will have the form of eq (76) but different coefficients, say e and f instead of \hat{e} and \hat{f} .

The coefficients a_i and b_i are determined from e_i and f_i by a recursive formula which involves constructing an $n \times n$ triangular matrix Q with elements $q_{i,j}$ as follows:

$$b_0 = e_0$$

$$q_{1,i} = (e_i - e_0 f_i) / a_1 \quad i = 1, 2, \dots, N,$$

where $q_{1,1} = 1$, giving a_1 , and

$$b_1 = f_1 - q_{1,2}.$$

The second row:

$$q_{2,i} = (f_i - q_{1,i+1} - b_1 q_{1,i}) / a_2 \quad i = 2, 3, \dots, N,$$

where $q_{2,2} = 1$, giving a_2 , and

$$b_2 = q_{1,2} - q_{2,3}.$$

Elements outside the matrix are assigned a value of zero. The remaining rows for $m = 3$ to N are as follows:

$$q_{m,i} = (q_{m-2,i} - q_{m-1,i+1} - b_{m-1} q_{m-1,i}) / a_m \quad i = m, m+1, \dots, N,$$

where $q_{m,m} = 1$, giving a_m , and

$$b_m = q_{m-1,m} - q_{m,m+1}.$$

The second method determines the continued fraction coefficients a_i and b_i directly from the asymptotic series coefficients c_i . This method is preferable to the first because the loss of accuracy in inverting the matrix in eq (77) can be more than the loss in this second method.

It has been pointed out* that the second method is probably a variant of the Viskovakoff algorithm described by Khovanskii (ref 12) and as such is unstable — subject to accumulation of errors. However, it is sufficiently stable to obtain the required coefficients.

*Private communication with AN Stokes, CSIRO, Wembley, Western Australia.

12. The Application of Continued Fractions and their Generalizations to Problems of Approximate Analysis, by AN Khovanskii; a monograph in Russian, 1956.

The coefficients are derived as follows. The well-known recursive relations that give the N th stage of a continued fraction as a rational function are used (ref 13).

$$F_N(y) = A_N(y)/B_N(y) , \quad (78)$$

where

$$\begin{aligned} A_N &= \sum_{i=0}^N e_i y^i \\ &= (y + b_N) A_{N-1} + a_N A_{N-2} \end{aligned} \quad (79)$$

$$\begin{aligned} B_N &= \sum_{i=0}^N f_i y^i \\ &= (y + b_N) B_{N-1} + a_N B_{N-2} , \end{aligned} \quad (80)$$

in which $A_{-1} = 1$, $A_0 = b_0$, $B_{-1} = 0$, and $B_0 = 1$. Again $y = 1/x$. The long division indicated in eq (78) is then carried out, giving a quotient in terms of a_i , b_i , and y that can be equated, term by term, to the first $2N-1$ terms of the asymptotic series.

The long division is carried out with A_N and B_N written in descending powers of y . The quotient is then in descending powers of y or ascending powers of x . Fortunately, the first $2N+1$ terms determined for any N are identical to the same initial terms for any larger value of N . This will be proven later. The first few equations obtained from the division are as follows:

$$\begin{aligned} b_0 &= C_0 \\ a_1 &= C_1 \\ -a_1 b_1 &= C_2 \\ a_1 (b_1^2 - a_2) &= C_3 \\ a_1 (2 a_2 b_1 - b_1^3 + a_2 b_2) &= C_4 \end{aligned} \quad (81)$$

From these equations a_i and b_i can be determined, since the coefficients C_i are known. However, a simpler method is available.

The long division indicated in eq (78) can be carried to $2N+1$ valid places; but beyond $N+1$ places, terms from the original dividend are no longer entering the remainder. Therefore terms in the later part of the quotient have a simplified form. Since term $n+1$ of

13. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, ed by M Abramowitz and IA Stegun; National Bureau of Standards Applied Mathematics Series, vol 55, p 19, 1964.

the quotient is equal to the n th asymptotic coefficient, designate it C_m . Note that the C 's are numbered from 0 to N . Let the coefficient of y^m in B_n be $B_{n,m}$. Then

$$C_{N+j} = - \sum_{i=1}^N B_{N,N-i} C_{N+j-i}, \quad 1 \leq j \leq N. \quad (82)$$

Here the C 's are numerical constants. The unknowns, the a 's and b 's, are in the terms of B . Suppose that these unknowns have been determined up to $n = N-1$. Then eq (82) will contain two unknowns, a_N and b_N . By using eq (82) for $j = N-1$ and N , the unknowns can be evaluated. The index N can then be increased by 1 and the process repeated. The process can start with $N = 2$ if a_1 , b_0 , and b_1 are provided, but these are easily determined from eq (81). The terms of B_N are determined from eq (80), which gives for each term

$$B_{n,m} = B_{n-1,m-1} + b_n B_{n-1,m} + a_n B_{n-2,m}. \quad (83)$$

Any $B_{n,m}$ is zero if m is greater than n .

When $j = N-1$ is used in eq (82) in the process described above, the coefficient of the $(2N-1)$ power of x is being evaluated. This term is expected to contain a_N and b_N , but — as will be proven later — because the coefficient of b_N is zero, a_N is the only unknown in a linear equation and can be easily evaluated. The next term determined with $j = N$ contains a_N and b_N , but now only b_N is unknown and is easily evaluated.

As an example, the C_n 's through $n = 10$ are listed in table 1. These are the asymptotic series coefficients given by eq (72). The corresponding a_n 's and b_n 's as determined above are also listed. A more complete list of the a 's and b 's can be obtained from the FORTRAN program in appendix C.

Table 1. Asymptotic series coefficients, C_n , and the corresponding continued fraction coefficients, a_n and b_n .

n	c_n	a_n	b_n
0	1.	0	1.
1	0.10416	0.10416	-0.80208
2	0.08355	-0.58764	-2.28555
3	0.12823	-2.29072	-3.77864
4	0.29185	-5.11525	-5.27462
5	0.88163	-9.06285	-6.77193
6	3.32141		
7	14.99576		
8	78.92301		
9	474.45154		
10	3207.49009		

A FORTRAN program to compute the continued fraction coefficients for the series given by eq (72) is given in appendix C. This program can be easily modified to determine the other set.

Two Proofs

In this section proofs will be given of two facts used in the previous section. Following this, the number of terms required, the accuracy, and similar topics will be discussed. To prove that the first $2N+1$ terms of the quotient A_N/B_N are equal to the same terms when N is a larger integer, use long division on eq (79) and (80) to obtain

$$A_N/B_N = A_{N-1}/B_{N-1} + a_n (A_{N-2} B_{N-1} - A_{N-1} B_{N-2}) / (B_N B_{N-1}) \quad (84)$$

If the first quotient on the right is to have terms equal to the quotient on the left up through term $2N-1$, the remainder must have no terms with y to a higher power than $-(2N-1)$. The final divisor, $B_N B_{N-1}$, contains y to the $(2N-1)$ and lower powers. Therefore, the proof is complete if the numerator of the remainder is a constant. To show this, use eq (79) and (80) to evaluate B_{N-1} and A_{N-1} ; it can be shown that

$$\begin{aligned} A_{N-2} B_{N-1} - A_{N-1} B_{N-2} &= -a_{N-1} (A_{N-3} B_{N-2} - A_{N-2} B_{N-3}) \\ &= (-1)^N a_{N-1} a_{N-2} \dots a_1 \end{aligned}$$

The right-hand product is obtained by repeatedly applying the middle result. The product of a 's is a constant, completing the proof.

The second proof required is that in the quotient of A_N/B_N , C_{2N} (the coefficient of y^{-2N}) will contain no a_i or b_i to higher than term N and C_{2N+1} (the coefficient of y^{-2N-1}) will involve no a_i to higher than term $N+1$ and no b_i to higher than term N .

The first part is intuitively obvious. Since from the preceding proof C_{2N} will be the same when derived from the ratio A_x/B_x for any x as long as it is N or greater, we need consider only the case where x is N . But since from eq (79) and (80) A_N and B_N contain no a 's or b 's of greater than term N , C_{2N} cannot contain a 's or b 's of higher terms.

By the same argument C_{2N+1} can contain no a 's or b 's to higher terms than $N+1$. There remains to be proven only that b_{N+1} cannot exist in C_{2N+1} or that its coefficient, which we will call E , is zero. Applying eq (82) for $N+1$ and $j = N$ gives

$$C_{N+1+N} = - \sum_{i=1}^{N+1} B_{N+1, N+1-i} C_{N+1+N-i}$$

E , the coefficient of b_{N+1} in this expression from eq (83), takes the form

$$E = - \sum_{i=1}^{N+1} B_{N, N+1-i} C_{2N+1-i}$$

But by choosing $j = N$ in eq (82) we see that terms 1 to N for C_{2N} equal terms 2 to $N+1$ in E , so

$$E = -B_{N,N} C_{2N} + C_{2N} .$$

However, since $B_{N,N}$, the coefficient of y^n in B_n , is always 1 by eq (80), $E = 0$. Therefore b_{N+1} does not exist in C_{2N+1} .

Number of Terms

The number of terms or stages to use in the continued fraction was arrived at by a trial and error process. For a given number of terms, a real positive argument was decreased until the accuracy began to drop. The magnitude just before this drop was considered to be the optimum point to increase the number of stages by one. Because the argument to the continued fractions is $z^{3/2}$, we took the larger of the magnitudes of the real and imaginary parts of $z^{3/2}$ as the test number. This number is then compared to the $3/2$ power of the points determined along the real axis by trial and error.

The above method appears to work well although it involves no thorough understanding of the way complex numbers affect the successive convergents of a continued fraction. Table 2 shows the points down to which a given number of stages gives full accuracy for positive real arguments and lists the $3/2$ power of these numbers as used in the FORTRAN program list called ZMLA5.

Division Lines

The power series method is now to be used for small arguments and the continued fraction method for large arguments. The exact dividing line between them is needed. The division line of figure 4 was arrived at by computing the functions along rays from the origin, using both power series and continued fractions. The number of decimal places to which the functions determined by the two methods agree tends to reach a maximum at some distance from the origin along each ray. At distances short of this maximum we can assume that the continued fraction method is less accurate than the power series. At distances beyond the maximum, the power series is assumed to be less accurate. The maximum therefore indicates the ideal place to change from one method to the other if the decision is to be based solely on accuracy. This method was used to determine figure 4.

A complication arises, however. Along certain rays from the origin, h_1 and its derivative reach a maximum number of places at very different distances from h_2 and its derivative. The principal problem is at $\pm 60^\circ$ but persists from about 30° to 90° . At 60° , h_1 is small in magnitude and h_2 is large. The power series method cannot compute the small values accurately due to loss in accuracy in subtraction in eq (48). The accuracy of the continued fraction for h_2 is poor at 60° because eq (69) becomes a nonalternating series and continued fraction approximations are not known to improve the accuracy of nonalternating asymptotic series as they do for alternating series.

A reasonable solution to this problem is to compute h_1 by continued fractions and h_2 by power series for arguments at these angles and magnitudes from 4 to 10. However, as will be shown later, the above solution has not been employed at this time since this area is not of great importance for normal mode computations. Instead, the argument was chosen

Table 2. Cut-off points for determining the number of stages in the continued fractions.

Number of Stages	Real Argument x	Program Test Value $x^{3/2}$
1	10^6	10^9
2	80	715.0
3	35	207.0
4	22	103.0
5	13	47.0
6	11	36.4
7	9	27.0
8	8	22.6
9	7	18.5
10	6.5	16.6
11	6	14.7
12	5.8	14.0
13	5.5	12.9
14	5.3	12.2
15	5.1	11.5
16	4.9	10.8
17	4.5	9.5
18	4.4	9.2

that gave equivalent accuracy for the two methods. Along 60° this minimum accuracy is 9 decimal places.

The following relationship exists between h_1 and h_2 for positive and negative values of the imaginary part of the arguments:

$$h_1(z^*) = [h_2(z)]^*$$

where the * means complex conjugate. Thus, the above discussion at 60° can be translated to -60° . Also, the functions actually need to be computed only in quadrants I and II. They could then be evaluated in quadrants III and IV by the above relationship. The above relationship explains the symmetry of figure 4 about the real axis.

COMPARATIVE ACCURACY

The accuracy of the three methods – power series, asymptotic series and continued fractions – has been determined on a CDC computer with 48 bits or 14.4 decimal places of accuracy in the floating point word. Since this differs from the double precision word length of 60 bits or 18.1 decimal places that applies to the preceding part of this report, these results are for comparative and illustrative purposes only.

Accuracy is determined by computing the functions and either comparing the answers for the several different computing methods or computing the wronskian. The wronskian is a constant given by the relationship

$$h_1 h_2' - h_2 h_1' = -1.45749544104i = -i 96^{1/3}/\pi . \quad (85)$$

The wronskian will determine the accuracy of the functions if it can be computed without loss of accuracy. If the two products in it are large, though, accuracy will be lost in the subtraction. This generally happens for arguments near the negative real axis. Here accuracy must be determined by comparing answers from different methods. The accuracy of the functions and their derivatives will generally be about equal.

Figure 5 illustrates the accuracy that is obtained in different parts of the complex plane of the argument, z , by using the power series method. On the inner contour, the functions h_1 and h_2 and their derivatives have 12 places of accuracy. On the outer contour, the accuracy is 11 places. As expected, the accuracy is best for arguments of small magnitude. The accuracy remains best in directions from the origin in which the functions are large in magnitude. This is because less accuracy is lost in subtraction. Accuracy must be lost when individual terms of the series are large but the sum is small.

Figure 6 shows accuracy contours for the asymptotic expansion with both the direct and continued fraction evaluation of the series. Here, the best accuracy is obtained for large arguments, and accuracy decreases toward the origin. As can be seen, each of the two methods is better in some directions from the origin. The choice of methods then depends upon which directions are of most value to the normal mode program. The dots on the figure show the locations at which the functions were evaluated in a typical surface duct run. Although arguments can lie anywhere in the plane, most of them follow this pattern. They lie just above the negative real axis and in a narrow angle above the positive real axis. The continued fraction method is distinctly better on this positive side. Since computing time also favors the continued fraction method, it is clearly the method to use.

If the 12-place accuracy contour from figure 6 lies inside that for figure 5 at some angle from the origin, 12 places can be obtained at any range along this angle by using either power series or asymptotic expansion in the interval of overlap. If the asymptotic expansion contour lies outside the other, there is an interval in which 12 places cannot be obtained. Only some lesser number of places can be obtained in this interval. These contours apply when both functions and their derivatives are all computed by a single method. As mentioned earlier, increased accuracy could be obtained in some areas by computing the two functions by different methods.

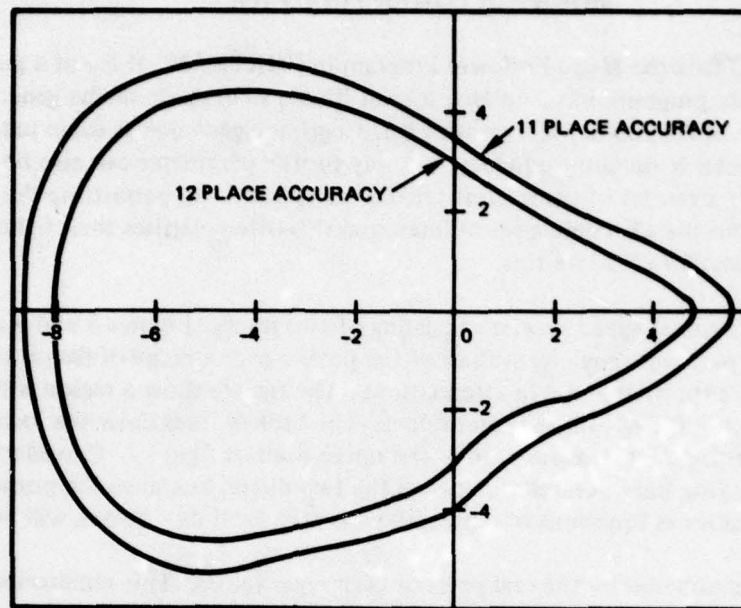


Figure 5. Locus of arguments for which the power series evaluation of the modified Hankel functions gives 12 and 11 decimal places of accuracy for a computer word length of 14.4 decimal places.

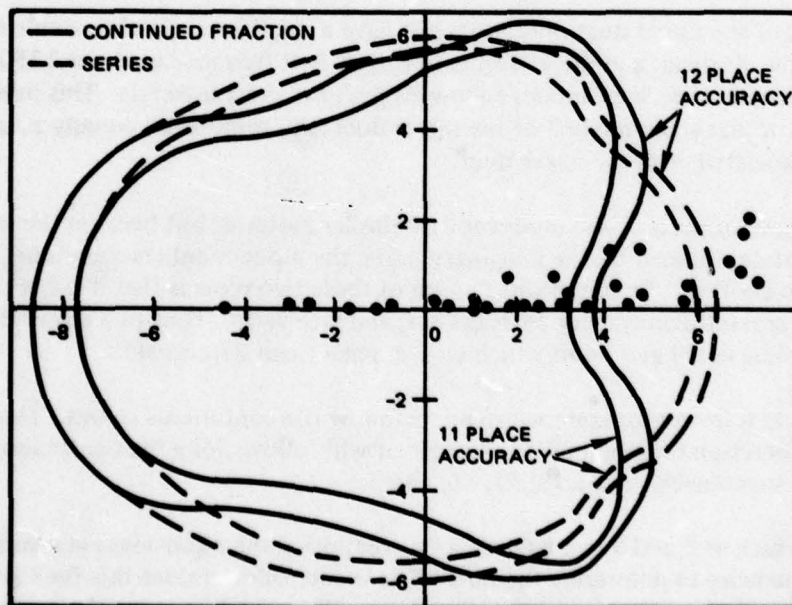


Figure 6. Locus of arguments for which the direct and continued fraction evaluation of the asymptotic series gives 11 and 12 decimal place accuracy. The arguments at which the modified Hankel functions were evaluated in a typical normal-mode run are shown.

MODE FOLLOWER PROGRAM

Appendix D lists the Mode Follower Program in FORTRAN. It is not a part of the general normal mode program, but is related in that it uses some parts of the general program. The purpose of the mode follower is to trace a given eigenvalue as some parameter is varied. This parameter is usually frequency, but any profile parameter can also be varied. The eigenvalues at a given set of parameters are discrete points. By permitting the parameter to vary, the eigenvalues become a set of lines, and this often clarifies their behavior at the fixed points. Figures 7-9 illustrate this.

Figure 7 is a sound speed profile consisting of two ducts. Figures 8 and 9 show the real and imaginary parts of some eigenvalues of the profile over a range of frequencies. The imaginary parts are expressed as mode attenuations. The figures show a region where both ducts are exerting an influence on the eigenvalues. The broken lines show the location of eigenvalues for a profile that consists of only the upper duct of figure 7. Considerable time could be spent studying the interaction between the two ducts, but since the purpose here is to illustrate eigenvalues as functions of a parameter, only a brief description will be given.

Modes are numbered by the real parts of their eigenvalues. This numbering is consistent with the number of beats or changes of π in the phase of the depth functions. Thus the eigenvalue of a mode numbered 1 in a profile consisting of only the upper duct lies exactly over the eigenvalues of a mode in the double duct in figures 8 and 9, but this mode in the double duct changes number each time it crosses the real part of another mode. The depth function actually gains an additional beat each time this happens. The background of modes that are being crossed consists of the higher order, untrapped modes associated with the lower duct.

Mode 2, of the upper duct only, does not have a single mode in the double duct that overlies it exactly. Instead, a mode attempts to follow it at frequencies above 1350 Hz. Below this frequency, successive modes follow its path for short intervals. This interplay between modes occurs when mode 2 of the upper duct is in some sense equally as untrapped as the modes associated with the lower duct.

