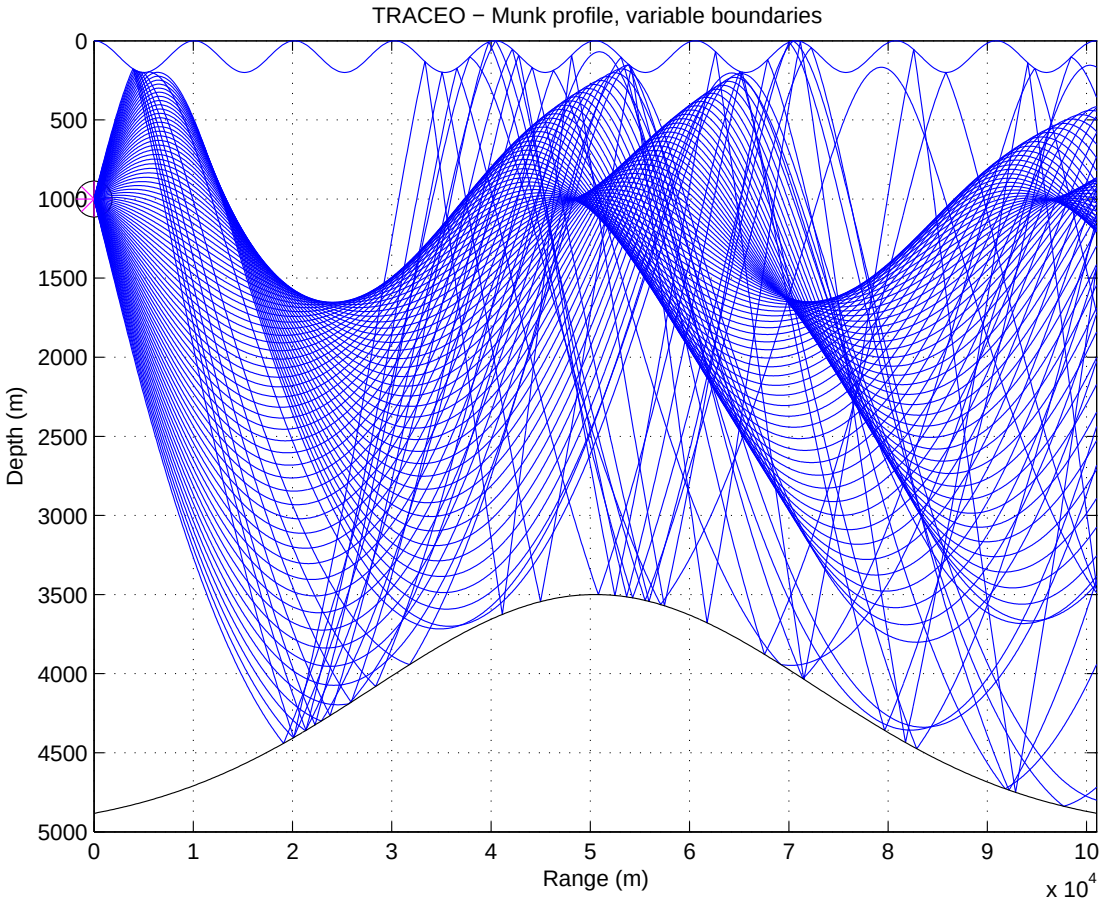
The TRACEO
ray tracing program



Orlando Camargo Rodríguez
(http://w3.ualg.pt/˜orodrig, orodrig@ualg.pt)
Physics Department
Signal Processing Laboratory
Faculdade de Ciências e Tecnologia
Universidade do Algarve
10/01/2011

Contents

Chapter 1

Introduction

This document describes the first (distribution-ready) release of **TRACEO**, a ray tracing model written in Fortran 77 but tested with the GNU gfortran compiler, and available under a Creative Commons license. The current version of the model replaces the models previously known as **TRACE** (a standard ray tracing model) and **TRACEO** (an adaptation of **TRACE**, which allowed to consider the presence of a single object, located between the acoustic source and the array of receivers). Not only the former **TRACE** and **TRACEO** were merged into what is now **TRACEO**, as the original code was carefully rewritten, in order to allow the optional inclusion of one or more objects, to allow upper and lower boundaries with range-dependent properties (including compressional and shear velocities and attenuations), to allow eigenray calculations at the positions specified by array coordinates, and to output the results (rays, arrivals, amplitudes, acoustic pressure and particle velocity components) as Matlab MAT files[1]. **TRACEO** can handle a particular set of analytical sound speed profiles, or general tabulated sound speed profiles or fields. The receiver array can be horizontal, vertical, rectangular, or it can have an arbitrary curvilinear shape; the hydrophones are not required to be equidistant. Rays can be partially or totally reflected on any boundary of the waveguide, or be completely absorbed. **TRACEO** was developed in order to model acoustic propagation in environments, which available models were not able to handle (like wavy surfaces, complex bathymetries, depth and range variations of sound speed, etc.), and for applications in the areas of geoacoustics, vector sensor arrays, underwater communications and acoustic barriers.

TRACEO strongly benefited from the availability of the Bellhop ray tracing model[1], one of the components of the Acoustic Toolbox[2]; Bellhop

[1] A standalone version of **TRACEO** not linked to the Matlab's engine, which writes huge ascii files, is also available by request.

is developed (and constantly updated) by Michael Porter from HLS research. **TRACEO** borrows many methods from Bellhop, but goes beyond Bellhop's capabilities by allowing calculations in the following cases:

- using a set of analytical profiles;
- positioning targets between the source and the array of receivers;
- considering boundaries with range-dependent properties (which also account for shear velocity and attenuation);
- considering boundaries, which besides being partially or totally reflective can be completely absorbent.

The present document was not written with the intention of being just **TRACEO**'s manual. Ray tracing and Gaussian beams are so compelling subjects, that the presentation of **TRACEO** was not considered to be complete without a detailed discussion of Ray tracing and Gaussian beams in independent chapters. Researchers with more interest in applications can skip directly to the chapters describing the model (installation, the general structure of the input file, etc.), and to the discussion of relevant model numerical issues, particular examples and comparisons with other models. The final chapter presents some conclusions and future directions of potential development. An appendix contains additional topics covering ray tracing modeling in more complex cases (accounting for ocean currents and earth's curvature) and within the context of the Hamiltonian formalism.

Before proceeding a final word of acknowledgement should be addressed to those, which one way or another made possible the development of **TRACEO**. First of all is Michael Porter, Bellhop's author, whose constantly updated code was a valuable guide to understand and test most the methods implemented in **TRACEO**. Michael Porter is also a co-author of *COMPUTATIONAL OCEAN ACOUSTICS* [3], a central reference for underwater acoustic modeling. Second in the list goes Mikhail Mikhailovich Popov, whose excellent *RAY THEORY AND GAUSSIAN BEAM METHOD FOR GEOPHYSICISTS* (available in the internet) provides an excellent discussion of Gaussian beam theory. Finally, a word of gratitude should be addressed to Vlastislav Cerveny, Ludek Klimes and Petr Bulant, members of the consortium research project *Seismic waves in complex 3-D structures* (SW3D), which maintain an online library of valuable papers, related to Gaussian beams; the consortium provides also an extensive set of ray tracing codes, oriented to seismic applications, which share a lot of methodologies used in acoustic underwater ray models. To all of them I would like to present my deepest gratitude.

Chapter 2

Ray tracing theory

2.1 The acoustic wave equation

The starting point for the discussion of the ray tracing is given by the acoustic wave equation, which in the case of a watercolumn with a constant density can be written as

$$\boldsymbol{\nabla}^2 p - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = s(\mathbf{r}_0, t) \ , \tag{2.1}$$

where $p(\mathbf{r}, t)$ stands for the pressure of the acoustic wave, $s(\mathbf{r}_0, t)$ represents the signal transmitted by the acoustic source, $\mathbf{r}_0$ represents the source position and $\boldsymbol{\nabla}$ represents the nabla differential operator. Applying a Fourier transform to both sides of Eq.(2.1) one can obtain the so called Helmholtz equation:

$$\left[\boldsymbol{\nabla}^2 + \frac{\omega^2}{c^2}\right] P(\mathbf{r}, \omega) = S(\mathbf{r}_0, \omega) \ , \tag{2.2}$$

where

$$P(\mathbf{r}, \omega) = \int_{-\infty}^{\infty} p(\mathbf{r}, t) e^{-i\omega t} dt \ ,$$

and

$$S(\mathbf{r}_0, \omega) = \int_{-\infty}^{\infty} s(\mathbf{r}_0, t) e^{-i\omega t} dt \ .$$

Let's consider a plane wave-like approximation to the solution of Eq.(2.2) and write that [4]

$$P(\omega) = A e^{-i\omega\tau} \ , \tag{2.3}$$

where A represents a slowly changing wave amplitude, and $\omega\tau$ stands for a rapidly evolving phase; the surfaces with constant $\omega\tau$ represent the *wavefronts*; analogously, the surfaces with constant τ are called *timefronts*. By

placing Eq.(2.3) into the homogeneous form of Eq.(2.2), considering the high frequency approximation

$$\frac{\boldsymbol{\nabla}^2 A}{A} \ll k^2$$

(where $k = \omega/c$) and separating the real and imaginary terms of the equation, one can obtain the *Eikonal equation*:

$$(\boldsymbol{\nabla}\tau)^2 = \frac{1}{c^2} \tag{2.4}$$

and the *transport equation*:

$$2\,(\boldsymbol{\nabla}A \cdot \boldsymbol{\nabla}\tau) + A\boldsymbol{\nabla}^2\tau = 0\ . \tag{2.5}$$

The following sections will describe the solution of Eqs.(2.4)–(2.5).

2.2 Solving the Eikonal equation

The Eq.(2.4) can be rewritten as

$$|\boldsymbol{\nabla}\tau| = \frac{1}{c}\ , \tag{2.6}$$

which can be further simplified as

$$\frac{d\tau}{ds} = \frac{1}{c}\ , \tag{2.7}$$

where ds stands for the distance traveled by the acoustic wave. Therefore, it follows that

$$d\tau = \frac{ds}{c}\ , \tag{2.8}$$

stands for the travel time along ds. For a wave propagating between two points A and B the total travel time corresponds then to

$$\tau = \int_A^B \frac{ds}{c}\ . \tag{2.9}$$

2.2.1 Lagrangian's formalism and Fermat's principle

Based on Lagrangian's formalism one can write Eq.(2.9) as

$$\tau = \int_A^B \mathcal{L}\ ds\ , \tag{2.10}$$

where $\mathcal{L}$ represents the system's Lagrangian:

$$\mathcal{L} = \frac{1}{c} \ . \tag{2.11}$$

Within the context of the formalism $\mathcal{L}$ is supposed to be a function of coordinates and generalized velocities[5]:

$$\mathcal{L}(x, y, z, \dot{x}, \dot{y}, \dot{z}) = \frac{1}{c}\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \ , \tag{2.12}$$

where

$$\dot{x} = \frac{dx}{ds} \ , \quad \dot{y} = \frac{dy}{ds} \ , \quad \dot{z} = \frac{dz}{ds} \ .$$

In this way, the perturbation of the travel time corresponds to

$$\delta\tau = \int_A^B \left\{ \left[\frac{\partial \mathcal{L}}{\partial x}\delta x + \frac{\partial \mathcal{L}}{\partial \dot{x}}\delta\dot{x} \right] + \left[\frac{\partial \mathcal{L}}{\partial y}\delta y + \frac{\partial \mathcal{L}}{\partial \dot{y}}\delta\dot{y} \right] + \left[\frac{\partial \mathcal{L}}{\partial z}\delta z + \frac{\partial \mathcal{L}}{\partial \dot{z}}\delta\dot{z} \right] \right\} ds =$$

$$= \underbrace{\frac{\partial \mathcal{L}}{\partial \dot{x}}\delta x + \frac{\partial \mathcal{L}}{\partial \dot{y}}\delta y + \frac{\partial \mathcal{L}}{\partial \dot{z}}\delta z \Big|_A^B}_{=0} +$$

$$+ \int_A^B \left\{ \left[\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{ds}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}} \right) \right] \delta x + \left[\frac{\partial \mathcal{L}}{\partial y} - \frac{d}{ds}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}} \right) \right] \delta y + \left[\frac{\partial \mathcal{L}}{\partial z} - \frac{d}{ds}\left(\frac{\partial \mathcal{L}}{\partial \dot{z}} \right) \right] \delta z \right\} ds \ .$$

According to Fermat's principle

$$\delta\tau = 0 \ ,$$

which implies that

$$\begin{aligned} \frac{d}{ds}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}} \right) - \frac{\partial \mathcal{L}}{\partial x} &= 0 \ , \\ \frac{d}{ds}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}} \right) - \frac{\partial \mathcal{L}}{\partial y} &= 0 \ , \\ \frac{d}{ds}\left(\frac{\partial \mathcal{L}}{\partial \dot{z}} \right) - \frac{\partial \mathcal{L}}{\partial z} &= 0 \ . \end{aligned} \tag{2.13}$$

On the other side, using Eq.(2.12) one can obtain that

$$\frac{\partial \mathcal{L}}{\partial \dot{x}} = \frac{\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}}\mathcal{L} = \mathcal{L}\dot{x} = \frac{1}{c}\frac{dx}{ds} \ ,$$

$$\frac{\partial \mathcal{L}}{\partial \dot{y}} = \frac{\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}} \mathcal{L} = \mathcal{L}\dot{y} = \frac{1}{c}\frac{dy}{ds} \ ,$$
$$\frac{\partial \mathcal{L}}{\partial \dot{z}} = \frac{\dot{z}}{\sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}} \mathcal{L} = \mathcal{L}\dot{z} = \frac{1}{c}\frac{dz}{ds} \ .$$

Therefore, one can conclude that the solution of the Eikonal equation requires the solution of the following system of equations:

$$\frac{d}{ds}\left(\frac{1}{c}\frac{dx}{ds}\right) = \frac{\partial}{\partial x}\left(\frac{1}{c}\right) \quad , \quad \frac{d}{ds}\left(\frac{1}{c}\frac{dy}{ds}\right) = \frac{\partial}{\partial y}\left(\frac{1}{c}\right) \quad ,$$
$$\frac{d}{ds}\left(\frac{1}{c}\frac{dz}{ds}\right) = \frac{\partial}{\partial z}\left(\frac{1}{c}\right) \ . \tag{2.14}$$

2.2.2 Sound slowness

The reciprocal of sound speed is present sistematically in the system given by Eq.(2.14). Such fact suggest that the system can be greatly simplified by defining the reciprocal of sound speed as a parameter of its own, called sound slowness σ:

$$\sigma = \frac{1}{c} \ . \tag{2.15}$$

With this new parameter the system Eq.(2.14) becomes

$$\frac{d}{ds}\left(\sigma\frac{dx}{ds}\right) = \frac{\partial \sigma}{\partial x} \ , \quad \frac{d}{ds}\left(\sigma\frac{dy}{ds}\right) = \frac{\partial \sigma}{\partial y} \ , \quad \frac{d}{ds}\left(\sigma\frac{dz}{ds}\right) = \frac{\partial \sigma}{\partial z} \ . \tag{2.16}$$

An additional simplification can be achieved by considering sound slowness as a vector parameter:

$$\boldsymbol{\sigma} = [\sigma_x \ , \sigma_y \ , \ \sigma_z] \ . \tag{2.17}$$

Following this new definition each term inside parenthesis implies that

$$\frac{dx}{ds} = \frac{\sigma_x}{\sigma} \quad , \quad \frac{dy}{ds} = \frac{\sigma_y}{\sigma} \quad , \quad \frac{dz}{ds} = \frac{\sigma_z}{\sigma} \quad , \tag{2.18}$$

which transforms the system Eq.(2.16) into

$$\frac{d\sigma_x}{ds} = \frac{\partial \sigma}{\partial x} \quad , \quad \frac{d\sigma_y}{ds} = \frac{\partial \sigma}{\partial y} \quad , \quad \frac{d\sigma_z}{ds} = \frac{\partial \sigma}{\partial z} \quad , \tag{2.19}$$

or, more compactly, in vector form:

$$\frac{d\boldsymbol{\sigma}}{ds} = \boldsymbol{\nabla}\sigma \ . \tag{2.20}$$

2.2.3 Cilindrical symmetry (two-dimensional case)

In a system with cilindrical symmetry (see Fig.2.1) the Eikonal equation follows directly from Eq.(2.18) and Eq.(2.19), by replacing x with r and nullifying any derivative along y:

$$\frac{dr}{ds} = \frac{\sigma_r}{\sigma} \ , \qquad \frac{dz}{ds} = \frac{\sigma_z}{\sigma} \ ; \tag{2.21}$$

under this conditions the system Eq.(2.16) becomes

$$\frac{d\sigma_r}{ds} = \frac{\partial \sigma}{\partial r} \ , \qquad \frac{d\sigma_z}{ds} = \frac{\partial \sigma}{\partial z} \ . \tag{2.22}$$

In Eq.(2.22) $\sigma_r(s)$ and $\sigma_z(s)$ represent, respectively, the horizontal and vertical components of sound slowness:

$$\boldsymbol{\sigma}(s) = [\sigma_r(s) \ , \ \sigma_z(s)] \ .$$

The differential of travel time can be written as

$$d\tau = \frac{ds}{c} = \sigma \ ds = \frac{\sigma^2}{\sigma_r} dr \ .$$

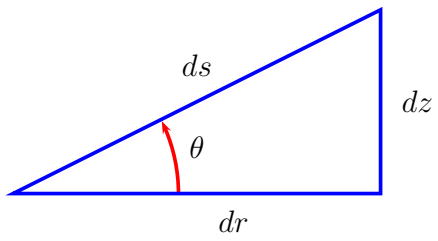


Figure 2.1: Ray slope, θ, ray step ds and horizontal and vertical steps dr and dz for the case of cilindrical symmetry; the parameters are related through the following relationships:

$$dr = \cos\theta \ ds \ , \ dz = \sin\theta \ ds \ , \ \tan\theta = dz/dr \ , \ ds = \sqrt{(dr)^2 + (dz)^2} \ .$$

The system of equations given by Eq.(2.21) and Eq.(2.22) can be rewritten also in a more classical fashion as[3]

$$\begin{aligned} \frac{dr}{ds} &= c(s)\sigma_r(s) \ , \quad \frac{d\sigma_r}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial r} \ , \\ \frac{dz}{ds} &= c(s)\sigma_z(s) \ , \quad \frac{d\sigma_z}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial z} \ . \end{aligned} \tag{2.23}$$

or, more compactly, in vector form:

$$\frac{d\mathbf{r}}{ds} = c(s)\boldsymbol{\sigma}(s) \quad , \quad \frac{d\boldsymbol{\sigma}}{ds} = -\frac{1}{c^2}\boldsymbol{\nabla} c \quad . \tag{2.24}$$

When sound speed depends on depth only the horizontal slowness is preserved:

$$\frac{d\sigma_r}{ds} = 0 \ ;$$

this, in combination with flat boundaries, allows to infer the classical form of Snell's law along the ray:

$$\sigma_r(s) = \frac{\cos\theta(s)}{c(s)} = \text{ constant } \ . \tag{2.25}$$

The set of initial conditions required to solve the two-dimensional form of the Eikonal equation is given by

$$r(0) = r_0 \ , \quad z(0) = z_0 \ , \quad \sigma_r(0) = \frac{\cos\theta(0)}{c(0)} \ , \quad \sigma_z(0) = \frac{\sin\theta(0)}{c(0)} \ ,$$

where $\theta(0)$ represents the launching angle, $[r_0, z_0]$ stands for the source position and $c(0)$ is the sound speed at the source position.

