



Pázmány Péter Catholic University

Faculty of Information Technology and Bionics

# OTDK Konferencia

## Creating a high-performance sonar simulation environment for river bed exploration

Hajdara Tamás

Computer Science Engineering MSc

2023

Supervisor: Balogh Gábor Dániel

Faculty mentor: Dr. Reguly István Zoltán

This research is supported by the ÚNKP-22-2 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.



MINISTRY OF CULTURE  
AND INNOVATION

# Abstract

In the field of waterborne disaster recovery, the submarine-mounted or boat-mounted sonars are used to map lake-, river-, and seabeds. For greater distances (depths) and in the presence of sediment or pollution, even extremely high illumination does not make the environment suitable for observation with a conventional camera. Therefore, sonar imaging is the only way to detect static or moving objects in the water.

The aim of the larger research is to implement a system that will make the search and rescue of victims in a disaster much easier and possible, even in difficult conditions. At the present, sonars can obtain depth images of the bottom, but the diver does not have a continuous and up-to-date visual information corresponding to its orientation during the dive. The planned system and procedure proposes a solution that makes it meaningful and allows the diver to the environment according to the diver's orientation. Together, the solution relies on sensors, deep network machine learning based image processing and AR visualization technology.

To develop this system, data is needed, which is exceedingly difficult and expensive to obtain using real-life measurements - yet large amounts are required for implement and train machine learning models in particular. For this purpose, the currently available data is incomplete and has a poor resolution, so it cannot be used properly. To create a learning database, the goal is to build a simulation environment to create point clouds that would augment the learning database with high accuracy, high quality, and high resolution data.

The Devito [1] library was used for my studies. They primarily designed this library to create wave propagation kernels for seismic inversion problems. In this term, we implemented a simulation environment for wave propagation in a water tank with microphones. Using the recorded sound-waves we measured the distance between the source and the object in front of the device.

# Contents

|                                                  |           |
|--------------------------------------------------|-----------|
| <b>Abstract</b>                                  | <b>ii</b> |
| <b>1 Introduction</b>                            | <b>2</b>  |
| <b>2 Background</b>                              | <b>4</b>  |
| 2.1 Sonar . . . . .                              | 4         |
| 2.1.1 Operation of active sonars . . . . .       | 5         |
| 2.1.2 Beamforming . . . . .                      | 5         |
| 2.1.3 Representation of sonar image . . . . .    | 6         |
| 2.2 Related works - Field II . . . . .           | 7         |
| <b>3 Tools and methodologies</b>                 | <b>8</b>  |
| 3.1 Devito . . . . .                             | 8         |
| 3.2 Mathematical model . . . . .                 | 9         |
| 3.2.1 Finite difference discretization . . . . . | 9         |
| 3.2.2 Acoustic wave equation . . . . .           | 10        |
| 3.2.3 Discretization . . . . .                   | 11        |
| 3.2.4 Sources and receivers . . . . .            | 12        |
| 3.2.5 Define the physical problem . . . . .      | 13        |
| <b>4 Software and System</b>                     | <b>16</b> |
| 4.1 Purpose and scope . . . . .                  | 16        |
| 4.2 Simulator structure . . . . .                | 17        |
| 4.2.1 Sonar object . . . . .                     | 17        |
| 4.2.2 Description of commands . . . . .          | 19        |
| 4.3 Beamforming . . . . .                        | 19        |
| 4.4 Peak finding . . . . .                       | 20        |
| <b>5 Results</b>                                 | <b>22</b> |
| 5.1 Source . . . . .                             | 22        |

|          |                                     |           |
|----------|-------------------------------------|-----------|
| 5.2      | Sanity testing . . . . .            | 23        |
| 5.3      | Artificial river-bed . . . . .      | 24        |
| 5.4      | Peak finding . . . . .              | 25        |
| 5.5      | Error of the measurements . . . . . | 27        |
| 5.6      | Runtime and memory usage . . . . .  | 28        |
| <b>6</b> | <b>Summary</b>                      | <b>30</b> |
|          | <b>Bibliography</b>                 | <b>33</b> |

# Chapter 1

## Introduction

My study is part of a project, which is motivated by the sinking of the boat Hableány in the Danube. It aims to implement a system that will facilitate the search and rescue of the victims in the event of a disaster, even under challenging circumstances. During the rescue, the professionals and the rescue workers encountered numerous difficulties. They were essentially in the dark in the river because of the visibility conditions and the strong river currents.

With the current methods, it is only possible to create the depth/relief record of the bottom, or riverbed's current state, from the water surface. These records were studied by the divers, who then needed to navigate from memory, essentially with no orientation information. With strong river flow, objects with unknown geometry had to be approached blindly, by using only their memory from the previously recorded images. This makes the rescue dangerous, slow, and difficult. There is a need for the diver to have continuous and fresh image information on the concerned area and objects, which corresponds to his current orientation.

With the rise of virtual reality (VR) and augmented reality (AR) applications, various display devices appeared. With these virtual objects can be displayed in 3D in such a way that the user perceives the virtual objects as being surrounded by a virtual environment. Navigating in heavily polluted water the visual is not or only partially can be matched with the virtual information, but it is necessary to spatially match the virtual 3D map with the object or topography in the real world. The virtual view should display the 3D map correctly in terms of distance and orientation to the diver. Also, if the diver changes position/orientation intentionally or unintentionally, this has to change the virtual view accordingly.

The latest advances in machine learning and artificial intelligence allow us to use a

partially known image or supplement information collected by a sensor with a different modality, based on a previous known image data. The information that is supplemented is qualitative information e.g. colour, depth, resolution, detail, gap filling. With these the incomplete data can be transformed into a photo-realistic image. Separately training a neural network for each of the subtasks involved here can result a system, where the generated view can be parameterized. Machine learning requires a database for training. To create this database, the goal of this work is to create a simulation environment that can generate high-accuracy, high-quality, and high-resolution data. This can be used to teach the neural network that can later be used in low visibility environments to augment/transform the sonar point-cloud based image to a conventional camera image, even in the case where there is no camera image of sufficient quality, or camera is not available at all.

Our work focuses on the development of a parallel implementation in Devito that can simulate the propagation of acoustic signals in different media using the acoustic wave equation and reconstruction of the map of the river bed based on the received signals. In our case we used the beamforming technique [2] for object search in the simulation. It is a simpler but faster computation, and with it static or moving object can be detected in the water. The code of the project is available on github [3].

## Chapter 2

# Background

### 2.1. Sonar

The sound waves travel farther in the water than do radar and light-waves. Therefore, sonar, short for *Sound Navigation and Ranging*, is helpful for exploring and mapping the lake-, river- and seabeds. Over the past decades, the price and size reductions of sonar systems enabled their usage in several underwater applications, such as navigation, localization, seabed mapping and obstacle detection. [4], [5]

The method of the sonars is based on sound propagation to detect, communicate with, and characterize submerged objects. It captures the reflected acoustic signals from the objects by the sonar device, where the acoustic data is then processed and analyzed.

There are two types of sonar systems, passive and active sonars. It is possible to detect noise from marine objects, such as ships and animals, with a passive sonar system. Passive sonar does not emit its own signal. Rather, it only detects sound waves coming towards to it.

Passive sonar can not measure the distance of an object. With multiple passive sonar devices, the sound source can be located with the triangulation technique.

Active sonar transducers transmit acoustic signal or pulse of sound into the water and then listen. If an object is in the signal's way, the sound bounces back from an object and returns an echo to the sonar transducer. By determining the time between the emission of the acoustic signal and its reception, the transducer can determine the range and the orientation of the object. The sonars that we are simulating in this study are classified as active type.

### 2.1.1 Operation of active sonars

Active sonars are devices that use acoustic energy to locate and survey objects in desired region. The acoustic signal are transmitted through the water by the transducer, which is a device that converts electrical signal into ultrasound energy. When the pulses collide with any object present in the propagation path, part of the acoustic energy is reflected back and captured by the sonar transducer, while another part is completely absorbed.

By knowing the time interval between the the emission of the pulse and its reception and the speed of sound in the medium, the pulse distance can be calculated between the object and the sonar head. The strength of the backscattered energy from the object determines the echo intensity.

The returning echos cause water pressure changes, during the measurement the sonar transducer detects these and converts them into electrical voltage. It is possible to produce a series of pulse distances and echo intensities, by analyzing the echo returns.

### 2.1.2 Beamforming

With a multiple number of sources, beamforming can be used for object search in the simulation. Beamforming is a signal processing technique. Combining the sources in such a way that signals at particular angles experience constructive interference while others experience destructive interference. The beamforming technique involves sending the pulse from each projector at slightly different times, so that every signal reaches the object at exactly the same time, producing the effect of a single strong pulse. With this technique it is possible to search for objects directionally and return the distance from the receivers.

