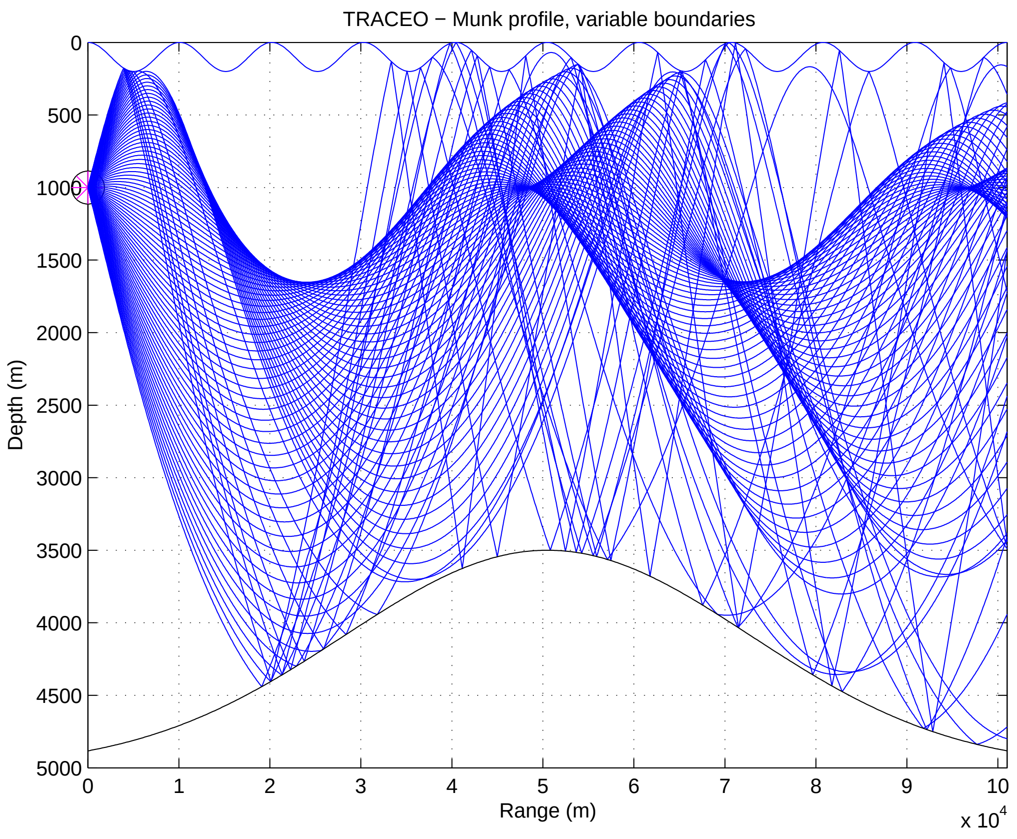
# The TRACEO ray tracing program (v1.0)



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

03/05/2015

# Contents

# Chapter 1

# Introduction

This document describes the latest release of **TRACEO**, a ray tracing model written in Fortran 77 and tested with the GNU `gfortran` compiler. The current release of the model updates the latest version of **TRACEO** by introducing the following modifications:

- Code improvements.
- Variable names are all strings with eighth characters.
- Matlab MAT files are now generated using own subroutines.
- The model can now be compiled on any OS for which `gfortran` is available.

A detailed theoretical description of ray tracing can be found in the original manual together with a detailed description of the model [1] . This document describes only the new conventions used for variable names; for accuracy issues check [1] and [2]. Users with a preference for coding in C can consider using **cTraceo**, a version of **TRACEO** written in C by Emanuel Ey [3].

# Chapter 2

# Model description

The Fortran sources of the model are provided with a simple `makefile`. The `gfortran` GNU compiler is required for installation; invoking `make` on the command line should allow to produce a binary called `traceo.exe`; the user should place this binary 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.

Besides the Matlab mat files **TRACEO** writes a short `munk.log` ASCII file, describing the type of output and the calculation time. The structure of the input file is described in detail in the manual.

