Torbjørn Ringholm *Target depth estimation using hull mounted active sonars. Proceedings of the 37th Scandinavian Symposium on Physical Acoustics, Geilo 2 February – 5 February, 2014.*

Target depth estimation using hull mounted active sonars

Torbjørn Ringholm

Norwegian University of Science and Technology, Department of Mathematical Sciences, Postboks 7491, TRONDHEIM, Norway

Abstract

High false alarm rates are a problem in anti-submarine warfare in littoral waters using active broadband sonars. Automatic classification procedures may help combat this problem by filtering out detections due to non-threatening targets. An interesting feature for classification purposes is the depth of the target. Using sonars with vertical beamforming capabilities, the received signal from a target can be used to find a plausible estimate of the target's depth given an initial guess of the target's horizontal distance from the ship, the bottom profile and a profile for the speed of sound.

The estimation is done by an optimization procedure which varies the relevant parameters and models signals based on these parameters, then comparing the modelled signals with the received signal to find which parameters fit the received signal best. The modelling is based on a ray tracing procedure to find eigenrays for a proposed target depth, storing vertical arrival angles and arrival times for these eigenrays and synthesizing a signal based on the arrival angles and arrival times for comparison with the recorded signal. The ray tracing procedure is done numerically using Lybin, a platform developed by the Norwegian Defence Logistics Organization (NDLO). The validity of the eigenray finding procedure is confirmed, and results from testing the optimization procedure on synthetic data are presented along with a plan for further development.

Contents

1 Introduction

Sonar (SOund Navigation And Ranging) equipment is used, among other things, to detect underwater objects by propagation of sound waves. Two main kinds of sonar systems are in use - *passive* sonars only record environmental sound without disturbing the surroundings, while *active* sonars work by emitting a pulse of sound (called a *ping*) into the ocean and recording the resulting environmental sound [?]. If one considers the ocean as a system, these approaches are equivalent to either observing the system passively, or by actively exciting the system with some input (the ping) and observing the response (the recorded sound). Active sonar systems are often able to obtain more information about their surroundings than passive systems, especially due to their ability to measure *travel times* (the time from a ping is emitted until an echo is heard), which is vital to distance estimation. They are therefore often used in anti-submarine warfare.

When using active sonars in anti-submarine warfare, classification of targets is a key issue - if the ping elicits a response from the surroundings in the form of an echo from a target, is the target a shoal of fish, an oil pipeline, a submarine, rock formations or something entirely different? Such questions often arise when using active sonar in littoral waters - oceanic regions with a high occurrence of disturbing elements. Alarms from non-threatening objects are a problem since they complicate the tactical picture, and automatic classification schemes that can identify the source of an echo and filter away such false alarms are needed in order to simplify the work of the operator [?]. Automatic classification may also prove useful in the development of navigational systems for autonomous underwater vehicles, helping them navigate successfully in littoral waters.

Regular sonar processing focuses mainly on estimating the range (horizontal distance from sonar) and bearing of a target. A drawback with this approach is that large objects such as oil pipelines could be considered moving targets. A ship moving in parallel with the pipeline emitting pings at different points in time would place the pipeline at different points in space, thus creating the illusion of a moving target and causing a false alarm. Therefore, one important clue in automatic classification is estimating the target depth, as knowing the depth of the target could eliminate some options; if the target is located on the sea bottom, chances are that it is, in fact, a pipeline or some other large bottom litter object and thus not hostile, such that it may be deprioritized in favor of targets located closer to the surface.

Active sonar systems work by having a transmitter emit a ping of known frequency and amplitude into the ocean, then recording acoustic data through a receiver consisting of an array of underwater microphones called hydrophones, essentially listening for echoes of the emitted ping. The transmitter and the receiver are often assumed to be located at the same place, although they may be located on different parts of the ship. Once recorded, the acoustic data is processed to determine where the sound came from and at what times echoes of the ping return to the ship.

Determining the directionality of the signals is usually done by the *beamforming* technique – analogous to the human ear, the hydrophone array may be used to identify the direction from which a sound originates by observing time delays and phase shifts between recordings from different hydrophones. This is usually done in the horizontal plane to determine the North-South-East-

West directionality of the signal, but although such *horizontal* beamforming is standard in most sonars, some sonars also have good *vertical* beamforming capabilities, meaning that one can obtain vertical directionality for the signal as well. Knowing the vertical directionality of a signal allows us to determine not only the range of the target, but also its depth [?]. After beamforming, the acoustic data is processed to minimize noise and increase the *Signal-to-Noise Ratio* (SNR) of the recording. This is often accomplished by the use of matched filtering, essentially looking for the known form of the emitted ping in the received signal by means of convolution [?].

The result of this processing is a data set containing digital acoustic data sampled over a period of time and distributed over several channels corresponding to the beamformed angles. The problem of depth estimation of a target now becomes part of the inverse problem of determining the properties of the ship's surroundings from the recorded acoustic data. If no other information about the ocean were given, this would prove to be quite a hard problem, as the inverse problem would encompass ab initio estimation of the shape and properties of the ocean floor, the target's range (horizontal distance from the ship), the sound speed dependency on depth, et cetera, in addition to the estimation of the target depth. However, assumptions can be made about these additional parameters from previous measurements, giving at the very least an initial guess for the properties of the surroundings.

The proposed method for solving the inverse problem is by modelling and optimization. If a sufficiently accurate mathematical model is available, one may simulate the propagation of sound in the ocean and use this to synthesize signals. Working with the assumption that a modelled signal resembles the recorded signal if the model parameters resemble the real world parameters, one could try to fit the model parameters for the environment, including target depth, sound speed profile and bottom depth, to create a modelled signal that resembles the recorded signal as closely as possible. Due to the nonlinear nature of sound wave propagation in water, some care must be taken in creating such a mathematical model. Here, we shall use a ray backpropagation scheme [?]. This entails using *ray tracing* to obtain *eigenrays*. Ray tracing is a procedure in which the path of sound rays are calculated from an approximation of the nonlinear wave equation, similar to the tracing of light rays done in optics, to determine the different paths taken by the emitted ping and its echoes. Eigenrays are sound rays that reach a certain depth at a certain range, and are pivotal to determining the times at which and directions in which the echoes from a ping are recorded.

Calculating the path of sound rays in sea water is a non-trivial task. Due to differences in salinity and temperature at different ocean depths, the speed of sound will vary accordingly, producing refraction effects [?]. Matters are further complicated due to reflection of rays at the sea bottom and the surface. Obtaining an analytical solution for the path of a sound ray will therefore be an impossible problem in all but the simplest cases, introducing the need for numerical methods for obtaining these paths. This problem has been thoroughly analyzed, leading to tools such as Lybin, a platform developed by the Norwegian Defence Logistics Organization (NDLO) which, among other things, can compute ray paths accurately and effectively [?]. Lybin is used extensively for this purpose in this project.

Another point of interest is the determination of a proper objective function

for use in the optimization procedure. The inverse problem should not be ill-posed, that is, there should exist a unique, stable solution to the problem. The choice of objective function will influence the problem's properties in this respect. In addition, the optimization procedure should not be too costly in terms of computational power, meaning the objective function should not be too time consuming to evaluate. Some objective functions will be presented which try to address these issues.

The optimization procedure itself should be chosen so as to effectively and reliably produce satisfactory results. Since the objective function will have local minima, which should be avoided, there is also a need for an initialization procedure, in which a suitable starting point close to the global minimum is obtained with as little effort as possible.

Finally, the procedure's accuracy and stability in the presence of noise must be tested. Unfortunately, obtaining real life acoustic test data is hard. This research is carried out in collaboration with the Norwegian Defense Research Establishment (FFI), who have extensive amounts of acoustic data to test the procedure on. However, due to the classified nature of this data, results obtained when using it cannot be presented here. Instead, we shall use *synthesized* data - signals created by means of the acoustic model - for testing the procedure. Hopefully, the synthesized signals are modelled with sufficient fidelity so as to be interchangeable with real signals when it comes to testing.

2 Theory

This section will provide a theoretical background for the problem - first formulating the problem in a general manner and giving an overview of the proposed solution method. We then expand on some key subjects, providing a derivation of the mathematical model for wave propagation, a presentation of how to obtain eigenrays from this model, and some considerations on the effect of signal attenuation and reflection on signal strength. Furthermore, we present how to model a signal on the results of ray tracing, and how to remove noise from recorded signals. Lastly, the optimization method is presented, discussing the choice of objective function and how to represent the sound speed profile in a manner that is susceptible to ordinary optimization methods.

Some topics that are vital to the full problem will not be explored; beamforming and matched filtering will not be explained further, as we assume that the data obtained is already processed using these techniques. Technical details regarding the sonar equipment are not discussed, and neither are environmental effects such as absorption and reflection coefficients. This is because neither of these topics are directly connected to the numerical experiments carried out later, in contrast with topics such as analytical methods for finding eigenrays, which are used to verify the numerical eigenray finding scheme and therefore explored in greater detail.

2.1 Problem formulation and overview

Assume that a set of digital acoustic data is given, and that it is sampled at times $\{t_j\}_{j=1}^{N_t}$, beamformed in the directions $\{\theta_i\}_{i=1}^{N_\theta}$ and matched filtered to yield a set of measurements $\{S_{ij}\} = \{S(\theta_i, t_j)\}$ for the acoustic intensity S at

the receiver. This set of measurements is what we consider as the *signal* emitted by the system. Thus, we have a signal recorded in the directions θ_i and at the times t_j. Given that the signal contains echoes from a ping, our problem now consists of finding the target depth z_t the ping was reflected from.

Complications arise due to the recorded intensity being dependent upon many other quantities such as the geometry of the surroundings, sound speed, absorption coefficients of the bottom and so forth. In order to estimate the target depth, these quantities must either be known beforehand or estimated along with the target depth. Thus, our problem of target depth estimation is an inverse problem that is intractable unless certain other parameters can be determined simultaneously, such as the target range r_t and the depth of the sonar system, z_s. Another important factor, the sound speed profile $c(z)$, will generally be a function varying with depth. The bottom profile, $z_b(r)$, is in reality a function varying with the distance from the ship. However, we shall consider only flat bottom profiles, and denote the bottom depth only as z_b henceforth. A simplified drawing of the problem and the geometric quantities involved is given in figure 1.

Other parameters of importance are associated with the sonar system itself and the characteristics of the emitted ping, for example the ping's bandwidth, B, and the sonar's vertical beamwidth, θ_{BW}. These parameters can be considered as known, and are important in determining statistical quantities such as the uncertainty in measurements of arrival angles and arrival times.

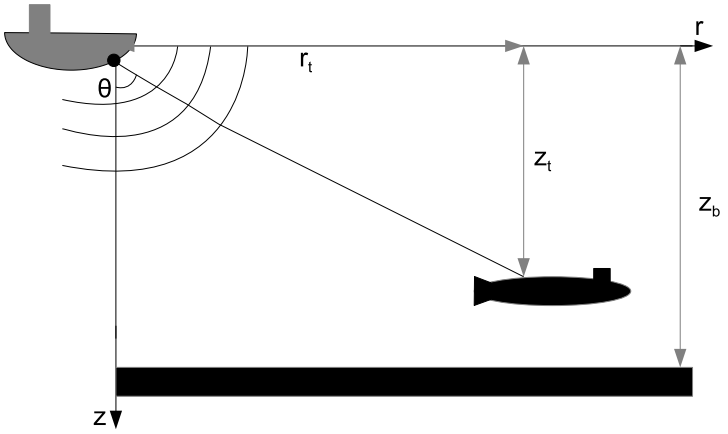

Figure 1: Simplified overview of geometric parameters. The line from the ship to the submarine shows the direct path taken by sound waves as a ray perpendicular to the wave fronts.

Whereas z_t is completely unknown beforehand, some assumptions may be made about the remaining environmental parameters:

- r_t can be estimated by regular horizontal beamforming and processing, providing an initial guess.
- $z_b(r)$ can also be considered known to a certain extent if historical bathymetric observations are available, providing an initial estimate. It is typically represented as a piecewise linear function. However, as stated earlier, we will consider only constant bottom depths $z_b(r) = z_b$, i.e. flat seabeds.

- z_s may vary slightly according to sea activity; in rough seas, the ship will bob significantly up and down and so will the sonar system, as it is fixed onto the vessel. The sonar will, however, have a certain expected depth which may be used as an initial guess.
- The sound speed profile, $c(z)$, will depend on depth, and is not known exactly. As is the subject of a later discussion, sound speed profiles can be estimated satisfactorily from historical observations, and by representing them as piecewise linear functions, we can optimize with respect to sound speed profiles by use of Empirical Orthogonal Functions (EOFs) [?].

Note that these additional parameters could, as a rough approximation, be considered as known and thus excluded from the optimization procedure. However, this would put the algorithm at considerable risk of mismatching, an undesirable situation in which reliable solutions cannot be obtained due to discrepancies between the modelled situation and the real physical situation [?].

2.2 Solution approach

We shall determine z_t and the other parameters by a method based on comparisons of the recorded signal and signals obtained by use of a mathematical model; if the modelled signal resembles the received signal closely enough, the parameters used in the modelling should be close to the correct parameters, and should yield a good estimate of the real parameters. It is therefore important to obtain an accurate mathematical model of sound propagation and a good method of modelling signals based on this, so that the modelled signals closely resemble real signals.

The modelling is based on finding *arrival times*, the time delays from the ping is emitted until the echoes are recorded at the receiver, and *arrival angles*, the directions in which the echoes are recorded after beamforming. With knowledge of these, a signal can be constructed. The arrival times and angles are calculated by use of eigenrays, paths which the sound follows to get to a specified depth at a specified distance from the ship [?]. For a set of candidate parameters, including z_t, we want to find eigenrays for z_t at r_t; we know that the sound must follow the eigenray paths to reach the target, and once reflected from it, must follow eigenray paths back toward the receiver. Hence, all possible paths for the ping are given by combinations of eigenrays.

Figure 2 shows an example of five eigenrays. Consider the sound travelling along any of the five rays from the source to the target. Upon reflection, it may follow either of the five eigenrays back toward the source again. In total, this gives 25 possible combinations of eigenray paths for the sound to follow. The eigenrays' exit angles at the source then become the arrival angles when propagated backward, and the arrival times become the times needed for the sound to travel along eigenrays forward and backward.