2.3 Solving the transport equation

The solution of the Eikonal equation allows to calculate the Jacobian J, between the usual set of cartesian coordinates (x, y, z) and a particular set of ray coordinates (s, α_1, α_2), where s stands for the ray arclenght and $\alpha_{1,2}$ stand for some sort of auxiliary ray angles. In general, J represents the cross section of a ray tube propagating from the source to the receiver. Let us define $\mathbf{e}_s$ as the unitary vector tangent to the ray:

$$\mathbf{e}_s = \begin{bmatrix} dx/ds \\ dy/ds \\ dz/ds \end{bmatrix} \ . \tag{2.26}$$

Therefore, the first term in the transport equation, combined with the Eikonal equation, can be rewritten as

$$\boldsymbol{\nabla} A \cdot \boldsymbol{\nabla}\tau = \frac{dA}{ds}\mathbf{e}_s \cdot \frac{d\tau}{ds}\mathbf{e}_s = \frac{dA}{ds}\mathbf{e}_s \cdot \frac{1}{c}\mathbf{e}_s = \frac{1}{c}\frac{dA}{ds} \ .$$

As for the second term, let us notice that the analytical properties of the Jacobian[3] allow to write that

$$\boldsymbol{\nabla}^2\tau = \boldsymbol{\nabla}\cdot\boldsymbol{\nabla}\tau = \frac{1}{J}\frac{d}{ds}\left(\frac{J}{c}\right) \ .$$

The previous expressions transform the transport equation into

$$\frac{2}{c}\frac{dA}{ds} + \frac{A}{J}\frac{d}{ds}\left(\frac{J}{c}\right) = 0 \ , \tag{2.27}$$

which can be easily integrated to provide the result

$$A = \frac{A_0}{\sqrt{J/c(s)}} \ ,$$

where A_0 stands for a constant, which depends of the type of acoustic source. The solution to the wave equation can then be written as

$$P(\mathbf{r},\omega) = A_0\sqrt{\frac{c(s)}{J}}\ e^{-i\omega\tau} \ .$$

In the proximity of a point source one can identify the ray coordinates (s,α_1,α_2) with the spherical coordinates (s,θ,ϕ), centered at the source position (see Fig.2.2); therefore, it can be written that

$$c(s) \approx c(0) \quad \text{and} \quad J \approx s^2\cos\theta(0) \ ,$$

with $\theta(0)$ standing for the launching angle. On the oher side, close to the source, the acoustic field can be approximated as a spherical wave, which implies that

$$A_0\sqrt{\frac{c(0)}{s^2\cos\theta(0)}}\ e^{-i\omega s/c(0)} = \frac{1}{4\pi s}\ e^{-i\omega s/c(0)} \ ;$$

it follows from this relationship that

$$A_0 = \frac{1}{4\pi}\sqrt{\frac{\cos\theta(0)}{c(0)}} \ .$$

The classical solution can then be written explicitely as

$$P(\mathbf{r},\omega) = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos\theta(0)}{J}}\ e^{-i\omega\tau} \ . \tag{2.28}$$

Unfortunately, the classical solution based on the Jacobian suffers from a serious drawback. Each time a ray tube narrows into a single point the Jacobian becomes zero (the points at which $J=0$ are called *caustics*), and the acoustic field exhibits a singularity at such point. Removing such singularities from the solution can be achieved by substituting the classical solution with the solution based on Gaussian beams.

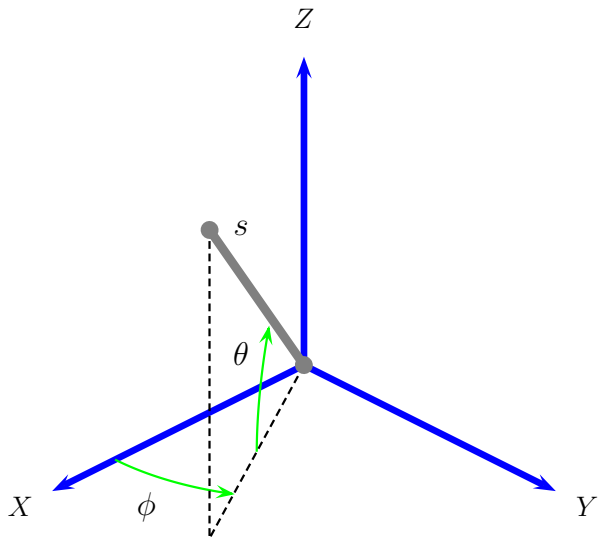


Figure 2.2: Ray coordinates in the proximity to the source.

2.4 Transmission loss

The definition of transmission loss according to [2] is

$$TL = -20 \log \left| \frac{P(\mathbf{r}, \omega)}{P_0} \right| \ , \tag{2.29}$$

where

$$P_0 = 1/(4\pi) \ .$$

Bellhop however, calculates transmission loss as

$$TL = -20 \log |P(\mathbf{r}, \omega)| \ . \tag{2.30}$$

The same definition is adopted by **TRACEO**.

2.5 Particle velocity

Particle velocity calculations are becoming more and more relevant in different applications of underwater acoustics, in part due to the recent development of vector sensor arrays[6]. Particle velocity can be calculated from acoustic pressure using the relationship[7]

$$\mathbf{v} = -\frac{i}{\omega \rho} \boldsymbol{\nabla} P \ , \tag{2.31}$$

where ρ represents the watercolumn density, and ω stands for the frequency of the propagating wave.

Chapter 3

Gaussian beams

3.1 The Gaussian beam approximation

The starting point for the approximation of the acoustic field as a Gaussian beam is given by the analytical expression

$$P(s,\mathbf{n}) = \frac{1}{4\pi}\sqrt{\frac{c_0(s)}{c_0(0)}\frac{\cos\theta(0)}{\det\mathbf{Q}}}\exp\left\{-i\omega\left[\tau(s)+\frac{1}{2}\left(\mathbf{Mn}\cdot\mathbf{n}\right)\right]\right\}\ , \tag{3.1}$$

where $c_0(s)$ stands for sound speed along a ray trajectory:

$$c_0(s) = c(s,\mathbf{0})$$

and $\mathbf{n}$ represents the normal to the ray, which can be considered as two-dimensional vector:

$$\mathbf{n} = \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}\ ,$$

such that $\mathbf{n}\cdot\mathbf{e}_s = 0$, where $\mathbf{e}_s$ was already defined by Eq.(2.26). In what it follows the projection of sound slowness along $\mathbf{n}$ will be written as $\boldsymbol{\sigma}_n$.

While $\mathbf{n}$ is a real vector both matrices $\mathbf{M}$ and $\mathbf{Q}$ are complex. Therefore, the complex part of the product $\mathbf{Mn}\cdot\mathbf{n}$ induces a Gaussian decay of ray amplitude along the normal (see Fig.3.1), while the real part introduces phase corrections to the travel time along $\mathbf{n}$. Additionally, a proper choice of initial conditions for $\mathbf{Q}$ will ensure that $\det\mathbf{Q} \neq 0$, which frees the Gaussian beam approximation of singularities.

Both components of $\mathbf{n}$ and $\boldsymbol{\sigma}_n$ can be considered as being dependent of a particular set of local ray parameters, let's say, ray arclength s, plus angles α_1 and α_2. At any point of the ray one can introduce a set of three orthogonal unit vectors, known as the *polarization vectors*; naturally, the

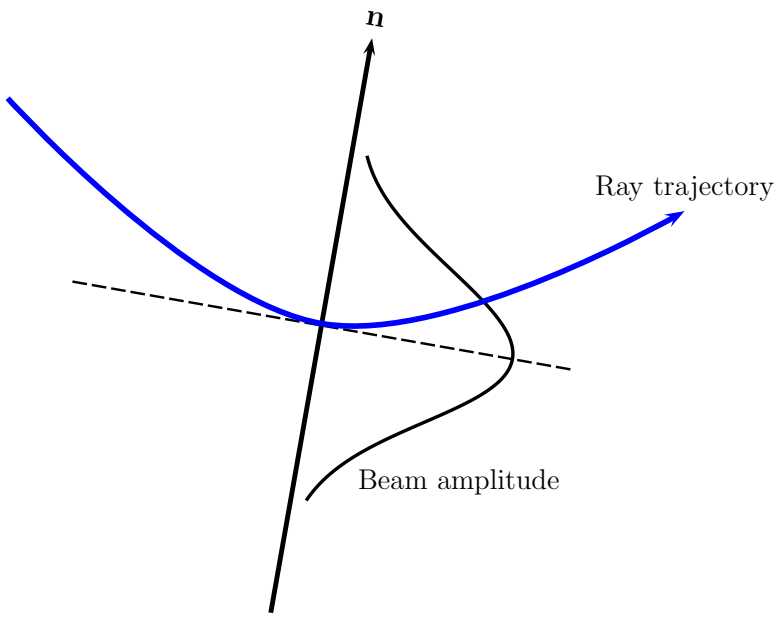


Figure 3.1: Gaussian beams: amplitude decay along the normal.

first polarization vector is $\mathbf{e}_s$; the other two polarization vectors, which are going to be represented as $\mathbf{e}_1$ and $\mathbf{e}_2$, are within the plane perpendicular to $\mathbf{e}_s$ (see Fig.3.2).

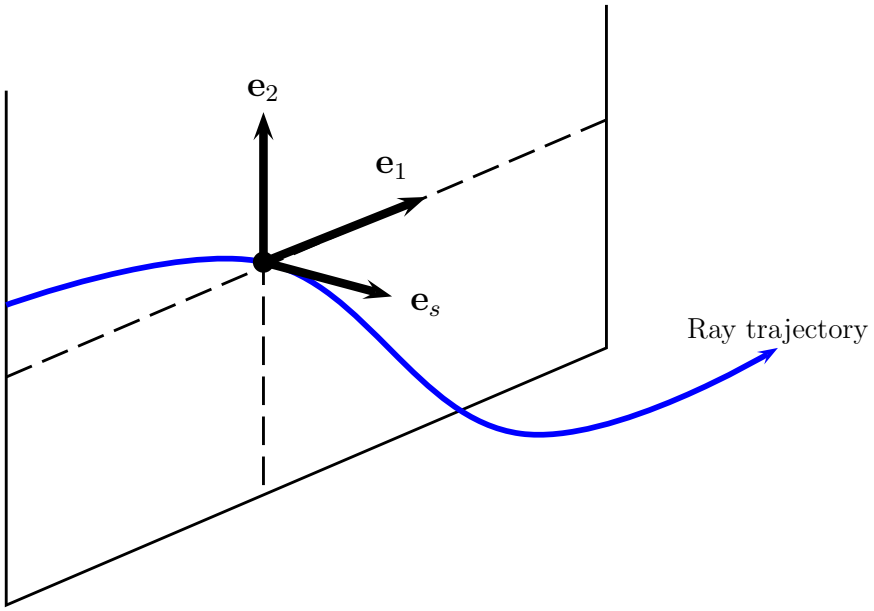


Figure 3.2: Gaussian beams: polarization vectors.

The vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ define the possible orientations of $\mathbf{n}$ and $\boldsymbol{\sigma}_n$ at any coordinate s of the ray:

$$\mathbf{n} = n_1\mathbf{e}_1 + n_2\mathbf{e}_2 \quad \text{and} \quad \boldsymbol{\sigma}_n = \sigma\mathbf{e}_1 + \sigma\mathbf{e}_2 \ .$$

Besides matrices $\mathbf{Q}$ and $\mathbf{M}$ the Gaussian beam approximation involves two more matrices, called $\mathbf{P}$ and $\mathbf{C}$; all four matrices are related through the following relationships:

$$\frac{d}{ds}\mathbf{Q} = c_0\mathbf{P} \quad , \quad \frac{d}{ds}\mathbf{P} = -\frac{1}{c_0^2}\mathbf{C}\mathbf{Q} \quad , \quad \mathbf{M} = \mathbf{P}\mathbf{Q}^{-1} \quad , \tag{3.2}$$

where

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \ , \tag{3.3}$$

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \ , \tag{3.4}$$

and

$$\mathbf{C} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \ , \tag{3.5}$$

where

$$C_{ij} = \frac{\partial^2 c}{\partial n_i \partial n_j} \ .$$

The components of $\mathbf{P}$ and $\mathbf{Q}$ correspond to partial derivatives of the ray normal and normal slowness, along the auxiliary parameters α_1 and α_2[5]:

$$p_{ij} = \frac{\partial \sigma_i}{\partial \alpha_j} \quad \text{and} \quad q_{ij} = \frac{\partial n_i}{\partial \alpha_j} \quad .$$

3.2 Initial conditions on P and Q

Generally speaking, the only way to ensure that the matrices $\mathbf{P}(s)$ and $\mathbf{Q}(s)$ are complex, being $\mathbf{C}(s)$ a real matrix, is by selecting a proper choice of complex initial conditions for both matrices. As shown by the literature such proper choice is a matter of intense debate, seemingly not yet solved. There are however particular choices of real $\mathbf{P}(0)$ and $\mathbf{Q}(0)$, which are relevant for the discussion of Gaussian beams. Due to such relevance those choices are presented in this section for the case of a point source and for the case of a linear source.

3.2.1 Point source

For the case of a point source the conditions can be shown to be[5]:

$$\mathbf{P}(0) = \begin{bmatrix} 1/c(0,0,0) & 0 \\ 0 & \cos\theta(0)/c(0,0,0) \end{bmatrix}$$

and[8]

$$\mathbf{Q}(0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} ,$$

where $c(0,0,0)$ stands for sound speed at the source position.

3.2.2 Line source

For the case of a line source the conditions become[5]

$$\mathbf{P}(0) = \begin{bmatrix} 0 & 0 \\ 0 & 1/c(0,0,0) \end{bmatrix}$$

and

$$\mathbf{Q}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} .$$

3.3 Reduction to the 2.5D case

Let us consider an environment with the following set of conditions:

$$\mathbf{C} = \begin{bmatrix} c_{nn} & 0 \\ 0 & 0 \end{bmatrix}$$

(i.e., there is no dependence on the second component of $\mathbf{n}$, and indexing is being omitted by obvious reasons) and

$$\mathbf{Q}(s) = \begin{bmatrix} q(s) & 0 \\ 0 & q_{\perp}(s) \end{bmatrix}$$

so the matrix contains only diagonal elements, and we introduced the notation

$$q(s) = q_{11}(s) \quad , \quad q_{\perp}(s) = q_{22}(s) ,$$

so $\det\mathbf{Q} = q(s)q_{\perp}(s)$. It follows from the system given by Eq.(3.2) that

$$p_{12}(s) = p_{21}(s) = 0 ,$$

and

$$\frac{d}{ds}q = c_0\, p \quad , \quad \frac{d}{ds}p = -\frac{c_{nn}}{c_0^2}\, q \ ,$$

$$\frac{d}{ds}q_\perp = c_0\, p_\perp \quad , \quad \frac{d}{ds}p_\perp = 0$$

where $p = p_{11}$ and $p_\perp = p_{22}$. The solution of the last equation is trivial and corresponds to

$$p_\perp(s) = p_\perp(0) = \text{constant} \ .$$

It follows further that

$$q_\perp(s) = p_\perp(0) I_c(s) + q_\perp(0) \ ,$$

where

$$I_c(s) = \int c_0(s) ds \ .$$

Recalling $\mathbf{n}$ as two-dimensional vector one gets that

$$\mathbf{Mn} \cdot \mathbf{n} = \frac{p(s)}{q(s)} n_1^2 + \frac{1}{I_c(s)} n_2^2 \ . \tag{3.6}$$

In the case of a point source it follows that

$$q_\perp(s) = \frac{\cos\theta(0)}{c(0,0,0)} I_c(s) \sim s \ ,$$

which indicates that the parameter is proportional to the ray path.