# Chapter 3

# Examples

This chapter showcases **TRACEO** capabilities through the presentation of different examples. All the M-files mentioned in this chapter are distributed together with the model's code.

**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).

## 3.1 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 [4]). 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 `'ray00001'`, `'ray00002'`,..., 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.3.1. The ray plot shown in Fig.3.2 is produced by running the command

```
>> rco_varbounds
```

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.3.3). 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.3.4, is produced by running the command

```
>> rco_varbounds_field
```

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.

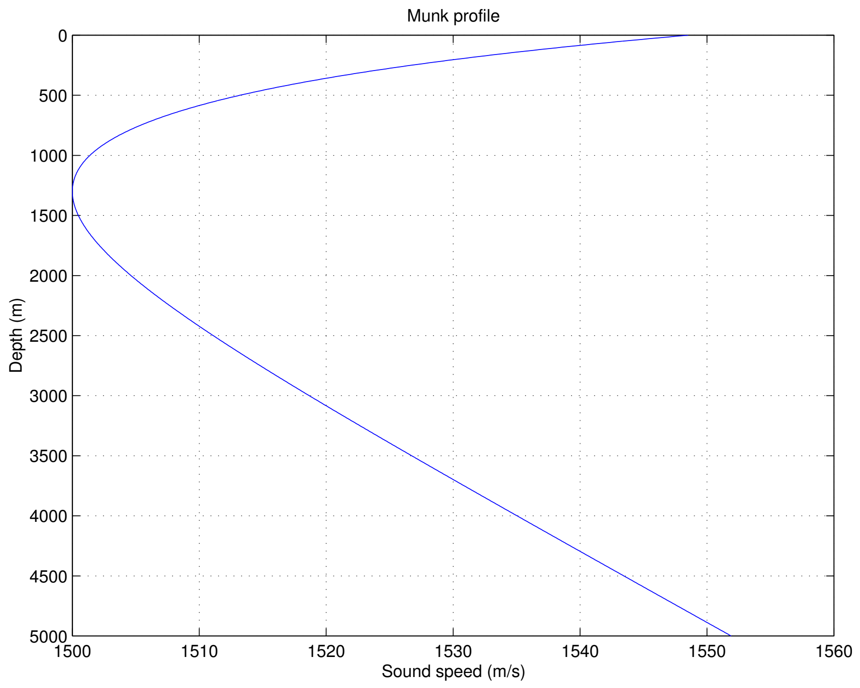


Figure 3.1: Canonical Munk profile.

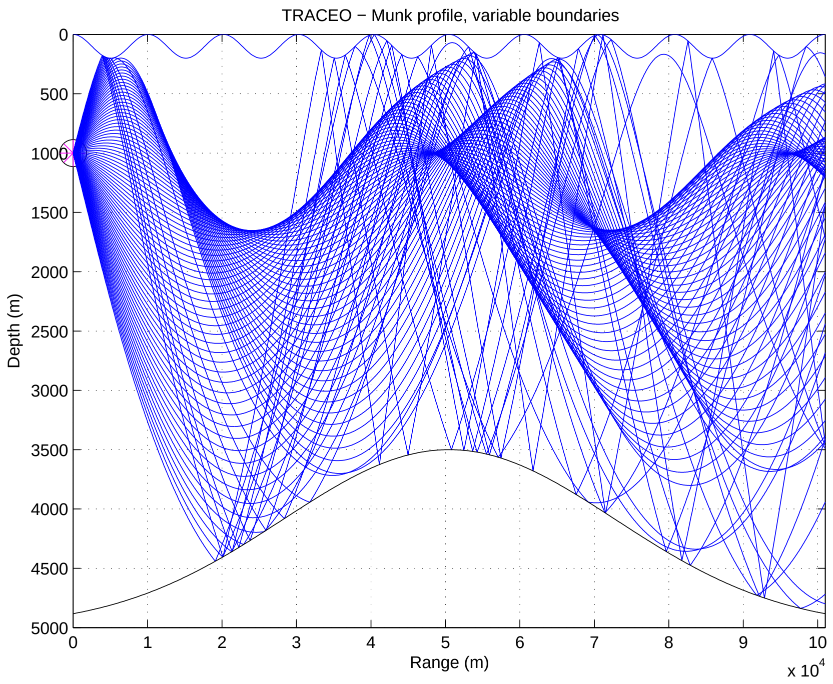


Figure 3.2: **TRACEO** ray trace.