Using the arrival angles and travel times we can synthesize signals by assuming that each arrival results in a Gaussian shaped signal centered at the arrival time and angle, and superpositioning these signals. A typical signal is shown in figure 3. Note that this is a synthesized signal, thus containing more visible arrivals and less noise than a real recorded signal would, as reverberation and other environmental effects that would diminish signal strength are not accounted for. Also note that the signal is stronger in some places due to several

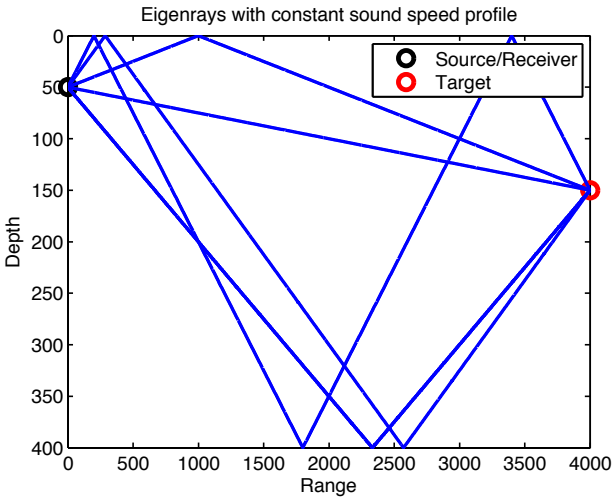

Figure 2: Five eigenrays with constant sound speed profile. $z_s = 50$ m, $z_t = 150$ m, $r_t = 4000$ m.

close arrivals superpositioning on top of each other. The solution procedure can be separated into five subsections, visualized in figure 4:

- Guessing model parameters - Obtain a guess for the relevant model parameters.
- Ray tracing - Based on the model parameters, calculate the trajectories of sound rays.
- Finding eigenrays – Given the results from the ray tracing and a guess for the target depth and range along with all other environmental and sonar parameters, find the exit angles and travel times of the eigenrays.
- Modelling signals – Using the eigenrays, calculate arrival angles and times. Use these to synthetize signals.
- Optimization – Compare the modelled signal to the recorded signal, and search for parameters that yield a modelled signal that resembles the recorded signal even more.

2.3 Mathematical model of sound propagation

Some assumptions must be made in order to obtain a feasible mathematical model. First, we may assume that the speed of sound in the ocean greatly exceeds the speed of both target and source, such that target and source can be considered stationary. Next, we assume that the time scale of sound propagation is so small that temporal variations in ocean conditions are negligible, and thus that sound speed is independent of time. We also assume that source and target can be considered as point masses, disregarding their geometrical shape. This assumption is valid for purposes of calculating the propagation of sound, but

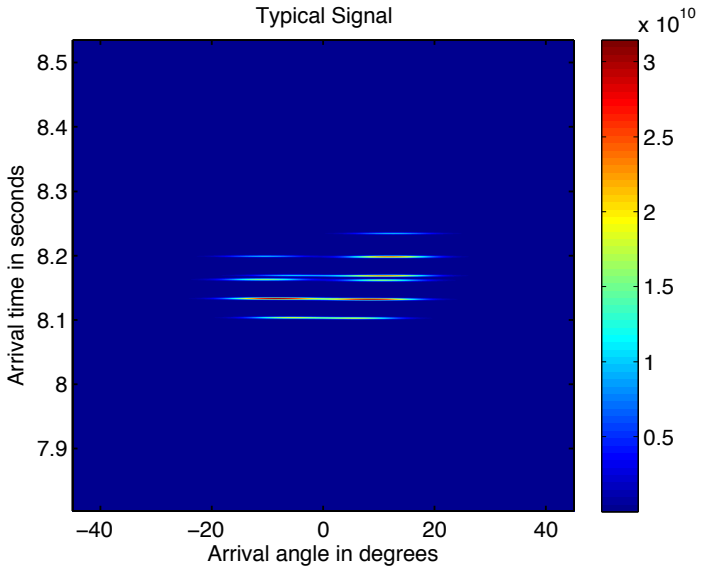

Figure 3: Synthesized signal.

as we shall see, corrections must be made to account for the surface area of the target when the strength of the reflected signal is taken into consideration. Next, we assume that source and receiver are located at the same position (a reasonable assumption for hull mounted sonars) and that signals are emitted omnidirectionally - that is, sound is emitted spherically from the transmitter. Finally, the matched filter used in preprocessing the data compresses pulses in time to a width of $1/B$, where B is the signal's bandwidth **[?]**. Thus, a signal will be of millisecond duration. We therefore assume that the ping's duration is so short as to be considered instantaneous. This yields a mathematically tractable yet still physically feasible model for propagation of the signal.

Considering the case without bottom or surface collision, i.e. the source and target being submerged in an infinitely deep body of water, the signal will expand outward with time, finally hitting the target. The ping is then reflected

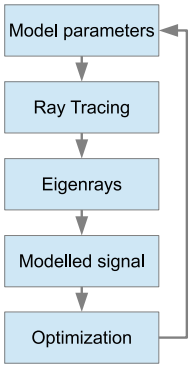

Figure 4: Flowchart of the solution process.

from the target, which acts as a new source, and propagates back toward the receiver, following the same path as the forward propagating sound wave. The signal finally reaches the receiver at a certain angle, the arrival angle, and a certain amount of time after the ping was emitted, the arrival time.

If the signal should hit the surface or the bottom, it will reflect and continue propagating as if emitted from a source located at the point of reflection, albeit with a weaker signal due to absorption effects. This additional signal will also propagate toward the target, reflect and return to the point of reflection, from which it continues propagating back toward the transmitter/receiver. This yields an alternative path for the signal to travel, adding an arrival angle and an arrival time.

2.3.1 The ray equation

To obtain an equation to model the propagation of sound in the ocean, we follow [?] and start with the wave equation for pressure, which we assume the sound waves will act according to. From this, we shall describe the paths taken by the sound by rays perpendicular to the wave fronts of the pressure wave by means of characteristics of the eikonal equation. The wave equation for pressure is given by:

$$\Delta p - \frac{1}{c^2}\frac{\partial^2 p}{\partial^2 t} = 0.$$

Given proper boundary and initial values, this could be solved numerically by finite difference or finite elements methods, but such methods are ineffective at the scale of underwater acoustical problems; a tractable solution would require a discretization with many nodes, and since our approach relies on determining the propagation of sound for many different configurations of the problem parameters, these methods would be too computationally expensive [?]. However, further assumptions and modifications can be made that yield simpler, more effective ways of solving the problem numerically. The first of these is using the one-dimensional Fourier transform as explained in [?] to eliminate time dependency and arrive at the Helmholtz equation:

$$\Delta p + \frac{\omega^2}{c^2} p = 0 \tag{1}$$

As ω, the angular frequency of the signal, is known, this becomes a problem in the spatial variables, represented in the vector $\mathbf{r}$. We now seek a solution to the Helmholtz equation in the form of a ray series:

$$p(\mathbf{r}) = \mathrm{e}^{i\omega\tau(\mathbf{r})} \sum_{j=0}^{\infty} \frac{A_j(\mathbf{r})}{(i\omega)^j}, \tag{2}$$

yielding for the gradient:

$$\nabla p = \mathrm{e}^{i\omega\tau} \left[i\omega\nabla\tau \sum_{j=0}^{\infty} \frac{A_j}{(i\omega)^j} + \sum_{j=0}^{\infty} \frac{\nabla A_j}{(i\omega)^j} \right] \tag{3}$$

and for the Laplacian:

$$\Delta p = \mathrm{e}^{i\omega\tau}\left[(-\omega^2|\nabla\tau|^2 + i\omega\Delta\tau)\sum_{j=0}^{\infty}\frac{A_j}{(i\omega)^j} + 2i\omega(\nabla\tau)^T\sum_{j=0}^{\infty}\frac{\nabla A_j}{(i\omega)^j} + \sum_{j=0}^{\infty}\frac{\Delta A_j}{(i\omega)^j}\right] \quad (4)$$

Inserting (2) and (4) into (1) now gives

$$(-\omega^2|\nabla\tau|^2 + i\omega\Delta\tau)\sum_{j=0}^{\infty}\frac{A_j}{(i\omega)^j} + 2i\omega(\nabla\tau)^T\sum_{j=0}^{\infty}\frac{\nabla A_j}{(i\omega)^j} + \sum_{j=0}^{\infty}\frac{\Delta A_j}{(i\omega)^j} = -\frac{\omega^2}{c^2}\sum_{j=0}^{\infty}\frac{A_j}{(i\omega)^j},$$

and by equating terms of equal order in ω, we get:

$$\begin{aligned}
\mathcal{O}(\omega^2) &: |\nabla\tau|^2 = \frac{1}{c^2} \\
\mathcal{O}(\omega) &: \Delta\tau A_0 + 2(\nabla\tau)^T\nabla A_0 = 0 \\
\mathcal{O}(\omega^{1-j}) &: \Delta\tau A_j + 2(\nabla\tau)^T\nabla A_j + \Delta A_{j-1} = 0 \qquad\qquad j = 1, 2...
\end{aligned}$$

The first of these is called the *Eikonal* equation, while the rest are called the *transport* equations - the eikonal equation describes the propagation of the pressure wave, while solving the transport equations yields the amplitude of the pressure wave. Typically, as a high-frequency approximation ($\omega \gg 0$), all equations but the eikonal and the first transport are neglected. In this case, as we are only interested in the path taken by the ray, so we disregard all transport equations and turn to the eikonal equation to obtain these paths. The equation may be solved by the method of characteristics. As can be seen from (3), ∇p is proportional to $\nabla\tau$, meaning $\nabla\tau$ is perpendicular to the wave fronts, and so we introduce the characteristic, or *ray trajectory*, $\mathbf{x}(s) = [x(s), y(s)]^T$, by

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} = c\nabla\tau \quad \Rightarrow \quad \left|\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s}\right|^2 = c^2|\nabla\tau|^2 = 1, \quad (5)$$

where s is the arc length along the ray. Differentiating once more with respect to s and applying the chain rule now yields:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{1}{c}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s}\right) &= \begin{bmatrix} \frac{\partial^2\tau}{\partial x^2} & \frac{\partial^2\tau}{\partial x\partial y} \\ \frac{\partial^2\tau}{\partial x\partial y} & \frac{\partial^2\tau}{\partial y^2} \end{bmatrix}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} = \begin{bmatrix} \frac{\partial^2\tau}{\partial x^2} & \frac{\partial^2\tau}{\partial x\partial y} \\ \frac{\partial^2\tau}{\partial x\partial y} & \frac{\partial^2\tau}{\partial y^2} \end{bmatrix} c\nabla\tau \\
&= \frac{c}{2}\frac{\mathrm{d}}{\mathrm{d}s}|\nabla\tau|^2 = \frac{c}{2}\frac{\mathrm{d}}{\mathrm{d}s}\frac{1}{c^2} = -\frac{1}{c^2}\nabla c
\end{aligned}$$

Changing to cylindrical coordinates, $\mathbf{x}(s) = [r(s)\, z(s)]^T$, and defining

$$\xi = \frac{1}{c}\frac{\mathrm{d}r}{\mathrm{d}s}, \qquad \eta = \frac{1}{c}\frac{\mathrm{d}z}{\mathrm{d}s}$$

we get a system of ODEs:

$$\begin{aligned} \xi &= \frac{1}{c}\frac{\mathrm{d}r}{\mathrm{d}s} \qquad \frac{\mathrm{d}\xi}{\mathrm{d}s} = -\frac{1}{c^2}\frac{\partial c}{\partial r} \\ \eta &= \frac{1}{c}\frac{\mathrm{d}z}{\mathrm{d}s} \qquad \frac{\mathrm{d}\eta}{\mathrm{d}s} = -\frac{1}{c^2}\frac{\partial c}{\partial z}. \end{aligned} \tag{6}$$

This system is solvable, given the right initial values, but is not very descriptive. To get a solution of the more intuitive form $z(r)$ we may observe that

$$\frac{\mathrm{d}r}{\mathrm{d}s} = c\xi \Rightarrow \begin{cases} c\eta = \dfrac{\mathrm{d}z}{\mathrm{d}s} = \dfrac{\mathrm{d}z}{\mathrm{d}r}\dfrac{\mathrm{d}r}{\mathrm{d}s} = c\xi\dfrac{\mathrm{d}z}{\mathrm{d}r} & \Rightarrow \dfrac{\mathrm{d}z}{\mathrm{d}r} = \dfrac{\eta}{\xi} \\ \dfrac{\mathrm{d}\xi}{\mathrm{d}s} = \dfrac{\mathrm{d}\xi}{\mathrm{d}r}\dfrac{\mathrm{d}r}{\mathrm{d}s} = c\xi\dfrac{\mathrm{d}\xi}{\mathrm{d}r} = -\dfrac{1}{c^2}\dfrac{\partial c}{\partial r} & \Rightarrow \dfrac{\mathrm{d}\xi}{\mathrm{d}r} = -\dfrac{1}{\xi c^3}\dfrac{\partial c}{\partial r} \\ \dfrac{\mathrm{d}\eta}{\mathrm{d}s} = \dfrac{\mathrm{d}\eta}{\mathrm{d}r}\dfrac{\mathrm{d}r}{\mathrm{d}s} = c\xi\dfrac{\mathrm{d}\eta}{\mathrm{d}r} = -\dfrac{1}{c^2}\dfrac{\partial c}{\partial z} & \Rightarrow \dfrac{\mathrm{d}\eta}{\mathrm{d}r} = -\dfrac{1}{\xi c^3}\dfrac{\partial c}{\partial z}. \end{cases}$$

These three equations can be further manipulated to form one second-order ODE by differentiating the first equation with respect to r and substituting the remaining two:

$$\frac{\mathrm{d}^2 z}{\mathrm{d}r^2} = \frac{\dfrac{\mathrm{d}\eta}{\mathrm{d}r}\xi - \eta\dfrac{\mathrm{d}\xi}{\mathrm{d}r}}{\xi^2} = \frac{-\dfrac{\partial c}{\partial z} + \dfrac{\eta}{\xi}\dfrac{\partial c}{\partial r}}{\xi^2 c^3} = \frac{-\dfrac{\partial c}{\partial z} + \dfrac{\mathrm{d}z}{\mathrm{d}r}\dfrac{\partial c}{\partial r}}{\xi^2 c^3}. \tag{7}$$

