

ALGORITHMS AND INTERFACE FOR OCEAN ACOUSTIC RAY-TRACING
(Developed in MATLAB)

Technical Report No.
NIO/TR – 09/2005

T.V.Ramana Murty
M.M.Malleswara Rao
S.Surya Prakash
P.Chandramouli
K.S.R.Murthy



Regional Centre, Visakhapatnam

NATIONAL INSTITUTE OF OCEANOGRAPHY
DONA PAULA, GOA – 403 004, INDIA

CONTENTS

	Page No.
ABSTRACT	
1. INTRODUCTION	4
2. ELEMENTS OF OCEAN ACOUSTICS	6
3. RAY THEORY	8
4. COMPUTATION OF CANONICAL SOUND SPEED PROFILE	11
5. METHODOLOGY	12
6. FLOW CHARTS	16
7. SOFTWARE STRUCTURE	21
8. CASE STUDY	28
9. RESULTS	31
10. ACKNOWLEDGEMENTS	31
11. REFERENCES	32
12. APPENDIX-I CODING	33

ABSTRACT

An integrated friendly-user interactive ocean application package has been developed for computation of eigenrays for a given source – receiver configuration and subsurface and bottom environmental conditions utilizing the ray theory in MATLAB to study the Ocean Acoustic Propagation Characteristics (OAPS). This package facilitates automatic Graphical Visualization of acoustic ray parameters and generation of tabular forms of ray history for ray analysis. Results of numerical experiments conducted for Munk 's canonical sound speed profile off Visakhapatnam waters are detailed as a case study.

The simulated acoustic ray parameters of canonical ocean, off Visakhapatnam continental slope regions, reveals that the even very flat angle rays can scan entire water column with repeated reflections from both surface and bottom due to depth limited nature of sound channel and weak gradients of sound speed prevailing in the medium on either side of the sound channel axis. This nature of phenomenon usually occurs in the case of shallow-water channel.

INTRODUCTION:

Acoustic waves which travel long distances in the ocean in the form of ducted propagation with less attenuation relative to other forms of energy, are used in a large number of applications in the ocean (Urlick,R.J, 1983); for example, submarine signaling and detection, precise bathymetric measurements, sub-bottom profiling, fish finding etc. This is possible with the tracing of ray paths of sound pulses from a given source to a receiver in the ducted ocean of area of interest. Some of the common forms of ducts are: *mixed layer sound channel*, *deep sound channel* and *shallow water sound channel*. The surface sound channel (i.e., mixed layer sound channel) occurs when the upper layer exhibits isothermally leading to depth wise increase in the sound velocity, due to hydrostatic pressure. In this case the axis of the channel lies close to the sea surface. If an acoustic source is placed within this layer of constant velocity gradient, the acoustic energy gets trapped within this channel and propagation takes place along the rays that are in the form of arc of circles of radius of curvature $R=C_0/g$ where C_0 is surface sound velocity and g is velocity gradient. In the case of deep sound channel transmission experiments, sound sources and receivers are both placed preferably around the SOFAR (SOund Fixing And Ranging) channel axis (i.e., around the locus of points of minimum sound velocity). For example, in the tropical seas the sound velocity is between 1 - 1.5 km below the ocean surface. This is due to the opposing effects of temperature, which dominates in the upper layers and causes a rapid decrease in sound speed with depth, and due to pressure, which increases eventually results in an increase of sound velocity. This configuration of the sound velocity profile in the vertical section of an ocean, which traps the sound rays around the sound axis, is called sound channel or SOFAR channel. Sound rays propagating along the SOFAR channel axis are refracted downward when propagating in the region of negative sound velocity gradient above the level of minimum sound velocity and upward when propagating in the region of positive gradient below it. The important feature of sound propagation along the SOFAR channel is that energy losses due to geometric spreading and scattering associated with the boundaries are

minimized and consequently large ranges of sound propagation are possible for the same source power (Officer C.B, 1958).

In the present report, an automatic version of interface software (in MATLAB) to find out eigen rays (rays connecting between source and receiver) and related ray parameters are presented taking the example of Munk's canonical sound speed profile for continental slope waters off Visakhapatnam. This package works for nature of range independent as well as range-dependent environment with and without sloping bottom conditions.

The systematic development of methodology starting from wave equation and its numerical computations with sample example detailed in the following sections.

ELEMENTS OF OCEAN ACOUSTICS

Mathematical Formulation:

Wave Equation:

Acoustic wave propagation is an unsteady fluid flow problem related to mechanical vibration of ocean water in the range of tens of hertz to few hundred kilohertz. These vibrations are assumed to cause small departures from what is otherwise a static state in the fluid. As these vibrations are very fast and small, we ignore heat transfer between fluid elements. We also ignore viscosity since fluid motions are small. Now the equations governing this flow field are the continuity equation (Ramana Murty & Mahadevan, 1994)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{V} = 0 \quad (1)$$

and the momentum equation,

$$\rho \frac{d\mathbf{V}}{dt} = -\nabla P - \rho g \mathbf{k} \quad (2)$$

where ρ is the density of the fluid, \mathbf{V} is the velocity of a fluid element with components (u,v,w) , P is the pressure, \mathbf{k} is a unit vector in the vertical (z) direction and g is the acceleration due to gravity.

The flow variables associated with acoustic wave propagation problem, are treated as small departures from the undisturbed fluid state, and they are represented in the form $\rho = \rho_0 + \rho_1$, $\mathbf{V} = \mathbf{V}_0 + \mathbf{V}_1$ and $P = P_0 + P_1$. Here, the subscripts 'o' denotes the undisturbed fluid state (i.e. the static state with $\mathbf{V}_0 = 0$) and '1', the departures in field variables due to the presence of acoustic waves. Then, substituting these flow variables in Eqns. (1) and (2) and neglecting terms involving products of acoustic wave induced variables, we get the linearized form of the governing equations as

$$\frac{\partial \rho_1}{\partial t} + \nabla \cdot (\rho_0 \mathbf{V}_1) = 0 \quad (3)$$

and

$$\rho_0 \frac{d\mathbf{V}_1}{dt} = -\nabla P_1 \quad (4)$$

eliminating \mathbf{V}_1 between these equations, we get

$$\frac{\partial^2 \rho_1}{\partial t^2} - \nabla^2 P_1 = 0 \quad (5)$$

The frequency of acoustic oscillations is quite high and the associated pressure fluctuations in the fluid could be assumed as adiabatic. Then the equation of state is given by $P_1 = C^2 \rho_1$ and eqn. (5) can be written as (subscript is omitted for convenience).

$$\frac{\partial^2 P}{\partial t^2} - C^2 \nabla^2 P = 0, \quad (6)$$

Where C is the sound speed through the propagating medium. The above equation, referred to as the wave equation, defines the pressure field associated with acoustic waves propagating in ocean waters.

The solution to this equation (i.e. to the acoustic wave propagation problem in ocean waters) can be approached by acoustic ray theory applicable to high frequency waves or the normal mode theory. We will discuss here the ray theory, which has been successfully used in ocean acoustic tomography and other relevant sound propagation studies(REF:..).

RAY THEORY:

Ray theory is widely used in underwater acoustics since it has many advantages. Some of them are 1) Rays are easily drawn 2) Real boundary conditions are inserted easily, e.g. a sloping bottom 3) It is independent of source.

This method is based on an equation called eikonal equation. This eikonal equation is obtained from the wave equation by assuming the solution in a particular form.

We initially seek a periodic solution in time to the wave equation in the form $P(x,y,z,t) = p(x,y,z) \exp(-i\omega t)$. This reduces the wave equation to the Helmholtz equation given by,

$$\nabla^2 p + k^2 p = 0 \quad (7)$$

Where $k = \omega / C$, is the wave number. The variation in sound speed in the ocean is very small and the solution to the Helmholtz equation can be assumed in the form

$$p(x,y,z) = A(x,y,z) \exp[i k_0 W(x,y,z)], \quad (8)$$

Where A is a slowly varying function of position, k_0 is a constant reference wave number and W is called the eikonal function. The functional form of this solution can be easily recognized to be the same as that of a plane wave solution in a homogeneous medium. i.e.,

$$P(x,y,z,t) = A \exp\{i S(x,y,z,t)\} \quad (9)$$

Where A is the constant amplitude of the wave and

$$S(x,y,z,t) = k_x X + k_y Y + k_z Z - \omega t$$

Is called the phase function. Here k_x , k_y , k_z are the components of the wave number vector K of the plane wave. Note that, for a non-homogeneous medium, from eqn. (8), the phase function can be defined as

$$S(x,y,z,t) = k_0 W(x,y,z) - \omega t$$

Substituting the solution, eqn. (8), in the Helmholtz equation, the real part gives

$$\nabla^2 W + W \nabla^2 C^2 / (k_0^2 C^2) = -2 \nabla W \cdot \nabla C / (k_0 C) \quad (10)$$

The RHS of the above equation can be shown to be small when the variation of C is small over one wavelength of an acoustic wave. This condition is

satisfied in the case of high frequency sound propagation in ocean waters.

Then we get

$$\nabla W = n^2 \mathbf{r} \quad (11)$$

where the refractive index $n = C_0 / C$, is a function of position and $C_0 = \omega / k_0$.

The eqn. (11) is called the eikonal equation.

Now at any time t , $S(x,y,z,t) = \text{constant}$, implies that the eikonal function is constant. Hence $W = \text{constant}$ defines a surface of constant phase and ∇W defines the normal to the surface of constant phase or the direction of ray propagation or ray path. In eq. (11), ∇W is a vector in the direction of ray path and n is the magnitude of this vector.