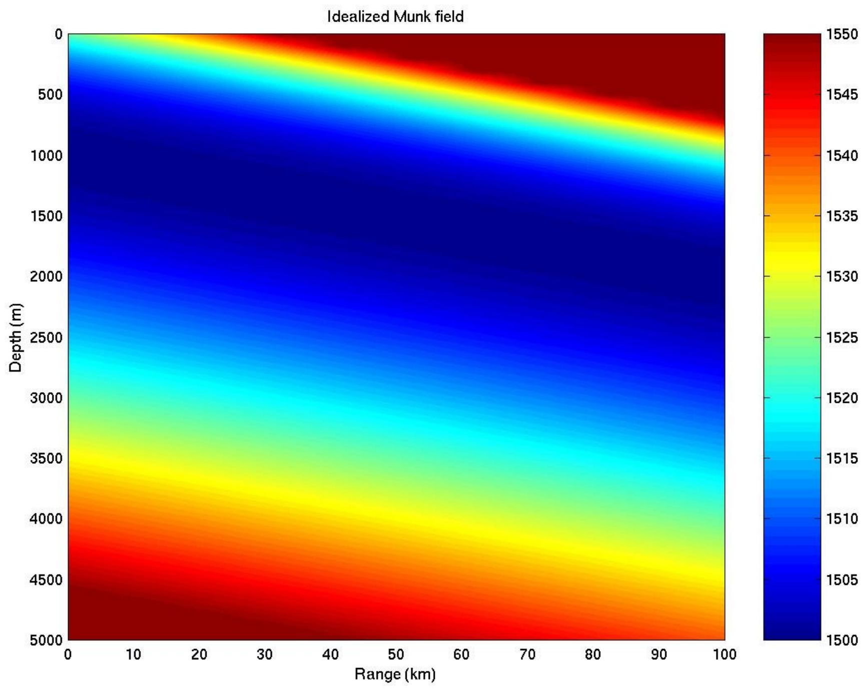


Figure 3.3: Idealized Munk field.

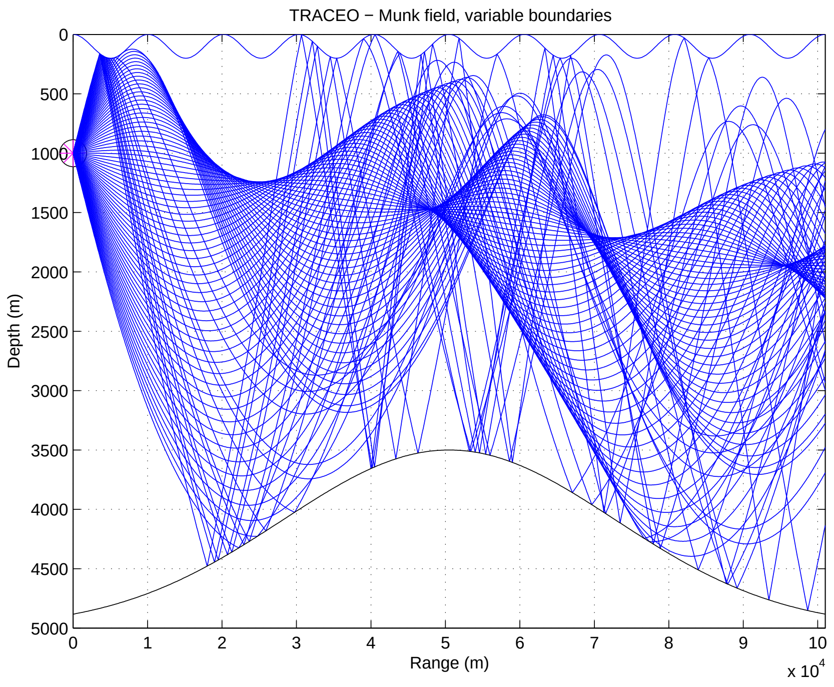


Figure 3.4: **TRACEO** ray trace

## 3.2 Shallow water

The shallow water case corresponds to a Pekeris waveguide with flat boundaries, as shown in Fig.3.5. Running the command

```
>> rco_flat
```

produces Fig.3.6, 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.3.7, 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.