Next, note that from (5) we have:

$$\begin{aligned} 1 = \left(\frac{\mathrm{d}r}{\mathrm{d}s}\right)^2 + \left(\frac{\mathrm{d}z}{\mathrm{d}s}\right)^2 = \left(\frac{\mathrm{d}r}{\mathrm{d}s}\right)^2 + \left(\frac{\mathrm{d}z}{\mathrm{d}r}\frac{\mathrm{d}r}{\mathrm{d}s}\right)^2 = \left(\frac{\mathrm{d}r}{\mathrm{d}s}\right)^2 \left(1 + \left(\frac{\mathrm{d}z}{\mathrm{d}r}\right)^2\right) \\ \Rightarrow \left(\frac{\mathrm{d}r}{\mathrm{d}s}\right)^2 = \frac{1}{1 + \left(\dfrac{\mathrm{d}z}{\mathrm{d}r}\right)^2}. \end{aligned}$$

Using this, we may rewrite ξ^2 by:

$$\xi^2 = \frac{1}{c^2}\left(\frac{\mathrm{d}r}{\mathrm{d}s}\right)^2 = \frac{1}{c^2}\frac{1}{1 + \left(\dfrac{\mathrm{d}z}{\mathrm{d}r}\right)^2}$$

and finally, by substituting this into (7) we arrive at the *ray equation*:

$$\frac{\mathrm{d}^2 z}{\mathrm{d}r^2} = \frac{1}{c}\left[1 + \left(\frac{\mathrm{d}z}{\mathrm{d}r}\right)^2\right]\left[-\frac{\partial c}{\partial z} + \frac{\mathrm{d}z}{\mathrm{d}r}\frac{\partial c}{\partial r}\right].$$

We can easily impose initial conditions for this. Any ray trajectory must start at the source depth, so $z(0) = z_s$. In addition, we require $z'(0) = \tan(\theta_0)$, where θ_0 is the exit angle of the ray [**?**]. Assuming c to be invariant within the

geographical region surrounding the sonar, meaning that c is independent of r, the equation is simplified and our initial value problem becomes:

$$\frac{\mathrm{d}^2 z}{\mathrm{d}r^2} = -\frac{1}{c}\frac{\partial c}{\partial z}\left[1 + \left(\frac{\mathrm{d}z}{\mathrm{d}r}\right)^2\right], \qquad z(0) = z_s, \quad z'(0) = \tan(\theta_0). \tag{8}$$

Additionally, if the path is known, we can calculate the travel time by

$$t = \int_0^{r_t} \frac{\sqrt{1 + (z'(r))^2}}{c(z(r))}\mathrm{d}r. \tag{9}$$

2.3.2 Modelling signal strength

A propagating signal is subject to several gain and loss mechanisms altering the signal's strength - the magnitude of its intensity - upon arrival at the receiver. Knowledge of these is needed to synthesize a realistic signal and for other purposes, such as estimating the probability of detection of a signal.

The ping is emitted with a certain intensity called the *Source Level* (SL) [?]. The energy contained in the ping when emitted will be spread out as time goes on, thus reducing the intensity of the signal as it propagates. Some events, like bottom collisions or surface collisions, will accelerate this intensity loss by absorption or scattering of the acoustic waves. These losses, along with the attenuation loss due to geometric spreading of the signal, constitute the *Transmission Loss* (TL) [?].

As the target is, in reality, a reflecting surface and not a point mass, it will reflect a larger portion of the signal that reaches it, in effect acting as an amplifier for the reflected signal. This is represented by a gain called *Target Strength* (TS), which acts as a correction to the assumption that target and receiver are point masses [?]. Furthermore, due to the beamforming done at the receiver, there will be a gain in signal strength due to the amalgamation of several recorded signals into one directed signal [?]. This gain is represented in the *Directivity Index* (DI).

We must also consider that the signal is recorded in the presence of noise, which has an intensity called the *Noise Level* (NL). The logarithmic *sonar equation* combines all of these quantities to estimate the important *Signal-to-Noise Ratio* [?]:

$$SNR = SL - 2TL + TS - NL + DI,$$

which attempts to capture the effect of the different gain and loss mechanisms. Alternatively, for detection purposes, one could calculate the *Signal Excess* (SE), the strength of the recorded signal compared to some value for the *Noise Threshold* (NT). The NT is a preset level for separating noise from signal; if the SNR of some part of a signal is below this threshold, that part is considered noise. The SE is given by:

$$SE = SNR - NT = SL - 2TL + TS - NL + DI - NT. \tag{10}$$

Thus, if SE < 0 for some part of the signal, that part is considered noise. The SE is used to estimate the probability of detection of individual arrivals, and to remove noise from signals [?].

2.4 Finding eigenrays

The next problem at hand is using the ray equation to find the sound rays that reach the target depth at the target range - the eigenrays. For the proposed method, the most important aspect of the eigenrays is their arrival angles, so the problem consists of finding all θ_0 such that for a solution of the initial value problem (8) with $z'(0) = \tan(\theta_0)$, we have $z(r_t) = z_t$. The exact paths traced by the eigenrays are not of importance to this application - the other quantities of interest are the travel time and number of bottom and/or surface reflections of the eigenrays, as these may be used in the synthetization of signals. These quantities can be found, if only θ_0 is known.

Eigenrays can be found either analytically or numerically, depending on the complexity of the environment. For example, if the speed of sound is constant and the bottom perfectly flat, it is a simple geometric problem to find exit angles of the eigenrays given environmental parameters. However, if the bottom is irregular and the speed of sound varying with depth (as is the case in the ocean), analytical solutions are more or less impossible to obtain, and numerical strategies must be employed. The forward problem of calculating the path of a sound ray in the ocean, given initial angle, initial depth and sound speed profile, can be done quite comfortably by means of numerical ray tracing procedures. [?] However, the backward problem of calculating the initial angle from the target depth and range, i.e. finding which initial angles result in rays hitting the target, is a harder task. We here present first some analytical solutions to the ray equation (8) along with some considerations about eigenrays before introducing a numerical backpropagation scheme to identify eigenrays based on the forward propagation of several rays.

2.4.1 Analytical solutions

In most cases, $c(z)$ is of such a form that finding an analytical solution to (8) is virtually impossible. However, for some choices of $c(z)$, analytical solutions can be found, providing a useful basis for testing the accuracy and stability of the numerical eigenray schemes. Two such choices are the constant sound speed profile and the linear sound speed profile.

Constant sound speed profile:
In this case, since c is constant, equation (8) has a simple solution:

$$\begin{aligned} &\frac{d^2z}{dr^2} = 0, \qquad z(0) = z_s, \quad z'(0) = \tan(\theta_0) \\ &\Rightarrow z(r) = r\tan(\theta_0) + z_s. \end{aligned}$$

Thus, the exit angle for a direct eigenray (with no surface or bottom reflection) is given by

$$z(r_t) = z_t \quad \Rightarrow \quad \tan(\theta_0) = \frac{z_t - z_s}{r_t}.$$

Note that this result is easily obtained by geometric means - the path follows a straight line from $(0, z_s)$ to (r_t, z_t), and the above formula follows by trigonometry. Taking into account bottom and surface reflections is not very difficult. We

know that the ray will follow a straight line path before and after reflections. Assuming that reflections are specular, the slope of the path is the same before and after, and so we may find the initial angle through geometric considerations.

The ray will have to travel the same distance horizontally for all possible combinations of reflections, and it is the vertical distance traveled that varies. If the first reflection is a surface reflection, the ray will have to travel an additional distance of $2z_s$ vertically, or an additional $2(z_b - z_t)$ if the first reflection is a bottom reflection. For each subsequent surface reflection, the ray will travel an additional vertical distance of $2z_t$, and for each bottom reflection an additional $2(z_b - z_t)$, such that for a ray with m surface collisions and n bottom collisions, we find for the exit angle $\theta_0^{(m,n)}$:

$$\begin{aligned}\tan(\theta_0^{(m,n)}) &= \frac{z_t - z_s + 2z_s + 2(m-1)z_t + 2n(z_b - z_t)}{r_t} \\ &= \frac{(2(m-n)-1)z_t + z_s + 2nz_b}{r_t},\end{aligned}$$

if the first collision is a surface collision, and

$$\begin{aligned}\tan(\theta_0^{(m,n)}) &= \frac{z_t - z_s + 2(z_b - z_t) + 2mz_t + 2(n-1)(z_b - z_t)}{r_t} \\ &= \frac{(2(m-n)+1)z_t - z_s + 2nz_b}{r_t}\end{aligned}$$

if the first collision is a bottom collision. Also, finding the travel time $t^{(m,n)}$ is simple – from equation (9) we get:

$$t^{(m,n)} = \frac{r_t}{c\cos(\theta_0^{(m,n)})}.$$

An example of eigenrays found with a constant sound speed profile is shown in figure 5. We may note that the reflected rays in this case follow a strict pattern of subsequent surface and bottom reflections.

Linear sound speed profile:
In this case, $c(z) = \frac{c_b - c_0}{z_b} z + c_0$, where c_b is the sound speed at the bottom, yielding $\frac{\partial c}{\partial z} = \frac{c_b - c_0}{z_b}$. Thus, equation (8) becomes

$$\frac{d^2z}{dr^2} = -\frac{1}{z+\gamma}\left[1 + \left(\frac{dz}{dr}\right)^2\right], \qquad z(0) = z_s, \qquad z'(0) = \tan(\theta_0),$$

where $\gamma = \frac{c_0 z_b}{c_b - c_0}$. This can be rearranged to find:

$$\begin{aligned}&\frac{d^2z}{dr^2}(z+\gamma) + \left(\frac{dz}{dr}\right)^2 = \frac{d^2}{dr^2}\left[\frac{z^2}{2} + \gamma z\right] = -1 \\ &\Rightarrow \frac{z^2}{2} + \gamma z = -\frac{r^2}{2} + Ar + B. \qquad (11)\end{aligned}$$

By imposing initial conditions, we find

$$\begin{aligned}A &= \tan(\theta_0)(z_s + \gamma), \\ B &= z_s(\gamma + \frac{z_s}{2}).\end{aligned}$$

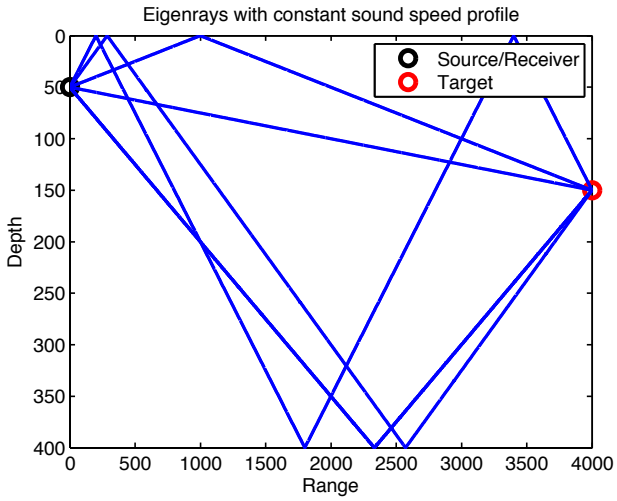

Figure 5: Five eigenrays with constant sound speed profile. $z_s = 50$ m, $z_t = 150$ m, $r_t = 4000$ m.

Substituting this into (11), rearranging and solving a quadratic equation with respect to z now yields

$$z(r) = -\gamma \pm \sqrt{(z_s+\gamma)^2 + 2\tan(\theta_0)(z_s+\gamma)r - r^2},$$

where the choice of sign depends on which convention for the sign of z is used, and the sign of γ. We shall adopt the convention that $z \geq 0$, such that the positive solution is used if $\gamma > 0$ and the negative if $\gamma < 0$, i.e.

$$z(r) = -\gamma + \mathrm{sgn}(\gamma)\sqrt{(z_s+\gamma)^2 + 2\tan(\theta_0)(z_s+\gamma)r - r^2},$$

where sgn is the sign function.

Dealing with reflections is somewhat more intricate for the linear sound speed profile than for the constant sound speed profile. We first need to find the ranges $\{r_n\}_{n=1}^N$ at which reflections occur, determine whether the reflection is from the surface or the sea bottom, and then find reflection angles before determining the further propagation of the ray. It is now suitable to generalize somewhat. We look for a solution of the ray equation on several intervals $\{I_n\}_{n=0}^N$, where $I_n = [r_n, r_{n+1}]$ and $r_0 = 0$. Let $\{z_n\}_{n=0}^N$ be the collection of the $z(r_n)$, and $\{\theta_n\}_{n=0}^N$ be the angles which the ray path forms with the horizontal at the points $\{(r_n, z_n)\}_{n=0}^N$. Repeating the preceding calculations on each interval and imposing the initial conditions $z(r_n) = z_n$, $z'(r_n) = \tan(\theta_n)$ gives us the path restricted to an interval I_n as

$$z(r)|_{I_n} = -\gamma + \mathrm{sgn}(\gamma)\sqrt{(z_n+\gamma)^2 + 2\tan(\theta_n)(z_n+\gamma)(r-r_n) - (r-r_n)^2}. \quad (12)$$

Assuming that z_{n+1} is known, we now find an expression for which range the reflection occurs at by solving $z(r_{n+1})|_{I_n} = z_{n+1}$ with respect to r_{n+1}, which gives:

$$r_{n+1} = r_n + \tan(\theta_n)(z_n+\gamma) \pm \sqrt{\tan^2(\theta_n)(z_n+\gamma)^2 + (z_n+\gamma)^2 - (z_{n+1}+\gamma)^2}.$$

Again, some care must be taken in order to choose the right sign. There are two possible values of z_{n+1}: $z_{n+1} = z_b$ and $z_{n+1} = 0$. Also note that we require $r_{n+1} > r_n$. If $z_{n+1} = 0$ and $\gamma > 0$, we have

$$(z_n + \gamma)^2 - (z_{n+1} + \gamma)^2 \geq 0,$$

such that the positive solution must be chosen to satisfy $r_{n+1} > r_n$. Conversely, if $z_{n+1} = 0$ and $\gamma < 0$, we have

$$(z_n + \gamma)^2 - (z_{n+1} + \gamma)^2 \leq 0,$$

such that $r_{n+1} > r_n$ for either solution. We then choose the negative solution, since we want the first point of contact with the surface. In summary, if $z_{n+1} = 0$, we have:

$$r_{n+1} = r_n + \tan(\theta_n)(z_n + \gamma) + \mathrm{sgn}(\gamma)\sqrt{\tan^2(\theta_n)(z_n + \gamma)^2 + (z_n + \gamma)^2 - \gamma^2}. \quad (13)$$

A similar calculation for the case $z_{n+1} = z_b$ yields

$$r_{n+1} = r_n + \tan(\theta_n)(z_n + \gamma) - \mathrm{sgn}(\gamma)\sqrt{\tan^2(\theta_n)(z_n + \gamma)^2 + (z_n + \gamma)^2 - (z_b + \gamma)^2}. \quad (14)$$

From the last result, we observe that given an exit angle θ_n such that

$$\tan^2(\theta_n) < \left(\frac{z_b + \gamma}{z_n + \gamma}\right)^2 - 1,$$

there can be no bottom reflection. In addition, it can be shown that $\mathrm{sgn}(z''(r)) = -\mathrm{sgn}(\gamma)$, so that if $\gamma > 0$, a bottom reflection will happen before a surface reflection if possible. Also note that if $\gamma > 0$ and $\tan(\theta_n) < 0$, we have $r_{n+1} < r_n$, which is non-admissible. Thus, we know that if $\gamma > 0$ and

$$\tan(\theta_n) > \sqrt{\left(\frac{z_b + \gamma}{z_n + \gamma}\right)^2 - 1}, \quad (15)$$

we will have $z_{n+1} = z_b$. Otherwise, $z_{n+1} = 0$. A similar calculation can be done if $\gamma < 0$; we will have $z_{n+1} = 0$ if

$$\tan(\theta_n) < -\sqrt{\left(\frac{\gamma}{z_n + \gamma}\right)^2 - 1}, \quad (16)$$

otherwise, $z_{n+1} = z_b$.