Let $d\mathbf{r} = (dx, dy, dz)$ be the displacement vector along the ray path and $ds = |d\mathbf{r}|$. Then

$$n \frac{d\mathbf{r}}{ds} = n \left(\frac{dx}{ds}, \frac{dy}{ds}, \frac{dz}{ds} \right) = \nabla W \quad (12)$$

Now

$$\begin{aligned} \frac{d}{ds} \left(n \frac{dx}{ds} \right) &= \frac{d}{ds} \left(\frac{\partial W}{\partial x} \right) = \frac{\partial}{\partial x} \left(\frac{dW}{ds} \right) \\ &= \frac{\partial}{\partial x} \left(\frac{\partial W}{\partial x} \frac{dx}{ds} + \frac{\partial W}{\partial y} \frac{dy}{ds} + \frac{\partial W}{\partial z} \frac{dz}{ds} \right) \\ &= \frac{\partial}{\partial x} \left(n^2 \frac{dx}{ds} \right) = \frac{\partial}{\partial x} \left(n^2 \right) \frac{dx}{ds} + n^2 \frac{\partial}{\partial x} \left(\frac{dx}{ds} \right) \end{aligned}$$

Hence we get

$$\frac{d}{ds} \left(n \frac{dx}{ds} \right) = \frac{\partial n}{\partial x} \frac{dx}{ds} + n \frac{d}{ds} \left(\frac{dx}{ds} \right) \quad \text{and} \quad \frac{d}{ds} \left(n \frac{dz}{ds} \right) = \frac{\partial n}{\partial z} \frac{dz}{ds} + n \frac{d}{ds} \left(\frac{dz}{ds} \right) \quad (13 a, b, c)$$

These equations state that the variation along the ray path of the product of the index of refraction and the direction cosine is equal to the space rate of variation of the index of refraction with respect to the appropriate coordinate.

Eqns. (13) are the generalized form of Snell's law.

Let us consider the sound propagation problem in two dimension in x-z plane (see Fig. A1). Then along the ray path

$$\frac{dx}{ds} = \cos \theta \quad \text{and} \quad \frac{dz}{ds} = \sin \theta \quad (14 \text{ a,b})$$

Substituting eq. (14 a) in (13 a) and $\mathbf{n} = \mathbf{C}_0 / \mathbf{C}$, we get along the x-direction

$$\frac{d}{ds} \left(\frac{C_0}{C} \cos \theta \right) = \frac{1}{C} \frac{dC_0}{dx} \cos \theta - \frac{C_0}{C^2} \frac{dC}{dx} \cos \theta - \frac{C_0}{C} \sin \theta \frac{d\theta}{ds} \quad (15)$$

Similarly along the z-direction

$$\frac{d}{ds} \left(\frac{C_0}{C} \sin \theta \right) = \frac{1}{C} \frac{dC_0}{dz} \sin \theta + \frac{C_0}{C^2} \frac{dC}{dz} \sin \theta + \frac{C_0}{C} \cos \theta \frac{d\theta}{ds} \quad (16)$$

Eliminating dC/ds from eqs. (15) and (16), we get

$$\frac{d\theta}{ds} = \frac{1}{C} \left(\sin \theta \frac{dC_0}{dx} - \cos \theta \frac{dC_0}{dz} \right) - \frac{C_0}{C^2} \left(\sin \theta \frac{dC}{dx} + \cos \theta \frac{dC}{dz} \right) \quad (17)$$

Now the eqs. (14) and (17) describe the ray paths and the numerical integration of these equations for a given sound speed distribution, provides the coordinates of the ray paths.

Intensity computations—The acoustic intensity at any point along the ray path is given by the expression (Krol H.R, 1979).

$$Intensity = I_0 \frac{X}{C} \left(\sin \theta_0 / X \cos \theta \right) \left(\sin \theta / Z \right) \quad \dots(18)$$

where I_0 = reference intensity at the source; θ_0 = eigen ray angle at the source with reference to the horizontal; θ = ray angle at the point of measurement with reference to the horizontal; X=distance from the source to the point of measurement; Z = vertical distance at the point of measurement between eigen ray and ray in its immediate vicinity; and θ_0 = angle between eigen ray and ray in its immediate vicinity at the source.

At the point where Z = 0, the above expression will not hold good (tends to infinity mathematically) and the previous value only gets substituted for computational continuity.

Considering the phase shift for any given frequency the intensity has been computed following (Moler C B & Soloman L.P, 1970).

$$\frac{I}{I_0} = \left| \sum_j P_j e^{i2\pi f t_j} \right|^2 \quad \dots(19)$$

where $P_k = \sqrt{I_k/I_0}$ and t_k are the relative pressures and arrival times of the eigen rays. These have wider applications in signal detection.

COMPUTATION OF CANONICAL SOUND SPEED PROFILE :

For simulation studies a parametric representation of the sound-speed profile is needed. In the present study, we construct a representative sound speed profile of waters of Visakhapatnam in an analytical form following Munk'(1974).

The mathematical expression for generation of canonical sound speed profile has been detailed. The theory enables us to present the sound-speed profile at any geographical location with the help of a few parameters obtained from hydrographic data, apart from its implication in ray theory and normal mode theory.

The Canonical expression for the sound velocity profile C(Z) (Munk,1974) expressed as function of a depth, stratification scale and perturbation coefficient as follows:

$$C(Z) = C_1 \left[1 - \exp(-2W/B) \right] \quad (20)$$

where $W = Z - Z_1$, (Z_1 is depth of sound channel axis)

$$B = a/b \text{ (Stratification scale),}$$

$$a = a^3/b^2 \text{ (Perturbation coefficient)}$$

$$a = \frac{1}{2} \frac{r_1^2 r_3^2 + r_2^2 r_2^2}{r_1^2 r_3^2 + r_2^2}, \quad b = \frac{3}{4} \frac{r_1^2 r_2^2 + r_2^2 r_1^2}{r_1^2 r_3^2 + r_2^2}$$

$$r_1 = \int_{Z_1}^Z W^4 dW, \quad r_2 = \int_{Z_1}^Z W^5 dW, \quad r_3 = \int_{Z_1}^Z W^6 dW, \quad 21 \text{ (a,b,c)}$$

$$r_1 = \int_{Z_1}^Z f(W) W^2 dW \quad 21 \text{ (d)}$$

$$r_2 = \int_0^1 f(W) W^3 dW \quad (21) \text{ (e)}$$

The above integrals can be evaluated using natural cubic spline interpolation technique for given sound speed profile, depth of sound channel axis, speed of sound at channel axis and upper and lower bounds of sound speed profile to get stratification scale and perturbation coefficient.

METHODOLOGY

Numerical ray tracing --- The solution of ray tracing with initial conditions specifying any given point (source coordinates and direction) on the ray is usually known as the initial-value problem.

The numerical computations in the form of algorithm (Ramana Murty et.al, 1990) for Eqs 14(a,b) and 17 are :

$$\begin{aligned} X_{ij}^1 &= f_i(X_1, X_2, X_3, X_4) \\ X_i &= X_i + ds \sum_{j=1}^4 (Z_j - B_j Z_j') A_j \quad \dots(22) \\ Z_i' &= Z_i + 3 \sum_{j=1}^4 (Z_j - B_j Z_j') A_j + C_j Z_j \end{aligned}$$

where $i = 1, 2, 3, 4$; $Z_1 = 1$; $Z_2 = \cos \theta$; $Z_3 = \sin \theta$; $Z_4 = \cos \theta V_z / V$;

and with initial conditions of

$$X_1(0) = 0$$

$$X_2(0) = X_0 \text{ (source X coordinate)}$$

$$X_3(0) = Z_0 \text{ (source Y coordinate)}$$

$$X_4(0) = \theta_0 \text{ (horizontal angle)}$$

the coefficients of A_j , B_j and C_j being:

$$A_1 = 1/2, B_1 = 2, C_1 = 1/2$$

$$A_2 = 1 - \sqrt{1/2}, B_2 = 1, C_2 = 1 - \sqrt{1/2}$$

$$A_3 = 1 - \sqrt{1/2}, B_3 = 1, C_3 = 1 - \sqrt{1/2}$$

$$A_4 = 1/6, B_4 = 1, C_4 = 1/2$$

In the present study Runge-Kutta-Gill's method has been chosen to obtain the numerical solution. For the present analysis, step-size (ds) is chosen depending on the medium at the end point of the ray segment. In the

region of thermocline or halocline, ds is considered small compared to that within the layers further below and characterized by high gradients in sound speed. The changes in the range(dx), depth(dz) and the angle($d\theta$) are computed. During integration, if the ray path does not pass through the nodal points which would define sound speed field for computing the gradients and the refractive index, they may be restricted to pass through arbitrary points in the plane. In such case, suitable interpolation function has to be defined to evaluate both sound speed and its partial derivatives for a given ds . A 3-point Lagrangian interpolation is used here to construct such a function along the vertical and in the neighborhood of the sound speed field taking 2 points above and one point below and vice versa, prior to averaging. Knowing the information along ds and at arbitrary points, integration is carried out using the Runge-Kutta-Gill's technique. At the upper and lower boundaries, the ray gets reflected following equal angle law. Irregular bathymetric features and the curvature of the earth have not been considered here. The integration repeats until all rays within a given beam arrive at the receiver. If not, the algorithm takes on computation of the ray having the next assigned angle and follows the same procedure ('Shooting method'). The ray co-ordinates are kept in memory space allocated in a matrix form. The time of flight (T_i) is determined following the recurrence formula

$$T_i = T_i + \sum_{j=1}^4 \left(Y_2 + B_j Z_2 \right) \left(Y_3 + B_j Z_3 \right) A_j ds \quad \dots(22)$$

where i covers full range.

Once the eigen rays are identified from the ray computation, the intensity calculations are made following the ray tube method.

Generation canonical sound speed profile:

The collected hydrographic data on temperature, salinity off waters of Visakhapatnam during March, 11-20, 1987 on ORV Sagarkanya has been used to generate canonical sound speed profile based on algorithm developed by Ramana Murty et.al, 1989 as a input profile for conducting numerical acoustic modeling. Initially the data was subjected to cubic spline (for continuous data) and weighted parabolic (for discrete data) interpolation (Reininger and Ros 1968) to obtain observations at closely spaced depths (10m) intervals. From these data the sound speed has been computed following Chen Millerio(1977) to obtain the sound speed at the channel axis (C_1), the depth of the channel axis (Z_1) and upper and lower bounds of integration (Z_2, Z_3) for the integrals appearing in Eqs.21(a-e) for computation of the stratification scale B and perturbation coefficient β .

Computed Canonical sound channel model parameters:-

$Z_2 = 250(m), Z_3 = 2490(m), Z_1 = 1490(m), C_1 = 1493.72(m/s)$

$B = 1.4148339(km), \beta = 6.27009 \times 10^{23}$

Using Canonical sound channel model parameters the following Munk's canonical sound speed profile is generated using Eq. 20

Sound speed profile (Here column 1 is Depth(m) and column 2 is Sound speed(m/sec))

Before starting of the profile (-1 x) is been given which are detailed below:

'-1' is the flag which indicates starting of the profile.

'x' indicates the range of the profile with reference to the previous profile

File should contain two sets of profiles.