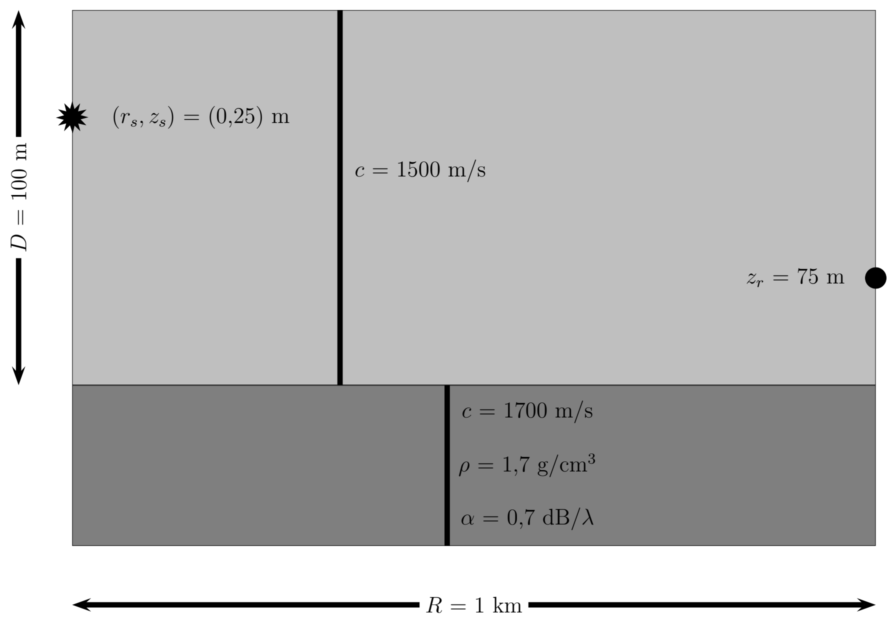


Figure 3.5: The flat Pekeris waveguide (vacuum on top).

The change of the output option `'RCO'` to `'ARI'` can be seen by running the M-file

```
>> ari_flat
```

which produces a mat file, called `'ari.mat'`; again, ray information is stored in matrices `'ray00001'`, `'ray00002'`,..., but now each matrix contains the following information:

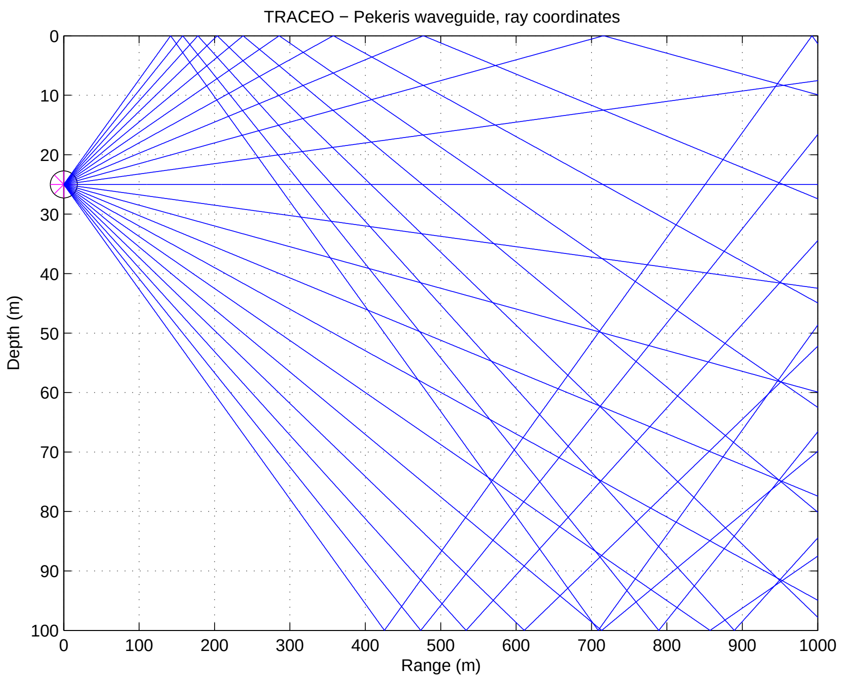


Figure 3.6: Pekeris waveguide: ray coordinates (top vacuum, bottom elastic).

| | |
|---|---|
| Row 1: | ray range $r$; |
| Row 2: | ray depth $z$; |
| Row 3: | ray travel time $\tau$; |
| Row 4: | real part of ray amplitude $\mathrm{Re}(a)$; |
| Row 5: | imaginary part of ray amplitude $\mathrm{Im}(a)$; |
| Row 6: | phase for caustic correction. |

Eigenray calculations shown in Fig.3.8) are produced by running the M-file

```
>> eig_flat
```

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

```
>> eig_wedge
```

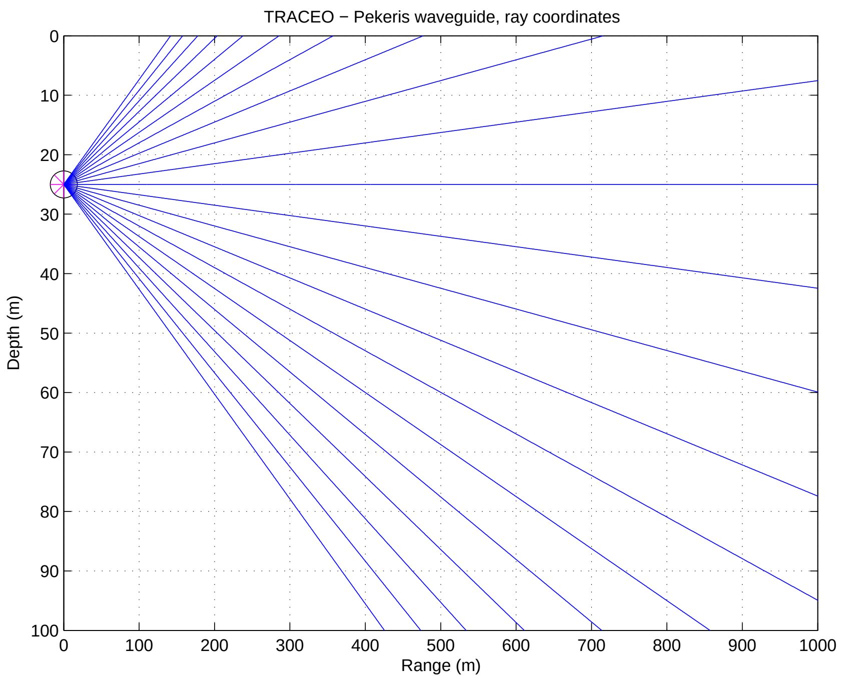


Figure 3.7: Pekeris waveguide: ray coordinates (top and bottom absorbers).

which produces Fig.3.9; 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.3.8), which is produced by running the command

```
>> adp_flat
```

with the output option `'ADP'`. Arrival calculations produce a mat file, called `'aad.mat'`; arrival information is stored in vectors `'aad00001'`, `'aad00002'`,..., one per eigenray; each vector contains the following information:

| | |
|---|---|
| Element 1: | hydrophone range $r_h$; |
| Element 2: | hydrophone depth $z_h$; |
| Element 3: | eigenray travel time $\tau$; |
| Element 4: | real part of eigenray amplitude $\mathrm{Re}(a)$; |
| Element 5: | imaginary part of eigenray amplitude $\mathrm{Im}(a)$; |
| Element 6: | phase for caustic correction. |

Arrival predictions shown in Fig.3.10 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.3.11; the figure is produced by running the command

```
>> pav_2o
```

with the output option `'PAV'`. Acoustic pressure and particle velocity calculations produce a mat file, called `'pav.mat'` with the complex matrices/vectors for pressure and particle components; the coordinates of the array are stored together with the requested quantities. Fig.3.11 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.