The imaginary parts of the modes follow similar patterns; but because the mode numbering is not determined by the imaginary parts, the mode numbers sometime jump from one line to another. An important feature of these two plots is that if the real parts of the eigenvalues cross, the imaginary parts do not; and vice versa. Thus two eigenvalues do not tend to become equal at a point which would make them degenerate.

The mode follower program will tend to follow the continuous curves. Thus if started in the right direction on mode 59 at 1450 Hz, it will follow along the continuous mode which becomes successively mode 58, 57, 56, and 55.

Figures such as 8 and 9 can be drawn by computing the eigenvalues at a sufficient number of frequencies to determine the lines. The mode follower does this for a given eigenvalue while adjusting the step size so the mode will not be lost, or so the program will correctly follow the mode. The step size is permitted to become large where the eigenvalue can be approximated by a parabolic curve, but it shortens when extrapolation to the next point becomes less accurate.

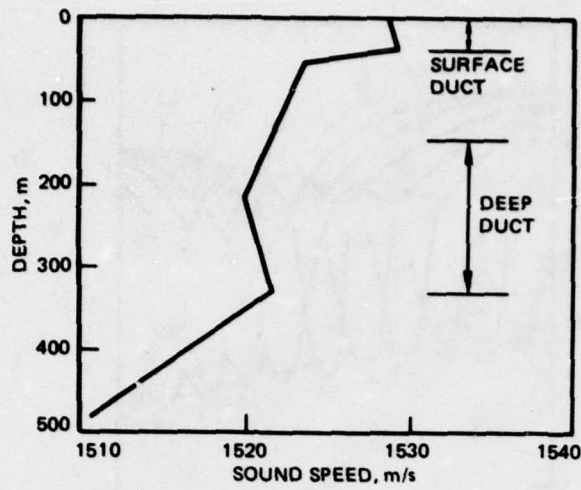


Figure 7. A five-layer approximation to the sound speed of a surface duct overlying a refractive duct.

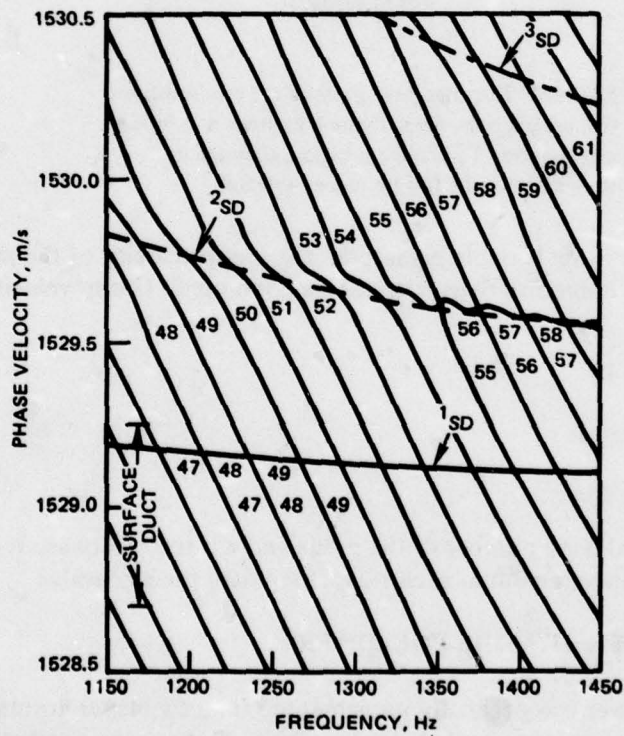


Figure 8. The real part of the eigenvalues as continuous functions of frequency for the profile of figure 7. Some mode numbers are given. The first three modes for the surface duct only (SD) are shown as broken lines. That for mode 1 coincides with an existing line.

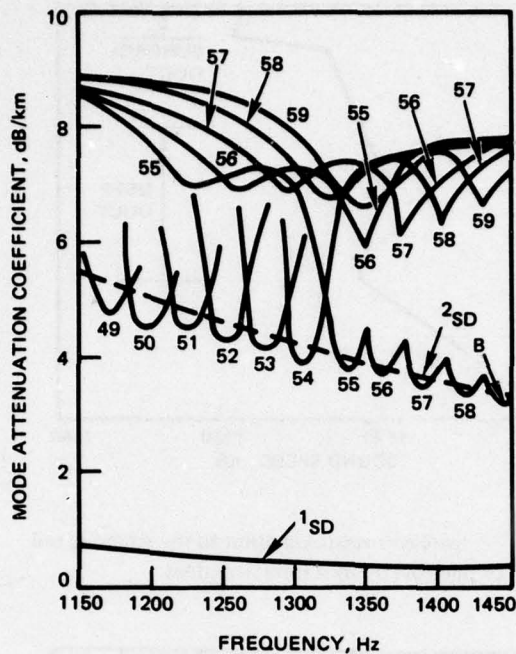


Figure 9. The imaginary parts of the modes whose real parts are shown in figure 8 expressed as mode attenuation. To avoid confusion, they are not shown across the full frequency interval.

When frequency is the variable parameter, the group velocity of the mode can be computed easily since a numerical derivative can be computed. Group velocity is given by the relationship

$$\begin{aligned}
 C_g &= d\omega/d(\text{Re } k) \\
 &\cong \Delta\omega/\Delta(\text{Re } k) \\
 &\cong -\Delta f v^2/f \Delta v,
 \end{aligned}$$

where k is the horizontal wave number of the mode and v is the real phase velocity. The mode follower prints this value out at each step, along with the eigenvalue.

IMPLEMENTATION OF THE MODE FOLLOWER

The mode follower was originally implemented for a two-layer normal mode which differed from the n -layer program in that the derivative dG/dv of the characteristic equation was evaluated along with G . The iteration for roots of G was thus Newton-Raphson and is given by the relationship

$$v_{i+1} = v_i - G/G' \quad (86)$$

This is simpler than the secant iteration of eq (15), in which G' must be evaluated numerically. Because of the simpler iteration, an effective scheme for mode following was available.

Since considerable effort was required to develop a similar scheme for the n-layer case, the two-layer mode follower will be briefly described to serve as an introduction to the n-layer case.

The two-layer mode follower employs one iterative step of eq (86) at each point where G is evaluated. Thus, a root that is inexact but sufficiently exact is obtained. The original estimate is obtained by extrapolating from the three most recent roots. If this estimate is sufficiently close to the true root, the single iterative step will make a small correction, G/G' , that will bring the estimate very close to the true root. By using the size of this correction to control the step size, the program is self-regulating. The program works well when a permissible value of G/G' of 10^{-6} to 10^{-4} is used. Outside this interval the step size is either doubled or halved.

The multiple-layer program differs from this in several details. The extrapolation from the previous three points is done not only for the phase velocity but also for the numerical derivative,

$$D^{-1} = \Delta v / \Delta G.$$

Lagrange three-point interpolation is used, given by the form

$$v(x) = \frac{v(x_1)(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} - \frac{v(x_2)(x-x_1)(x-x_3)}{(x_1-x_2)(x_2-x_3)} + \frac{v(x_3)(x-x_1)(x-x_2)}{(x_1-x_3)(x_2-x_3)}, \quad (87)$$

where x is the new value of the parameter that is being varied (usually frequency) and x_1 , x_2 , and x_3 are the three previous values, x_1 being the most recent. To extrapolate the derivative, v is replaced by D^{-1} in eq (87). Both quantities are complex numbers.

The determinant is now evaluated at this new phase velocity to give a value G_0 . Next a corrected value of phase velocity, v_0 , is obtained:

$$v_0 = v - G_0 D^{-1}. \quad (88)$$

In the two-layer case, the size of the correction, GD^{-1} , is used to control the step size. Because the numerical derivative is less precise, we evaluate G once more at this new position, obtaining G_1 . A new numerical derivative is next calculated:

$$D_0^{-1} = (v_0 - v) / (G_1 - G_0).$$

This derivative is now compared with the extrapolated value to determine whether the step size should be changed. To do this an error

$$E = |1 - D_0/D|^{-2}$$

is computed. Good results have been obtained by keeping E between 10^{-5} and 10^{-2} . If E becomes larger than this, the step size is halved and the extrapolation is tried again. Should halving the step size five times fail to obtain a value of E less than 10^{-2} , the mode is presumed to be lost and the program halts.

If E is less than 10^{-2} , the step is successful and the stored values are updated for the next step. Before v is stored, though, the iterative step of eq (88) is applied one more time to obtain a more precise value of v . This requires little extra effort because the quantities G_1 and D_0^{-1} are already available.

If the error E is less than 10^{-5} , the next step size is doubled.

It is possible for the extrapolation to be too successful. That is, if v is very near the true root, G_0 and Δv will be very small. The numerical derivative may then be inaccurate. Therefore, before the error term E is computed, a quantity

$$F = |v/\Delta v|^2$$

is computed. If F is greater than 10^{28} , the extrapolated derivative is used rather than the computed derivative and the program proceeds to the next step. If F is greater than 10^{34} , the step size is doubled before proceeding to the next step.

The other principal part of the program is the initialization which must evaluate v at three values of x to obtain the numbers needed for the first extrapolation, eq (87).

INPUT AND OUTPUT

The first input card contains the maximum number of steps allowed, the limits applied to E and F , and keys which control both the amount of detail in the printout and whether the profile parameters are to be read in or retained from the previous run. Default values are supplied when these items are left blank. Next the profile parameters are read in. These are an older style and only permit specification of the absorption loss at the top of a layer. The sound speed gradient is assumed to be real at the top of any layer.

A final card indicates which variable — frequency, sound speed, depth, gradient, absorption, or density — will be varied, by specifying a number called n_x in the program, from 1 to 6. The next number, n_y , specifies which layer the variable will be in. This layer number is not needed if frequency is selected. A third number, n_z , indicates, if zero, that the profile will remain continuous as the selected parameter is varied. If n_z is not zero, the selected parameter moves alone without a compensating motion in other profile parameters. The card next gives the initial and final value of the parameter to be varied and the initial step size. Finally, the particular mode to be followed is indicated by giving an approximate phase velocity and an initial step size. These must be chosen such that the subsequent iteration will converge on the correct mode.

The principal output of this program is the print statement at line 314. Each line of output contains the value of the parameter being varied, the complex phase velocity, the determinant G , the derivative D^{-1} , the error term E , the mode attenuation, the mode group velocity (if frequency is the parameter being varied), and the step number. After the final step, the profile in its final form is printed out.

CONCLUSIONS

1. An effective program for computing propagation loss in a layered ocean by normal modes has been developed. Complete documentation for the program is contained herein.
2. Sediment layers are modeled as fluids in which densities, sound speeds, and absorption can be specified. This permits a complete wave solution for bottom reflected sound energy.
3. A continued fraction technique for evaluating asymptotic series is shown to give superior results in evaluating the auxiliary functions required in this program, the modified Hankel functions of order $1/3$.
4. A mode follower program given here is useful in tracing eigenvalues. Such traces are needed to understand the eigenvalue structure.

RECOMMENDATIONS

1. Improve the mode locating ability of this normal-mode program to make it self-contained. It currently requires user interaction to locate eigenvalues.
2. Investigate methods to incorporate the effect of rough boundaries into this program.

REFERENCES

1. The Bilinear Modified-Index Profile, by WH Furry, in Propagation of Short Radio Waves, DE Kerr, ed; MIT Rad Lab series, vol 13, p 140-168, McGraw-Hill, New York, 1951.
2. Navy Underwater Sound Laboratory Report 111, Theory of the Anomalous Propagation of Acoustic Waves in the Ocean, by HW Marsh, 1950.
3. Normal-Mode Theory Applied to Short-Range Propagation in an Underwater Acoustic Surface Duct, by MA Pedersen and DF Gordon; J Acoust Soc Am, vol 37, p 105-118, January 1965.
4. Naval Air Development Center Report NADC-72002-AE, Normal Mode Solutions and Computer Programs for Underwater Sound Propagation, by CL Bartberger and LL Ackler, 4 April 1973.
5. A Normal Mode Theory of an Underwater Acoustic Duct by Means of Green's Functions, by RL Deavenport; Radio Sci, vol 1, p 709-724, 1966.
6. Analytic Description of the Low-Frequency Attenuation Coefficient, by WH Thorp; J Acoust Soc Am, vol 42, p 270, 1967.
7. Some Effects of Velocity Structure on Low-Frequency Propagation in Shallow Water, by AO Williams; J Acoust Soc Am, vol 32, p 363-365, March 1960.
8. Sound Attenuation as a Function of Depth in the Sea Floor, by EL Hamilton; J Acoust Soc Am, vol 59, p 528-535, March 1976.

9. **Sound Propagation in a Channel with Lossy Boundaries**, by HP Bucker; *J Acoust Soc Am*, vol 48, p 1187-1194, November 1970.
10. **The Connection Between Normal Modes and Rays in Underwater Acoustics**, by KM Guthrie; *J of Sound and Vibration*, vol 32 no 2, p 289-293, 1974.
11. **Tables of the Modified Hankel Functions of Order One-Third and their Derivatives**, Harvard University Computation Laboratory; Harvard University Press, Cambridge MA, 1945.
12. **The Application of Continued Fractions and their Generalizations to Problems of Approximate Analysis**, by AN Khovanskii; a monograph in Russian, 1956.
13. **Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables**, ed by M Abramowitz and IA Stegun; National Bureau of Standards Applied Mathematics Series, vol 55, p 19, 1964.

APPENDIX A: NORMAL MODE PROGRAM IN FORTRAN

This program consists of the main program and seven subroutines. The main program handles the input and output and performs much of the computation. This includes profile preparation, mode search, determination of depth function coefficients, normalization, computation of depth functions, and summation of modes. Auxiliary functions are performed by the subroutines SETUP and DET, which set up the determinant, then evaluate it. This is the determinant from which eigenvalues are determined. The subroutine HZERO determines the Hankel functions of order zero, second type, which gives the range dependence of the modes. Only a single term of the asymptotic expansion is needed for this function.

Subroutine HANKEL evaluates the modified Hankel functions of order $1/3$, by which the depth dependence of the modes is expressed. The majority of computing time is usually expended in this subroutine. Subroutine CFR is used by subroutine HANKEL to evaluate continued fractions. Subroutine RCOEF evaluates and prints reflection coefficients when they are requested.

```

1      C      THIS IS THE MAIN PART OF NLAYNM
2      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
3      DOUBLE PRECISION LAMBDA,LAMBDI
4      INTEGER COL
5      REAL R ATTEN, T RE, RX
6      DIMENSION LOSPCH(5,25)
7      COMMON /HAN/ H2R,H2I,H1R,H1I,H2PR,H2PI,H1PR,H1PI,R
8      COMMON/INPUT/ Z(12), N, OMEGA, V, VI, CON(12), GSQ(12),
9      1 CAY(12), LAMBDA, LAMBDI, G(12)
10     2,RHO(12), GI(12), G SQI(12), CAYI(12)
11     COMMON /EXPO/ EXSUM, CNTR, RATIO(25)
12     COMMON/DETMNT/ A(25,4), Q(25,4)
13     COMMON/PARTS/ ZT(12),ZTI(12),ZB(12),ZBI(12)
14     COMMON/REFL/ AF(12,200), AG(12,200), BF(12,200), BG(12,200),
15     2 EIGEN(350), EIGENI(350), B (25,4), BI(25,4), CB(12), CBI(12),
16     3 CAYSQ(12), CAYSQI(12), NN
17     DIMENSION D(350), DI(350), F(100), FI(100), HZERO2(350),
18     * DA(350), SRES(350), GAMMAI(12), BLPK(12),
19     4 HZER2I(350), DPK(12), GCU(12), GCUI(12), CI(12),
20     3 PHASE V(350), PHASI V(350), UU(2000), UUI(2000)
21     COMMON /LIMIT/ TLIM, EXPONT, SLIM
22     DIMENSION LOSS(101)
23     DIMENSION C(12), DEPTH(52), DBLOSS(350), COL(120),
24     1CONTR(10), EF(2), FMAG(350), FANG(100),
25     2GAMMA(12),JSMBL(10),JCOUNT(5),JCOU(5),LEVEL(41),PLEV(5),RLOSS(100)
26     3 , RLOS(101),RECVRS(51),TEST(3),      ING(11)
27     EQUIVALENCE (FF,EF(1)),(DEPTH(1),SOURCE),(DEPTH(2),RECVRS(1)),
28     1      (RLOS(2),RLOSS(1))
29     COMMON /AHZERO/ HZEROR,HZEROI
30     DATA ( CONTR(I), I=1,4) /110.D0,95.D0,80.D0,-1000.D0/,
31     1 (J SMBL(I), I=1,3) /1H1,1H*,1H8/,
32     *(ING(I), I = 1,10)/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/,
33     2 (TEST(I), I=1,3) /.2D0,1.D0,5.D0/
34     TLIM = 25.
35     SLIM = -8.54988
36     C READ IN PARAMETERS
37     1 READ 11, K1, K2, K3, K4, K5, K6, K7, K8, K9
38     C KEYS& 1-DEPT FN PRINTOUT, 2-LOSS PRINTOUT, 3-REFLECTION COEFF PRINTOUT
39     C 4-CHANGE CONTOURS, 5-CONTINUE MODES
40     11 FORMAT (10I4)
41     PRINT 13, K1, K2, K3, K4, K5, K6, K7, K8, K9
42     13 FORMAT (6H1KEYS , 10I4)
43     IF (K1 .LT. 7) GO TO 5
44     READ 11, M
45     READ 434, (SRES(I), I = 1,M)
46     434 FORMAT (5D16.7)
47     5 IF (K8 .NE. 1) GO TO 8
48     READ 20, T LIM
49     SLIM = -DCBRT(TLIM**2)
50     PRINT 30, TLIM, SLIM
51     8 EXPONT = DEXP((TLIM + TLIM) / 3.)
52     K6 = K6 + 1
53     INSUR = 0
54     IF (K2 .LT. 10) GO TO 16
55     K2 = K2 - 10
56     INSUR = 1

```

```

57      16  MPCH = 0
58      IF (K5 .LT. 10) GO TO 17
59      K5 = K5 - 10
60      MPCH = 1
61      17  IF (K4 .NE. 1) GO TO 3
62      READ 20, (CONTR(I), I = 1,9)
63      CONTR(10) = -1000.00
64      READ 4, (J SMBL(I), I = 1,9)
65      4   FORMAT (9A1)
66      3   READ 10, N, FREQ
67      10  FORMAT (I2, D10.1)
68      IF (N.EQ.0) GO TO 999
69      2   PRINT 12, N, FREQ
70      12  FORMAT (I3, 8H LAYERS, ,F10.1, 3H HZ)
71      READ 20, (Z(I), I=1,N)
72      PRINT30, (Z(I), I=1,N)
73      READ 20, (C(I), I = 1,N)
74      20  FORMAT (8D10.4)
75      PRINT30, (C(I), I = 1,N)
76      READ 20, (CB(I), I = 1,N)
77      PRINT30, (CB(I), I = 1,N)
78      READ 20, (GAMMA(I), I = 1,N)
79      PRINT30, (GAMMA(I), I = 1,N)
80      READ 20, (DPK(I), I = 1,N)
81      PRINT 30, (DPK(I), I = 1,N)
82      READ 20, (BLPK(I), I = 1,N)
83      PRINT30, (BLPK(I), I = 1,N)
84      READ 20, (RHO(I), I = 1,N)
85      PRINT 30, (RHO(I), I = 1,N)
86      IF (FREQ .GT. 0.) GO TO 18
87      FREQ = - FREQ
88      ATTN = 0.
89      GO TO 19
90      18  F SQ = (FREQ / 1000.)**2
91      ATTN = .1 * F SQ / (1. + F SQ) + 40. * F SQ / (4100. + F SQ)
92      19  ATTN = ATTN * 1.0936
93      PRINT 14,ATTN
94      14  FORMAT (8H ATTN = ,G10.5, 5HDB/KM )
95      ATTN = ATTN / 1000.00
96      30  FORMAT (9F14.5)
97  C COMPLETE PROFILE
98      DO 33 I = 1,N
99      IF (RHO(I) .EQ. 0.) RHO(I) = 1.02
100     IF (CB(I) .NE. 0.) GO TO 31
101     CB(I) = C(I+1)
102     31  IF (C(I) .NE. 0.) GO TO 32
103     C(I) = CB(I-1)
104     32  IF (DPK(I) .GE. 0.) GO TO 34
105     CI(I) = CBI(I-1)
106     GO TO 36
107     34  CI(I) = 0.
108     IF (DPK(I) .EQ. 0.) GO TO 36
109     T = 27287.52708 / DPK(I)
110     CI(I) = T - SQRT((T - C(I)) * (T + C(I)))
111     36  CBI(I) = 0.
112     IF (GAMMA(I) .NE. 0.) GO TO 38
113  C     **BOTH SOUND SPEEDS GIVEN