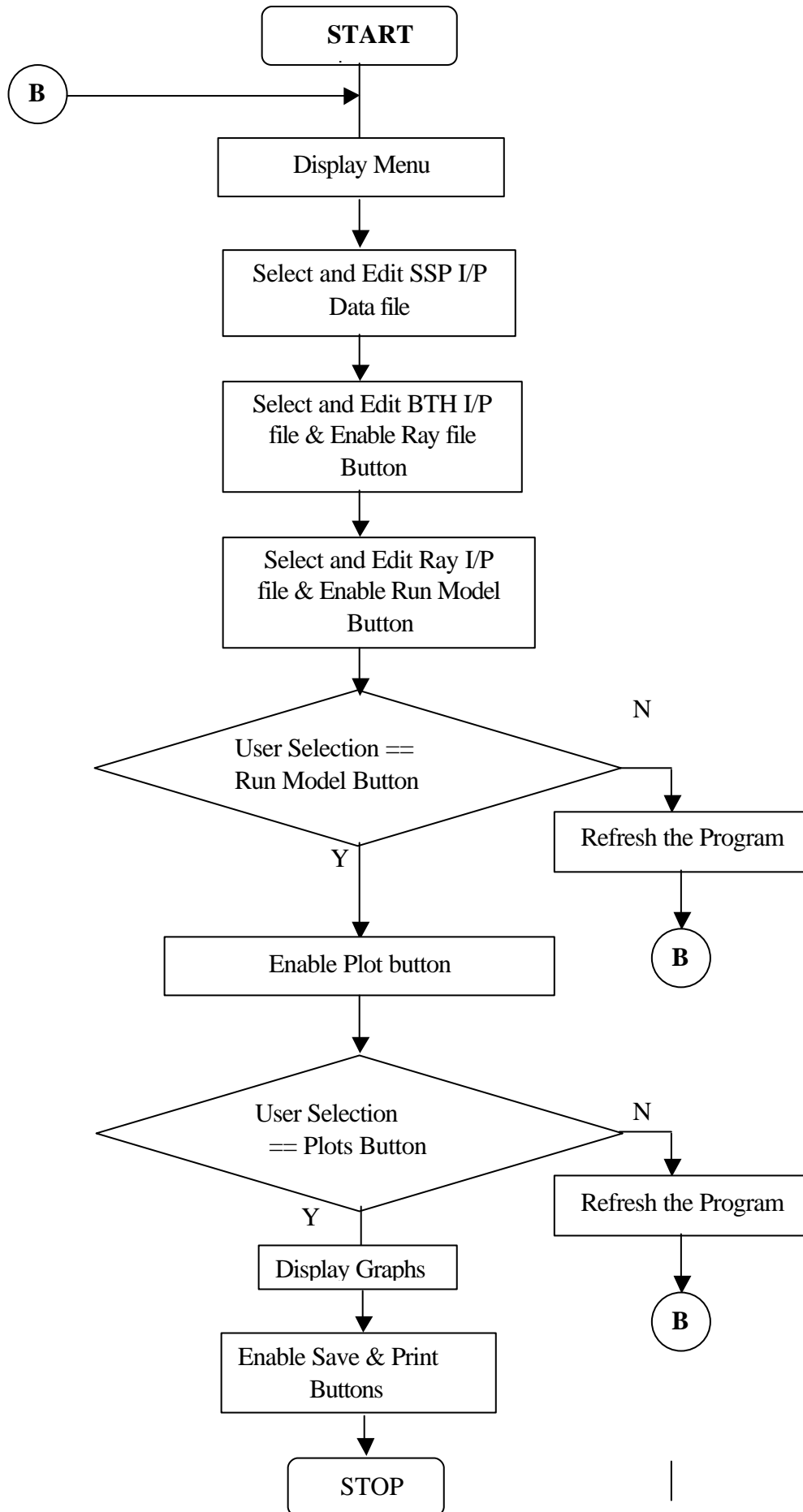
- (i) Range Independent – In this case the profile should be repeated twice
- (ii) Range Dependent -- In this case the profiles are to be followed in range dependent nature.

Depth(m)	Speed(m/sec)	Depth(m)	Speed(m/sec)	Depth(m)	Speed(m/sec)	Depth(m)	Speed(m/sec)	Depth(m)	Speed(m/sec)
-1	0	590	1505.864	1190	1494.695	1790	1494.455	2390	1498.894
0	1541.59	600	1505.527	1200	1494.627	1800	1494.501	2400	1498.99
10	1540.642	610	1505.197	1210	1494.561	1810	1494.549	2410	1499.086
20	1539.709	620	1504.873	1220	1494.498	1820	1494.598	2420	1499.182
30	1538.792	630	1504.556	1230	1494.438	1830	1494.647	2430	1499.279
40	1537.888	640	1504.245	1240	1494.38	1840	1494.699	2440	1499.377
50	1537	650	1503.94	1250	1494.326	1850	1494.751	2450	1499.475
60	1536.126	660	1503.641	1260	1494.273	1860	1494.804	2460	1499.574
70	1535.266	670	1503.349	1270	1494.224	1870	1494.859	2470	1499.673
80	1534.42	680	1503.062	1280	1494.177	1880	1494.914	2480	1499.772
90	1533.587	690	1502.781	1290	1494.132	1890	1494.971	2490	1499.872
100	1532.768	700	1502.506	1300	1494.09	1900	1495.029		
110	1531.963	710	1502.237	1310	1494.051	1910	1495.087		
120	1531.171	720	1501.974	1320	1494.014	1920	1495.147		
130	1530.391	730	1501.716	1330	1493.979	1930	1495.208		
140	1529.625	740	1501.463	1340	1493.946	1940	1495.27		
150	1528.871	750	1501.216	1350	1493.916	1950	1495.333		
160	1528.129	760	1500.974	1360	1493.888	1960	1495.396		
170	1527.4	770	1500.738	1370	1493.863	1970	1495.461		
180	1526.683	780	1500.506	1380	1493.839	1980	1495.527		
190	1525.978	790	1500.28	1390	1493.818	1990	1495.593		
200	1525.284	800	1500.059	1400	1493.799	2000	1495.661		
210	1524.603	810	1499.843	1410	1493.782	2010	1495.729		
220	1523.932	820	1499.631	1420	1493.767	2020	1495.799		
230	1523.273	830	1499.425	1430	1493.755	2030	1495.869		
240	1522.625	840	1499.223	1440	1493.744	2040	1495.94		
250	1521.988	850	1499.026	1450	1493.735	2050	1496.012		
260	1521.362	860	1498.833	1460	1493.729	2060	1496.085		
270	1520.746	870	1498.645	1470	1493.724	2070	1496.159		
280	1520.141	880	1498.462	1480	1493.721	2080	1496.233		
290	1519.546	890	1498.283	1490	1493.72	2090	1496.308		
300	1518.961	900	1498.108	1500	1493.721	2100	1496.385		
310	1518.387	910	1497.938	1510	1493.724	2110	1496.461		
320	1517.822	920	1497.772	1520	1493.728	2120	1496.539		
330	1517.267	930	1497.61	1530	1493.735	2130	1496.617		
340	1516.722	940	1497.452	1540	1493.743	2140	1496.697		
350	1516.187	950	1497.299	1550	1493.753	2150	1496.777		
360	1515.66	960	1497.149	1560	1493.764	2160	1496.857		
370	1515.143	970	1497.003	1570	1493.778	2170	1496.939		
380	1514.635	980	1496.861	1580	1493.793	2180	1497.021		
390	1514.137	990	1496.724	1590	1493.809	2190	1497.104		
400	1513.646	1000	1496.589	1600	1493.828	2200	1497.187		
410	1513.165	1010	1496.459	1610	1493.848	2210	1497.271		
420	1512.693	1020	1496.332	1620	1493.869	2220	1497.356		
430	1512.228	1030	1496.209	1630	1493.892	2230	1497.442		
440	1511.772	1040	1496.09	1640	1493.917	2240	1497.528		
450	1511.325	1050	1495.974	1650	1493.943	2250	1497.615		
460	1510.885	1060	1495.861	1660	1493.97	2260	1497.702		
470	1510.454	1070	1495.752	1670	1493.999	2270	1497.791		
480	1510.03	1080	1495.647	1680	1494.03	2280	1497.879		
490	1509.615	1090	1495.544	1690	1494.061	2290	1497.969		
500	1509.207	1100	1495.445	1700	1494.095	2300	1498.059		

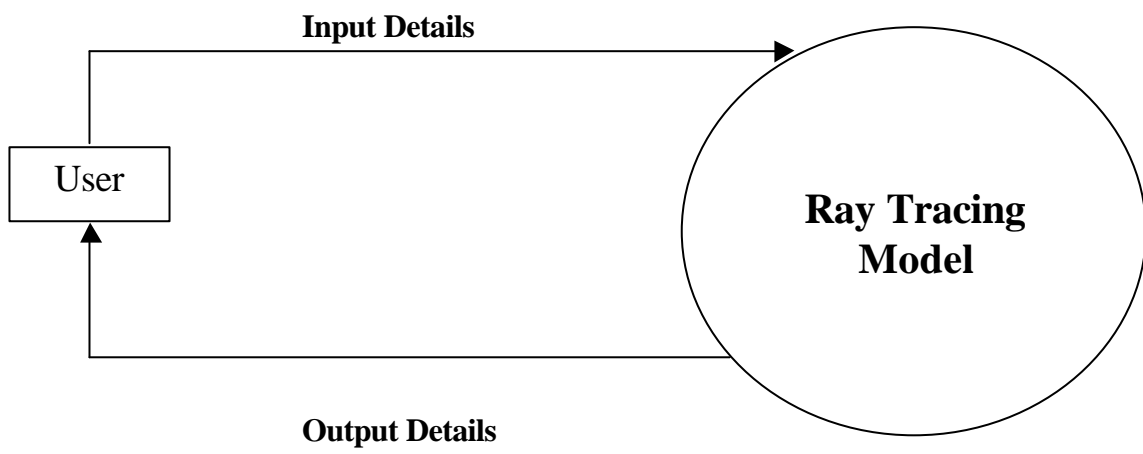
510	1508.806	1110	1495.35	1710	1494.129	2310	1498.149
520	1508.413	1120	1495.257	1720	1494.166	2320	1498.24
530	1508.028	1130	1495.168	1730	1494.203	2330	1498.332
540	1507.65	1140	1495.081	1740	1494.242	2340	1498.424
550	1507.278	1150	1494.998	1750	1494.282	2350	1498.517
560	1506.914	1160	1494.918	1760	1494.323	2360	1498.611
570	1506.557	1170	1494.841	1770	1494.366	2370	1498.705
580	1506.207	1180	1494.766	1780	1494.41	2380	1498.799

Source and Receiver Depth at 1490m.