To obtain the angles θ_{n+1} at each reflection point, we use the law of specular reflection, $\theta_{n+1} = -\theta_{\mathrm{in}}$, where θ_{in} is the incoming angle from the left, given by

$$\tan(\theta_{\mathrm{in}}) = \lim_{r \to r_{n+1}^-} z'(r)|_{I_n}.$$

This leads to the condition

$$\tan(\theta_{n+1}) = -\lim_{r \to r_{n+1}^-} z'(r)|_{I_n},$$

from which we find

$$\tan(\theta_{n+1}) = \sqrt{(\tan^2(\theta_n)+1)\left(\frac{z_n+\gamma}{\gamma}\right)^2 - 1}, \qquad z_{n+1} = 0 \tag{17}$$

$$\tan(\theta_{n+1}) = -\sqrt{(\tan^2(\theta_n)+1)\left(\frac{z_n+\gamma}{z_b+\gamma}\right)^2 - 1}, \qquad z_{n+1} = z_b. \tag{18}$$

We have now fully determined the propagation of the rays in the linear sound speed case - given initial angle θ_0 along with initial depth $z_0 = z_s$, we can determine the first reflection depth z_1 by use of condition (15) or (16), then find the first reflection range range r_1 by use of (13) or (14) and the first reflection angle θ_1 by use of (17) or (18). These may again be used to find z_2, r_2 and θ_2, etc. The path the ray will follow between these points is given by (12).

An example of eigenrays for the linear sound speed profile case is given in figure 6. Comparing this with figure 5, we see that the rays are quite similar, yet curved in the linear sound speed profile case. This allows for rays that follow paths of surface reflections only, in contrast with the constant sound speed case, where rays had to follow a pattern of surface reflections followed by bottom reflections. This is important since bottom reflections absorb more energy from the signal than surface reflections, and we can see that it is now possible for a signal to follow a path of significantly less loss than with constant sound speed. In fact, this is possible for any sound speed profile in which the sound speed is increasing over a certain depth interval [?].

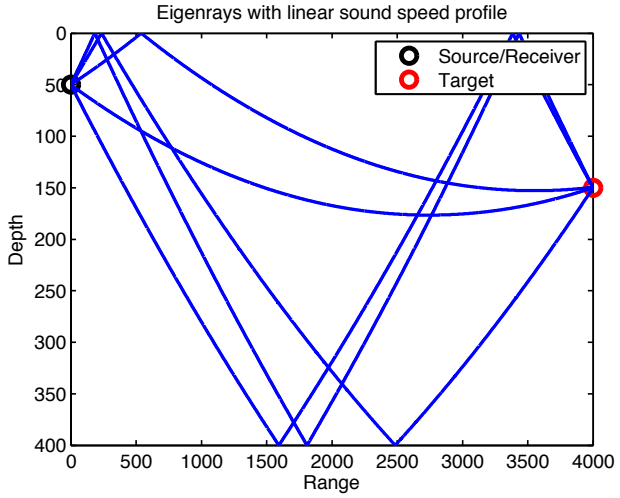

Figure 6: Eigenrays with linear sound speed profile. $z_s = 50$ m, $z_t = 150$ m, $r_t = 4000$ m.

Using equations (12-14) and (17-18), it is now possible to solve for the initial angle θ_0, given target depth z_t, source depth z_0, and target range r_t. The angle at which the ray hits the target is irrelevant. The direct path should pose no problems – we have one equation and one unknown, so the solution should be readily available. For each additional reflection, we introduce three unknowns;

an additional z_n, θ_n and r_n. We now look for rays with certain *histories* - the number, order and type of reflections (bottom or surface). Assuming knowledge of a ray's history, we know all the z_n. In addition, we have one equation for each of the θ_n and r_n, leaving us with as many equations as unknowns and the possibility of solving the set of equations recursively. However, although interesting, this is considered out of scope for this project, and we shall not attempt to solve these equations here.

The travel time is given by (9), which may be split this into several integrals if the number of reflections, N, of the ray is known:

$$t = \int_0^{r_t} \frac{\sqrt{1 + (z'(r))^2}}{c(z(r))} \mathrm{d}r = \sum_{n=0}^{N-1} \int_{r_n}^{r_{n+1}} \frac{\sqrt{1 + (z'(r))^2}}{c(z(r))} \mathrm{d}r + \int_{r_N}^{r_t} \frac{\sqrt{1 + (z'(r))^2}}{c(z(r))} \mathrm{d}r.$$

Once again, attempting to obtain an analytical expression for these integrals is considered outside the scope of the project, although it may be possible.

2.4.2 Numerical solutions

As can be seen from the preceding section, for all but the simplest choices of sound speed profile and bottom profile, numerical schemes are needed to find eigenrays due to the nonlinear nature of the ray equation making analytical solutions exceedingly hard to obtain.

Most numerical eigenray finding schemes are based on a procedure in which a large number of ray paths with varying exit angles are calculated by use of some numerical method such as a Runge-Kutta method or a linear multistep method on the set of ODEs given in (6). The rays that reach the specified target depth at the specified target range are then identified as eigenrays. These rays' exit angles are stored along with their histories, and their travel times are calculated by use of a numerical approximation to (9). Ray tracing procedures have been studied extensively, and several good programs for this purpose are available, one of which is Lybin, a platform developed by the Norwegian Defence Logistics Organization, which will be used here [?].

Lybin is a black box system, meaning its inner workings are not generally known. It allows for two different approaches to calculating eigenrays numerically, which we will name *method 1* and *method 2*. Method 1 relies heavily on Lybin, which through a built-in function can report all *families* of rays that enter a certain depth cell encompassing the target depth at the target range. A ray family is a collection of rays that share the same history. The function passes information about these families; the rays' mean travel time, mean exit angles, transmission losses and certain other statistics, such as maximum and minimum angles within each family. The mean exit angle and mean travel time of a family is considered as the exit angle and travel time of the eigenray belonging to that family.

Method 2 approach relies on Lybin only for tracing the paths of a multitude of rays leaving the source at increasing exit angles. Each ray is then inspected to see whether it lands closer to the target depth at the target range than the preceding ray and the next ray. If this is the case, the travel times of the three rays under consideration are calculated numerically, and the exit angles and travel times of the three rays are interpolated to obtain an approximation to

the exit angle and travel time of the eigenray they enclose. This leaves us more in control of the process, although the method is slower and may run into some issues with target depths that lie close to the surface or the bottom.

We thereby have two methods for finding eigenrays numerically - one that is fast, but not directly controllable and one that is slow, but allows for more transparency.

2.5 Modelling the signal

Infinitely many eigenrays can be found, but most of these result in arrivals that are so weak as to be indiscernible from noise. We therefore search for a large number of eigenrays, but keep only a few of them, typically those with least transmission loss. The optimum number of eigenrays to use must be decided as a trade-off between accuracy and speed; more eigenrays will result in better accuracy, at the expense of execution time. This is the subject of one of the tests done, as presented in section 4.

After N eigenrays have been found, we may construct a synthesized signal based on these, to use for comparison with the recorded signal. Recall that the signal consists of intensity measurements recorded in the directions θ_i and at the times t_j. Each eigenray presents a path the sound will follow from the source to the target and, by the assumption of stationary conditions, a path the sound will follow from the target back to the source after reflection. Thus, the sound will follow all possible combinations of eigenrays forward and back again, resulting in N^2 distinct *arrivals*, whose arrival times are the travel times along the eigenrays followed toward the target, in addition to the travel times along the eigenrays followed back toward the source. The arrival angle, the direction in which each arrival is recorded, is thereby the exit angle of the eigenray followed back to the source. Intuitively, one may consider standing in a cave and shouting; one may hear several echoes coming from different locations, due to the different times used by the sound to travel toward the walls before reflecting and the different arrival angles from the paths taken by the sound.

Furthermore, all arrivals are assumed to result in a Gaussian signal given by

$$S_m(\theta_i, t_j) = A_m \exp\left(-\frac{1}{2}\left[\left(\frac{t_j - t_m}{\sigma_t}\right)^2 + \left(\frac{\theta_i - \theta_m}{\sigma_\theta}\right)^2\right]\right),$$

where the t_m are the arrival times, θ_m the arrival angles, σ_t the signal's standard deviation in time and σ_θ its standard deviation in angle. Typically, $\sigma_t = 1/B$, where B is the signal's bandwidth, and $\sigma_\theta = \theta_{BW}/2$, where θ_{BW} is the beamwidth of the receiver [**?**]. The amplitudes, A_m, are chosen depending on the desired signal-to-noise ratio if noise is present, by letting

$$A_m = 10^{\frac{SNR}{10}} NL, \tag{19}$$

where SNR is the logarithmic signal-to-noise ratio and NL is the noise level. Superpositioning all these signals into one yields the noiseless synthesized signal:

$$S = \sum_{m=1}^{N^2} S_m. \tag{20}$$

This is the basis for the model signals which we try to fit to the recorded signal. If we are to use the signal as a substitute for a real signal for testing the solution method, it can be subjected to additive white noise to create a more realistic signal:

$$S = n(\mu_n, \sigma_n) + \sum_{m=1}^{N^2} S_m,$$

where $n(\mu_n, \sigma_n)$ is a random Gaussian process with expectation μ_n and standard deviation σ_n. We can use an acoustic model to calculate expected noise and reverberation levels given the surroundings, however, we shall use the simplifying assumption that $\mu_n = 70$ dB and $\sigma_n = 60$ dB re 1 μPa, respectively. Bear in mind that acoustic levels under water differ from those above water, and that this noise level therefore is not excessive. Now, the amplitudes A_m serve a purpose. They should be chosen as in (19). Typically, the *SNR* will lie in the range 10-30 dB. Also, if needed, individual differences can be made in the A_m to simulate the effects of signal attenuation from reflections. This is not done here, but may be an interesting topic for further improvement of the method.

2.6 Signal processing

After synthetization of a test signal, or after a real signal has been obtained, one should try to remove ambient noise as the optimization procedure may become unstable in the presence of noise. Since noise levels may vary with time due to engine noise etc., one should first try to normalize noise levels by making a local in time estimate of noise levels. This is done by employing a normalisation scheme called cell-averaging constant false alarm rate (CFAR) filtering [**?**]. The CFAR filter normalizes the signal levels against a local estimate of noise levels; proceeding entrywise through the signal, each entry is, in turn, considered a Cell Under Test (CUT). A local estimate of the noise around the CUT is obtained as the average of the surrounding cells; the closest cells, or *guard cells*, are not included in this estimate as they may be corrupted by the signal contained in the CUT. Figure 7 gives an illustration; only the cells marked WINDOW are used in the noise level estimates.

The size of the guard band and the windows are chosen according to which type of signal the CFAR filter is applied to. In our case, we shall use 40 entries in each window and a guard band of size $100\sigma_t$ to ensure that the guard band is large enough to contain a full arrival. These sizes are chosen heuristically, and have not created problems thus far. A better choice may be found, but this is outside the scope of this project, and may be followed up in later work.

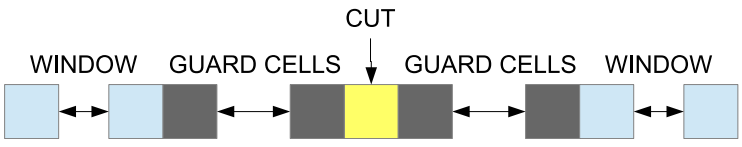

Figure 7: Illustration of CFAR filtering

After a local noise level estimate is obtained, the CUT is normalized by dividing the recorded intensity by the noise level estimate. This yields an es-

timate of the *SNR* levels in the signal, which can be used for estimating the signal excess as described in (10). The noise threshold used here is 13 dB. The SE estimate can in turn be used for *thresholding*, essentially removing all signal entries with an SE estimate lower than 0. This should remove most of the noise, and the remaining signal should be well suited for optimization afterwards.

2.7 Representing the sound speed profile by use of EOFs

Due to temperature and salinity fluctuations caused by seasonal variations and geographical differences such as ocean currents and sea depth, the sound speed profile, which dictates much of the propagation behaviour of sound, will vary between geographical regions and with time. Also, ocean temperature and salinity is strongly dependent on depth, leading to depth dependent sound speed. This complicates the matter of choosing the correct sound speed profile to match the environment. However, we shall assume range- and time independent sound speed profiles for the ranges considered here (0-10 km), as this simplifies matters considerably.