```

```

114     IF (BLPK(I) .LE. 0.) GO TO 37
115     T = 2/287.52708 / BLPK(I)
116     CBI(I) = T - SQRT ((T - CB(I)) * (T + CB(I)))
117 37    T = C(I) * (C(I)**2 - 3. * CI(I)**2)
118     TI = CI(I) * (3. * C(I)**2 - CI(I)**2)
119     IF (BLPK(I) .LT. 0.) GO TO 39
120     TEMP = CB(I)**2 - CBI(I)**2
121     TEMP1 = 2. * CB(I) * CBI(I)
122     DENOM = TEMP**2 + TEMP1**2
123     TEMP = TEMP / DENOM
124     TEMP1 = -TEMP1 / DENOM
125     GAMMA(I) = 0.5 * (C(I) - (T * TEMP - TI * TEMP1)) /
126     * (Z(I+1) - Z(I))
127     GAMMAI(I) = 0.5 * (CI(I) - (T * TEMP1 + TI * TEMP)) /
128     * (Z(I+1) - Z(I))
129     IF (I .EQ. N) GO TO 37
130     GO TO 33
131 C    **SPECIAL CASE, GRADIENT REAL NUMBER
132 39    IF (CI(I) .EQ. 0.) GO TO 42
133     TEMP = CB(I)**2
134     TEM = TEMP**2
135     TEMP1 = CI(I)
136     COEF1 = CI(I)
137     COEF2 = 2. * TEMP * CI(I) + TI
138     COEF3 = 2. * T * CB(I)
139     COEF4 = TEM * CI(I) - TEMP * TI
140     OLDFN = 1.D20
141     DO 41 J = 1,10
142     FN = (((COEF1 * TEMP1) + COEF2) * TEMP1 + COEF3) * TEMP1 + COEF4
143     FP = ((4. * COEF1 * TEMP1) + 2. * COEF2) * TEMP1 + COEF3
144     TEMP1 = TEMP1 - FN / FP
145     IF (FN .GE. OLDFN) GO TO 43
146     OLDFN = FN
147 41    CONTINUE
148 43    CBI(I) = TEMP1
149     GAMMA(I) = .5 * (.5 * (CI(I) * (TEMP - CBI(I)**2) - TI) /
150     * (CB(I) * TEMP1) + C(I)) / (Z(I+1) - Z(I))
151     GO TO 28
152 42    GAMMA(I) = C(I) - T / (CB(I)**2 * (Z(I+1) - Z(I)) * 2.)
153     GO TO 33
154 C    **SOUND SPEED AND GRADIENT GIVEN
155 38    IF (I .EQ. N) GO TO 33
156     T = C(I) * (C(I)**2 - 3. * CI(I)**2)
157     TI = CI(I) * (3. * C(I)**2 - CI(I)**2)
158     IF (BLPK(I) .EQ. 0.) GO TO 29
159     IF (BLPK(I) .LT. 0.) GO TO 28
160     TEMP = (BLPK(I) / 54575.05416)**2 / (Z(I+1) - Z(I)) * 0.5
161     T = T * TEMP
162     TI = TI * TEMP
163     TEMP = .5 * C(I) / (Z(I+1) - Z(I)) - GAMMA(I) + T
164     T = -(TI - SQRT( TI * TI + T * TEMP)) / T
165     CB(I) = 54575.05416 * T / BLPK(I) / (1. + T * T)
166     CBI(I) = CB(I) / T
167     GO TO 37
168 C    **SPECIAL CASE, GRADIENT REAL NUMBER
169 28    TEMP = C(I) - 2. * GAMMA(I) * (Z(I+1) - Z(I))
170     TEM = TEMP**2 + CI(I)**2

```

```

171      XRE = ( T * TEMP + TI * CI(I) ) / TEM
172      XIM = -( T * CI(I) - TI * TEMP ) / TEM
173      TEM = XRE**2 + XIM**2
174      CB(I) = SQRT((XRE + SQRT(TEM)) * .5)
175      CBI(I) = .5 * XIM / CB(I)
176      GAMMAI(I) = 0.
177      GO TO 33
178      29  TEMP = C(I) - 2. * GAMMA(I) * (Z(I+1) - Z(I))
179      CB(I) = SQRT ( T / TEMP)
180      GAMMAI(I) = .5 * (CI(I) - TI / CB(I)**2) / (Z(I+1) - Z(I))
181      GO TO 33
182      27  N = N - 1
183      33  CONTINUE
184      C  COMPUTE USEFULL QUANTITIES
185      PRINT 58
186      58  FORMAT (7X,6H RE M ,8X,6H IM M ,9X,5H L/KM,8X,6H RE C ,8X,
187      * 6H IM C ,5X,12H RE C BOTTOM,4X,12H IM C BOTTOM,10X,9H GRADIENT )
188      OMEGA = 6.283185307D0 * FREQ
189      DO 40 I = 1,N
190      TEMP = C(I)**2 + CI(I)**2
191      CAY(I) = OMEGA * C(I) / TEMP
192      CAYI(I) = -OMEGA * CI(I) / TEMP
193      CAY SQ(I) = CAY(I)**2 - CAYI(I)**2
194      CAY SQI(I) = 2.DO * CAY(I) * CAYI(I)
195      TEMDR = -2. * (GAMMA(I) * CAY SQ(I) - GAMMAI(I) * CAY SQI(I))
196      TEMDI = -2. * (GAMMA(I) * CAY SQI(I) + GAMMAI(I) * CAY SQ(I))
197      G CU(I) = (TEMDR * C(I) + TEMDI * CI(I)) / TEMP
198      G CUI(I) = (TEMDI * C(I) - TEMDR * CI(I)) / TEMP
199      TEM1 = DCBRT(-DSQRT( GAMMA(I)**2 + GAMMAI(I)**2) * 2.*OMEGA**2)
200      TEMI = DATAN (ABS(GAMMAI(I) / GAMMA(I)))/ 3.
201      CRTG = TEM1 * DCOS(TEMI)
202      CRTGI = TEM1 * DSIN(TEMI)
203      IF (GAMMA(I) .LT. 0.) CRTG = -CRTG
204      IF (GAMMAI(I) .LT. 0.) CRTGI = -CRTGI
205      G(I) = (C(I) * CRTG + CI(I) * CRTGI) / TEMP
206      GI(I) = (C(I) * CRTGI - CI(I) * CRTG) / TEMP
207      CON(I) = G(I) * C(I) - GI(I) * CI(I)
208      CON(I) = OMEGA**2 / CON(I)**2
209      XMI = -GI(I) * (Z(I+1) - Z(I))
210      XM = -G(I) * (Z(I+1) - Z(I))
211      DPK(I) = -8686.DO * CAYI(I)
212      PRINT 30, XM, XMI, DPK(I), C(I), CI(I), CB(I), CBI(I)
213      * ,GAMMA(I), GAMMAI(I)
214      G SQI(I) = 2. * G(I) * GI(I)
215      40  G SQ(I) = G(I)**2 - GI(I)**2
216      C  FIND MODES
217      NXTRA=0
218      IJ FLAG=0
219      NN = NN + 1
220      IF (K5 .EQ. 1) GO TO 15
221      DO 50 NN = 1,350
222      15  IF (IJ FLAG .EQ. 1) GO TO 53
223      52  IF (NXTRA .GT. 0) GO TO 44
224      READ 60, V,VI,STEP,STEP1,NXTRA
225      60  FORMAT (4D10.4,I10)
226      IF (NXTRA .GE.0) GO TO 62
227      V = V + VI * 1.0-10

```

```

228      VI = STEP + STEPI * 1.D-10
229      GO TO 85
230      62 IF (V) 142,301,70
231      142 IF(STEP) 44,44,143
232      C SEARCH FOR MODE
233      143 SIZE3 = -1.
234      SIZE2=0
235      IJ FLAG=1
236      V=-V
237      IF (NXTRA) 55,55,54
238      55 NXTRA = 20
239      54 XTRA = NXTRA
240      HOP = (STEP - V) / XTRA
241      HOPI=0.
242      IF(STEPI.NE.0.) HOPI=(STEPI-VI)/XTRA
243      DO 47 IJ = 1,NXTRA
244      CALL SETUP
245      DET = VEL
246      DETI = VELI
247      CALL DETNT(N,VEL,VELI)
248      DELTA = VEL
249      DELTAI = VELI
250      SIZE = DELTA*DELTA + DELTAI*DELTAI
251      PRINT 56, V, VI, SIZE, VEL, VELI
252      56 FORMAT (2F12.3, 3D17.5)
253      IF ((SIZE2.LT.SIZE3).AND.(SIZE.GT.SIZE2)) GO TO 45
254      46 SIZE3=SIZE2
255      SIZE2=SIZE
256      V = V + HOP
257      VI=VI+HOPI
258      GO TO 47
259      45 V = V - HOP
260      TEMP = HOP / (SIZE - SIZE2)
261      DELTAI = TEMP * (DET * VELI - DETI * VEL)
262      TEMP = .5D0 * (SIZE3 - SIZE) / (SIZE3 + SIZE - SIZE2 - SIZE2)
263      DELTA = HOP * TEMP
264      IF(HOPI.EQ.0) GO TO 49
265      VI=VI-HOPI
266      DELTAI=HOPI*TEMP
267      GO TO 49
268      47 CONTINUE
269      IJ FLAG=0
270      NXTRA=0
271      GO TO 52
272      53 SIZE3 = -1.
273      SIZE2 = 0
274      GO TO 46
275      44 NXTRA = NXTRA - 1
276      V = 3. * (PHASE V(NN-1) - PHASE V(NN-2)) + PHASE V(NN-3)
277      VI = 3. * (PHASI V(NN-1) - PHASI V(NN-2)) + PHASI V(NN-3)
278      STEP = (PHASE V(NN-1) - PHASE V(NN-2)) * .0001
279      70 CALL SETUP
280      CALL DETNT(N,DET,DETI)
281      80 FORMAT (/, 2D20.11, 4D13.4)
282      VEL = DET
283      VELI = DETI
284      DELTA = STEP

```

```

285      DELTI = STEPI
286      IF (DELTA .NE. 0.) GO TO 49
287      IF (DELTI .EQ. 0.) DELTA = .01
288      49  SIZE2 = 100.
289      RX = DET**2 + DETI**2
290      IF (K6 .LT. 3) PRINT 80, V, VI, DET, DETI, SIZE, CNTR
291      J = 0
292      48  J = J + 1
293      IF (J .GT. 15) GO TO 51
294      V = V + DELTA
295      VI = VI + DELTI
296      IF (VI) 83,84,85
297      83  DELTI = DELTI - VI
298      84  VI = 1.D-18
299      85  CALL SETUP
300      NNN = N + N - 1
301      DO 82 IA = 1, NNN
302      DO 82 IB = 1, 4
303      BI(IA, IB) = Q(IA, IB)
304      82  B(IA, IB) = A(IA, IB)
305      CALL DETNT(N, DET, DETI)
306      IF (K6 .NE. 1) GO TO 72
307      71  PRINT 81, V, VI, DET, DETI, SIZE, CNTR
308      81  FORMAT (2D20.11, 4D13.4)
309      72  IF (NXTRA .LT. 0) GO TO 51
310      TEMNR = DET * DELTA - DETI * DELTI
311      TEMNI = DETI * DELTA + DET * DELTI
312      TEMDR = VEL - DET
313      TEMDI = VELI - DETI
314      TEMDEN = TEMDR*TEMDR + TEMDI*TEMDI
315      IF (TEMDEN .EQ. 0.) GO TO 51
316      TEMRNU = TEMNR*TEMDR + TEMNI*TEMDI
317      TEMINU = TEMNI*TEMDR - TEMNR*TEMDI
318      DELTA = TEMRNU/TEMDEN
319      DELTI = TEMINU/TEMDEN
320      C * * THE NEXT CONSTANT DEPENDS ON WORD LENGTH AND SIZE OF PHASE VELOCITY * *
321      IF (ABS(DELTA) .LT. 1.D-14) GO TO 51
322      SIZE = DELTA*DELTA + DELTI*DELTI
323      IF ((SIZE.GT.SIZE2).AND.(J.GT.3)) GO TO 51
324      92  SIZE2 = SIZE * 2.
325      VEL = DET
326      VELI = DETI
327      GO TO 48
328      C FIND DEPTH FUNCTIONS
329      51  IF (INSUR .EQ. 0) GO TO 61
330      TRE = (DET**2 + DETI**2) / RX
331      IF (TRE .LT. 1E-10) GO TO 61
332      PRINT 998, NN, TRE
333      998 FORMAT (5H MODE ,I4,23H FAILED TO CONVERGE -- , E9.2)
334      GO TO 999
335      61  IF (MPCH .EQ. 0) GO TO 63
336      IF (NXTRA .LT. 0) GO TO 63
337      TEM1 = V * 1.D4
338      COL(1) = TEM1
339      TEMP = COL(1)
340      COL(2) = (TEM1 - TEMP) * 1.D10
341      TEM1 = VI * 1.D4

```

```

342 COL(3) = TEM1
343 TEMP = COL(3)
344 COL(4) = (TEM1 - TEMP) * 1.D10
345 COL(5) = -NN
346 PUNCH 64, (COL(I), I = 1,5)
347 64 FORMAT (5I10)
348 63 AF(1,NN) = A(1,3)
349 AG(1,NN) = Q(1,3)
350 BF(1,NN) = -A(1,4)
351 BG(1,NN) = -Q(1,4)
352 PHASE V(NN) = V
353 PHASI V(NN) = VI
354 IF (K6 .EQ. 1) GO TO 73
355 74 PRINT B1, V, VI, DET, DET1, SIZE, CNTR
356 73 LL = N - 1
357 IF (LL-1) 95,96,97
358 96 I = 0
359 GO TO 98
360 97 DO 110 J = 2,LL
361 I = J + J - 2
362 TEMNR = A(I,2)*AF( J-1 ,NN) - Q(I,2)*AG( J-1 ,NN)
363 TEMNI = Q(I,2)*AF( J-1 ,NN) + A(I,2)*AG( J-1 ,NN)
364 TEMDR = A(I,3)*A( I+1 ,4) - Q(I,3)*Q( I+1 ,4) -
365 1 A(I,4)*A( I+1 ,3) + Q(I,4)*Q( I+1 ,3)
366 TEMDI = Q(I,3)*A( I+1 ,4) + A(I,3)*Q( I+1 ,4) -
367 1 Q(I,4)*A( I+1 ,3) - A(I,4)*Q( I+1 ,3)
368 TEMDEN = TEMDR*TEMDR + TEMDI*TEMDI
369 TEMRNU = TEMNR*TEMDR + TEMNI*TEMDI
370 TEMINU = TEMNI*TEMDR - TEMNR*TEMDI
371 TEMP = TEMRNU / TEMDEN
372 TEMPI = TEMINU / TEMDEN
373 BF(J,NN) = -(TEMP*A( I+1 ,4) - TEMPI*Q( I+1 ,4))
374 BG(J,NN) = -(TEMPI*A( I+1 ,4) + TEMP*Q( I+1 ,4))
375 AG(J,NN) = TEMPI*A( I+1 ,3) + TEMP*Q( I+1 ,3)
376 110 AF(J,NN) = TEMP*A( I+1 ,3) - TEMPI*Q( I+1 ,3)
377 98 TEMNR = -(A(I+2,2) * AF(LL,NN) - Q(I+2,2) * AG(LL,NN))
378 TEMNI = -(Q( I+2 ,2)*AF(LL,NN) + A( I+2 ,2)*AG(LL,NN))
379 TEMDEN = A( I+2 ,3)*A( I+2 ,3) + Q( I+2 ,3)*Q( I+2 ,3)
380 TEMRNU = TEMNR*A( I+2 ,3) + TEMNI*Q( I+2 ,3)
381 TEMINU = TEMNI*A( I+2 ,3) - TEMNR*Q( I+2 ,3)
382 BF(N,NN) = TEMRNU / TEMDEN
383 BG(N,NN) = TEMINU / TEMDEN
384 95 AF(N,NN) = 0.
385 AG(N,NN) = 0.
386 C FIND NORMALIZING FACTOR
387 D(NN) = 2.12429296D0 * RHO(1)**3 / G(1)
388 DI(NN) = 0.
389 DO 111 I = 2,N
390 TEMRSP = AF( I-1 ,NN)*B( 2*I-2 ,2) - AG( I-1 ,NN)*BI( 2*I-2 ,2) +
391 1 BF( I-1 ,NN)*B( 2*I-2 ,1) - BG( I-1 ,NN)*BI( 2*I-2 ,1)
392 TEMISP = AG( I-1 ,NN)*B( 2*I-2 ,2) + AF( I-1 ,NN)*BI( 2*I-2 ,2) +
393 1 BG( I-1 ,NN)*B( 2*I-2 ,1) + BF( I-1 ,NN)*BI( 2*I-2 ,1)
394 AX1 = TEMRSP*TEMRSP - TEMISP*TEMISP
395 AX1I = TEMRSP * TEMISP
396 AX1I = AX1I + AX1I
397 TEMDR = (G(I-1)**2 + GI(I-1)**2)
398 TEMDI = G(I)**2 + GI(I)**2