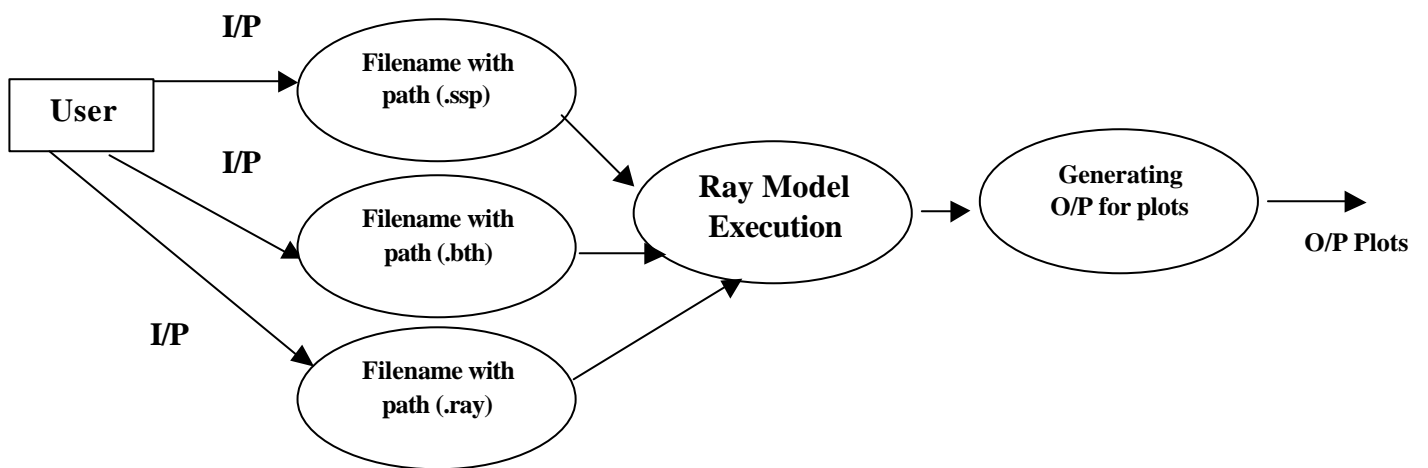
Flow Charts



Context Level DFDs

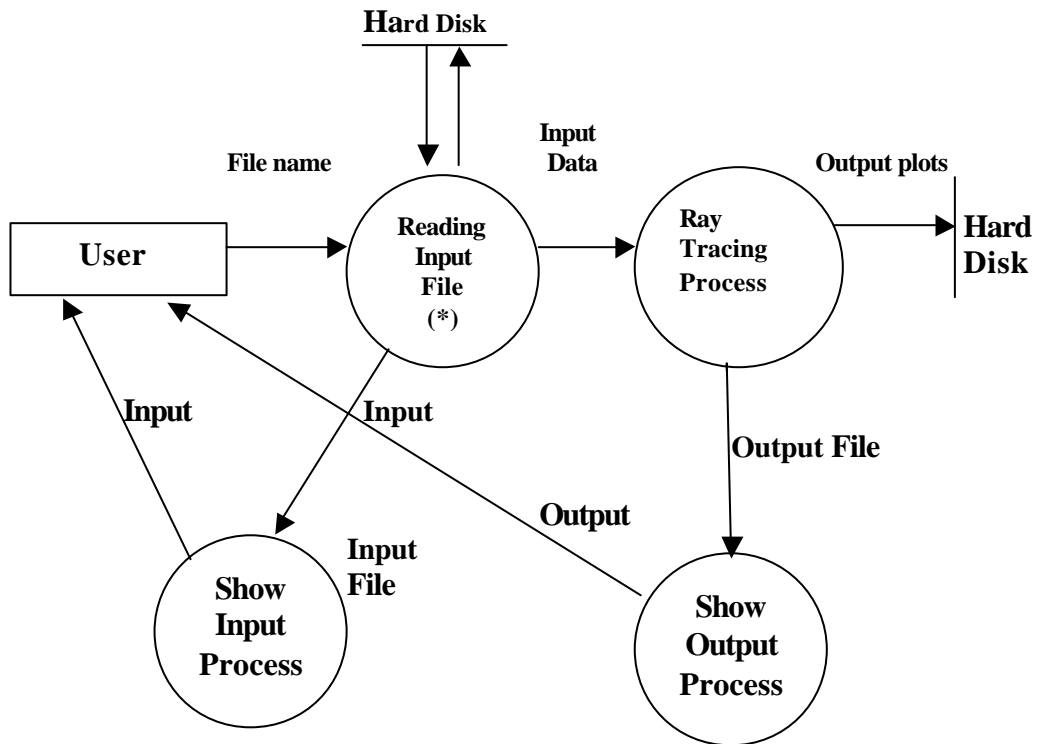


LOWER LEVEL DFD FOR RAY TRACING



Top Level DFDs

Top Level DFD for Ray Tracing



(*) —————> Files of (SSP,BTH AND RAY) have to be given simultaneously.

The following input files are used in the present developed software (Appendix-I) and sequential step wise operations are detailed in the next section.

INPUT FILES

Filename: munk.ssp

Filename: test.bth

Filename: ex1.ray

SOFTWARE STRUCTURE

A Main MATLAB script, Ray, is invoked from the MATLAB Command Window. This brings up the Main Graphical User Interface (GUI), which is populated by 'PUSH BUTTONS' i.e., Ray File, SSP File, BTH File, RUN MODEL, PLOTS, REFRESH, Print Plots, SAVE PLOT and EXIT is shown in FIG 1.

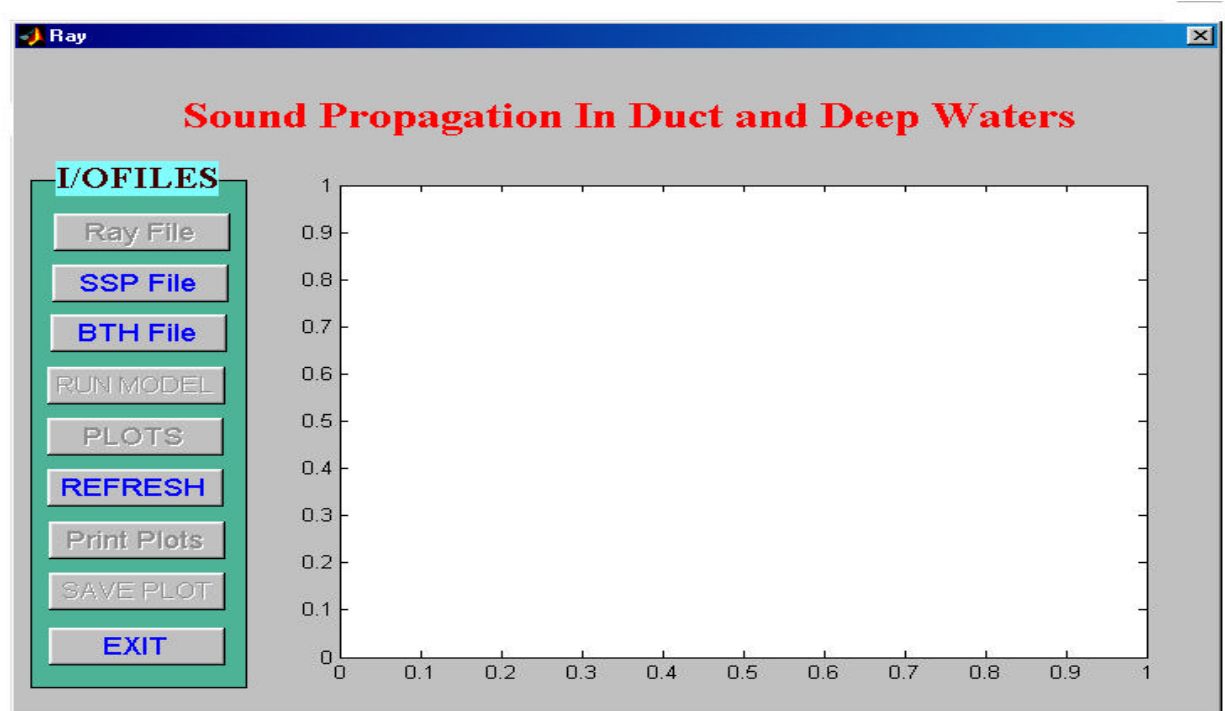


FIG 1.

Step wise Process

On clicking on the PUSH BUTTON named “SSP File” allows the user to select the input for the list box as shown in FIG 2.

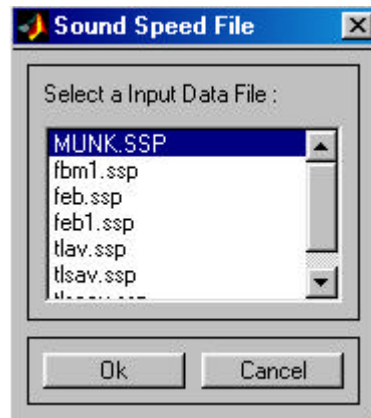


FIG 2.

Next selecting the Bathymetry profile by clicking on the PUSH BUTTON named “BTH File” as shown in FIG 3. This enables the PUSH BUTTON 's named Ray File and RUN MODEL.

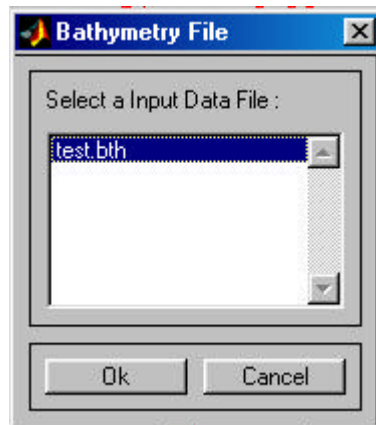


FIG 3.