3.4 Reduction to the 2D case

The 2D case is identified here as the waveguide with cylindrical symmetry discussed in section 2.2.3, so the coordinates r, z and θ stand for horizontal distance, depth and ray slope related to the horizontal, respectively. The Gaussian beam expression for this case follows readily from the expression for the 2.5 case, by taking $n_2 = 0$ and $n = n_1$, which provides the expression

$$P(s,n) = \frac{1}{4\pi} \sqrt{\frac{c(s)}{c(0)} \frac{\cos\theta(0)}{q_\perp(s) q(s)}} \ \exp\left[-i\omega \left(\tau(s) + \frac{1}{2} \frac{p(s)}{q(s)} n^2 \right) \right] \ ; \tag{3.7}$$

in this expression $p(s)$ and $q(s)$ are related through the so-called *dynamic equations*:

$$\frac{dq}{ds} = c(s) p(s) \quad , \quad \frac{dp}{ds} = -\frac{c_{nn}}{c^2} q(s) \ . \tag{3.8}$$

The second-order derivative along the normal can be written in terms of derivatives along r and z as[1]

$$c_{nn} = \left(\frac{dr}{dn}\right)^2 c_{rr} + 2\left(\frac{dr}{dn}\right)\left(\frac{dz}{dn}\right) c_{rz} + \left(\frac{dz}{dn}\right)^2 c_{zz} \ , \tag{3.9}$$

where

$$c_{rr} = \frac{\partial^2 c}{\partial r^2} \quad , \quad c_{zz} = \frac{\partial^2 c}{\partial z^2} \quad , \quad c_{rz} = \frac{\partial^2 c}{\partial r \partial z} \ ,$$

and

$$\frac{dr}{dn} = -\sin\theta \quad , \quad \frac{dz}{dn} = \cos\theta \quad .$$

Those derivatives can be identified as the components of the polarization vector $\mathbf{e}_1(s)$, which in the 2D case correspond to:

$$\mathbf{e}_s(s) = \begin{bmatrix} \cos\theta(s) \\ \sin\theta(s) \end{bmatrix} \quad \text{and} \quad \mathbf{e}_1(s) = \begin{bmatrix} -\sin\theta(s) \\ \cos\theta(s) \end{bmatrix}$$

(see Fig.3.3).

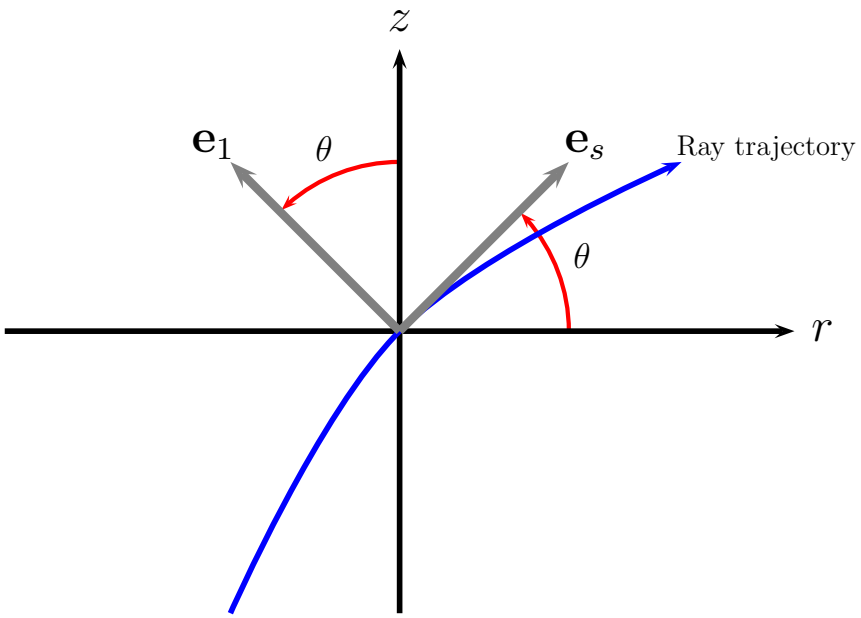


Figure 3.3: Polarization vectors for the 2D case.

The beam width and curvature, $L(s)$ and $K(s)$, respectively[1, 9], can be calculated from $p(s)$ and $q(s)$ by comparing the following expressions:

$$i\omega\frac{1}{2}\frac{p}{q}n^2 = \frac{i\omega K(s)}{2c(s)}n^2 + \frac{1}{L^2}n^2 \ ,$$

which yields that

$$K(s) = c(s)\mathrm{Re}\left[\frac{p(s)}{q(s)}\right] \ , \tag{3.10}$$

and

$$L(s) = \sqrt{\frac{-2}{\omega \mathrm{Re}\left[\frac{p(s)}{q(s)}\right]}} \ . \tag{3.11}$$

As shown by Eq.(3.11) the Gaussian beam approximation requires that $p(s)/q(s) > 0$. To this end it would be sufficient to select a non-zero real value for $p(0)$, plus the choice $q(0) = i\varepsilon$, being ε a real number, as small as possible.

Chapter 4

Attenuation

As shown in the previous chapters the general solution for the acoustic pressure can be written as

$$P(\mathbf{r}, \omega) = Ae^{-i\omega\tau}$$

with

$$A = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos\theta(0)}{J}}$$

for the classical solution, and

$$A = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos\theta(0)}{Sq(s)}}\ \exp\left[-\frac{1}{2}i\omega\frac{p(s)}{q(s)}n^2\right]$$

for the Gaussian beam approximation. Either expression does not take into account the dissipation of energy due to material absorption, and due to the transfer of energy every time the acoustic wave bounces on a boundary. Such dissipation is frequency-dependent (and more relevant as frequency increases), and implies using a corrected amplitude a, which corresponds to the original amplitude, multiplied by two decaying factors ϕ_r and ϕ_V:

$$a = A \times \phi_r \times \phi_V$$

where ϕ_r and ϕ_V stand, respectively, for the decay due to boundary reflections, and due to volume absorption. Both factors are described in the following sections.

4.1 Boundary reflections

The decaying factor ϕ_r is given by the expression

$$\phi_r = \prod_{i=1}^{n_r} R_i\ , \tag{4.1}$$

where n_r represents the total number of boundary reflections, and R_i is the reflection coefficient at the ith reflection. The case with no reflections ($n_r = 0$) corresponds to $\phi_r = 1$. Generally speaking, boundaries can be one of four types:

- Absorvent: the wave energy is transmitted completely to the medium above the boundary, so $R = 0$ and ray propagation is terminated at the boundary.
- Rigid: the wave energy is reflected completely on the boundary, with no phase change, so $R = 1$.
- Vacuum: the wave energy is reflected completely on the boundary, with a phase change of π radians, so R = -1.
- Elastic: the wave energy is partially reflected, with R being a complex value and $|R| < 1$.

The calculation of the reflection coefficient for an elastic medium (see Fig.4.1) is given by the following expression[10]:

$$R(\theta) = \frac{D(\theta)\cos\theta - 1}{D(\theta)\cos\theta + 1} \ , \tag{4.2}$$

where

$$D(\theta) = A_1 \left(A_2 \frac{1 - A_7}{\sqrt{1 - A_6^2}} + A_3 \frac{A_7}{\sqrt{1 - A_5/2}} \right) \ ,$$

$$A_1 = \frac{\rho_2}{\rho_1} \quad , \quad A_2 = \frac{\tilde{c}_{p2}}{c_{p1}} \quad , \quad A_3 = \frac{\tilde{c}_{s2}}{c_{p1}} \quad ,$$

$$A_4 = A_3 \sin\theta \quad , \quad A_5 = 2A_4^2 \quad , \quad A_6 = A_2 \sin\theta \quad , \quad A_7 = 2A_5 - A_5^2 \quad ,$$

$$\tilde{c}_{p2} = c_{p2} \frac{1 - i\tilde{\alpha}_{cp}}{1 + \tilde{\alpha}_{cp}^2} \quad , \quad \tilde{c}_{s2} = c_{s2} \frac{1 - i\tilde{\alpha}_{cs}}{1 + \tilde{\alpha}_{cs}^2} \quad ,$$

$$\tilde{\alpha}_{cp} = \frac{\alpha_{cp}}{40\pi \log e} \quad , \quad \tilde{\alpha}_{cs} = \frac{\alpha_{cs}}{40\pi \log e} \quad ,$$

where the units of attenuation should be given in dB/λ.

In general the reflection coefficient is real when $\alpha_{cp} = \alpha_{cs} = 0$, and the angle of incidence θ is less than the critical angle θ_{cr}, with θ_{cr} given by the expression

$$\theta_{cr} = \arcsin\left(\frac{c_{p1}}{c_{p2}}\right) \ . \tag{4.3}$$

Moreover, attenuation is negligible when $\theta < \theta_{cr}$, and for small θ the energy transfered to shear waves in the elastic medium is only a small fraction of the total energy transfered.

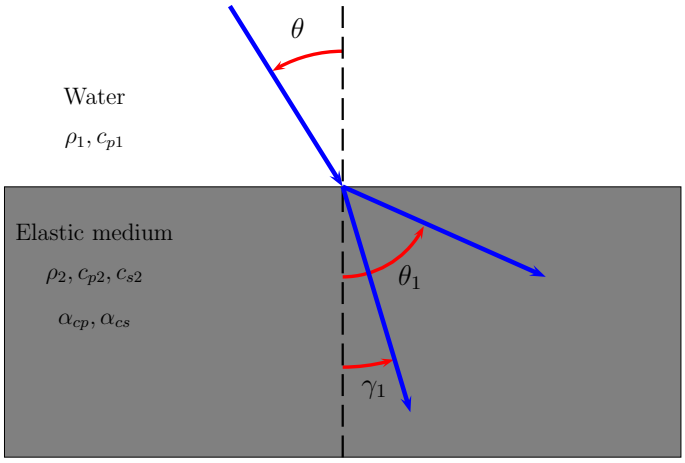


Figure 4.1: Ray reflection on an elastic media.

4.2 Volume attenuation

Volume attenuation in the ocean has a chemical nature, and it is induced by relaxation processes of salt constituents like $MgSO_4$, $B(OH)_3$ and $MgCO_3$.

The factor ϕ_V is given by the decaying exponential

$$\phi_V = \exp\left(-\alpha_T s\right) \ , \tag{4.4}$$

where s is the ray arclength and α_T is the Thorpe (frequency dependent) attenuation coefficient in dB/m, given by[2]

$$\alpha_T = \frac{40 f^2}{4100 + f^2} + \frac{0.1 f^2}{1 + f^2} \ , \tag{4.5}$$

with the frequency given in kHz.

Chapter 5

Numerical issues

5.1 Geometric beams

As shown by numerical calculations the solution of the dynamic equations (see Eq.(3.8)) with complex values induces the formation of weird artifacts, which reveal themselves as patterns of self-interference after the ray is reflected on a boundary. In some cases such interference is unrealistically intense despite the expected exponential decay in amplitude along the normal direction. To erradicate such artifacts **TRACEO** uses the approximation of geometric beams[11, 12], which relies on the following expression for the calculation of the acoustic field:

$$P(s,n) = \phi(s,n) a e^{-i[\omega\tau(s,n)-\phi_c]} \ ; \tag{5.1}$$

in the above expression $\phi(s,n)$ is a hat window (see Fig.5.1), defined as

$$\phi(s,n) = \begin{cases} \dfrac{W(s)-n(s)}{W(s)} & \text{when } n(s) \leq W(s) \\ \\ 0 & \text{otherwise} \end{cases} \ , \tag{5.2}$$

with $W(s)$ representing the beam width:

$$W(s) = \left| \frac{q(s)\Delta\theta}{c(0)\cos\theta(s)} \right| \ ; \tag{5.3}$$

in this expression $\Delta\theta$ stands for the angle step. Moreover, the dynamic equations are integrated using the initial conditions:

$$p(0) = 1 \quad \text{and} \quad q(0) = 0 \ ,$$

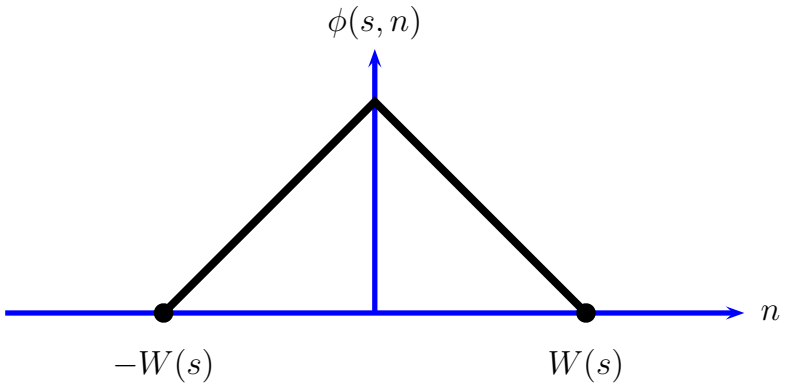


Figure 5.1: Geometric beams: the hat function.

which provide the following expression for the amplitude:

$$a(s) = \sqrt{\frac{c(0)\cos\theta(0)}{q(s)}\frac{c(s)}{I_c(s)}}\,\phi_r(s)\phi_V(s)\ . \tag{5.4}$$

By using real values of $p(s)$ and $q(s)$ it becomes necessary to compensate the travel time along the normal direction; such compensation can be written as

$$\tau(s,n) = \tau(s) + \delta\tau(s,n) \tag{5.5}$$

where $\tau(s)$ is the classical travel time (see Eq.(2.9)); the additional term corresponds to

$$\delta\tau(s,n) = (\Delta\mathbf{r}\cdot\mathbf{e}_s)\frac{d\tau}{ds}\ , \tag{5.6}$$

where

$$\Delta\mathbf{r} = \mathbf{r}_h - \mathbf{r}(s) = \begin{bmatrix} r_h - r(s) \\ z_h - z(s) \end{bmatrix}$$

with $\mathbf{r}_h$ and $\mathbf{r}(s)$ representing, respectively, the position of the hydrophone and the position of the given point along the ray. The total acoustic field is then calculated as a superposition of all ray influences.

A drawback of the geometric beams is that they are affected by caustics, as the classical solution, because of the changes in sign of $q(s)$ as the integration proceeds along the ray arclength. As discussed in [11] the general behaviour of the solution can still be accurate by introducing an amplitude correction ϕ_c into Eq.(5.1), which can be written as

$$\phi_c = (-i)^{m(s)}\ , \tag{5.7}$$

where i represents the imaginary unit, and $m(s)$ corresponds to the number of times that $q(s)$ vanishes in the interval $[0,s]$ (this function is also known as the KMAH index).

5.2 Updating p and q after reflections

Modeling of the propagating wave would be incomplete without properly updating the values of $p(s)$ and $q(s)$ after each boundary reflection. Following Bellhop the method used to update the values is given by:

$$p' = p + qr_n \quad , \quad q' = q \ , \tag{5.8}$$

where p and q stand for the values before reflection, and p' and q' stand for the values after. The correction r_n is given by the expression

$$r_n = r_m \frac{4C_n - 2r_m C_s}{c} \tag{5.9}$$

where c stands for the value of sound speed at the boundary,

$$r_m = T_g / T_h$$

and

$$T_g = (\boldsymbol{\sigma} \cdot \boldsymbol{\tau}_b) \ , \ T_h = (\boldsymbol{\sigma} \cdot \mathbf{n}_b) \ , \ C_n = (\boldsymbol{\nabla} c \cdot \boldsymbol{\sigma}_n) \ , \ C_s = (\boldsymbol{\nabla} c \cdot \boldsymbol{\sigma}) \ ;$$

$\boldsymbol{\tau}_b$ corresponds to the boundary tangent, $\mathbf{n}_b$ is the boundary normal and $\boldsymbol{\nabla} c$ is the sound speed gradient; additionally, $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_n$ stand for the slowness and normal slowness, which can be written as

$$\boldsymbol{\sigma} = \frac{1}{c} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \quad , \quad \boldsymbol{\sigma}_n = \frac{1}{c} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \ ;$$

all parameters should be calculated at the reflection point.

5.3 Solving the Eikonal equations

The Eikonal equations (see Eq.(2.23)) are integrated by **TRACEO** using a method of the Runge-Kutta type, known as the Runge-Kutta-Fehlberg method (hereafter called RKF). As with other Runge-Kutta methods the RKF method starts by rewriting the original system of equations, as a linear differential vector equation

$$\frac{d\mathbf{y}}{ds} = \mathbf{f} \ ,$$

where

$$\mathbf{y} = \begin{bmatrix} r \\ z \\ \sigma_r \\ \sigma_z \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} \sigma_r/\sigma \\ \sigma_z/\sigma \\ \partial\sigma/\partial r \\ \partial\sigma/\partial z \end{bmatrix} \ ;$$

at each step of integration the method proceeds through a set of intermediate steps. After each integration the RKF method provides not one, but two different solutions. If the solutions differe in more than a particular given threshold the ray step ds can be halved, and the integration repeated, providing a mean to control the accuracy of the solution. In order to prevent an infinite loop **TRACEO** stops the halving (and interrupts the calculations) if the sucessive comparisons of solutions fail to converge.

5.4 Solving the dynamic equations

Instead of integrating the dynamic equations simultaneously with the Eikonal equations **TRACEO** relies on the accuracy of ray coordinates, and uses a simple Euler method to integrate the dynamic equations. It had been noticed that the adoption of such strategy greatly simplifies the flow of calculations and the debugging of the code, without compromising accuracy.