There is some difficulty in optimizing with respect to the sound speed profile $c(z)$, the main challenge being that it is a function, implying the need for variational methods whereas standard numerical optimization methods optimize with respect to scalar quantities. We therefore want to represent the sound speed profile by means of scalars, preferably as few as possible, to simplify optimization. We begin by approximating $c(z)$ by a continuous, piecewise linear function:

$$c(z) = \begin{cases} c_0 \dfrac{z_1 - z}{z_1} + c_1 \dfrac{z}{z_1} & 0 \leq z < z_1 \\ c_1 \dfrac{z_2 - z}{z_2 - z_1} + c_2 \dfrac{z - z_1}{z_2 - z_1} & z_1 \leq z < z_2 \\ \vdots & \\ c_{n-1} \dfrac{z_n - z}{z_n - z_{n-1}} + c_n \dfrac{z - z_{n-1}}{z_n - z_{n-1}} & z_{n-1} \leq z \leq z_n \end{cases}$$

where $\{z_j\}_{j=1}^n$ is some partition of the ocean depth and $\{c_j\}_{j=1}^n$ the sound speed at these depths. This function is completely determined by the depth partitioning $\{z_j\}_{j=1}^n$ and the coefficients $\{c_j\}_{j=1}^n$, such that we may represent $c(z)$ by a vector $\mathbf{c} = [c_0\, c_1\, \ldots\, c_n]^T$ and a vector $\mathbf{z} = [z_0\, z_1\, \ldots\, z_n]^T$.

Changing the sound speed profile now amounts to changing the c_i coefficients. However, there may be impractically many of these. In addition, an initial guess for the c_i is needed. Both these issues can be solved by the use of historical data and EOFs. Extracting EOFs from a data set is equivalent to using Principal Components Analysis on the data set, in which the most vital characteristics of the data is isolated by means of a Singular Value Decomposition (SVD) [**?**]. Assume that m measurements of the sound speeds at the specified depths have been made and are recorded in the vectors $\mathbf{c_1}, \mathbf{c_2} ... \mathbf{c_m}$. We may calculate the mean of these recordings, $\bar{\mathbf{c}}$, and use an SVD on the matrix formed by zero-mean column vectors, $[\mathbf{c_1} - \bar{\mathbf{c}}, \mathbf{c_2} - \bar{\mathbf{c}}, ..., \mathbf{c_m} - \bar{\mathbf{c}}]$ to obtain the EOFs $\mathbf{v_1}, \mathbf{v_2} ... \mathbf{v_m}$ as specified in [**?**]. Using these, we can represent the sound speed profile as

$$\mathbf{c} = \bar{\mathbf{c}} + \sum_{k=1}^{m} \gamma_k \mathbf{v_k}$$

where the γ_j are the weighting coefficients for the EOFs. If the data is well correlated, the first few EOFs will account for most of the variation in the data, and we may therefore truncate the expansion of $\mathbf{c}$ after the first few EOFs; typically, three EOFs will account for >95% of the total variation in the data. This is an acceptable error, and we therefore let

$$\mathbf{c} = \bar{\mathbf{c}} + \gamma_1 \mathbf{v_1} + \gamma_2 \mathbf{v_2} + \gamma_3 \mathbf{v_3}.$$

Now, by varying γ_1, γ_2 and γ_3, we also vary $c(z)$ in an efficient manner which is susceptible to ordinary optimization methods.

2.8 Optimization

The third and final part of the solution procedure, optimization, is done by comparing the recorded signal to a modelled signal, then attempting to modify the optimization parameters in the modelled signal in order to obtain a better fit. The choice of which parameters to optimize with respect to is a matter of complexity and accuracy. By optimizing with respect to too few or inconsequential parameters, we risk obtaining a sub-optimal fit, and as a result, an incorrect target depth estimate. On the other hand, if too many parameters are included, the computational complexity of the problem may become insurmountable. The parameters to optimize with respect to should therefore be chosen carefully.

Of course, z_t should be among the optimization parameters. Other suitable candidates for optimization parameters are r_t, z_b, z_s and c, as these parameters influence the eigenray paths used in the modelled signal. As explained in the preceding section, optimization with respect to c entails optimizing with respect to two or three weighting coefficients γ_1, γ_2 and γ_3.

2.8.1 Objective function

In order to compare the two signals, we need an objective function. First, since the signal is sampled at discrete N_t discrete points in time and beamformed in N_θ discrete angles, it can be represented in a matrix:

$$S = \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1N_\theta} \\ S_{21} & S_{22} & & \vdots \\ \vdots & & \ddots & \\ S_{N_t 1} & \dots & & S_{N_t N_\theta} \end{bmatrix}$$

where each S_{ij} is the intensity sampled at the time t_i and in direction θ_j. Similarly, the modelled signal is given by

$$M = \begin{bmatrix} M_{11} & M_{12} & \dots & M_{1N_\theta} \\ M_{21} & M_{22} & & \vdots \\ \vdots & & \ddots & \\ M_{N_t 1} & \dots & & M_{N_t N_\theta} \end{bmatrix}.$$

The most obvious objective function for comparing the two signals, which we will name the *full objective function*, is now given by

$$f(M;S) = ||S - M||^2, \tag{21}$$

for some matrix norm $||\cdot||$. We shall use the Frobenius norm, effectively finding the root-mean-square distance between the two signals,

$$||A||_F = \sqrt{\sum_{i=0}^{N_t}\sum_{j=0}^{N_\theta}|A_{ij}|^2}.$$

If the observed signal can be exactly reproduced, the function will have a unique minimum when $S = M$, a desirable property. However, when dealing with real world signals and in the presence of random noise it is almost certainly impossible to obtain a perfect reproduction of the recorded signal by modelling, and so the unique minimum will not be attained. Nevertheless, this objective function remains a viable option.

A problem with the direct comparison mentioned above is that it is computationally expensive, since each evaluation requires the formation of a full model signal M. An approximation can be done by considering the signal in vector form. Let

$$\mathbf{s} = \begin{bmatrix} S_{11} \\ \vdots \\ S_{N_t 1} \\ S_{12} \\ \vdots \\ S_{1N\theta} \\ \vdots \\ S_{N_t N_\theta} \end{bmatrix} \quad \text{and} \quad \mathbf{m} = \begin{bmatrix} M_{11} \\ \vdots \\ M_{N_t 1} \\ M_{12} \\ \vdots \\ M_{1N\theta} \\ \vdots \\ M_{N_t N_\theta} \end{bmatrix}.$$

We now have

$$f(M;S) = ||\mathbf{s} - \mathbf{m}||_2^2 = \mathbf{s}^T\mathbf{s} - 2\mathbf{s}^T\mathbf{m} + \mathbf{m}^T\mathbf{m}.$$

Since the term $\mathbf{s}^T\mathbf{s}$ is independent of $\mathbf{m}$, it can be considered constant and therefore irrelevant to optimization. Moreover, the modelled signal M is a superposition of signals from the arrivals, as explained in (20), so we may write

$$\mathbf{m} = \sum_{k=1}^{N^2}\mathbf{m_k},$$

where each $\mathbf{m_k}$ corresponds to the partial signal resulting from the k'th arrival. From this, we see that by disregarding the $\mathbf{s}^T\mathbf{s}$ term, we can form the equivalent objective function

$$\bar{f}(M;S) = -2\mathbf{s}^T\mathbf{m} + \mathbf{m}^T\mathbf{m} = \mathbf{m}^T\mathbf{m} - 2\sum_{k=1}^{N^2}\mathbf{m_k}^T\mathbf{s}.$$

If the $\mathbf{m}^T\mathbf{m}$ term could now be disregarded, we would arrive at a much more computationally efficient objective function. Since each of the $\mathbf{m_k}$, due to the Gaussian shape of the signal they contain, are mostly zeroes, we can compute the sum term very quickly by simply truncating the $\mathbf{m_k}$ to a smaller size containing only nonzero entries and computing the inner product of the truncated vector with the corresponding entries in the recorded signal, essentially exploiting the sparsity of the $\mathbf{m_k}$ signals.

In fact, as we shall see in section 4, this approach works well for a range of problems, but not in all cases. We therefore introduce the objective function, which we will name the *simplified objective function*, given by

$$g(M;S) = -\mathbf{s}^T\mathbf{m} = -\sum_{k=1}^{N^2} \mathbf{m_k}^T\mathbf{s} \tag{22}$$

as an inferior, yet more efficient alternative to the full objective function. The problem with the simplified objective function is that if the recorded signal has an area in which arrivals are clustered (that is, several arrivals that are so close in time and angle that they superposition on top of each other), this simplified objective function will value modelled signals which gather all arrivals in this cluster area higher than those where arrivals are more spread out.

2.8.2 Optimization algorithm

The black box nature of Lybin makes partial derivatives of any objective function with respect to the problem parameters impossible to obtain, leaving us with the choice of either a derivative-free optimization algorithm or using numerical gradients in a more sophisticated algorithm. As the objective functions are generally computationally expensive to compute, we would like to limit the amount of evaluations needed. Calculating numerical gradients calls for several evaluations per approximation, thus favouring derivative-free algorithms. Due to its robustness and ease of implementation, the algorithm chosen here is the derivative-free Nelder-Mead algorithm [**?**].

To avoid local minima, the Nelder-Mead algorithm requires that the optimization start reasonably close to the global minimum. To find such an initial guess, an exhaustive search method is employed, computing the objective function values with different problem parameters and choosing the parameters that yield the lowest objective function value. This approach quickly runs into the curse of dimensionality, as an increasing number of optimization parameters necessitates a large number of evaluations to obtain a reasonable initial guess.

For example, with two parameters one may wish to check the objective function values for five choices of each parameter, necessitating 25 evaluations of the objective function. If a third parameter were introduced, and one wants to check the objective function values for five choices of this parameter as well, 125 evaluations of the cost function are needed; a large increase in function evaluations.

3 Implementation and test setup

3.1 Implementation

The method outlined in section 2 has been implemented in MATLAB. Figure 8 shows a schematic overview of the process. Due to the modular structure of the program, implementation of the method is done by use of many separate functions, details about which can be found in Appendix A. For even more detailed descriptions of the functions, please review the source code.

Lybin is accessed through the binary interface LybinCom 6.1 and incorporated in the MATLAB code through a COM server [?]. Since Lybin is a 32-bit-only program, a 32-bit version of MATLAB had to be used in order for the LybinCOM extension to function. The Nelder-Mead optimization algorithm being used is already implemented as the built-in MATLAB function `fminsearch.m`, and is the only non-basic built-in MATLAB function used in the code.

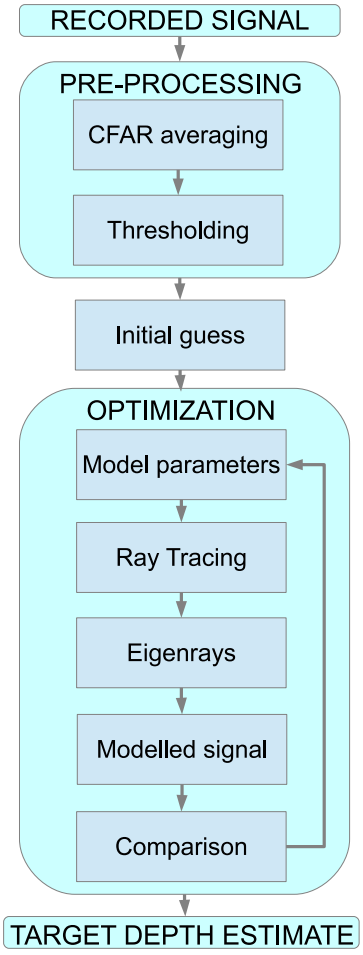

Figure 8: Flowchart of the complete solution process.

3.2 Test setup

3.2.1 Verification of numerical eigenray estimates

A test was done to check whether the eigenray candidates produced numerically in fact reach the specified depth at target range, and in which cases the eigenray estimates might fail. Five eigenrays were calculated by use of Lybin for each set of environment parameters (r_t, z_b and z_t). The target range was varied from 1000 m to 10 000 m in steps of 1000 m, the bottom depth from 100 m to 1000 m in steps of 100 m, and the target depth was varied from 50 m to 850 m in steps of 200 m. The source depth z_s was kept constant at 50 m throughout the test. A linear sound speed profile was used, in which sound speed varied from 1480 m/s at the surface to 1500 m/s at the bottom.

The numerical eigenray procedure produced five exit angles $\{\theta_i\}_{i=1}^5$ for each set of parameters; these exit angles were used as initial conditions in an analytical ray tracing, as described in section 2.4.1. The resulting analytical depth at target range given the numerical exit angles, $z(r_t;\theta_i)$, was compared with the desired target depth z_t, giving the mean error in eigenray depth at target range:

$$E = \frac{1}{5}\sum_{i=1}^{5} |z_t - z(r_t;\theta_i)|.$$

Both numerical schemes presented in section 2.4.2 were tested. Recall that *method 1* signifies the method based on Lybin's detection of ray families entering the depth cell containing the target depth at target range, and *method 2* signifies the more transparent method of calculating ray trajectories that enclose the target ray and interpolating between these.

3.2.2 Optimization test on synthesized data

Obtaining real acoustic data to test the procedure on is not easy. FFI has several sets of recorded data which may be used, all of which are classified, such that publishing results based on this data is disallowed. It was therefore necessary to test the procedure on synthesized data, modelled as proposed in section 2.5 with added Gaussian noise. To obtain a more realistic signal and to test the method's sensitivity to disturbances, the arrival angles and arrival times used in synthesizing the received signal were considered Gaussian distributed random processes, as proposed in [?]:

$$\theta_i = n\left(\bar{\theta_i}, \frac{\theta_{BW}}{\sqrt{s}}\right), \quad t_i = n\left(\bar{t_i}, \frac{1}{B\sqrt{s}}\right).$$

Here, $n(\mu,\sigma)$ specifies a Gaussian process with expected value μ and standard deviation σ; $\bar{\theta_i}$ and $\bar{t_i}$ are the arrival angles and arrival times as found by the numerical eigenray scheme, θ_{BW} is the vertical beamwidth of the sonar, B is the bandwidth of the sonar and $s = 10^{SNR/10}$ is the linear signal-to-noise ratio of the echoes. The θ_i and t_i were obtained by first calculating $\bar{\theta_i}$ and $\bar{t_i}$ numerically, then adding Gaussian noise to these values. By varying the SNR values, s is also varied, allowing us to test the method's stability in the presence of different levels of noise in the signal and inaccuracies in measurements.