Clicking on the PUSH BUTTON “Ray File” allows the list box as shown in FIG 4. on selecting the file from the list pops up the Ray file which consists of data regarding source, receiver, depth etc.,

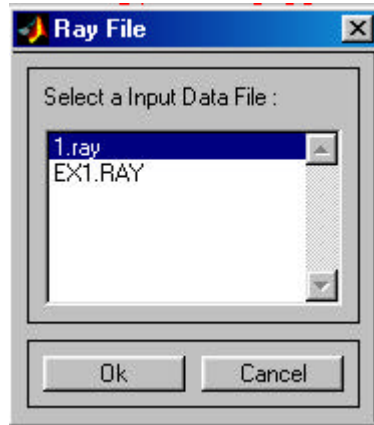


FIG 4.

Now the system is ready for execution as the input files i.e., Sound Speed Profiles (.ssp), Bathymetry File (.bth) and Ray File (.ray) are given. Click on the PUSH BOTTON named RUN MODEL, after the execution of program the PUSH BOTTON 's namely PLOTS, SAVE PLOT and Print Plots are enabled.

On clicking on the plots displays the list box which is shown in FIG 5. which lists the four varieties of plots that has to be displayed namely Eigenrays, Bathmetry, Ztheta and Stick.



FIG 5.

Selecting Eigenrays in the list box (FIG 5.) displays the plot in the form as shown in FIG 6.

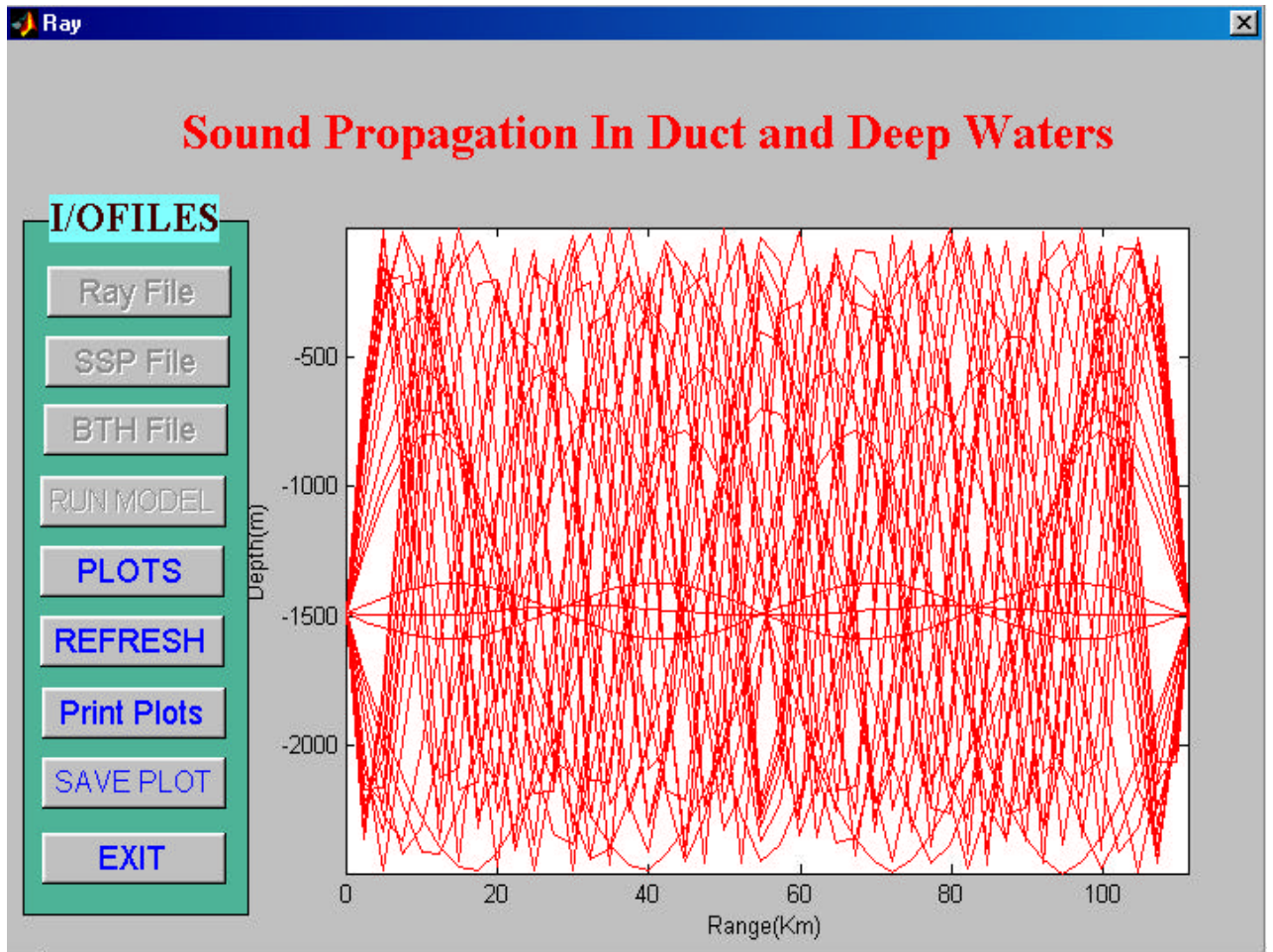


FIG 6.

Selecting Bathymetry in the list box (FIG 5.) displays the plot in the form as shown in FIG 7.

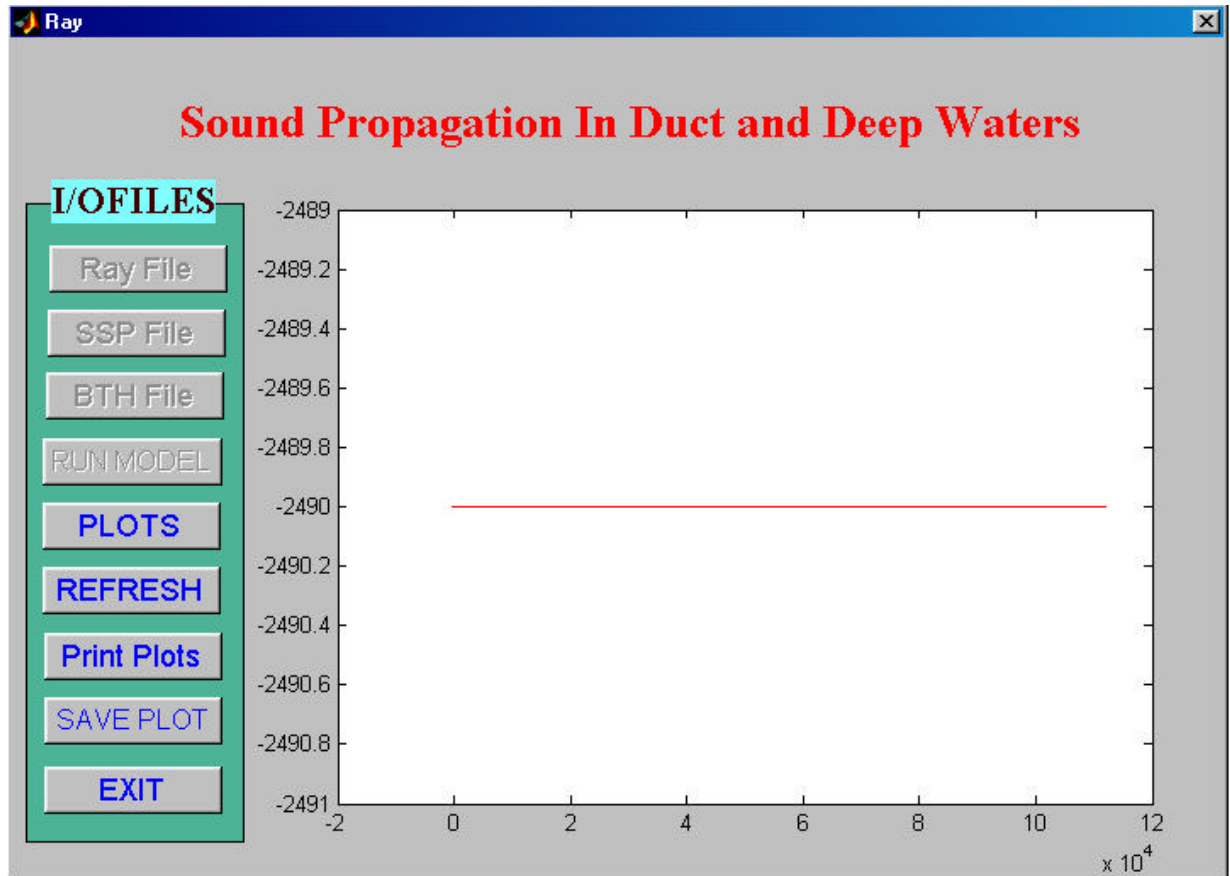


FIG 7.

Selecting Ztheta in the list box (FIG 5.) displays the plot in the form as shown in FIG 8.