In the record of the received signals each of the measured values is referred to as bin, while the group of bins obtained from the same incoming echo is denoted as beam. Therefore, whenever a sonar transducer oriented in a given direction emits a sound pulse, a beam is produced. By combining the records of the received signals, the set of beams forms an acoustic image of the reflected surfaces in front of the sonar device.

Acoustic absorption in water is frequency dependent, thus the operational frequency impacts on the coverage range and resolution of active sonars. Lower frequencies have higher wavelengths and reach longer distances than higher frequencies, although higher frequencies have smaller wavelengths and produce data with fine details of objects. The different frequencies with the corresponding wavelength and with the maximum range

that can be achieved with the sonar transducer is presented in Table 2.1

| Frequency | Wavelength | Maximum Range   |
|-----------|------------|-----------------|
| 100 Hz    | 15 m       | 1000 km or more |
| 1 kHz     | 1.5 m      | 100 km or more  |
| 10 kHz    | 15 cm      | 10 km           |
| 25 kHz    | 6 cm       | 3 km            |
| 50 kHz    | 3 cm       | 1 km            |
| 100 kHz   | 1.5 cm     | 600 m           |
| 500 kHz   | 3 mm       | 150 m           |
| 1 Mhz     | 1.5 mm     | 50 m            |

Table 2.1: Maximum range per frequency and corresponding wavelength. Source: [6]

### 2.1.3 Representation of sonar image

The raw sonar data is represented as a polar image, where the columns indicate individual beams, rows denote range samples and pixel values correspond to the reflected acoustic intensities. This polar representation of the data can be converted to Cartesian coordinates for better human interpretation. In the Cartesian system the image preserves the geometry of the object, but the loss of resolution increases with the distance. The sonar data displayed in polar and Cartesian coordinates is illustrated in Figure 2.1, 2.2

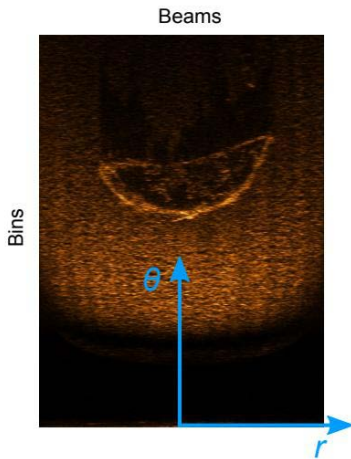


Figure 2.1: Sonar image

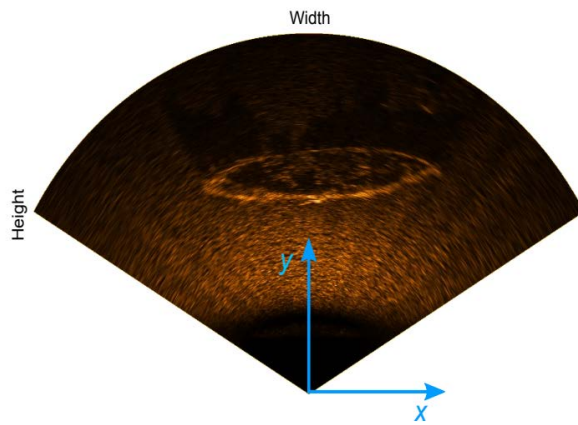


Figure 2.2: Cartesian image

Representation of sonar image. Source: [6]

## 2.2. Related works - Field II

Field II is a closed-source simulation software. "The Field program system uses the concept of spatial impulse responses as developed by Tupholme and Stepanishen in series of papers. [7]–[9]" The concept is to find the ultrasound field for both the pulsed and continuous wave case, it relies on linear systems theory. This is done through the spatial impulse response. This response gives the emitted ultrasound field at a specific point in a space as function of time, when the transducer is excited by a Dirac delta function. The field for any kind of excitation can then be found by just convolving the spatial impulse response with the excitation function. The impulse response will vary as a function of position relative to the transducer.

The received response in the echo can be found by convolving the transducer excitation function with the spatial impulse response of the emitting aperture, with the spatial impulse response of the receiving aperture. The result of the continuous wave case is found by Fourier transforming the spatial impulse response for the given frequency.

There is a major concern with this software, it is the time for one simulation. The program runs only on CPU, only on 1 core. Also the spatial impulse responses are calculated using sampling frequencies in the  $GHz$  range due to the sharp discontinuities of the responses. The Field program can handle the discontinuities, but it is using 100 Mhz sampling and approximation.

In our simulation we are using wave equations, which can be approximately solved by finite difference methods.

## Chapter 3

# Tools and methodologies

Stencil computation is the essence of many applications in the field of high-performance computing, such as computational fluid dynamics, image processing, and geophysical applications, involving the solution of the wave equations. For applications to solve partial differential equations in multidimensional grids stencil computations are often used [10]. Typically a 3D grid is iteratively traversed from a starting cell in stencil computation. Each grid cell is updated based on a set of coefficients and the values of its neighbouring cells. In the context of our wave equation solver, the stencil computation uses finite-difference time marching kernels that update the data set in a grid in one of the iterations, to use it in the next one.

### 3.1. Devito

Devito is a package to implement optimized stencil computation from high-level symbolic problem definitions. It employs automated code generation just-in-time compilation to execute optimized computational kernels on several platforms. It allows to build complex solvers with high-level code.

Devito is a domain specific language for formulating partial differential solvers in the form of finite difference operators. It compiles the formulation into C code optimized for the target platform. The main target application is producing wave-propagation kernels for use in imaging frameworks. One example for usage is create these kernels for seismic inversion problems [11]. Seismic inversion is the process of transforming seismic reflection data into a quantitative rock-property description of a reservoir.

Devito is available as a Python package and can be integrated with different packages such as SymPy and NumPy.

## 3.2. Mathematical model

Seismic modelling uses acoustic wave equation. These equations are in the time-space domain, and the finite-difference method is suitable to solve them.

### 3.2.1 Finite difference discretization

Many of the numerical challenges faced when computing ordinary differential equations (ODE) or partial differential equations (PDE) can be captured by a simple ODE:  $u'' + u = 0$ . The simplest model of the system has the following form:

$$u'' + \omega^2 u = 0, u(0) = I, u'(0) = 0, t \in (0, T] \quad (3.1)$$

Here,  $\omega$  and  $I$  are given constants. The exact solution is:

$$u(t) = I \cos(\omega t) \quad (3.2)$$

That is,  $u$  oscillates with a constant amplitude  $I$  and angular frequency  $\omega$ . The corresponding period of oscillation is  $P = 2\pi/\omega$  and measured in rad/s. The number of periods per second is  $f = \omega/(2\pi)$  and measured in Hz. Both are referred as frequency. Very often  $u(t)$  represents a position or a displacement of a particular point in the system. The derivative  $u'(t)$  has the interpretation of velocity, and  $u''(t)$  is the associated acceleration.

To formulate the finite difference method we need to discretize the domain, then fulfill the equation at the discrete time points, replace the derivatives by finite differences and at the end formulate a recursive algorithm.

#### Discretizing the domain

The domain is discretized by introducing a uniformly partitioned time dimension. The points in the mesh are  $t_n = n\Delta t, n = 0, 1, \dots, N_t$ , where  $\Delta t = T/N_t$  is the constant length of the time steps. The mesh function  $u^n$  approximates the exact solution at mesh points  $n = 0, 1, \dots, N_t$ .

#### Fulfilling the equation at discrete time points

The differential equation is satisfied at the mesh point, where solution is found.

$$u''(t_n) + \omega^2 u(t_n) = 0, n = 1, \dots, N_t \quad (3.3)$$

## Replacing derivatives by finite differences

The derivative  $u''(t_n)$  needs to be replaced by a finite difference approximation. Equation 3.4 gives a second order approximation to the second order derivative.

$$u''(t_n) \approx \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2} \quad (3.4)$$

After inserting into Equation 3.3 yields 3.5, with replacing also the first order derivative in the initial condition by a finite difference.

$$\begin{aligned} \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2} &= -\omega^2 u^n \\ \frac{u^1 - u^{-1}}{2\Delta t} &= 0 \end{aligned} \quad (3.5)$$

## Formulating a recursive algorithm

Equation 3.6 formulates the recursive algorithm, where we can assume that we have already computed  $u^{n-1}$  and  $u^n$ .

$$u^{n+1} = 2u^n - u^{n-1} - \Delta t^2 \omega^2 u^n \quad (3.6)$$

After these steps the equation can be solved.