```

```

399      TEMP = (RHO(I-1) / RHO(I)) / TEMDI
400      TEM1 = (ZB(I-1) * G(I-1) + ZBI(I-1) * GI(I-1)) / TEMDR
401      * -(ZT(I) * G(I) + ZTI(I) * GI(I)) * TEMP
402      TEM1I = (ZBI(I-1) * G(I-1) - ZB(I-1) * GI(I-1)) / TEMDR
403      * -(ZTI(I) * G(I) - ZT(I) * GI(I)) * TEMP
404      TEMRSP = AF(I-1, NN) * B(2*I-1, 2) - AG(I-1, NN) * BI(2*I-1, 2) +
405      1 BF(I-1, NN) * B(2*I-1, 1) - BG(I-1, NN) * BI(2*I-1, 1)
406      TEMISP = AG(I-1, NN) * B(2*I-1, 2) + AF(I-1, NN) * BI(2*I-1, 2) +
407      1 BG(I-1, NN) * B(2*I-1, 1) + BF(I-1, NN) * BI(2*I-1, 1)
408      AX2 = TEMRSP * TEMRSP - TEMISP * TEMISP
409      AX2I = TEMRSP * TEMISP
410      AX2I = AX2I + AX2I
411      TEMDR = RHO(I-1) / (G CU(I-1)**2 + G CUI(I-1)**2)
412      TEMDI = RHO(I) / (G CU(I)**2 + G CUI(I)**2)
413      TEM2 = G CU(I-1) * TEMDR - G CU(I) * TEMDI
414      TEM2I = G CUI(I) * TEMDI - G CUI(I-1) * TEMDR
415      TEMR1 = AX1 * TEM1 - AX1I * TEM1I
416      TEMI1 = AX1I * TEM1 + AX1 * TEM1I
417      TEMR2 = AX2 * TEM2 - AX2I * TEM2I
418      TEMI2 = AX2I * TEM2 + AX2 * TEM2I
419      D(NN) = D(NN) + TEMR1 / RHO(I-1) + TEMR2
420      DI(NN) = DI(NN) + TEMI1 / RHO(I-1) + TEMI2
421      111 CONTINUE
422      IF (K1 .GT. 3) DA(NN) = DSQRT((D(NN)**2 + DI(NN)**2) * FREQ /
423      * PHASE V(NN))
424      EIGEN(NN) = LAMBDA
425      EIGENI(NN) = LAMBDI
426      IF (K6 .GT. 2) GO TO 131
427      L = 0
428      K = 24
429      DO 112 I = 1, N
430      L = L + 1
431      COL(L) = SNGL(ZT(I)) * 100.
432      L = L + 1
433      COL(L) = SNGL(ZTI(I)) * 1000.
434      K = K + 1
435      COL(K) = SNGL(ZB(I)) * 100.
436      K = K + 1
437      COL(K) = SNGL(ZBI(I)) * 1000.
438      112 CONTINUE
439      PRINT 130, (COL(I), I=1, L)
440      PRINT 130, (COL(I), I=25, K)
441      130 FORMAT (4H Z = , 11(I8, I5))
442      M = N + N
443      PRINT 132, (RATIO(I), I = 1, M)
444      132 FORMAT (11(1X, 2F5.3))
445      131 DB LOSS(NN) = - LAMBDI * 8686.00
446      PHINV = V * PHASE V(NN-1) / ((V - PHASE V(NN-1)) * FREQ)
447      PRINT 109, NN, EIGEN(NN), EIGENI(NN), D(NN), DI(NN), PHINV, DB LOSS(NN)
448      109 FORMAT (3H N=, I5, 10H LAMBDA =, 2E15.7, 4H D=, 2E15.7,
449      * 12H INT RANGE =, F8.0, 6H L/K =, F8.5)
450      IF (K3 .EQ. 0) GO TO 50
451      CALL RCOEF (K3)
452      50 CONTINUE
453      C READ IN SOURCE AND RECEIVERS DEPTHS
454      301 NRT = NR
455      NR = 0

```

```

456      NN = NN - 1
457      K1P1 = K1 + 1
458      IF (K1 .NE. 3) GO TO 321
459      NR = NRT
460      GO TO 501
461 321  READ 20, SOURCE
462 320  NR = NR + 1
463      READ 20, RECVRS(NR), FINAL, STEPP
464      IF (NR.GT.50) GO TO 300
465 350  IF (RECVRS(NR) .EQ.0.) GO TO 300
466 310  IF (FINAL .EQ.0.) GO TO 320
467 330  RECVRS(NR+1) = RECVRS(NR) + STEPP
468      IF (RECVRS(NR+1) .GT. FINAL) GO TO 320
469 340  NR = NR + 1
470      IF(NR .GT. 50) GO TO 300
471      GO TO 330
472 300  PRINT 303
473 303  FORMAT (/21H SOURCE AND RECEIVERS )
474      PRINT 21,(DEPTH(I), I = 1,NR)
475 21  FORMAT (8F10.2)
476 C COMPUTE DEPTH FUNCTIONS
477      DO 500 I = 1,NN
478          LOC = 1
479          DO 305 J = 1,NR
480              IF ((J .EQ. 1) .AND. (K1 .GT. 5)) GO TO 305
481              LCTR = 0
482 380  IF((DEPTH(J) .GE. Z(LOC)).AND.(DEPTH(J) .LT. Z(LOC+1)))GO TO 360
483 371  IF (LOC .GE. N) GO TO 385
484 370  LOC = LOC + 1
485          GO TO 380
486 385  IF (DEPTH(J) .GE. Z(LOC)) GO TO 360
487 390  LOC=1
488          LCTR=LCTR+1
489          IF (LCTR .GT. 2) GO TO 305
490          GO TO 380
491 360  X1 = CAY (LOC) - EIGEN (I)
492          X2 = CAY (LOC) + EIGEN (I)
493          X3 = CAYI(LOC) - EIGENI(I)
494          X4 = CAYI(LOC) + EIGENI(I)
495          TEMP = X1 * X2 - X3 * X4
496          TEMPI = X1 * X4 + X3 * X2
497          TEMDEN = G SQ(LOC) **2 + G SQI(LOC)**2
498          ZE = (TEMP * GSQ(LOC) + TEMPI * G SQI(LOC)) / TEMDEN
499          ZE1 = (TEMPI * GSQ(LOC) + TEMP * GSQI(LOC)) / TEMDEN
500          TEM1 = ZE
501          IF (ZE .GT. -7.5) GO TO 438
502          S = CAY(LOC)
503          T = CAYI(LOC)
504          DO 437 K = 1,20
505              TEMP = S**2 + T**2
506              TEMPI = (EIGENI(I) * S - EIGEN(I) * T) / TEMP
507              TEMP = (EIGEN(I) * S + EIGENI(I) * T) / TEMP
508              ZE = ((1. + TEMP) * (1. - TEMP) + TEMPI**2) * CON(LOC)
509              ZE1 = -2. * TEMPI * TEMP * CON(LOC)
510              ZR = ZE / -7.5
511              IF (DABS(ZR-1.) .LT. 1.D-3) GO TO 438
512              S = EIGEN(I) + (S - EIGEN(I)) / ZR

```

```

513      437 CONTINUE
514      438 IF (G(LOC) .LT. 0.) GO TO 439
515          ZE = G(LOC) * (DEPTH(J) - Z(LOC)) + TEM1
516          IF (ZE .GT. -7.5) GO TO 442
517          F(J) = 1.D-12
518          FI(J) = 0.
519          GO TO 305
520      439 ZE = G(LOC) * (DEPTH(J) - Z(LOC)) + ZE
521          IF (ZE .GT. SLIM) GO TO 442
522          F(J) = 1.D-12
523          FI(J) = 0.
524          GO TO 305
525      442 ZEI =G1(LOC) * (DEPTH(J) - Z(LOC)) + ZE1
526      302 CALL HANKEL(ZE,ZEI,1)
527          F(J) = (AF(LOC,I)*H1R - AG(LOC,I)*H1I + BF(LOC,I)*H2R - BG(LOC,I)
528          1 *H2I) * RHO(LOC)
529          FI(J) = (AG(LOC,I)*H1R + AF(LOC,I)*H1I + BG(LOC,I)*H2R + BF(LOC,I)
530          1 *H2I) * RHO(LOC)
531      305 CONTINUE
532          IF (K1 .EQ. 2) GO TO 451
533          GO TO 432
534      451 PRINT 270, DEPTH(NR)
535      270 FORMAT(7H1DEPTH ,F5.1,6X,3HE-8,17X,3HE-6,17X,3HE-4,17X,3HE-2,
536          * 17X,3HE 0 )
537      432 IF (K1 .LT. 4) GO TO 431
538          IF (K1 .GT. 5) GO TO 433
539          SRES(I) = (F(1)**2 + FI(1)**2) / DA(I)
540          GO TO 500
541      431 TEMDEN = D(I)*D(I) + DI(I)*DI(I)
542          TEMRE = F(1)*D(I) + FI(1)*DI(I)
543          FD = TEMRE/TEMDEN
544          FDI = (D(I) * FI(1) - DI(I) * F(1)) / TEMDEN
545      433 DO 400 K = 2,NR
546          J = K - 1
547          L = J * NN - NN + I
548          IF (K1 .LT. 6) GO TO 435
549          FF = SRES(I) * (F(K)**2 + FI(K)**2) / DA(I)
550          GO TO 436
551      435 FF = FD * F(K) - FDI * FI(K)
552          FFI = FD * FI(K) + FDI * F(K)
553      436 UU(L) = FF
554          UUI(L) = FFI
555      452 GO TO (400,410,420,400,400,400,400), X1P1
556      C PLOT DEPTH FUNCTIONS
557      420 DO 210 II = 1,120
558          210 COL(II) = 1H
559          DO 220 II = 20,100,20
560          220 COL(II) = 1HI
561          FE = FF * FF + FFI * FFI
562          IF ((FE.GT.1E-20).AND.(FE.LT.10000.)) GO TO 240
563          GO TO 250
564      240 INT = 100.DO + 2.17147D0 * DLOG(FE)
565          COL(INT) = 1H*
566          GO TO 225
567      250 COL(2) = 1H*
568      225 PRINT 260, COL
569      260 FORMAT (120A1)

```

```

570      GO TO 400
571  C PRINT DEPTH FUNCTIONS
572      410 F MAG(J) = SQRT (FF * FF + FFI * FFI)
573          IF (FF) 430,440,450
574      430 F ANG(J) = ATAN(FFI / FF) * 57.2957795100 + 180.00
575          GO TO 400
576      440 F ANG(J) = 90.
577          GO TO 400
578      450 F ANG(J) = ATAN(FFI / FF) * 57.2957795100
579          170 FORMAT ( 10F12.4)
580          180 FORMAT(/10E12.3)
581      400 CONTINUE
582          IF (K1.EQ.1) GO TO 441
583          GO TO 500
584      441 PRINT 180, (F MAG(K), K = 1,J)
585          PRINT 170, (F ANG(K), K = 1,J)
586      500 CONTINUE
587  C CALCULATE ATTENUATION AND READ IN RANGES
588      IF ((K1 .EQ. 4).OR.(K1 .EQ. 5)) PRINT 180, (SRES(K), K= 1,NN)
589      IF (K1 .EQ. 5) PUNCH 434, (SRES(K), K = 1,NN)
590      NR=NR-1
591      IF (K2 .IT. 3) GO TO 501
592      IF (K2 .EQ. 4) K8 = 3
593      IF (K2 .EQ. 3) K8 = 2
594      K2 = 0
595      501 KX = K2 + 1
596          GO TO (561,551,551), KX
597      551 PRINT 533, NN,N, C(1), Z(2), C(2), Z(3), C(3), Z(4), C(4),
598          * SOURCE, RECVRS(40), FREQ
599      533 FORMAT (1H1, 2I5, 10F10.4)
600          ICTR=0
601          R LOS(1) = 120.
602          LEVEL(1) = 1
603          DO 562 I = 1,5
604              P LEV(I)=40.
605              J COU(I)=4
606              J COUNT(I)=-6
607      562 CONTINUE
608          IF((K2 .EQ. 2).AND.(NR .GT. 5))GO TO 772
609          GO TO 561
610      772 NR = 5
611      561 NL = NN
612          PRINT 524, NL
613      524 FORMAT (18, 13H MODES IN SUM )
614          LL = 1
615          IF (K9 .GT. 0) NL = K9
616          READ 20, RANGE, FINAL R, STEP R
617          IF (K8 .EQ. 3) PUNCH 30, RANGE, FINAL R, STEP R
618          IF (RANGE) 563,1,564
619      563 NN = NN + 1
620          READ 11, K1, K2, K3, K4, K5, K6, K7, K8, K9
621          PRINT 11, K1, K2, K3, K4, K5, K6, K7, K8, K9
622          GO TO 301
623      564 IF (FINALR .LE. 0.) GO TO 550
624          FINAL R = FINAL R + 1.D-3
625      560 IF (RANGE .GE. FINALR) GO TO 561
626      550 R ATTEN = RANGE * ATTEN - 9.9429946

```

```

627      IF (K1 .GT. 5) RATTEN = 0.00
628      IF (K1 .GT. 5) GO TO 536
629      IF (K7 .EQ. 2) RATTEN = 4.342944800 * DLOG(FREQ)
630      IF (RANGE * DB LOSS(NL) .LT. 15.04) GO TO 522
631      521 NL = NL - 1
632      522 DO 520 I = 1,NL
633          IF (K7 .LT. 2) GO TO 523
634          FMAG(I) = PHASE V(I)
635          GO TO 520
636      523 TEMP RE = EIGEN(I) * RANGE
637          TEMPIM = EIGENI(I)*RANGE
638          CALL HZERO(TEMPRE,TEMPIM)
639          HZERO2(I) = HZEROR
640          IF (K7 .EQ. 0) GO TO 520
641          FMAG(I) = HZEROR**2 + HZEROI**2
642      520 HZER2I(I) = HZEROI
643      536 L = 0
644          DO 540 J = 1,NR
645              FF = 0
646              FFI = 0
647              TEMP = 0.00
648              DO 530 I = 1,NL
649                  K = L + I
650                  IF (K1 .LT. 6) GO TO 537
651                  TEMP = TEMP + UU(K)
652                  GO TO 530
653      537 IF (K7 .EQ. 0) GO TO 534
654          TEMP = TEMP + FMAG(I) * (UUI(K)**2 + UU(K)**2)
655          GO TO 530
656      534 TEMIM = UUI(K) * HZERO2(I) + UU(K) * HZER2I(I)
657          TEMRE = UU(K) * HZERO2(I) - UUI(K) * HZER2I(I)
658          FF = FF + TEMRE
659          FFI = FFI + TEMIM
660      530 CONTINUE
661          IF (K1 .GT. 5) GO TO 535
662          IF (K7 .GT. 0) GO TO 535
663          TEMP = FF**2 + FFI**2
664      535 T RE = TEMP
665          RX = -4.3429448 * ALOG(T RE) + R ATTEN
666          R LOSS(J) = RX
667          IF (K4 .LT. 2) GO TO 545
668          T RE = -4.3429448 * ALOG(UU(K)**2 + UUI(K)**2)
669          PRINT 170, RECVRS(J), RX, T RE
670          IF (K4 .NE. 3) GO TO 545
671      545 CONTINUE
672          L = L + NN
673          IF (K8 .LT. 2) GO TO 540
674          IF (K8 .EQ. 3) GO TO 538
675          LPCH = -RLOSS(J) * 10.00 + 1400.500
676          IF (LPCH .LT. 0) LPCH = 0
677          IF (LPCH .GT. 999) LPCH = 999
678          LOSPCH(J,LL) = LPCH
679          IRNG = RANGE / 1000.00
680          IF (LL .EQ. 25) PUNCH 903, IRNG, (LOSPCH(J,LLL),LLL = 1,25)
681      903 FORMAT (15,25I3)
682          GO TO 540
683      538 CONTINUE

```

```

684      LOSS(J) = (140.05 - RX) * 10.
685      IF (LOSS(J) .LT. 0) LOSS(J) = 0
686      IF (LOSS(J) .GT. 999) LOSS(J) = 999
687      540 CONTINUE
688      GO TO (770,780,716),KX
689      C PLOT DB LOSSES
690      712 COL(15)=1HI
691      COL(39)=1HI
692      COL(63)=1HI
693      COL(87)=1HI
694      COL(111)=1HI
695      COL(27)=1HX
696      COL(51)=1HX
697      COL(75)=1HX
698      COL(99)=1HX
699      I PLACE = 135
700      DO 787 J = 1,NR
701      I PLACE = I PLACE - 24
702      783 IPLOT = P LEV(J) - R LOSS(J)
703      IF (I PLOT .GT. 10) GO TO 776
704      GO TO 777
705      776 P LEV(J) = P LEV(J) - 20.
706      J COUNT(J) = J COUNT(J) - 2
707      J COU(J)=J COU(J)-2
708      786 COL(I PLACE + 1) = 1H0
709      IF (P LEV(J) - 100.) 778,779,781
710      778 JC = J COU(J) + 1
711      COL(IPLACE) = ING(JC)
712      GO TO 783
713      779 COL(I PLACE) = 1H0
714      GO TO 782
715      781 JC = J COUNT(J) + 1
716      COL(IPLACE) = ING(JC)
717      782 COL(I PLACE - 1) = 1H1
718      GO TO 783
719      777 IF (I PLOT .LT. -9) GO TO 784
720      GO TO 785
721      784 P LEV(J) = P LEV(J) + 20.
722      J COU(J)=J COU(J)+2
723      J COUNT(J) = J COUNT(J) + 2
724      GO TO 786
725      785 IPP = I PLACE + IPLOT
726      COL(IPP) = 1H+
727      787 CONTINUE
728      GO TO 750
729      C CONTOUR LOSS FIELD
730      780 DO 590 JJ = 1,120
731      590 COL(JJ) = 1H
732      COL(31)=1HI
733      COL(61)=1HI
734      COL(91)=1HI
735      DO 640 JJ = 2,41
736      LEV = 1
737      620 IF (RLOS (JJ) .LT. CONTR(LEV)) GO TO 600
738      GO TO 610
739      600 LEV=LEV+1
740      GO TO 620

```

```

741      610 IF (LEV .EQ. LEVEL(JJ)) GO TO 640
742      650 IF (LEV .GT. LEVEL(JJ)) GO TO 660
743      GO TO 670
744      660 II = LEVEL(JJ)
745      GO TO 680
746      670 II = LEV
747      680 JJJ = 124 - 3*JJ
748      COL(JJJ) = J SMBL(II)
749      LEVEL(JJ) = LEV
750      640 CONTINUE
751      COL(1) = 1HI
752      PRINT 261, (COL(II), II = 1,119)
753      716 DO 690 JJ = 1,120
754      690 COL(JJ) = 1H
755      ICTR = ICTR + 1
756      IF (ICTR .EQ. 10) GO TO 700
757      GO TO 714
758      700 TEMP = (RANGE + 1.) / 10000.
759      IND = TEMP
760      COL(2) = ING(IND+1)
761      TEMP1 = IND
762      TFMP = (TFMP - TEMP1) * 10.
763      IND = TEMP
764      COL(3) = ING(IND+1)
765      TEMP1 = IND
766      IND = (TEMP - TEMP1) * 10.
767      COL(5) = ING(IND+1)
768      COL(4)=1H.
769      COL(6)=1HK
770      COL(7)=1HY
771      COL(8)=1HD
772      COL(9) = 1HS
773      ICTR=0
774      714 GO TO (710,712),K2
775      710 COL(31)=1HI
776      COL(61)=1HI
777      COL(91)=1HI
778      DO 720 JJ = 1,40
779      TEMP = LEVEL(JJ)
780      TEMP1 = 0.
781      830 IF (LEVEL(JJ) .GT. LEVEL(JJ+1)) GO TO 730
782      GO TO 740
783      730 II = LEVEL(JJ) - 1
784      KK = 1
785      860 EX = (CONTR(II) - R LOS (JJ) ) / (RLOS (JJ+1) - CONTR(II))
786      DO 760 LL = 1,3
787      IF (EX .LT. TEST(LL)) GO TO 800
788      760 CONTINUE
789      LL = 4
790      800 JJLL = 125 - 3*JJ - LL
791      COL(JJLL) = J SMBL(II)
792      GO TO (810,820),KK
793      810 LEVEL(JJ) = LEVEL(JJ) - 1
794      GO TO 830
795      740 IF (LEVEL(JJ) .LT. LEVEL(JJ+1)) GO TO 840
796      GO TO 720
797      840 II = LEVEL(JJ)

```

```

798      KK=2
799      GU IO 860
800      820  LEVEL(JJ) = LEVEL(JJ) + 1
801          GO TO 740
802      720  LEVEL(JJ) = TEMP
803          COL(1) = 1HI
804      750  PRINT 261, (COL(11), I1 = 1,119)
805      261  FORMAT (1X, 119A1)
806          GO TO 581
807      C PRINT DB LOSSES
808      770  PRINT 580, RANGE, (R LOSS(K), K = 1,NR)
809          LL = LL + 1
810          IF (LL .GT. 25) LL = 1
811      580  FORMAT (F9.0, 2X, 10F6.1)
812      581  RANGE = RANGE + STEP R
813          IF (KB .NE. 3) GO TO 560
814      980  PUNCH 980, (LOSS(I), I= 1,NR)
815      980  FORMAT (2613)
816          GO TO 560
817      999  STOP
818          END