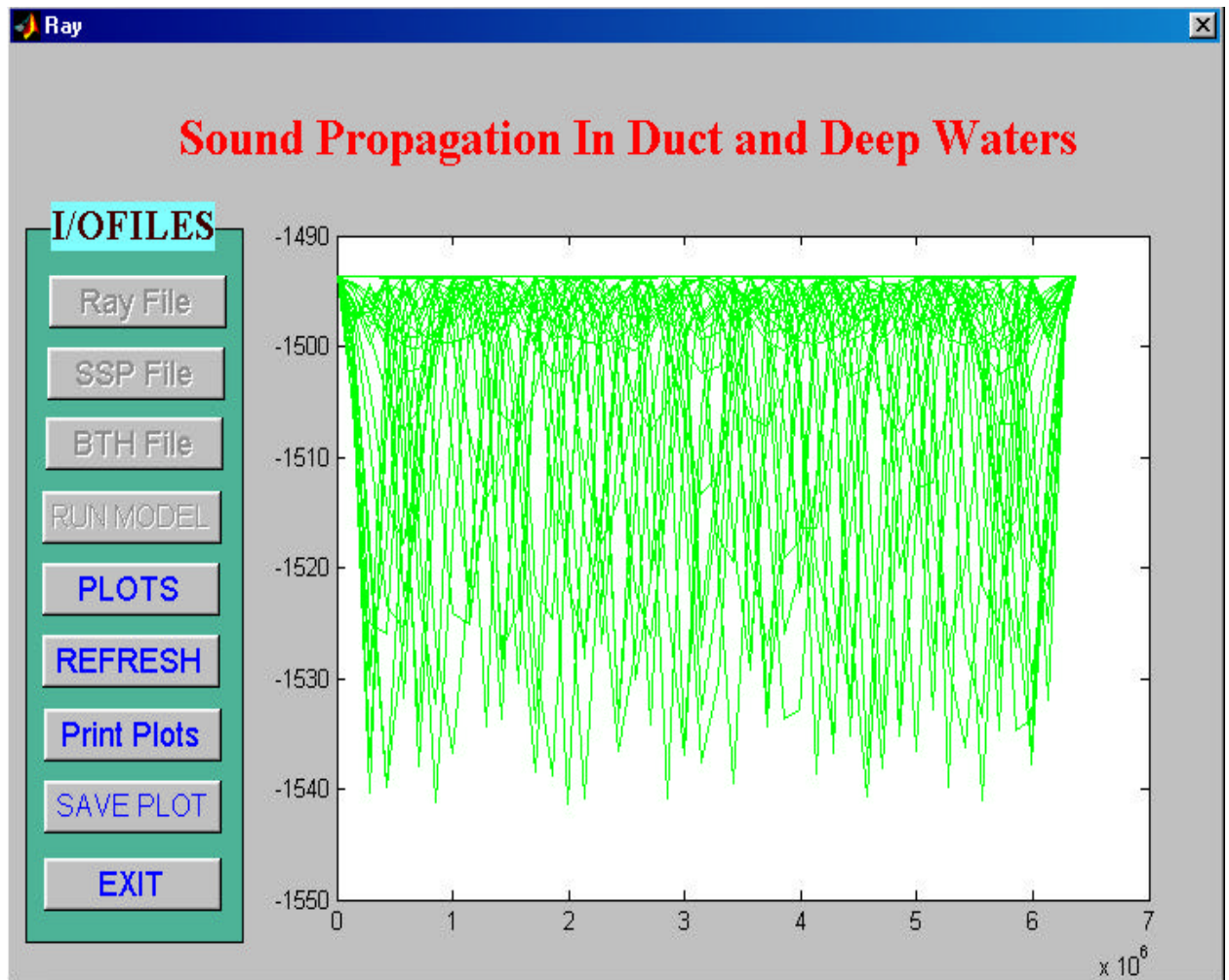


FIG 8.

Selecting Stick in the list box (FIG 5.) displays the plot in the form as shown in FIG 9.

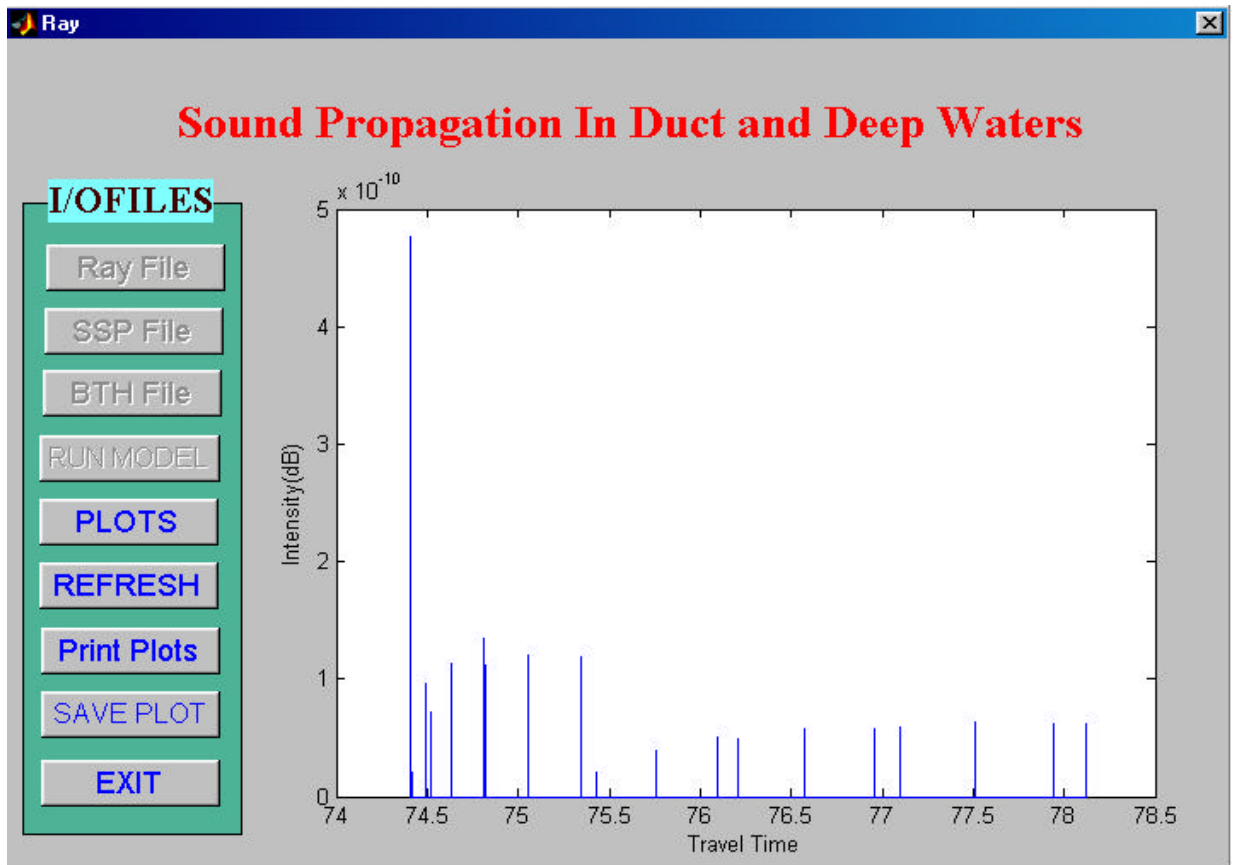


FIG 9.

Above plots can be saved by clicking on the PUSH BUTTON named SAVE PLOT, which pops a input box as shown in FIG 10 asking for the name by which the plot has to be stored.



FIG 10.

CASE STUDY

This interface is applied for Ocean Data. MUNK 's canonical Sound Speed Profiles are taken into consideration. The output is as shown in FIG 11-13.

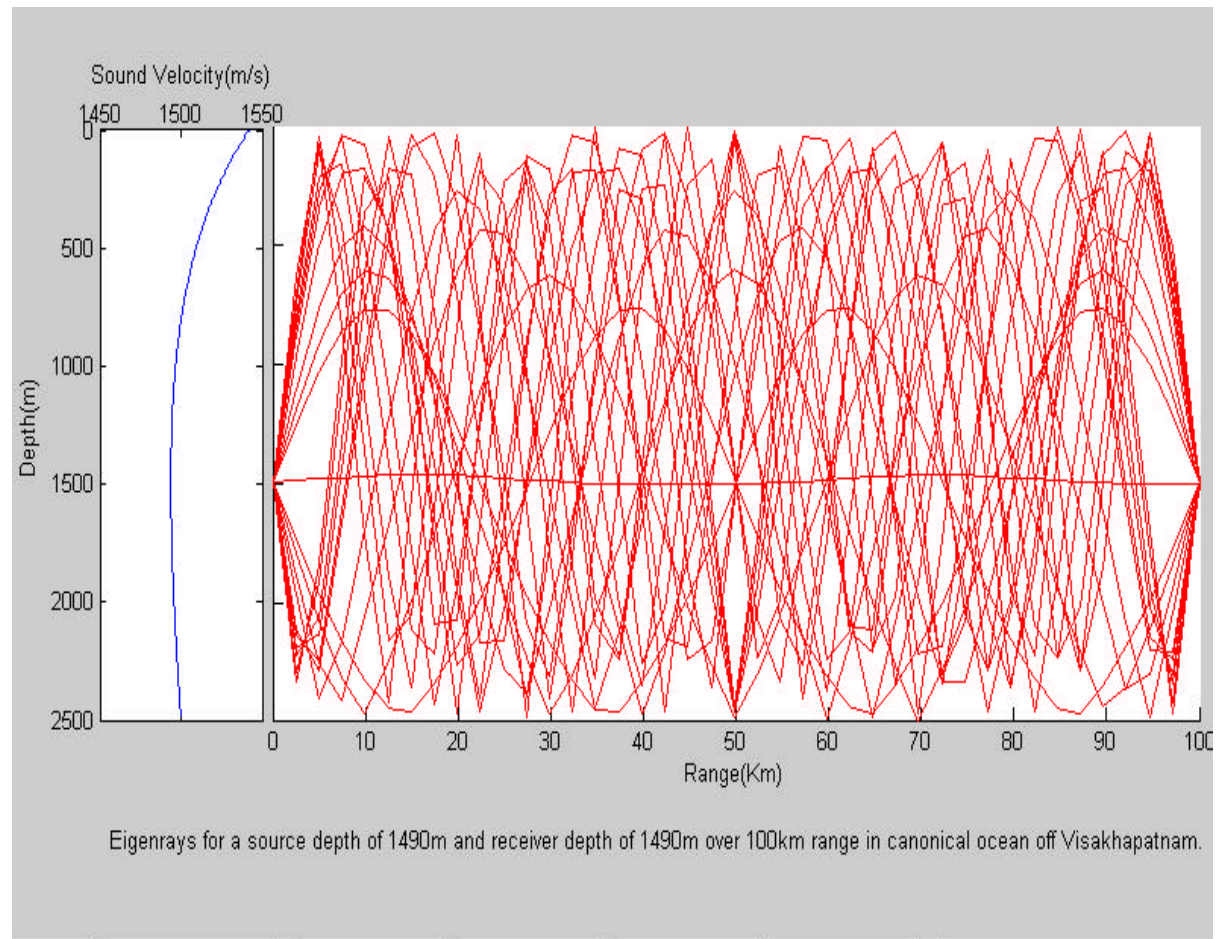


FIG 11.