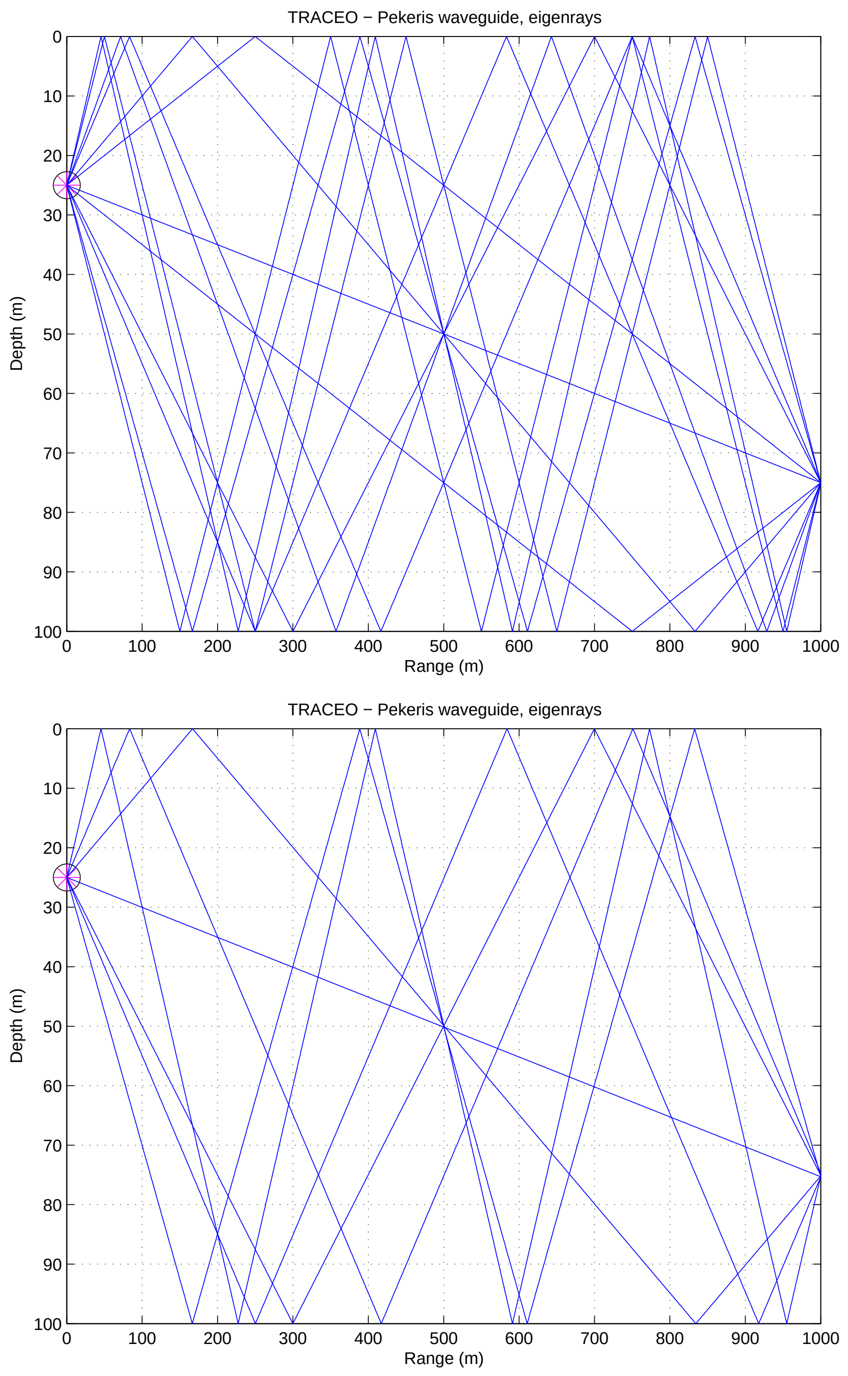


Figure 3.8: Pekeris waveguide: eigenrays calculated by Regula Falsi (top) and by proximity (bottom).