5.5 Calculation of derivatives

Calculating the derivatives of sound speed is one (among many) of the most important tasks for an accurate calculation of ray coordinates, and further calculations of ray influence at a given point of the array. To this end **TRACEO** relies on three different types of piecewise interpolation:

- Linear interpolation: the method is trivial and does not require explanation; it is applied every time the interpolation point is located between the first or last pair of tabulated coordinates; being linear, no second derivative is calculated.

- Parabolic interpolation: the method implemented is called *barycentric parabolic interpolation*; it is applied every time the interpolating point is located between the first or last three tabulated coordinates. The method can be described as follows: let us consider a set of three points x_1, x_2 and x_3, and the corresponding function values $f(x_1)$, $f(x_2)$ and $f(x_3)$. At a given point x (see Fig.5.2) the interpolant can be written as

$$f(x) = f(x_1) + a_2\,(x - x_1)\,(x - x_3) + a_3\,(x - x_1)\,(x - x_2)\ . \qquad (5.10)$$

  It follows from this expression that

$$a_2 = \frac{f(x_2) - f(x_1)}{(x_2 - x_1)\,(x_2 - x_3)} \quad \text{e} \quad a_3 = \frac{f(x_3) - f(x_1)}{(x_3 - x_1)\,(x_3 - x_2)}\ .$$

The approximations to the derivatives become

$$\frac{df}{dx} = a_2 \left(2x - x_1 - x_3\right) + a_3 \left(2x - x_1 - x_2\right)$$

and

$$\frac{d^2 f}{dx^2} = 2 \left(a_2 + a_3\right) \ .$$

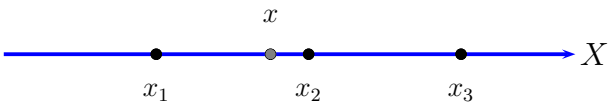


Figure 5.2: 1D Barycentric parabolic interpolation.

- Cubic interpolation: the method of interpolation (not surprisingly called *barycentric cubic interpolation*) is analogous to the previous one, but using four points instead of three. Cubic interpolation is applied when the interpolation point is surrounded by four points (two points back and two points forward).

One of the advantages of the barycentric interpolators is that they can be easily extended to handle different numbers of points, or to handle interpolation and calculation of derivatives at higher dimensions. For instance, the explicit form ot the 2D barycentric parabolic interpolator corresponds to

$$\begin{aligned}
f(x,y) \ = \ & a_{11} \left(x - x_2\right) \left(x - x_3\right) \left(y - y_2\right) \left(y - y_3\right) \ + \\
+ \ & a_{12} \left(x - x_1\right) \left(x - x_3\right) \left(y - y_2\right) \left(y - y_3\right) \ + \\
+ \ & a_{13} \left(x - x_1\right) \left(x - x_2\right) \left(y - y_2\right) \left(y - y_3\right) \ + \\
+ \ & a_{21} \left(x - x_2\right) \left(x - x_3\right) \left(y - y_1\right) \left(y - y_3\right) \ + \\
+ \ & a_{22} \left(x - x_1\right) \left(x - x_3\right) \left(y - y_1\right) \left(y - y_3\right) \ + \\
+ \ & a_{23} \left(x - x_1\right) \left(x - x_2\right) \left(y - y_1\right) \left(y - y_3\right) \ + \\
+ \ & a_{31} \left(x - x_2\right) \left(x - x_3\right) \left(y - y_1\right) \left(y - y_2\right) \ + \\
+ \ & a_{32} \left(x - x_1\right) \left(x - x_3\right) \left(y - y_1\right) \left(y - y_2\right) \ + \\
+ \ & a_{33} \left(x - x_1\right) \left(x - x_2\right) \left(y - y_1\right) \left(y - y_2\right) \ ,
\end{aligned} \tag{5.11}$$

(see Fig.5.3) where the coefficients can be calculated as:

$$a_{11} = \frac{f(x_1, y_1)}{\left(x_1 - x_2\right) \left(x_1 - x_3\right) \left(y_1 - y_2\right) \left(y_1 - y_3\right)} \ ,$$

$$a_{12} = \frac{f(x_2, y_1)}{\left(x_2 - x_1\right) \left(x_2 - x_3\right) \left(y_1 - y_2\right) \left(y_1 - y_3\right)} \ ,$$

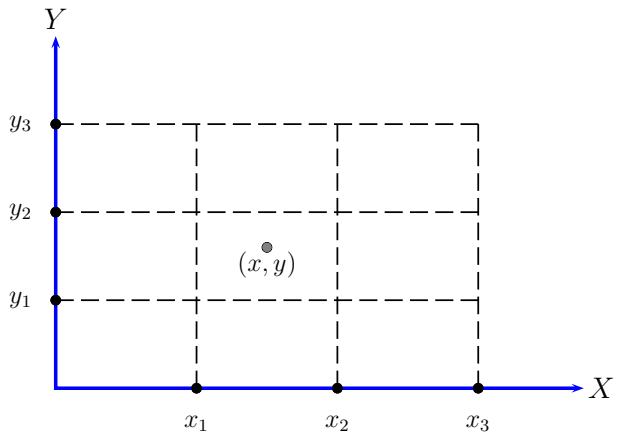


Figure 5.3: 2D Barycentric parabolic interpolation.

...

Using the notation

$$p_i(x) = \prod_{j=1, j\neq i}^{3} d_j(x) \ , \quad d_i(x) = (x - x_i) \quad \text{and} \quad S_i(x) = 2x - \sum_{j=1, j\neq i}^{3} x_j \tag{5.12}$$

one can write the 2D interpolator compactly as

$$f(x, y) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} \ p_j(x) \ p_i(y) \ , \tag{5.13}$$

with the coefficients given by

$$a_{ij} = \frac{f(x_j, y_i)}{p_j(x_j) \ p_i(y_i)} \ .$$

The explicit expressions for the partial derivatives correspond to

$$\frac{\partial f}{\partial x} = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} \ S_j(x) \ p_i(y) \ , \tag{5.14}$$

$$\frac{\partial f}{\partial y} = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} \ p_j(x) \ S_i(y) \ , \tag{5.15}$$

$$\frac{\partial^2 f}{\partial x^2} = 2\sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} \ p_i(y) \ , \tag{5.16}$$

$$\frac{\partial^2 f}{\partial y^2} = 2\sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} \ p_j(x) \ , \tag{5.17}$$

and

$$\frac{\partial^2 f}{\partial x \partial y} = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij}\ S_j(x)\ S_i(y)\ . \tag{5.18}$$

5.6 Calculation of normals

Theoretically, in order to calculate the ray influence as accurately as possible the vector $\Delta\mathbf{r}$ (see section 5.1) is required to be perpendicular to the polarization vector $\mathbf{e}_s(s)$ (or, alternatively, parallel to $\mathbf{e}_1(s)$). For an arbitrary position of the hydrophone such condition implies traveling along the ray, until the condition $(\Delta\mathbf{r} \cdot \mathbf{e}_s) = 0$ is fullfiled. In such case the normal corresponds to

$$n = |\Delta\mathbf{r}|\ .$$

Numerical calculations indicate that there is no significant loss of accuracy (and requires much less computations) to take

$$\Delta\mathbf{r} = \begin{bmatrix} 0 \\ z_h - z(s) \end{bmatrix}\ ,$$

(i.e., at each hydrophone range one interpolates ray depth, so $\Delta\mathbf{r}$ becomes a vertical vector, connecting the ray to the hydrophone) and to calculate the normal as

$$n = |\Delta\mathbf{r} \cdot \mathbf{e}_1(s)|\ .$$

5.7 Refraction correction

Besides the integration of the dynamic equations, and the update of $p(s)$ and $q(s)$ after reflections, **TRACEO** borrows from Bellhop an undocumented correction of refraction, which can be written as

$$p' = p + q\ \tilde{r}_n\ ,$$

where $'$ stands for the corrected value,

$$\tilde{r}_n = -\tilde{r}_m \frac{(2C_{nj} - \tilde{r}_m C_{sj})}{c}\ ,$$

$$\tilde{r}_m = \sigma_r(s)/\sigma_z(s)\ ,$$

$$C_{nj} = \delta\boldsymbol{\nabla} c \cdot \boldsymbol{\sigma}_n \quad , \quad C_{sj} = \delta\boldsymbol{\nabla} c \cdot \boldsymbol{\sigma} \quad ,$$

c represents the sound speed at the arrival position, and $\delta\boldsymbol{\nabla} c$ represents the "jump" of the gradient, i.e., the variation of $\boldsymbol{\nabla} c$ between the initial and final positions.

5.8 Ray reflection at a boundary

The laws of specular reflection simply state that the angle of reflection should be equal to the angle of incidence, and that the incident ray, together with the surface normal and the reflected ray, should all lie in a common plane. Implementing such statement in terms of angles can become quite inefficient if one keeps in mind that the boundary is not necessarily flat; in such case the calculation of the angles of incidence and reflection requires calculating ray and boundary slopes, using inverse trigonometric functions to get the ray/boundary angles relative to the horizontal, and figuring out how to combine them. Such approach can become quite cumbersome. An efficient method for the calculation of ray reflection consists in using the expression

$$\mathbf{e}'_s = \mathbf{e}_s + (2\cos\theta_1)\,\mathbf{n}_b \ , \tag{5.19}$$

where $\mathbf{n}_b$ represents the boundary's normal at the point of incidence, $'$ stands for the values after reflection, and

$$\cos\theta_1 = \mathbf{n}_b \cdot (-\mathbf{e}_s) \ .$$

Thus, every time a ray intersects a boundary, **TRACEO** calculates the point of ray-boundary intersection, which becomes the point of incidence; the coordinates of the point are used to calculate the boundary's normal, and Eq.(5.19) is applied; the integration of the Eikonal equation is then restarted at the point of incidence (as if the source was now located at the boundary), with the launching angle being defined by the components of $\mathbf{e}'_s$.

5.9 Orientation of boundary normals

When a boundary is interpolated **TRACEO** calculates the normal at the interpolation point. Such normal always points towards the bottom. However, in order to calculate correctly the reflection of the ray at the bottom the bottom's normal is flipped in the opposite direction, so the ray returns to the watercolumn. As for a reflection on an object the reflection on the lower side of the object (i.e. the side closer to the surface) is equivalent to a bottom reflection, while the reflection on the upper side (i.e. the side closer to the bottom) is equivalent to a surface reflection.

5.10 Ray-boundary intersection

As **TRACEO** calculates progressively ray coordinates it interpolates the altimetry and the bathymetry at each ray range. It also checks if the range

is inside an object "box", i.e., if ray range is within the range interval, which defines a given object; if the range is outside the object is ignored; otherwise, an additional test is performed to test if the ray is inside the object. If ray depth is above the altimetry, below the bathymetry, or inside the object, **TRACEO** divides the interval between the initial and final positions into n parts, and calculates the initial vertical distance Δz_0, between the ray and the boundary; then, moving progressivel along the ray segment it calculates the product $p = \Delta z_0 \Delta z_i$, with Δz_i representing the vertical distance between the ray and the boundary at the ith step, until a change of sign in p is detected. At such stage the calculation is interrupted, and the intersection is calculated through linear interpolation between the last two points. By proceeding in this way **TRACEO** is able to handle situations, in which the ray segment can intersect the boundary at more than one point.

5.11 Calculation of particle velocity

TRACEO ignores the ω and ρ factors in the calculation of particle velocity (see section 2.5). Therefore, what **TRACEO** presents as particle velocity is nothing more than the horizontal and vertical derivatives of acoustic pressure, multiplied by the complex unit i. To calculate such derivatives at a given position **TRACEO** performs an additional set of calculations, determining the acoustic pressure above and below the given point, and also on both sides. The points aligned along the horizontal are used to calculate the horizontal derivative at the center, using a barycentric parabolic interpolator; likewise, the points aligned along the vertical are used to calculate the vertical derivative at the center. In order to obtain an accurate approximation to the real derivatives the spacing between the points is defined, arbitrarily, as being equal to $\lambda/10$, unless the hydrophones are spaced closer than such value. The advantage of such strategy, despite its computational cost, resides on its independence of the approximation used to calculate the acoustic pressure.

5.12 Eigenray search

Eigenray search is perhaps one of the most difficult problems to tackle within the context of ray tracing. Stated simply as the task of calculating a set of rays, which connect the source to the receiver, it is important to keep in mind that in general rays can be sent backwards to the source. Therefore, if a ray misses the receiver when propagating forward it still has the chance to hit the receiver on its way back to the source. Further complications arise when

the rays can be absorbed by the boundaries, or when small variations of the launching angle lead to non-linear variations of ray trajectories (a situation, typical of the placing of objects inside the waveguide). In order to provide a robust method of eigenray search **TRACEO** uses two different approaches to the problem, namely:

1. If *all* rays are propagating forwards, **TRACEO** interpolates ray depth at each array range for every launching angle. This procedure generates a matrix of the form:

$$\begin{bmatrix} & r_1 & r_2 & r_3 & \dots & r_m \\ \theta_1 & z_1(\theta_1) & z_2(\theta_1) & z_3(\theta_1) & \dots & z_m(\theta_1) \\ \theta_2 & z_1(\theta_2) & z_2(\theta_2) & z_3(\theta_2) & \dots & z_m(\theta_2) \\ \theta_3 & z_1(\theta_3) & z_2(\theta_3) & z_3(\theta_3) & \dots & z_m(\theta_3) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_n & z_1(\theta_n) & z_2(\theta_n) & z_3(\theta_n) & \dots & z_m(\theta_n) \end{bmatrix}$$

   where $z_i(\theta_j)$ represents ray depth at range i and launching angle j. Then, at the ith range, with the hydrophone located at depth z_h, **TRACEO** calculates the function

$$f(\theta) = z_h - z_i(\theta)$$

   using the correspond row of the matrix; if an eigenray exists in the interval i and $i+1$, the function $f(\theta)$ will switch signs between θ_i and θ_{i+1}; in such case the *Regula Falsi* method is used to find the zero of the function. Once the zero is found the ray is calculated, and written to the output file as an eigenray. In order to avoid an infinite loop the eigenray search is interrupted if the number of iterations is greater than a given limit. Particular care is taken in order to deal with rays, which for any reason do not reach the given array range. The search is interrupted if a "returning" ray is detected. The *Regula Falsi* method is computationally accurate and efficient, as long as the function $z(\theta)$ can be properly computed, which won't be the case for returning rays.

2. As an alternative to the previous method **TRACEO** can use a less accurate (but stable) search of eigenrays by *proximity*. At each range ray depth z is calculated, and for each depth (if there is more than one) **TRACEO** calculates the difference

$$|z_h - z| \ ,$$

   where z_h represents hydrophone depth; if the difference is less than a given threshold **TRACEO** writes the ray to the output file as an

eigenray. Certainly, the accuracy of the method depends on the choice of the threshold, and on the number of launching angles (the more, the better). Until a better approach is idealized for dealing with returning rays the proximity method seems to offer a reasonable compromise between accuracy and stability.

Eigenrays are not grouped by **TRACEO** according the coordinates of the array; they are witten progressively, one after another, to the output file. It is up to the user to group them according to the position of each hydrophone.

5.13 Calculation of amplitudes and arrivals

Ray amplitudes and travel times are calculated using the same methods of eigenray search, with the difference of being the only data written to the output file, together with the coordinates of the hydrophone. As in the case of eigenrays it is up to the user to group ray amplitudes and ray arrivals according to the position of each hydrophone.

Chapter 6

Model description

6.1 General strategy of calculations

Fig.6.1 provides a general view of the waveguide handled by **TRACEO**. The source can be located anywhere inside a range "box" (hereafter called `rbox`), and ray tracing is terminated every time a ray exits the box. Placing the source on any of the `rbox` ranges prevents any calculations of being performed. For the sake of numerical stability it is strongly recommended (but not required) that both surface and bottom coordinates are defined beyond the ranges of the `rbox`.

Launching angles can be defined clockwise or counter clockwise, as desired, but launching angles too close to 90° (which are detected by the condition $|\cos\theta| < \varepsilon$) are skipped. Additionally, **TRACEO** uses the convention that rays launched towards the surface have a *positive* launching angle, while negative launching angles indicate propagation of rays towards the bottom. Depending on the reflection coefficient R any boundary waveguide (including objects) can be one of four different types, namely absorbent ($R = 0$), rigid ($R = 1$), vacuum ($R = -1$) or elastic (R is calculated as described in section 4.1). Rays arriving at an absorbing boundary are no longer traced. Another condition for ray termination is that $|R| < \varepsilon$. The array can be horizontal, vertical, rectangular or linear.

The general strategy of calculations can be resumed as follows:

- For every launching angle trace the ray, until it is terminated or exits the `rbox`.
- At every step of ray integration check if the ray is above the surface, below the bottom or inside an object. For a positive check determine the intersection of the ray with the boundary, reflect the ray, and continue with the integration of ray equations.