1.  $u^0 = I$
2. compute  $u^1$ , where the discretization implies that  $u^{-1} = u^1$
3. for  $n = 1, 2, \dots, N_t - 1$  compute  $u^{n+1}$

### 3.2.2 Acoustic wave equation

A wide range of physical processes lead to wave motion, where signals are propagated through a medium in space and time. The shape of the signals may undergo changes as they travel through matter, but usually not so much that the signals cannot be recognized at some later point in space and time. A wave motion can be described by the equation  $u_{tt} = \nabla(c^2 \nabla u) + f$ , which can be approximately solved by finite difference methods.

The acoustic wave equation for the square slowness  $m$ , defined as  $m = \frac{1}{c^2}$ , where  $c$  is the speed of sound in the given physical media, and a source  $q$  is given by:

$$m \frac{\partial^2 u(x, t)}{\partial t^2} - \nabla^2 u(x, t) = q \text{ in } \Omega \quad (3.7)$$

To guarantee unicity of the solution, the initial condition are zero:

$$\begin{aligned} u(x, 0) &= 0 \\ \frac{\partial u(x, t)}{\partial t} \Big|_{t=0} &= 0 \end{aligned} \quad (3.8)$$

The boundary conditions are Dirichlet conditions, where  $\delta\Omega$  is the surface of the boundary of the model  $\Omega$ :

$$u(x, t)|_{\delta\Omega} = 0 \quad (3.9)$$

In the field the sound wave propagates in every direction to an infinite distance. However solving the wave equation in a discrete finite domain is not feasible. In order to compensate, absorbing boundary conditions [12] or perfectly matched layers [13] are required to mimic an infinite domain. These two methods allow us to approximate an infinite media by damping and absorbing the waves at the limit of the domain to avoid reflections.

We used absorbing damping mask, the core idea is to extend the physical domain and to add a mask in the extension that will absorb the incident waves. The wave equation can be rewritten as:

$$m \frac{\partial^2 u(x, t)}{\partial t^2} - \nabla^2 u(x, t) + \eta \frac{\partial u(x, t)}{\partial t} = q \text{ in } \Omega \quad (3.10)$$

where  $\eta$  is the damping mask which is equal to zero inside the physical domain and increasing inside the absorbing layer.

### 3.2.3 Discretization

Devito is a finite-difference domain-specific language (DSL) that solves the discretized wave-equation on a Cartesian grid. The wave equation is approximated by derivatives with finite differences. Both the spatial domain and time domains are discretized. Finite-difference discretization methods convert partial different equations into a system of linear equations that can be solved by linear algebra techniques. The finite-difference approximation is derived from Taylor expansions of the continuous field after removing the error term.

#### Time discretization

From the Taylor expansion, the second order time derivative is:

$$\frac{\partial^2 u(x, t)}{\partial t^2} = \frac{u(x, t + \Delta t) - 2u(x, t) + u(x, t - \Delta t)}{\Delta t^2} \quad (3.11)$$

where  $u$  is the discrete wavefield,  $\Delta t$  is the discrete time-step.

### Spatial discretization

The discrete Laplacian is defined as the sum of the second order spatial derivatives in the three dimensions:

$$\begin{aligned} \Delta u(x, y, z, t) = \sum_{j=1}^{j=\frac{k}{2}} \left[ \alpha_j \left( u(x + j\Delta x, y, z, t) + u(x - j\Delta x, y, z, t) + \right. \right. \\ \left. \left. u(x, y + j\Delta y, z, t) + u(x, y - j\Delta y, z, t) + \right. \right. \\ \left. \left. u(x, y, z + j\Delta z, t) + u(x, y, z - j\Delta z, t) \right) \right] + \\ 3\alpha_0 u(x, y, z, t). \end{aligned} \quad (3.12)$$

where  $k$  is the dimension of the space,  $\alpha_j$  is the appropriate finite difference weight for the approximation and the grid is divided spatially, where  $\Delta x, \Delta y, \Delta z$  are the space-steps of this division.

### Wave equation

With the space and time discretization defined, the wave-equation can be fully discretized with the combination of time and space discretizations. The second order in time and  $k^{th}$  order in space discrete stencil to update one grid point at position  $x, y, z$  at time  $t$ :

$$\begin{aligned} u(x, y, z, t + \Delta t) = & 2u(x, y, z, t) - u(x, y, z, t - \Delta t) + \\ & \frac{\Delta t^2}{m(x, y, z)} \left( \Delta u(x, y, z, t) + q(x, y, z, t) \right). \end{aligned} \quad (3.13)$$

#### 3.2.4 Sources and receivers

A **source** is positioned at a single or a few physical locations where artificial pressure is injected into the domain we want to model. The **receivers** are a set of microphones or hydrophones, that are used to measure the resulting wave and create a set of measurements called a shot record.

In order to create the numerical model we need to solve the wave equation and implement source and receiver interpolation to inject the source and record the wave at sparse point locations in the grid. The source time signature is defined by the user, in the following Section 3.2.5, for the examples it will be modelled using a Ricker wavelet [14] defined in Equation 3.14.

$$q(t) = (1 - 2\pi^2 f_0^2 (t - \frac{1}{f_0})^2) e^{-\pi^2 f_0^2 (t - \frac{1}{f_0})^2} \quad (3.14)$$

### 3.2.5 Define the physical problem

For defining the physical model we need the physical dimensions of interest and the velocity profile of this domain must be created. In my study there is a simple velocity model, that is essentially a water tank. The velocity of the sound in distilled water is  $1496.7m/s$  and in the glass is  $5640m/s$ . The velocity field of the model can be seen in Figure 3.1.

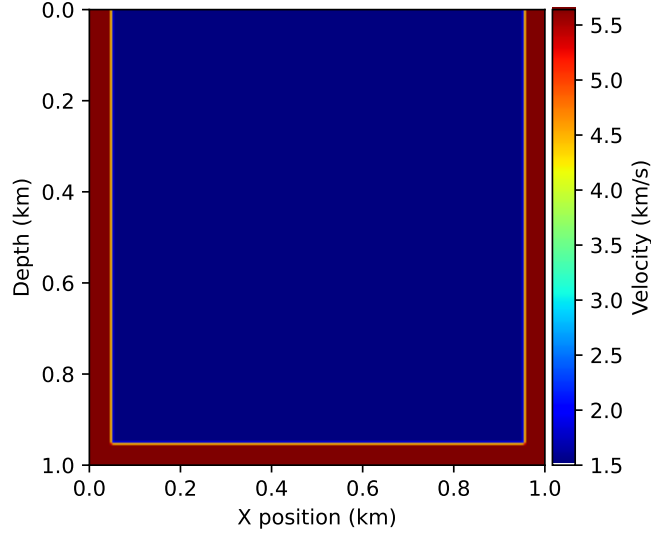


Figure 3.1: Velocity field of the model

To include the source time signature, defined in Equation 3.14, the time duration and the timestep size need to be defined. For a fixed time order this number decreases as the space order increases. It is given by the Courant–Friedrichs–Lewy condition (CFL conditions), and the Devito framework calculates it. Therefore with the provided critical timestep size, it is possible to fully discretize the model time axis as an array. The sources and receivers can be defined with Devito’s `SparseTimeFunction` type, which encapsulates sparse point data and allows to inject and interpolate values into and out of the computational grid. In the Figure 3.2 the sources and receivers are positioned at the top layer of the model.

In Devito the discrete wavefield can be defined as a `TimeFunction`. Through the wavefield object the derivatives and the discrete Laplacian can be easily accessed. After constructing the wavefield, the stencil (PDE rewritten as a time marching updating equation), the source injection and receiver interpolation, the Devito `Operator` can be created. The `Operator` will generate the C code runtime. After executing the operator for a number of timesteps, the receiver symbol’s data is populated due to the interpola-

tion expression. This data is visualized in Figure 3.3.