For all tests, the nonlinear sound speed profile shown in figure 9 was used. The parameters used in the tests were target ranges from 2000 m to 10 000 m

in steps of 2000 m, bottom depths from 200 m to 1000 m in steps of 200 m, and target depths from 50 m to 50 m above bottom depth in steps of 100 m. The source depth was held constant at 5 m. In addition, all tests were done with use of three eigenrays, then redone with five eigenrays, in an attempt to determine how many eigenrays should be used in modeling signals to achieve a reasonable estimate of target depth. Due to time constraints, no tests were carried out in which the sound speed was varied as outlined in section 2.7, and as such, these tests are a priority in future work. The source depth was not varied, either.

For each set of parameters, five iterations were done in which a signal was synthesized by the method described above, and the optimization procedure applied to this signal in order to estimate the target depth. The mean error of these target depths estimates were then calculated. Both objective functions given in section 2.8.1 were used in the test, to see whether they yielded different results. While applying the simplified objective function (22), both methods of estimating eigenrays were used. Only method 1 was used while applying the full objective function (21). Again due to time constraints, the parameter range was shortened for the runs with full objective function; target ranges were varied from 2000 m to 10 000 m in steps of 4000 m, bottom depths from 200 m to 1000 m in steps of 400 m, and target depths from 50 m to 150 m above bottom depth in steps of 200 m. The results of the test have been analyzed in four ways:

- The first analysis determines whether three eigenrays are sufficient in modeling signals, or if five eigenrays should be used.
- The second analysis looks at the error in target depth estimation as a function of SNR and target range, to see how sensitive the procedure is to increasing noise levels, and how sensitive it is to increasing target range. It also looks at the differences between the two numerical eigenray methods, when employed in evaluating the simplified objective function.
- The third analysis is similar to the second, as it looks at the error in target depth estimation as a function of SNR and bottom depth to further determine the sensitivity of the procedure to noise, and to investigate the sensitivity to bottom depths. A comparison between the two numerical eigenray methods is given here as well.
- The fourth analysis compares the results obtained by use of the full objective function with the results obtained by use of the simplified objective function, to see whether the simplified objective function is a reasonable approximation.

3.2.3 Optimization test on real data

The optimization routine has been tested on real data as well, albeit without definitive results. It appears to work, in the sense that the code runs without crashing, and that it produces estimates of the target depth; however, the validity of these estimates have not been checked. There is also a problem with finding proper arrival times in the real data, due to arrival times not being provided along with the acoustic data. An ad hoc fix to this problem has been applied, but it is not known whether this fix is correct. Therefore, and due to the classified nature of the data these test have been applied to, no results from the optimization test on real data is presented at this time.

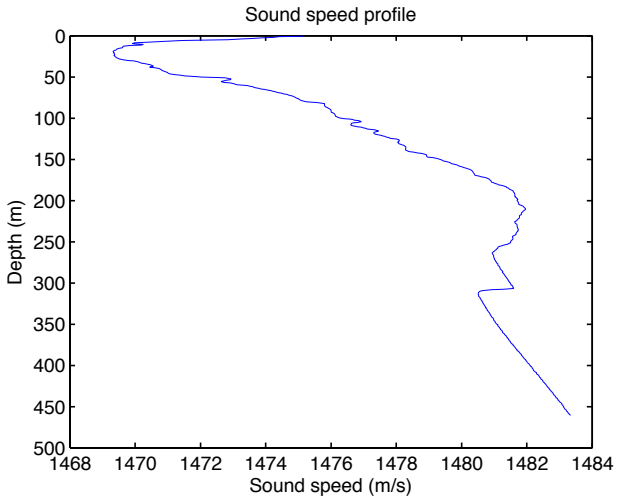

Figure 9: Sound speed profile used for testing.

4 Results and discussion

4.1 Verification of numerical eigenray estimates

Figure 10 shows the results of the verification test. Note that it is impossible that target depth is greater than bottom depth, and so the mean error in these cases is presented as 0 in the figures. From looking at the the figure, it is evident that method 1 is slightly less accurate than method 2. Note, however, that method 1 seldom produces a mean error larger than 5 m, which is acceptable. Also note that both methods show diminishing accuracy as the target range increases, and as the bottom depth decreases. This is to be expected, as ray tracing at large ranges requires more steps with the underlying numerical scheme than ray tracing at close range in deep waters, thus accumulating a larger numerical error, and since ray tracing in shallow waters is more sensitive to errors due to a higher number of bottom reflections. We may also note that the eigenray estimates are poorer for shallow target depths, especially for method 1, as shown by the upper-left plot. The reason is unknown, but examination of this problem, along with the question of why eigenray method 1 yields poorer results than method 2, is considered as future work and requires an investigation of internal algorithms of Lybin.

4.2 Optimization test on synthesized data

4.2.1 Number of eigenrays for signal modelling

Figure 11 shows an example of the maximum, minimum and mean estimates of target depth obtained by use of the simplified cost function with numerical eigenray method 2. From the figure, we see that using five eigenrays for calculating arrivals on which to model signals provides more consistent and correct estimates for the target depth than using three eigenrays. It is worth noting that estimates with use of three eigenrays produces a clear bias of 15 m toward

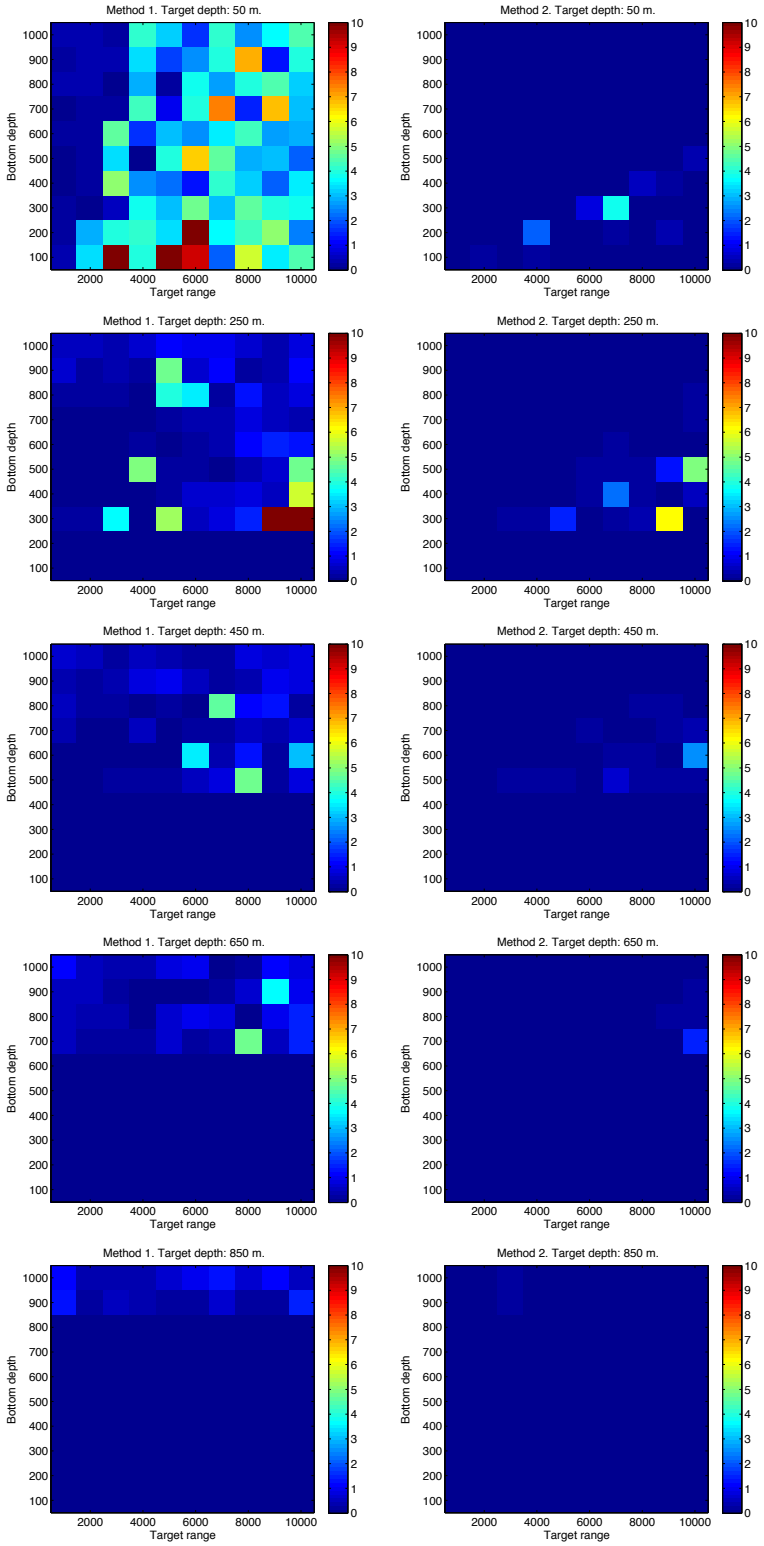

Figure 10: Mean error in eigenray depth at target range. Left column: Method 1. Right column: Method 2.

the bottom. The reason for this is unkown. That five eigenrays is superior to three is to be expected, since using more eigenrays provides more information, which should lead to better estimates. Note that the estimates improve as SNR increases. Also, the results are satisfying enough to discourage the use of more than five eigenrays. For the sake of brevity, we shall henceforth consider only results obtained with the use of five eigenrays.

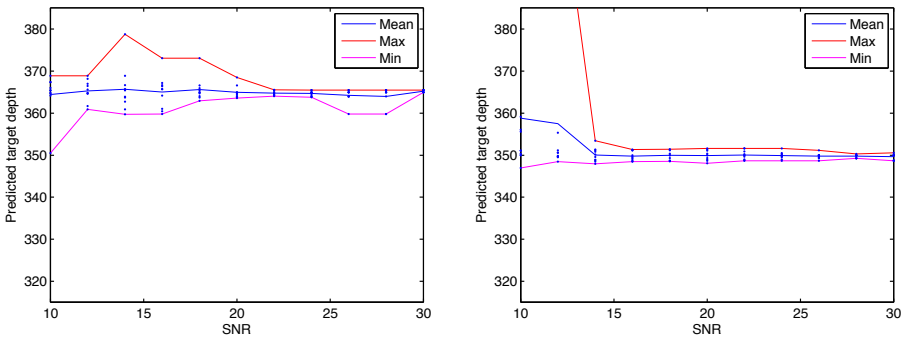

Figure 11: Estimates of target depth as a function of SNR. Target depth: 350 m. Bottom depth: 800 m. Target range: 6000 m. Left: 3 eigenrays. Right: 5 eigenrays. Obtained by use of simplified cost function with numerical eigenray method 2.

4.2.2 Estimation error as a function of SNR and target range

The plots in figure 12 show the error in the target depth estimation as a function of *SNR* and target range for four different target depths. The left column contains errors occurring with use of method 1 for obtaining eigenrays, while the right column contains errors occurring with use of method 2.

First, we may observe that the errors are mostly within acceptable range for classification purposes; we only need an approximate estimate for target depth to say whether it is close to the bottom or not, and estimates with errors of less than 50 m are good enough for this purpose. Second, we may note the markedly better performance obtained by use of method 2, as compared to method 1, in nearly all cases but those with target depth 50 m and target range 2000 or 10 000 m, along with the case where the target depth is 350 m and the target range is 10 000 m. This may be attributable to the slightly more inaccurate eigenray estimates provided by method 1, as observed from figure 10. The sound speed profile chosen for this test is more irregular than a linear sound speed profile and as such, the differences in eigenray accuracy between method 1 and method 2 could be exacerbated in this case, leading to poor performance in estimating target depth. However, the irregularities may well be the effect of an implementation error, and further investigation into the matter is in order.

Looking at the results from method 2, we see two irregular events with range 2000 m and target depths 50 and 150 m; with these parameters we recieve poor estimates of the target depths, whereas with all other ranges and the same target depths we find good estimates. This may be due to the sound speed profile being

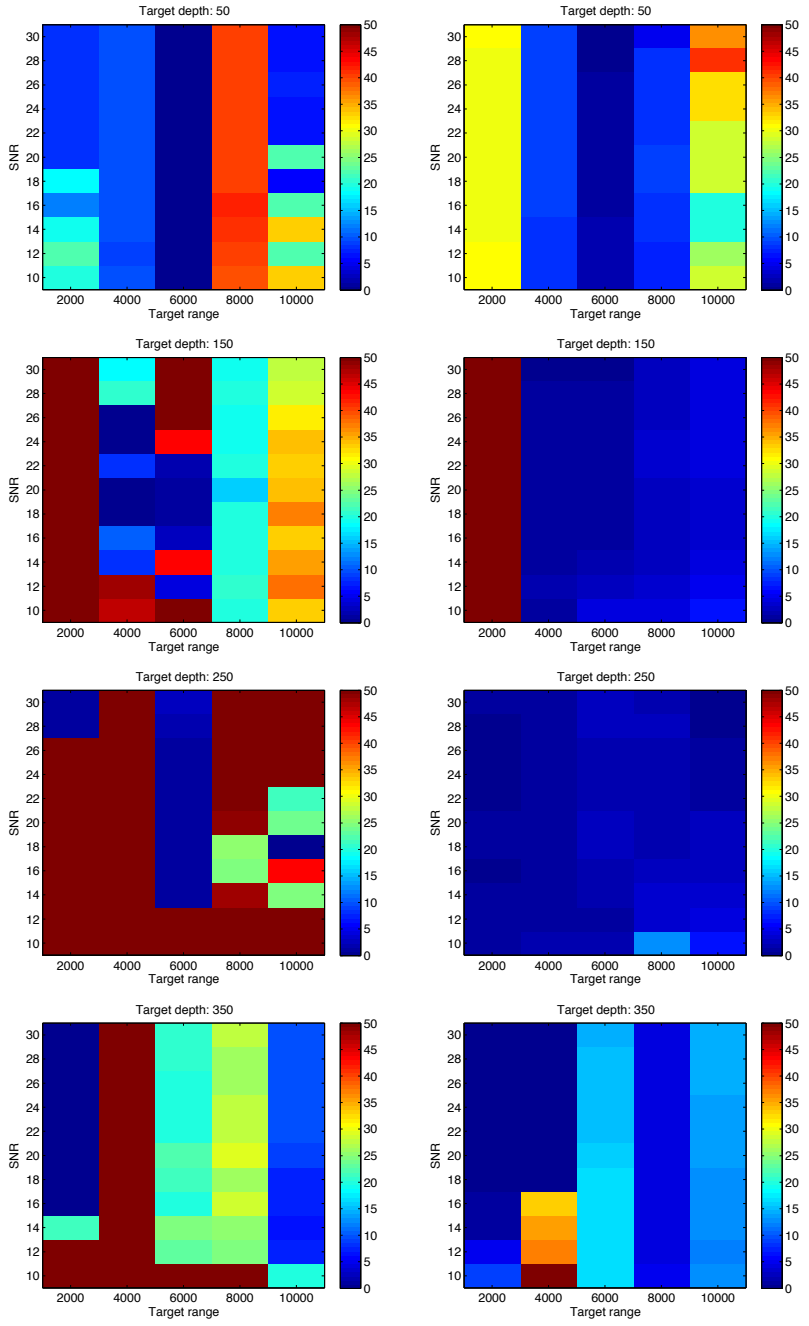

Figure 12: Mean error in estimates of target depth as a function of SNR and target range. Bottom depth: 400 m. Left column: Method 1. Right column: Method 2.