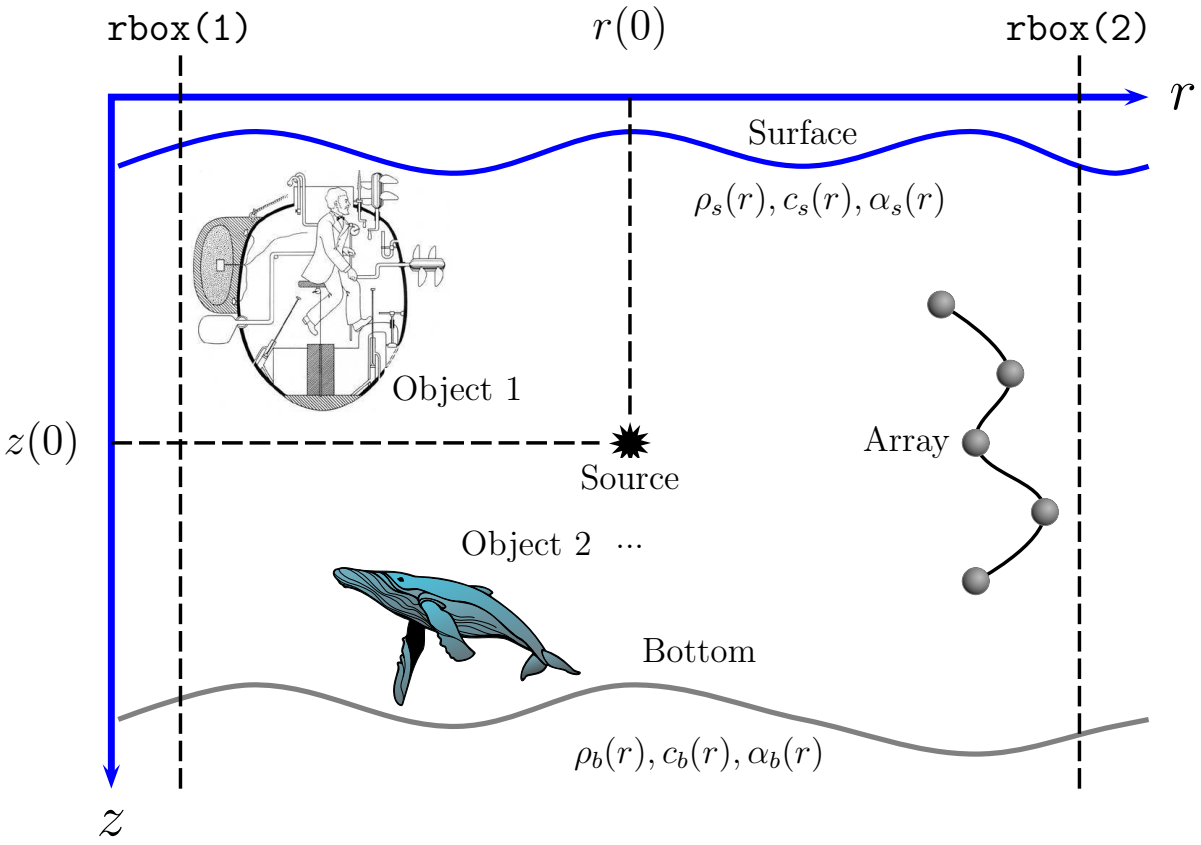


Figure 6.1: General view of **TRACEO**'s waveguide.

- After exiting the `rbox` calculate the (p, q) parameters and the amplitude of the ray.
- Use ray coordinates and amplitudes to calculate the output requested by the user.

6.2 Installation

The Fortran sources of the model are provided with a simple `makefile`. Matlab and the gfortran GNU compiler are required for installation. The `makefile` uses the definitions of Matlab directories and libraries on the machine were the model was originally compiled; before invoking `make` on the command line the user should review the correspondent definitions and adapt them to his local machine. After a sucessful compilation the user can place the resulting binary (`traceo.exe`) and the shell script `runtraceo` in a directory, where the system can find them. The input file has the `in` extension

and can be written from Matlab using the M-file `wtraceoinfil.m`. After creating the input file (for instance, `munk.in`) the user can run the model with the command `runtraceo munk`; according to the desired output the model will create one of the following Matlab mat files:

1. `rco.mat`: ray coordinates;
2. `ari.mat`: ray or eigenray information (ray coordinates, plus travel times and amplitudes);
3. `aad.mat`: arrivals and delays information;
4. `cpr.mat`: coherent acoustic pressure;
5. `ctl.mat`: coherent transmission loss;
6. `pvl.mat`: particle velocity;
7. `pav.mat`: coherent acoustic pressure and particle velocity.

Although many of the output parameters are expected to be complex linkage issues with the Matlab engine forced a workaround of packing complex vectors as real two-dimensional matrices (first row represents the real part, second row the complex part), and separating complex matrices into a real and a complex part. Users are welcome to fix this issue. Besides the Matlab mat files **TRACEO** writes a short `munk.log` ASCII file, describing the type of array and the calculation time. The structure of the input file will be described in the following section.

6.3 The input file

The general structure of the input file (hereafter called INFIL) can be better understood if one thinks of it as composed of blocks; each block describes a particular element of the waveguide, from top to bottom. In order to provide a friendly view of the INFIL the blocks are separated with a long line, which is ignored by the model. The structure of the INFIL is as follows:

```
Title
Source       Block
Altimetry    Block
Sound Speed  Block
Objects      Block
Bathymetry   Block
Array        Block
Output       Block
```

The `Title` is a character string, which is written in the LOGFIL (the file with the `*.log` extension).

The structure of each block is as follows:

`Source Block`:

```
ds                          ray step
rx,zx                       source coordinates
rbox(1),rbox(2)             range box
freqx                       source frequency
nthtas                      number of launching angles
theta(1), theta(nthtas)     first and last launching angles
```

Optionally, the user can set *ds* to zero; in such case **TRACEO** uses the ranges of the `rbox` to calculate a preliminary step.

`Altimetry Block`:

```
 atype  surface type
aptype  surface properties
aitype  interpolation type
atiu    attenuation units
nati    number of surface coordinates
```

`atype` can be one of the following characters:

`'A'`	absorbent surface
`'E'`	elastic surface
`'R'`	rigid surface
`'V'`	vacuum over surface

`aptype` can be one of the following characters:

`'H'`	homogeneous surface
`'N'`	non-homogeneous surface

`aitype` can be one of the following strings:

`'FL'`	flat surface
`'SL'`	surface with a slope
`'2P'`	piecewise linear interpolation
`'3P'`	piecewise parabolic interpolation
`'4P'`	piecewise cubic interpolation

`atiu` can be one of the following characters:

`'F'`	dB/kHz
`'M'`	dB/meter
`'N'`	dB/neper
`'Q'`	Q factor
`'W'`	dB/λ

(same units used by Bellhop; for specific details regarding the definitions of these units see [2]).
`nati` is the number of surafce coordinates.

For `aptype` = `'H'` the properties and coordinates of the surface are specified as follows:

```
cpati(1),csati(1),rhoati(1),apati(1),asati(1)
 rati(1),zati(1)
 rati(2),zati(2)
 rati(3),zati(3)
 ...
 rati(nati),zati(nati)
```

Alternatively, for `aptype` = `'N'` the properties and coordinates of the surface are specified as:

```
 rati(1),zati(1),cpati(1),csati(1),rhoati(1),apati(1),asati(1)
 rati(2),zati(2),cpati(2),csati(2),rhoati(2),apati(2),asati(2)
 rati(3),zati(3),cpati(3),csati(3),rhoati(3),apati(3),asati(3)
 ...
 rati(nati),zati(nati),cpati(nati),...
```

`Sound Speed Block`:

```
 cdist    type of sound speed distribution
 cclass   class of sound speed
 nr0,nz0  number or points in range, number of points in depth
```

`cdist` can be one of the following strings:

`'c(z,z)'`	sound speed profile	$c = c(z)$
`'c(r,z)'`	sound speed field	$c = c(r, z)$

For a sound speed field both range and depth derivatives are calculated using a bi-dimensional barycentric parabolic interpolator, on the grid of `nr0`×`nz0` points. For a sound speed profile all range derivatives are zero; depth derivatives are calculated depending on the value of `cclass`, which can be one of the following strings:

'ISOV'	isovelocity	profile
'LINP'	linear	profile
'PARP'	parabolic	profile
'EXPP'	exponential	profile
'N2LP'	n^2-linear	profile
'ISQP'	inverse-square gradient	profile
'MUNK'	Munk	profile
'TABL'	tabulated	profile

All but the last `cclass` describe analytical profiles, whose derivatives can be calculated explicitely. The parameters of those profiles can be specified through a simple list of the form

```
z0(1),c0(1)
z0(2),c0(2)
```

Here it follows a description of each analytical profile:

- Isovelocity profile:

$$c(z) = c_0 = \text{ constant } \ , \ \frac{dc}{dz} = 0 \ , \ \frac{d^2c}{dz^2} = 0 \ ;$$

  therefore, only the value `c0(1)` is used during the calculations.

- Linear profile:

$$c(z) = c_0 + k\,(z - z_0) \ , \ \frac{dc}{dz} = k \ , \ \frac{d^2c}{dz^2} = 0 \ ;$$

  the parameters are calculated as $z_0 =$ `z(1)`, $c_0 =$ `c(1)` and

$$k = \frac{\texttt{c(2)-c(1)}}{\texttt{z(2)-z(1)}} \ .$$

- Parabolic profile:

$$c(z) = c_0 + k\,(z - z_0)^2 \ , \ \frac{dc}{dz} = 2k\,(z - z_0) \ , \ \frac{d^2c}{dz^2} = 2k \ ;$$

  the parameters are calculated as $z_0 =$ `z(1)`, $c_0 =$ `c(1)` and

$$k = \frac{\texttt{c(2)-c(1)}}{(\texttt{z(2)-z(1)})^2} \ .$$

- Exponential profile:

$$c(z) = c_0 e^{-k(z-z_0)} \ ,$$

$$\frac{dc}{dz} = -kc_0 e^{-k(z-z_0)} \ , \ \frac{d^2c}{dz^2} = k^2 c_0 e^{-k(z-z_0)} \ ;$$

the parameters are calculated as $z_0 =$ `z(1)`, $c_0 =$ `c(1)` and

$$k = \frac{1}{\texttt{z(2)-z(1)}} \ln\left[\frac{\texttt{c(1)}}{\texttt{c(2)}}\right] \ .$$

- n^2-linear profile:

$$c(z) = \frac{c_0}{\left[1 + k\left(z - z_0\right)\right]^{1/2}} \ ,$$

$$\frac{dc}{dz} = \frac{-kc_0}{2\left[1 + k\left(z - z_0\right)\right]^{3/2}} \ , \ \frac{d^2c}{dz^2} = \frac{3k^2 c_0}{4\left[1 + k\left(z - z_0\right)\right]^{5/2}} \ ;$$

the parameters are calculated as $z_0 =$ `z(1)`, $c_0 =$ `c(1)` and

$$k = \frac{1}{\texttt{z(2)-z(1)}} \left[\left(\frac{\texttt{c(1)}}{\texttt{c(2)}}\right)^2 - 1\right] \ .$$

- Inverse-square gradient profile:

$$c(z) = c_0 \left\{1 + \frac{k\left(z - z_0\right)}{\left[1 + k^2\left(z - z_0\right)^2\right]^{1/2}}\right\} \ ,$$

$$\frac{dc}{dz} = \frac{kc_0}{2\left[1 + k^2\left(z - z_0\right)^2\right]^{3/2}} \ , \ \frac{d^2c}{dz^2} = \frac{-3k^3 c_0\left(z - z_0\right)}{\left[1 + k^2\left(z - z_0\right)^2\right]^{5/2}} \ ;$$

the parameters are calculated as $z_0 =$ `z(1)`, $c_0 =$ `c(1)` and

$$k = \frac{1}{\texttt{z(2)-z(1)}} \sqrt{\frac{a}{1-a}} \ ,$$

with

$$a = \left[\frac{\texttt{c(1)}}{\texttt{c(2)}} - 1\right]^2 \ .$$

- Munk profile:

$$c(z) = c_0 \left[1 + \varepsilon(\eta + e^{-\eta} - 1)\right] \ ,$$

$$\frac{dc}{dz} = \frac{2\varepsilon c_1}{B}(1 - e^{-\eta}) \ , \ \frac{d^2c}{dz^2} = \frac{4\varepsilon c_1}{B^2} e^{-\eta} \ ,$$

with the parameters $\varepsilon = 7,4 \times 10^{-3}$, $\eta = 2\,(z - z_0)\,/B$, B = 1,3 km, (z_0 represents the depth of the channel axis and c_0 the corresponding value of sound speed); the parameters are calculated as z_0 = `z(1)` and c_0 = `c(1)`. The remaining values are ignored.

The combination `cdist = 'c(z,z)'` and `cclass = 'TABL'` requires the sound profile to be specified as follows:

```
z0(1),c0(1)
z0(2),c0(2)
...
z0(nz0),c0(nz0)
```

The specification `cdist = 'c(r,z)'` (a sound speed field) should be followed by `cclass = 'TABL'`, otherwise the program stops execution; the sound speed field should be specified as follows:

```
r0(1),r0(2),r0(3),...,r0(nr0)
z0(1),z0(2),z0(3),...,z0(nz0)
 c(1,1) c(1,2) ...  c(1,nr0)
 c(2,1) c(2,2) ...  c(2,nr0)
 c(3,1) c(3,2) ...  c(3,nr0)
 ...
 c(nz0,1) c(nz0,2) ...  c(nz0,nr0)
```

When specifying the sound speed profile or field it is highly recommended to use an evenly spaced grid, avoiding vertical segments where a smooth variation is followed by an isovelocity layer. Including such segments introduce unrealistic artifacts, which result from the calculation of inaccurate sound speed gradients.

After the `Sound Speed Block` it should follow the single line

```
nobj
```

which indicates the number of objects defined inside the waveguide. If `nobj` = 0 the `Objects Block` is empty; otherwise (with `nobj` > 0) the INFIL should contain the following line

```
oitype
```

which describes the method of interpolation (`'2P'`, `'3P'` or `'4P'`, no other types are allowed), to be applied to the boundaries of *all* objects. For each object it should be defined an `Object Block` with the following structure:

`otype`	object type
`obju`	attenuation units
`no`	number of coordinates
`ocp(i), ocs(i), orho(i) ...`	Object c_p, c_s, etc.
`ro(1),zdn(1),zup(1)`	
`ro(2),zdn(2),zup(2)`	
...	
`ro(no),zdn(no),zup(no)`	

`otype` and `obju` are the same as those indicated for the `Altimetry Block`. The list above shows that the upper and lower boundaries of the object are sampled along a common range interval.

`Bathymetry Block`: the structure of this block is identical to the structure of the `Altimetry Block`.

`Array Block`:

`artype`	`array type`
`nra nza`	`number of hydrophones along range and depth`
`r(1) r(2) .... r(nra)`	`hydrophones ranges`
`z(1) z(2) .... z(nza)`	`hydrophones depths`

The option `artype` can correspond to one of the following strings:

`'RRY'`	Rectangular	aRray
`'HRY'`	Horizontal	arRay
`'VRY'`	Vertical	arRay
`'LRY'`	Linear	arRay

The option `artype` = `'LRY'` requires that `nra` = `nza`.

`Output Block`:

`outype`	`output type`
`miss`	`eigenray parameter`

The option `outype` defines the type of output and can correspond to one of the following strings:

'RCO'	output Ray COordinates;
'ARI'	output All Ray information;
'ERF'	output Eigenrays (use Regula Falsi);
'EPR'	output Eigenrays (use PRoximity method);
'ADR'	output Amplitudes and Delays (use Regula falsi);
'ADP'	output Amplitudes and Delays (use Proximity method);
'CPR'	output Coherent acoustic PRessure;
'CTL'	output Coherent Transmission Loss;
'PVL'	output coherent Particle VeLocity;
'PAV'	output Coherent acoustic Pressure And Particle velocity.

The `miss` parameter is used as a threshold to find eigenrays and to calculate arrivals.

Chapter 7

Examples and comparisons

This chapter is divided in two sections. The first section showcases **TRACEO** capabilities through the presentation of different examples; the second section adresses the model's accuracy when compared to other models. All the M-files mentioned in this chapter are distributed together with the model's code.

7.1 Examples

TRACEO examples are organized as follows:

- Deep water ray traces with variable boundaries and no objects:
  - Munk profile;
  - idealized Munk field;
- Shallow water examples for a Pekeris waveguide:
  - calculation of rays, eigenrays and travel times (no objects).
  - transmission loss and particle velocity calculations (two objects).

7.1.1 Deep water

A classical ray tracing test consists in the calculations of ray coordinates for a Munk profile and flat boundaries, with a source at 1000 m depth and a propagating range of 100 km (see [11]). In order to show **TRACEO**'s ability to deal simultaneously with refraction and reflection over range the test was extended by including variable boundaries: on top an idealized sinusoidal

surface (a feature which can be of interest for the study of scattering problems), while on bottom the variable bathymetry was given by a Gaussian sea mountain. In a second test the model's stability is further demonstrated, by replacing the Munk profile with an idealization of a Munk field. Both ray traces require the calculation of ray coordinates; in every case **TRACEO** produces a mat file, called `'rco.mat'`, which contains a vector of launching angles, and a set of matrices called `'ray1'`, `'ray2'`,..., one per each launching angle. Ray information is stored in each matrix as follows:

Row 1:	ray range r;
Row 2:	ray range z.

The Munk profile used in the first test can be seen in Fig.7.1. The ray plot shown in Fig.7.2 is produced by running the command

```
>> varbounds_profile.m
```

inside Matlab's prompt. As shown by the figure ray boundary reflections and refraction are properly handled by the model.

The second example idealizes a waveguide with the same boundaries as the first example, but considers a sequence of Munk profiles, with the axis channel depth deepening from 1000 m to 2000 m, when going from 0 to 100 km (see Fig.7.3). Such field reproduces approximately, on a small scale, the variation of sound speed when moving from the poles to the equator. The resulting ray trace, shown in Fig.7.4, is produced by running the command

```
>> varbounds_field.m
```

The figure confirms the expected channeling of ray energy along range, from shallow to deep waters, which is induced by the deepening of the deep sound channel. As in the first example, **TRACEO** keeps a proper handling of boundary reflections.