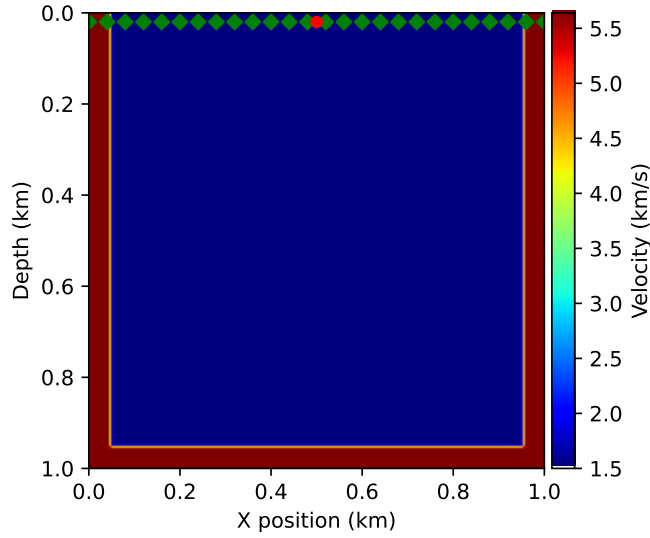


Figure 3.2: Velocity field of the model, with sources and receivers

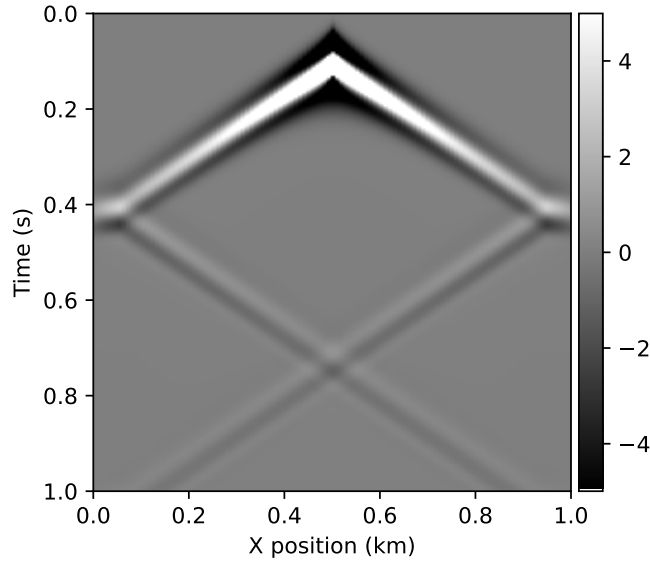


Figure 3.3: Shot record (receiver values over time)

Figure 3.3 show the shot record, where we can see the receiver values over time. The vertical axis shows the simulation time, and the horizontal axis represents the horizontal size of the model. The coloring shows the echo intensity of the received signal, the intensity is illustrated in the colorbar. At the start of the simulation the source emits an acoustic signal, and this high-intensity data is received by the receivers, and it is shown in the first part of the shot record. Later on the backscattered signals from different directions reaches the receivers. These data has a lower intensity than the emitted signal, because

the objects reflects back part of the acoustic energy, while another part is absorbed by them.

In the Figures 3.4, 3.5, 3.6, 3.7 we can see snapshots of the simulation, where the propagation of the acoustic wave is illustrated by.

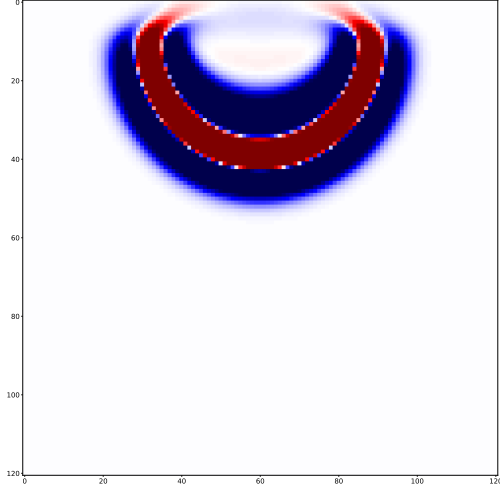


Figure 3.4: Transducer emits the acoustic signal

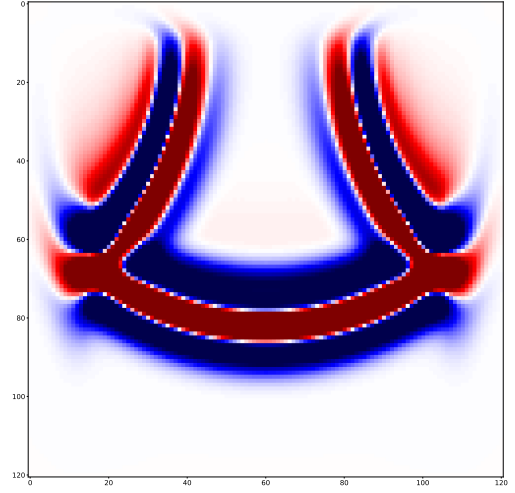


Figure 3.5: Signal reach the side and bounce back

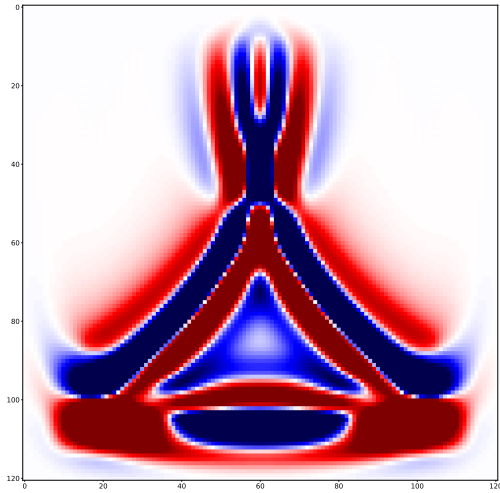


Figure 3.6: Signal reaches the bottom of the water tank

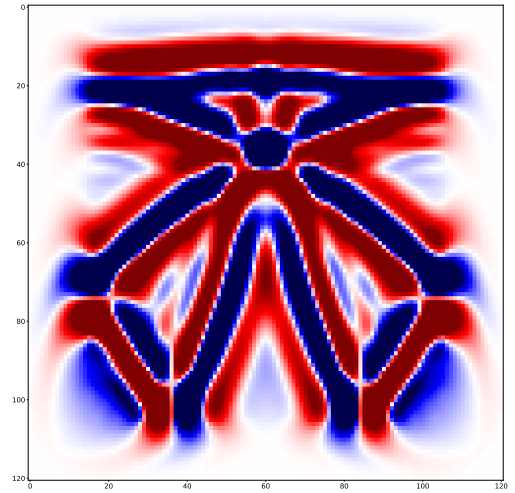


Figure 3.7: Echo of the signal reaches the receiver

## Chapter 4

# Software and System

### 4.1. Purpose and scope

The purpose of the study is to develop a simulation environment for exploring river beds using a sonar signal. The implementation is done with the Devito [1] framework, which gives a solution to simulate the propagation and detection of the reflected signals.

The study and the developed software fit into a research, where we would like to create a sonar-based augmented reality system. A challenging problem which arises in this domain is the scarcity of measurement data. To obtain these data with real life measurements is extraordinarily difficult and expensive. However a large amount of data is required to implement and train machine learning models. To put it another way, the currently available data is incomplete and has a poor resolution, so it can not be used accurately.

To overcome the emerging problem, the simulation environment gives us a solution. The scope of the simulation is to create artificial data to create a learning database for deep learning and machine learning. The analyzed and provided data includes the signal of the reflected wave, captured with the receivers, and a generated point cloud, which describes the surface of the river bed. Another key point to remember is that the artificially generated data has higher accuracy and higher resolution, then the data we already have.

Another advantage of the simulation, that it is possible to measure from different sonar positions with different angles. In the future work we would like to expand the domain, where the measurements can be done. This suggests that the system can process and provide 3-dimensional data. To achieve this result, the domain has to be slit into several planes and then measurements have to be made on these 2-dimensional

planes. At the end of the simulation, the resulting two dimensional point clouds have to be merged. The outcome will be a 3-dimensional point cloud of the river bed.

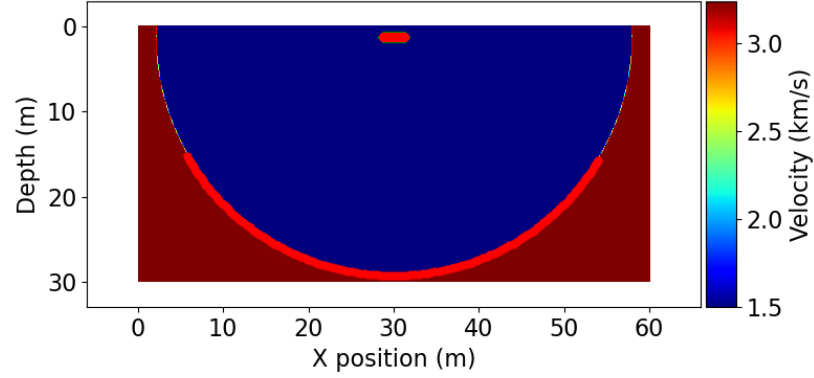


Figure 4.1: Generated point cloud

Figure 4.1 shows an example of the generated point cloud. The transducer emitted acoustic signal between  $30^\circ$  and  $150^\circ$  per degree. At each angle, the distance between the object from which the acoustic signal backscattered and the transducer was calculated. These resulting points were plotted with the velocity profile in the background. Both of the horizontal and vertical axes are extended, this ensures the limits and helps to show the measurement inaccuracy, if there is one.