used which, due to its shape in the section 0-200 m, may make depth estimation in this depth range difficult. Rays will tend to curve toward areas of lower sound speed, meaning that in the channel between 0 and 200 m, there will be many eigenrays with small differences in exit angles and arrival times, making it hard to determine the exact depth of the target[?]. However, despite high errors we can conclude that a target is in this channel, giving important information for classification purposes. We may also note that estimates generally decrease in quality with increasing target range, as is to be expected. It is also worth noting that in both cases, the error decreases with increasing *SNR* in most cases where the estimation does not fail completely. In any case, the results are good enough to use for classification, at least when using eigenray method 2; the results from method 1 are not always as good.

4.2.3 Estimation error as a function of SNR and bottom depth

The plots in figure 13 show the error in the target depth estimation as a function of SNR and bottom depth for four different target ranges. The left column contains errors occurring with use of method 1 for obtaining eigenrays, while the right column contains errors occurring with use of method 2.

Again we see that although most estimates using both methods are suitable for classification purposes, method 2 is superior to method 1 in most cases, with the exception of the cases where target range is 4000 m and bottom depth is 200 m or 800 m. As before, this may be attributable to the inaccuracy in eigenray calculation when using method 1, or it may be due to implementation errors.

Looking at the results for method 2, we still see a general trend of improving estimates with higher SNR, and we can see a trend of better estimates with larger bottom depths, as is to be expected since ray tracing in shallow water more sensitive to errors in initial angle than ray tracing in deep waters, due to a larger amount of bottom reflections. Also note that method 2 still fails in some cases with small target ranges (2000 m and 4000 m). This may be attributable to the sound speed profile used, as explained previously. Once again, we may conclude that the results obtained by use of method 2 are acceptable for classification purposes while the results from method 1 are not always on par.

4.2.4 Comparison of objective functions

Figures 14 and 15 show the error in target depth estimation as a function of SNR and bottom depth, and as a function of SNR and target range, respectively. In both figures, the left column contains estimation errors occurring after applying the full objective function evaluated by use of numerical eigenray method 1, while the middle and right columns contain estimation errors occurring after use of the simplified objective function evaluated by use of numerical eigenray methods 1 and 2, respectively. In figure 15, five target depths are used while the bottom depth is kept at 1000 m, and in figure 14, three target ranges are used while the target depth is kept at 50 m.

In figure 14, we can see that the full objective function outperforms the simplified objective function. Note that the simplified objective function with eigenray method 1 actually performs better than the one evaluated by use of method 2 here, giving acceptable estimates of target depth in all cases, which would imply that the eigenray estimations work correctly for method 1 in this

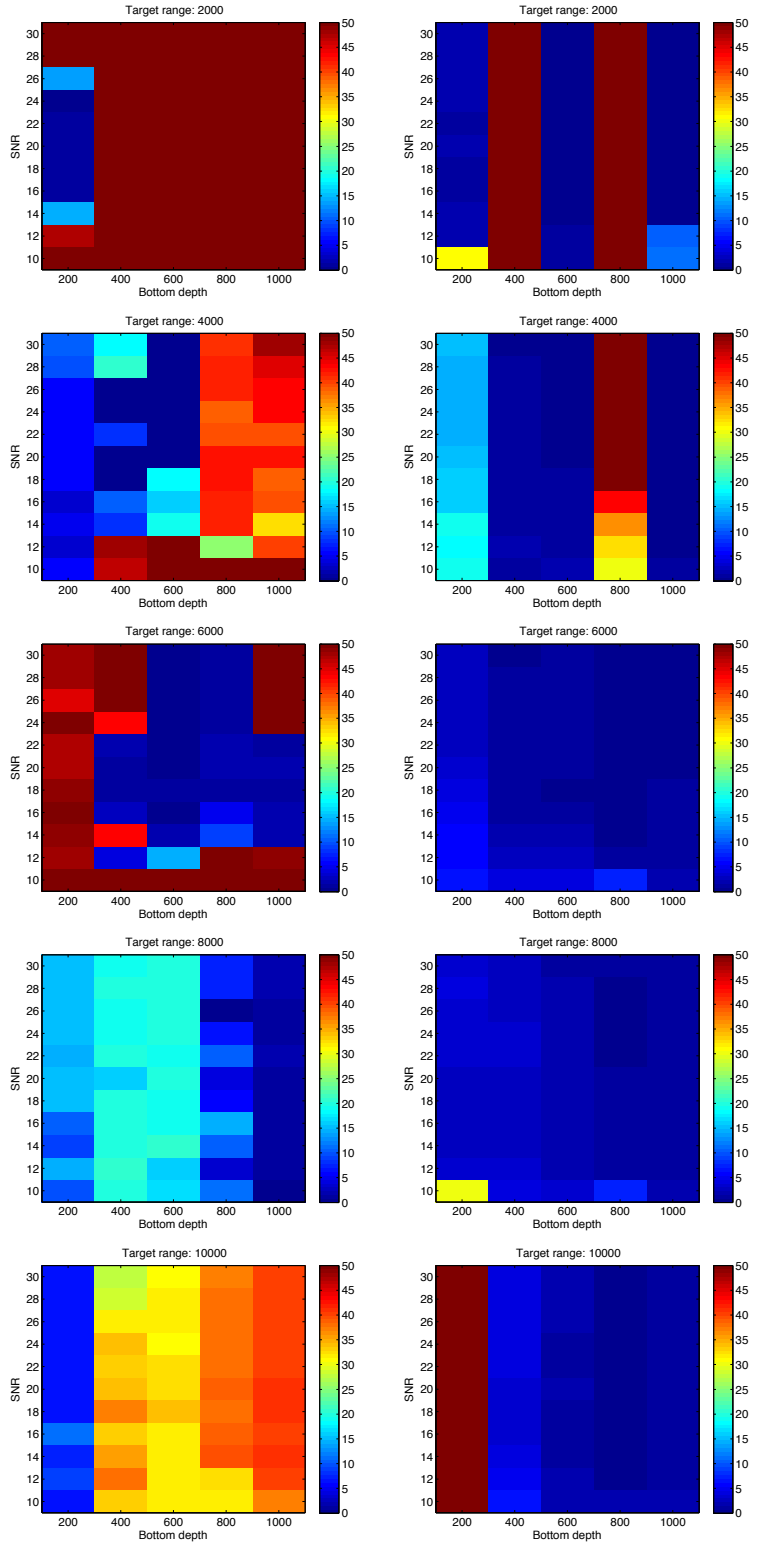

Figure 13: Mean error in estimates of target depth as a function of SNR and bottom depth. Target depth: 150 m. Left column: Method 1. Right column: Method 2.

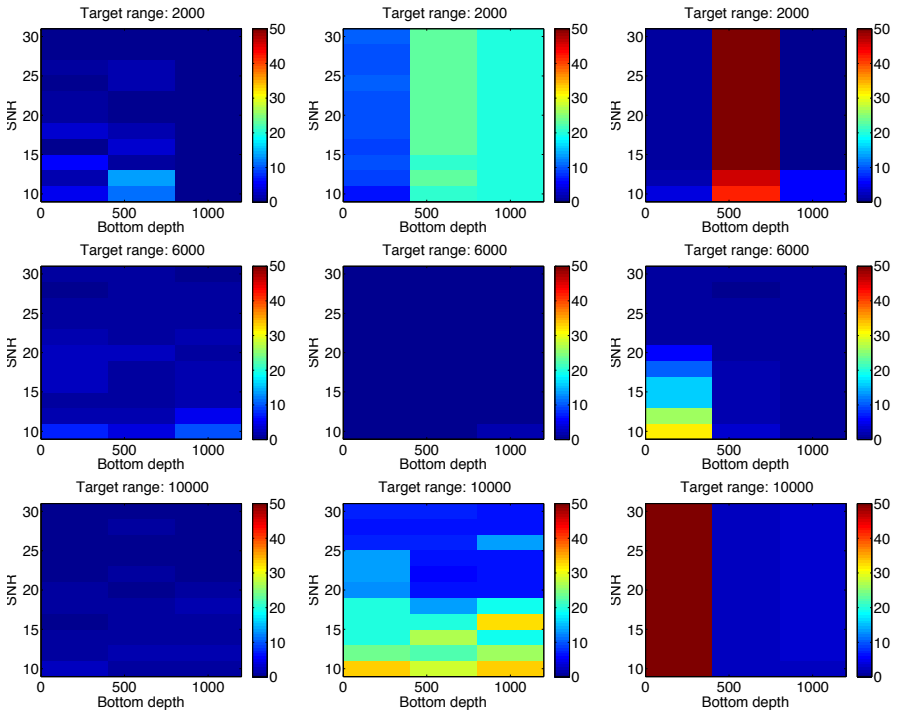

Figure 14: Mean error in estimates of target depth as a function of SNR and bottom depth. Target depth: 50 m. Left: Full objective function. Middle: Simplified objective function, using method 1. Right: Simplified objective function, using method 2.

case. These results are to be expected, as the simplified objective function is an approximation to the full objective function, and we would expect the full objective function to perform better, given reliable eigenray estimates.

However, in figure 15 we see that although all three methods show mostly acceptable results, the simplified objective function with eigenray method 2 works best in all but two cases; with target range 2000 m and target depth either 250 m or 450 m. Also note that the full objective function and the simplified objective function with eigenray method 1 have similar patterns in where their estimates break down, which may be interpreted as further proof that numerical eigenray method 1 is somewhat unsound and should be further looked into. It would also be interesting to observe what results can be obtained by use of the full objective function evaluated with eigenray method 2; such a test should be prioritized in the future.

4.2.5 Execution time

An important part of the solution method is its execution time. Using the simplified objective function evaluated with method 2, the optimization procedure took 8 minutes on average in the worst cases (large target range), and 2.5 minutes in the best cases (small target range). This is mostly due to the need to call on Lybin many times to find the ray paths while modeling signals to evaluate

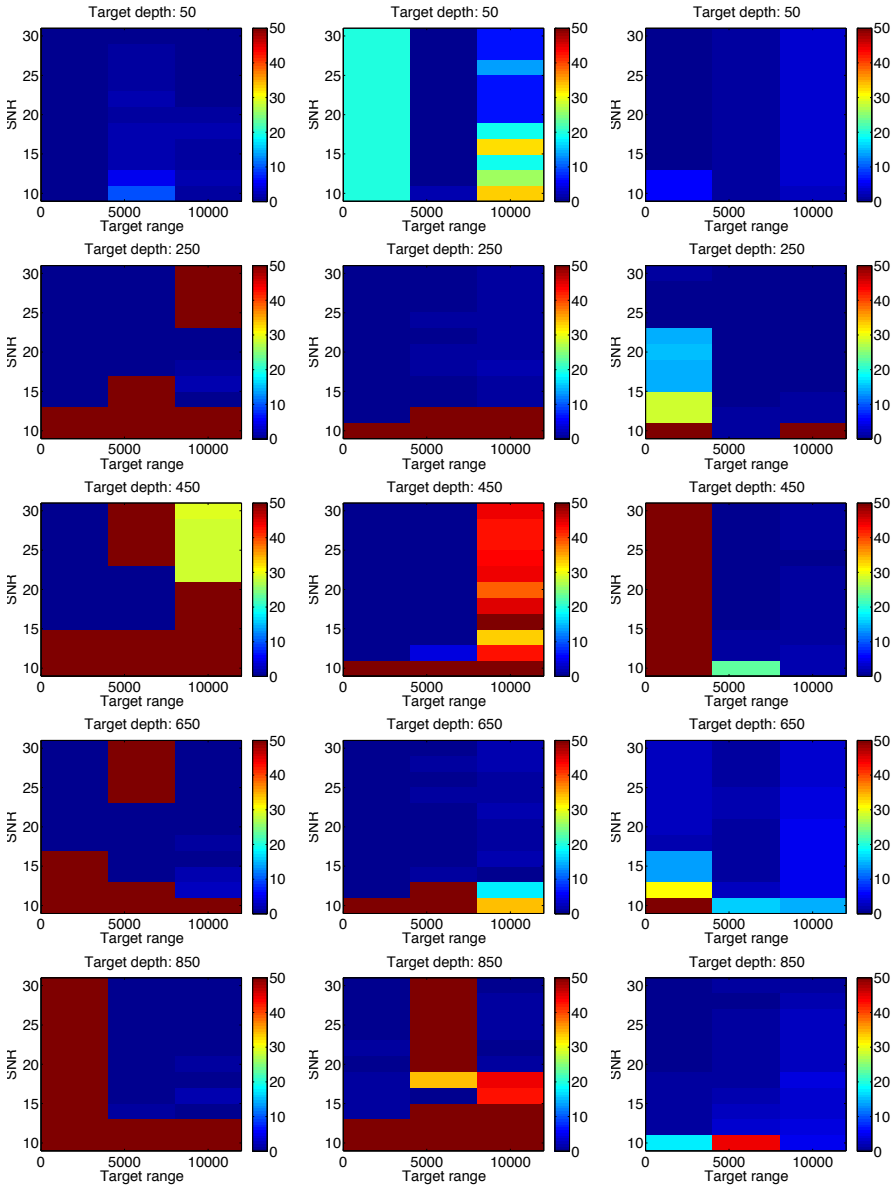

Figure 15: Mean error in estimates of target depth as a function of SNR and target range. Bottom depth: 1000 m. Left: Full objective function. Middle: Simplified objective function, using method 1. Right: Simplified objective function, using method 2.

the objective function. It is not known whether the slowdown is attributable to MATLAB's plugin module being slow or whether Lybin does an unnecessary amount of work in order to supply the ray tracing, but the time cost is quite prohibitive. In contrast, evaluating with method 1, while inaccurate (as seen above), is 20-30 times faster than method 2 due to it needing only one call to Lybin. Therefore, if method 1 could be made more accurate, it would provide a faster alternative to method 2.