```

```

1      SUBROUTINE SETUP
2      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
3      DOUBLE PRECISION LAMBDA,LAMBDI
4      COMMON /HAN/ H2R,H2I,H1R,H1I,H2PR,H2PI,H1PR,H1PI,EXPONT
5      COMMON /EXPO/ EXSUM, CNTR, RATIO(25)
6      COMMON/DETMNT/ A(25,4),Q(25,4)
7      COMMON/INPUT/ Z(12), N, OMEGA, V, VI, CON(12), GSQ(12),
8      1 CAY(12), LAMBDA, LAMBDI, G(12)
9      2,RHO(12), GI(12), G SQI(12), CAYI(12)
10     COMMON /LIMIT/ TLIM, EXPON, SLIM
11     COMMON/PARTS/ ZT(12),ZTI(12),ZB(12),ZBI(12)
12     DENOM = V * V + VI * VI
13     LAMBDA = OMEGA * V / DENOM
14     LAMBDI = -OMEGA * VI / DENOM
15     M = N - 1
16     DO 10 I = 0,M
17     IF (I .EQ. 0) GO TO 35
18     IF (ZR .GT. -7.4) GO TO 25
19     IF (G(I) .LT. 0.) GO TO 25
20     ZE = G(I) * (Z(I+1) - Z(I)) + ZE
21     IF (ZE .LT. -7.5) ZE = -7.5
22     GO TO 26
23     25 CONTINUE
24     ZE = G(I) * (Z(I+1) - Z(I)) + ZR
25     IF (ZE .LT. SLIM) ZE = SLIM
26     26 CONTINUE
27     ZQ = GI(I) * (Z(I+1) - Z(I)) + ZI
28     30 ZB(I) = ZE
29     ZBI(I) = ZQ
30     CALL HANKEL(ZE,ZQ,0)
31     ZB(I) = ZE
32     ZBI(I) = ZQ
33     RATIO(2*I) = EXPONT
34     A(2*I,1) = H2R * RHO(I)
35     Q(2*I,1) = H2I * RHO(I)
36     A(2*I,2) = H1R * RHO(I)
37     Q(2*I,2) = H1I * RHO(I)
38     A(2*I+1,1) = H2PR * G(I) - H2PI * GI(I)
39     Q(2*I+1,1) = H2PI * G(I) + H2PR * GI(I)
40     A(2*I+1,2) = H1PR * G(I) - H1PI * GI(I)
41     Q(2*I+1,2) = H1PI * G(I) + H1PR * GI(I)
42     35 CONTINUE
43     GSABS = G SQ(I+1)**2 + G SQI(I+1)**2
44     X1 = CAY(I+1) - LAMBDA
45     X2 = CAY(I+1) + LAMBDA
46     X3 =CAYI(I+1) - LAMBDI
47     X4 =CAYI(I+1) + LAMBDI
48     X = X1 * X2 - X3 * X4
49     Y = X1 * X4 + X3 * X2
50     ZT(I+1) = (X * G SQ(I+1) + Y * G SQI(I+1)) / GSABS
51     ZTI(I+1) = (Y * G SQ(I+1) - X * G SQI(I+1)) / GSABS
52     ZR = ZT(I+1)
53     ZI = ZTI(I+1)
54
55     ZE = ZR
56     ZQ = ZI

```

```

57     IF (ZR .GT. -7.5) GO TO 40
58     S = CAY(I+1)
59     T = CAYI(I+1)
60     CON = (G(I+1) * S + GI(I+1) * T) / (S**2 + T**2)
61     CON = 1. / CON**2
62     DO 36 J = 1,20
63     TEMP = S**2 + T**2
64     TEMPI = (LAMBDA * S - LAMBDA * T) / TEMP
65     TEMP = (LAMBDA * S + LAMBDA * T) / TEMP
66     ZR = ((1. + TEMP) * (1. - TEMP) + TEMPI**2) * CON(I+1)
67     R = ZR / -7.5
68     IF (DABS(R-1.) .LT. 1.D-3) GO TO 41
69     S = LAMBDA + (S - LAMBDA) / R
70     36 CONTINUE
71     41 ZI = -2. * TEMPI * TEMP * CON(I+1)
72     ZT(I+1) = ZR
73     ZTI(I+1) = ZI
74
75     40 CONTINUE
76     CALL HANKEL(ZR,ZI,0)
77     ZT(I+1) = ZR
78     ZTI(I+1) = ZI
79     RATIO(2*I+1) = EXPONT
80     IF (I .NE. 0) GO TO 45
81     A(1,3) = H2R * RHO(1)
82     Q(1,3) = H2I * RHO(1)
83     A(1,4) = H1R * RHO(1)
84     Q(1,4) = H1I * RHO(1)
85     GO TO 10
86     45 CONTINUE
87     A(2*I,3) = -H2R * RHO(I+1)
88     Q(2*I,3) = -H2I * RHO(I+1)
89     A(2*I,4) = -H1R * RHO(I+1)
90     Q(2*I,4) = -H1I * RHO(I+1)
91     A(2*I+1,3) = -H2PR * G(I+1) + H2PI * GI(I+1)
92     Q(2*I+1,3) = -H2PI * G(I+1) - H2PR * GI(I+1)
93     A(2*I+1,4) = -H1PR * G(I+1) + H1PI * GI(I+1)
94     Q(2*I+1,4) = -H1PI * G(I+1) - H1PR * GI(I+1)
95     10 CONTINUE
96     A( 2*N-2 ,4) = 0.
97     Q( 2*N-2 ,4) = 0.
98     A( 2*N-1 ,4) = 0.
99     RETURN
100    END

```

```

1      SUBROUTINE DETNT(N,DET,DETI)
2      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
3      DOUBLE PRECISION DET, DETI
4      COMMON /EXPO/ EXSUM, CNTR, RATIO(25)
5      COMMON/DETMNT/ A(25,4), Q(25,4)
6      DLOSS = 1.
7      CNTR = 0.
8      DET = A(1,3)
9      DETI = Q(1,3)
10     LIM = N + N - 3
11     DO 100 I=1,LIM,2
12     J = I
13     K = 4
14     L = I
15     M = 3
16     II = 1
17     GO TO 500
18 10    J = I + 1
19     K = 2
20     L = J
21     M = 1
22     GO TO 600
23 30    J = I + 2
24     L = J
25     II = 2
26     GO TO 600
27 40    L = I + 1
28     M = 2
29     GO TO 500
30 50    K = 3
31     M = 3
32     II = 3
33     GO TO 600
34 60    K = 4
35     IF (I .EQ. LIM) GO TO 70
36     M = 4
37     II = 4
38     GO TO 600
39 70    K = 3
40     II = 2
41     GO TO 700
42 500  C = A(L,M)*A(L,M) + Q(L,M)*Q(L,M)
43 80    B = (A(J,K)*A(L,M) + Q(J,K)*Q(L,M)) / C
44     BI = (Q(J,K)*A(L,M) - A(J,K)*Q(L,M))/C
45     GO TO (10,50), II
46 600  TD = A(J,K) - (A(L,M)*B - Q(L,M)*BI)
47     TDI = Q(J,K) - (A(L,M)*BI + Q(L,M)*B)
48     TEM = TD**2 + TDI**2
49     TEMP = A(J,K)**2 + Q(J,K)**2
50     TEMP = TEM / TEMP
51     IF (II .EQ. 2) GO TO 92
52     IF (II .EQ. 4) GO TO 92
53     Q(J,K) = Q(J,K) * 10.D-18
54     A(J,K) = A(J,K) * 10.D-18
55     IF (TEMP .GT. 10.D-35) GO TO 92
56     CNTR = CNTR + 1.

```

```
57      GO TO 90
58      92  A(J,K) = TD
59      Q(J,K) = TDI
60      90  GO TO (700,40,60,70),II
61      700 C = DET*A(J,K) - DETI*Q(J,K)
62      DETI = DET*Q(J,K) + DETI*A(J,K)
63      DET = C
64      GO TO (30,100), II
65      100 CONTINUE
66      RETURN
67      END
```

```

1      SUBROUTINE RCOEF (K3)
2      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
3      COMMON/INPUT/ Z(12), N, OMEGA, V, VI, GCU(12), GSQ(12),
4      1 CAY(12), LAMBDA, LAMBDI, G(12)
5      2,RHO(12), GI(12), G SQI(12), CAYI(12)
6      DIMENSION RR(12), RI(12), RA(12), RT(12), CYSQ(12), CYSQI(12)
7      COMMON/REFL/ AF(12,200), AG(12,200), BF(12,200), BG(12,200),
8      2 EIGEN(350), EIGENI(350),BR(25,4), BI(25,4), CB(12), CBI(12),
9      3 CAYSQ(12), CAYSQI(12), NN
10     NM = N - 1
11     I = K3
12     IF (I .GT. NM) I = NM
13     110 J = I + 1
14     K = J + 1
15     IF (NN .NE. 1) GO TO 102
16     L = I + 1
17     TEMP = CB(I)**2 + CBI(I)**2
18     CY = OMEGA * CB(I) / TEMP
19     CYI = -OMEGA * CBI(I) / TEMP
20     CYSQ(I) = CY**2 - CYI**2
21     CYSQI(I) = CY * CYI
22     CYSQI(I) = CYSQI(I) + CYSQI(I)
23     102 EL SQ = C YSQ(I) - EIGEN(NN)**2 + EIGENI(NN)**2
24     ELSQI = C YSQI(I) - 2.DO * EIGEN(NN) * EIGENI(NN)
25     TEMP = ELSQ + DSQRT (ELSQ**2 + ELSQI**2)
26     IF (TEMP .LE. 0.DO) GO TO 107
27     EL = DSQRT (TEMP + .5DO)
28     ELI = ELSQI / (EL + EL)
29     103 A = AF(I,NN)*BR(J,2) - AG(I,NN)*BI(J,2)
30     * + BF(I,NN)*BR(J,1) - BG(I,NN)*BI(J,1)
31     B = AF(I,NN)*BI(J,2) + AG(I,NN)*BR(J,2)
32     * + BF(I,NN)*BI(J,1) + BG(I,NN)*BR(J,1)
33     E = AF(I,NN)*BR(K,2) - AG(I,NN)*BI(K,2)
34     * + BF(I,NN)*BR(K,1) - BG(I,NN)*BI(K,1)
35     F = AF(I,NN)*BI(K,2) + AG(I,NN)*BR(K,2)
36     * + BF(I,NN)*BI(K,1) + BG(I,NN)*BR(K,1)
37     C = (F * EL - E * ELI) / (ELSQ + ELSQI)
38     D = -(E * EL + F * ELI) / (ELSQ + ELSQI)
39     TEMP = (A + C)**2 + (B + D)**2
40     RR(I) = (A**2 - C**2 + B**2 - D**2) / TEMP
41     RI(I) = -2.DO * (A * D - B * C) / TEMP
42     10  FORMAT (10D13.5)
43     RA(I) = 0
44     IF (CB(I) .GT. V) GO TO 104
45     RX = CB(I) / V
46     RA(I) = ACOS(RX) * 57.296
47     104 RT(I) = RR(I)**2 + RI(I)**2
48     RT(I) = 1.DO / RT(I)
49     RI(I) = -DATAN2 (RI(I), RR(I)) * 57.296DO
50     RR(I) = -4.34294DO * DLOG (RT(I))
51     IF (K3 .NE.1) GO TO 108
52     I = I + 1
53     IF (I .LT. N) GO TO 110
54     105 CONTINUE
55     PRINT 106, (RR(I), I = 1,NM)
56     PRINT 106, (RI(I), I = 1,NM)

```

```
57      PRINT 106, (RT(I), I = 1,NM)
58      PRINT 106, (RA(I), I = 1,NM)
59      106  FORMAT (9G13.4)
60      RETURN
61      108  PRINT 109, Z(L), RR(I), RT(I), RI(I), RA(I)
62      109  FORMAT (9H AT DEPTH, F7.0,8H YD, R = ,F9.4 ,7H DB, OR,D12.4,
63      * 8H, PH A = ,F9.3,16H DEGREES, GR A = ,F8.2,9H DEGREES. )
64      RETURN
65      107  EL = 0.00
66      ELI = DSQRT ( DABS (ELSQ))
67      GO TO 103
68      END
```

```

1      SUBROUTINE HANKEL (ZR, ZI, IH)
2      COMMON /HAN/ H2R,H2I,H1R,H1I,H2PR,H2PI,H1PR,H1PI,R
3      INTEGER FLPS, FLOUAD
4      DOUBLE PRECISION ZMLA2, TLIM, R
5      COMMON /LIMIT/ TLIM,EXPONT
6      DOUBLE PRECISION ZR,ZI,H2R,H2I,H1R,H1I,H2PR,H2PI,H1PR,H1PI,A0,
7      1 A,B0,B,C0,C,D0,D, C4, D4,K,K1,K2,CON4,STORE1,STORE2,
8      2 STORE3,STORE4,STORE5,STORE6,STORE7,STORE8,STORE9,STOR10,STOR11,
9      3 STOR12,STOR13,STOR14,STOR15,STOR16,STOR18,C1,C2,C3,CPR,CPI,CTHR,
10     4 CTP,FR,FI,FPR,FPI, F1R,F1I,F2R,F2I,GR,G1,GPR,GPI,G1R,
11     5 G1I,G2R,G2I,H11R,H11I,H12R,H12I,H11PR,H11PI,H12PR,H12PI,
12     6 H21PR,H21PI,H22PR,H22PI,H21R,H21I,H22R,H22I, PI,SR,SI,SPR,
13     7 SPI,S2, THR,X,XR,XI,XPR,XPI,YR,YI, ZM,ZMSQ,ZM1R,ZM1I,
14     8 EXPONT,ABK, ZRTM2R,ZRTM2I,ZRTM4R,ZRTM4I,ZRT2R,ZRT2I,ZRT2M,ZRT4R,
15     9 ZRT4I, Z32R,Z32I,STP,STOR17,STHR,C5,D5,FPI12
16     DIMENSION A(40),B(40),C(40),D(40),C4(20),D4(20),ZMLA2(40),
17     1 XPR(40),XPI(40),C5(20),D5(20),ZMLA5(20)
18     DATA (A(I),I=1,36) /
19     * -1.5507278615487157D-001, 5.16909287182905237D-03,
20     1 -7.17929565531812830D-05, 5.43886034493797596D-07,
21     2 -2.58993349758951237D-09, 8.46383495944285089D-12,
22     3 -2.01519879986734545D-14, 3.65072246352779973D-17,
23     4 -5.20045934975470047D-20, 5.97753948247666720D-23,
24     5 -5.66054875234532879D-26, 4.49249900979787999D-29,
25     6 -3.03137585006604588D-32, 1.76038086531129261D-35,
26     7 -8.89081245106713441D-39, 3.94096296589855249D-42,
27     8 -1.54547567290139313D-45, 5.39998488085741835D-49,
28     9 -1.69172458673478018D-52, 4.77888301337508527D-56,
29     A -1.22347235365465573D-59, 2.85191690828591078D-63,
30     B -6.07825428023425146D-67, 1.18901687798009614D-70,
31     C -2.14237275311729034D-74, 3.56705420099448941D-78,
32     D -5.50471327313964415D-82, 7.89545793623012643D-86,
33     E -1.05526034966989126D-89, 1.31742865127327249D-93,
34     F -1.53977168218007537D-97, 1.68834614274131072D-101,
35     G -1.74020422875830830D-105, 1.68919067050893836D-109,
36     H -1.54687790339646370D-113, 1.33859285513712678D-117/
37
38     DATA (B(I),I=1,36) /
39     * -5.6524893762022989D-002, 1.34583080385769022D-03,
40     1 -1.49536755984187802D-05, 9.58568948616588476D-08,
41     2 -3.99403728590245199D-10, 1.16784715962060000D-12,
42     3 -2.52780770480649350D-15, 4.21301284134415583D-18,
43     4 -5.57276830865629078D-21, 5.99222398780246320D-24,
44     5 -5.34066309073303316D-27, 4.00950682487464952D-30,
45     6 -2.57019668261195482D-33, 1.4231432351182437D-36,
46     7 -6.87508809232765398D-40, 2.92308167190801615D-43,
47     8 -1.10221782500302268D-46, 3.71117112795630532D-50,
48     9 -1.12255630004727929D-53, 3.06709371597617291D-57,
49     A -7.60687925589328600D-61, 1.72023501942408096D-64,
50     B -3.56156318721341813D-68, 6.77618566821426585D-72,
51     C -1.18880450319548524D-75, 1.92925106003811301D-79,
52     D -2.90462369773880309D-83, 4.06810041700112477D-87,
53     E -5.31361078500669380D-91, 6.48792525641842955D-95,
54     F -7.42155714529676224D-99, 7.96988525053346460D-103,
55     G -8.05038914195299455D-107, 7.66265861598419431D-111,
56     H -6.88468878345390325D-115, 5.84835948305632284D-119/

```

57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113

DATA (C(I), I=1,36) /
* -3.1014557230974314D-002, 6.46136608978631547D-04,
1 -6.52663241392557118D-06, 3.88490024038426856D-08,
2 -1.52349029269971316D-10, 4.23191747972142544D-13,
3 -8.76173391246671935D-16, 1.40412402443376913D-18,
4 -1.79326184474300016D-21, 1.86798108827395850D-24,
5 -1.61729964352723680D-27, 1.18223658152575789D-30,
6 -7.39359963430742897D-34, 4.00086560298021048D-37,
7 -1.89166222363130519D-40, 7.88192593179710497D-44,
8 -2.91599183566300591D-47, 9.64283014438824705D-51,
9 -2.86732980802505116D-54, 7.70787582802433108D-58,
A -1.88226515946870112D-61, 4.19399545336163351D-65,
B -8.56092152145669220D-69, 1.60677956483796775D-72,
C -2.78230227677570174D-76, 4.45881775124311176D-80,
D -6.63218466643330621D-84, 9.18076504212805399D-88,
E -1.18568578614594524D-91, 1.43198766442747010D-95,
F -1.62081229703165829D-99, 1.72280218647072522D-103,
G -1.72297448391911713D-107, 1.62422179856628689D-111,
H -1.44568028354809692D-115, 1.21690259557920616D-119/

DATA (D(I), I=1,36) /
* -2.2609957504809195D-001, 9.42081562700383155D-03,
1 -1.49536755984187802D-04, 1.24613963320156502D-06,
2 -6.39045965744392318D-09, 2.21890960327913999D-11,
3 -5.56117095057428569D-14, 1.05325321033603896D-16,
4 -1.56037512642376142D-19, 1.85758943621876359D-22,
5 -1.81582545084923127D-25, 1.48351752520362032D-28,
6 -1.02807867304478193D-31, 6.11951591098084481D-35,
7 -3.16254052247072083D-38, 1.43231001923492791D-41,
8 -5.73153269001571794D-45, 2.04114412037596793D-48,
9 -6.51082654027421986D-52, 1.87092716674546548D-55,
A -4.86840272377170304D-59, 1.15255746301413424D-62,
B -2.49309423104939269D-66, 4.94661553779641407D-70,
C -9.03491422428568780D-74, 1.52410833743010928D-77,
D -2.38179143214581853D-81, 3.45788535445095606D-85,
E -4.67597749080589054D-89, 5.90401198334077089D-93,
F -6.97626371657895650D-97, 7.73078869301746066D-101,
G -8.05038914195299455D-105, 7.89253837446372014D-109,
H -7.29777011046113744D-113, 6.37471183653139190D-117/