## 4.2. Simulator structure

The developed software comes with a command-line interface (CLI). The CLI was implemented with the help of Typer<sup>1</sup>. Typer is a Python library for building CLI applications. It gives an easy to use interface for the final users.

### 4.2.1 Sonar object

The simulation has a main object called **Sonar**. Table 4.2.1 shows the command line arguments for customizing the parameters of the simulation. The **Sonar** object receives these parameters and creates the simulation model.

---

<sup>1</sup><https://typer.tiangolo.com/>

| Argument | Purpose                                                      |
|----------|--------------------------------------------------------------|
| size_x   | Size of the model's domain in horizontal direction. (m)      |
| size_y   | Size of the model's domain in vertical direction. (m)        |
| f0       | Center frequency of the signal. (kHz)                        |
| v_env    | The speed of the acoustic signal in the medium. (km/s)       |
| ns       | Number of the sources/receivers in the transducer.           |
| posx     | Relative position of the transducer in horizontal direction. |
| posy     | Relative position of the transducer in vertical direction.   |

Table 4.1: General command line arguments for customizing the simulation parameters

From the speed and the center frequency of the acoustic signal the spatial distance and the shape of the model can be calculated. Moreover the higher is the center frequency, the higher will be the resolution. Unfortunately, this approach results in problems related to the optimisation. Higher frequency sampling requires a smaller time step, leading to a larger number of elements in the grid. Therefore higher resolution increases the simulation time and the memory usage. In the future, it will be necessary to remedy this problem so that we can use the frequencies of real sonars in our calculations. As already mentioned the frequency of the transducer affects the wavelength of the emitted signal and the maximum distance it can reach. These values are shown in Table 2.1. Our goal in the research is to use a sonar with center frequency of  $500kHz$ .

The simulation time can be calculated from the position of the transducer. To ensure that the simulated acoustic signal can reach the bottom and the reflected signal reach the receivers in every scenario, the simulation time is overestimated.

The **Model** can be created from the given and calculated parameters. This is a Devito object, where the physical dimension of interest and the velocity profile of this physical domain is encapsulated. The simulation time is known, and the **Model** utility provides the critical timestep size, so time axis of the model can be fully discretized as an array. The **Sonar** also creates and stores the **SparseTimeFunction** object that encapsulates the sparse point data of sources and receivers. Similarly the discrete wavefield is defined as a **TimeFunction**, and the **Operator** is defined as well. The **Operator** will generate the C code. The generated code is independent of the angle at which the sources transmit the acoustic signal therefore it will not cause any problems.

### 4.2.2 Description of commands

| Command | Purpose                                                                 |
|---------|-------------------------------------------------------------------------|
| plot    | Display different type of plots.                                        |
| beams   | Run the simulation in different angles.                                 |
| analyse | Takes the angles and recordings, process them and create a plot.        |
| snaps   | Create an animation of the wave propagation and the signal on receivers |

Table 4.2: List of current commands

Table 4.2.2 shows the commands that are implemented in the software. All of the commands initializes the **Sonar** base object, and take the arguments listed at Table 4.2.1 as inputs. Also to modify the environment there are 3 more arguments called **bottom**, **obstacle** and **r**. With the **bottom** parameter, the shape of the simulated river-bed can be easily controlled. The **obstacle** parameter also belongs to this settings, it has boolean type, and with enabling it the river-bed will be populated with some obstacles. These obstacles ensures that the some part of the transmitted wave will propagate in the direction of the receivers. For the debugging cases and sanity testing we are using a circle as the environment, the **r** parameter set the radius of it.

With **plot** command after the initialization plots can be created and showed, such as the velocity profile of the model or the shot record. The **beams** has three more arguments, with those the start and last angle of the simulation and the step between these can be modified. The simulation is running with these settings, and saves the angles and the resulting distances in an **.npy** file, which is the standard binary file format for persisting a NumPy array on disk. It stores all the necessary information to reconstruct the array correctly. This output file serves as an input file for the **analyse** procedure. It parses the input data, calculates the coordinates from the distance between the sonar and the found object and creates an output file, where the figure is saved. Furthermore the **snaps** command creates an animation with the wave propagation and the received signal through the simulation time.

## 4.3. Beamforming

Figure 4.2, 4.3. 4.4 show snapshots from the wave propagation. With multiple number of sources, in the particular angle, where the transducer is headed, there is constructive interference between the signals, by contrast in other angles, there is destructive interfer-

ence. Therefore, by using the beamforming technique, the backscattered signal will have the largest intensity from the direction, where the sonar is facing.

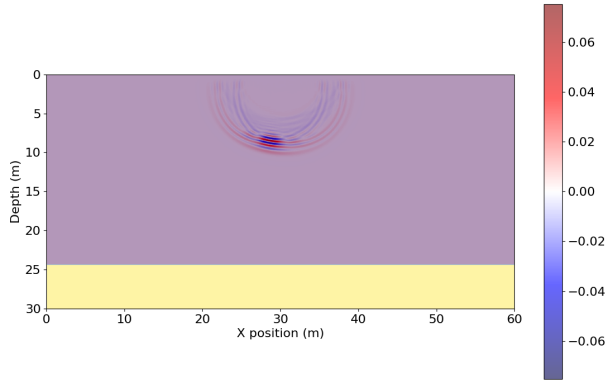


Figure 4.2: Sonar transducer emits the acoustic signal

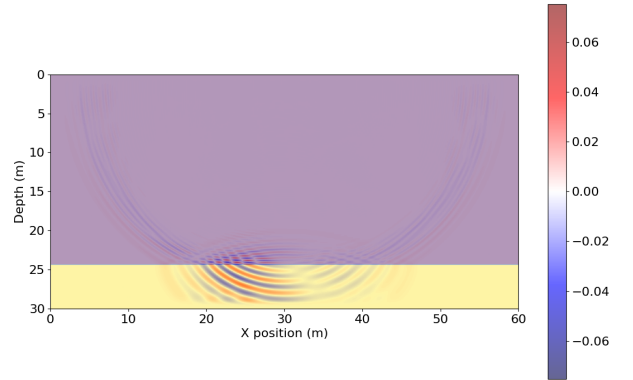


Figure 4.3: The acoustic signal hits the bottom

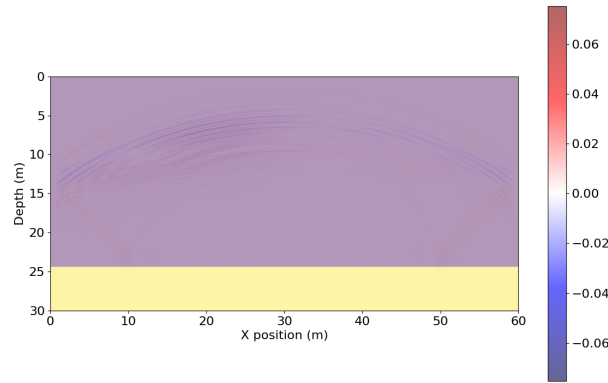


Figure 4.4: Part of the signal propagates back

## 4.4. Peak finding

Peak finding is one of the hardest problem in the simulation, that needs to be solved. For calculating the distance between the object and the sonar, we need to observe and draw conclusions from the received signal. By finding the first peak in the data of the receivers, and given the time of it's appearance the distance between the object and the sonar can be calculated. In order to find this peak, we are using the signal processing module from the SciPy package. More specifically the `find_peaks` and `peak_prominences`. The `find_peaks` finds all local maxima, by comparison of neighboring values and `peak_prominences` calculates the prominence of each peak in signal. The prominence of a peak measures how much a peak stands out from the surrounding

baseline of the signal and is defined as the vertical distance between the peak and its lowest contour line. By using Equation 4.1 and choosing the first element from  $p_{result}$  we get the first peak. Another important aspect is that the sources and receivers are placed in the same coordinates, thus the emitted signal appears on the receivers data. For this reason we are using a cut off method, where we are filtering the signal and do not take into account the part, where the emitted signal occurs.

This method is the simplest approach to approximate the occurrence of the first peak. In simple examples the results are satisfying. In the other hand having a complex domain, it will result more and more inaccurate data.

$$p := \text{array of the found peaks prominences}, p_{result} = p - \bar{p} > \sigma(p) \quad (4.1)$$

Figure 4.5 shows the result of the peak finding algorithm. The signal is calculated as the average of the amplitude of the received data over time.