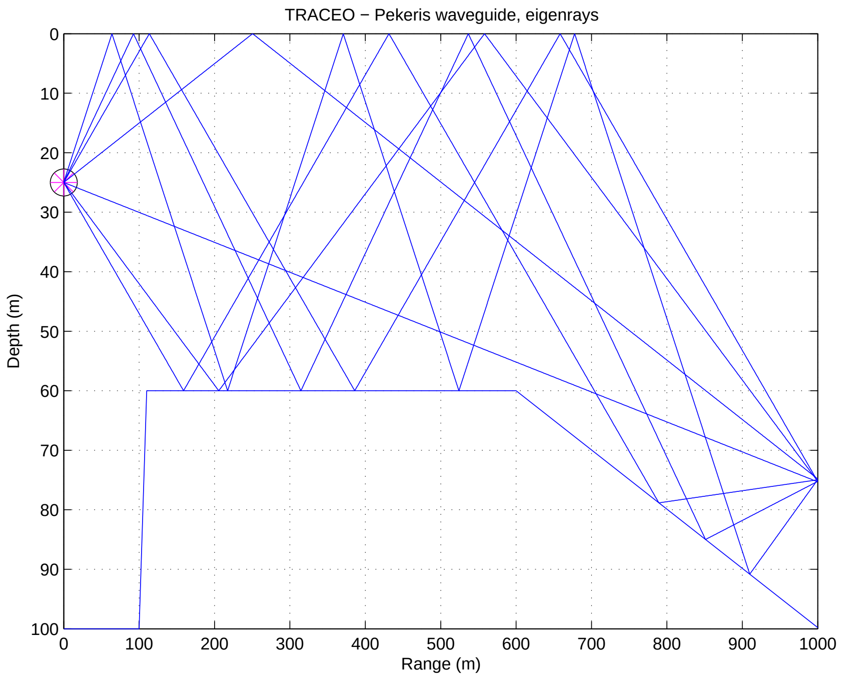


Figure 3.9: Pekeris waveguide with a wedge: eigenrays calculated by proximity.