DATA(C4(I), I=1,19) /
* .1041666666666666666D000, -.5876374421296296294D000,
* -.2290716053934337712D001, -.5115246914604383039D001,
* -.9062847663874030839D001, -.1413420435039637896D002,
* -.2032967817611733257D002, -.2764948541118776109D002,
* -.3609376712592949187D002, -.4566262114618547916D002,
* -.5635611849738394099D002, -.6817431262327333036D002,
* -.8111724483308463849D002, -.9518494701182174927D002,
* -.1103774469890957246D003, -.1266948822584689090D003,
* -.1441391353710093869D003, -.1627327073751970376D003,
* -.1826444261146441383D003/

DATA(D4(I), I=1,19) /
* -.1458333333333333333D000, -.5242693865740740734D000,
* -.2190010740010626122D001, -.4986228080782250417D001,
* -.8910269876251375731D001, -.1396107513192384956D002,
* -.2013810597032928847D002, -.2744104880120119211D002,

```

114 * -.3586970305443466737D002, -.4542393171194671533D002,
115 * -.5610363644805757000D002, -.6790874395729851569D002,
116 * -.8083919802992951438D002, -.9489495581082038481D002,
117 * -.1100759893558574910D003, -.1263823305425793323D003,
118 * -.1438158255226323270D003, -.1624125356051020867D003,
119 * -.1826430810500566861D003/
120 DATA(C5(I), I=1,19) /
121 * -.802083333333333322D000, -.2285545023696682453D001,
122 * -.3778635359389885240D001, -.5274623160711059862D001,
123 * -.6771926170404923857D001, -.8269954941340274075D001,
124 * -.9768433620097970692D001, -.1126721374768006806D002,
125 * -.1276620743800279721D002, -.1426535892246475774D002,
126 * -.1576463088778327385D002, -.1726399730395179650D002,
127 * -.1876343942052581175D002, -.2026294344140785724D002,
128 * -.2176249668603070326D002, -.2326204842605378820D002,
129 * -.2476104285001339985D002, -.2625430055269059493D002,
130 * -.2772097924164600240D002/
131 DATA(D5(I), I=1,19) /
132 * -.677083333333333322D000, -.2202914798206278015D001,
133 * -.3712590308861503947D001, -.5218010289498877822D001,
134 * -.6721592615807936173D001, -.8224186533352991837D001,
135 * -.9726176955678393129D001, -.11227766967105787810002,
136 * -.1272907520499096141D002, -.14230176248921136010002,
137 * -.1573111965670700007D002, -.17231939820272207020002,
138 * -.1873266140557309258D002, -.2023330232819351955D002,
139 * -.2173387494048982910D002, -.2323435992394561375D002,
140 * -.2473404635295374440D002, -.2622252877088035571D002,
141 * -.2758686258346552864D002/
142 DATA (ZMLA5(I), I=1,17) / 1.E9,715., 207., 103., 47., 36.4, 27.,
143 * 22.6, 18.5, 16.6, 14.7, 14., 12.9, 12.2, 11.5, 10.8, 9.2 /
144 DATA LA2, LA5 /36,17/
145 DATA (ZMLA2(I),I=1,40) /
146 12.6944301D-12,4.7348244D-6,7.0803713D-4,9.6398932D-3,
147 24.9271494D-2,1.5267301D-1,3.5324772D-1,6.7835277D-1,
148 31.1475215D0,1.7731141D0,2.5617749D0,3.9957181D0,5.2159327D0,
149 46.6020702D0,8.1490972D0,9.8514112D0,1.2320405D1,1.4917923D1,
150 51.7163634D1,1.9540309D1,2.3030246D1,2.6446844D1,2.9292820D1,
151 63.3549744D1,3.7541246D1,4.0802980D1,4.5784933D1,5.0287133D1,
152 75.3917166D1,5.9582285D1,6.4537858D1,7.0701377D1,7.5978472D1,
153 88.0163399D1,8.6945766D1,9.2594255D1,9.983446D1,
154 91.0575193D2,1.1039828D2,1.1820169D2/
155 DATA C1 / 0.57735 026918962576D0 /
156 DATA C2 / 0.66666 66666666666D0 /
157 DATA C3 / 0.86602 540378443864D0 /
158 DATA PI / 3.14159265358979324D0 /
159 DATA FPI12 / 1.30899693899574718D0/
160 DATA CON4 / .7071067811865475244D0/
161 AO = 9.30436716929229427D-01
162 BO = 6.78298725144275871D-01
163 CO = 4.65218358464614714D-01
164 DO = 6.78298725144275871D-01
165 K = 0.85366721883895156D0
166 ZMSQ = ZR*ZR +ZI*ZI
167 RZR = ZR
168 TEMP = ZI
169 IF (TEMP .LT. 0.) TEMP = -TEMP
170 IF (RZR .LE. 0.) GO TO 51

```

```

171      IF (RZR .GT. 4.4) GO TO 120
172      TEM1 = 7. - .2632 * RZR**2
173      IF (TEMP .GT. TEM1) GO TO 120
174      GO TO 53
175      51  IF (RZR .LT. -9.) GO TO 120
176      TEM1 = 4.4 + .1375 * RZR
177      IF (TEMP .GT. TEM1) GO TO 120
178      53  FLPS = 1
179      STORE1 = ZR*ZR-ZI*ZI
180      STORE2 = 2.*ZR*ZI
181      XR = STORE1+ZR -STORE2*ZI
182      XI = STORE1*ZI +STORE2*ZR
183      DO 55 MLS=1,LA2
184      IF (ZMSQ - ZMLA2(MLS)) 62,62,55
185      55  CONTINUE
186      62  FR = AO
187      FI = 0.0
188      XPR(1) = XR
189      XPI(1) = XI
190      DO 65 M = 1,MLS
191      FR=FR+A(M)*XPR(M)
192      FI=FI+A(M)*XPI(M)
193      XPR(M+1)=XR*XPR(M)-XI*XPI(M)
194      XPI(M+1)=XI*XPR(M)+XR*XPI(M)
195      65  CONTINUE
196      GR=BO
197      GI=0.0
198      DO 72 M = 1,MLS
199      GR=GR+B(M)*XPR(M)
200      GI=GI+B(M)*XPI(M)
201      72  CONTINUE
202      X =ZR*GR-ZI*GI
203      GI=ZR*GI+ZI*GR
204      GR=X
205      SR=-C1*(GI-FI-FI)
206      SI=C1*(GR-FR-FR)
207      H2R=GR-SR
208      H2I=GI-SI
209      GO TO 317
210      120 FLPS = 0
211      ZM = DSQRT(ZMSQ)
212      ZRT2M = DSQRT(ZM)
213      IF (ZR .LT. 0.00) GO TO 125
214      ZRT2R = DSQRT (0.500 * (ZR + ZM))
215      ZRT2I = ZI / (ZRT2R + ZRT2R)
216      Z32R = ZR*ZRT2R - ZI*ZRT2I
217      Z32I = ZR*ZRT2I + ZI*ZRT2R
218      GO TO 130
219      125 ZRT2I = DSQRT (0.500 * (ZM - ZR))
220      IF (ZI .LT. 0.00) ZRT2I = -ZRT2I
221      ZRT2R = ZI / (ZRT2I + ZRT2I)
222      Z32R = ZR*ZRT2R - ZI*ZRT2I
223      Z32I = ZR*ZRT2I + ZI*ZRT2R
224      ZM1R = DABS(Z32I)
225      IF (ZM1R .LT. TLIM) GO TO 130
226      R = (TLIM / ZM1R)
227      Z32R = Z32R * R

```

```

228      R = DCBRT(R)
229      ZRT2R = ZRI2R * R
230      ZRT2I = ZRT2I * R
231      ZRT2M = ZRT2M * R
232      R = R * R
233      ZM = ZM * R
234      ZMSQ = ZM**2
235      ZR = ZR * R
236      ZI = ZI * R
237      130 ZRT4R = DSQRT (0.5D0 * (ZRT2R + ZRT2M))
238          ZRT4I = 0.5D0 * ZRT2I / ZRT4R
239          ZRTM4R = ZRT4R/ZRT2M
240          ZRTM4I = -ZRT4I/ZRT2M
241          IF (ZR .GT. 0.) GO TO 210
242          IF (ZMIR .LT. TLIM) GO TO 210
243          ABK = ABS(K2)
244          IF (Z32I .GT. 0.) GO TO 205
245          K1 = K * EXPONT
246          K2 = K / EXPONT
247          Z32I = -TLIM
248          GO TO 220
249      205 K2 = K * EXPONT
250          K1 = K / EXPONT
251          Z32I = TLIM
252          GO TO 220
253      210 K2 = C2 * Z32I
254          S2 = DEXP(K2)
255          K2 = K*S2
256          K1 = K/S2
257      220 THR = FPI12 - C2 * Z32R
258          STHR = DSIN(THR)
259          CTHR = DCOS(THR)
260          STP = -C3*CTHR +0.5*STHR
261          CTP = -C3*STHR -0.5*CTHR
262          TEMP = DABS (Z32R)
263          TEM1 = DABS (Z32I)
264          IF (TEMP .LT. TEM1) TEMP = TEM1
265      230 DO 235 ML = 1,LAS
266          IF (TEMP .GT. ZMLA5(ML)) GO TO 250
267      235 CONTINUE
268      250 CONTINUE
269          YR = Z32I
270          YI = -Z32R
271          CALL CFR (YR, YI, F2R, F2I, C4, C5, ML)
272          CPR = F2R
273          CPI = F2I
274          STORE3=K2*(ZRTM4R+F2R-ZRTM4I+F2I)
275          STORE4=K2*(ZRTM4I+F2R+ZRTM4R+F2I)
276          H22R =STORE3+CTHR-STORE4*STHR
277          H22I =STORE3*STHR+STORE4*CTHR
278          IF (ZR) 280,270,270
279      270 FLQUAD =0
280          GO TO 300
281      280 IF (ZI) 290,310,310
282      290 FLQUAD = 1
283      300 H2R = H22R
284          H2I = H22I

```

```

285      GO TO 317
286      310 FLQUAD = -1
287      YR = -Z32I
288      YI = Z32R
289      CALL CFR (YR, YI, F1R, F1I, C4, C5, ML)
290      CPR = F1R
291      CPI = F1I
292      STORE5=K1*(ZRTM4R*F1R-ZRTM4I*F1I)
293      STORE6=K1*(ZRTM4R*F1I+ZRTM4I*F1R)
294      H21R=STORE5*CTP-STORE6*STP
295      H21I=STORE5*STP+STORE6*CTP
296      H2R=H21R+H22R
297      H2I=H21I+H22I
298      317 IF (IH .EQ. 2)GO TO 80
299      60 IF (FLPS .NE. 1) GO TO 320
300      70 H1R = GR+SR
301      H1I = GI+SI
302      GO TO 362
303      320 IF (FLQUAD .LT. 0)GO TO 340
304      330 YR = -Z32I
305      YI = Z32R
306      CALL CFR (YR, YI, F1R, F1I, C4, C5, ML)
307      340 STORE7=K1*(ZRTM4R*F1R-ZRTM4I*F1I)
308      STORE8=K1*(ZRTM4I*F1R+ZRTM4R*F1I)
309      H11R=STORE7*CTHR+STORE8*STHR
310      H11I=STORE7*(-STHR)+STORE8*CTHR
311      IF (FLQUAD .LE. 0) GO TO 360
312      350 STORE9=K2*(ZRTM4R*F2R-ZRTM4I*F2I)
313      STOR10=K2*(ZRTM4I*F2R+ZRTM4R*F2I)
314      H12R = STORE9*CTP+STOR10*STP
315      H12I = STORE9*(-STP)+STOR10*CTP
316      H1R = H11R+H12R
317      H1I = H11I+H12I
318      GO TO 362
319      360 H1R = H11R
320      H1I = H11I
321      362 IF (IH .EQ. 1)GO TO 999
322      80 IF (FLPS .NE. 1) GO TO 380
323      90 FPR = C0
324      FPI = 0.0
325      DO 92 M = 1,MLS
326      FPR=FPR+C(M)*XPR(M)
327      FPI=FPI+C(M)*XPI(M)
328      92 X =-(STORE1*FPR-STORE2*FPI)
329      FPI=-(STORE1*FPI+STORE2*FPR)
330      FPR = X
331      GPR = D0
332      GPI = 0.0
333      DO 94 M = 1,MLS
334      GPR=GPR+D(M)*XPR(M)
335      GPI=GPI+D(M)*XPI(M)
336      94 SPR=-C1*(GPI-FPI-FPI)
337      SPI=C1*(GPR-FPR-FPR)
338      H2PR=GPR-SPR
339      H2PI=GPI-SPI
340      GO TO 414
341      380 YR = Z32I

```

```

342      YI = -Z32R
343      CALL CFR (YR, YI, G2R, G2I, D4, D5, ML)
344      STOR11 = K2*(ZRT4R*G2R-ZRT4I*G2I)
345      STOR12 = K2*(ZRT4R*G2I+ZRT4I*G2R)
346      H22PR=STOR11*STHR+STOR12*CTHR
347      H22PI=STOR11*(-CTHR) +STOR12*STHR
348      390 IF (FLQUAD .LT. 0) GO TO 410
349      400 H2PR = H22PR
350      H2PI = H22PI
351      GO TO 414
352      410 YR = -Z32I
353      YI = Z32R
354      CALL CFR (YR, YI, G1R, G1I, D4, D5, ML)
355      STOR13 = K1*(ZRT4R*G1R-ZRT4I*G1I)
356      STOR14 = K1*(ZRT4R*G1I+ZRT4I*G1R)
357      H21PR=STOR13*(-STP) -STOR14*CTP
358      H21PI=STOR14*(-STP) +STOR13*CTP
359      H2PR = H21PR+H22PR
360      H2PI = H21PI+H22PI
361      414 IF (IH .EQ. 2) GO TO 999
362      100 IF (FLPS .NE. 1) GO TO 420
363      110 H1PR = GPR+SPR
364      H1PI = GPI+SPI
365      GO TO 999
366      420 IF (FLQUAD .LT. 0) GO TO 440
367      430 YR = -Z32I
368      YI = Z32R
369      CALL CFR (YR, YI, G1R, G1I, D4, D5, ML)
370      440 STOR15 = K1*(ZRT4R*G1R -ZRT4I*G1I)
371      STOR16 = K1*(ZRT4R*G1I +ZRT4I*G1R)
372      H11PR = STOR15*STHR -STOR16*CTHR
373      H11PI = STOR15*CTHR +STOR16*STHR
374      450 IF (FLQUAD .GT. 0) GO TO 470
375      460 H1PR = H11PR
376      H1PI = H11PI
377      GO TO 999
378      470 STOR17 = K2*(ZRT4R*G2R -ZRT4I*G2I)
379      STOR18 = K2*(ZRT4R*G2I +ZRT4I*G2R)
380      H12PR = STOR17*(-STP) +STOR18*CTP
381      H12PI = STOR17*(-CTP) -STOR18*STP
382      H1PR = H12PR+H11PR
383      H1PI = H12PI+H11PI
384      999 CONTINUE
385      RETURN
386      END

```

©PRT,S J.CFR

```
1      SUBROUTINE CFR(X, Y, SR, SI, A, B, M)
2      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
3      DIMENSION A(1), B(1)
4      SR = 0.00
5      SI = 0.00
6      DO 10 J = 1, M
7          I = M - J + 1
8          TEMR = X + SR + B(I)
9          TEMI = Y + SI
10         TEMP = A(I) / (TEMR**2 + TEMI**2)
11         SR = TEMR * TEMP
12         SI = -TEMI * TEMP
13     CONTINUE
14     SR = SR + 1.00
15     RETURN
16     END
```

```

1      SUBROUTINE HZERO(PARTR,PARTI)
2      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
3      DOUBLE PRECISION PARTI, PARTR, K2, IMZ12
4      COMMON /AHZERO/ HZEROR,HZEROI
5      D2 = PARTR**2 + PARTI**2
6      K2 = .7978845608*EXP(PARTI)
7      D = SQRT(D2)
8      RLZ12 = (SQRT((PARTR + D)/2.))/D
9      IF(D - PARTR) 9,9,10
10     IMZ12 = 0
11     GO TO 11
12     IMZ12 = (SQRT((D - PARTR)/2.))/D
13     COST = COS(PARTR)
14     SINT = -SIN(PARTR)
15     HZEROR = K2*(RLZ12*COST - IMZ12*SINT)
16     HZEROI = K2*(IMZ12*COST + RLZ12*SINT)
17     RETURN
18     END

```

APPENDIX B: SAMPLE RUN

This appendix gives a brief discussion of the input-output, then lists an input deck and shows the resulting output. The input deck is really three separate runs that are stacked to run consecutively. The input to a single run consists of several parts given in table B1. The table gives the number of cards and the location of the FORTRAN input statements in Program MAIN. The last three of these are open-ended. That is, more modes, receiver depths, or ranges are read in until a blank card specifies the end of the list. A blank range card sends the program back to the beginning. A negative range sends the program back to read a new source and new receivers after reading another key card. The program halts when a blank "n and freq" card is encountered.

Table B2 gives most of the functions of the key card by which integers are read into control keys 1-9. Some of these will require additional information, which is read in immediately following the key card.

The output of the program is usually printed through FORTRAN print statements. Cards are also punched when key 5 is 10 or key 8 is 2, 3, or 4. In the first case each card contains a complete eigenvalue that can be read into future runs.

When key 8 = 2, propagation losses for 25 consecutive ranges per card are punched for each receiver depth, with a maximum of 5 receiver depths. The losses can be read into a plot program with a format of (5X,25F3.1). Each loss must then be subtracted from 140. This format allows losses to tenths of a dB from 40.1 to 140.0 dB.

When key 8 is 3, losses for up to 26 receiver depths are punched on one card for each range. These cards can be used in a contour plotting program. They can be read with a format of (26F3.1) and must also be subtracted from 140.

Table B1. Input cards to the normal mode program.

Input	Function	Number of Cards	Location in Program MAIN
Control keys	Selects options	1 or more	37-65
n and freq	Determines number of layers and frequency; also halts program	1	66
Profile	Specifies depths, sound speeds, gradients, attenuations, and densities	7	71-85
Modes	Searches for or specifies modes	1 or more	224
Source and receivers	Specifies a source depth and one or more receiver depths	2 or more	461-463
Ranges	Specifies a sequence of ranges; also directs continuation	1 or more	616

Table B2. Functions of the control keys.

Key	Setting	Effect	Function Affected
1	0	No output Print Plot on printer	Depth functions
2	0 1	Print losses Contour on printer	Propagation losses
3	0 1 $k > 1$	No output Print all interfaces Print interface k	Reflection coefficients
4	$k > 0$	Change levels and symbols	Contour on printer
5	0 1 $k + 10$	Sum only those given Add to existing sum Punch modes on cards	Number of modes
6	0 1 2	Long print Short print Shortest print	Steps in mode iteration
7	0 1	Phased addition Random-phase addition	Mode sum
8	1 2 3	Change T-lim Punch losses for up to 5 receivers Punch losses for up to 26 receivers	Loss plot input Contour plot input
9	0 k	No effect Use only 1st k modes	Number of modes

The first profile in the input-output is a surface duct, 100 m deep. For the 500 Hz frequency, 3 modes are found by searching from a phase velocity of 1520.5–1523 m/s. Two additional modes are found by extrapolation. Forty receiver depths are then specified from 3 to 120 m, and propagation loss contours are drawn for a source at a depth of 20 m. The modes are added in random phase, and loss contours of 80, 90, and 100 dB are requested to be represented by the symbols 8, 9, and 0. A negative range then causes the program to read new control keys, source and receivers. The depth functions are then printed out as amplitudes and phase angles and propagation losses are computed.