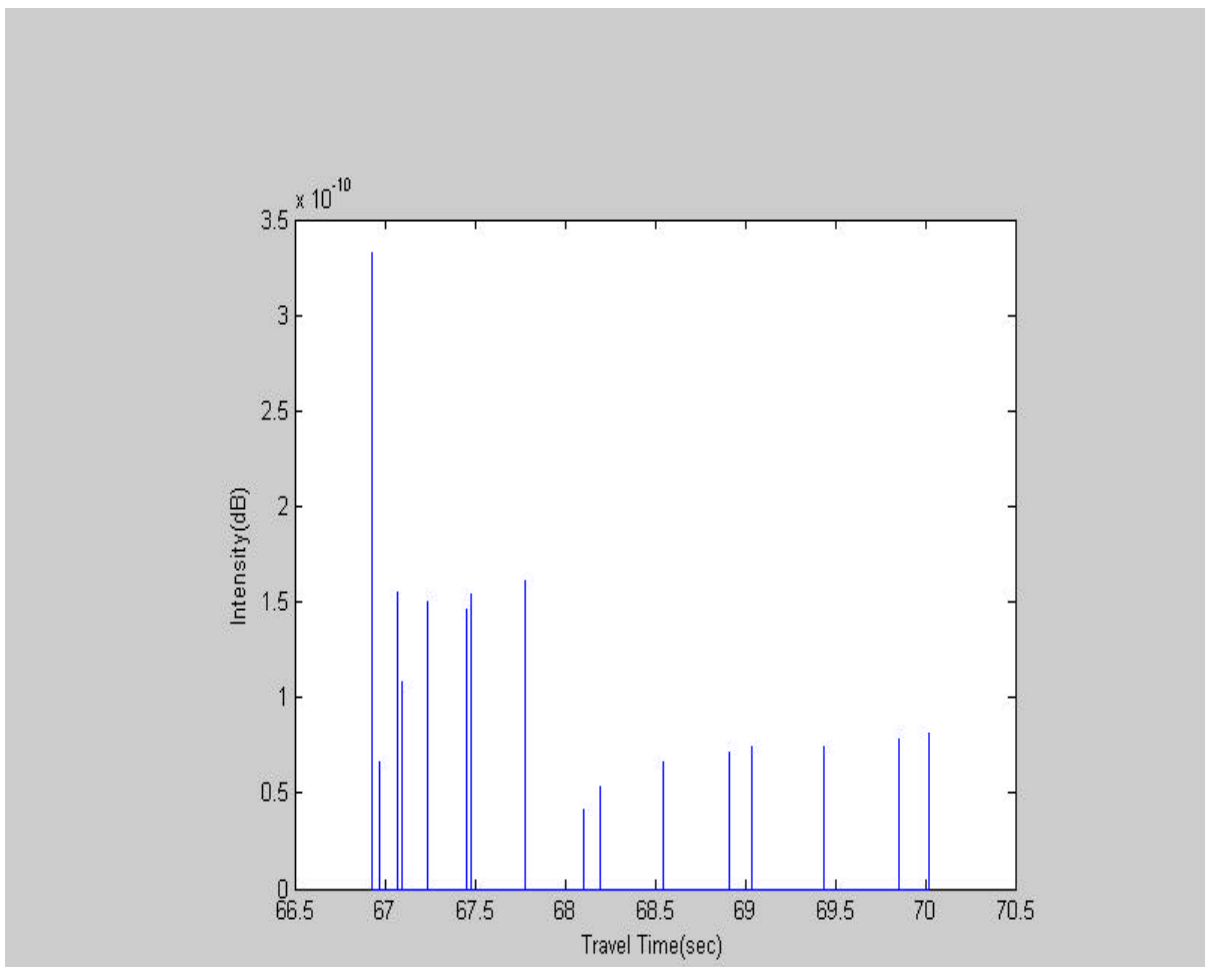


FIG 12.

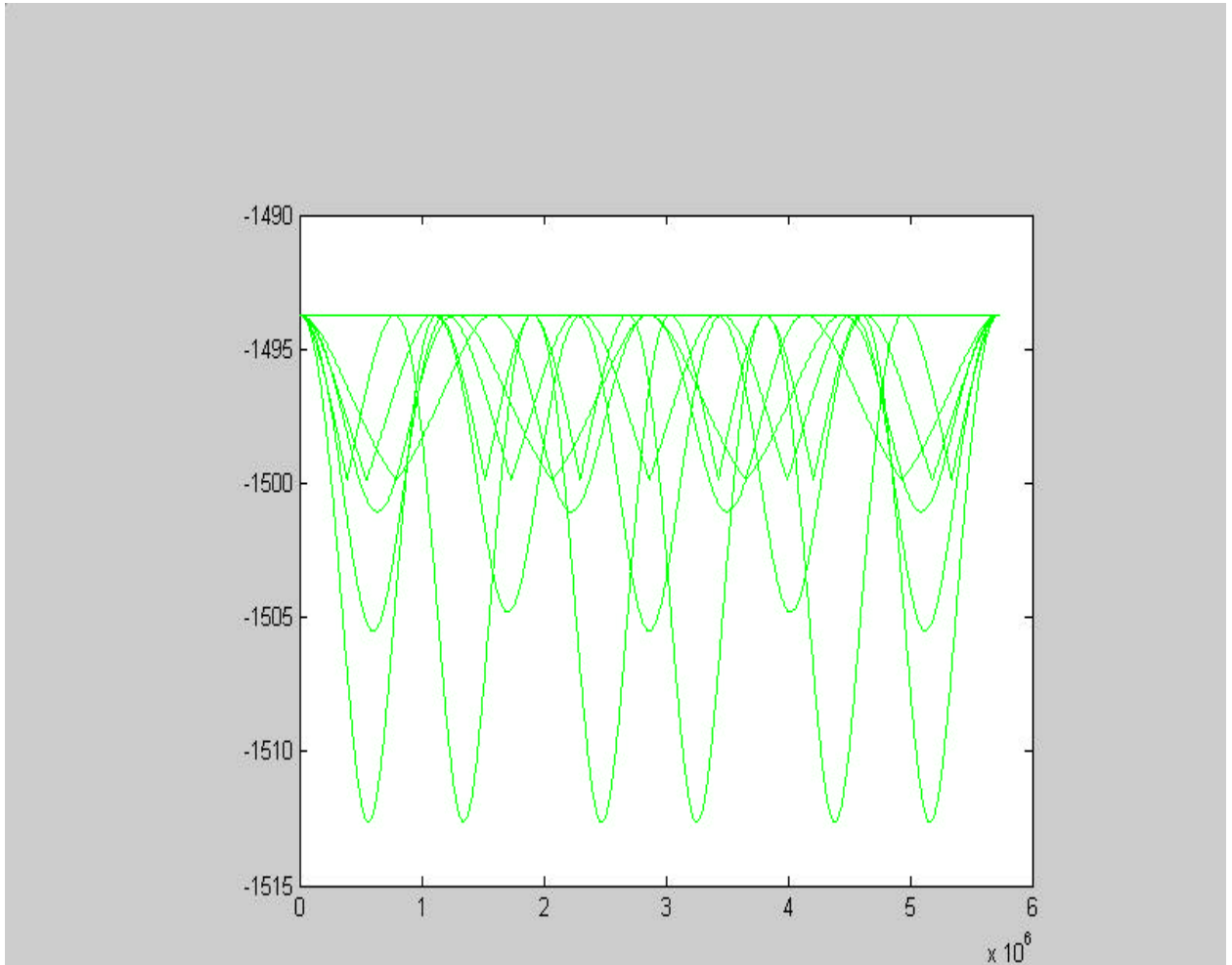


FIG 13.

Results

Following Munk's canonical theory, canonical parameters (i.e., B the stratification scale and β the perturbation coefficient) in adiabatic ocean obtained using deduced SOFAR channel parameters (i.e., C_1 sound speed at the channel axis, Z_1 the depth of the channel axis) from the CTD data collected on board ORV Sagarkanya Cruise No.30 at 17° 36.21'N, 84° 35.31'E, in the Bay of Bengal to develop the stratified model for implementation in the numerical ray tracing to study the ocean acoustic parameters namely eigen rays, travel time, turning depths, transmission loss etc., for source-receiver configuration kept at channel axis.

The simulated acoustic ray parameters of canonical ocean reveals that the even very flat angle rays can scan entire water column with repeated reflections from both surface and bottom due to depth limited nature of sound channel and weak gradients of sound speed prevailing in the medium on either side of the sound channel axis. This nature of phenomenon usually occurs in the case of shallow-water channel. Since the results are generated through simplest sound channel (exponential stratified model) above and beneath the sound axis would be more useful as aprior knowledge before conducting actual sound transmission experiment in the area of interest.

Acknowledgements

The authors are thankful to Dr.S.R.Shetye, Director, National Institute of oceanography, for his constant encouragement and keen interest in this study.

References

1. Urick,R.J, (1983) Principles of Underwater Sound, McGraw-Hill, Inc.
2. Officer C.B, (1958) Introduction To The Theory Of Sound Transmission, McGraw-Hill, Inc. USA.
3. Ramana Murty T.V, Mahadevan.R (March 1994), Stochastic Inverse Method for Ocean Acoustic Tomography Studies – A Simulation Experiment, IIT Technical Report No.31.
4. Krol H R, J Acoust Soc Am, 53(1973)864.
5. Moler C B & Solomom L P, J Acoust Soc Am,48(1970)739.
6. Munk W H(1974) Sound Channel in an exponentially stratified ocean with application to SOFAR J.Acoust.SOC.AM.55 220-226.
7. Ramana Murty T V, Somayajulu Y K & Sastry J S (1990) Computations of some acoustic ray parameters in the Bay of Bengal, IJMS,Vol. 19, pp 235-245.
8. Ramana Murty T V, Prasanna Kumar S, Somayajulu Y K, Sastry J S and Rui J P De Figueiredo. (October 1989) Canonical sound speed profile for the central Bay of Bengal. Earth Planet Sci., Vol 98, No: 3, July, pp 255-263.
9. Reiniger R F and Ros C K. (1968) A method of interpolation with application to oceanographic data; Deep Sea Res.55 185-193.
10. Chen C T and Millero F J. (1977) Speed of sound in sea water at higher pressure; J.Acoust.Soc,Am.55, 129-135.