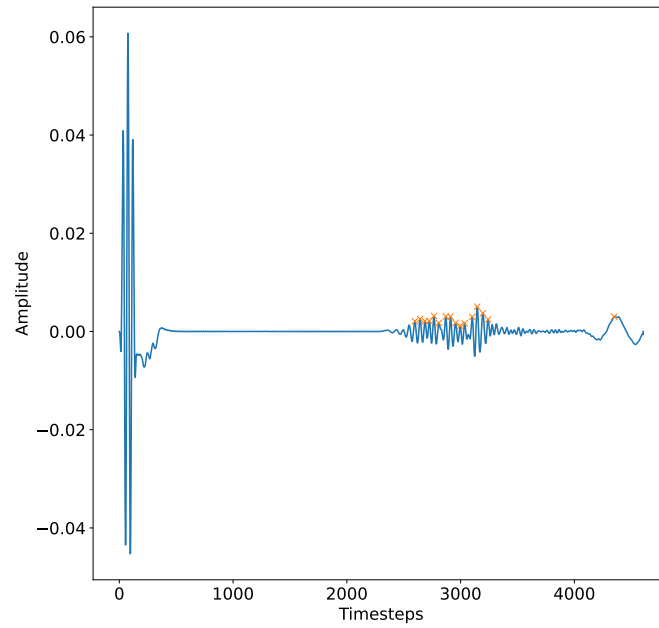


Figure 4.5: Result of the peak finding algorithm

# Chapter 5

## Results

### 5.1. Source

As I mentioned in Section 3.2.4 in the Devito framework the source time signature is modelled using Ricker wavelet. The Ricker wavelet is preferable in seismic imaging problems, on the other hand the sonars and for creating beamforming the sources with sinus wave is a much better choice. The Sine wavelet is defined in Equation 5.1.

$$q(t) = \sin(\pi^2 f_0^2 (t - \frac{1}{f_0})) + \sin(3 * (\pi^2 f_0^2 (t - \frac{1}{f_0}) + \pi)/4) \quad (5.1)$$

Figure 5.1, 5.2 shows the Ricker wavelet, which is emitted by the sources, and in the same timesteps it is visible that the correct signal appears on the receiver data.

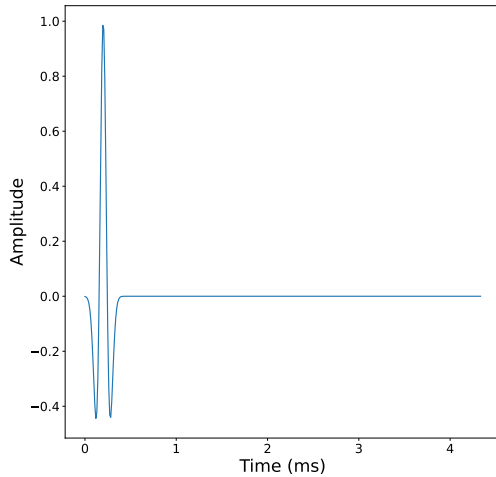


Figure 5.1: Ricker wavelet, plotted from source data

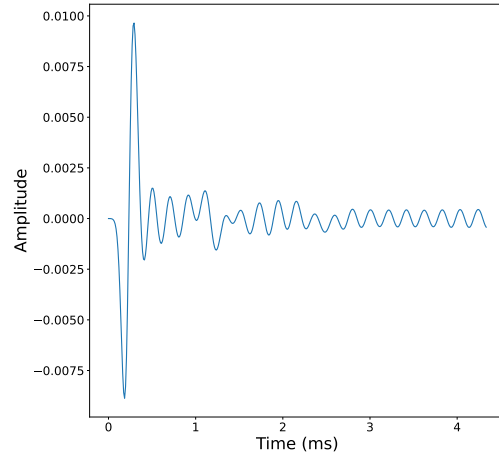


Figure 5.2: Ricker wavelet appearing on receivers

Figure 5.3, 5.4 shows the Sine wavelet. The statement also holds true here, that the correct signal appears on the receiver data. This signal is a better directed wavelet, the constructive interference between the signals are stronger thus it can be used for object search with the beamforming technique.

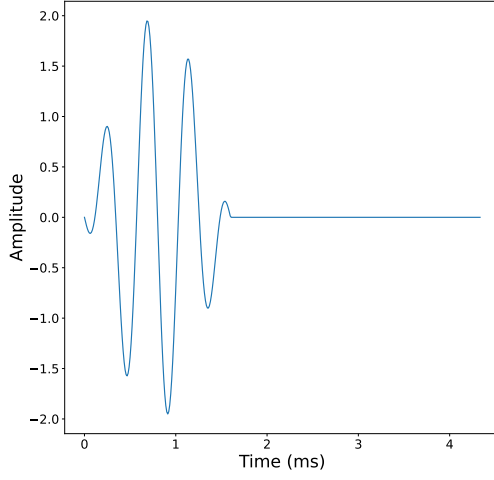


Figure 5.3: Sine wavelet, plotted from source data

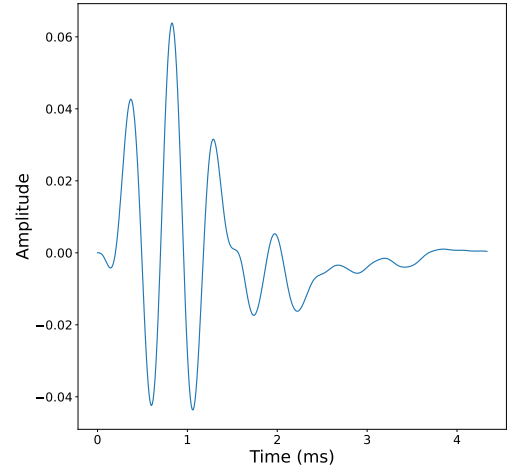


Figure 5.4: Sine wavelet appearing on receivers

## 5.2. Sanity testing

For testing the simulation and the correctness of the beamforming, we created a simulation environment, where the bottom of the model is a circle. The transducer's position is close to the origin of the circle, this ensures that the wave propagates back into the direction of the receivers.

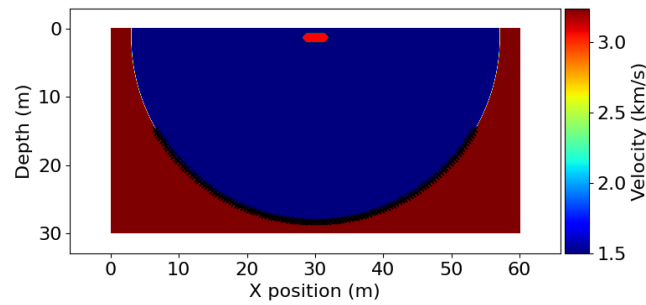


Figure 5.5: Simulation environment with circle bottom for sanity checks

Figure 5.5 shows the model's velocity and the retrieved values from the simulation. It can be stated that beamforming technique works well, although there is some error of measurement.

### 5.3. Artificial river-bed

To simulate the real world examples and environments, the bottom of the simulation's domain can be populated with obstacles. These obstacles ensures that the wave propagates into the direction of the receivers.

In the following simulation, the sonar is facing to  $65^\circ$ . Figure 5.6, 5.7, 5.8 shows a simulation, where the bottom is flat and there are no obstacles. In this case the backscattered signal never reaches the receiver, only a really low intensity part of it, which can be discarded. In contrast Figure 5.9, 5.10, 5.11 shows a simulation, where there are some obstacles on the bottom. In this case a part of the backscattered signal reaches the receivers, and has a higher amplitude value, so the bottom of the simulation can be mapped.

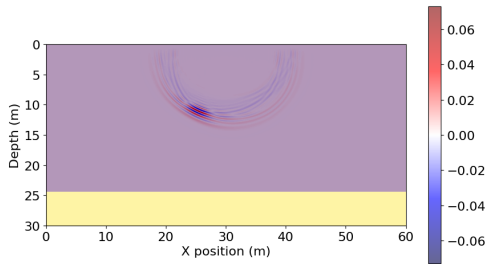


Figure 5.6: Sonar transducer emits the acoustic signal in the environment without obstacles

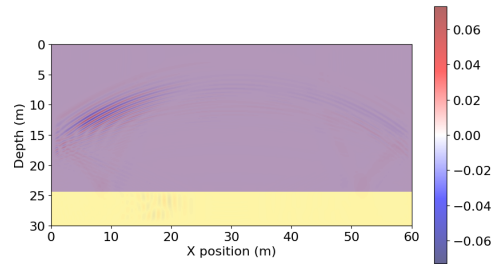


Figure 5.7: The acoustic signal hits the bottom in the environment without obstacles

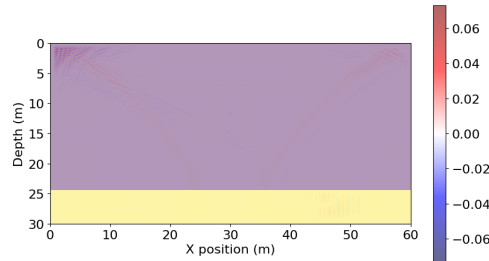


Figure 5.8: Part of the signal propagates back in the environment without obstacles