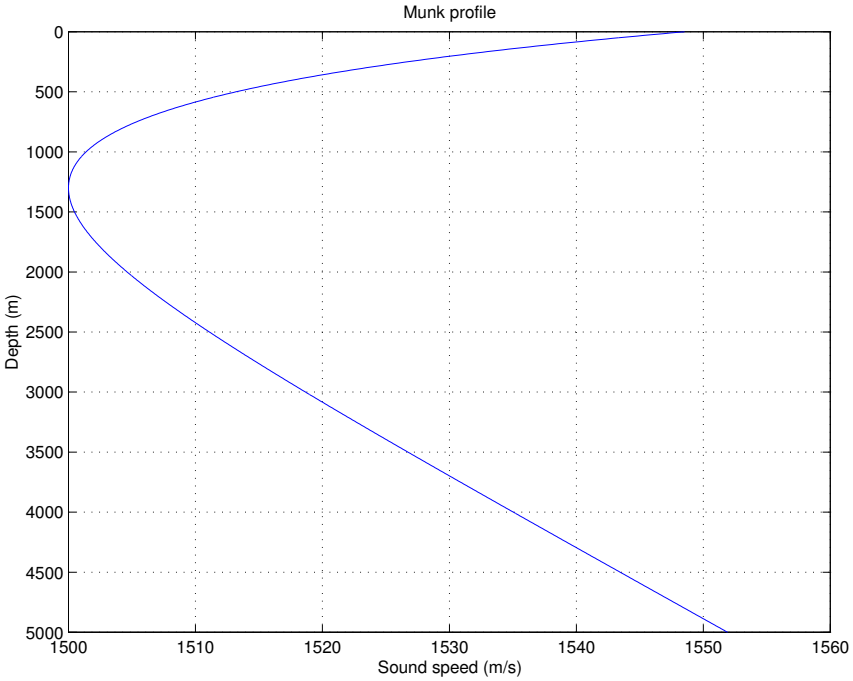


Figure 7.1: Canonical Munk profile.

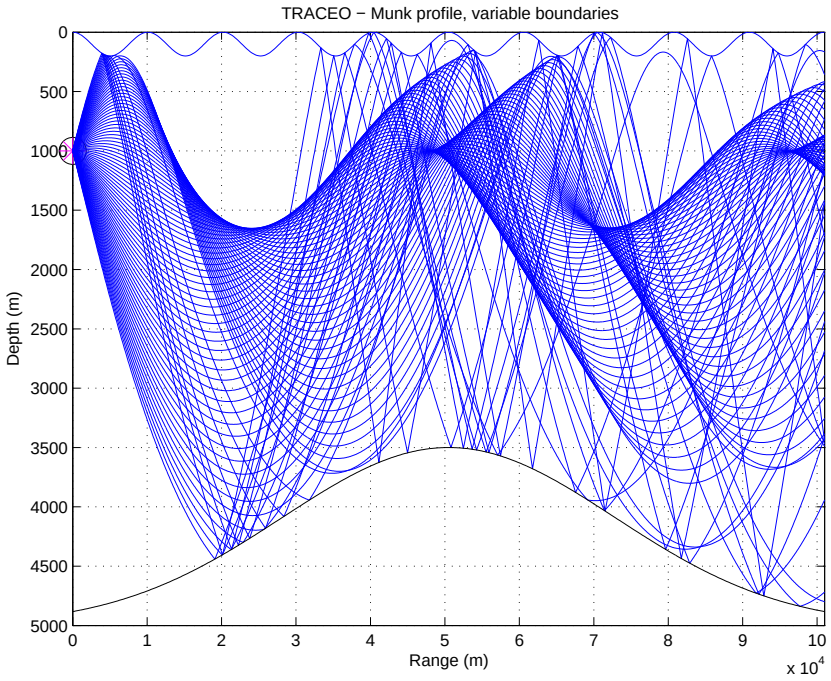


Figure 7.2: **TRACEO** ray trace.

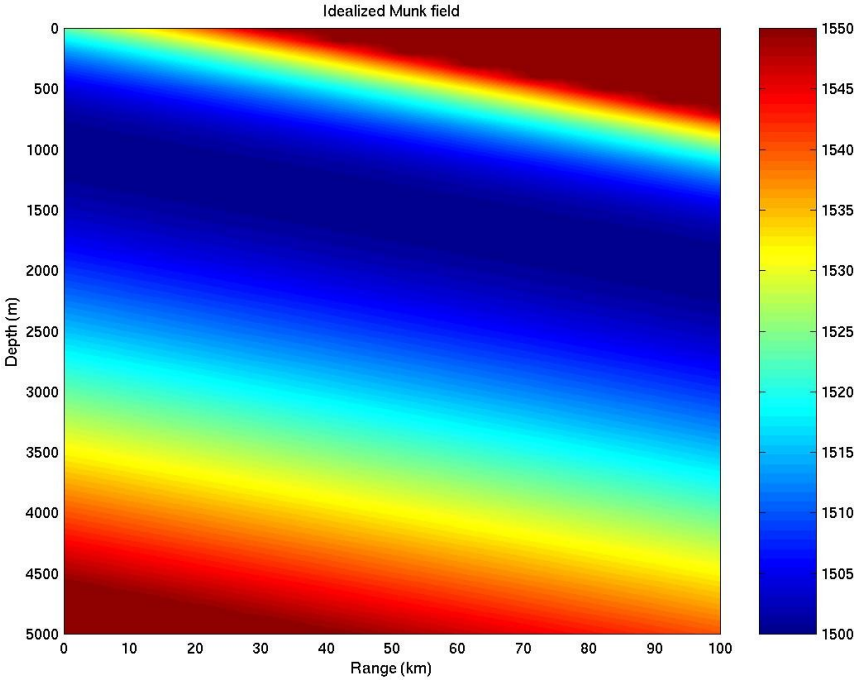


Figure 7.3: Idealized Munk field.

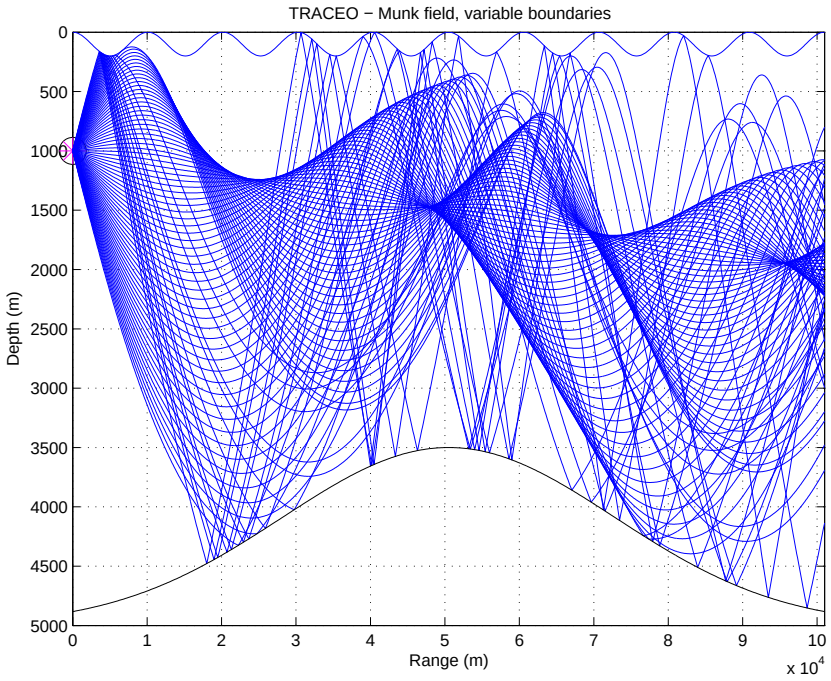


Figure 7.4: **TRACEO** ray trace

7.1.2 Shallow water

The shallow water case corresponds to a Pekeris waveguide with flat boundaries, as shown in Fig.7.5. Running the command

```
>> pekeris_rco.m
```

produces Fig.7.6, which at a first glance does not look particularly interesting; however, the ray pattern changes drastically if the boundary types are changed from `'V'` (for the surface) and `'E'` (for the bottom) to `'A'`; in such case, running the previous command produces now the ray trace shown in Fig.7.7, which predicts the lack of wave interference, previously induced by ray reflections. Additionally, other patterns will be generated by changing to `'A'` only one of the two boundary types.

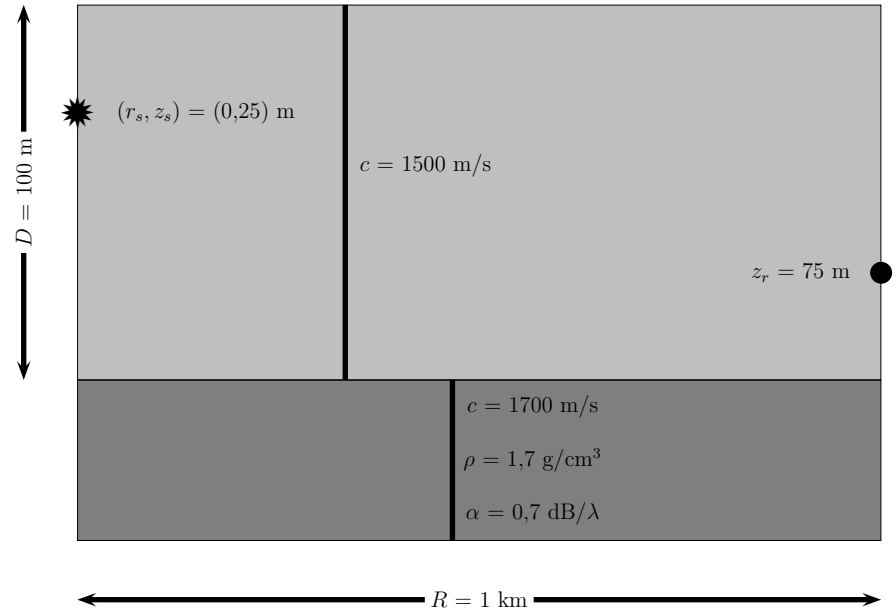


Figure 7.5: The flat Pekeris waveguide (vacuum on top).

If the output option `'RCO'` is changed to `'ARI'` running the command

```
>> pekeris_rco.m
```

produces now a mat file, called `'ari.mat'`; again, ray information is stored in matrices `'ray1'`, `'ray2'`,..., but now each matrix contains the following information:

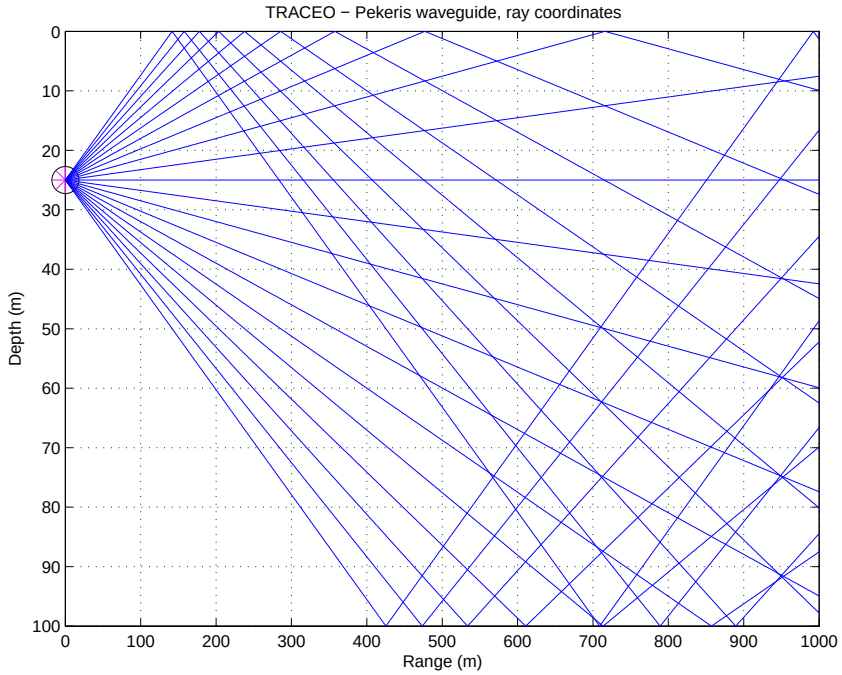


Figure 7.6: Pekeris waveguide: ray coordinates (top vacuum, bottom elastic).

Row 1:	ray range r;
Row 2:	ray depth z;
Row 3:	ray travel time τ;
Row 4:	real part of ray amplitude $\mathrm{Re}(a)$;
Row 5:	imaginary part of ray amplitude $\mathrm{Im}(a)$.

Eigenray calculations shown in Fig.7.8) are produced by running the command

```
>> pekeris_eig.m
```

with the output options `'ERF'` and `'EPR'`. Eigenray calculations produce a `'eig.mat'` mat file, with information stored as in the `'ari.mat'` output file. The second case uses five times more launching angles than the first, and still so the number of eigenrays is smaller. At a first glance eigenray search by proximity seems inefficient.

The advantage of eigenray search by proximity over search by regula falsi is revealed by running the command

```
>> pekeris_wedge_eig.m
```

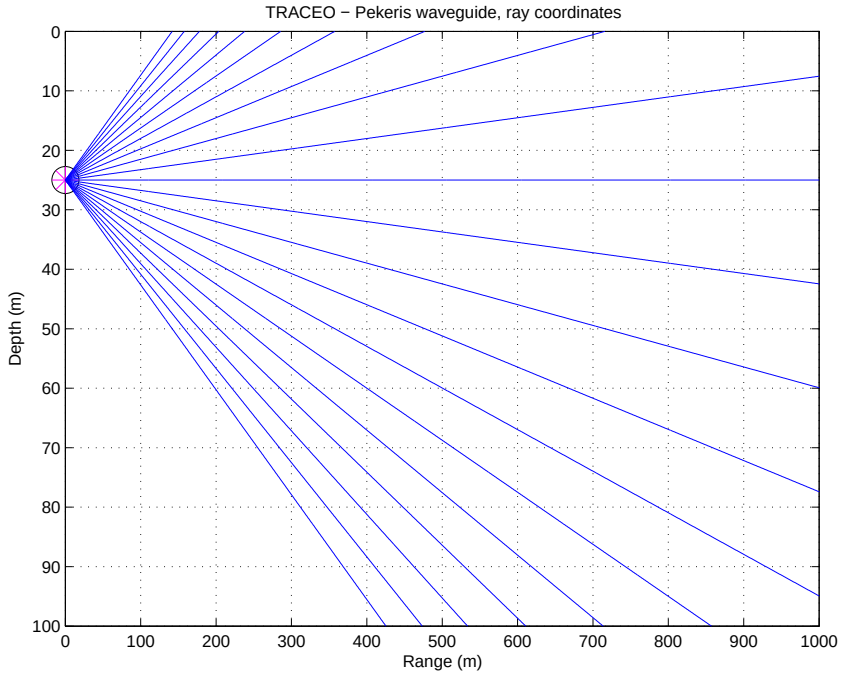


Figure 7.7: Pekeris waveguide: ray coordinates (top and bottom absorbers).

which produces Fig.7.9; in fact, eigenray search in this case is only possible with the `'EPR'` option.

A demonstration of **TRACEO**'s arrival calculations is shown in Fig.7.8), which is produced by running the command

```
>> pekeris_aad.m
```

with the output option `'ADR'`. Arrival calculations produce a mat file, called `'aad.mat'`; arrival information is stored in vectors `'aad1'`, `'aad2'`,..., one per each eigenray; each vector contains the following information:

Element 1:	hydrophone range r_h;
Element 2:	hydrophone depth z_h;
Element 3:	eigenray travel time τ;
Element 4:	real part of eigenray amplitude Re(a);
Element 5:	imaginary part of eigenray amplitude Im(a).

Arrival predictions shown in Fig.7.10 indicate a system of arrivals, clustered in groups of four arrivals; as expected from the symmetry between the source and the receiver central arrivals overlap, transforming the quadruplet groups in groups of triplets.

Transmission loss calculations with two ellipsoidal objects, at the positions $[25, 75]$ m and $[75, 25]$ m, are shown in Fig.7.11; the figure is produced by running the command

```
>> pekeris_pav2o.m
```

with the output option `'PAV'`. Acoustic pressure and particle velocity calculations produce a mat file, called `'pav.mat'`; for a rectangular array the information is stored in matrices `'rp'`, `'ip'`, `'ru'`, `'iu'` and `'rw'`, `'iw'`, which contain the real and imaginary parts of p, u and w, respectively. For a linear array the information is stored in vectors `'p'`, `'u'` and `'w'`, each containing the following information:

Row 1:	real part;
Row 2:	imaginary part.

In both cases the coordinates of the array are stored, together with the requested quantities. Fig.7.11 reveals a partial blocking of the acoustic field in the waveguide, with clear differences in the interference patterns of p, u and w. Although no clear diffraction is visible around the objects all cases indicate that the presence of the objects induces a significant redistribution of wave energy.