The optimization procedure took 90 minutes on average to finish when using the full objective function, making this option prohibitively slow. This is attributable to the need for a fully formed model signal for comparison with the received signal. In any case, an optimization procedure requiring less function evaluations would be useful. Specifically, the initial guess routine is quite slow due to the high amount of function evaluations involved. A more effective way of producing initial guesses would probably speed up the execution time considerably.

5 Future work and conclusion

5.1 Future work

There are several directions in which to continue work on the estimation procedure; refining the method to produce more accurate results, reducing the execution time, and testing the method further to ensure its stability and reliability. Here, several topics are presented in which improvements can be made.

Objective function

- As of now, a signal comparison technique is mostly being used, in which whole signals are compared. However, if discrete arrival angles and times could be extracted from the received signal, a direct comparison between modeled and recorded arrival angles and times might be quicker and cleaner. If so, ambiguities about which angles and times to compare must be resolved.

- If the simplified cost function is used, weighting the contributions of different arrivals by their probability of detection (which can be calculated from estimates of signal excess) may reduce the influence of weak arrivals on the cost function, essentially giving more value to matching arrivals which have a higher probability of showing up in the recorded signal.

- A cost function based on Bayesian inference may be a better choice than the direct comparison of signals; Bayesian inference has previously been used with good results in acoustic and oceanographic applications [?] [?]. There may also be a way in which the direct comparison method can be interpreted in terms of Bayesian inversion, placing the method on firmer theoretical ground. Also, if an objective function based on Bayesian probabilities is used, a natural extension to estimating target depth through information gained from several pings may be obtained.

Signal processing

- A more rigorous way of choosing the size of guard bands and windows for CFAR averaging should be found.
- If an objective function based on extracting discrete arrival times and angles from the received signal is to be used, it relies on signal processing to extract this information.

Optimization algorithm

- As of now, the Nelder-Mead algorithm is being used. This is not necessarily the best solution, and a better algorithm may exist. Possible candidates are simulated annealing; a derivative-free stochastic optimization method, or gradient methods such as the BFGS method using numerical gradients [?].
- A faster method for producing initial guesses would be of great practical value as it would reduce the time needed for optimization, which is currently quite slow. It may be possible in some cases to solve a coarser version of the problem, that is, with a less detailed sound speed profile or with a coarser computational grid for ray tracing, and use the solution of this as an initial guess for the optimization method.

Analytical solutions for eigenrays

- An analytic solution to finding the initial angles of eigenrays in the case where the sound speed profile is linear might be attainable, as outlined in section 2.4.1. If so, it might be possible to obtain analytical solutions for initial angles of eigenrays in the case of piecewise linear sound speed profiles, which would essentially eliminate the need for numerical methods for obtaining eigenrays.

Testing and debugging

- Further investigate why eigenray estimates are poorer for small target depths when using a linear sound speed profile, as observed in section 4.1.
- Further investigate why eigenray estimates using method 1 are poorer than method 2, and why target depth estimation using method 1 yields worse results than using method 2. As a part of this, it might be interesting to test whether target depth estimation using the full objective function and numerical eigenray method 2 gives better results than with numerical eigenray method 1.
- Test optimization with respect to EOF coefficients.
- It may be of interest to use bottom profiles instead of flat sea floors, to check the method's sensitivity to varying bathymetric conditions.

5.2 Conclusion

Estimation of target depth has been carried out on synthesized acoustic data using two different objective functions. The results obtained are acceptable for classification purposes, and the best results were obtained while using the simplified objective function evaluated using numerical eigenray method 2. It remains to be seen whether this method is successful when optimizing with respect to sound speed profile and source depth and if so, whether the success transferable to real scenarios. The anomalies encountered while employing numerical eigenray method 1 during estimation need to be investigated further, as alleviating these would result in a faster procedure.

Appendices

A Implementation details

A.1 Parameters

To simplify parameter passing and to promote consistent naming throughout the code, two structs are used for parameter passing – one containing all computation parameters, option flags etc., and one containing signal data and information about the signal.

Parameter struct

The parameter struct has five fields which are again structs: geometry, ray, env (short for environment), signal and comp (short for computation). These are loosely thematically organized, although not very well organized, since the fields have grown with little planning. Each of these structs has fields with variables pertaining to the problem.

geometry contains information about the geometry of the ocean, with the following fields:

- *bottomdepth*: constant bottom depth. This should be extended to a bottom profile later.
- *sourcedepth*: depth of the source (sonar).
- *targetdepth*: depth of the target.
- *range*: range of the target.

ray contains information about the ray tracing, with the following fields:

- *maxcollisionnumber*: the maximum number of collisions to allow in ray tracing
- *numrays*: the number of rays to be traced with Lybin when using numerical eigenray method 2.

env contains information about the environment, with the following fields:

- *constSOS*: speed of sound to use for computations with constant speed of sound.
- *linSOS*: 1x2 array containing sound speed at $z = 0$ and $z = z_b$ for use in the linear speed of sound computations.
- *meanSOS*: mean speed of sound.
- *coeffs*: 1x3 array containing EOF coefficients for the three most significant EOFs.
- *windSpeed*: takes values 1-9 to inform Lybin of severity of wind speed.

- *bottomType*: takes values 1-9 to inform Lybin of absorption properties of bottom.

signal contains information about the emitted ping, acoustic quantities and recording capabilities of the sonar, with the following fields:

- *SNR*: Signal-to-Noise Ratio of the signal, used for synthetization of signals.
- *linearSNR*: linear Signal-to-Noise Ratio, for convenience.
- *bandwidth*: bandwidth of the signal.
- *verticalBeamwidth*: vertical beamwidth of the sonar.
- *verticalBeamwidthRad*: vertical beamwidth of the sonar in radians, for convenience.
- *verticalAngleResolution*: vertical angle resolution, used in synthetization and creation of windows for the objective function.
- *verticalAngleResolutionRad*: vertical angle resolution in radians, for convenience.
- *samplingFrequency*: frequency at which the signal is sampled.
- *samplingPeriod*: time between each sample.
- *stoptime*: probably obsolete, used earlier in synthetization of a signal to specify when to stop the signal.
- *sigmat*: σ_t, each arrival's standard deviation in time.
- *sigmaphi*: σ_θ, each arrival's standard deviation in angle.
- *sigmatarrival*: σ_t/s, the standard deviation in time of arrival times.
- *sigmaphiarrival*: σ_θ/s, the standard deviation in time of arrival angles.
- *FM_frequency*: frequency of an FM signal.
- *FM_pulse_length*: pulse length of an FM signal.
- *SL*: Source Level of the signal.
- *TS*: Target Strength.
- *NL*: ambient Noise Level.
- *directivity*: directivity of the sonar.

comp contains information about the computational details, with the following fields: Note: To use a sound speed choice, all previous sound speed flags must be set to 0. For example, to use calculations with sound speed from EOFs, EOFspeed must be set to 1, while constspeed, linspeed and pwlinspeed must be set to 0. To use linear sound speed, linspeed must be set to 1, while only constspeed needs to be set to 0.

- *constspeed*: toggle calculation with constant sound speed on/off.

- *linspeed*: toggle calculation with linear sound speed on/off.
- *pwlinspeed*: toggle calculation with piecewise linear sound speed on/off.
- *EOFspeed*: toggle calculation with sound speed from EOFs on/off.
- *useLybin*: toggle use of Lybin in ray tracing on/off.
- *realdata*: flag to signify real data is being used, not synthesized.
- *quadint*: toggle use of quadratic interpolation on/off. If set to off, linear interpolation is used.
- *rangeCellLength*: set length of range cells used in Lybin calculations.
- *angleNoise*: toggle additive noise to arrival angles on/off.
- *arrivalTimeNoise*: toggle additive noise to arrival times on/off.
- *soundSpeedNoise*: toggle additive noise to sound speed profile on/off (for use with EOFs).
- *numrandcoeffs*: number of random EOF coefficients to use.
- *noiseThreshold*: noise threshold (in linear scale) for use in thresholding.
- *traveltimetrap*: toggle trapezoidal rule use in travel time calculations on/off. If set to off, the midpoint rule is used.
- *maxArrivalNumber*: maximum number of eigenrays to use in calculations of arrivals.
- *bottomhits*: max number of bottom reflections in ray tracing.
- *surfacehits*: max number of surface reflections in ray tracing.
- *visualEigenRay*: set to 0 for eigenray method 1, set to 1 for eigenray method 2.
- *angles*: angles to use in calculations.
- *rayleighBottom*: toggles Rayleigh bottom loss model on/off.
- *multCost*: set to 0 for full objective function, set to 1 for simplified objective function.

Signal struct

The signal struct is smaller, containing the following fields:

- *signal*: signal data.
- *tmin*: time in seconds at which the signal starts.
- *tmax*: time in seconds at which the signal ends.
- *badstart*: number of entries removed from start signal due to cell averaging.
- *badend*: number of entries removed from end of signal due to cell averaging.

A.2 Functions

Computation functions

- `arrivals.m` combines exit angles and one-way travel times obtained from ray tracing into arrival angles and two-way travel times. If Lybin is being used, it also returns an estimate of the transmission loss along each path.
- `cellAverage.m` employs the CFAR cell-averaging technique to estimate SNR levels in a signal.
- `clean.m` is an umbrella function for removing noisy data from a signal, calling cellAverage first, then removeUnderThreshold and returning the cleaned data..
- `fastcost.m` calculates the objective function based on inner products in a quick manner by not forming the full modelled signal, instead matching an arrival to the signal by windowing.
- `fastcostnew.m` is an attempt to address the window localization issue with fastcost, however, it has not yet been effective.
- `fastcostopt.m` does the same as fastcost but with an additional input d containing target depth and bottom depth data, to comply with MATLAB's convention for input to minimization functions.
- `findArrivalAnglesAndTimes.m` implements numerical eigenray method 2 . It runs through all rays generated by Lybin, finding the ones that lie closest to the desired target depth at the target range and interpolating to find arrival angles and times. It also returns an estimate of the transmission loss of each ray.
- `gethistory.m` reads through a ray's path data (depths and ranges) and assigns a history to it, based on which kinds of collisions occur, and the order in which they occurred.
- `gettransloss.m` reads through a ray's path data (depths and ranges) and estimates the transmission loss based on bottom collisions and signal attenuation.
- `initialGuess.m` loops through a specified range of bottom depths and target depths to determine an initial guess for fminsearch to work from.
- `interpolationtype.m` checks the histories of three rays to find out which kind of interpolation can be applied to find eigenrays – if all rays have the same history, quadratic interpolation can be used. If not, linear interpolation or no interpolation must be used.
- `linint.m` generically interpolates a value linearily.
- `makefilename.m` makes a filename based on parameters. optimization.m takes an uncleaned signal, parameters and an initial guess for bottom depth as input and tries to optimize with respect to bottom depth and target depth as outlined previously.

- `model.m` is used to make the modelled signal for the full objective function.
- `optimization.m` runs the optimization procedure.
- `planewavereflection_2layered.m` is made by KTH, and computes a reflection coefficient for a collision based on bottom properties and the angle of the incoming ray with the bottom.
- `quadint.m` generically interpolates a value quadratically.
- `raytrace.m` calculates exit angles and one-way travel times for eigenrays, based on the analytic model with constant speed of sound
- `raytracelybin.m` feeds parameters for Lybin, calls Lybin to do calculations and then findArrivalAnglesAndTimes to find angles and travel times for all eigenrays.
- `readSignal.m` reads a real signal and translates its data into the parameter and signal structs.
- `removeUnderThreshold.m` takes in a cell averaged signal and removes all entries below a certain threshold, typically 10-13 dB.
- `sndSpeedFromProfile.m` interpolates a sound speed from a given sound speed profile, for use in traveltime.
- `synthesize.m` synthesizes a signal by finding arrivals through arrivals, then modelling a signal by the method explained earlier.
- `testsetup.m` initializes a parameter struct with certain default values.
- `traveltime.m` calculates travel time from a ray's range and depth information.
- `traveltimetrap.m` slightly more accurately calculates travel time from a ray's range and depth information using the trapezoid rule.

Test functions

- `convtest.m` is used for testing convergence properties of Lybin's calculations to the analytic results.
- `funkytest.m` is used for testing analytic solutions to the linear sound speed case.
- `lybinspeedtest.m` is used for testing the speed of Lybin in calculating eigenrays.
- `pingtest.m` and `pingtesting.m` are used for testing the ability of the method to estimate target depths of real signals.
- `largeInitialGuess.m`, `largeoptimtest.m`, `largetest.m` and `runtest.m` are used for testing convergence properties of the method for synthesized signals.
- `test.m` is a general test function, to try out changes in code.

- `testdata.m` is used for testing the method on real data.
- `testoptim.m` is used for testing the optimization algorithm.

Plotting functions

- `lybinvsanalytic.m` plots arrivals extracted from Lybin's raytracing versus those obtained from analytic expressions.
- `rayanim.m` animates rays returned by Lybin.
- `signalplot.m` plots the signal as an image.
- `signalplot2.m` plots the signal as an image, while adjusting for removed entries due to cell averaging.
- `snrtestplot.m` plots the results of the SNR test.