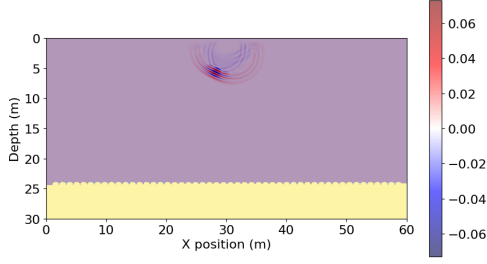


Figure 5.9: Sonar transducer emits the acoustic signal in the environment with obstacles

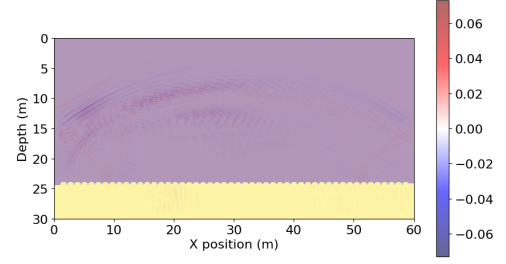


Figure 5.10: The acoustic signal hits the bottom in the environment with obstacles

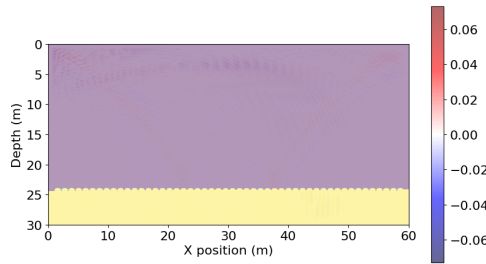


Figure 5.11: Part of the signal propagates back in the environment with obstacles

## 5.4. Peak finding

As I mentioned in Section 3.2.5 the physical dimension and the velocity profile of the model must be defined as well as the sources and receivers. In this project the dimension are the following:

- Dimensions of the water tank:  $60m * 30m$
- Velocity profile:
  - The medium is water, where the speed of sound is  $1496.7m/s$
  - Border layers are glass, where the speed of sound is  $5640m/s$

The sources' and receivers' coordinates are the same, they are placed at the top of the water tank. The distances from each other, and also the angle where they are facing is parameterized. Namely, there are 128 sources and receivers, and the distance between them is  $0.02m$ . We have been dealing with the problem of which method of the peak finding is the most accurate, and has the lowest measurement error. For this, we have been done the simulation between  $5KHz$  and  $50kHz$ .

There are two main approaches to calculate the first peak, that occurs in the received signal. One of the method is described in Section 4.4, where the peaks are choosed, if the prominence of it is bigger than a threshold value. This threshold value comes from the average and standard deviation of the found peaks. The other method is to take a signal, which is backscattered in ideal circumstances, then correlate this with the received signal and take the value, where the correlation has a maximum.

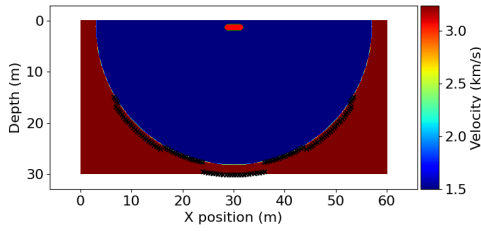


Figure 5.12: Environment with circle bottom, source with  $5kHz$  peak frequency, peak finding with prominence

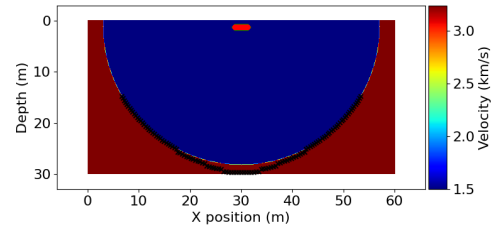


Figure 5.13: Environment with circle bottom, source with  $5kHz$  peak frequency, peak finding with correlation

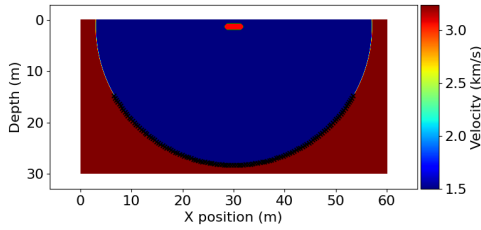


Figure 5.14: Environment with circle bottom, source with  $50kHz$  peak frequency, peak finding with prominence

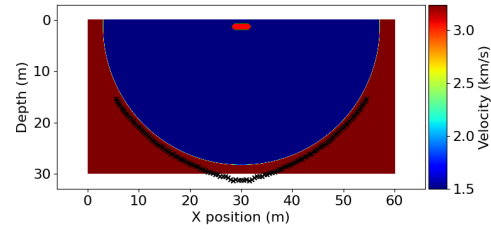


Figure 5.15: Environment with circle bottom, source with  $50kHz$  peak frequency, peak finding with correlation

Figure 5.12, 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19 shows the result of the simulation. In the first four images we used a simpler domain, the bottom of the water-tank is a circle as mentioned in Section 5.2. By contrast in the second part we used a more complex

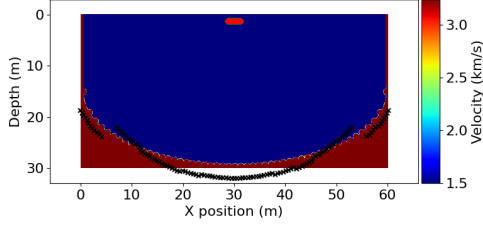


Figure 5.16: Environment with ellipse bottom, source with  $5kHz$  peak frequency, peak finding with prominence

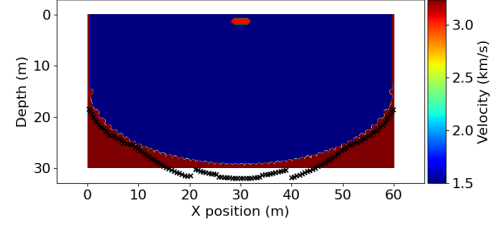


Figure 5.17: Environment with ellipse bottom, source with  $5kHz$  peak frequency, peak finding with correlation

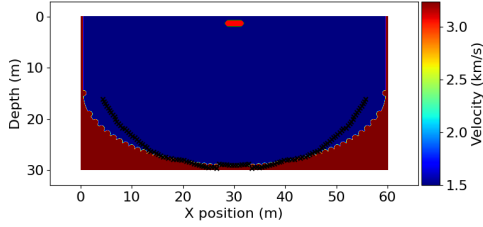


Figure 5.18: Environment with ellipse bottom, source with  $50kHz$  peak frequency, peak finding with prominence

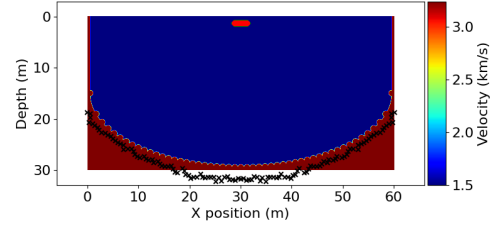


Figure 5.19: Environment with ellipse bottom, source with  $50kHz$  peak frequency, peak finding with correlation

domain, with an artificial river bed.

## 5.5. Error of the measurements

To calculate the measurement error we used the Mean Squared Error ( $MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$ ), where  $\hat{Y}_i$  is the set of predicted values and  $Y_i$  is the set of distances being predicted. Table 5.1 shows these measurement errors. There is a significant difference between the complex domain and the simpler one. There are two possible reasons for them. Firstly we know the exact distance between the sources and the bottom, with the simpler domain, and in the other case solving the equation of ellipse is not feasible, because of the resolution. For this reason the real distances are a good approximation. The other reason of the higher errors is the complex bottom of the domain, the algorithms to find

the object in every direction are not the smoothest. This is still an active research, we want to include more literature methods and also refine and make some clarification for the currently used methods.

| Frequency | Bottom  | Peak finding method | Error   |
|-----------|---------|---------------------|---------|
| 5 kHz     | ellipse | prominence          | 11.5966 |
| 5 kHz     | ellipse | correlation         | 11.3789 |
| 50 kHz    | ellipse | prominence          | 11.5355 |
| 50 kHz    | ellipse | correlation         | 11.6894 |
| 5 kHz     | circle  | prominence          | 0.9865  |
| 5 kHz     | circle  | correlation         | 0.4051  |
| 50 kHz    | circle  | prominence          | 0.0006  |
| 50 kHz    | circle  | correlation         | 2.735   |