Appendix-I

This invokes the GUI interface for "Sound Propagation in Duct and Deep waters" Created on September 10th 2004 --- NIO, RC, Visakhapatnam

```
function varargout = ray(varargin)
%   Ref: WHOI Technical Report – WHOI-93-10-“Ocean Acoustic Ray-Tracing
%   Software RAY”, Bowlin,et.al., October, 1992
%   ray M-file for ray.fig
%   ray, by itself, creates a new ray or raises the existing
%   singleton*.
%
%   H = ray returns the handle to a new ray or the handle to
%   the existing singleton*.
%
%   ray('Property','Value',...) creates a new ray using the
%   given property value pairs. Unrecognized properties are passed via
%   varargin to ray_OpeningFcn. This calling syntax produces a
%   warning when there is an existing singleton*.
%
%   ray('CALLBACK') and ray('CALLBACK',hObject,...) call the
%   local function named CALLBACK in ray.M with the given input
%   arguments.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% Edit the above text to modify the response to help ray

% Last Modified by GUIDE v2.5 01-Oct-2004 16:32:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @ray_OpeningFcn, ...
                  'gui_OutputFcn', @ray_OutputFcn, ...
                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

% End initialization code - DO NOT EDIT

% --- Executes just before ray is made visible.
function ray_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%           command line (see VARARGIN)

% Choose default command line output for ray
handles.output = hObject;
handles.root  = "";

handles.Refresh= "";
set( [handles.Ssp] , 'enable', 'on' );
set( [handles.Bth] , 'enable', 'on' );
set( [handles.Print] , 'enable', 'off' );
set( [handles.Ray] , 'enable', 'off' );
set( [handles.Plots] , 'enable', 'off' );
set( [handles.runModel] , 'enable', 'off' );
set( [handles.EXIT] , 'enable', 'on' );
set( [handles.Refresh] , 'enable', 'on' );
set( [handles.Save] , 'enable', 'off' );

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ray wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = ray_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in Ray.
function Ray_Callback(hObject, eventdata, handles)
% hObject    handle to Ray (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

global FNAME
d = dir('*.ray');

if isempty( d )
    errorDlg( 'Ray Files Not Found', 'Error' );
else
    filenames = {d.name};
    [ s, v ] = listdlg( 'PromptString', 'Select a Input Data File :',...
        'SelectionMode', 'single',...
        'ListSize',[150,90],...
        'Name','Ray File',...
        'ListString', filenames );
    handles.viewSourceFile = char( filenames( s ) );
    handles.root = handles.viewSourceFile( 1:length( handles.viewSourceFile ) - 4
);
    FNAME=handles.viewSourceFile
    if ( isempty( handles.viewSourceFile ) )
        warnDlg( 'No Input Ray File Has Been Selected', 'Warning' );
    else
        h = questdlg('Save files with(.ray) Extension','Warning','OK','OK')
        if (h=='OK')
            eval( [ '!write ' handles.viewSourceFile ] );
        end
    end

    set( hObject, 'Value', 1.0 ); % show pushed
end
set( [handles.runModel] , 'enable', 'on' );
% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in Ssp.
function Ssp_Callback(hObject, eventdata, handles)
% hObject    handle to Ssp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

d = dir('*.ssp');

if isempty( d )
    errorDlg( 'Sound Speed Files Not Found', 'Error' );
else
    filenames = {d.name};
    [ s, v ] = listdlg( 'PromptString', 'Select a Input Data File :',...
        'SelectionMode', 'single',...
        'ListSize',[150,90],...
        'Name','Sound Speed File',...
        'ListString', filenames );
    handles.viewSourceFile = char( filenames( s ) );
    handles.root = handles.viewSourceFile( 1:length( handles.viewSourceFile ) - 4 );

```

```

if ( isempty( handles.viewSourceFile ) )
    warndlg( 'No Input Sound Speed File Has Been Selected', 'Warning' );
else
    h = questdlg('Save files with(.ssp) Extension','Warning','OK','OK')
    if (h=='OK')
        eval( [ '!write ' handles.viewSourceFile ] );
    end
end

set( hObject, 'Value', 1.0 ); % show pushed
end

% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in Bth.
function Bth_Callback(hObject, eventdata, handles)
% hObject    handle to Bth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

d = dir('*.bth');

if isempty( d )
    errordlg( 'Bathymetry Files Not Found', 'Error' );
else
    filenames = {d.name};
    [ s, v ] = listdlg( 'PromptString', 'Select a Input Data File :',...
        'SelectionMode', 'single',...
        'ListSize',[150,90],...
        'Name','Bathymetry File',...
        'ListString', filenames );
    handles.viewSourceFile = char( filenames( s ) );
    handles.root = handles.viewSourceFile( 1:length( handles.viewSourceFile ) - 4
);

if ( isempty( handles.viewSourceFile ) )
    warndlg( 'No Input Bathymetry File Has Been Selected', 'Warning' );
else
    h = questdlg('Save files with(.bth) Extension','Warning','OK','OK')
    if (h=='OK')
        eval( [ '!write ' handles.viewSourceFile ] );
    end
end
set( hObject, 'Value', 1.0 ); % show pushed
end

set( [handles.Ray], 'enable', 'on' );

```

```

set( [handles.Ssp] , 'enable', 'off' );
set( [handles.Bth] , 'enable', 'off' );

% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in Print.
function Print_Callback(hObject, eventdata, handles)
% hObject    handle to Print (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
print -depsc2 -tiff ms.eps
print -dtiff -r200 ms.tiff

% --- Executes on button press in EXIT.
function EXIT_Callback(hObject, eventdata, handles)
% hObject    handle to EXIT (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

closereq;

% --- Executes on button press in runModel.
function runModel_Callback(hObject, eventdata, handles)
% hObject    handle to runModel (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global FNAME
dis=dos(['ray ' FNAME])    % Executes Ray.exe with the name as selected in the
                        % list box when Push Botton "Ray File" is clicked.

if (dis==0)
eval(['!ray <' FNAME ' > out.txt'])
%eval(['!ray <' FNAME ' > out.txt | exit &'])
eval('!exit&')
msgbox('Model Executed Successfully','Success','help','modal');
set( [handles.Ray], 'enable', 'off' );
set( [handles.Plots] , 'enable', 'on' );
set( [handles.runModel] , 'enable', 'off' );
else
msgbox('Check input files','Error','Error','modal');
set( [handles.runModel] , 'enable', 'off' );
end

% --- Executes on button press in Plots.
function Plots_Callback(hObject, eventdata, handles)
% hObject    handle to Plots (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
eval('load ex1.mat')
str={'Eigenrays','Bathmetry','Ztheta','Stick'};
[ s, v ] = listdlg('PromptString','Select a file:',...

```

```

        'SelectionMode','single',...
        'ListSize',[90,90],...
        'Name','Select a Plot',...
        'ListString',str)
if(v==0)
    msgbox('No Plot is selected','Warning','warn','modal');
end
switch(s)
    case{1}
        eval('plotpath(eigenrays)');           % Plots Eigenrays
        set( [handles.Save] , 'enable', 'on' );
        set( [handles.Print] , 'enable', 'on' );
    case{2}
        eval('plotbath(btab)');               % Plots Bathymetry
        set( [handles.Save] , 'enable', 'on' );
        set( [handles.Print] , 'enable', 'on' );
    case{3}
        eval('plotzthe(eigenrays)');          % Plots Ztheta
        set( [handles.Save] , 'enable', 'on' );
        set( [handles.Print] , 'enable', 'on' );
    case{4}
        eval('stickem');                       % Plots Stick
        set( [handles.Save] , 'enable', 'on' );
        set( [handles.Print] , 'enable', 'on' );
end
%eval('clear all')
% --- Executes on button press in Refresh.
function Refresh_Callback(hObject, eventdata, handles)
% hObject    handle to Refresh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
closereq
ray

% --- Executes when figure1 window is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton26.
function pushbutton26_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% --- Executes on button press in Save.
function Save_Callback(hObject, eventdata, handles)
% hObject    handle to Save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
answer = inputdlg('Give Name:','Saving The Plot',1);
answer=char(answer);
R={''};
if (isempty(answer) | isequal(answer,R))
    warndlg( 'File Name is not given', 'Warning' );
else
    %cd pics
    hgsave(gca,answer);
end
%cd ..
eval('clear all')

```

The different plotting files i.e., (.m) files are given below:

- 1) plotpath.m ✎ Used for plotting Eigenrays
- 2) plotzthe.m ✎ Used for plotting Ztheta
- 3) stickem.m ✎ Used for plotting Stick
- 4) plotbath.m ✎ Used for plotting Bathymetry

The coding of the these files are given below.

PLOTPATH.M

```
function plotpath(paths,range,color)
%
% function plotpath(paths,range,color)
%
% plots all paths in paths in color color. Assumes that
% the range is the first column of paths
% the depth is the second column of paths
% the ray number is the last column of paths
% The color defaults to green if not specified.

% THis is for plotting rays for NUKE'S ONLY
%   change ranges to km from m

if (nargin==1),
    range=[-90 90];
    color='r';
elseif (nargin==2),
    if (isstr(range)), color=range; range=[-90 90];
    else color='r';
    end;
end;

range=range*pi/180;

%%% r = paths(:,1);
r = paths(:,1)./1000.;
z = -paths(:,2);
[n,m] = size(paths);
idex = find(r(1:(n-1)) > r(2:n) );
sdex = [1 ; idex+1];
idex = [idex; size(r,1)];
lastii = 1;
axis([min(r),max(r),min(z),max(z)]);
xlabel('Range(Km)')
ylabel('Depth(m)')
for i = 1:length(idex)

    v = sdex(i):idex(i);
    if ( paths(sdex(i),4)>=range(1) & paths(sdex(i),4)<=range(2) ),
        %%%% plot(r(v),z(v),color); hold on;
        line(r(v),z(v),'color',color);
        %%% pause;
    end;
end;
%%% hold off;
```

STICKEM.M

```
-----  
%          stickem.m  
%  
%  
% -----  
[x,y] = stick(eigenfront(:,7),eigenfront(:,23));  
plot(x,y);  
xlabel('Travel Time')  
ylabel('Intensity(dB)')
```

PLOTZTHE.M

```
function plotztheta(paths,color)  
%  
% function plotztheta(paths,color)  
%  
% plots all paths in paths in color color. Assumes that  
% the range is the first column of paths  
% the depth is the second column of paths  
% the ray number is the last column of paths  
% The color defaults to green if not specified.  
  
if (nargin < 2); color = 'g'; end;  
  
s = paths(:,1);  
z = -paths(:,5);  
t = paths(:,3);  
a = paths(:,4);  
[n,m] = size(paths);  
  
plot(s*180/pi,z,color);
```

PLOTBATH.M

```
function plotbath(btab,color)  
%  
% function plotbath(btab,color)  
%  
% plots a rough version of the bathymetry in btab by  
% just drawing a straight line for each bathymetry section.  
% The color defaults to red if not specified.  
if (nargin < 2); color = 'r'; end;  
%plottedit('hideFilemenu');  
plot(btab(:,1),-btab(:,2),color);
```