The second profile consists of two negative gradients over two layers of sediments in shallow water. A velocity discontinuity exists at the top of each sediment layer. Negative numbers in the input for the attenuation at the bottom of the sediment layers serve as flags to request that the gradients at the top of the layers have no imaginary parts and that the attenuation at the bottom will be whatever results from this. The change in $\text{Im}C$ from 37.9 to 23.7 in the deeper sediment layer indicates that the attenuation changed by about 60 percent through the layer.

A final layer of negative sound speed gradient must always be added. A gradient of -0.1 is chosen here for the top of this layer.

The first three modes are determined by reading in approximate values. The magnitudes of the depth functions are plotted on a log scale at 2-m depths from 30 to 80 m. Reflection coefficients are computed at interface 2, which is the water-sediment interface.

The final profile is a deep-water profile with a 40-m deep sediment layer. The attenuation increases from 2 dB/km to 2.5 dB/km through this layer. The first mode, the first bottom-reflected mode, and a higher bottom-reflected mode are found. Each step of the mode iteration is printed out. Reflection coefficients are again computed. The amplitudes and phase of the depth functions are printed out at 500-m depth and at each even 1000-m depth for a 100-m source depth.

On the last two profiles, a much larger set of modes is required to compute correct propagation losses.

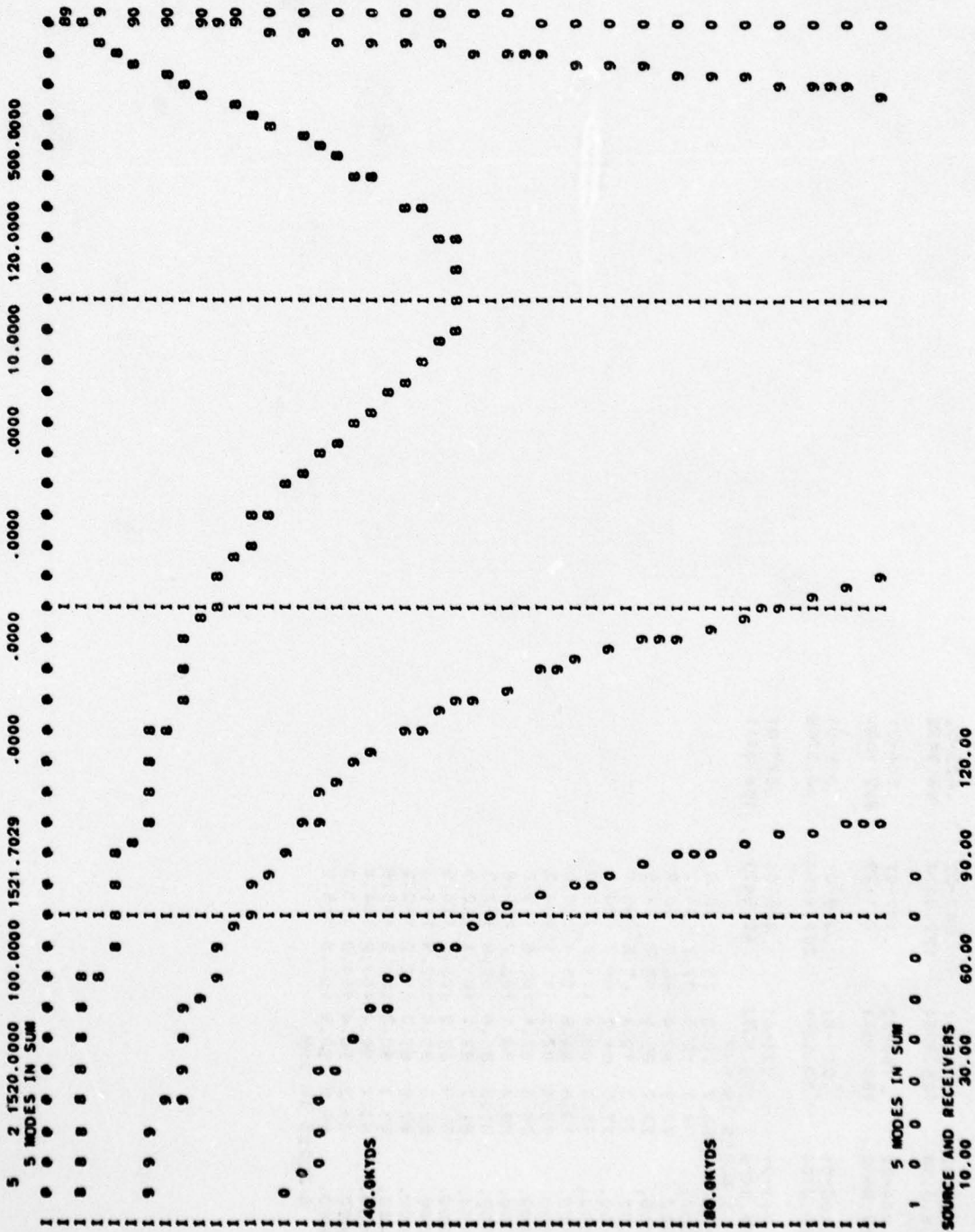
The sample run given here required 6 seconds on a UNIVAC 1110, Exec 8 operating system. The cost of the run was \$1.20.

NORMAL MODE(0).INPUT

1 INPUT DECK STARTS AT LINE 3, ENDS AT LINE 65.
2 123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
3 1 1 1 1 2
4 100. 90. 80. -1000.
5 0
6 098
7 2 500.
8 100.
9 1520.
10 0
11 .017 -.1
12 0
13 0
14 0
15 -1520.5 1523. 30
16 -1. 2
17 0
18 10.
19 3. 120. 3.
20 0
21 4000. 100000. 4000.
22 -1.
23 1
24 10.
25 30. 120. 30.
26 0
27 5000. 100000. 5000.
28 0
29 2 2 1
30 5 1500.
31 0. 51. 73. 73.3 373.3
32 1542.2 1536.8 1606.45 1684.
33 1523.42
34 1.5 1.5 -.1
35 .12 .73 .73
36 -1. -1.
37 1.68 1.91 1.91
38 1527.18 .16
39 1530.64 .13
40 1533.49 .11
41 0
42 60.
43 30. 80. 2.
44 0
45 0
46 1 6
47 8 100.
48 55. 146. 402. 960. 2286. 4390. 4430.
49 1544.9 1542.6 1517.9 1495.0 1483.2 1497.8 1541.7
50 1533.4
51 1. -.1
52 .02 .025
53 .025
54 1.54 2.5
55 -1483.5 1484.5 10
56 -1533.4 .1 1534.4 10

57	-1600.	.2	1602.	10
58	0			
59	100.			
60	500.			
61	1000.	5000.	1000.	
62	0			
63	0			
64	0			
65	0			
66	123456789	123456789	123456789	123456789

Z = 148 564 0 0
 M = 3 LAMBDA = .2063772+01
 1522.341 .211 -23202-003 -2860812-03 D = -2457285+02 .8311493+01 INT RANGE = 6385. L/M = 2.48490
 1522.424 .211 -75319-003 -27433-001 -82123-003
 1522.508 .211 -13283-002 -36444-001 -77326-003
 1522.591 .211 -17893-002 -42300-001 -39025-003
 -15230464302+004 .53327340407+000 -2374-001 -1860-001 -1789-002 .0000
 -15233048329+004 .50934840311+000 .8377-015 -.3968-015 -3366-024 .0000
 Z = 885 1360 131 418
 Z = 428 1360 0 0
 M = 4 LAMBDA = -.2062353+01
 -6895903-03 D = -.1425623+02 .5114411+01 INT RANGE = 4428. L/M = 5.98978
 .15246739840+004 .92837597939+000 -.7307-002 -.2996-003 -3366-024 .0000
 .15247283934+004 .88008242577+000 .1040-014 -.1501-014 .1148-016 .0000
 Z = 1265 2344 248 721
 Z = 808 2344 0 0
 M = 5 LAMBDA = .2060427+01
 -1189291-02 D = -.9215171+01 .3160372+01 INT RANGE = 3263. L/M = 10.33018
 SOURCE AND RECEIVERS
 10.00 3.00
 24.00 27.00
 48.00 51.00
 72.00 75.00
 96.00 99.00
 120.00



.151-01	.716-02	.143-02	.448-03
.1725	-.0318	-4.0388	-78.8365
.625-02	.128-01	.847-02	.403-02
5.3138	188.4864	172.8414	66.9433
.244-02	.768-02	.197-01	.146-01
269.9452	150.1511	2.1129	222.9555
.138-01	.221-01	.316-01	.358-01
209.0704	22.5897	201.6754	14.5748
.213-01	.221-01	.455-01	.677-01
186.2405	-70.1573	42.6473	159.3411
5 MODES IN SUM			
5000.	68.7	70.3	75.1
10000.	74.3	72.1	77.7
15000.	72.7	80.4	86.0
20000.	74.2	85.2	89.7
25000.	77.3	79.2	87.6
30000.	77.5	81.7	91.7
35000.	77.3	88.4	111.0
40000.	78.8	84.8	95.4
45000.	79.9	84.1	95.1
50000.	79.8	87.1	102.6
55000.	80.4	88.1	103.2
60000.	81.3	86.8	98.8
65000.	81.7	87.7	101.1
70000.	81.9	89.2	105.1
75000.	82.5	89.0	102.5
80000.	83.0	89.0	102.2
85000.	83.3	90.0	104.5
90000.	83.7	90.4	104.8
95000.	84.2	90.4	104.0
100000.	84.5	90.9	104.8
5 MODES IN SUM			

KEYS 2 0 2 0 0 1 0 0 0

5 LAYERS. 1500.0 MZ

1542.20000	51.00000	73.00000	73.30000	373.30000
.00000	1523.42000	.00000	1684.00000	.00000
.00000	.00000	1.50000	.00000	.00000
.00000	.00000	.12000	.73000	.73000
.00000	.00000	-1.00000	.00000	.00000
.00000	.00000	1.68000	1.91000	1.51000

ATTEN = .99703-001DB/KM

RE M	IM M	L/KM	RE C	IN C	RE C BOTTOM	IM C BOTTOM	GRADIENT
-8.81029	.00000	.00000	1542.20000	.00000	1536.80000	.00000	-.10644
-6.84816	.00000	.00000	1536.80000	.00000	1523.42000	.00000	-.61622
.12031	-.00042	180.00229	1606.00000	5.67131	1606.45019	5.67131	1.50000
114.58114	-2.58229	1095.01391	1684.00000	37.55189	2467.43825	23.72336	1.50000
39.43319	-1.30230	1095.01391	2467.43825	81.52581	.00000	.00000	-.10000

M= 1 LAMBDA = .6171335+01 -1.9028 DB, OR .6452+000, PH A = 113.538 DEGREES, GR A = 4.02 DEGREES.

M= 2 LAMBDA = .6157371+01 -2.236 DB, OR .6059+000, PH A = 96.596 DEGREES, GR A = 5.57 DEGREES.

M= 3 LAMBDA = .6145950+01 -2.2234 DB, OR .5993+000, PH A = 87.310 DEGREES, GR A = 6.57 DEGREES.

AT DEPTH 73. YD. R = 1.9028 DB, OR .6452+000, PH A = 113.538 DEGREES, GR A = 4.02 DEGREES.

AT DEPTH 73. YD. R = 2.1761 DB, OR .6059+000, PH A = 96.596 DEGREES, GR A = 5.57 DEGREES.

AT DEPTH 73. YD. R = 2.2234 DB, OR .5993+000, PH A = 87.310 DEGREES, GR A = 6.57 DEGREES.

SOURCE AND RECEIVERS

60.00	30.00	32.00	34.00	36.00	38.00	40.00	42.00
44.00	46.00	48.00	50.00	52.00	54.00	56.00	58.00
60.00	62.00	64.00	66.00	68.00	70.00	72.00	74.00
76.00	78.00	80.00					

APPENDIX C: HANKEL FUNCTION PARAMETERS

This appendix gives the FORTRAN statements for two programs associated with the modified Hankel functions. Program PWRTRN computes the power series coefficients, d_m , from eq (57), then determines the truncation points from eq (59). The truncation points for the other three sets of coefficients can be determined by changing line 9. Different computer word lengths can be accommodated by changing line 16.

The second program, CFC, determines the asymptotic series coefficients C_m from eq (72), then determines the continued fraction coefficients as indicated by eq (81)–(83). The second set of coefficients can be determined by changing the 4 in line 11 to a 16.

```

1      PROGRAM PWRTRN
2      C ** THIS PROGRAM DETERMINS TRUNCATION POINTS FOR THE POWER SERIES.
3      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
4      DIMENSION D(50), ALOGD(50)
5      D(1) = 1.
6      ALOGD(1) = 0.
7      P = 3.
8      DO 50 I = 2,50
9      D(I) = D(I-1) / P / (P-2)
10     P = P + 3.
11     IF (D(I) .LE 0.) GO TO 50
12     ALOGD(I) = ALOG10(D(I))
13 50    CONTINUE
14     PRINT 60, D, ALOGD
15 60    FORMAT (10E12.6)
16     DH = 18.
17     M = 1
18     DO 10 K = 2,50
19 30    P = M - K
20     Z = (ALOGD(K) - ALOGD(M) + DH) / 3. / P
21     IF (P .GT. -1.1) GO TO 20
22     A = ALOGD(M) - ALOGD(M+1) - 3. * Z
23     IF (A .GT. 0.) GO TO 20
24     M = M + 1
25     GO TO 30
26 20    L = K - 1
27     MM = M - 1
28     AZ = EXP (Z * 2.3025851)
29     AZSQ = AZ * AZ
30     PRINT 40, L, MM, Z, AZ, AZSQ
31 40    FORMAT (2I5, 4E15.8)
32 10    CONTINUE
33     END

```

```

1      PROGRAM CFC
2      C ** THIS PROGRAM COMPUTES A SET OF SERIES COEFFICIENTS AND THEN
3      C ** COMPUTES THE CORRESPONDING CONTINUED FRACTION COEFFICIENTS.
4      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
5      DIMENSION COEF(21,23,3), CHECK(20), C(82), S(10), A(20), B(20)
6      C(1) = 1.
7      BOTTOM = 1.
8      TOP = 1.
9      DO 2 I = 1,45
10     X = 48 * I
11     Y = 9 * (I + I - 1)**2 - 4
12     C(I+1) = C(I) * Y / X
13     2 CONTINUE
14     PRINT 20, (C(I), I = 1,40)
15     FORMAT (5G20.9)
16     11 FORMAT (/)
17     DO 100 I = 1,11
18     COEF(I,I,3) = 0.
19     COEF(I,I+1,3) = 0.
20     COEF(I,I+2,3) = 0.
21     100 CONTINUE
22     A(1) = C(2)
23     COEF(2,2,3) = 1.
24     DO 140 I = 3,21
25     DO 110 J = 2,I
26     COEF(I,J,1) = COEF(I-1,J,3)
27     COEF(I,J,2) = COEF(I-2,J,3)
28     COEF(I,J,3) = COEF(I-1,J-1,3)
29     110 CONTINUE
30     IF (I .EQ. 3) GO TO 150
31     CON = 0.
32     AT = 0.
33     BT = 0.
34     K = I - 3
35     DO 120 J = 3,I
36     K = K + 1
37     CON = C(K) * COEF(I,J-1,3) + CON
38     AT = C(K) * COEF(I,J-1,2) + AT
39     BT = C(K) * COEF(I,J-1,1) + BT
40     120 CONTINUE
41     PRINT 160, CON, AT, BT
42     CHECK(I-2) = BT
43     A(I-2) = -(CON + C(K+1)) / AT
44     150 CONTINUE
45     CON = 0.
46     AT = 0.
47     BT = 0.
48     K = I - 2
49     DO 130 J = 3,I
50     K = K + 1
51     CON = C(K) * COEF(I,J-1,3) + CON
52     AT = C(K) * COEF(I,J-1,2) + AT
53     BT = C(K) * COEF(I,J-1,1) + BT
54     130 CONTINUE
55     PRINT 160, CON, AT, BT
56     PRINT 11

```

```

57      B(I-2) = -(CON + A(I-2) * AT + C(K+1)) / BT
58      DO 140 J = 2,I
59      COEF(I,J,3) = COEF(I,J,3) + A(I-2) * COEF(I,J,2) + B(I-2) *
60      * COEF(I,J,1)
61      140 CONTINUE
62      PRINT 20, A, B, CHECK
63      160 FORMAT (5G20.9)
64      K = -2
65      J = 0
66      DO 30 M = 1,18,3
67      J = J + 3
68      K = K + 3
69      PUNCH 200, (A(I), I = K, J)
70      PUNCH 200, (B(I), I = K, J)
71      30 CONTINUE
72      200 FORMAT (5X, 1H+, 3(E21.15, 1H, ))
73      END

```

APPENDIX D: MODE FOLLOWER PROGRAM IN FORTRAN

The FORTRAN statements of the Mode Follower program are given here. This is the main body of the program. The following auxiliary subroutines from appendix A are required: SETUP, DET, HANKEL, and CFR.