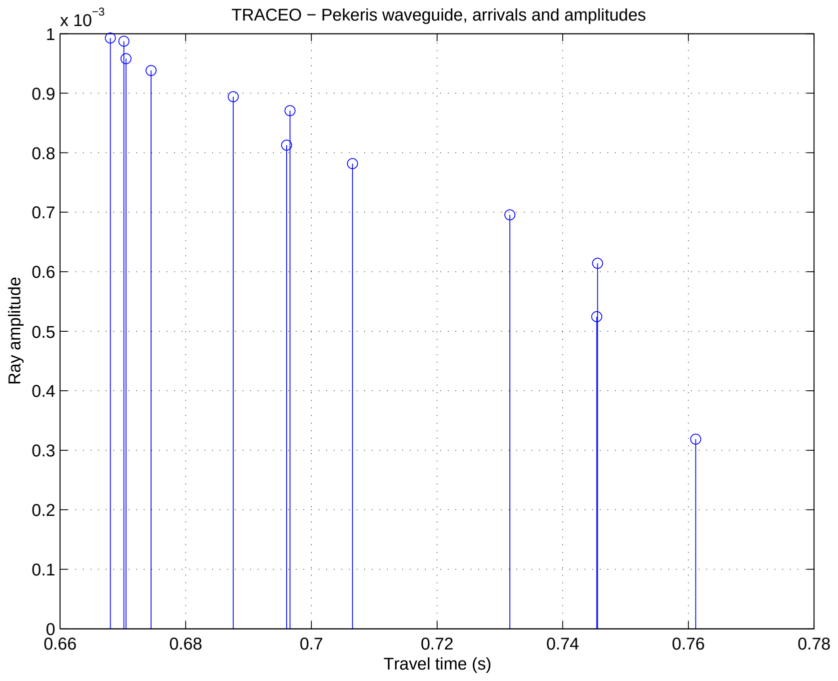
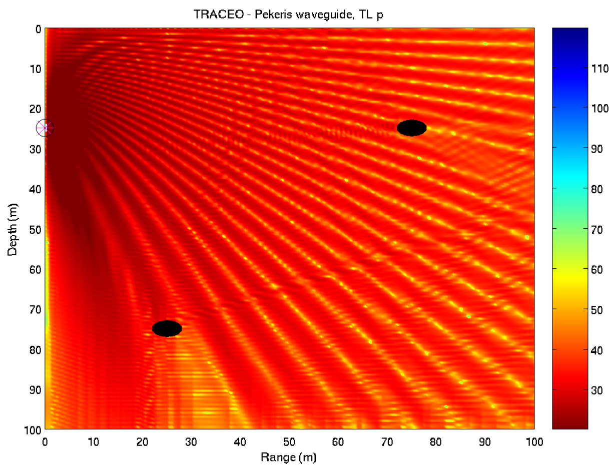
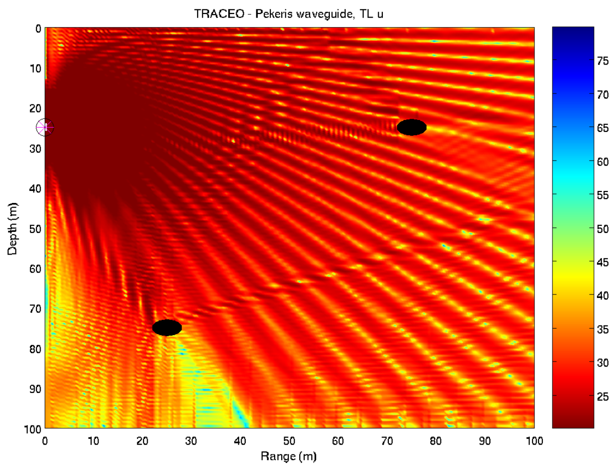


Figure 3.10: Pekeris waveguide: travel times and amplitudes calculated by Regula Falsi.





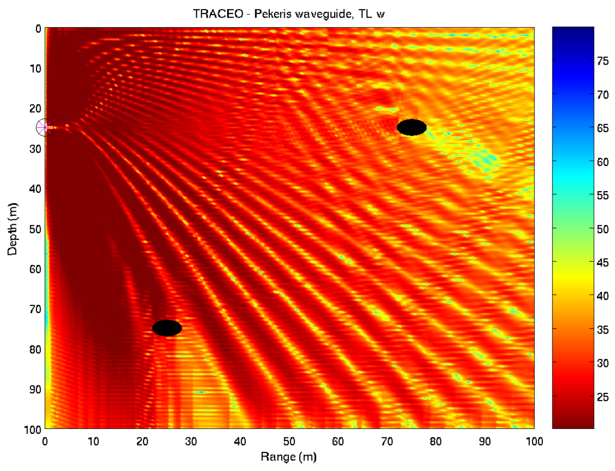


Figure 3.11: Pekeris waveguide: transmission loss for pressure (top), horizontal component of particle velocity (middle), and vertical component of particle velocity (bottom).

# Bibliography

[1] Rodríguez O.C. The TRACEO ray tracing program. SENSOCEAN Internal Report 1, SiPLAB, Campus de Gambelas, 800 FARO-PORTUGAL, January 2011.

[2] Rodríguez O.C., Collis J., Simpson H., Ey E., Schneiderwind J., and Felisberto P. Seismo-acoustic ray model benchmarking against experimental tank data. *J. Acoust. Soc. Am*, 132(2), August 2012.

[3] Ey. E. and Rodríguez O.C. cTraceo - User manual. SENSOCEAN Internal Report 1, SiPLAB, Campus de Gambelas, 800 FARO-PORTUGAL, January 2012.

[4] Porter M.B. and Liu Y-C. Finite-Element Ray Tracing. In *Theoretical and Computational Acoustics*, volume 2, pages 947–956, World Scientific Publishing Co., 1994.