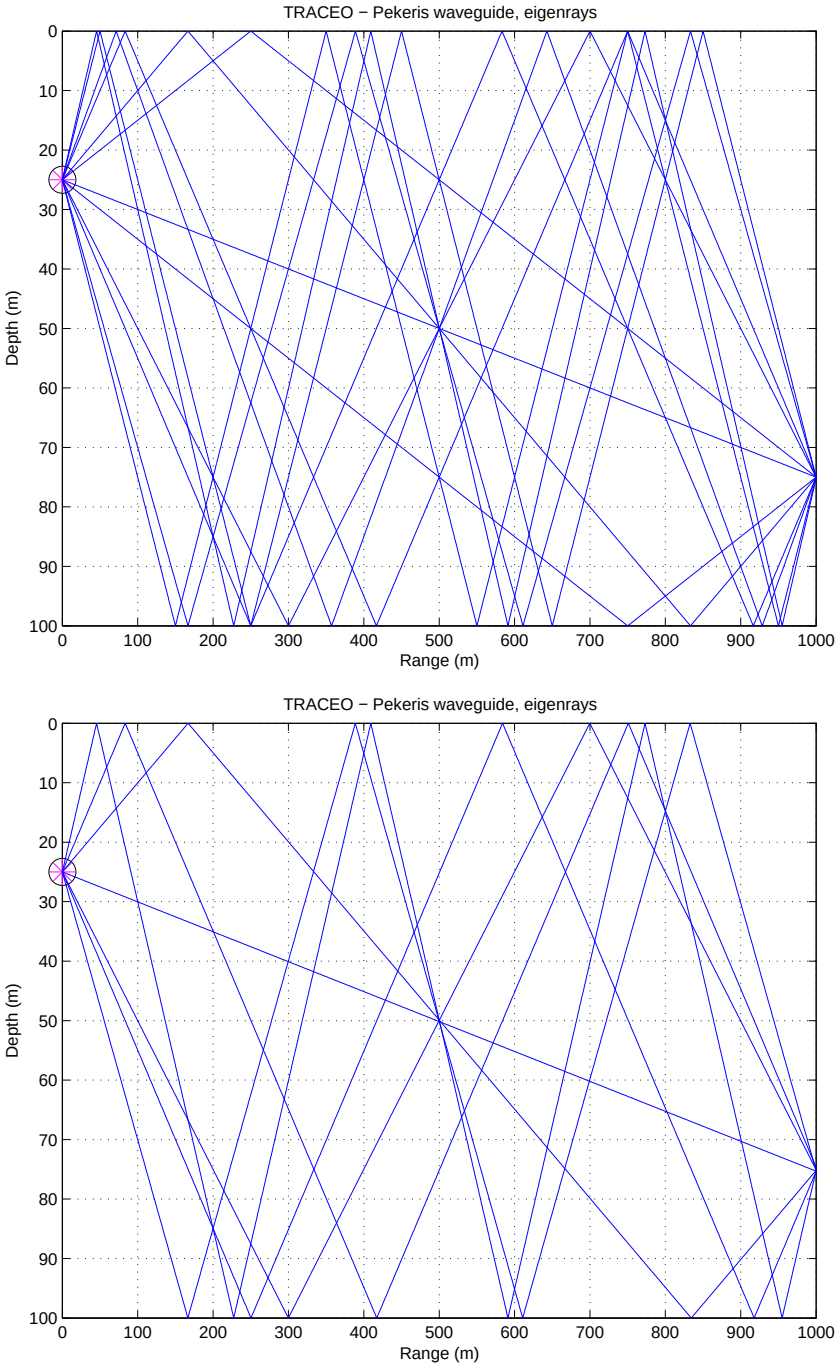


Figure 7.8: Pekeris waveguide: eigenrays calculated by Regula Falsi (top) and by proximity (bottom).

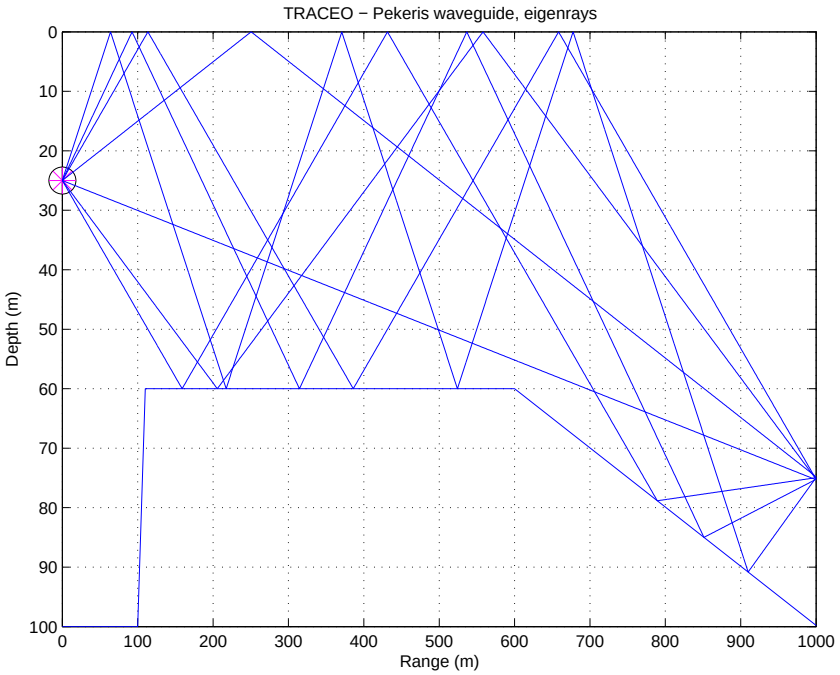


Figure 7.9: Pekeris waveguide with a wedge: eigenrays calculated by proximity.

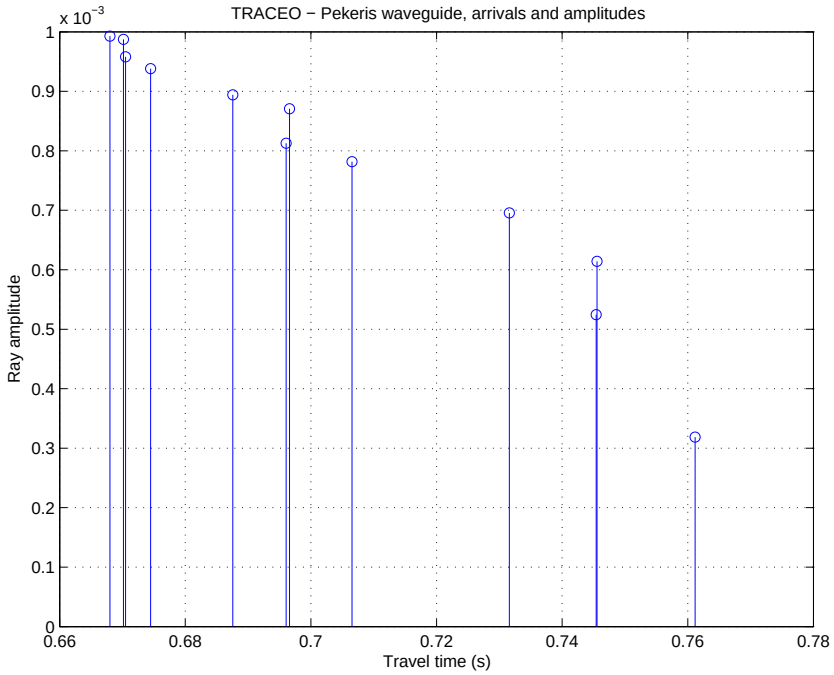


Figure 7.10: Pekeris waveguide: travel times and amplitudes calculated by Regula Falsi.

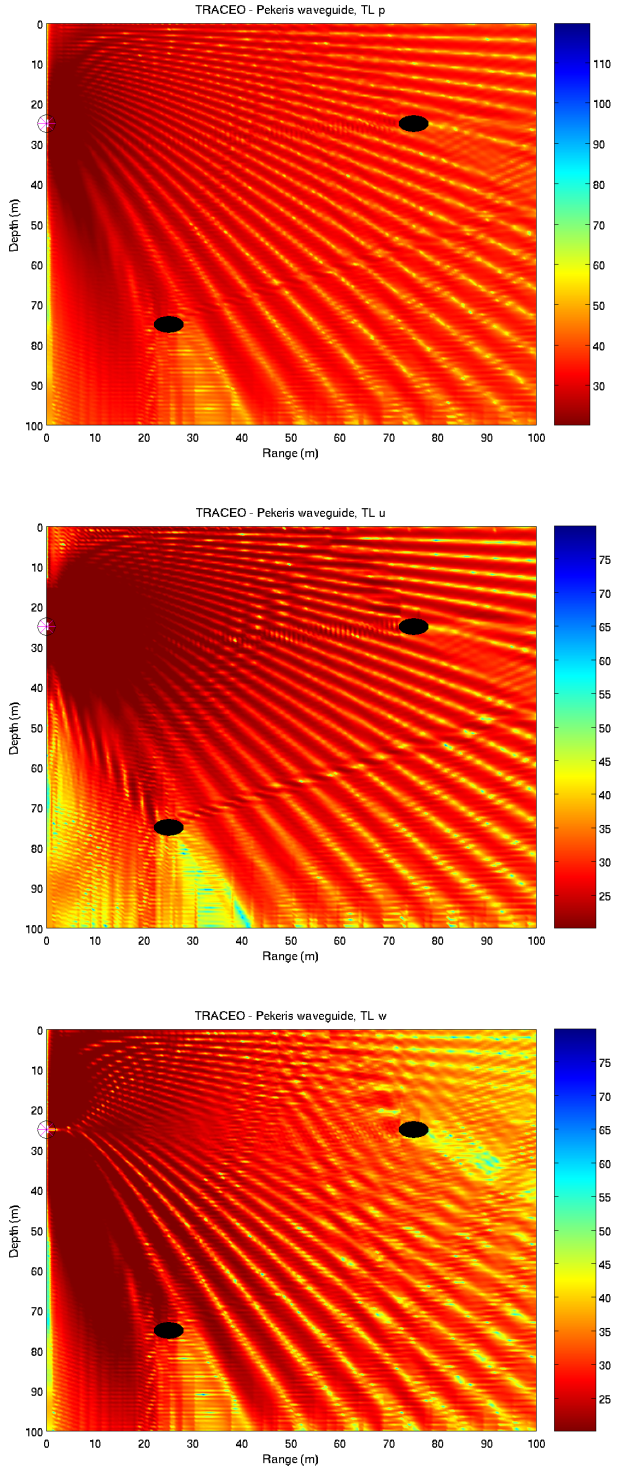


Figure 7.11: Pekeris waveguide: transmission loss for pressure (top), horizontal component of particle velocity (middle), and vertical component of particle velocity (bottom).

7.2 Model accuracy

This section discusses **TRACEO** accuracy by comparing the model results, with the predictions calculated by other models. The following cases are considered:

- Comparison with KRAKEN for a Munk profile.
- Comparison with UAN models.

7.2.1 Comparison with KRAKEN

The discussion presented in [11, 12] and [1] share in common a calculation of transmission loss for canonical Munk profile shown in Fig.7.1. Source frequency is 50 Hz, and the source is located at 1000 m. 51 rays were traced with **TRACEO** between -14° and 14°, restricting the ray fan exclusively to waterborne rays (see Fig.7.12). The ray plot indicates the existence of three large shadow zones, two of them (10-50 km and 70-100 km) in the upper part of the waveguide and the other one (40-80 km) in the lower part of it.

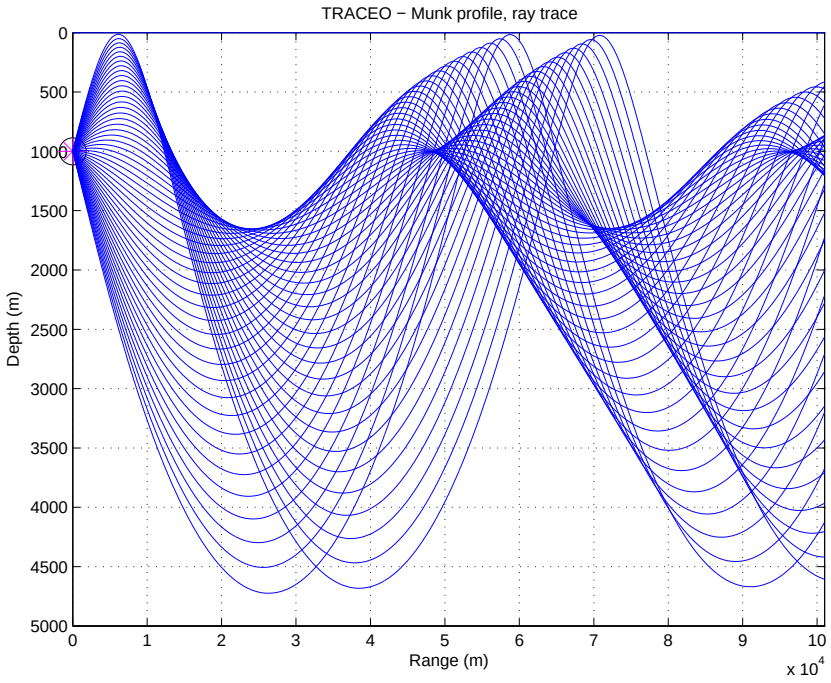


Figure 7.12: **TRACEO** ray trace.

Those rays were used to calculate a curve a transmission loss, for a receiver at 800 m depth. The comparison of **TRACEO** with Bellhop and KRAKEN

(see Fig.7.13) reveals a good agreement in the general trend of transmission loss calculated by both models. The differences in amplitude are negligible, except in the shadow zones, where rays contribute poorly to the field.

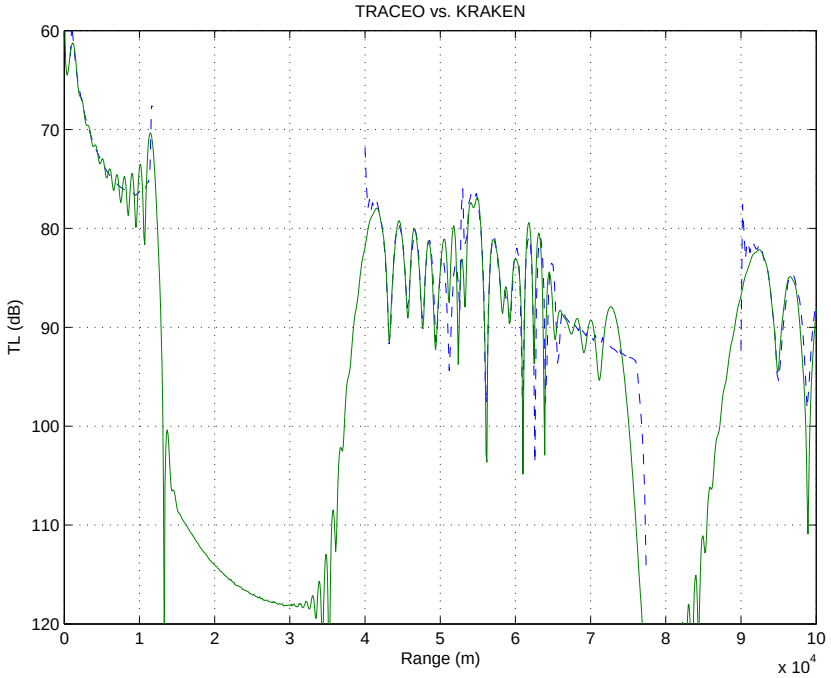


Figure 7.13: Transmission loss comparison: KRAKEN, solid line and **TRACEO**, dash-point line.

7.2.2 Comparison with UAN models

The last comparison can be seen in Fig.7.14, which shows the transmission loss calculated at 25,6 kHz by **TRACEO** and the acoustic models JEPE, REV3D and XRAY, used by the partners of the UAN project. Although there are no two identical matches all the models exhibit the same trend up until the range of 7 km. The ability of **TRACEO** to provide similar transmission loss levels to the ones provided by the different models constitutes an important validation of the model's accuracy.

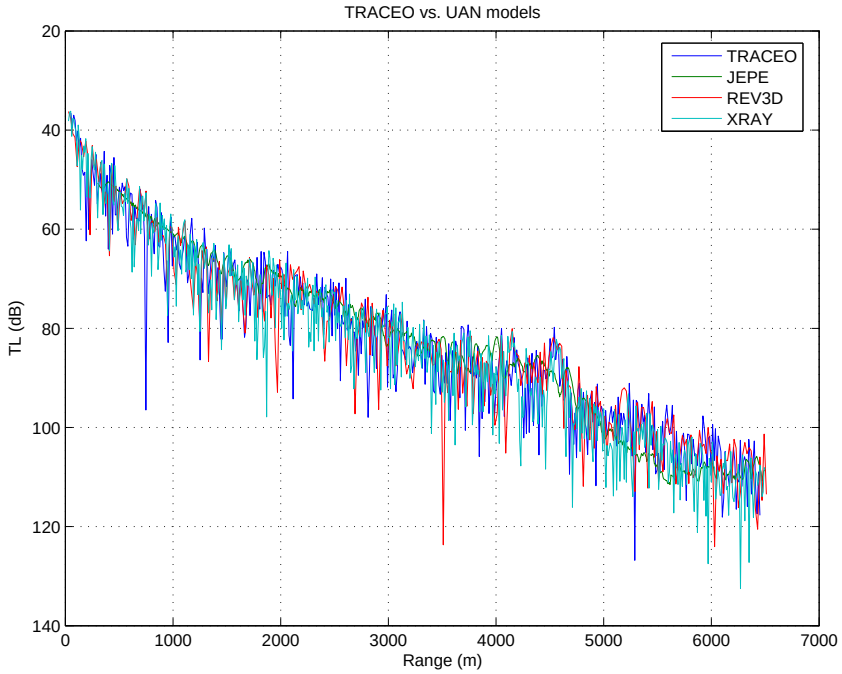


Figure 7.14: Transmission loss comparison: **TRACEO** vs. JEPE, REV3D and XRAY.

Chapter 8

Conclusions and future work

Hopefully, members of the acustic community will find **TRACEO** a useful and reliable tool of research. The model was written with modularity, stability and accuracy in mind and the examples and comparisons presented in this document are expected to speak by themselves regarding the potential of the model. There remain, however, several degrees of freedom worth of further development, namely:

- Code optimization; analysis of the subroutines will reveal repeating patterns of code, which some programmers will consider annoying. But such patterns exist in order to make the code as understanble as possible, and anybody capable of improving the code is invited to do so and to share such knowledge with the community.