AD-A072 201

NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA
UNDERWATER SOUND PROPAGATION-LOSS PROGRAM. COMPUTATION BY NORMA--ETC(U)
MAY 79 D F GORDON
NOSC-TR-393

F/G 17/1

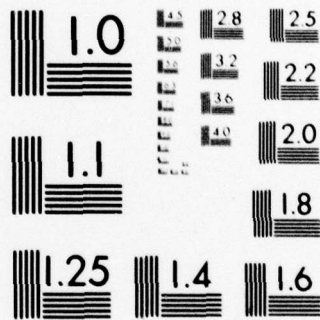
UNCLASSIFIED

NL

2 OF 2
AD
A072: 01



END
DATE
FILMED
9-79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

1      PROGRAM MFOLLO
2      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
3      COMMON /INPUT/ Z(10),N,OMEGA,V,VI,GCU(10),GSQ(10),CAY(10),LAMBDA,L
4      LAMBDI,G(10),RHO(10),GI(10),GSQI(10),CAYI(10)
5      COMMON /DETMNT/ A(21,4),Q(21,4)
6      REAL INCA, INCB, INCC, INCD, INCE, LAMBDA, LAMBDI
7      DIMENSION T(4), PV(4), W(8), WI(8), CB(10), CBI(10), C(10),
8      1 CAY SQ(10), GAMMA(10), DPK(10), GCU(10), CI(10), CR(10),PVI(4)
9      2 , CAYSQI(10), SR(4), SI(4)
10     CHNG = 1. / 8192.
11     CHNGI = 0.
12     4 CONTINUE
13 C** KO - TOTAL STEP LIMIT, K1, K2 PRINT KEYS, K3 = i KEEPS SAME PROFILE
14 C** FOR NEXT RUN.
15     READ 10, KO, K1, K2, K6, K3, TLIM, BLIM, RATIO, EX
16     PRINT 10,KO, K1, K2, K6, K3, TLIM, BLIM, RATIO, EX
17     10 FORMAT (5I4, 4E10.1)
18     IF (TLIM .EQ. 0.) TLIM = 1.E-5
19     IF (BLIM .EQ. 0.) BLIM = 1.E-2
20     IF (EX .EQ. 0.) EX = 28.
21     RLIM = 10.**EX
22     IF (RATIO .EQ. 0.) RATIO = 2.
23     IF (KO .EQ. 0) KO = 300
24     IF (K3 .NE. 0) GO TO 128
25     30 READ 1240, N,FREQ ,ATTEN
26 C** STOP IF N = 0. THIS IS THE ONLY PROGRAMED STOP.
27     IF (N.EQ.0) GO TO 1200
28     PRINT 1250, N,FREQ
29 C** PARAMETERS READ IN BELOW ARE THOSE AT THE TOP OF EACH LAYER.
30 C** READ IN VELOCITIES.
31     READ 1260, (C(I),I=1,N)
32     PRINT 1280, (C(I),I=1,N)
33 C** READ IN DEPTHS.
34     READ 1260, (Z(I),I=1,N)
35     PRINT 1280, (Z(I),I=1,N)
36 C** READ IN GRADIENTS
37     READ 1260, (GAMMA(I),I=1,N)
38     PRINT 1280, (GAMMA(I),I=1,N)
39 C** READ IN ATTENUATION FACTOR IN LOSS PER KILOMETER.
40     READ 1260, (DPK(I),I=1,N)
41     PRINT 1280, (DPK(I),I=1,N)
42 C** READ IN DENSITIES (BLANK INPUT IMPLIES SEA WATER DENSITY).
43     READ 1260, (RHO(I),I=1,N)
44     PRINT 1280, (RHO(I),I=1,N)
45     128 CONTINUE
46     NUMBER = 1
47     JX = 0
48 C** NX = VARIABLE, NY = LAYER NUMBER, NZ = CONTINUITY
49     READ 119, NX, NY, NZ, PK, VALL,DP, V, VI, STEP, STEPI
50     PRINT 21, NX, NY, NZ, PK, VALL,DP, V, VI, STEP, STEPI
51     119 FORMAT (3I2, 4X, 7D10.2)
52     21 FORMAT (10H VARIABLE ,I2, 10H LAYER NO , I2,12H CONTINUITY
53     * I2, / 7G15.5)
54     PK = PK - DP
55 C** START NEW CYCLE BY INCREMENTING VARIABLE.
56     107 PK = PK + DP

```

```

57         IF (DP) 108,999,109
58 C** CHECK IF DESIRED LIMIT OF VARIABLE HAS BEEN REACHED.
59 108 IF (PK .LT. VALL) GO TO 3
60     GO TO 133
61 109 IF (PK .GT. VALL) GO TO 3
62 133 GO TO (131,101,102,103,104,105),NX
63 131 FREQ = PK
64     GO TO 106
65 101 C(NY) = PK
66     IF (NZ .NE. 0) GO TO 106
67 134 IF (NY .EQ. N) GO TO 135
68     GAMMA(NY) = 0.
69     IF (NY .LT. 2) GO TO 106
70 135 GAMMA(NY-1) = 0.
71     GO TO 106
72 102 Z(NY) = PK
73     IF (NZ .EQ. 1) GO TO 106
74     IF (NY .LT. N) GO TO 134
75     IF (NUMBER .EQ. 1) GO TO 106
76     C(NY) = 0.
77     GO TO 106
78 103 GAMMA(NY) = PK
79     IF (NZ .NE.0) GO TO 106
80     J = NY + 1
81     DO 121 I = J,N
82     C(I) = 0.
83 121 CONTINUE
84 104 DPK(NY) = PK
85     GO TO 106
86 105 RHO(NY) = PK
87 106 CONTINUE
88
89 C** COMPLETE PROFILE **
90     DO 100 I=1,N
91 C** SET UNSPECIFIED DENSITIES TO 1.02 (SEA WATER).
92     IF (RHO(I).NE.0.) GO TO 40
93     RHO(I)=1.02
94 40     IF (I.EQ.1) GO TO 50
95 C**     COMPUTE VELOCITY AT BOTTOM OF PREVIOUS LAYER.
96     TEMP=C(I-1)**2
97     TEMDR=C(I-1)**2
98     TEMDI=(TEMDR+TEMDR+TEMDR-TEMP)*C(I-1)
99     TEMDR=(TEMDR-TEMP-TEMP-TEMP)*C(I-1)
100    TEMP=(GAMMA(I-1)+GAMMA(I-1))*(Z(I)-Z(I-1))-C(I-1)
101    TEMDEN=TEMP**2+C(I-1)**2
102    TEM1=(TEMDI+C(I-1)-TEMDR*TEMP)/TEMDEN
103    TEM1I=(-TEMDI+TEMP-TEMDR*C(I-1))/TEMDEN
104    CB(I)=SQRT(.5*(TEM1+SQRT(TEM1**2+TEM1I**2)))
105    CBI(I)=TEM1I/(CB(I)+CB(I))
106 50     IF (C(I).NE.0) GO TO 60
107 C** IF VELOCITY WAS UNSPECIFIED USE VELOCITY AT BOTTOM OF PREVIOUS LAYER
108     C(I)=CB(I)
109 60     IF (DPK(I).NE.0.) GO TO 70
110     CI(I)=0.
111     GO TO 80
112 C** IF ATTENUATION IS TO BE APPLIED TO A LAYER, COMPUTE COMPLEX VELOCITY
113 C** KEEP ABSOLUTE C EQUAL TO GIVEN REAL C FOR SIMPLICITY.

```

```

114 70 TEMP=54578.*FREQ
115 TEMDI=DPK(I)*C(I)
116 TEMDR=TEMP**2+TEMDI**2
117 CI(I)=TEMDI*TEMP*C(I)/TEMDR
118 C(I)=TEMP**2*C(I)/TEMDR
119 80 IF (GAMMA(I).NE.0.) GO TO 100
120 IF (I.EQ.N) GO TO 90
121 C** COMPUTE GRADIENT IF NOT GIVEN.
122 GAMMA(I)=(C(I+1)**2-C(I)**2)*C(I)/(2.*C(I+1)**2*(Z(I+1)-Z(I)))
123 IF (I.EQ.N) GO TO 90
124 GO TO 100
125 C** REDUCE LAYERS BY ONE IF FINAL POINT ONLY DEFINES GRADIENT IN LAST LAYER.
126 90 N=N-1
127 100 CONTINUE
128
129 C** COMPUTE USEFULL QUANTIES **
130 OMEGA=6.283185307*FREQ
131 DO 120 I=1,N
132 TEMP=C(I)**2+CI(I)**2
133 CAY(I)=OMEGA*C(I)/TEMP
134 CAYI(I)=-OMEGA*CI(I)/TEMP
135 CAYSQ(I)=CAY(I)**2-CAYI(I)**2
136 CAYSQI(I)=2.*CAY(I)*CAYI(I)
137 TEMDR=-2.*GAMMA(I)*CAYSQ(I)
138 TEMDI=-2.*GAMMA(I)*CAYSQI(I)
139 GCU(I)=(TEMDR*C(I)+TEMDI*CI(I))/TEMP
140 GCUI(I)=(TEMDI+C(I)-TEMDR*CI(I))/TEMP
141 TEMP=EXP(ALOG(GCU(I)**2+GCUI(I)**2)/6.)
142 GI(I)=TEMP*SIN(ATAN2(GCUI(I),ABS(GCU(I))))/3.)
143 G(I)=SQRT(TEMP**2-GI(I)**2)
144 IF (GAMMA(I).LT.0.) GO TO 110
145 G(I)=-G(I)
146 110 GI(I)=-GI(I)
147 C** XMI IS A LAYER STRENGTH PARAMETER USED ONLY TO COMPARE WITH OTHER MODE
148 XMI=-GI(I)*(Z(I+1)-Z(I))
149 XM=-G(I)*(Z(I+1)-Z(I))
150 GSQI(I)=2.*G(I)*GI(I)
151 120 GSQ(I)=G(I)**2-GI(I)**2
152 IF (JX .GT. 0) GO TO 113
153 C** GO TO INITIAL 3 STEPS OR TO THE STANDARD STEP.
154 IF (NUMBER - 4) 71,111,122
155 71 CALL SETUP
156 CALL DETNT(N,DET,DETI)
157 VEL=DET
158 VELI=DETI
159 DELTA=STEP
160 DELTI=STEPI
161 IF(DELTA.NE.0.)GO TO 250
162 IF(DELDI.EQ.0.)DELTA=.01
163 250 SIZE2=100.
164 IF (K6.LT.3) PRINT 1320, V,VI,DET,DETI,A(21,4),Q(21,4)
165 C** ITERATE FOR MODE UP TO 7 STEPS.
166 DO 310 J=1,12
167 V=V+DELTA
168 VI=VI+DELDI
169 C** DO NOT PERMIT IMAGINARY PART TO BECOME NEGATIVE.
170 IF (VI) 260,270,280

```

```

171      260      DELTI=DELTI-VI
172      270      VI=1.E-18
173      C** SET UP DETERMINANT FOR PHASE VELOCITY V + VI
174      280      CALL SETUP
175              CALL DETNT (N,DET,DETI)
176              IF (K6.NE.1) GO TO 300
177      PRINT 1330, V, VI, DET, DETI, SLR, SLI
178      300      TEMNR = DELTA
179              TEMNI = DELTI
180              TEMDR=VEL-DET
181              TEMDI=VELI-DETI
182              TEMDEN=TEMDR+TEMDR+TEMDI+TEMDI
183              IF (TEMDEN.EQ.0.) GO TO 320
184              TEMRNU=TEMNR+TEMDR+TEMNI+TEMDI
185              TEMINU=TEMNI+TEMDR-TEMNR+TEMDI
186              SLR = TEMRNU / TEMDEN
187              SLI = TEMINU / TEMDEN
188              IF (J .GT. 3) GO TO 125
189              SR(4-NUMBER) = SLR
190              SI(4-NUMBER) = SLI
191      125      DELTA = DET * SLR - DETI * SLI
192              DELTI = DET * SLI + DETI * SLR
193              SIZE=DELTA*DELTA+DETI*DETI
194      C** DISCONTINUE ITERATION AFTER 2ND STEP IF CORRECTION STEP IS MORE THAN
195      C** PREVIOUS STEP.
196              IF ((SIZE.GT.SIZE2).AND.(J.GT.2)) GO TO 320
197              SIZE2=SIZE*2.
198              VEL=DET
199              VELI=DETI
200      310      CONTINUE
201      320      CONTINUE
202      51      PV(4-NUMBER) = V
203              PVI(4-NUMBER)= VI
204              NUMBER = NUMBER + 1
205              GO TO 107
206      C** START STANDARD STEP, EXTRAPOLATE PHASE VELOCITY AND SLOPE.
207      111      INCA = DP
208              INCB = DP
209              INCC = DP
210      122      INCD = -INCB - INCC
211              T(1) = INCB * INCD
212              T(2) = INCB * INCC
213              T(3) = INCD * INCC
214              DO 112 IS = 1,3
215                  W(IS+4) = -SR(IS) / T(IS)
216                  WI(IS + 4) = -SI(IS) / T(IS)
217                  W(IS) = -PV(IS) / T(IS)
218              112      WI(IS) = -PVI(IS) / T(IS)
219              113      INCD = INCA + INCB
220                  INCE = INCD + INCC
221                  T(1) = INCD * INCE
222                  T(2) = INCA * INCE
223                  T(3) = INCA * INCD
224                  SLOP = 0.
225                  SLOPI = 0.
226                  SUM = 0.
227                  SUMI = 0.

```

```

228      DO 114 IS = 1,3
229      SLOP = SLOP + W(IS + 4) * T(IS)
230      SLOPI = SLOPI + WI(IS+4) * T(IS)
231      SUM = SUM + W(IS) * T(IS)
232      114 SUMI = SUMI + WI(IS) * T(IS)
233      V = SUM
234      VI = SUMI
235      CALL SETUP
236      CALL DETNT (N,DET,DETI)
237      C** EVALUATE DETERMINANT AT THE EXTRAPOLATED POINT.
238      VEL = DET
239      VELI = DETI
240      C** ITERATE FOR THE ROOT USING EXTRAPOLATED SLOPE.
241      DELTA = DET * SLOP - DETI * SLOPI
242      DELTI = DET * SLOPI + DETI * SLOP
243      IF (K1 .EQ. 1) PRINT 1330, V, VI, DET, DETI, DELTA, DELTI
244      V = V + DELTA
245      VI = VI + DELTI
246      IF (VI .GE. 0.) GO TO 124
247      DELTI = DELTI - VI
248      CHNGI = CHNGI - VI
249      VI = 0.
250      C** RE-EVALUATE AT NEW POINT.
251      124 CALL SETUP
252      CALL DETNT (N, DET, DETI)
253      TEMNR = DELTA
254      TEMNI = DELTI
255      TEMDR=VEL-DET
256      TEMDI=VELI-DETI
257      TEMDEN=TEMNR*TEMDR+TEMDI*TEMDI
258      IF (TEMDEN .EQ. 0.) GO TO 123
259      TEMRNU=TEMNR*TEMDR+TEMNI*TEMDI
260      TEMINU=TEMNI*TEMDR-TEMNR*TEMDI
261      C** EVALUATE SLOPE (RECIPROCAL ACTUALLY USED).
262      SLR = TEMRNU / TEMDEN
263      SLI = TEMINU / TEMDEN
264      DELTA = DET * SLR - DETI * SLI
265      DELTI = DET * SLI + DETI * SLR
266      IF (K1 .EQ. 1) PRINT 1330, V, VI, DET, DETI, DELTA, DELTI
267      C** CORRECT PHASE VELOCITY TO BEST VALUE.
268      V = V + DELTA
269      VI = VI + DELTI
270      TEMP = V**2 / (TEMNR**2 + TEMNI**2)
271      C** WAS INCREMENT LARGE ENOUGH TO PERMIT EVALUATION OF SLOPE.
272      IF (TEMP .LT. RLIM) GO TO 123
273      IF (TEMP .LT. 1.E34) GO TO 141
274      SLR = SLOP
275      C** IF NOT, USE EXTRAPOLATED SLOPE.
276      SLI = SLOPI
277      GO TO 141
278      123 CONTINUE
279      C** IF SO, FIND 1 - RATIO OF SLOPES.
280      TEMDEN = (SLR**2 + SLI**2)
281      TEMDR = SLR * SLOP + SLI * SLOPI - TEMDEN
282      TEMDI = SLR * SLOPI - SLI * SLOP
283      TEMP = (TEMDR**2 + TEMDI**2) / TEMDEN**2
284      IF (TEMP .GT. TLIM) GO TO 116

```

```

285 C** SLOPE RATIO TOO GOOD. DOUBLE STEP.
286 141 DP = DP * RATIO
287 GO TO 117
288 116 IF (TEMP .LT. BLIM) GO TO 117
289 PRINT 130, PK,V,VI,DET,DETI,SLR,SLI,TEMP,DBLOSS,NUMBER
290 130 FORMAT (1X,E14.6,E16.9,E13.7,6E10.3,I5)
291 C** SLOPE RATIO TOO POOR. HALVE STEP.
292 IF (NUMBER .LT. 7) GO TO 126
293 PK = PK - DP
294 DP = DP / RATIO
295 INCA = DP
296 JX = JX + 1
297 IF (K2 .EQ. 1) PRINT 118, PK, V, VI, DET, DETI
298 C** STOP ON 5 SUCCESSIVE FAILURES. MODE IS LOST.
299 IF (JX .LT. 5) GO TO 107
300 PRINT 810, N, FREQ
301 810 FORMAT (I4, G12.5)
302 3 DO 801 I = 1,N
303 PRINT 800, C(I), Z(I), GAMMA(I), DPK(I), RHO(I), G(I)
304 800 FORMAT (10G12.5)
305 801 CONTINUE
306 GO TO 4
307 126 PRINT 127, N, TEMP
308 127 FORMAT (7H NUMBER, I3, 22H FAILED, SLOPE RATIO ,F10.6)
309 C** UPDATE ALL QUANTITIES FOR NEXT STEP.
310 117 INCB = INCB
311 INCB = INCA
312 INCA = DP
313 PV(3) = PV(2)
314 PVI(3) = PVI(2)
315 PV(2) = PV(1)
316 PVI(2) = PVI(1)
317 PV(1) = V
318 PVI(1) = VI
319 JX = 0
320 DENOM = V * V + VI * VI
321 LAMBDI = -OMEGA * VI / DENOM
322 DB LOSS = -8686. * LAMBDI
323 SR(3) = SR(2)
324 SR(2) = SR(1)
325 SR(1) = SLR
326 SI(3) = SI(2)
327 SI(2) = SI(1)
328 SI(1) = SLI
329 GV = V**2 / (V - FREQ * (V-PV(2)) / INCB)
330 PRINT 118, PK,V,VI,DET,DETI,SLR,SLI,TEMP,DBLOSS,GV,NUMBER
331 118 FORMAT (E15.7,E16.9,E13.7,6E10.3,F11.5,I5)
332 NUMBER = NUMBER + 1
333 C** CHECK TOTAL NUMBER OF STEPS.
334 IF (NUMBER .GT. K0) GO TO 3
335 GO TO 107
336 999 STOP
337 1250 FORMAT (I3,8H LAYERS, ,F10.1,3H HZ)
338 1260 FORMAT(6E10.4)
339 1270 FORMAT (8H ATTEN = ,G10.5,5HDB/KM)
340 1280 FORMAT (8F14.5)
341 1320 FORMAT (/,6E18.9)

```

```
342 1330 FORMAT (6E18.9)
343 1240 FORMAT(12,E10.1, E10.2)
344 1290 FORMAT (7X,6H RE M ,8X,6H IM M ,8X,6H L/KYD,8X,6H RE C ,8X,6H IM C
345 1 ,5X,12H RE C BOTTOM,4X,12H IM C BOTTOM)
346 1200 CONTINUE
347 END
```

INITIAL DISTRIBUTION

ASST. SECRETARY OF THE NAVY (R&ES)

OFFICE OF NAVAL RESEARCH

ONR-102-OS
ONR-480
ONR-485

CHIEF OF NAVAL OPERATIONS

NOP-085
NOP-951E

CHIEF OF NAVAL MATERIAL

NMAT-08T2
ASW-122

NAVAL RESEARCH LABORATORY

CODE 8100
CODE 8120

NAVAL OCEAN RESEARCH & DEVELOPMENT ACTIVITY

CODE 320
CODE 480
CODE 500
CODE 520
CODE 600
CODE 200

NAVAL OCEANOGRAPHY COMMAND

NAVAL OCEANOGRAPHIC OFFICE

CODE 3440

NAVAL ELECTRONIC SYSTEMS COMMAND

PME-124
NELEX-320

NAVAL SEA SYSTEMS COMMAND

NSEA-003
NSEA-63R
NSEA-63R-23
NSEA-63D
PMS-407

NAVAL AIR SYSTEMS COMMAND

NAIR-370

NAVAL AIR DEVELOPMENT CENTER

NAVAL COASTAL SYSTEMS CENTER

NAVAL INTELLIGENCE SUPPORT CENTER

NAVAL POSTGRADUATE SCHOOL

NEW LONDON LABORATORY

NAVAL UNDER WATER SYSTEMS CENTER

NAVAL UNDERWATER SYSTEMS CENTER

WHITE OAK LABORATORY

NAVAL SURFACE WEAPONS CENTER

D. W. TAYLOR NAVAL SHIP RESEARCH

AND DEVELOPMENT CENTER

FLEET NUMERICAL WEATHER CENTRAL

WOODS HOLE OCEANOGRAPHY INSTITUTION

WOODS HOLE, MA 02543

MARINE PHYSICAL LABORATORY

UNIVERSITY OF CALIFORNIA, SAN DIEGO

BLDG. 106

SAN DIEGO, CA 92152

APPLIED RESEARCH LABORATORY

THE PENNSYLVANIA STATE UNIVERSITY

P.O. BOX 30

STATE COLLEGE, PA 16801

THE UNIVERSITY OF TEXAS

APPLIED RESEARCH LABORATORY

P.O. BOX 8029

AUSTIN, TX 78712

APPLIED PHYSICS LABORATORY

UNIVERSITY OF WASHINGTON

1013 N.E. 40TH ST.

SEATTLE, WA 98195

NATIONAL RESEARCH COUNCIL

ATTN: COMMITTEE UNDERSEA WARFARE

2101 CONSTITUTION AVE., NW

WASHINGTON, DC 20418

DEFENSE DOCUMENTATION CENTER

(12)