Table 5.1: Measurement errors of the simulations

## 5.6. Runtime and memory usage

| Frequency | x     | y     | Mesh size     | Discrete Timestep | Memory usage |
|-----------|-------|-------|---------------|-------------------|--------------|
| 5 kHz     | 600   | 300   | 180.000       | 0.02 s            | 0.0384 GB    |
| 50 kHz    | 6000  | 3000  | 18.000.000    | 0.002 s           | 3.84 GB      |
| 500 kHz   | 60000 | 30000 | 1.800.000.000 | 0.0002 s          | 384 GB       |

Table 5.2: Mesh size, discrete time step and memory usage of different frequencies

| Frequency | CPU        | V100      | A100      |
|-----------|------------|-----------|-----------|
| 5 kHz     | 2.6029 s   | 8.5523 s  | 2.5058 s  |
| 50 kHz    | 165.5181 s | 63.0359 s | 28.8858 s |

Table 5.3: Runtime of measurements on CPU and GPUs, 1 beam

| Frequency | CPU          | V100        | A100        |
|-----------|--------------|-------------|-------------|
| 5 kHz     | 54.0213 s    | 48.1218 s   | 75.2012 s   |
| 50 kHz    | 19250.8684 s | 6967.5089 s | 3030.1550 s |

Table 5.4: Runtime of measurements on CPU and GPUs, 120 beams

Table 5.2 demonstrates the number of points, mesh sizes, the discrete time step and

memory usage of the simulation with different source frequencies. It is visible that the mesh size and the memory usage is increasing exponentially with the frequency. The discrete time step is also increasing with the frequency, but not as fast as the mesh size. We need to reach the 500 kHz frequency to simulate the real world environment, but it consumes a lot of memory, that is why optimization is needed. Table 5.3, and Table 5.4 indicates the runtimes of different simulations on CPU and GPUs with 5kHz and 50kHz sources. In the case of the first table the simulation was run in 1 direction, and in the second one, the simulation was run in 120 directions. These results refer to the whole measurement, including the construction of the model, not only running the simulation in the corresponding directions. The Intel®Xeon®Gold 6226R @ 2.90 GHz CPU, NVIDIA®V100 GPU and NVIDIA®A100 GPU was used for the measurements. With the measurements on the GPU significant improvements can be achieved on the bigger problems. Using the GPU, there is a bigger initial overhead of allocations and data movement than using the CPU, this can be seen where the problem is not big enough to saturate the GPU. In our case running the simulation in 1 angle, the CPU achieved better results.

Table 5.5 shows the approximate runtimes of the measurement only in 1 direction, without the overhead of constructing the sonar model.

| Frequency | CPU           | V100          | A100           |
|-----------|---------------|---------------|----------------|
| 5 kHz     | approx. 0.42s | approx. 0.37s | approx. 0.57 s |
| 50 kHz    | approx. 155 s | approx. 57 s  | approx. 25 s   |

Table 5.5: Runtime of measurements in 1 direction on CPU and GPUs

# Chapter 6

## Summary

During my project I got acquainted with the continuous wave equation and the discretization of it. For the simulation I used the Devito Python package. It was a must to get to know this package and its key features. It gives a functional language to express finite difference operators, a straightforward mechanism to adjust discretization and constructs to express sparse operators (e.g interpolation). With this package wave propagation kernels can be implemented from high-level symbolic problem definitions.

With the help of the simulation the backscattered acoustic wave appears on the receivers. Later on modifications and observations can be done on the received signal. Using this signal as an input, point clouds of the mapped river bed can be created. These recordings of the signal and the point clouds will serve as a training database to implement and train machine learning models.

The simulation comes with a command line interface, which makes the usage of the developed software more user friendly. We checked the correctness of the results that we have, but in the future we will have more work related to the object detection and distance calculation. There are several methods developed for finding the first peak in the received signal, but the measurement error occurs in several cases.

In the future work we would like to introduce 3-dimensional domains as well. One problem with the simulation is foreseeable. To have higher resolution data, the frequency of the sonar needs to be higher. Higher frequency sampling requires a smaller time step, leading to a larger number of elements in the grid. Therefore higher resolution increases the simulation time and the memory usage. A solution would be, if we could split the simulation domain, and use the relevant part of the grid for the simulation, namely use the part, where the sonar is facing. With this the calculation needs to be done on less data point. It will reduce the memory usage and the simulation time as well. Also,

in this implementation for measuring in multiple directions, multiple measurements are required. This can be solved by using a multi-beam source. We can do the implementation in combining two different approach. One of them is using a latency profile, which defines the direction of the beams. The other one is using two sine waves with different frequencies and a band-pass filter on the received signal, which produces the separate results. This method can lead to better measurement results.

# Bibliography

- [1] M. Louboutin, M. Lange, F. Luporini, *et al.*, “Devito (v3.1.0): An embedded domain-specific language for finite differences and geophysical exploration,” *Geoscientific Model Development*, vol. 12, no. 3, pp. 1165–1187, 2019. DOI: 10.5194/gmd-12-1165-2019. [Online]. Available: <https://www.geosci-model-dev.net/12/1165/2019/>.
- [2] B. Van Veen and K. Buckley, “Beamforming: A versatile approach to spatial filtering,” *IEEE ASSP Magazine*, vol. 5, no. 2, pp. 4–24, 1988. DOI: 10.1109/53.665.
- [3] T. Hajdara, G. D. Balogh, and I. Z. Reguly, *sonar-FWI*, version 1.0.0. [Online]. Available: <https://github.com/bgd54/sonar-FWI>.
- [4] R. Cerqueira, T. Trocoli, G. Neves, S. Joyeux, J. Albiez, and L. Oliveira, “A novel gpu-based sonar simulator for real-time applications,” *Computers Graphics*, vol. 68, no. Supplement C, pp. 66–76, 2017, ISSN: 0097-8493. DOI: <http://dx.doi.org/10.1016/j.cag.2017.08.008>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0097849317301371>.
- [5] G. Neves, M. Ruiz, J. Fontinele, and L. Oliveira, “Rotated object detection with forward-looking sonar in underwater applications,” *Expert Systems with Applications*, vol. 140, p. 112870, 2020, ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2019.112870>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417419305809>.
- [6] R. Cerqueira, “A multi-device sonar simulator for real-time underwater applications,” Ph.D. dissertation, Federal University of Bahia, 2019.
- [7] G. E. Tupholme, “Generation of acoustic pulses by baffled plane pistons,” *Mathematika*, vol. 16, no. 2, pp. 209–224, 1969. DOI: 10.1112/S0025579300008184.
- [8] P. R. Stepanishen, “The time-dependent force and radiation impedance on a piston in a rigid infinite planar baffle,” *The Journal of the Acoustical Society of America*, vol. 49, no. 3B, pp. 841–849, 1971. DOI: 10.1121/1.1912424. eprint: <https://doi.org/10.1121/1.1912424>.

//doi.org/10.1121/1.1912424. [Online]. Available: <https://doi.org/10.1121/1.1912424>.

- [9] P. R. Stepanishen, “Transient radiation from pistons in an infinite planar baffle,” *The Journal of the Acoustical Society of America*, vol. 49, no. 5B, pp. 1629–1638, 1971. DOI: 10.1121/1.1912541. eprint: <https://doi.org/10.1121/1.1912541>. [Online]. Available: <https://doi.org/10.1121/1.1912541>.
- [10] P. P. L. Thiago Carrijo Nasciutti Jairo Panetta, “Evaluating optimizations that reduce global memory accesses of stencil computations in gpgpus,” *Concurrency and Computation: Practice and Experience*, 2019. DOI: <https://doi.org/10.1002/cpe.4929>.
- [11] M. Louboutin, F. Luporini, P. Witte, *et al.*, “Scaling through abstractions—high-performance vectorial wave simulations for seismic inversion with devito,” *arXiv preprint arXiv:2004.10519*, 2020.
- [12] B. Engquist and A. Majda, “Absorbing boundary conditions for numerical simulation of waves,” *Proceedings of the National Academy of Sciences*, vol. 74, no. 5, pp. 1765–1766, 1977. DOI: 10.1073/pnas.74.5.1765. eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.74.5.1765>. [Online]. Available: <https://www.pnas.org/doi/abs/10.1073/pnas.74.5.1765>.
- [13] J.-P. Berenger, “A perfectly matched layer for the absorption of electromagnetic waves,” *Journal of Computational Physics*, vol. 114, no. 2, pp. 185–200, 1994, ISSN: 0021-9991. DOI: <https://doi.org/10.1006/jcph.1994.1159>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999184711594>.
- [14] N. Ricker, “Wavelet functions and their polynomials,” *Geophysics*, vol. 9, no. 3, pp. 314–323, 1944.