- Translation to other programming languages; generally speaking Fortran statements are efficiently converted into machine code, but there are limitations of the Fortran 77 standard (like the need for initial memory allocation and no access to pointers in memory), which are higly desirably for optimization and/or for the development of interfaces[1].

- Parallelization (and cloud computing); ray calculation is an ideal task for distributed calculations, either way by using multi-processor machines or by distributing the calculations in a computer cloud. It is an interesting option for cases, which require a huge amount of repetitive calculations.

- Extension to the three-dimensional case; among the modelling tools used in underwater acoustics ray tracing has no match in terms of computational efficiency; thus, three-dimensional modeling with ray

[1] A C version is presently under way.

models represents an important alternative. **TRACEO** was written in order to allow easily such extension, but some elements of the code will require an ellaborate rewriting, in particular, to allow the inclusion of three-dimensional objects, to update the dynamic equations after boundary reflections and to calculate the normals and the ray influence at a given position of the array.

Appendix A

Additional topics

Besides accounting only for sound speed and being limited to the Lagrangian formalism there are additional approaches, with alternatives forms of the Eikonal equations, inclusion of ocean currents and discussion of ray tracing in terms of the Hamiltonian formalism. A brief discussion is presented here as an extension to the topics already presented.

A.1 Vector form of Eikonal equations

Alternatively with the compact form

$$\frac{d\mathbf{r}}{ds} = c\,\boldsymbol{\sigma} \quad , \quad \frac{d\boldsymbol{\sigma}}{ds} = \frac{1}{c^2}\boldsymbol{\nabla} c \quad , \tag{A.1}$$

the change of ds by $d\tau$ allows to obtain the following Eikonal equations[13]

$$\frac{d\mathbf{r}}{d\tau} = c^2\boldsymbol{\sigma} \quad , \quad \frac{d\boldsymbol{\sigma}}{d\tau} = \frac{1}{c}\boldsymbol{\nabla} c \quad . \tag{A.2}$$

A.2 Eikonal equations including wavenumber

The vector form of Eikonal equations can be rewritten in order to include the wavenumber as[14]:

$$\frac{d}{ds}\left(k\frac{d\mathbf{r}}{ds}\right) = \boldsymbol{\nabla} k \ . \tag{A.3}$$

The previous equation can be written as the system:

$$\frac{d\mathbf{r}}{ds} = \mathbf{e}_s \quad \text{and} \quad \frac{d\mathbf{e}_s}{ds} = \frac{1}{k}\boldsymbol{\nabla} k - \frac{1}{k}\frac{dk}{ds}\mathbf{e}_s \ . \tag{A.4}$$

A.3 Wavenumber 2D Eikonal equations

The system given by Eq.(A.4) can be written in cylindrical coordinates as[14]:

$$\frac{d}{dr}\cos\theta = \frac{1}{k}\frac{\partial k}{\partial r}\frac{\sin^2\theta}{\cos\theta} - \frac{1}{k}\frac{\partial k}{\partial z}\sin\theta \ , \tag{A.5}$$

$$\frac{d}{dr}\sin\theta = \frac{1}{k}\frac{\partial k}{\partial z}\cos\theta - \frac{1}{k}\frac{\partial k}{\partial r}\sin\theta \ . \tag{A.6}$$

If integration over z is required the system becomes:

$$\frac{d}{dz}\cos\theta = \frac{1}{k}\frac{\partial k}{\partial r}\sin\theta - \frac{1}{k}\frac{\partial k}{\partial z}\cos\theta \ , \tag{A.7}$$

$$\frac{d}{dz}\sin\theta = \frac{1}{k}\frac{\partial k}{\partial z}\frac{\cos^2\theta}{\sin\theta} - \frac{1}{k}\frac{\partial k}{\partial r}\cos\theta \ . \tag{A.8}$$

Independently of integrating over r or z it holds valid in both cases that

$$\frac{dz}{dr} = \frac{\sin\theta}{\cos\theta} \ .$$

A.4 Ray slope 2D Eikonal equations

The 2D Eikonal equations with the substitution of slowness by θ can be written as follows[15]:

$$\frac{d\theta}{dr} = \frac{1}{c}\frac{\partial c}{\partial r}\tan\theta - \frac{1}{c}\frac{\partial c}{\partial z} \ , \tag{A.9}$$

$$\frac{dz}{dr} = \tan\theta \ , \tag{A.10}$$

$$\frac{d\tau}{dr} = \frac{\sec\theta}{c} \ . \tag{A.11}$$

A.5 Earth 2D Eikonal equations

The equations from section A.4 can be easily extended in order to account for earth's curvature, and correspond to[15]:

$$\frac{d\theta}{d\tilde{r}} = f_e\frac{1}{c}\frac{\partial c}{\partial \tilde{z}} - \frac{1}{c}\frac{\partial c}{\partial \tilde{r}}\tan\theta - \frac{1}{R_e} \ , \tag{A.12}$$

$$\frac{d\tilde{z}}{d\tilde{r}} = f_e\tan\theta \quad , \tag{A.13}$$

$$\frac{d\tau}{d\tilde{r}} = f_e\frac{\sec\theta}{c} \ . \tag{A.14}$$

where $\tilde{z}$ lies along the radius of the earth, $\tilde{z} = 0$ at the surface, $\tilde{z} = R_e$ at the earth's center, R_e represents the earth's radius, $\tilde{r}$ is the distance traveled along a circular arc at the sea level, and

$$f_e = \frac{R_e - \tilde{z}}{R_e} \ .$$

A.6 Including ocean currents

The Eikonal equations can be extended in order to handle ocean currents; the system can be written compactly in vector form as[16]

$$\frac{d\mathbf{r}}{d\tau} = \frac{c^2}{\Omega}\boldsymbol{\sigma} + \mathbf{v} \ , \tag{A.15}$$

$$\frac{d\boldsymbol{\sigma}}{d\tau} = -\frac{\Omega}{c}\boldsymbol{\nabla} c - \boldsymbol{\sigma} \times (\boldsymbol{\nabla} \times \mathbf{v}) - (\boldsymbol{\sigma} \cdot \boldsymbol{\nabla})\,\mathbf{v} \ , \tag{A.16}$$

where $\mathbf{v}$ represents the vector of currents and

$$\Omega = \frac{c}{c\,(1 + \mathbf{v} \cdot \boldsymbol{\sigma})} \ . \tag{A.17}$$

A.7 Hamiltonian formalism

Within the context of the Hamiltonian formalism the travel time can be written as

$$\tau = \int_A^B (\sigma_x \dot{x} + \sigma_y \dot{y} + \sigma_z \dot{z} - H)\, ds \ , \tag{A.18}$$

where $\mathcal{H}$ represents the system's Hamiltonian:

$$\mathcal{H}(x, y, z, \sigma_x, \sigma_y, \sigma_z) = \sigma_x \dot{x} + \sigma_y \dot{y} + \sigma_z \dot{z} - \sigma \ , \tag{A.19}$$

or compactly, in vector notation:

$$\mathcal{H}(x, y, z, \sigma_x, \sigma_y, \sigma_z) = \boldsymbol{\sigma} \cdot \mathbf{e}_s - \sigma \ . \tag{A.20}$$

The perturbation in travel time becomes

$$\delta\tau = \int_A^B [\dot{x}\delta\sigma_x + \sigma_x \delta\dot{x} + \dot{y}\delta\sigma_y + \sigma_y \delta\dot{y} + \dot{z}\delta\sigma_z + \sigma_z \delta\dot{z} - (\ldots)]\, ds$$

where

$$(\ldots) = \frac{\partial \mathcal{H}}{\partial \sigma_x}\delta\sigma_x + \frac{\partial \mathcal{H}}{\partial x}\delta x + \frac{\partial \mathcal{H}}{\partial \sigma_y}\delta\sigma_y + \frac{\partial \mathcal{H}}{\partial y}\delta y + \frac{\partial \mathcal{H}}{\partial \sigma_z}\delta\sigma_z + \frac{\partial \mathcal{H}}{\partial z}\delta z \ .$$

It follows then that

$$\delta\tau = \int_A^B \left[(\ldots) + \sigma_x \delta\dot{x} + \sigma_y \delta\dot{y} + \sigma_z \delta\dot{z}\right] ds \ .$$

where

$$(\ldots) =$$

$$= \left(\dot{x} - \frac{\partial \mathcal{H}}{\partial \sigma_x}\right) \delta\sigma_x - \frac{\partial \mathcal{H}}{\partial x}\delta x + \left(\dot{y} - \frac{\partial \mathcal{H}}{\partial \sigma_y}\right) \delta\sigma_y - \frac{\partial \mathcal{H}}{\partial y}\delta y + \left(\dot{z} - \frac{\partial \mathcal{H}}{\partial \sigma_z}\right) \delta\sigma_z - \frac{\partial \mathcal{H}}{\partial z}\delta z \ .$$

Let us notice now that

$$\int_A^B \left(\sigma_x \delta\dot{x} + \sigma_y \delta\dot{y} + \sigma_z \delta\dot{z}\right) ds =$$

$$= \underbrace{\sigma_x \delta x + \sigma_y \delta y + \sigma_z \delta z|_A^B}_{=0} - \int_A^B \left(\dot{\sigma_x} \delta x + \dot{\sigma_y} \delta y + \dot{\sigma_z} \delta z\right) ds \ ,$$

so the perturbation in travel time becomes

$$\delta\tau = \int_A^B \left[\begin{array}{l} \left(\dot{x} - \dfrac{\partial \mathcal{H}}{\partial \sigma_x}\right) \delta\sigma_x - \left(\dot{\sigma}_x + \dfrac{\partial \mathcal{H}}{\partial x}\right) \delta x + \\ \left(\dot{y} - \dfrac{\partial \mathcal{H}}{\partial \sigma_y}\right) \delta\sigma_y - \left(\dot{\sigma}_y + \dfrac{\partial \mathcal{H}}{\partial y}\right) \delta y + \\ \left(\dot{z} - \dfrac{\partial \mathcal{H}}{\partial \sigma_z}\right) \delta\sigma_z - \left(\dot{\sigma}_z + \dfrac{\partial \mathcal{H}}{\partial z}\right) \delta z \end{array} \right] ds \ .$$

According to Fermat's principle $\delta\tau = 0$, which allows to infer the following system of equations

$$\begin{aligned} \frac{dx}{ds} &= \frac{\partial \mathcal{H}}{\partial \sigma_x} \ , \qquad \frac{d\sigma_x}{ds} = -\frac{\partial \mathcal{H}}{\partial x} \ , \\ \frac{dy}{ds} &= \frac{\partial \mathcal{H}}{\partial \sigma_y} \ , \qquad \frac{d\sigma_y}{ds} = -\frac{\partial \mathcal{H}}{\partial y} \ , \\ \frac{dz}{ds} &= \frac{\partial \mathcal{H}}{\partial \sigma_z} \ , \qquad \frac{d\sigma_z}{ds} = -\frac{\partial \mathcal{H}}{\partial z} \ . \end{aligned} \tag{A.21}$$

The Hamiltonian can equally be rewritten in order to proceed with an integration along travel time, and by substituting the components of sound

slowness with the components of the wavenumber. In fact, by taking into account that

$$ds = d\tau/\sigma$$

and that

$$k_x = \omega\sigma_x \ , \quad k_y = \omega\sigma_y \ , \quad k_z = \omega\sigma_z \ ,$$

one can obtain the Hamiltonian[17][1]:

$$\mathcal{H} = \omega^2 - c^2k^2 \tag{A.22}$$

(this expression corresponds to the Hamiltonian given by Eq.(A.19), multiplied by the factor $-\omega^2/\sigma$); the system of equations becomes

$$\frac{dx}{d\tau} = \frac{\partial\mathcal{H}}{\partial k_x} \ , \quad \frac{dk_x}{d\tau} = -\frac{\partial\mathcal{H}}{\partial x} \ ,$$

$$\frac{dy}{d\tau} = \frac{\partial\mathcal{H}}{\partial k_y} \ , \quad \frac{dk_y}{d\tau} = -\frac{\partial\mathcal{H}}{\partial y} \ ,$$

$$\frac{dz}{d\tau} = \frac{\partial\mathcal{H}}{\partial k_z} \ , \quad \frac{dk_z}{d\tau} = -\frac{\partial\mathcal{H}}{\partial z} \ .$$

As shown by the two-dimensional case with cylindrical symmetry follows automatically from this case as

$$\frac{dr}{d\tau} = \frac{\partial\mathcal{H}}{\partial k_r} \ , \quad \frac{dk_r}{d\tau} = -\frac{\partial\mathcal{H}}{\partial r} \ ,$$

$$\frac{dz}{d\tau} = \frac{\partial\mathcal{H}}{\partial k_z} \ , \quad \frac{dk_z}{d\tau} = -\frac{\partial\mathcal{H}}{\partial z} \ .$$

There are also alternativa approaches, which consider a Hamiltonian written in terms of s or r. For the first case and for two-dimensional case with cylindrical symmetry one can obtain the Hamiltonian

$$\mathcal{H} = \sigma_r\dot{r} + \sigma_z\dot{z} - \sigma \ , \tag{A.23}$$

related to the system of equations

$$\frac{dr}{ds} = \frac{\partial\mathcal{H}}{\partial\sigma_r} \ , \quad \frac{d\sigma_r}{ds} = -\frac{\partial\mathcal{H}}{\partial r} \ ,$$

[1]The complete expression for the Hamiltonian corresponds to

$$\mathcal{H} = (\omega - \mathbf{k}\cdot\mathbf{v})^2 - c^2k^2 \ ,$$

where $\mathbf{v}$ represents the vector of currents.

$$\frac{dz}{ds} = \frac{\partial \mathcal{H}}{\partial \sigma_z} \ , \qquad \frac{d\sigma_z}{ds} = -\frac{\partial \mathcal{H}}{\partial z} \ ;$$

as for the second case the Hamiltonian and associated system of equations correspond to[18]

$$\mathcal{H} = -\sqrt{\frac{1}{c^2} - \sigma_z^2} \ , \tag{A.24}$$

and

$$\frac{dz}{dr} = \frac{\partial \mathcal{H}}{\partial \sigma_z} \ , \qquad \frac{d\sigma_z}{dr} = -\frac{\partial \mathcal{H}}{\partial z} \ .$$

Bibliography

[1] Porter M.B. and Bucker H.P. Gaussian beam tracing for computing ocean acoustic fields. *J. Acoust. Soc. America*, 82(4):1349–1359, 1987.

[2] Porter M. The KRAKEN normal mode program. Technical report, SACLANT UNDERSEA RESEARCH (memorandum), San Bartolomeo, Italy, 1991.

[3] Jensen F., Kuperman W., Porter M., and Schmidt H. *Computational Ocean Acoustics*. AIP Series in Modern Acoustics and Signal Processing, New York, 1994.

[4] Buckingham M.J. Ocean acoustic propagation models. Technical Report EUR 13810, Comission of the European Communities, 1991.

[5] Popov M.M. Ray theory and Gaussian beam method for geophysicists. Technical Report EDUFBA, Universidade Federal da Bahia, 2002.

[6] Santos P., Rodríguez O.C., Felisberto P., and Jesus S.M. Geoacoustic inversion with a vector sensor array. *J. Acoust. Soc. Am*, 5(128):2652–2663, November 2010.

[7] Schmidt H. SAFARI, Seismo–Acoustic Fast field Algorithm for Range – Independent enviroments. User's Guide. Technical report, SACLANT UNDERSEA RESEARCH CENTRE (SM-113), La Spezia, Italy, 1987.

[8] Popov M.M. and Pšenčík. Computation of ray amplitudes in homogeneous media with curved interfaces. *Studia Geoph. et Geod.*, (22):248–258, 1978.

[9] Červený V., Popov M., and Pšenčík I. Computation of wave fields in inhomogeneous media - Gaussian beam approach. *Gephys. J. R. astr. Soc.*, 70:109–128, 1982.

[10] Papadakis P.J., Taroudakis M.I., and Papadakis J.S. Recovery of the properties of an elastic bottom using reflection coefficient measurements. In *Proceedings of the 2nd. European Conference on Underwater Acoustics*, volume II, pages 943–948, Copenhagen, Denmark, 1994.

[11] Porter M. B. and Liu Y-C. Finite-Element Ray Tracing. In *Theoretical and Computational Acoustics*, volume 2, World Scientific Publishing Co., 1994.

[12] Baxley P.A., Bucker H., and Porter M.B. Comparison of beam tracing algorithms. In *Proceedings of the 5th. European Conference on Underwater Acoustics*, Lyon, France, July 2000.

[13] Červený V. and Pšenčík I. Ray amplitudes of seismic body waves in laterally inhomogeneous media. *Gephys. J. R. astr. Soc.*, 57:91–106, 1979.

[14] Collins M.D. and Kuperman W.A. Focalization: Environmental focusing and source localization. *J. Acoust. Soc. America*, 90(3):1410–1422, September 1991.

[15] Bowlin J.B., Spiesberger J.L., Duda T.F., and Freitag L.F. Ocean Acoustical Ray-Tracing Software Ray. Technical report, Woods Hole Oceanographic Institution, October 1992.

[16] Dushaw B.D. and Colosi J.A. Ray Tracing for Ocean Acoustic Tomography. Technical Report APL-UW TM 3-98, Applied Physics Laboratory University of Washington, December 1998.

[17] Jones R.M., Riley J.P., and Georges T.M. HARPO – A versatile three-dimensional Hamiltonian ray-tracing program for acoustic waves in an ocean with irregular bottom. Technical report, NOAA, October 1986.

[18] Beron-Vera F.J., Brown M.G., Tomsovic S., Virovlyansky A.L., Wolfson M.A., and Zaslavsky G.M. Ray dynamics in a long-range acoustic propagation experiment. *J. Acoust. Soc. America*, 114(1):123–130, 2003.